

## **Chương 7: Kết nối tới một Cơ sở dữ liệu**

### **SỬ DỤNG ĐỐI TƯỢNG *SqlConnection* ĐỂ KẾT NỐI VỚI CƠ SỞ DỮ LIỆU *SOLSEVER***

Tải System.Data.SqlClient namespace vào project

Bộ khởi tạo: SqlConnection()

1: SqlConnection()

2: SqlConnection(string *connectionString*)

Khởi tạo một đối tượng mới SqlConnection bằng phát biểu sau:

```
SqlConnection mySqlConnection = new SqlConnection();  
mySqlConnection.ConnectionString =  
"server=localhost;database=Northwind;uid=sa;pwd=sa";
```

**server** chỉ định tên máy tính có trình SqlServer đang chạy.

**database** chỉ định tên cơ sở dữ liệu

**uid** tên tài khoản

**pwd** mã dẫn nhập

chú ý : chỉ thiết lập connectionString khi đối tượng kết nối của bạn đã đóng

#### **KẾT NỐI TRỰC TIẾP:**

```
string connectionString =  
"server=localhost;database=Northwind;uid=sa;pwd=sa";  
SqlConnection mySqlConnection = new SqlConnection(connectionString);
```

#### **KẾT NỐI ĐƠN GIẢN:**

```
SqlConnection mySqlConnection =  
new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa");
```

#### **THỜI GIAN CHỜ KẾT NỐI (*connection timeout*)**

```
string connectionString =  
"server=localhost;database=Northwind;uid=sa;pwd=sa;" +  
"connection timeout=10";
```

CHÚ Ý: mặc định connection timeout = 15 giây

connection timeout = 0 chờ đợi vô thời hạn (nên tránh thiết lập này)

#### **KẾT NỐI SỬ DỤNG QUYỀN ĐĂNG NHẬP HỆ THỐNG:**

```
string connectionString =  
"server=localhost;database=Northwind;integrated security=SSPI";
```

#### **MỞ VÀ ĐÓNG MỘT KẾT NỐI:**

```
mySqlConnection.Open();
```

```
mySqlConnection.Close();
```

### **THÍ DU VỀ KẾT NỐI:**

Listing 7.1: MYSQLCONNECTION.CS

```
/*  
 MySqlConnection.cs illustrates how to use a  
 SqlConnection object to connect to a SQL Server database  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class MySqlConnection  
{  
    public static void Main()  
    {  
        // formulate a string containing the details of the  
        // database connection  
        string connectionString =  
            "server=localhost;database=Northwind;uid=sa;pwd=sa";  
  
        // create a SqlConnection object to connect to the  
        // database, passing the connection string to the constructor  
        SqlConnection mySqlConnection =  
            new SqlConnection(connectionString);  
  
        // open the database connection using the  
        // Open() method of the SqlConnection object  
        mySqlConnection.Open();  
  
        // display the properties of the SqlConnection object  
        Console.WriteLine("mySqlConnection.ConnectionString = "+  
            mySqlConnection.ConnectionString);  
        Console.WriteLine("mySqlConnection.ConnectionTimeout = "+  
            mySqlConnection.ConnectionTimeout);  
        Console.WriteLine("mySqlConnection.Database = "+  
            mySqlConnection.Database);  
        Console.WriteLine("mySqlConnection.DataSource = "+  
            mySqlConnection.DataSource);  
        Console.WriteLine("mySqlConnection.PacketSize = "+  
            mySqlConnection.PacketSize);  
        Console.WriteLine("mySqlConnection.ServerVersion = "+  
            mySqlConnection.ServerVersion);  
        Console.WriteLine("mySqlConnection.State = "+  
            mySqlConnection.State);  
        Console.WriteLine("mySqlConnection.WorkstationId = "+  
            mySqlConnection.WorkstationId);  
  
        // close the database connection using the Close() method  
        // of the SqlConnection object  
        mySqlConnection.Close();  
    }  
}
```

```
}
```

The output from this program is as follows:

```
mySqlConnection.ConnectionString = server=localhost;database=Northwind;uid=sa;  
mySqlConnection.ConnectionTimeout = 15  
mySqlConnection.Database = Northwind  
mySqlConnection.DataSource = localhost  
mySqlConnection.PacketSize = 8192  
mySqlConnection.ServerVersion = 08.00.0194  
mySqlConnection.State = Open  
mySqlConnection.WorkstationId = JMPRICE-DT1
```

## **BỘ NHÓM NHỮNG KẾT NỐI(CONNECTION POOLING)**

Sự mở và đóng kết nối tiêu phí nhiều thời gian . do đó ADO .NET tự động lưu giữ những kết nối trong Một bể chứa , nó cung cấp một sự cải tiến lớn về thực thi kết nối . bạn không cần chờ đợi một kết nối Trống đến cơ sở dữ liệu trong khi một kết nối đã có hiệu lực. Khi bạn đóng một kết nối , nó chưa thực sự đã đóng , kết nối của bạn được đánh dấu là chưa dùng đến và được dự trữ trong một bể chứa, sẵn sàng để sử dụng trở lại

Sau đó nếu bạn cung cấp chi tiết kết nối tương tự trong connection string ( database name, uid, password) thì kết nối dự trữ trong pool sẽ được khôi phục và bạn tiếp tục sử dụng nó để truy cập dữ liệu

Khi sử dụng đối tượng SqlConnection bạn có thể bạn có thể chỉ định số lượng kết nối lớn nhất cho phép trong pool bằng cách chỉ định Max pool size (mặc định là 100) và Min pool Size .

```
SqlConnection mySqlConnection =  
    new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa;" +  
        "max pool size=10;min pool size=5");
```

### **Listing 7.2: CONNECTIONPOOLING.CS**

```
/*  
ConnectionPooling.cs illustrates connection pooling  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class ConnectionPooling  
{  
    public static void Main()  
    {  
        // create a SqlConnection object to connect to the database,  
        // setting max pool size to 10 and min pool size to 5  
        SqlConnection mySqlConnection =  
            new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa;" +  
                "max pool size=10;min pool size=5");  
  
        // open the SqlConnection object 10 times  
        for (int count = 1; count <= 10; count++)  
        {  
            Console.WriteLine("count = "+ count);  
  
            // create a DateTime object and set it to the
```

```

        // current date and time
        DateTime start = DateTime.Now;

        // open the database connection using the
        // Open() method of the SqlConnection object
        mySqlConnection.Open();

        // subtract the current date and time from the start,
        // storing the difference in a TimeSpan
        TimeSpan timeTaken = DateTime.Now - start;

        // display the number of milliseconds taken to open
        // the connection
        Console.WriteLine("Milliseconds = "+ timeTaken.Milliseconds);

        // display the connection state
        Console.WriteLine("mySqlConnection.State = "+
        mySqlConnection.State);

        // close the database connection using the Close() method
        // of the SqlConnection object
        mySqlConnection.Close();
    }
}

```

**The output from this program is as follows:**

```

count = 1
Milliseconds = 101
mySqlConnection.State = Open
count = 2
Milliseconds = 0
mySqlConnection.State = Open
count = 3
Milliseconds = 0
mySqlConnection.State = Open
count = 4
Milliseconds = 0
mySqlConnection.State = Open
count = 5
Milliseconds = 0
mySqlConnection.State = Open
count = 6
Milliseconds = 0
mySqlConnection.State = Open
count = 7
Milliseconds = 0
mySqlConnection.State = Open
count = 8
Milliseconds = 0
mySqlConnection.State = Open
count = 9
Milliseconds = 0
mySqlConnection.State = Open
count = 10

```

Milliseconds = 0  
mySqlConnection.State = Open

## **TRUY XUẤT TRẠNG THÁI CỦA ĐỐI TƯỢNG KẾT NỐI:**

Bạn sử dụng thuộc tính **state** của kết nối để lấy thông tin về trạng thái hiện tại của kết nối đến cơ sở dữ liệu, thuộc tính state trả về một hằng từ bảng liệt kê **connectionstate**.

Table 7.4: ConnectionState CONSTANTS

TÊN HẲNG	MÔ TẢ
Broken	Hỏng kết nối. điều này xảy ra sau khi bạn mở đối tượng kết nối. bạn có thể đóng kết nối và mở lại
Closed	Kết nối đã đóng.
Connecting	Kết nối đang thiết lập sự truy cập đến cơ sở dữ liệu.
Executing	Kết nối đang thực thi một lệnh (command).
Fetching	Kết nối đang nhận thông tin từ cơ sở dữ liệu.
Open	Kết nối đang mở.

Thí dụ sau đây sử dụng thuộc tính state để kiểm tra trạng thái kết nối có phải đang đóng không trước khi mở kết nối

```
if (mySqlConnection.State == ConnectionState.Closed)
{
    mySqlConnection.Open();
}
```

## **SỬ DỤNG CÁC BIẾN CỐ CỦA ĐỐI TƯỢNG KẾT NỐI:**

Những lớp kết nối có hai biến cố hữu ích: **StateChange** và **InfoMessage**.

### **BIẾN CỐ *StateChange*:**

Biến cố StateChange phát ra khi trạng thái của kết nối thay đổi, bạn có thể sử dụng biến cố này để theo dõi trạng thái của đối tượng kết nối.

Phương thức nắm giữ một biến cố được biết như một bộ xử lý sự kiện (event handler ).

Bạn gọi phương thức này khi một sự kiện đặc trưng được tung ra. Tất cả các phương thức xử lý biến cố đều phải trả về một giá trị void và nhận hai tham số. tham số thứ nhất là một đối tượng ( của lớp System.Object), và nó đại diện cho đối tượng phát ra biến cố.

Chú ý: lớp System.Object là lớp cơ sở của tất cả các lớp. nói cách khác , tất cả các lớp đều bắt nguồn từ lớp System.Object.

Tham số Second là một đối tượng của lớp bắt nguồn từ lớp System.EventArgs .

lớp System.EventArgs là lớp cơ sở nắm giữ dữ liệu về biến cố và mô tả những chi tiết về biến cố. trong trường hợp của biến cố StateChange , đối tượng Second này là thuộc về lớp StateChangeEventArgs

Thí dụ dưới đây định nghĩa một phương thức tên StateChangeHandler để xử lý biến cố StateChange . chú ý rằng tham số Second cho phương thức này là đối tượng StateChangeEventArgs. Bạn lấy thông tin trạng thái nguyên thủy của kết nối sử dụng thuộc tính OriginalState của đối tượng này, và thông tin trạng thái hiện tại sử dụng thuộc tính CurrentState.

```
public static void StateChangeHandler(object mySender, StateChangeEventArgs myEvent)
{
    Console.WriteLine("mySqlConnection State has changed from "+
        myEvent.OriginalState + "to " + myEvent.CurrentState
    );
}
```

Để theo dõi một biến cố, bạn phải đăng kí phương thức xử lí biến cố (event handler method)  
 Với biến cố đó . thí dụ: phát biểu dưới đây đăng kí phương thức xử lí biến cố \_  
 StateChangeHandler() với biến cố StateChange của đối tượng mySqlConnection

## **ĐĂNG KÍ MỘT PHƯƠNG THỨC XỬ LÝ BIẾN CỐ:**

```
mySqlConnection.StateChange +=
    new StateChangeEventHandler(StateChangeHandler);
```

bất cứ khi nào biến cố StateChange phát khởi , thì phương thức StateChangeHandler() sẽ được gọi, , thì  
 phương thức StateChangeHandler() sẽ được gọi, nó hiển thị trạng thái hiện tại của đối tượng  
 mySqlConnection

```
/*
    StateChange.cs illustrates how to use the StateChange event
*/

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

class StateChange
{
    // define the StateChangeHandler() method to handle the
    // StateChange event
    public static void StateChangeHandler(
        object mySender, StateChangeEventArgs myEvent)
    {
        Console.WriteLine("mySqlConnection State has changed from "+
            myEvent.OriginalState + "to " +
            myEvent.CurrentState);
    }

    public static void Main()
    {
        // create a SqlConnection object
        SqlConnection mySqlConnection =
            new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa");

        // monitor the StateChange event using the StateChangeHandler() method
        mySqlConnection.StateChange +=
            new StateChangeEventHandler(StateChangeHandler);

        // open mySqlConnection, causing the State to change from Closed
        // to Open
        Console.WriteLine("Calling mySqlConnection.Open()");
        mySqlConnection.Open();

        // close mySqlConnection, causing the State to change from Open
        // to Closed
        Console.WriteLine("Calling mySqlConnection.Close()");
        mySqlConnection.Close();
    }
}
```

```
}  
}
```

The output from this program is as follows:

```
Calling mySqlConnection.Open()  
mySqlConnection State has changed from Closed to Open  
Calling mySqlConnection.Close()  
mySqlConnection State has changed from Open to Closed
```

## **BIẾN CỐ InfoMessage :**

Biến cố InfoMessage khởi phát khi cơ sở dữ liệu trả về một thông tin cảnh báo tạo ra từ cơ sở dữ liệu. Bạn sử dụng biến cố InfoMessage để theo dõi những thông báo này. Để có được những thông báo này, bạn đọc nội dung của tập hợp lỗi (Errors collection) từ đối tượng SqlInfoMessageEventArgs.

Bạn có thể cung cấp thông tin và thông báo lỗi nhờ sử dụng SQL Server PRINT hoặc những phát biểu RAISEERROR, được mô tả trong chương 4, “ giới thiệu về lập trình Transact-SQL”.

Phương thức InfoMessageHandler() dưới đây được sử dụng để xử lý biến cố InfoMessage. chú ý rằng sự sử dụng tập hợp ERRORS (Errors collection) để hiển thị thông báo.

```
public static void InfoMessageHandler(object mySender, SqlInfoMessageEventArgs myEvent)  
{  
    Console.WriteLine("The following message was produced:\n" +myEvent.Errors[0]);  
}
```

Chú thích : nếu bạn đang sử dụng những bộ cung cấp quản lí OLE DB, bạn thay thế *SqlInfoMessageEventArgs* với *OleDbInfoMessageEventArgs*. nếu bạn đang sử dụng bộ cung cấp quản lí ODBC, bạn thay thế *SqlInfoMessageEventArgs* với *OdbcInfoMessageEventArgs*..

### **Listing 7.4: INFOMESSAGE.CS**

```
/*  
    InfoMessage.cs minh họa sử dụng biến cố InfoMessage như thế nào  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
class InfoMessage  
{  
    // định nghĩa phương thức InfoMessageHandler() để xử lý sự cố  
    // InfoMessage event  
    public static void InfoMessageHandler(  
        object mySender, SqlInfoMessageEventArgs myEvent)  
    {  
        Console.WriteLine("The following message was produced:\n" +  
            myEvent.Errors[0]);  
    }  
  
    public static void Main()  
    {  
        // create a SqlConnection object  
        SqlConnection mySqlConnection =  
            new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa");
```

```

// monitor the InfoMessage event using the InfoMessageHandler() method
mySqlConnection.InfoMessage +=
new SqlInfoMessageEventHandler(InfoMessageHandler);

// open mySqlConnection
mySqlConnection.Open();

// create a SqlCommand object
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

// run a PRINT statement
mySqlCommand.CommandText =
"PRINT 'This is the message from the PRINT statement'";
mySqlCommand.ExecuteNonQuery();

// run a RAISERROR statement
mySqlCommand.CommandText =
"RAISERROR('This is the message from the RAISERROR statement', 10, 1)";
mySqlCommand.ExecuteNonQuery();

// close mySqlConnection
mySqlConnection.Close();
}
}

```

**The output from this program is as follows:**

The following message was produced:

System.Data.SqlClient.SqlError: This is the message from the PRINT statement

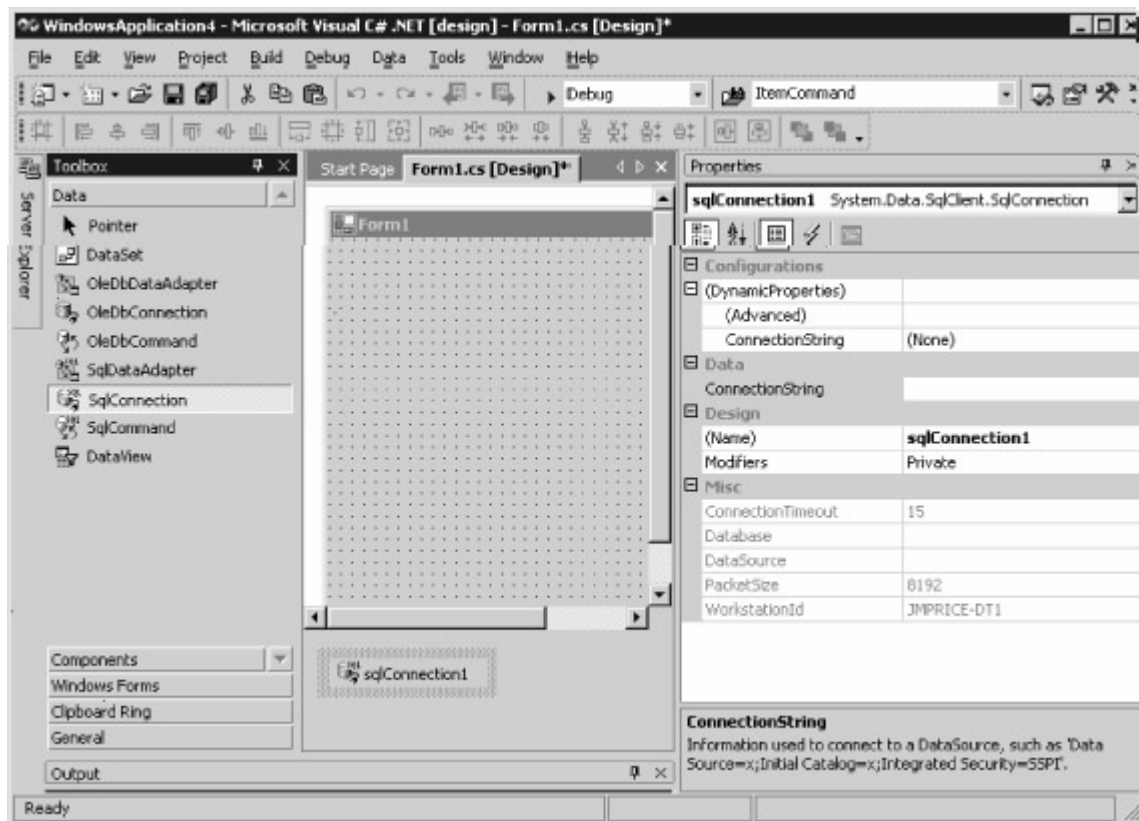
The following message was produced:

System.Data.SqlClient.SqlError: This is the message from the RAISERROR statement

## **TAO MỘT ĐỐI TƯỢNG KẾT NỐI SỬ DỤNG VISUAL STUDIO .NET**

Để tạo một đối tượng kết nối sử dụng Visual Studio .NET, bạn kéo một đối tượng SqlConnection Từ tab data trong toolbox đến form của bạn . bạn sẽ gọi lại một đối tượng kết nối (SqlConnection) Cho phép bạn kết nối với cơ sở dữ liệu SQL Server. Bạn cũng có thể làm tương tự với đối tượng OleDbConnection để kết nối với cơ sở dữ liệu OLE DB.

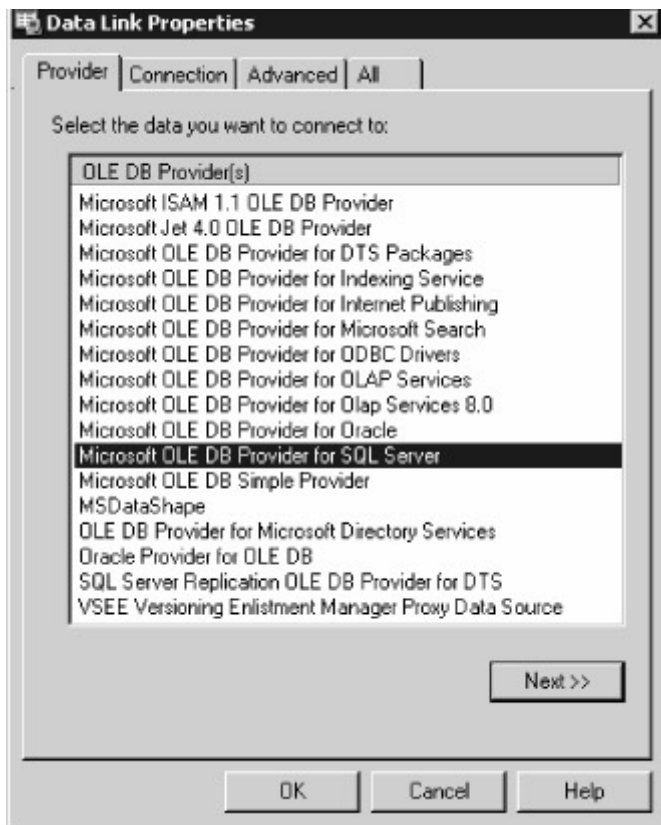




Hình 7.1 : tạo một đối tượng SqlConnection với Visual Studio .NET

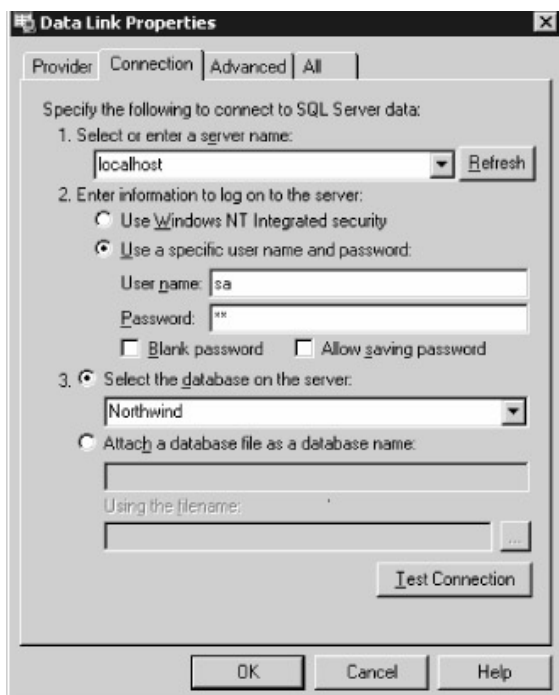
Một khi bạn đã tạo một đối tượng SqlConnection, đối tượng này sẽ xuất hiện trên khay dưới form. Khay này được sử dụng để chứa những thành phần không hiển thị như đối tượng SqlConnection. Những đối tượng khác hiển thị trên khay là đối tượng SqlCommand. Những đối tượng này được xem không hiển thị vì chúng không được nhìn thấy khi bạn khởi chạy form. Đương nhiên bạn vẫn có thể làm việc với chúng một cách trực quan khi thiết kế form.

Bên phải của form, bạn chú ý đến cửa sổ thuộc tính, nơi mà bạn sử dụng để thiết đặt những thuộc tính cho đối tượng SqlConnection. Để thiết đặt thuộc tính ConnectionString\_ mô tả chi tiết về cơ sở dữ liệu kết nối, bạn có thể gõ trực tiếp chuỗi kết nối vào hoặc click vào danh sách sổ xuống và thiết đặt ConnectionString một cách trực quan. Để làm điều này, bạn chọn New Connection trong danh sách Sổ xuống, sẽ hiển thị hộp thoại Data Link Properties. Hộp thoại này chứa bốn Tab, Tab Provider cho phép bạn chọn lựa kiểu của bộ cung cấp mà bạn muốn kết nối, như hình 7.2



Hình 7.2: chọn lựa bộ cung cấp dữ liệu

Click nút Next để chuyển đến Tab Connection ( bạn cũng có thể nhấn Tab Connection cách trực tiếp)  
Nơi bạn nhập chi tiết về kết nối đến cơ sở dữ liệu của bạn. như hình 7.3 dưới đây



Hình 7.3: nhập những chi tiết về kết nối

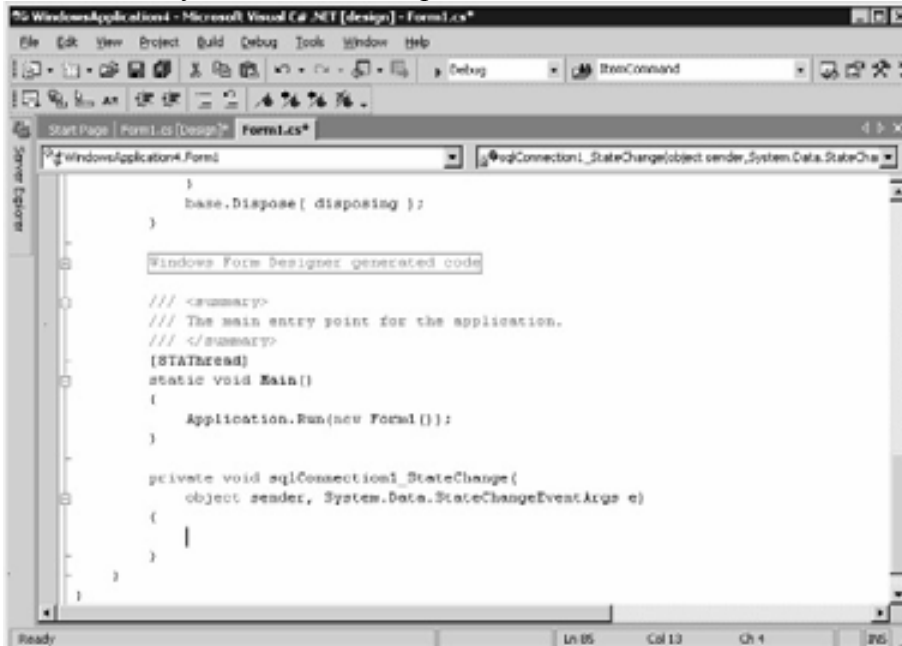
Cảnh báo: vì những lí do an toàn, không chọn hộp check “Allow Saving Password”. Nếu bạn thực hiện điều này Password của bạn sẽ được lưu trong code, và người nào đó có thể đọc được Password của bạn trong code.

Sau khi bạn đã nhập chi tiết về kết nối của bạn, bạn có thể click nút Test Connection để bảo đảm những chi tiết là chính xác . click OK để lưu thiết lập

Trên máy tính của tôi, thuộc tính ConnectionString của đối tượng kết nối đến cơ sở dữ liệu SQL Server Northwind được thiết lập như sau:

data source=localhost;initial catalog=Northwind;persist security info=False;  
user id=sa;pwd=sa;workstation id=JMPRISE-DT1;packet size=4096

bạn có thể thêm code cho một biến cố trong VS .NET. thí dụ, bạn muốn thêm code cho biến cố State-change của đối tượng SqlConnection1 đã được tạo. để làm điều này, trước tiên chọn SqlConnection1 trên khay, rồi click vào nút Events (biểu tượng tia sét) bên trên cửa sổ properties, một danh sách các biến cố của đối tượng SqlConnection1 sẽ được hiển thị trong cửa sổ properties. Double click vào tên của biến cố mà bạn muốn viết mã code. VS >NET sẽ hiển thị khung code và tạo Một khung sườn của phương thức event handler cho bạn, như trình bày trong hình 7.7 dưới đây. Vị trí con nháy sẽ là nơi bạn nhập code.



```
private void sqlConnection1_StateChange(  
object sender, System.Data.StateChangeEventArgs e)  
{ // đoạn code bạn nhập dưới đây  
  Console.WriteLine("State has changed from "+  
    e.OriginalState + "to "+e.CurrentState  
);
```

Sau khi bạn đã tạo đối tượng SqlConnection bạn có thể làm việc với các đối tượng ADO >NET khác như đối tượng SqlCommand v.v bạn sẽ học cách thực hiện điều này với VS .NET trong chương 8.

## **CHƯƠNG 8: THỰC THI CÁC LỆNH CƠ SỞ DỮ LIỆU: EXECUTING DATABASE COMMANDS**

Những lệnh về cơ sở dữ liệu được thực thi bởi đối tượng Command, và là bộ phận của bộ cung cấp được quản lí. Có ba lớp Command : SqlCommand, OleDbCommand, và OdbcCommand. Bạn sử dụng

Đối tượng command để thực thi một phát biểu SQL Select, Insert, Update, hoặc Delete. Bạn cũng có thể sử dụng đối tượng Command để gọi một Stored procedure ( phương thức được thiết lập sẵn bởi ngôn ngữ SQL server được dự trữ trong cơ sở dữ liệu) , hoặc truy xuất những hàng và cột trong một bảng chỉ định; đối tượng Command truyền thông với cơ sở dữ liệu nhờ sử dụng đối tượng Connection

### **LỚP SQLCOMMAND:**

Bạn sử dụng một đối tượng của lớp SqlCommand để thực thi một lệnh trên một cơ sở dữ liệu SQL Server, một đối tượng của lớp OleDbCommand để thực thi một lệnh trên bất cứ cơ sở dữ liệu nào hỗ trợ OLE DB, như Oracle hoặc Access, và một đối tượng của lớp OdbcCommand để thực thi một lệnh

trên cơ sở dữ liệu nào hỗ trợ ODBC. bảng dưới đây trình bày một vài thuộc tính và phương thức của SqlCommand.

Table 8.1: SqlCommand PROPERTIES

THUỘC TÍNH	Kiểu	MÔ TẢ
CommandText	string	Lấy hoặc thiết đặt phát biểu SQL, gọi stored procedure , hoặc bảng để truy xuất dữ liệu
CommandTimeout	int	Lấy hoặc thiết đặt số giây chờ đợi trước khi kết thúc một cố gắng thực thi lệnh (command). Thời gian mặc định là 30 giây.
CommandType	CommandType	Lấy hoặc thiết đặt một giá trị cho biết thuộc tính CommandText được thể hiện như thế nào. Những giá trị hợp lệ là : CommandType.Text, CommandType.StoredProcedure, và CommandType.TableDirect. Text cho biết command là một phát biểu SQL. StoredProcedure cho biết command là một lệnh gọi stored procedure . TableDirect cho biết tên của một bảng, mà từ đó tất cả các hàng và cột được truy xuất. mặc định là Text.
Connection	string	Trả về tên của một kết nối cơ sở dữ liệu.
DesignTimeVisible	bool	Lấy hoặc thiết đặt một giá trị Boolean cho biết một đối tượng Command có được hiển thị trong một Control của bộ thiết kế windows form. Giá trị mặc định là false.
Parameters	SqlParameterCollection	Trả về những tham số( parameters) (nếu có) để cung cấp cho Command . khi sử dụng một kết nối (SqlConnection) những tham số này được lưu trữ trong một đối tượng SqlParameterCollection
Transaction	SqlTransaction	Lấy hoặc thiết đặt giao dịch dữ liệu cho Command (database transaction
UpdatedRowSource	UpdateRowSource	Lấy hoặc thiết đặt những kết quả của thực thi Command được ứng dụng như thế nào đến một đối tượng DataRow khi một phương thức Update() của đối tượng DataAdapter được gọi.
PHƯƠNG THỨC	GIÁ TRỊ TRẢ VỀ	MÔ TẢ
Cancel()	void	Hủy bỏ thực thi của lệnh (command).
CreateParameter()	SqlParameter	Tạo một tham số mới cho command.
ExecuteNonQuery()	int	Sử dụng để thực thi những phát biểu SQL không trả về một tập giá trị. Những phát biểu này bao gồm INSERT, UPDATE, và DELETE, những phát biểu thuộc ngôn ngữ định nghĩa dữ liệu, hoặc những lệnh gọi Stored Procedure Không trả về một tập dữ liệu. Nó trả về một giá trị kiểu Int là số hàng của cơ sở dữ liệu bị ảnh hưởng bởi thực thi Command, nếu có.
ExecuteReader()	SqlDataReader	Được sử dụng để thực thi những phát biểu SQL SELECT, những lệnh bảng trực tiếp (TableDirect commands) Hoặc Stored Procedure trả về một tập giá trị trong đối tượng DataReader.
ExecuteScalar()	object	Được sử dụng để thực thi những phát biểu SQL SELECT trả về một giá trị đơn (những giá trị khác bị bỏ qua). Kết quả trả về của Command là một đối tượng.
ExecuteXmlReader()	XmlReader	Được sử dụng để thực thi những phát biểu SQL SELECT để trả về cơ sở dữ liệu XML. Trả về tập kết quả trong một đối tượng XmlReader. Chỉ ứng dụng cho lớp SqlCommand.
Prepare()	void	Tạo một bản dịch dự phòng của command, đôi khi , những kết quả nhanh hơn trong sự thực thi command.

PHƯƠNG THỨC	GIÁ TRỊ TRẢ VỀ	MÔ TẢ
ResetCommandTimeout()	void	Thiết lập lại thuộc tính CommandTimeout về giá trị mặc định.

**Chú thích** Mặc dù lớp SqlCommand được dành riêng cho SQLServer, rất nhiều thuộc tính và phương thức trong lớp này cũng tương tự như các thuộc tính và phương thức trong những lớp *OleDbCommand* và *OdbcCommand*.

**Kĩ xảo** Bạn thực sự sẽ làm tốt hơn khi từ bỏ T-SQL EXECUTE Command và sử dụng CommandType.StoredProcedure để thực thi một stored procedure. Bởi vì bạn có thể đọc được những giá trị được trả về từ một Stored Procedure thông qua phát biểu RETURN, cái mà bạn có thể thực hiện khi thiết định CommandType là Stored Procedure.

## **TAO MỘT ĐỐI TƯỢNG SOLCOMMAND:**

có hai cách để tạo một đối tượng SqlCommand:

- sử dụng một trong số những bộ khởi tạo SqlCommand.
- gọi phương thức CreateCommand() của một đối tượng SqlConnection.

**CHÚ Ý :** bạn có thể sử dụng những cách tương tự như trình bày dưới đây để tạo đối tượng *OleDbCommand* hoặc *OdbcCommand*.

## **TAO MỘT ĐỐI TƯỢNG SOLCOMMAND SỬ DỤNG BỘ KHỞI TẠO:**

```
SqlCommand()
SqlCommand(string commandText)
SqlCommand(string commandText, SqlConnection mySqlConnection)
SqlCommand(string commandText, SqlConnection mySqlConnection, SqlTransaction
mySqlTransaction)
```

**CommandText :** chứa phát biểu SQL của bạn, lệnh gọi stored procedure, hoặc bảng để truy xuất dữ liệu.

**mySqlConnection:** là đối tượng SqlConnection.

**mySqlTransaction:** là đối tượng SqlTransaction của bạn.

Trước khi bạn sử dụng một đối tượng SqlCommand đầu tiên bạn cần có một đối tượngSqlConnection, Để giao tiếp với một cơ sở dữ liệu SQL Server:

```
mySqlConnection.ConnectionString =
"server=localhost;database=Northwind;uid=sa;pwd=sa";
```

Tiếp theo bạn có thể tạo một đối tượng SqlCommand mới sử dụng phát biểu sau:

```
SqlCommand mySqlCommand = new SqlCommand();
```

Sau đó thiết đặt thuộc tính Connection của mySqlCommand là mySqlConnection:

```
mySqlCommand.Connection = mySqlConnection;
```

và đối tượng mySqlCommand sẽ sử dụng ngay đối tượng mySqlConnection để giao tiếp với cơ sở dữ liệu.

bây giờ, thuộc tính CommandType của đối tượng Connection sẽ quyết định kiểu của Command được thực thi. Bạn có thể sử dụng bất cứ giá trị nào trong kiểu liệt kê System.Data.CommandType để chỉ định thuộc tính CommandType.

Table 8.3: CommandType ENUMERATION VALUES

GIÁ TRỊ	MÔ TẢ
Text	Cho biết command là một phát biểu SQL. Text là giá trị mặc định.
StoredProcedure	Cho biết Command là một lệnh gọi stored procedure.
TableDirect	Cho biết tên của một bảng, mà tất cả những hàng và cột sẽ được truy xuất. chú thích: những đối tượng SqlCommand không hỗ trợ TableDirect. Bạn phải sử dụng một đối tượng của một trong những lớp Command khác thay thế.

Bạn sẽ tìm hiểu cách sử dụng ba kiểu Command này trong chương này. Bây giờ tôi sẽ đặt trọng tâm vào kiểu Text Command, là một phát biểu SQL.

Bạn có thể thiết đặt Command được thực thi sử dụng thuộc tính CommandText của đối tượng Command của bạn. thí dụ sau đây thiết đặt thuộc tính CommandText của MySqlCommand với một phát biểu SELECT ..

```
mySqlCommand.CommandText =
"SELECT TOP 10 CustomerID, CompanyName, ContactName, Address " +
"FROM Customers " +
"ORDER BY CustomerID";
```

Bạn cũng có thể đặt đối tượng Command và Connection vào bộ khởi tạo trong cùng một bước với việc tạo một Command. Thí dụ:

```
SqlCommand mySqlCommand = new SqlCommand(
"SELECT TOP 5 CustomerID, CompanyName, ContactName, Address " +
"FROM Customers ",
mySqlConnection);
```

trong phần kế tiếp , bạn sẽ học cách tạo một đối tượng SqlCommand sử dụng phương thức CreateCommand() của một đối tượng SqlConnection.

## **KHOI TAO MỘT ĐỐI TƯỢNG *SqlCommand* SỬ DỤNG PHƯƠNG THỨC CreateCommand().**

Ngoài cách khởi tạo một đối tượng SqlCommand sử dụng những bộ khởi tạo, bạn có thể sử dụng phương thức CreateCommand của đối tượng SqlConnection. Phương thức này trả về một đối tượng SqlCommand mới. thí dụ:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

Đối tượng mySqlCommand sẽ sử dụng mySqlConnection để tương tác với cơ sở dữ liệu.

## **THỰC THI NHỮNG PHÁT BIỂU *SELECT* VÀ NHỮNG LỆNH TABLEDIRECT:**

Một TableDirect Command thực ra là một phát biểu SELECT, nó trả về tất cả những hàng và cột của một bảng chỉ định. Một đối tượng Command có ba phương thức bạn có thể sử dụng để thực thi một phát biểu SELECT hoặc một TableDirect Command.

Table 8.4: NHỮNG PHƯƠNG THỨC TRUY XUẤT THÔNG TIN TỪ CƠ SỞ DỮ LIỆU

PHƯƠNG THỨC	Kiểu TRẢ VỀ	MÔ TẢ
ExecuteReader()	SqlDataReader	Được dùng để thực thi một phát biểu SQL SELECT, TableDirect Commands hoặc các lệnh gọi stored procedure ,nó trả về một tập hợp



Table 8.4: NHỮNG PHƯƠNG THỨC TRUY XUẤT THÔNG TIN TỪ CƠ SỞ DỮ LIỆU

PHƯƠNG THỨC	Kiểu TRẢ VỀ	MÔ TẢ
		kết quả trong một đối tượng DataReader.
ExecuteScalar()	object	Được dùng để thực thi những phát biểu SQL SELECT và trả về một giá trị đơn (bỏ qua những giá trị khác). Giá trị đơn được trả về như một đối tượng.
ExecuteXmlReader()	XmlReader	Được dùng để thực thi những phát biểu SQL SELECT và trả về một XML data. Tập hợp kết quả trả về trong một đối tượng XmlReader . chỉ ứng dụng cho lớp SqlCommand.

## **THỰC THI PHÁT BIỂU *SELECT* SỬ DỤNG SỬ DỤNG PHƯƠNG THỨC *ExecuteReader()*:**

Hãy xem một thí dụ : thực thi một phát biểu SELECT sử dụng phương thức ExecuteReader(). Phương thức này trả về một tập hợp kết quả trong một đối tượng DataReader, mà bạn có thể dùng nó để đọc những hàng được trả về từ cơ sở dữ liệu. thí dụ , code sau đây khởi tạo những đối tượng cần thiết và thực thi một phát biểu SELECT truy xuất 05 dòng đầu tiên từ bảng Customers:

```
SqlConnection mySqlConnection =
    new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa");
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT TOP 5 CustomerID, CompanyName, ContactName, Address " +
    "FROM Customers " +
    "ORDER BY CustomerID";
mySqlConnection.Open();
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
```

Kỹ xảo: bạn chú ý rằng tôi đã không gọi phương thức Open() của đối tượng SqlConnection cho đến khi ngay trước sự gọi phương thức ExecuteReader() của đối tượng SqlCommand. Đây là sự định trước. Bằng cách mở kết nối vào thời điểm rất cuối cùng, bạn giảm thiểu chi phí thời gian kết nối với cơ sở dữ liệu và do đó gìn giữ những tài nguyên cơ sở dữ liệu.

Tập kết quả được trả về bởi mySqlCommand được lưu trữ trong mySqlDataReader. Và rồi bạn sẽ đọc những hàng từ mySqlDataReader sử dụng phương thức Read(). Phương thức này trả về một giá trị Boolean, true nếu Như có một hàng khác để đọc, ngược lại sẽ trả về giá trị false. Bạn có thể đọc một giá trị riêng rẽ của một cột trong một hàng từ mySqlDataReader bằng cách thông qua chỉ định tên của cột trong cặp dấu ngoặc vuông. Thí dụ, để đọc cột CustomerID, bạn dùng mySqlDataReader["CustomerID"].

Chú ý: bạn cũng có thể chỉ định cột mà bạn muốn đọc bằng cách đặt một giá trị số (chỉ số cột trong tập hợp) trong cặp dấu ngoặc vuông. Thí dụ, mySqlDataReader[0] cũng trả về giá trị của cột CustomerID. 0 tương ứng với cột đầu tiên trong bảng, như trong thí dụ này là cột CustomerID.

Bạn cũng có thể sử dụng phương thức Read() trong một phát biểu while loop để đọc tuần tự mỗi lượt một hàng, Như trình bày trong ví dụ dưới đây:

```
while (mySqlDataReader.Read())
{
    Console.WriteLine("mySqlDataReader[" CustomerID"] = " +
        mySqlDataReader["CustomerID"]);
}
```

```

        Console.WriteLine("mySqlDataReader[\" CompanyName\"] = " +
mySqlDataReader["CompanyName"]);
        Console.WriteLine("mySqlDataReader[\" ContactName\"] = " +
mySqlDataReader["ContactName"]);
        Console.WriteLine("mySqlDataReader[\" Address\"] = " +
mySqlDataReader["Address"]);
    }

```

### DƯỚI ĐÂY LÀ MỘT CHƯƠNG TRÌNH ĐẦY ĐỦ VỀ SỬ DỤNG ĐỐI TƯỢNG SQLDATAREADER:

```

/*
ExecuteSelect.cs illustrates how to execute a SELECT
statement using a SqlCommand object
*/

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

class ExecuteSelect
{
    public static void Main()
    {
        // create a SqlConnection object to connect to the database
        SqlConnection mySqlConnection =
        new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa"
        );

        // create a SqlCommand object
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // set the CommandText property of the SqlCommand object to
        // the SELECT statement
        mySqlCommand.CommandText =
        "SELECT TOP 5 CustomerID, CompanyName, ContactName, Address " +
        "FROM Customers " +
        "ORDER BY CustomerID";

        // open the database connection using the
        // Open() method of the SqlConnection object
        mySqlConnection.Open();

        // create a SqlDataReader object and call the ExecuteReader()
        // method of the SqlCommand object to run the SQL SELECT statement
        SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        // read the rows from the SqlDataReader object using
        // the Read() method
        while (mySqlDataReader.Read())
        {
            Console.WriteLine("mySqlDataReader[\" CustomerID\"] = " +
mySqlDataReader["CustomerID"]);
            Console.WriteLine("mySqlDataReader[\" CompanyName\"] = " +
mySqlDataReader["CompanyName"]);
            Console.WriteLine("mySqlDataReader[\" ContactName\"] = " +

```



```

        mySqlDataReader["ContactName"]);
        Console.WriteLine("mySqlDataReader[\" Address\"] = " +
        mySqlDataReader["Address"]);
    }

    // close the SqlDataReader object using the Close() method
    mySqlDataReader.Close();

    // close the SqlConnection object using the Close() method
    mySqlConnection.Close();
}

```

The output from this program is as follows:

```

mySqlDataReader["CustomerID"] = ALFKI
mySqlDataReader["CompanyName"] = Alfreds Futterkiste
mySqlDataReader["ContactName"] = Maria Anders
mySqlDataReader["Address"] = Obere Str. 57
mySqlDataReader["CustomerID"] = ANATR
mySqlDataReader["CompanyName"] = Ana Trujillo3 Emparedados y helados
mySqlDataReader["ContactName"] = Ana Trujillo
mySqlDataReader["Address"] = Avda. de la Constitución 2222
mySqlDataReader["CustomerID"] = ANTON
mySqlDataReader["CompanyName"] = Antonio Moreno Taquería
mySqlDataReader["ContactName"] = Antonio Moreno
mySqlDataReader["Address"] = Mataderos 2312
mySqlDataReader["CustomerID"] = AROUT
mySqlDataReader["CompanyName"] = Around the Horn
mySqlDataReader["ContactName"] = Thomas Hardy
mySqlDataReader["Address"] = 120 Hanover Sq.
mySqlDataReader["CustomerID"] = BERGS
mySqlDataReader["CompanyName"] = Berglunds snabbköp
mySqlDataReader["ContactName"] = Christina Berglund
mySqlDataReader["Address"] = Berguvsvägen 8

```

## **KIỂM SÓAT HÀNH VI CỦA LỆNH(COMMAND) SỬ DỤNG SỬ DỤNG PHƯƠNG THỨC *ExecutReader()***

Phương thức ExecuteReader() chấp nhận một tham số tùy chọn dùng điều khiển hành vi của Command. Những giá trị của tham số này đến từ lớp liệt kê System.Data.CommandBehavior, những giá trị này được trình bày trong bảng 8.5 dưới đây.

Giá trị	Mô tả
CloseConnection	Chỉ định rằng khi đối tượng SqlDataReader được đóng ,đối tượng SqlConnection cũng đóng theo.
Default	Chỉ định rằng đối tượng Command có thể trả về nhiều tập hợp kết quả.
KeyInfo	Chỉ định đối tượng Command chỉ trả về thông tin về những cột khóa chính trong tập hợp kết quả
SchemaOnly	Chỉ định đối tượng Command chỉ trả về thông tin liên quan đến các cột.
SequentialAccess	Cho phép một đối tượng DataReader đọc những hàng có những cột chứa đựng những giá trị nhị phân lớn. Sự truy cập tuần tự tạo cho DataReader đọc dữ liệu như một luồng. và bạn sử dụng phương thức GetBytes() hay GetChars() của đối tượng DataReader để đọc luồng này. <b>Chú ý :</b> bạn sẽ học chi tiết về đối tượng DataReader trong chương tới.

Giá trị	Mô tả
SingleResult	Chỉ định đối tượng Command trả về một tập kết quả đơn.
SingleRow	Chỉ định đối tượng Command trả về một hàng đơn.

## **SỬ DỤNG THUỘC TÍNH HÀNH VI CỦA COMMAND: *SingleRow***

Bạn sử dụng hành vi lệnh : SingleRow để chỉ định đối tượng Command trả về một hàng đơn. Thí dụ, cho là bạn có một đối tượng Command tên mySqlCommand với thuộc tính CommandText được gán như sau:

```
mySqlCommand.CommandText =
"SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice " +
"FROM Products";
```

Tiếp theo, ví dụ sau đây gán giá trị CommandBehavior. SingleRow Tới phương thức ExecuteReader(), chỉ định đối tượng Command chỉ truy xuất hàng đầu tiên .

```
SqlDataReader mySqlDataReader =
mySqlCommand.ExecuteReader(CommandBehavior.SingleRow);
```

Mặc dù phát biểu SELECT trước đó chỉ định tất cả những hàng đều sẽ được truy xuất từ bảng những sản phẩm,

*Dưới đây là minh họa hiệu quả của việc sử dụng CommandBehavior. SingleRow.*

```
/*
SingleRowCommandBehavior.cs illustrates how to control
the command behavior to return a single row
*/
using System;
using System.Data;
using System.Data.SqlClient;

class SingleRowCommandBehavior
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice " +
            "FROM Products";

        mySqlConnection.Open();

        // pass the CommandBehavior.SingleRow value to the
        // ExecuteReader() method, indicating that the Command object
        // only returns a single row
        SqlDataReader mySqlDataReader =
            mySqlCommand.ExecuteReader(CommandBehavior.SingleRow);

        while (mySqlDataReader.Read())
        {
            Console.WriteLine("mySqlDataReader[\" ProductID\"] = " +
```

```

mySqlDataReader["ProductID"]);
    Console.WriteLine("mySqlDataReader[\" ProductName\"] = " +
mySqlDataReader["ProductName"]);
    Console.WriteLine("mySqlDataReader[\" QuantityPerUnit\"] = " +
mySqlDataReader["QuantityPerUnit"]);
    Console.WriteLine("mySqlDataReader[\" UnitPrice\"] = " +
mySqlDataReader["UnitPrice"]);
}

mySqlDataReader.Close();
mySqlConnection.Close();
}

```

*The output from this program is as follows:*

```

mySqlDataReader["ProductID"] = 1
mySqlDataReader["ProductName"] = Chai
mySqlDataReader["QuantityPerUnit"] = 10 boxes x 20 bags
mySqlDataReader["UnitPrice"] = 18

```

## **SỬ DỤNG THUỘC TÍNH HÀNH VI CỦA COMMAND: *SchemaOnly***

Bạn sử dụng hành vi Command: SchemaOnly để chỉ định đối tượng Command chỉ trả về thông tin của những cột được truy xuất bởi một phát biểu SELECT, hay tất cả những cột nếu bạn sử dụng một Command TableDirect.

Thí dụ: cho là bạn có một đối tượng Command tên mySqlCommand với thuộc tính CommandText được gán như dưới đây:

```

mySqlCommand.CommandText =
"SELECT ProductID, ProductName, UnitPrice " +
"FROM Products " +
"WHERE ProductID = 1";

```

Tiếp theo, thí dụ sau gán giá trị CommandBehavior.SchemaOnly vào phương thức ExecuteReader(), nó chỉ định cho đối tượng Command trả lại thông tin về mô hình dữ liệu:

```

SqlDataReader productsSqlDataReader =
mySqlCommand.ExecuteReader(CommandBehavior.SchemaOnly);

```

trong thí dụ này, từ những cột ProductID, ProductName, và UnitPrice của bảng Products được sử dụng trong phát biểu SELECT trước đó, thông tin về những cột đó được truy xuất thay vì những giá trị cột.

bạn cũng có được thông tin về những cột sử dụng phương thức GetSchemaTable() của đối tượng SqlDataReader. Phương thức GetSchemaTable() trả lại một đối tượng DataTable với những cột chứa những chi tiết của những cột cơ sở dữ liệu được truy xuất:

```

DataTable myDataTable = productsSqlDataReader.GetSchemaTable();

```

Để hiển thị những giá trị trong đối tượng DataTable, bạn có thể sử dụng vòng lặp để hiển thị tên những cột Bảng dữ liệu và nội dung của mỗi cột của bảng dữ liệu :

```

foreach (DataRow myDataRow in myDataTable.Rows)
{
    foreach (DataColumn myDataColumn in myDataTable.Columns)
    {
        Console.WriteLine(myDataColumn + "= " +

```

```

        myDataRow[myDataColumn]);
        if (myDataColumn.ToString() == "ProviderType")
        {
            Console.WriteLine(myDataColumn + "= " +
                ((System.Data.SqlDbType) myDataRow[myDataColumn]));
        }
    }
}

```

chú ý mã này có hai vòng lặp lồng nhau. Vòng lặp phía ngoài lặp lại qua những đối tượng DataRow trong myDataTable, và vòng lặp bên trong lặp lại qua những đối tượng DataColumn trong DataRow hiện thời. Đừng lo lắng quá nhiều về những chi tiết về việc truy nhập một DataTable . bạn sẽ học chi tiết này trong chương 10 ,”sử dụng đối tượng Dataset để lưu trữ dữ liệu”.

Phát biểu if trong vòng lặp Foreach (vòng lặp trong) cần một chút giải thích. Điều mà Tôi đang làm là khảo sát myDataColumn để xem nó chứa ProviderType hay không. ProviderType chứa một giá trị số chỉ định kiểu SQL Sever của cột dữ liệu. tôi buộc số này vào System.Data.SqlDbType, đây là một sự liệt kê chỉ định những kiểu cột SQL Server, như bạn sẽ thấy sau trong phần “ Cung cấp tham số cho Command”. Bảng 9.8 trong phần này trình bày những giá trị liệt kê kiểu SqlDbType. Bằng cách chuyển số ProviderType đến SqlDbType, bạn có thể nhìn thấy tên thực của kiểu cột SQL Server.

Sự lặp lại đầu tiên của vòng lặp ngoài trình bày tất cả những giá trị đối tượng DataColumn của đối tượng DataRow đầu tiên. Điều này gây ra đầu ra sau đây sẽ được sản xuất và trình bày mô hình chi tiết của cột ProductID; chú ý số ProviderType và tên chỉ định ProductID là một kiểu int SQL Server;

```

ColumnName = ProductID
ColumnOrdinal = 0
ColumnSize = 4
NumericPrecision = 0
NumericScale = 0
IsUnique =
IsKey =
BaseCatalogName =
BaseColumnName = ProductID
BaseSchemaName =
BaseTableName =
DataType = System.Int32
AllowDBNull = False
ProviderType = 8
ProviderType = Int
IsAliased =
IsExpression =
IsIdentity = True
IsAutoIncrement = True
IsRowVersion =
IsHidden =
IsLong = False
IsReadOnly = True

```

**Ý nghĩa của những kết quả này được trình bày trong bảng 8.6.**

Table 8.6: những giá trị cột biểu đồ	
GIÁ TRỊ	MÔ TẢ
ColumnName	Tên của cột.
ColumnOrdinal	Số thứ tự cột.

Table 8.6: những giá trị cột biểu đồ

<b>GIÁ TRỊ</b>	<b>MÔ TẢ</b>
ColumnSize	Chiều dài Cực đại ( số kí tự) của một giá trị cột. Cho những kiểu chiều dài cố định SQL Server như int, ColumnSize là độ dài của kiểu này.
NumericPrecision	Tổng số lượng chữ số được dùng để đại diện cho một kiểu dấu chấm động. Một ví dụ về một kiểu dấu chấm động là kiểu float SQL Server.
NumericScale	Số tổng của những chữ số bao gồm những chữ số về bên trái và về bên phải của dấu phẩy ở số thập phân.
IsUnique	Giá trị Boole true/ false: chỉ định liệu hai hàng có thể có cùng giá trị trong cột hiện thời không.
IsKey	Giá trị Boole true/false chỉ định Liệu cột có phải là bộ phận của khóa chính.
BaseCatalogName	Tên tài liệu trong cơ sở dữ liệu chứa đựng cột. BaseCatalogName mặc định là Null.
BaseColumnName	Tên cột trong cơ sở dữ liệu. nó sẽ khác với ColumnName nếu bạn sử dụng một bí danh cho cột.
BaseSchemaName	Tên của mô hình trong cơ sở dữ liệu có chứa đựng cột. BaseSchemaName mặc định là Null.
BaseTableName	Tên của bảng hay view trong cơ sở dữ liệu chứa cột. BaseTableName mặc định là Null.
DataType	Kiểu .NET đã đại diện cho cột. bạn sẽ học về kiểu .NET trong chương tới.
AllowDBNull	Giá trị Boole true/false chỉ định liệu có phải cột có thể chấp nhận một cơ sở dữ liệu giá trị Null.
ProviderType	Chỉ định kiểu dữ liệu của cột.
IsAliased	Giá trị Boole true/false chỉ định liệu có phải tên cột là một bí danh.
IsExpression	Giá trị Boole true/false chỉ định liệu có phải cột là một biểu thức.
IsIdentity	Giá trị Boole true/false chỉ định liệu có phải cột là khóa.
IsAutoIncrement	Giá trị Boole true/false chỉ định liệu có phải cột sẽ được tự động gán một giá trị cho một hàng mới và giá trị này sẽ được tăng dần.
IsRowVersion	Giá trị Boole true/false chỉ định liệu có phải cột chứa đựng một hàng có dữ liệu cố định được định danh là không thể viết vào
IsHidden	Giá trị Boole true/false chỉ định liệu có phải cột được ẩn.
IsLong	Giá trị Boole true/false chỉ định liệu có phải cột chứa một đối tượng long nhị phân (BLOB). A BLOB chứa một chuỗi long của dữ liệu nhị phân.
IsReadOnly	Giá trị Boole true/false chỉ định liệu có phải cột có thể sửa đổi được.

Dưới đây là minh họa hiệu ứng của việc sử dụng CommandBehavior. SchemaOnly và hiển thị mô hình chi tiết cho những cột ProductID, ProductName, và UnitPrice.

### **Listing 8.3: SCHEMAONLYCOMMANDBEHAVIOR.CS**

```

/*
SchemaOnlyCommandBehavior.cs illustrates how to read a table schema
*/

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

class SchemaOnlyCommandBehavior
{
    public static void Main()

```

```

{
    SqlConnection mySqlConnection =
        new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa"
        );

    SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
    mySqlCommand.CommandText =
        "SELECT ProductID, ProductName, UnitPrice " +
        "FROM Products " +
        "WHERE ProductID = 1";

    mySqlConnection.Open();

    // pass the CommandBehavior.SchemaOnly constant to the
    // ExecuteReader() method to get the schema
    SqlDataReader productsSqlDataReader =
        mySqlCommand.ExecuteReader(CommandBehavior.SchemaOnly);
    // read the DataTable containing the schema from the DataReader
    DataTable myDataTable = productsSqlDataReader.GetSchemaTable();

    // display the rows and columns in the DataTable
    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        Console.WriteLine("\nNew column details follow:");
        foreach (DataColumn myDataColumn in myDataTable.Columns)
        {
            Console.WriteLine(myDataColumn + "= " +
                myDataRow[myDataColumn]);
            if (myDataColumn.ToString() == "ProviderType")
            {
                Console.WriteLine(myDataColumn + "= " +
                    ((System.Data.SqlDbType) myDataRow[myDataColumn]));
            }
        }
    }

    productsSqlDataReader.Close();
    mySqlConnection.Close();
}

```

Bạn cần phải chú ý những chi tiết khác nhau cho những cột ProductID, ProductName, và UnitPrice trong đầu ra mà đi theo sau:

```

New column details follow:
ColumnName = ProductID
ColumnOrdinal = 0
ColumnSize = 4
NumericPrecision = 0
NumericScale = 0
IsUnique =
IsKey =
BaseCatalogName =
BaseColumnName = ProductID
BaseSchemaName =
BaseTableName =

```

DataType = System.Int32  
AllowDBNull = False  
ProviderType = 8  
ProviderType = Int  
IsAliased =  
IsExpression =  
IsIdentity = True  
IsAutoIncrement = True  
IsRowVersion =  
IsHidden =  
IsLong = False  
IsReadOnly = True

New column details follow:

ColumnName = ProductName  
ColumnOrdinal = 1  
ColumnSize = 40  
NumericPrecision = 0  
NumericScale = 0  
IsUnique =  
IsKey =  
BaseCatalogName =  
BaseColumnName = ProductName  
BaseSchemaName =  
BaseTableName =  
DataType = System.String  
AllowDBNull = False  
ProviderType = 12  
ProviderType = NVarChar  
IsAliased =  
IsExpression =  
IsIdentity = False  
IsAutoIncrement = False  
IsRowVersion =  
IsHidden =  
IsLong = False  
IsReadOnly = False

Những chi tiết cột mới như sau:

ColumnName = UnitPrice  
ColumnOrdinal = 2  
ColumnSize = 8  
NumericPrecision = 0  
NumericScale = 0  
IsUnique =  
IsKey =  
BaseCatalogName =  
BaseColumnName = UnitPrice  
BaseSchemaName =  
BaseTableName =  
DataType = System.Decimal  
AllowDBNull = True  
ProviderType = 9  
ProviderType = Money  
IsAliased =  
IsExpression =

```
IsIdentity = False
IsAutoIncrement = False
IsRowVersion =
IsHidden =
IsLong = False
IsReadOnly = False
```

## **THỰC THI MỘT PHÁT BIỂU TABLEDIRECT SỬ DỤNG PHƯƠNG THỨC ExecuteReader():**

Khi bạn gán thuộc tính CommandType cho một đối tượng Command là TableDirect, nghĩa là bạn cho biết bạn muốn truy xuất tất cả những hàng và cột của một bảng cụ thể. bạn chỉ định tên của bảng cần truy xuất trong thuộc tính CommandText.

Chú ý: đối tượng SqlCommand không hỗ trợ kiểu Command TableDirect. Thí dụ trong phần này sẽ sử dụng đối tượng OleDbCommand để thay thế.

Như bạn đã biết, bạn có thể sử dụng đối tượng SqlConnection để kết nối với SQL Server. Bạn cũng có thể sử dụng OleDbConnection để kết nối với SQL Server. Bạn đơn giản gán SQLOLEDB cho Provider trong string kết nối chuyển vào bộ khởi tạo OleDbConnection. Thí dụ:

```
OleDbConnection myOleDbConnection =
new OleDbConnection(
"Provider=SQLOLEDB;server=localhost;database=Northwind;" +
"uid=sa;pwd=sa"
);
```

Tiếp theo, bạn tạo một đối tượng OleDbConnection:

```
OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();
```

Sau đó bạn gán CommandType của myOleDbConnection thành CommandType.TableDirect:

```
myOleDbCommand.CommandType = CommandType.TableDirect;
```

tiếp đó bạn chỉ định tên củabảng cần truy xuất sử dụng thuộc tính CommandText. Thí dụ sau gán thuộc tính CommandText của myOleDbCommand là Products:

kế tiếp bạn mở kết nối cơ sở dữ liệu :

```
myOleDbConnection.Open();
```

Cuối cùng bạn thực thi myOleDbCommand sử dụng phương thức ExecuteReader() :

```
OleDbDataReader myOleDbDataReader = myOleDbCommand.ExecuteReader();
```

Phát biểu SQL đã thực sự thực thi là SELECT \* FROM Products, nó truy xuất tất cả những hàng và cột từ bảng Products.

Dưới đây là cốt minh họa cho phần này.

### **Listing 8.4: EXECUTETABLEDIRECT.CS**

```
/*
ExecuteTableDirect.cs illustrates how to execute a
TableDirect command
*/
```



```

using System;
using System.Data;
using System.Data.OleDb;

class ExecuteTableDirect
{
    public static void Main()
    {
        OleDbConnection myOleDbConnection =
            new OleDbConnection(
                "Provider=SQLOLEDB;server=localhost;database=Northwind;" +
                "uid=sa;pwd=sa"
            );
        OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();

        // set the CommandType property of the OleDbCommand object to
        // TableDirect
        myOleDbCommand.CommandType = CommandType.TableDirect;

        // set the CommandText property of the OleDbCommand object to
        // the name of the table to retrieve from
        myOleDbCommand.CommandText = "Products";

        myOleDbConnection.Open();

        OleDbDataReader myOleDbDataReader = myOleDbCommand.ExecuteReader();

        // only read the first 5 rows from the OleDbDataReader object
        for (int count = 1; count <= 5; count++)
        {
            myOleDbDataReader.Read();
            Console.WriteLine("myOleDbDataReader[\" ProductID\"] = " +
                myOleDbDataReader["ProductID"]);
            Console.WriteLine("myOleDbDataReader[\" ProductName\"] = " +
                myOleDbDataReader["ProductName"]);
            Console.WriteLine("myOleDbDataReader[\" QuantityPerUnit\"] = " +
                myOleDbDataReader["QuantityPerUnit"]);
            Console.WriteLine("myOleDbDataReader[\" UnitPrice\"] = " +
                myOleDbDataReader["UnitPrice"]);
        }
        myOleDbDataReader.Close();
        myOleDbConnection.Close();
    }
}

```

*bạn chú ý chương trình này chỉ hiển thị năm hàng đầu trong bảng Products, mặc dù tất cả những hàng đều được truy xuất*

**The output from this program is as follows:**

```

myOleDbDataReader["ProductID"] = 1
myOleDbDataReader["ProductName"] = Chai
myOleDbDataReader["QuantityPerUnit"] = 10 boxes x 20 bags
myOleDbDataReader["UnitPrice"] = 18
myOleDbDataReader["ProductID"] = 2

```

```

myOleDbDataReader["ProductName"] = Chang
myOleDbDataReader["QuantityPerUnit"] = 24 - 12 oz bottles
myOleDbDataReader["UnitPrice"] = 19
myOleDbDataReader["ProductID"] = 3
myOleDbDataReader["ProductName"] = Aniseed Syrup
myOleDbDataReader["QuantityPerUnit"] = 12 - 550 ml bottles
myOleDbDataReader["UnitPrice"] = 10
myOleDbDataReader["ProductID"] = 4
myOleDbDataReader["ProductName"] = Chef Anton's Cajun Seasoning
myOleDbDataReader["QuantityPerUnit"] = 48 - 6 oz jars
myOleDbDataReader["UnitPrice"] = 22
myOleDbDataReader["ProductID"] = 5
myOleDbDataReader["ProductName"] = Chef Anton's Gumbo Mix
myOleDbDataReader["QuantityPerUnit"] = 36 boxes
myOleDbDataReader["UnitPrice"] = 21.35

```

## **THỰC THI MỘT PHÁT BIỂU SELECT SỬ DỤNG PHƯƠNG THỨC ExecuteScalar():**

Bạn sử dụng phương thức ExecuteScalar() để thực thi phát biểu SELECT SQL để trả về một giá trị đơn; những giá trị khác bị bỏ qua. Phương thức ExecuteScalar() trả về một giá trị đơn như là một đối tượng của lớp System.Object. Một sử dụng cho phương thức ExecuteScalar() sẽ thực hiện một phát biểu SELECT sử dụng một chức năng tổng thể như Count() để lấy số hàng trong một bảng. Những chức năng tổng thể này sẽ được trình bày trong chương 4, “giới thiệu về lập trình Transact-SQL”.

Thí dụ, phát biểu sau gán cho thuộc tính CommandText của đối tượng MySqlCommand một phát biểu SELECT sử dụng hàm Count(). Phát biểu này trả về tổng số hàng có trong bảng Products:

```

mySqlCommand.CommandText =
"SELECT COUNT(*) " +
"FROM Products";

```

Tiếp theo, phát biểu sau thực thi phát biểu SELECT trên sử dụng phương thức ExecuteScalar() :

```

int returnValue = (int) mySqlCommand.ExecuteScalar();

```

Bạn chú ý là tôi buộc kiểu đối tượng chung được trả lại từ ExecuteScalar() thành một int trước khi lưu giữ kết quả vào biến int "returnValue".

Code dưới đây minh họa cho sự sử dụng phương thức ExecuteScalar().

### **Listing 8.5: EXECUTESCALAR.CS**

```

/*
ExecuteScalar.cs illustrates how to use the ExecuteScalar()
method to run a SELECT statement that returns a single value
*/

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

class ExecuteScalar
{
    public static void Main()
    {

```

```

SqlConnection mySqlConnection =
new SqlConnection(
"server=localhost;database=Northwind;uid=sa;pwd=sa");
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT COUNT(*) " +
"FROM Products";
mySqlConnection.Open();

// call the ExecuteScalar() method of the SqlCommand object
// to run the SELECT statement
int returnValue = (int) mySqlCommand.ExecuteScalar();
Console.WriteLine("mySqlCommand.ExecuteScalar() = " + returnValue);

mySqlConnection.Close();
}
}

```

*The output from this program is as follows:*

mySqlCommand.ExecuteScalar() = 79

Tất nhiên, đầu ra của các bạn có thể thay đổi phụ thuộc vào số hàng trong bảng những sản phẩm của các bạn.

## **THỰC THI MỘT Command TRUY XUẤT DỮ LIỆU NHƯ MỘT XML SỬ DỤNG PHƯƠNG THỨC ExecuteXmlReader():**

Bạn sử dụng phương thức ExecuteXmlReader() để thực thi một phát biểu SELECT SQL để trả về dữ liệu XML. Phương thức ExecuteXmlReader() trả về những kết quả trong một đối tượng XmlReader, và rồi sau đó bạn sẽ sử dụng nó để đọc dữ liệu XML truy xuất được.

Chú ý phương thức ExecuteXmlReader() chỉ áp dụng cho lớp SqlCommand.

SQL Server mở rộng tiêu chuẩn SQL cho phép bạn truy vấn cơ sở dữ liệu và lấy những kết quả về như XML. Đặc biệt, bạn có thể thêm một mệnh đề FOR XML vào cuối của một phát biểu SELECT. Mệnh đề FOR XML Có cú pháp như dưới đây:

```

FOR XML
{RAW | AUTO | EXPLICIT}
[, XMLDATA]
[, ELEMENTS]
[, BINARY BASE64]

```

### **Bảng 8.7 trình bày sự mô tả về những từ khóa sử dụng trong mệnh đề FOR XML.**

Table 8.7: Những từ khóa cho XML	
Từ Khóa	Mô tả
FOR XML	Chỉ định SQL Server sẽ trả lại những kết quả như XML.
RAW	Chỉ định mỗi hàng trong tập hợp kết quả được trả về như một phần tử "hàng" XML. Những giá trị cột trở thành những thuộc tính của phần tử "hàng".
AUTO	Chỉ định mỗi hàng trong tập hợp kết quả được trả về như một phần tử XML. với tên của bảng sử dụng thay cho phần tử chung "hàng".
EXPLICIT	Chỉ định phát biểu SELECT của các bạn chỉ rõ mối quan hệ cha - con, rồi sẽ được sử dụng bởi SQL Server để tạo ra XML với cấu trúc mạng lưới thích hợp.

Table 8.7: Những từ khóa cho XML

Từ Khóa	Mô tả
XMLDATA	Chỉ định Định nghĩa kiểu tài liệu sẽ bao gồm trong XML được trả về..
ELEMENTS	Chỉ định những cột được trả về như những phần tử của hàng. nói cách khác, những cột được trả về như những thuộc tính của hàng. Bạn chỉ có thể sử dụng chọn lựa này với AUTO.
BINARY BASE64	Chỉ định bất kỳ dữ liệu nhị phân nào được trả về bởi câu truy vấn được mã hóa với BASE64. Nếu bạn muốn truy xuất dữ liệu nhị phân sử dụng kiểu RAW và EXPLICIT, rồi bạn phải sử dụng BASE64 Nhị phân. Trong kiểu AUTO, dữ liệu nhị phân được trả về như một sự tham khảo theo mặc định.

Bạn sẽ xem một thí dụ đơn giản về mệnh đề FOR XML sau đây, và bạn sẽ học chi tiết đầy đủ về mệnh đề này trong chương 16, “sử dụng sự hỗ trợ XML của SQL Server,”

Thí dụ sau đây gán một phát biểu SELECT sử dụng mệnh đề FOR XML AUTO cho thuộc tính CommandText của đối tượng MySqlCommand. Phát biểu SELECT này trả về năm hàng đầu tiên từ bảng Products theo dạng XML.

```
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID " +
"FOR XML AUTO";
```

Tiếp theo, phát biểu sau đây thực thi một SELECT sử dụng phương ExecuteXmlReader():

```
XmlReader myXmlReader = MySqlCommand.ExecuteXmlReader();
```

Chú thích, lớp XmlReader được định nghĩa trong không gian tên (namespace) *System.Xml*.

Để khởi sự đọc XML từ đối tượng XmlReader, bạn sử dụng phương thức Read(). Và rồi bạn kiểm tra để chắc chắn rằng chưa đọc tới hàng cuối cùng- sử dụng thuộc tính EOF của đối tượng XmlReader . EOF trả về *true* nếu như không còn hàng nào để đọc, ngược lại nó trả về *false*. Bạn sử dụng phương thức ReadOuterXml() để đọc XML thực tế từ đối tượng XmlReader.thí dụ sau đây minh họa làm thế nào để đọc XML từ myXmlReader:

```
myXmlReader.Read();
while (!myXmlReader.EOF)
{
    Console.WriteLine(myXmlReader.ReadOuterXml());
}
```

**Code dưới đây minh họa cách sử dụng phương thức ExecuteXmlReader().**

```
/*
ExecuteXmlReader.cs illustrates how to use the ExecuteXmlReader()
method to run a SELECT statement that returns XML
*/

using System;
using System.Data;
using System.Data.SqlClient;
using System.Xml;

class ExecuteXmlReader
```

```

{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=sa");
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // set the CommandText property of the SqlCommand object to
        // a SELECT statement that retrieves XML
        mySqlCommand.CommandText =
            "SELECT TOP 5 ProductID, ProductName, UnitPrice " +
            "FROM Products " + ORDER BY ProductID " + "FOR XML AUTO";

        mySqlConnection.Open();

        // create a SqlDataReader object and call the ExecuteReader()
        // method of the SqlCommand object to run the SELECT statement
        XmlReader myXmlReader = mySqlCommand.ExecuteXmlReader();

        // read the rows from the XmlReader object using the Read() method
        myXmlReader.Read();
        while (!myXmlReader.EOF)
        {
            Console.WriteLine(myXmlReader.ReadOuterXml());
        }

        myXmlReader.Close();
        mySqlConnection.Close();
    }
}

```

Bạn sẽ chú ý là tôi mang vào namespace System. Xml ngay trước đoạn code khởi đầu của chương trình này.

***Kết quả đầu ra của chương trình này như sau đây***

```

<Products ProductID="1" ProductName="Chai" UnitPrice="18.0000"/>
<Products ProductID="2" ProductName="Chang" UnitPrice="19.0000"/>
<Products ProductID="3" ProductName="Aniseed Syrup" UnitPrice="10.0000"/>
<Products ProductID="4" ProductName="Chef Anton's Cajun Seasoning"
UnitPrice="22.0000"/>
<Products ProductID="5" ProductName="Chef Anton's Gumbo Mix" UnitPrice="21.
3500"/>

```

## **THỰC THI NHỮNG LỆNH SỬA ĐỔI THÔNG TIN TRONG CƠ SỞ DỮ LIỆU:**

Bạn có thể sử dụng phương thức ExecuteNonQuery() của đối tượng Command để thực thi để thực thi bất cứ lệnh nào không trả về một tập kết quả từ cơ sở dữ liệu. trong phần này, bạn sẽ học cách sử dụng phương thức ExecuteNonQuery() để thực thi những lệnh sửa đổi thông tin trong cơ sở dữ liệu.

Bạn có thể sử dụng phương thức ExecuteNonQuery() để thực thi những phát biểu SQL như INSERT, UPDATE và DELETE. Bạn cũng có thể sử dụng phương thức ExecuteNonQuery() để gọi thủ tục nội (stored procedures) Không trả về một giá trị, hay những phát biểu thuộc ngôn ngữ định nghĩa dữ liệu (DDL) như CREATE TABLE Và CREATE INDEX. (DDL đã được trình khái quát trong chương 3, “giới thiệu về ngôn ngữ truy vấn có cấu trúc”) Bảng 8.8 tổng kết về phương thức ExecuteNonQuery().

Table 8.8: phương thức ExecuteNonQuery()

Phương thức	Kiểu trả về	Mô tả
ExecuteNonQuery()	int	dùng để thực hiện những câu lệnh SQL mà không trả lại một tập hợp kết quả, như những phát biểu INSERT, UPDATE, và DELETE, những phát biểu DDL hay những hàm gọi Stored procedure mà không trả lại một tập hợp kết quả. Giá trị int được trả lại là số lượng hàng trong cơ sở dữ liệu bị ảnh hưởng bởi lệnh, nếu có.

Bạn sẽ học cách thực hiện những phát biểu INSERT, UPDATE, hay DELETE như thế nào, và làm thế nào để thực hiện những phát biểu DDL trong mục này. Bạn sẽ học làm thế nào để thực thi một hàm gọi stored procedure sau trong mục "thực thi SQL Server Stored Procedure".

## **THỰC THI PHÁT BIỂU INSERT, UPDATE, VÀ DELETE SỬ DỤNG PHƯƠNG THỨC *ExecuteNonQuery*:**

Hãy xem, thí dụ dưới đây thực thi một phát biểu INSERT sử dụng phương thức ExecuteNonQuery(), trước tiên Cần một đối tượng Command:

```
SqlCommand mySqlCommand = MySqlConnection.CreateCommand();
```

Tiếp theo, bạn gán phát biểu INSERT cho thuộc tính CommandText của đối tượng Command của bạn. thí dụ sau đây phát biểu INSERT cho thuộc tính CommandText của mySqlCommand để thêm một hàng vào bảng Customers:

```
mySqlCommand.CommandText =
"INSERT INTO Customers (" +
" CustomerID, CompanyName" +
") VALUES (" +
" 'J2COM', 'Jason Price Corporation'" +
");";
```

Cuối cùng, bạn thực thi phát biểu INSERT này sử dụng phương thức ExecuteNonQuery():

```
int numberOfRows = mySqlCommand.ExecuteNonQuery();
```

phương thức này trả về một giá trị int cho biết tổng số hàng bị ảnh hưởng của lệnh (Command). Trong thí dụ này giá trị trả về là tổng số hàng được thêm vào bảng Customers, là 1, từ một hàng được thêm bởi phát biểu INSERT.

Hãy xem tiếp một ví dụ thực thi một phát biểu UPDATE để sửa đổi hàng mới vừa mới thêm vào. Code sau đây gán một phát biểu UPDATE cho thuộc tính CommandText của mySqlCommand để sửa đổi cột CompanyName cho hàng mới này, và sau đó gọi phương thức ExecuteNonQuery() để thực thi UPDATE:

```
mySqlCommand.CommandText =
"UPDATE Customers " +
"SET CompanyName = 'New Company' " +
"WHERE CustomerID = 'J2COM'";
numberOfRows = mySqlCommand.ExecuteNonQuery();
```

phương thức ExecuteNonQuery() trả về số lượng của hàng bị sửa đổi do phát biểu UPDATE, đó là 1, do một hàng đã được sửa đổi.

c cuối cùng, hãy quan sát một ví dụ thực thi một phát biểu DELETE để xóa hàng mới này:

```
mySqlCommand.CommandText =
```

```
"DELETE FROM Customers " +  
"WHERE CustomerID = 'J2COM'";  
numberOfRows = mySqlCommand.ExecuteNonQuery();
```

ExecuteNonQuery() lại trả về 1, bởi vì chỉ có một hàng bị xóa bởi phát biểu DELETE

Liệt kê dưới đây minh họa sự sử dụng phương thức ExecuteNonQuery() để thực thi phát biểu INSERT, UPDATE, và DELETE trình bày trong phần trên. Chương trình này làm nổi bật một thủ tục có tên DisplayRow() nó truy xuất và trình bày những chi tiết của một hàng được chỉ định trong bảng Customers. DisplayRow() Được sử dụng trong chương trình để trình bày kết quả của những phát biểu INSERT và UPDATE .

```
/*  
ExecuteInsertUpdateDelete.cs illustrates how to use the  
ExecuteNonQuery() method to run INSERT, UPDATE,  
and DELETE statements  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class ExecuteInsertUpdateDelete  
{  
    public static void DisplayRow(  
        SqlCommand mySqlCommand, string CustomerID)  
    {  
        mySqlCommand.CommandText =  
            "SELECT CustomerID, CompanyName " +  
            "FROM Customers " +  
            "WHERE CustomerID = '" + CustomerID + "'";  
  
        SqlDataReader mySqlDataReader =  
            mySqlCommand.ExecuteReader();  
  
        while (mySqlDataReader.Read())  
        {  
            Console.WriteLine("mySqlDataReader[\" CustomerID\"] = " +  
                mySqlDataReader["CustomerID"]);  
            Console.WriteLine("mySqlDataReader[\" CompanyName\"] = " +  
                mySqlDataReader["CompanyName"]);  
        }  
  
        mySqlDataReader.Close();  
    }  
  
    public static void Main()  
    {  
        SqlConnection mySqlConnection =  
            new SqlConnection(  
                "server=localhost;database=Northwind;uid=sa;pwd=sa");  
  
        // create a SqlCommand object and set its Commandtext property  
        // to an INSERT statement  
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
        mySqlCommand.CommandText =
```

```

        "INSERT INTO Customers (" +
        " CustomerID, CompanyName" +
        ") VALUES (" +
        " 'J2COM', 'Jason Price Corporation'" +
        ")";

mySqlConnection.Open();

// call the ExecuteNonQuery() method of the SqlCommand object
// to run the INSERT statement
int numberOfRows = mySqlCommand.ExecuteNonQuery();
Console.WriteLine("Number of rows added = " + numberOfRows);
DisplayRow(mySqlCommand, "J2COM");

// set the CommandText property of the SqlCommand object to
// an UPDATE statement
mySqlCommand.CommandText =
"UPDATE Customers " +
"SET CompanyName = 'New Company' " +
"WHERE CustomerID = 'J2COM'";

// call the ExecuteNonQuery() method of the SqlCommand object
// to run the UPDATE statement
numberOfRows = mySqlCommand.ExecuteNonQuery();
Console.WriteLine("Number of rows updated = " + numberOfRows);
DisplayRow(mySqlCommand, "J2COM");

// set the CommandText property of the SqlCommand object to
// a DELETE statement
mySqlCommand.CommandText =
"DELETE FROM Customers " +
"WHERE CustomerID = 'J2COM'";

// call the ExecuteNonQuery() method of the SqlCommand object
// to run the DELETE statement
numberOfRows = mySqlCommand.ExecuteNonQuery();
Console.WriteLine("Number of rows deleted = " + numberOfRows);

mySqlConnection.Close();
    }
}

```

**The output from this program is as follows:**

```

Number of rows added = 1
mySqlDataReader["CustomerID"] = J2COM
mySqlDataReader["CompanyName"] = Jason Price Corporation
Number of rows updated = 1
mySqlDataReader["CustomerID"] = J2COM
mySqlDataReader["CompanyName"] = New Company
Number of rows deleted = 1

```

## **THỰC THI PHÁT BIỂU DDL SỬ DỤNG PHƯƠNG THỨC `ExecuteNonQuery()`:**

Ngoài việc chạy những phát biểu INSERT, UPDATE, và DELETE, bạn cũng có thể sử dụng phương thức `ExecuteNonQuery()` để thực thi những phát biểu DDL như CREATE TABLE.



Hãy xem xét một thí dụ thực thi một phát biểu CREATE TABLE, tiếp theo một phát biểu ALTER TABLE, theo sau là một phát biểu DROP TABLE. Trước tiên cần một đối tượng Command:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

Tiếp theo bạn gán phát biểu CREATE TABLE cho thuộc tính CommandText của đối tượng Command. Thí dụ sau đây gán một phát biểu CREATE TABLE cho thuộc tính CommandText của đối tượng mySqlCommand\_ tạo ra một bảng tên MyPersons để lưu trữ thông tin về người:

```
mySqlCommand.CommandText =  
"CREATE TABLE MyPersons (" +  
" PersonID int CONSTRAINT PK_Persons PRIMARY KEY," +  
" FirstName nvarchar(15) NOT NULL," +  
" LastName nvarchar(15) NOT NULL," +  
" DateOfBirth datetime" +  
")";
```

Tiếp theo, bạn gọi phương thức ExecuteNonQuery() để thực thi phát biểu CREATE TABLE:

```
int result = mySqlCommand.ExecuteNonQuery();
```

do một phát biểu CREATE TABLE không ảnh hưởng bất kỳ hàng nào, ExecuteNonQuery() trả về giá trị -1.

Thí dụ tiếp theo thực thi một phát biểu ALTER TABLE để thêm một khóa phụ ràng buộc tới bảng MyPersons:

```
mySqlCommand.CommandText =  
"ALTER TABLE MyPersons " +  
"ADD EmployerID nchar(5) CONSTRAINT FK_Persons_Customers " +  
"REFERENCES Customers(CustomerID)";  
result = mySqlCommand.ExecuteNonQuery();
```

Một lần nữa, ExecuteNonQuery() Trả lại -1 do phát biểu ALTER TABLE không ảnh hưởng đến bất kỳ hàng nào.

Thí dụ cuối cùng thực thi một phát biểu DROP TABLE để xóa bảng MyPersons:

```
mySqlCommand.CommandText = "DROP TABLE MyPersons";  
result = mySqlCommand.ExecuteNonQuery();
```

ExecuteNonQuery() lại trả về -1.

Code dưới đây minh họa sự sử dụng phương thức ExecuteNonQuery() để thực thi những phát biểu DDL trình bày ở phần trên.

```
/*  
ExecuteDDL.cs illustrates how to use the ExecuteNonQuery()  
method to run DDL statements  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class ExecuteDDL
```

```

{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // set the CommandText property of the SqlCommand object to
        // a CREATE TABLE statement
        mySqlCommand.CommandText =
            "CREATE TABLE MyPersons (" +
            " PersonID int CONSTRAINT PK_Persons PRIMARY KEY," +
            " FirstName nvarchar(15) NOT NULL," +
            " LastName nvarchar(15) NOT NULL," +
            " DateOfBirth datetime" +
            ")";

        mySqlConnection.Open();

        // call the ExecuteNonQuery() method of the SqlCommand object
        // to run the CREATE TABLE statement
        Console.WriteLine("Creating MyPersons table");
        int result = mySqlCommand.ExecuteNonQuery();
        Console.WriteLine("mySqlCommand.ExecuteNonQuery() = " + result);

        // set the CommandText property of the SqlCommand object to
        // an ALTER TABLE statement
        mySqlCommand.CommandText =
            "ALTER TABLE MyPersons " +
            "ADD EmployerID nchar(5) CONSTRAINT FK_Persons_Customers " +
            "REFERENCES Customers(CustomerID)";

        // call the ExecuteNonQuery() method of the SqlCommand object
        // to run the ALTER TABLE statement
        Console.WriteLine("Altering MyPersons table");
        result = mySqlCommand.ExecuteNonQuery();
        Console.WriteLine("mySqlCommand.ExecuteNonQuery() = " + result);

        // set the CommandText property of the SqlCommand object to
        // a DROP TABLE statement
        mySqlCommand.CommandText = "DROP TABLE MyPersons";

        // call the ExecuteNonQuery() method of the SqlCommand object
        // to run the DROP TABLE statement
        Console.WriteLine("Dropping MyPersons table");
        result = mySqlCommand.ExecuteNonQuery();
        Console.WriteLine("mySqlCommand.ExecuteNonQuery() = " + result);

        mySqlConnection.Close();
    }
}

```

**Đầu ra của chương trình này như sau:**

```
Creating MyPersons table
mySqlCommand.ExecuteNonQuery() = -1
Altering MyPersons table
mySqlCommand.ExecuteNonQuery() = -1
Dropping MyPersons table
mySqlCommand.ExecuteNonQuery() = -1
```

## **GIỚI THIỆU VỀ TRANSACTION (NHỮNG GIAO DỊCH DỮ LIỆU)**

Trong Chương 3, bạn đã thấy bạn có thể nhóm những câu lệnh SQL lại cùng nhau như thế nào vào trong những giao dịch. và rồi Giao dịch được giao phó hay hồi phục như một đơn vị. ví dụ, trong trường hợp một giao dịch công việc ngân hàng, bạn có lẽ đã muốn rút tiền từ một tài khoản và chuyển tiền đặt vào tài khoản khác.

rồi bạn giao phó cả hai sự thay đổi này như một đơn vị, hay nếu xảy ra vấn đề, hoàn nguyên cả hai sự thay đổi này. Bạn sẽ được giới thiệu sử dụng những giao dịch trong ADO.NET Trong mục này. Có ba lớp Giao dịch SqlTransaction, OleDbTransaction, Và OdbcTransaction, và bạn sử dụng một đối tượng của một trong số những lớp này để đại diện cho một giao dịch trong ADO.NET. Tôi sẽ chỉ cách cho bạn làm sao để sử dụng một đối tượng của lớp SqlTransaction mục này.

Chúng ta hãy cho một ví dụ về giao dịch gồm có hai phát biểu INSERT. phát biểu INSERT đầu tiên sẽ thêm một hàng vào những bảng Customers, và phát biểu thứ hai sẽ thêm một hàng vào bảng Orders. Hàng mới trong bảng Orders sẽ tham chiếu với hàng mới trong bảng Customers, và hai phát biểu INSERT như dưới đây:

```
INSERT INTO Customers
CustomerID, CompanyName
) VALUES
'J3COM', 'Jason Price Corporation'
)
```

```
INSERT INTO Orders (
CustomerID
) VALUES (
'J3COM'
)
```

Bạn có thể sử dụng những bước sau đây để thực hiện hai sự phát biểu INSERT này sử dụng một đối tượng SqlTransaction :

1. Tạo ra một đối tượng SqlTransaction và khởi động giao dịch bằng cách gọi Phương thức BeginTransaction() của đối tượng SqlConnection.
2. Tạo một đối tượng SqlCommand để giữ câu lệnh SQL.
3. Đặt thuộc tính Giao dịch (Transaction) cho đối tượng SqlCommand tới đối tượng SqlTransaction được tạo ra trong bước 1.
4. đặt thuộc tính CommandText của đối tượng SqlCommand tới phát biểu INSERT đầu tiên. Phát biểu INSERT này thêm một hàng vào bảng Customers.
5. Chạy phát biểu INSERT đầu tiên sử dụng phương thức ExecuteNonQuery() của đối tượng SqlCommand. Phương thức này được sử dụng bởi vì phát biểu INSERT không trả lại một tập hợp kết quả.
6. đặt thuộc tính CommandText của đối tượng SqlCommand là phát biểu INSERT thứ hai. Phát biểu này thêm một hàng vào bảng Orders.
7. chạy phát biểu INSERT thứ hai sử dụng ExecuteNonQuery() của đối tượng SqlCommand.
8. Giao phó giao dịch Sử dụng phương thức Commit() của đối tượng SqlTransaction. thực thi này tạo thành hai hàng mới bổ sung bền vững trong cơ sở dữ liệu bởi những phát biểu INSERT.

**Danh sách 8.9:EXECUTETRANSACTION.CS**

```
/*
ExecuteTransaction.cs illustrates the use of a transaction
*/
using System;
using System.Data;
using System.Data.SqlClient;
class ExecuteTransaction
{
    public static void Main()
    {
        SqlConnection mySqlConnection = new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa");

        mySqlConnection.Open();

        // step 1: create a SqlTransaction object and start the transaction
        // by calling the BeginTransaction() method of the SqlConnection
        // object
        SqlTransaction mySqlTransaction =
            mySqlConnection.BeginTransaction();

        // step 2: create a SqlCommand object to hold a SQL statement
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // step 3: set the Transaction property for the SqlCommand object
        mySqlCommand.Transaction = mySqlTransaction;

        // step 4: set the CommandText property of the SqlCommand object to
        // the first INSERT statement
        mySqlCommand.CommandText =
            "INSERT INTO Customers (" +
            " CustomerID, CompanyName" +
            ") VALUES (" +
            " 'J3COM', 'Jason Price Corporation'" +
            ")";

        // step 5: run the first INSERT statement
        Console.WriteLine("Running first INSERT statement");
        mySqlCommand.ExecuteNonQuery();

        // step 6: set the CommandText property of the SqlCommand object to
        // the second INSERT statement
        mySqlCommand.CommandText =
            "INSERT INTO Orders (" +
            " CustomerID" +
            ") VALUES (" +
            " 'J3COM'" +
            ")";

        // step 7: run the second INSERT statement
        Console.WriteLine("Running second INSERT statement");
        mySqlCommand.ExecuteNonQuery();
    }
}
```

```

        // step 8: commit the transaction using the Commit() method
        // of the SqlTransaction object
        Console.WriteLine("Committing transaction");
        mySqlTransaction.Commit();
        mySqlConnection.Close();
    }
}

```

### **Chú thích**

*nếu bạn muốn huỷ những câu lệnh SQL tạo ra giao dịch, bạn có thể sử dụng phương thức Rollback() thay cho phương thức Commit(). Theo mặc định, những giao dịch là Roolback. Luôn luôn sử dụng phương thức Commit() hay Rollback() để rõ ràng chỉ ra liệu bạn muốn giao phó hay hồi nguyên những giao dịch của các bạn.*

### **Đầu ra từ chương trình này Như Đì theo sau:**

```

Running first INSERT statement
Running second INSERT statement
Committing transaction

```

Nếu bạn muốn chạy chương trình hơn một lần, bạn sẽ cần loại bỏ hàng được thêm vào bảng Customers và Orders sử dụng những phát biểu DELETE sau (bạn có thể thực hiện điều này nhờ sử dụng công cụ phân tích truy vấn):

```

DELETE FROM Orders
WHERE CustomerID = 'J3COM'

```

```

DELETE FROM Customers
WHERE CustomerID = 'J3COM'

```

### **CUNG CẤP NHỮNG THAM SỐ CHO COMMANDS:**

Trong những ví dụ bạn được nhìn thấy trên tới điểm này, những giá trị cho mỗi cột được đã được viết mã cố định trong những câu lệnh SQL. Chẳng hạn, Trong Danh sách 8.9, được trình bày trước đó, phát biểu INSERT mà thêm hàng vào bảng khách hàng là:

```

INSERT INTO Customers
CustomerID, CompanyName
) VALUES
'J3COM', 'Jason Price Corporation'
)

```

Như bạn có thể nhìn thấy, những giá trị cho những cột CustomerID và CompanyName đã được viết mã cố định Tới ' J3COM ' Và 'Jason Price Corporation'. Nếu bạn phải thực hiện nhiều phát biểu INSERT như vậy, nhập những giá trị cột bằng mã code sẽ mệt nhọc và không có hiệu quả. May mắn thay, bạn có thể sử dụng những tham số để giải quyết vấn đề này. Những tham số cho phép bạn chỉ định những giá trị cột khác nhau trong khi chạy chương trình của các bạn.

Để thực hiện một lệnh chứa những tham số, bạn sử dụng những bước cấp cao sau đây :

1. Tạo ra một đối tượng Lệnh (Command) đang chứa đựng một câu lệnh SQL với tham số placeholders. Placeholders (@....) này đánh dấu vị trí nơi một tham số sẽ được cung cấp.

2. Thêm những tham số vào đối tượng Lệnh.
3. Gán cho những tham số những giá trị chỉ định .
4. thực hiện lệnh (Command).

Chúng ta hãy xem xét những chi tiết của bốn bước khi sử dụng những tham số với SQL Server.

## **BUƯỚC 1: TẠO MỘT ĐỐI TƯỢNG COMMAND CHỨA NHỮNG CÂU LỆNH SQL VỚI NHỮNG THAM SỐ ĐỊA CHỈ:**

bất kỳ nơi nào mà bạn đặt một giá trị cột trong câu lệnh SQL của bạn, bạn chỉ định một tham số địa chỉ (placeholder) thay vào đó. Một tham số địa chỉ (placeholder) đánh dấu vị trí nơi một giá trị sẽ được cung cấp sau đó.

Cú pháp bạn sử dụng cho placeholders phụ thuộc vào cơ sở dữ liệu Bạn Sử dụng. Với máy chủ phục vụ SQL, tham số địa chỉ ( placeholders) sẽ là @CustomerID và @CompanyName. Phát biểu INSERT sau đây sử dụng tham số địa chỉ này Cho Những giá trị cột CustomerID, CompanyName, Và ContactName của bảng khách hàng:

```
INSERT INTO Customers
CustomerID, CompanyName, ContactName
) VALUES
(@CustomerID, @CompanyName, @ContactName
)
```

Bạn có thể sử dụng một tham số địa chỉ bất cứ nơi đâu một giá trị cột là hợp lệ trong một phát biểu Chọn lựa, Chèn, Cập nhật, hay Xóa .

```
SELECT *
FROM Customers
WHERE CustomerID = @CustomerID

UPDATE Customers
SET CompanyName = @CompanyName
WHERE CustomerID = @CustomerID

DELETE FROM Customers
WHERE CustomerID = @CustomerID
```

Chúng ta hãy xem xét tại mã nào đó mà tạo ra một đối tượng SqlCommand và gán thuộc tính CommandText của nó thành một phát biểu INSERT:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"INSERT INTO Customers (" +
" CustomerID, CompanyName, ContactName" +
") VALUES (" +
" @CustomerID, @CompanyName, @ContactName" +
");
```

phát biểu INSERT này sẽ được dùng để thêm một hàng vào bảng Customers. Những giá trị cột cho hàng này sẽ được chỉ rõ nhờ sử dụng những tham số. Mọi thứ đã thực hiện trong mã trước đây, sẽ tạo ra một đối tượng SqlCommand với một phát biểu INSERT có tham số địa chỉ. Trước khi bạn có thể thực hiện lệnh chèn này, bạn cần thêm những tham số thực tới đối tượng SqlCommand và bạn sẽ thực hiện điều này trong bước tiếp theo.

## **BUƯỚC 2: THÊM NHỮNG THAM SỐ VÀO ĐỐI TƯỢNG LỆNH(COMMAND)**

Để thêm những tham số vào đối tượng Command của bạn, bạn sử dụng phương thức Add(). Nó bị quá tải, và phiên bản được dùng trong mục này chấp nhận ba tham số:

- ♦ Chuỗi tham số địa chỉ cho tham số trong câu lệnh SQL của các bạn. Chẳng hạn, @CustomerID là placeholder đầu tiên trong phát biểu INSERT được trình bày trong mục trước đây.
- ♦ Kiểu cho cột trong cơ sở dữ liệu. với SQL Server, những kiểu này được định nghĩa trong lớp liệt kê System.Data.SqlDbType. Bảng 8.9 trình bày những kiểu cơ sở dữ liệu này:

**Bảng 8.9: những thành viên Liệt kê SqlDbType**

Thành viên	Mô tả
BigInt	Một số nguyên 64-bit giữa $-2^{63}$ (-9,223,372,036,854,775,808) và $2^{63}$ = (9,223,372,036,854,775,807).
Binary	Một mảng của những byte với chiều dài cực đại 8,000 phần tử.
Bit	Một giá trị số không dấu có thể là 0, 1, hay một tham chiếu null.
Char	Một chuỗi những ký tự không phải Unicode với chiều dài cực đại 8,000 ký tự.
DateTime	Một giá trị ngày tháng và thời gian giữa 12: 00: 00 _ 1, tháng 1 , 1753 và 11: 59: 59 PM _31, tháng 12, 9999. chính xác tới 3.33 mili-giây.
Decimal	giá trị số chính xác và quy mô, cố định giữa $-1038 + 1$ (và) $1038 - 1$ .
Float	Một số có dấu chấm động 64-bit giữa $-1.79769313486232 \text{ E}308$ và $1.79769313486232 \text{ E}308$ chính xác với 15 chữ số có nghĩa .
Image	Một mảng của những byte với chiều dài cực đại $2^{31}$ =(2,147,483,647).
Int	Một Số nguyên 32-bit được ký mẫu giữa $-2^{31}$ (-2,147,483,648) Và $2^{31}$ = (2,147,483,647).
Money	Một giá trị tiền tệ giữa -922,337,203,685,477.5808 và 922,337,203,685,477.5807. chính xác đối với 1/10,000 th của một đơn vị tiền tệ.
NChar	Một chuỗi của những ký tự Unicode với chiều dài cực đại 4,000 ký tự.
Ntext	Một chuỗi những ký tự Unicode với Một chiều dài cực đại $2^{30}$ ký tự = (1,073,741,823).
NVarChar	Một chuỗi những ký tự Unicode với Một chiều dài cực đại $2^{30}$ = (1,073,741,823).
Real	Một Số dấu chấm động 32-bit giữa $-3.402823 \text{ E}38$ và $3.402823 \text{ E}38$ , chính xác với bảy chữ số có nghĩa .
SmallDateTime	Một giá trị ngày tháng và thời gian giữa 12: 00: 00 AM _ 1, tháng 1.1900 và 11: 59: 59 _ 6, tháng 6, 2079. PM , chính xác đối với 1 phút.
SmallInt	Một Số nguyên 16-bit giữa $-2^{15}$ (-32,768) Và $2^{15}$ = (32,767).

Thành viên	Mô tả
SmallMoney	Một giá trị tiền tệ giữa -214,748.3648 và 214,748.3647. Chính xác đối với 1/10,000 th của một đơn vị tiền tệ.
Text	Một chuỗi những ký tự không phải Unicode với chiều dài cực đại $2^{31} = (2,147,483,647)$ .
Timestamp	Một chuỗi ngày tháng và thời gian trong định dạng yyyyymmddhhmmss.
TinyInt	một số nguyên không dấu 8-bit giữa 0 và $2^8 - 1$ (255).
UniqueIdentifier	Một Giá trị số nguyên 128-bit (16 byte) và là duy nhất ngang qua tất cả các máy tính và những mạng.
VarBinary	Một mảng những byte với chiều dài cực đại 8,000 phần tử.
VarChar	Một chuỗi những ký tự không phải Unicode với chiều dài cực đại 4,000 ký tự.
Variant	Một kiểu dữ liệu mà có thể chứa những số, những chuỗi, những byte hay những giá trị ngày tháng.

◆ Chiều dài cực đại của giá trị tham số. Bạn chỉ định tham số này chỉ khi sử dụng những kiểu độ dài biến số, chẳng hạn, Char và VarChar.

Trước đó trong bước 1, thuộc tính CommandText cho MySqlCommand đã có ba tham số địa chỉ placeholders và là tập hợp như sau:

```
mySqlCommand.CommandText =
"INSERT INTO Customers (" +
" CustomerID, CompanyName, ContactName" +
") VALUES (" +
" @CustomerID, @CompanyName, @ContactName" +
");
```

Những phát biểu sau đây sử dụng Thêm phương thức Add() để Thêm ba tham số Tới MySqlCommand:

```
mySqlCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5);
mySqlCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 40);
mySqlCommand.Parameters.Add("@ContactName", SqlDbType.NVarChar, 30);
```

Chú ý bạn gọi phương thức Add() thông qua thuộc tính Parameters của MySqlCommand. yêu cầu một vài giải thích. Một đối tượng SqlCommand cất giữ những tham số sử dụng Một đối tượng SqlParameterCollection, và là Một tập hợp Của những đối tượng SqlParameter (một đối tượng SqlParameter chứa đựng những chi tiết của một tham số). Một trong số những phương thức của SqlParameterCollection là Add(), và bạn thường sử dụng để thêm một đối tượng SqlParameter vào tập hợp. Bởi vậy, để thêm một tham số Tới MySqlCommand, Bạn gọi phương thức Add() thông qua thuộc tính Parameters của nó.

nếu bạn có thể thấy từ mã trước đây là thêm ba tham số đến MySqlCommand, tham số @CustomerID được định nghĩa như một NChar\_ một chuỗi của những ký tự Unicode với chiều dài cực đại 4,000 ký tự. Một giá trị của 5 được gởi đi như tham số thứ ba tới phương thức Add() cho @CustomerID, ý nghĩa là một "số cực đại 5 ký tự" có thể cung cấp như giá trị tham số. Tương tự, tham số @CompanyName và @ContactName được định nghĩa như một NVarChar\_ một chuỗi những ký tự Unicode- với chiều dài cực đại tương ứng 40 và 30 ký tự,



như chỉ định bởi tham số thứ ba tới phương thức Add(). Bạn sẽ thấy sự thiết đặt của những tham số này với những giá trị cụ thể trong bước tiếp theo.

### **BUỘC 3: GÁN NHỮNG THAM SỐ VỚI NHỮNG GIÁ TRỊ CỤ THỂ:**

Bạn sử dụng thuộc tính Value của mỗi tham số để đặt nó tới một Giá trị chỉ định trong đối tượng Lệnh (Command) các bạn. Những giá trị này sẽ thay thế cho tham số địa chỉ ( placeholders) trong câu lệnh SQL của bạn. Ví dụ sau đây sử dụng thuộc tính Value để đặt những Giá trị cho những tham số bổ sung Trong mục trước đây:

```
mySqlCommand.Parameters["@CustomerID"].Value = "J4COM";  
mySqlCommand.Parameters["@CompanyName"].Value = "J4 Company";  
mySqlCommand.Parameters["@ContactName"].Value = "Jason Price";
```

Trong ví dụ này, những tham số @CustomerID, @CompanyName, Và @ContactName được gán các giá trị tương ứng : J4COM, J4 Company, và Jason Price . Những giá trị này sẽ được thế cho tham số địa chỉ placeholders trong phát biểu INSERT, trở thành:

```
INSERT INTO Customers (  
    CustomerID, CompanyName, ContactName  
) VALUES (  
    'J4COM', 'J4 Company', 'Jason Price'  
)
```

Như bạn có thể nhìn thấy, những giá trị cột giống như những giá trị được chỉ định trong thuộc tính Value cho mỗi tham số.

Bạn cũng có thể thêm một tham số và đặt giá trị của nó trong cùng một bước. Chẳng hạn:

```
mySqlCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5).Value = "J4COM";
```

Bạn cũng có thể đặt một tham số với một giá trị null. như bạn đã học trong Chương 2, "Giới thiệu về những cơ sở dữ liệu, " Một cột định nghĩa với giá trị Null có thể chứa một giá trị Null. Một giá trị Null chỉ định một giá trị cột chưa được biết. Bạn chỉ định một tham số có thể chấp nhận một giá trị null bằng cách đặt thuộc tính IsNullable là true (nếu false thì ngược lại). Chẳng hạn:

```
mySqlCommand.Parameters["@ContactName"].IsNullable = true;
```

Bạn cũng có thể đặt thuộc tính Value của tham số với giá trị Null sử dụng lớp System.DBNull. Chẳng hạn:

```
mySqlCommand.Parameters["@ContactName"].Value = DBNull.Value;
```

Thuộc tính DBNull.Value trả lại một Giá trị null. Trong ví dụ này, cuối cùng phát biểu INSERT trở thành:

```
INSERT INTO Customers (  
    CustomerID, CompanyName, ContactName  
) VALUES (  
    'J4COM', 'J4 Company', NULL
```

)

việc duy nhất còn lại là thực hiện câu lệnh SQL.

## **BƯỚC 4: THỰC THI CÂU LỆNH (COMMAND)**

Để thực thi lệnh, bạn sử dụng một trong số những phương thức của đối tượng Lệnh (Command) . Chẳng hạn:

```
mySqlCommand.ExecuteNonQuery();
```

mã lệnh này chạy phát biểu INSERT để thêm hàng mới vào bảng Customers. Tôi sử dụng phương thức ExecuteNonQuery() vì một phát biểu INSERT không trả lại một tập hợp kết quả từ cơ sở dữ liệu. Bạn cũng có thể sử dụng phương pháp này để thực hiện phát biểu UPDATE và DELETE . Nếu bạn đang thực hiện một phát biểu SELECT , bạn sử dụng những phương thức ExecuteReader(), ExecuteScalar(), Hay ExecuteXmlReader() .

[Danh sách 8.10 minh họa bốn bước này.](#)

### **Danh sách 8.10: USINGPARAMETERS.CS**

```
/*
UsingParameters.cs illustrates how to run an INSERT
statement that uses parameters
*/

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

class UsingParameters
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        mySqlConnection.Open();

        // step 1: create a Command object containing a SQL statement
        // with parameter placeholders
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "INSERT INTO Customers (" +
            " CustomerID, CompanyName, ContactName" +
            ") VALUES (" +
            " @CustomerID, @CompanyName, @ContactName" +
            ")";

        // step 2: add parameters to the Command object
        mySqlCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5);
        mySqlCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 40);
        mySqlCommand.Parameters.Add("@ContactName", SqlDbType.NVarChar, 30);
```

```

// step 3: set the parameters to specified values
mySqlCommand.Parameters["@CustomerID"].Value = "J4COM";
mySqlCommand.Parameters["@CompanyName"].Value = "J4 Company";
mySqlCommand.Parameters["@ContactName"].IsNullable = true;
mySqlCommand.Parameters["@ContactName"].Value = DBNull.Value;

// step 4: execute the command
mySqlCommand.ExecuteNonQuery();
Console.WriteLine("Successfully added row to Customers table");

mySqlConnection.Close();
    }
}

```

*Đầu ra từ chương trình này Như sau:*

Successfully added row to Customers table

## **THỰC THI NHỮNG THỦ TỤC LƯU TRỮ TRONG SQL Server:**

Trong Chương 4, bạn đã thấy cách để tạo và thực thi những thủ tục lưu trữ (stored procedures) sử dụng phát biểu T-SQL như thế nào. Trong mục này, bạn sẽ thấy cách thực thi những thủ tục lưu trữ SQL Server như thế nào khi sử dụng ADO.NET. Trong Bảng 8.1, được trình bày trước đó trong chương này, Tôi đã đề cập đến CommandType của StoredProcedure. Mặc dầu bạn có thể sử dụng CommandType này để chỉ rõ là một lệnh(Command) sẽ thực hiện một thủ tục lưu trữ, bạn thật sự tốt hơn từ bỏ sử dụng lệnh(Command) thực thi T-SQL để Thực hiện một thủ tục lưu trữ. Đây là bởi vì bạn có thể đọc những giá trị được trả về từ một thủ tục lưu trữ thông qua một phát biểu RETURN, mà bạn không thể làm khi đặt CommandType là StoredProcedure. Đồng thời, nó làm cho mã của các bạn dễ hiểu hơn khi bạn sử dụng EXECUTE Command.

Có một đôi cách mà bạn có thể thực hiện một thủ tục lưu trữ (stored procedure) phụ thuộc vào thủ tục của các bạn có trả lại một tập hợp kết quả hay không. ( một tập hợp kết quả là một hoặc nhiều hàng được truy xuất từ một bảng bởi một phát biểu SELECT ). Bạn sẽ học hai cách này để thực hiện một thủ tục lưu trữ tiếp theo đây.

## **THỰC THI MỘT THỦ TỤC KỮU TRỮ (Stored Procedure) MÀ KHÔNG TRẢ LẠI MỘT TẬP HỢP KẾT QUẢ:**

*Nếu thủ tục của các bạn không trả lại một tập hợp kết quả, thì bạn sử dụng những bước sau đây để thực hiện nó:*

1. Tạo một đối tượng Command và gán thuộc tính CommandText với một phát biểu thực thi (EXECUTE) chứa lệnh gọi Procedure của bạn.
2. Thêm bất kỳ tham số được yêu cầu nào cho lệnh gọi thủ tục vào đối tượng Lệnh (Command) của bạn,
3. Thực thi đối tượng Command của các bạn sử dụng phương thức ExecuteNonQuery() .
4. Đọc những giá trị của bất kỳ tham số đầu ra nào.

Bạn sẽ thấy cách sử dụng bốn bước này để gọi những thủ tục lưu trữ SQL Sever sau đây như thế nào:

- ♦ Thủ tục đầu tiên, AddProduct(), sẽ trả lại một tham số đầu ra được định nghĩa sử dụng từ khóa OUTPUT.
- ♦ Thủ tục thứ hai, AddProduct2(), Sẽ trả về một tham số đầu ra sử dụng lệnh RETURN.

Những ví dụ này sẽ cho bạn thấy những cách có thể để thực hiện một thủ tục lưu trữ sử dụng ADO.NET và đọc những tham số đầu ra.

## **THỰC THI THỦ TỤC STORED PROCEDURE *AddProduct()*:**

Trong Chương 4, bạn đã thấy cách tạo ra một thủ tục lưu trữ (Stored Procedure) trong cơ sở dữ liệu Northwind SQL Server. Thủ tục bạn nhìn thấy có tên AddProduct(), và Danh sách 8.11 trình bày AddProduct.sql Nguyên bản, điều đó tạo ra thủ tục AddProduct(). Bạn đã thấy cách chạy nguyên bản (Script) này như thế nào Trong Chương 4. Nếu bạn đã chưa chạy nguyên bản (Script) này khi đọc Chương 4, và bạn muốn chạy ví dụ chương trình C# được trình bày sau, bạn sẽ cần chạy nguyên bản này. AddProduct() Thêm một hàng vào bảng Products và trả lại ProductID của hàng mới như một tham số Đầu ra.

### **Danh sách 8.11: ADDPRODUCT.SQL**

/\*

AddProduct.sql Tạo ra một thủ tục để thêm một hàng vào bảng Products sử dụng những giá trị gởi qua như những tham số Tới thủ tục. Thủ tục trả lại ProductID của hàng mới trong một tham số Đầu ra có tên @MyProductID

\*/

```
CREATE PROCEDURE AddProduct
    @MyProductID int OUTPUT,
    @MyProductName nvarchar(40),
    @MySupplierID int,
    @MyCategoryID int,
    @MyQuantityPerUnit nvarchar(20),
    @MyUnitPrice money,
    @MyUnitsInStock smallint,
    @MyUnitsOnOrder smallint,
    @MyReorderLevel smallint,
    @MyDiscontinued bit
```

AS

- thêm một hàng vào bảng Product

```
INSERT INTO Products (
    ProductName, SupplierID, CategoryID, QuantityPerUnit,
    UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel,
    Discontinued
) VALUES (
    @MyProductName, @MySupplierID, @MyCategoryID, @MyQuantityPerUnit,
    @MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, @MyReorderLevel,
    @MyDiscontinued
)
```

- sử dụng hàm SCOPE\_IDENTITY() để có giá trị khóa chính sau cùng, được chèn vào trong - - một bảng được thực hiện bên trong

- bộ phận cơ sở dữ liệu hiện thời và thủ tục lưu trữ,

- vì thế SCOPE\_IDENTITY trả lại ProductID cho hàng mới trong bảng Products trong trường hợp này

```
SELECT @MyProductID = SCOPE_IDENTITY()
```

Chú ý tham số Đầu ra có tên @MyProductID được trả về bởi thủ tục lưu trữ AddProduct(). Vì AddProduct() Không trả lại một tập hợp kết quả, bạn sử dụng tập hợp đầu tiên của những bước được phác thảo trước đó.

Chúng ta hãy khảo sát những chi tiết của bốn bước này để thực thi thủ tục lưu trữ này.

## **BUỐC 1: TẠO MỘT ĐỐI TƯỢNG COMMAND VÀ GÁN THUỘC TÍNH CommandText CỦA NÓ VỚI PHÁT BIỂU EXECUTE**

Bước đầu tiên của bạn là tạo ra một đối tượng Command và đặt thuộc tính CommandText của nó với một phát biểu EXECUTE có chứa lệnh gọi tới thủ tục lưu trữ AddProduct(); chú ý tham số địa chỉ placeholders thường đánh dấu vị trí nơi những giá trị tham số sẽ được thay thế trong bước 2

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
mySqlCommand.CommandText =  
"EXECUTE AddProduct @MyProductID OUTPUT, @MyProductName, " +  
"@MySupplierID, @MyCategoryID, @MyQuantityPerUnit, " +  
"@MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, " +  
"@MyReorderLevel, @MyDiscontinued";
```

Chú ý tham số địa chỉ (placeholder) đầu ra có tên @MyProductID. nó được dùng để cất giữ tham số đầu ra được trả về bởi thủ tục lưu trữ AddProduct(), những tham số địa chỉ (placeholders) khác được dùng để gửi những giá trị tới thủ tục lưu trữ AddProduct(), và thủ tục này sẽ sử dụng những giá trị đó trong phát biểu INSERT .

## **BUỐC 2: THÊM BẤT KỲ THAM SỐ CẦN THIẾT NÀO VÀ ĐỐI TƯỢNG COMMAND:**

Bước thứ hai của bạn là thêm bất kỳ tham số nào vào đối tượng Command của các bạn, nhớ đặt thuộc tính phương hướng (Direction) cho bất kỳ tham số đầu ra nào thành ParameterDirection.Output.

Trong ví dụ này, thủ tục lưu trữ AddProduct() chờ đợi một tham số đầu ra để cất giữ một giá trị ProductID cho hàng mới, và do đó bạn cần thêm một tham số đầu ra vào đối tượng Lệnh (Command) của bạn. Bạn thực hiện điều này bởi sự thiết đặt thuộc tính phương hướng Direction của tham số của bạn thành ParameterDirection.Output. thí dụ :

```
mySqlCommand.Parameters.Add("@MyProductID", SqlDbType.Int);  
mySqlCommand.Parameters["@MyProductID"].Direction =  
ParameterDirection.Output;
```

Những tham số khác được yêu cầu để gọi thủ tục lưu trữ AddProduct() :

```
mySqlCommand.Parameters.Add(  
"@MyProductName", SqlDbType.NVarChar, 40).Value = "Widget";  
mySqlCommand.Parameters.Add(  
"@MySupplierID", SqlDbType.Int).Value = 1;  
mySqlCommand.Parameters.Add(  
"@MyCategoryID", SqlDbType.Int).Value = 1;  
mySqlCommand.Parameters.Add(  
"@MyQuantityPerUnit", SqlDbType.NVarChar, 20).Value = "1 per box";  
mySqlCommand.Parameters.Add(  
"@MyUnitPrice", SqlDbType.Money).Value = 5.99;  
mySqlCommand.Parameters.Add(  
"@MyUnitsInStock", SqlDbType.SmallInt).Value = 10;  
mySqlCommand.Parameters.Add(  
"@MyUnitsOnOrder", SqlDbType.SmallInt).Value = 5;
```

```
mySqlCommand.Parameters.Add(
"@MyReorderLevel", SqlDbType.SmallInt).Value = 5;
mySqlCommand.Parameters.Add(
"@MyDiscontinued", SqlDbType.Bit).Value = 1;
```

Chú ý những kiểu tham số SqlDbType tương ứng tới những kiểu được chờ đợi bởi thủ tục lưu trữ AddProduct(). Những giá trị, những tham số được đặt ra để rồi được thế cho placeholders trong thực thi phát biểu được trình bày trong bước 1.

### **BUƯỚC 3: THỰC THI ĐỐI TƯỢNG LỆNH(COMMAND) SỬ DỤNG PHƯƠNG THỨC ExecuteNonQuery()**

Bước thứ ba của bạn là thực thi đối tượng Lệnh (Command) của bạn sử dụng phương thức ExecuteNonQuery(). Bạn sử dụng ExecuteNonQuery() Vì Thủ tục AddProduct() không trả lại một tập hợp kết quả. Chẳng hạn:

```
mySqlCommand.ExecuteNonQuery();
```

### **BUƯỚC 4: ĐỌC NHỮNG GIÁ TRỊ CỦA BẤT KỲ THAM SỐ ĐẦU RA NÀO:**

Bước cuối cùng của bạn là đọc những giá trị của bất kỳ tham số đầu ra nào. thủ tục lưu trữ AddProduct() sử dụng một tham số đầu ra có tên @MyProductID. Bạn đọc giá trị trả về này từ thuộc tính Value của @MyProductID:

```
Console.WriteLine("New ProductID = " +
mySqlCommand.Parameters["@MyProductID"].Value);
```

mã lệnh này sẽ hiển thị những giá trị của ProductID phát sinh bởi SQL Server cho hàng mới trong bảng Products.

**Danh sách 8.12 minh họa những bước để gọi thủ tục lưu trữ AddProduct() này .**

#### **Danh sách 8.12: EXECUTEADDPRODUCT.CS**

```
/*
ExecuteAddProduct.cs illustrates how to call the SQL Server
AddProduct() stored procedure
*/

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

class ExecuteAddProduct
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");
        mySqlConnection.Open();

        // step 1: create a Command object and set its CommandText
        // property to an EXECUTE statement containing the stored
        // procedure call
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

```

mySqlCommand.CommandText =
"EXECUTE AddProduct @MyProductID OUTPUT, @MyProductName, " +
"@MySupplierID, @MyCategoryID, @MyQuantityPerUnit, " +
"@MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, " +
"@MyReorderLevel, @MyDiscontinued";

// step 2: add the required parameters to the Command object
mySqlCommand.Parameters.Add("@MyProductID", SqlDbType.Int);
mySqlCommand.Parameters["@MyProductID"].Direction =
ParameterDirection.Output;
mySqlCommand.Parameters.Add(
"@MyProductName", SqlDbType.NVarChar, 40).Value = "Widget";
mySqlCommand.Parameters.Add(
"@MySupplierID", SqlDbType.Int).Value = 1;
mySqlCommand.Parameters.Add(
"@MyCategoryID", SqlDbType.Int).Value = 1;
mySqlCommand.Parameters.Add(
"@MyQuantityPerUnit", SqlDbType.NVarChar, 20).Value = "1 per box";
mySqlCommand.Parameters.Add(
"@MyUnitPrice", SqlDbType.Money).Value = 5.99;
mySqlCommand.Parameters.Add(
"@MyUnitsInStock", SqlDbType.SmallInt).Value = 10;
mySqlCommand.Parameters.Add(
"@MyUnitsOnOrder", SqlDbType.SmallInt).Value = 5;
mySqlCommand.Parameters.Add(
"@MyReorderLevel", SqlDbType.SmallInt).Value = 5;
mySqlCommand.Parameters.Add(
"@MyDiscontinued", SqlDbType.Bit).Value = 1;

// step 3: execute the Command object using the
// ExecuteNonQuery() method
mySqlCommand.ExecuteNonQuery();

// step 4: read the value of the output parameter
Console.WriteLine("New ProductID = " +
mySqlCommand.Parameters["@MyProductID"].Value);

mySqlConnection.Close();
}
}

```

### Đầu ra từ chương trình này Như sau:

New ProductID = 81

Tất nhiên, phụ thuộc vào những hàng hiện hữu trong bảng Products của các bạn, bạn sẽ có một kết quả khác.

### **THỰC THI THỦ TỤC LƯU TRỮ *AddProduct2()***

Như bạn sẽ thấy, Thủ tục lưu trữ AddProduct2() cũng tương tự như AddProduct() chỉ có điều nó sử dụng một phát biểu RETURN thay vì một tham số Đầu ra để trả về ProductID cho hàng mới. Danh sách 8.13 trình bày Nguyên bản (Script) AddProduct2.sql nó tạo ra thủ tục AddProduct2(). Bạn sẽ cần chạy nguyên bản (Script) này trước khi chạy chương trình C#.

### **Danh sách 8.13: ADDPRODUCT2. SQL**



/\*

AddProduct2.sql Tạo ra một thủ tục để thêm một hàng vào bảng Products sử dụng những giá trị gởi qua như những tham số tới thủ tục. Thủ tục trả về ProductID của hàng mới sử dụng một phát biểu RETURN.

\*/

```
CREATE PROCEDURE AddProduct2
    @MyProductName nvarchar(40),
    @MySupplierID int,
    @MyCategoryID int,
    @MyQuantityPerUnit nvarchar(20),
    @MyUnitPrice money,
    @MyUnitsInStock smallint,
    @MyUnitsOnOrder smallint,
    @MyReorderLevel smallint,
    @MyDiscontinued bit
```

AS

- khai báo biến @MyProductID  
DECLARE @MyProductID int

- chèn một hàng vào trong bảng sản phẩm

```
INSERT INTO Products (
    ProductName, SupplierID, CategoryID, QuantityPerUnit,
    UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel,
    Discontinued
) VALUES (
    @MyProductName, @MySupplierID, @MyCategoryID, @MyQuantityPerUnit,
    @MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, @MyReorderLevel,
    @MyDiscontinued )
```

- sử dụng hàm SCOPE\_IDENTITY() để lấy giá trị khóa chính sau cùng được chèn vào trong  
- một bảng được thực hiện bên trong bộ phận cơ sở dữ liệu hiện thời và thủ tục lưu trữ,  
- vì vậy SCOPE\_IDENTITY trả về ProductID cho hàng mới  
- trong bảng Products trong trường hợp này

```
SET @MyProductID = SCOPE_IDENTITY()
```

```
RETURN @MyProductID
```

Chú ý phát biểu RETURN ở cuối để trả về @MyProductID. Vì AddProduct2() không trả về một tập hợp kết quả của những hàng, bạn sử dụng tương tự như bốn bước được trình bày trong mục trước để thực hiện thủ tục lưu trữ sử dụng ADO.NET. Sự khác nhau duy nhất trong cấu trúc của lệnh EXECUTE của bạn là đặt thuộc tính CommandText trong bước 1. Để gọi AddProduct2() bạn đặt thuộc tính CommandText của đối tượng Lệnh(command) của bạn như sau:

```
mySqlCommand.CommandText =
    "EXECUTE @MyProductID = AddProduct2 @MyProductName, " +
    "@MySupplierID, @MyCategoryID, @MyQuantityPerUnit, " +
    "@MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, " +
    "@MyReorderLevel, @MyDiscontinued";
```

Chú ý đến sự thay đổi vị trí của tham số @MyProductID: nó được chuyển đến ngay sau từ khóa EXECUTE và đặt dấu bằng tới giá trị trả về bởi AddProduct2(). Sự thay đổi này được làm vì AddProduct2() sử dụng một phát biểu RETURN để xuất trả về giá trị @MyProductID. Phần còn lại của mã C# được yêu cầu để gọi



AddProduct2() cũng tương tự như đã trình bày trước đó trong Danh sách 8.12.

Ghi chú vì chỉ EXECUTE là khác nhau, Tôi đã bỏ qua chương trình gọi AddProduct2() trong sách này. Bạn có thể thấy chương trình này trong File ExecuteAddProduct2.cs mà tôi đã cung cấp. hãy thoải mái chạy thử và khảo sát nó.

## **THỰC THI MỘT THỦ TỤC LƯU TRỮ (Stored Procedure) MÀ TRẢ VỀ MỘT TẬP HỢP KẾT QUẢ:**

Nếu thủ tục của bạn thuộc loại trả về một tập kết quả, thì bạn sử dụng những bước sau để thực thi:

1. Tạo ra một đối tượng Command và gán thuộc tính CommandText của nó tới một phát biểu EXECUTE có chứa hàm gọi thủ tục của bạn.
2. thêm bất kỳ tham số yêu cầu nào vào đối tượng Command của bạn, nhớ gán thuộc tính Direction cho bất kỳ tham số đầu ra nào với ParameterDirection.Output.
3. thực thi Command của bạn sử dụng phương thức ExecuteReader(), lưu giữ đối tượng DataReader được trả về.
4. đọc những hàng trong tập hợp kết quả sử dụng đối tượng DataReader của bạn.
5. đóng đối tượng DataReader của bạn. bạn phải thực hiện điều này trước khi bạn có thể đọc bất kỳ tham số đầu ra nào.
6. đọc các tham số đầu ra (output parameters).

Trong ví dụ sau đây, bạn sẽ thấy một thủ tục lưu trữ có tên AddProduct3() nó sẽ trả lại một tập hợp kết quả cùng với một tham số đầu ra sử dụng một phát biểu RETURN.

Thủ tục AddProduct3() cũng tương tự như AddProduct2(), ngoại trừ nó cũng trả lại một tập hợp kết quả sử dụng một phát biểu SELECT. phát biểu SELECT này chứa những cột ProductName và UnitPrice cho hàng mới thêm vào bảng Products. Ngoài ra tập hợp kết quả này còn trả về ProductID của hàng mới bởi phát biểu RETURN, Danh sách 8.14 trình bày Nguyên bản (Script) AddProduct3.sql và script này tạo ra Thủ tục AddProduct3() . Bạn cần chạy nguyên bản này trước khi chạy được chương trình C#.

### **Danh sách 8.14: ADDPRODUCT3. SQL**

```
/*
AddProduct3.sql creates a procedure that adds a row to the
Products table using values passed as parameters to the
procedure. The procedure returns the ProductID of the new row
using a RETURN statement and returns a result set containing
the new row
*/

CREATE PROCEDURE AddProduct3
    @MyProductName nvarchar(40),
    @MySupplierID int,
    @MyCategoryID int,
    @MyQuantityPerUnit nvarchar(20),
    @MyUnitPrice money,
    @MyUnitsInStock smallint,
    @MyUnitsOnOrder smallint,
    @MyReorderLevel smallint,
    @MyDiscontinued bit
AS

    - declare the @MyProductID variable
    DECLARE @MyProductID int

    - insert a row into the Products table
```

```

INSERT INTO Products (
    ProductName, SupplierID, CategoryID, QuantityPerUnit,
    UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel,
    Discontinued
) VALUES (
    @MyProductName, @MySupplierID, @MyCategoryID, @MyQuantityPerUnit,
    @MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, @MyReorderLevel,
    @MyDiscontinued
)

```

- use the SCOPE\_IDENTITY() function to get the last
- identity value inserted into a table performed within
- the current database session and stored procedure,
- so SCOPE\_IDENTITY returns the ProductID for the new row
- in the Products table in this case

```

SET @MyProductID = SCOPE_IDENTITY()

```

- return the result set

```

SELECT ProductName, UnitPrice
FROM Products
WHERE ProductID = @MyProductID

```

- return @MyProductID

```

RETURN @MyProductID

```

Do bạn đã thấy những cơ sở cho mã thực thi sáu bước được trình bày trước trong mục này, Tôi sẽ đi thẳng tới mã với giải thích sơ lược. Danh sách 8.15 trình bày chương trình gọi AddProduct3(). Những thứ quan trọng cần chú ý :

### **Danh sách 8.15: EXECUTEADDPRODUCT3. CS**

```

/*
ExecuteAddProduct3.cs illustrates how to call the SQL Server
AddProduct3() stored procedure
*/

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

class ExecuteAddProduct3
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");
        mySqlConnection.Open();

        // step 1: create a Command object and set its CommandText
        // property to an EXECUTE statement containing the stored
        // procedure call
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "EXECUTE @MyProductID = AddProduct3 @MyProductName, " +
            "@MySupplierID, @MyCategoryID, @MyQuantityPerUnit, " +

```

```

"@MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, " +
"@MyReorderLevel, @MyDiscontinued";

// step 2: add the required parameters to the Command object
mySqlCommand.Parameters.Add("@MyProductID", SqlDbType.Int);
mySqlCommand.Parameters["@MyProductID"].Direction =
ParameterDirection.Output;
mySqlCommand.Parameters.Add(
"@MyProductName", SqlDbType.NVarChar, 40).Value = "Widget";
mySqlCommand.Parameters.Add(
"@MySupplierID", SqlDbType.Int).Value = 1;
mySqlCommand.Parameters.Add(
"@MyCategoryID", SqlDbType.Int).Value = 1;
mySqlCommand.Parameters.Add(
"@MyQuantityPerUnit", SqlDbType.NVarChar, 20).Value = "1 per box";
mySqlCommand.Parameters.Add(
"@MyUnitPrice", SqlDbType.Money).Value = 5.99;
mySqlCommand.Parameters.Add(
"@MyUnitsInStock", SqlDbType.SmallInt).Value = 10;
mySqlCommand.Parameters.Add(
"@MyUnitsOnOrder", SqlDbType.SmallInt).Value = 5;
mySqlCommand.Parameters.Add(
"@MyReorderLevel", SqlDbType.SmallInt).Value = 5;
mySqlCommand.Parameters.Add(
"@MyDiscontinued", SqlDbType.Bit).Value = 1;

// step 3: execute the Command object using the ExecuteReader()
// method
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

// step 4: read the rows using the DataReader object
while (mySqlDataReader.Read())
{
    Console.WriteLine("mySqlDataReader[\" ProductName\"] = " +
mySqlDataReader["ProductName"]);
    Console.WriteLine("mySqlDataReader[\" UnitPrice\"] = " +
mySqlDataReader["UnitPrice"]);
}

// step 5: close the DataReader object
mySqlDataReader.Close();

// step 6: read the value of the output parameter
Console.WriteLine("New ProductID = " +
mySqlCommand.Parameters["@MyProductID"].Value);

mySqlConnection.Close();
}
}

```

Phương thức ExecuteReader() được dùng để trả lại tập hợp kết quả chứa những cột ProductName và UnitPrice cho hàng mới.

Tập hợp kết quả rồi sẽ được đọc sử dụng một đối tượng SqlDataReader.

Đối tượng SqlDataReader được đóng trước khi có thể đọc được tham số đầu ra.

**Đầu ra từ chương trình này như sau:**

```
mySqlDataReader["ProductName"] = Widget  
mySqlDataReader["UnitPrice"] = 5.99  
New ProductID = 83
```

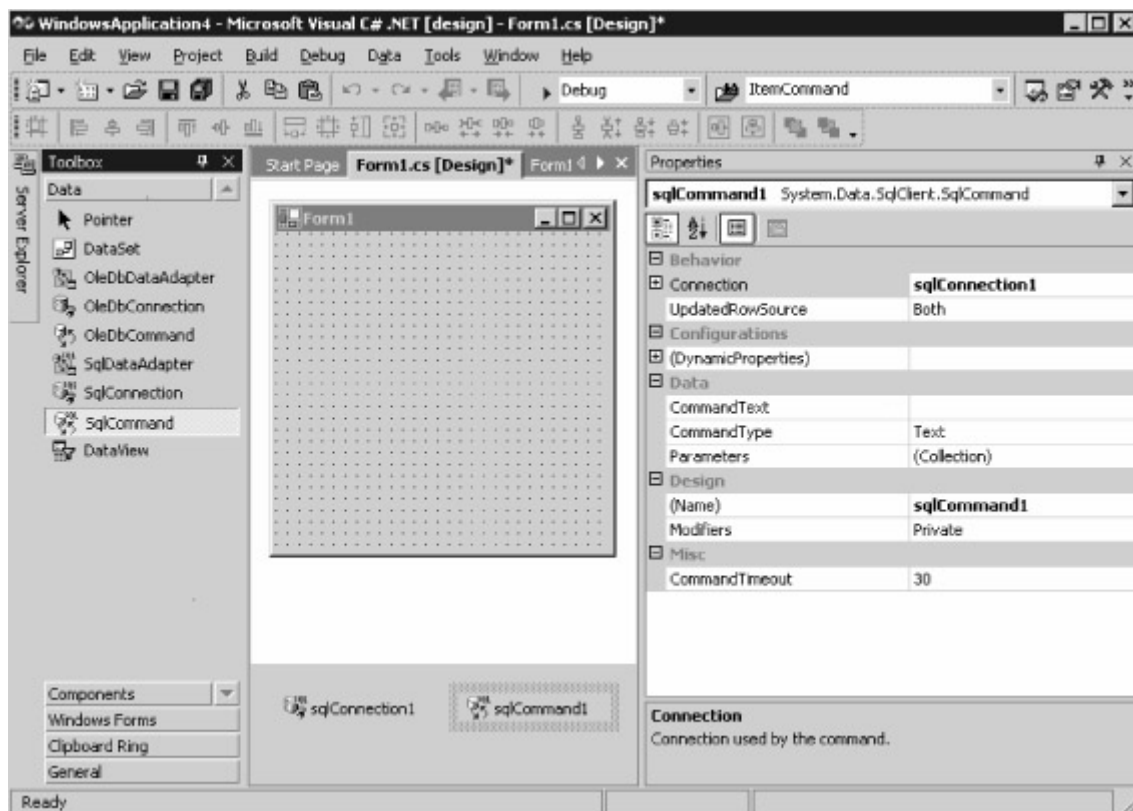
## **TAO MỘT ĐỐI TƯỢNG COMMAND SỬ DỤNG VISUAL STUDIO.NET**

Để tạo ra một đối tượng SqlCommand sử dụng Visual Studio.NET (VS.NET), bạn kéo một đối tượng SqlCommand từ tab Data của Toolbox vào form bạn. Bạn có thể cũng kéo một đối tượng OleDbCommand từ tab Data của Toolbox đến form của bạn.

Trước khi bạn thực hiện thủ tục được giải thích trong mục này, thực hiện các bước sau:

1. tạo một chương trình mới tên MyDataReader chứa một trình ứng dụng Windows.
2. thêm một đối tượng SqlConnection vào chương trình (quay trở lại chương trước đây để xem cách thêm một đối tượng SqlConnection sử dụng VS.NET) đối tượng này sẽ có tên mặc định là SqlConnection1.
3. Định hình đối tượng sqlCommand1 của bạn để truy nhập cơ sở dữ liệu Northwind của bạn.

Kéo một đối tượng SqlCommand vào form của bạn. Hình 8.1 cho thấy một form với một đối tượng SqlCommand. Đối tượng này được gán tên mặc định là sqlCommand1.



**Hình 8.1: Một đối tượng SqlCommand trong Một form**

sau đó Bạn đặt thuộc tính Kết nối (Connection ) cho sqlCommand1 sử dụng danh sách thả xuống (dropdown list) ở bên phải của thuộc tính connection trong cửa sổ thuộc tính (property window). Bạn có thể lựa chọn một đối tượng kết nối(Connection) hiện hữu trong danh sách thả xuống; bạn có thể cũng tạo ra một đối tượng Connection mới bởi lựa chọn New từ danh sách. Cho ví dụ này, chọn đối tượng sqlConnection1 hiện hữu cho thuộc tính Connection của đối tượng sqlCommand1, như trình bày trong hình 8.1

Bạn có thể sử dụng trình tạo truy vấn (Query Builder) để tạo một câu lệnh SQL bởi việc kích vào nút "..." ở bên phải của thuộc tính CommandText, và bạn có thể gán những tham số cho một lệnh (Command) bởi việc kích nút "..." ở bên trái của những thuộc tính tham số. Bạn sẽ gán thuộc tính CommandText của đối tượng SqlCommand của bạn đến một phát biểu SELECT truy xuất các cột CustomerID, CompanyName, and

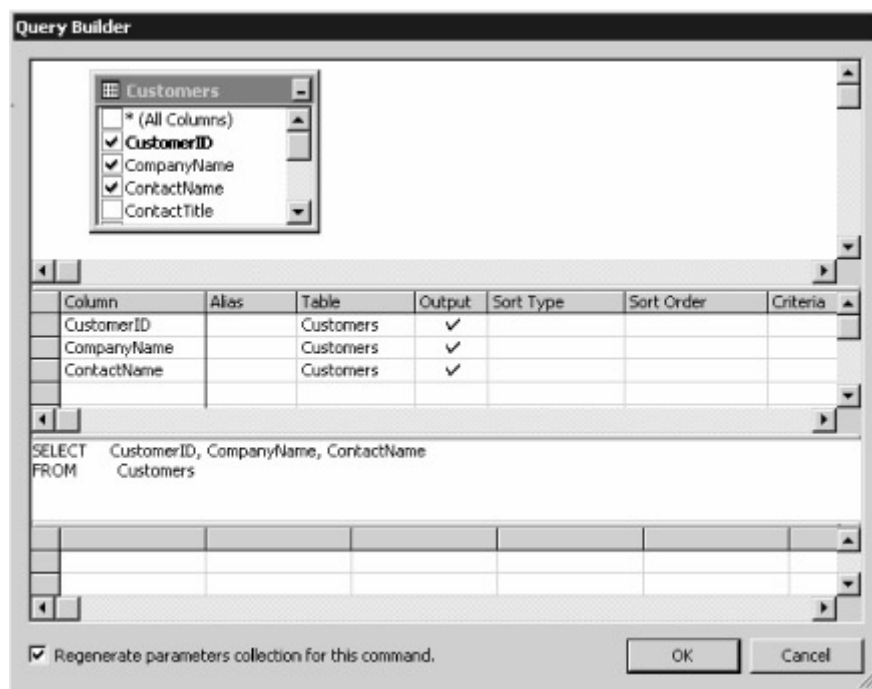
ContactName từ bảng Customers, Bạn sẽ xây dựng phát biểu SELECT này sử dụng trình tạo Query. Để bắt đầu, kích nút "..." ở bên phải của thuộc tính CommandText cho đối tượng SqlCommand của các bạn.

Trong hộp thoại Add Table, chọn bảng Customers, như trình bày trong hình 8.2. click nút Add để thêm bảng Customers vào query của bạn. click nút close để tiếp tục.



Hình 8.2: thêm bảng Customers vào truy vấn sử dụng hộp thoại Add Table

Tiếp theo, bạn xây dựng truy vấn của bạn sử dụng trình xây dựng truy vấn (Query Builder). Bạn lựa chọn những cột Bạn muốn truy xuất ở đây. Thêm những cột CustomerID, CompanyName, và ContactName sử dụng trình xây dựng truy vấn (Query Builder), như Hình 8.3.



Hình 8.3: Thêm những cột CustomerID, CompanyName, và ContactName vào truy vấn sử dụng Query Builder.

Kích nút Ok để tiếp tục. Thuộc tính CommandText của đối tượng SqlCommand của bạn đã được gán bằng phát biểu SELECT bạn tạo ra trong Query Builder.

Ghi chú lưu dự án MyDataReader của bạn bằng cách chọn File - Save All. Bạn sẽ thấy sự sử dụng đối tượng SqlCommand Bạn thêm vào dự án của bạn trong chương kế tiếp.

## **TÓM TẮT:**

Trong chương này, bạn đã học cách thực hiện những lệnh cơ sở dữ liệu như thế nào. Có ba lớp Lệnh SqlCommand, OleDbCommand, Và OdbcCommand. Bạn sử dụng một đối tượng Command để thực hiện một phát biểu SQL SELECT, INSERT, UPDATE, hay DELETE. Bạn có thể cũng sử dụng một đối tượng Command để thực hiện một sự gọi thủ tục lưu trữ( Stored Procedure), hay truy xuất tất cả những hàng và những cột từ một bảng cụ thể; nó được biết như một TableDirect Command. Bạn sử dụng một đối tượng của lớp SqlCommand để thực hiện một lệnh truy nhập một cơ sở dữ liệu SQL Server, một đối tượng của lớp SqlDataReader để đọc những hàng truy xuất được từ một cơ sở dữ liệu SQL Server, và một đối tượng của lớp SqlTransaction để đại diện cho một giao dịch cơ sở dữ liệu trong một cơ sở dữ liệu SQL Server.

Trong chương kế tiếp, bạn sẽ học những chi tiết của những đối tượng DataReader.

## **CHƯƠNG 9: SỬ DỤNG NHỮNG ĐỐI TƯỢNG DATAREADER ĐỂ ĐỌC NHỮNG KẾT QUẢ**

### **TỔNG QUAN:**

Đọc những hàng sử dụng một đối tượng DataReader (đôi khi được biết đến như một con trỏ firehose) diễn hình nhanh chóng hơn đọc từ một Dataset. Những đối tượng DataReader là bộ phận của những trình cung cấp được quản lý, và có ba lớp DataReader: SqlDataReader, OleDbDataReader, và OdbcDataReader. Bạn sử dụng một đối tượng DataReader để đọc những hàng truy xuất được từ cơ sở dữ liệu sử dụng một đối tượng Command.

Những đối tượng DataReader có thể được sử dụng để đọc những hàng chỉ theo một hướng xuôi. Những đối tượng DataReader đóng vai như một giải pháp đối với một đối tượng Dataset (Những đối tượng Dataset cho phép bạn lưu trữ một bản sao của những hàng từ cơ sở dữ liệu, và bạn có thể làm việc với bản sao này trong khi ngắt kết nối với cơ sở dữ liệu) Bạn không thể sử dụng một DataReader để sửa đổi những hàng trong cơ sở dữ liệu.

### **Những điểm chính trong chương này:**

- Lớp SqlDataReader
- Tạo một đối tượng SqlDataReader
- Đọc những hàng từ một đối tượng SqlDataReader
- Trả về những giá trị định kiểu mạnh của cột
- Đọc những giá trị Null
- Thực hiện nhiều câu lệnh SQL
- Sử dụng một đối tượng DataReader trong Visual Studio.NET

## **LỚP SQLDATAREADER:**

Bạn sử dụng một đối tượng của lớp SqlDataReader để đọc những hàng được truy xuất từ một cơ sở dữ liệu máy chủ phục vụ SQL, một đối tượng của lớp OleDbDataReader để đọc những hàng từ bất kỳ cơ sở dữ liệu nào hỗ trợ OLE DB, như Oracle hay Access, và một đối tượng của lớp OdbcDataReader để đọc những hàng từ bất kỳ cơ sở dữ liệu nào hỗ trợ ODBC. Bảng 9.1 cho thấy một số thuộc tính SqlDataReader.

**Bảng 9.1: những thuộc tính SqlDataReader**

Thuộc tính	Kiểu	Mô tả
Depth	int	Giữ một giá trị chỉ định chiều sâu của sự lồng nhau cho hàng hiện thời.
FieldCount	int	Giữ số cột trong hàng hiện thời.



Thuộc tính	Kiểu	Mô tả
IsClosed	bool	Giữ một giá trị bool chỉ định liệu đối tượng đọc dữ liệu đã được đóng.
RecordsAffected	int	Giữ số lượng hàng đã được thêm vào, được sửa đổi, hay được loại bỏ bởi sự thực hiện của câu lệnh SQL.

Ghi chú mặc đầu lớp SqlDataReader được dành riêng cho SQL Server, rất nhiều thuộc tính và những phương thức trong lớp này cũng tương tự như cây trong lớp OleDbDataReader và lớp OdbcDataReader.

**Bảng 9.2 cho thấy một số phương thức dùng chung trong SqlDataReader .**

Bảng 9.2: những phương thức của đt SqlDataReader

Phương thức	Giá trị trả về	Mô tả
GetBoolean()	bool	Trả lại giá trị của cột được chỉ rõ như một giá trị bool.
GetByte()	byte	Trả lại giá trị của cột được chỉ rõ như một Byte.
GetBytes()	long	Đọc một luồng của những giá trị byte từ cột được chỉ định vào trong một mảng byte. Giá trị dài được trả lại là số của những giá trị byte đọc được từ cột.
GetChar()	char	Trả lại giá trị của cột được chỉ rõ như một char.
GetChars()	long	đọc một luồng của những giá trị char từ cột được chỉ định vào trong một mảng char. Giá trị dài được trả lại là số của những giá trị char đọc từ cột.
GetDataTypeName()	string	trả về tên của kiểu dữ liệu nguồn cho cột được chỉ định.
GetDateTime()	DateTime	Trả lại giá trị của cột được chỉ rõ là một kiểu DateTime.
GetDecimal()	decimal	Trả lại giá trị của cột được chỉ định như một số thập phân.
GetDouble()	double	Trả lại giá trị của cột được chỉ rõ như một double.
GetFieldType()	Type	Trả lại kiểu của cột được chỉ định.
GetFloat()	float	Trả lại giá trị của cột được chỉ định như một float.
GetGuid()	Guid	trả về giá trị của cột được chỉ định như một khóa chính duy nhất toàn cục (GUID).
GetInt16()	short	Trả lại giá trị của cột được chỉ rõ như một short.
GetInt32()	int	Trả lại giá trị của cột được chỉ rõ như một int.
GetInt64()	long	Trả lại giá trị của cột được chỉ định như một long.
GetName()	string	Trả lại tên của cột được chỉ rõ.
GetOrdinal()	int	Trả lại vị trí số, hay số thứ tự, của cột được chỉ định (cột đầu tiên có một số thứ tự là 0).
GetSchemaTable()	DataTable	Trả lại một DataTable chứa đựng những chi tiết của những cột được cất giữ trong đối tượng đọc dữ liệu.
GetSqlBinary()	SqlBinary	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlBinary. Lớp SqlBinary được khai báo trong không gian tên (namespace) System.Data.SqlTypes .  Tất cả những phương thức GetSql* được chỉ định cho lớp SqlDataReader.
GetSqlBoolean()	SqlBoolean	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlBoolean.
GetSqlByte()	SqlByte	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlByte.
GetSqlDateTime()	SqlDateTime	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlDateTime.
GetSqlDecimal()	SqlDecimal	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlDecimal.

Phương thức	Giá trị trả về	Mô tả
GetSqlDouble()	SqlDouble	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlDouble.
GetSqlGuid()	SqlGuid	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlGuid.
GetSqlInt16()	SqlInt16	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlInt16.
GetSqlInt32()	SqlInt32	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlInt32.
GetSqlInt64()	SqlInt64	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlInt64.
GetSqlMoney()	SqlMoney	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlMoney.
GetSqlSingle()	SqlSingle	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlSingle.
GetSqlString()	SqlString	Trả lại giá trị của cột được chỉ rõ như một đối tượng SqlString.
GetSqlValue()	object	Trả lại giá trị của cột được chỉ rõ như một đối tượng.
GetSqlValues()	int	Sao chép giá trị của tất cả những cột trong hàng hiện thời vào trong một đối tượng mảng được chỉ định. Trị số Int trả về bởi phương thức này là số phần tử có trong mảng.
GetString()	string	Trả lại giá trị của cột được chỉ rõ như một chuỗi.
GetValue()	object	Trả lại giá trị của cột được chỉ rõ như một đối tượng.
GetValues()	int	Sao chép giá trị của tất cả những cột trong hàng hiện thời vào trong một đối tượng mảng được chỉ định. Trị số Int trả về bởi phương thức này là số phần tử có trong mảng.
IsDBNull()	bool	Trả lại một giá trị Bool cho biết liệu có phải cột chỉ định chứa một giá trị null.
NextResult()	bool	Di chuyển đối tượng đọc dữ liệu tới hàng kế tiếp trong tập hợp kết quả. Giá trị Bool trả về bởi phương thức này cho biết liệu có phải còn có hàng tiếp trong tập hợp kết quả.
Read()	bool	Di chuyển đối tượng đọc dữ liệu tới hàng kế tiếp trong tập hợp kết quả và đọc hàng. Giá trị Bool trả về bởi phương thức này cho biết liệu có phải còn có nhiều hàng hơn trong tập hợp kết quả.

Ghi nhớ những chi tiết cột DataTable bao gồm tên (tên cột được lưu trữ trong ColumnName của DataTable được trả lại), số thứ tự (được cất giữ ở ColumnOrdinal), chiều dài cực đại của giá trị được cất giữ trong cột (được cất giữ trong ColumnSize), sự chính xác và quy mô của một cột số (được cất giữ trong NumericPrecision và NumericScale), ngoài ra. Sự Chính xác là số tổng của những chữ số tạo ra một số, và quy mô là số tổng của những chữ số về bên phải dấu phẩy ở số thập phân. Bạn đã thấy cách đọc một mô hình sử dụng một chương trình trong chương trước đây.

**Mẹo nhỏ:** Không gian tên System.Data.SqlTypes cung cấp những lớp cho những kiểu dữ liệu bản ngữ được dùng bên trong SQL server. Những lớp này cung cấp một giải pháp nhanh chóng hơn và an toàn hơn cho những kiểu dữ liệu khác được trả về bởi những phương thức Get\*. Sử dụng những lớp trong namespace này giúp ngăn ngừa những lỗi do sự chuyển đổi kiểu gây ra sự mất chính xác. Bởi vì những kiểu dữ liệu khác được chuyển đổi xuôi ngược tới SqlTypes phía sau, rõ ràng việc tạo ra và sử dụng những đối tượng bên trong namespace này khiến cho mã thực thi nhanh chóng hơn nữa. Bạn sẽ học nhiều hơn về namespace SqlTypes sau Trong Mục " Sử dụng những phương thức GetSql\* để Đọc những giá trị Cột ".

## **TAO MỘT ĐỐI TƯỢNG SQLDATAREADER:**

Bạn có thể tạo ra một đối tượng DataReader với cách duy nhất là gọi phương thức ExecuteReader() của đối tượng Command. Những đối tượng Command đã được trình bày trong chương trước. thí dụ, code sau đây tạo những đối tượng cần thiết và thực thi một phát biểu SELECT truy xuất năm hàng đầu tiên từ bảng Product của cơ sở dữ liệu SQL Server Northwind, lưu giữ những hàng được trả về trong một đối tượng SqlDataReader:



```

SqlConnection mySqlConnection =
new SqlConnection(
"server=localhost;database=Northwind;uid=sa;pwd=sa"
);
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice, " +
"UnitsInStock, Discontinued " +
"FROM Products " +
"ORDER BY ProductID";
mySqlConnection.Open();
SqlDataReader productsSqlDataReader =
mySqlCommand.ExecuteReader();

```

Chú ý đối tượng SqlDataReader trả về bởi phương thức ExecuteReader() được cất giữ trong đối tượng productsSqlDataReader. Bạn sẽ học cách sử dụng productsSqlDataReader như thế nào trong mục sau đây.

## **ĐỌC NHỮNG HÀNG TỪ MỘT ĐỐI TƯỢNG SQLDATAREADER**

Bạn đọc những hàng từ một đối tượng DataReader sử dụng phương pháp Read() . Phương pháp này trả lại giá trị true (Bool) khi có hàng khác để đọc, ngược lại trả về false.

Bạn có thể đọc một giá trị cột riêng lẻ trong một hàng từ một DataReader bởi sự gọi qua tên của cột trong cặp ngoặc vuông. ví dụ, để đọc cột CustomerID, bạn sử dụng productsSqlDataReader[ " ProductID"]. Bạn có thể cũng chỉ rõ cột bạn muốn có bằng cách gọi qua một giá trị số trong dấu ngoặc vuông. Chẳng hạn, productsSqlDataReader[0] Cũng trả về giá trị cột ProductID.

Mẹo nhỏ: sự khác nhau giữa hai cách đọc giá trị một cột ở sự thực hiện: sử dụng con số chỉ những vị trí cột thay vì những tên cột dẫn đến sự thực hiện nhanh chóng hơn của mã. Chúng ta hãy xem xét hai đoạn mã minh họa hai cách đọc giá trị cột này. đoạn mã đầu tiên sử dụng những tên cột để đọc giá trị cột .

```

while (productsSqlDataReader.Read())
{
    Console.WriteLine(productsSqlDataReader["ProductID"]);
    Console.WriteLine(productsSqlDataReader["ProductName"]);
    Console.WriteLine(productsSqlDataReader["UnitPrice"]);
    Console.WriteLine(productsSqlDataReader["Discontinued"]);
}

```

Mặc dù đoạn mã thứ hai nhanh chóng hơn .nhưng nó ít linh hoạt hơn vì bạn phải viết mã cứng những số cột vị trí. Nếu những vị trí cột trong phát biểu SELECT được thay đổi, bạn cần thay đổi mã cứng vị trí cột -và đây là một cơn ác mộng bảo trì. Đồng thời, sự mã hóa cứng vị trí cột làm những chương trình của các bạn khó đọc hơn.

Có một giải pháp cho vấn đề này: bạn có thể gọi phương thức GetOrdinal() của đối tượng DataReader. phương thức GetOrdinal() trả lại vị trí của một cột có tên được đưa vào như một tham số; vị trí này được biết như thứ tự cột . Bạn có thể rồi sử dụng vị trí được trả về bởi GetOrdinal() để truy xuất những giá trị cột từ DataReader của các bạn.

Chúng ta hãy xem xét đoạn mã sử dụng phương thức GetOrdinal() để truy xuất vị trí của những cột trong phát biểu ví dụ SELECT sau:

```

int productIDColPos =
productsSqlDataReader.GetOrdinal("ProductID");
int productNameColPos =

```

```

productsSqlDataReader.GetOrdinal("ProductName");
int unitPriceColPos =
productsSqlDataReader.GetOrdinal("UnitPrice");
int discontinuedColPos =
    productsSqlDataReader.GetOrdinal("Discontinued");

```

Và bạn có thể sử dụng những giá trị int này để truy xuất những giá trị cột từ đối tượng productsSqlDataReader:

```

while (productsSqlDataReader.Read())
{
    Console.WriteLine(productsSqlDataReader[productIDColPos]);
    Console.WriteLine(productsSqlDataReader[productNameColPos]);
    Console.WriteLine(productsSqlDataReader[unitPriceColPos]);
    Console.WriteLine(productsSqlDataReader[discontinuedColPos]);
}

```

Cách này cho bạn những ưu điểm của cả hai phương án trên : sự thực hiện và tính linh hoạt cao.

Cảnh báo: Khi bạn kết thúc đọc những hàng từ đối tượng DataReader của bạn, hãy đóng nó bằng phương thức Close(). Lý do là một đối tượng DataReader đang trói buộc đối tượng Kết nối này, và những lệnh khác(Command) không thể được thực hiện khi có một DataReader đang mở và giữ kết nối này( đối tượng Connection). Ví dụ sau đây đóng productsSqlDataReader sử dụng phương thức Close() .

```

productsSqlDataReader.Close();

```

Một khi bạn đã đóng DataReader của bạn, Bạn có thể thực hiện những lệnh(Command) khác sử dụng đối tượng Kết nối (Connection)của bạn.

Liệt kê 9.1 những ví dụ về sử dụng mã được trình bày trong mục này.

### **Danh sách 9.1: USINGCOLUMNORDINALS.CS**

```

/*
UsingColumnOrdinals.cs illustrates how to use the GetOrdinal()
method of a DataReader object to get the numeric positions of a column
*/

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

class UsingColumnOrdinals
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        mySqlCommand.CommandText =
            "SELECT TOP 5 ProductID, ProductName, UnitPrice, " +

```

```

"UnitsInStock, Discontinued " +
"FROM Products " +
"ORDER BY ProductID";

mySqlConnection.Open();

SqlDataReader productsSqlDataReader =
mySqlCommand.ExecuteReader();

// sử dụng phương thức GetOrdinal() của đối tượng DataReader
// để lấy số chỉ vị trí của cột
int productIDColPos =
productsSqlDataReader.GetOrdinal("ProductID");
int productNameColPos =
productsSqlDataReader.GetOrdinal("ProductName");
int unitPriceColPos =
productsSqlDataReader.GetOrdinal("UnitPrice");
int unitsInStockColPos =
productsSqlDataReader.GetOrdinal("UnitsInStock");
int discontinuedColPos =
productsSqlDataReader.GetOrdinal("Discontinued");

while (productsSqlDataReader.Read())
{
    Console.WriteLine("ProductID = " +
productsSqlDataReader[productIDColPos]);
    Console.WriteLine("ProductName = " +
productsSqlDataReader[productNameColPos]);
    Console.WriteLine("UnitPrice = " +
productsSqlDataReader[unitPriceColPos]);
    Console.WriteLine("UnitsInStock = " +
productsSqlDataReader[unitsInStockColPos]);
    Console.WriteLine("Discontinued = " +
productsSqlDataReader[discontinuedColPos]);
}

productsSqlDataReader.Close();
mySqlConnection.Close();
}
}

```

**Đầu ra từ chương trình này như sau:**

```

ProductID = 1
ProductName = Chai
UnitPrice = 18
UnitsInStock = 39
Discontinued = False
ProductID = 2
ProductName = Chang
UnitPrice = 19
UnitsInStock = 17
Discontinued = False
ProductID = 3
ProductName = Aniseed Syrup
UnitPrice = 10
UnitsInStock = 13
Discontinued = False

```

```
ProductID = 4
ProductName = Chef Anton's Cajun Seasoning
UnitPrice = 22
UnitsInStock = 53
Discontinued = False
ProductID = 5
ProductName = Chef Anton's Gumbo Mix
UnitPrice = 21.35
UnitsInStock = 0
Discontinued = True
```

## **TRẢ VỀ NHỮNG GIÁ TRỊ CỘT ĐỊNH KIỂU MẠNH:**

Đến lúc này, bạn đã truy xuất những giá trị cột từ chỉ một `DataReader` như những đối tượng chung của lớp `System.Object` (những đối tượng thường được tham chiếu tới như một hiện thân của kiểu đối tượng C#).

Ghi chú: Tất cả các lớp trong C# đều được dẫn xuất từ lớp `System.Object`.

Tôi sẽ viết lại vòng lặp "while" được trình bày trong ví dụ Trong mục trước đây để cho thấy cách bạn cất giữ những giá trị cột như những đối tượng của lớp `System.Object` như thế nào:

```
while (productsSqlDataReader.Read())
{
    object productID =
        productsSqlDataReader[productIDColPos];
    object productName =
        productsSqlDataReader[productNameColPos];
    object unitPrice =
        productsSqlDataReader[unitPriceColPos];
    object unitsInStock =
        productsSqlDataReader[unitsInStockColPos];
    object discontinued =
        productsSqlDataReader[discontinuedColPos];

    Console.WriteLine("productID = " + productID);
    Console.WriteLine("productName = " + productName);
    Console.WriteLine("unitPrice = " + unitPrice);
    Console.WriteLine("unitsInStock = " + unitsInStock);
    Console.WriteLine("discontinued = " + discontinued);
}
```

Đoạn mã này dẫn đến kết quả đầu ra giống như trong Danh sách 9.1 ở trên. Tất cả những gì tôi đã trình bày trong danh sách 9.1 là để thuyết minh rõ rằng một `DataReader` trả về một giá trị cột như một đối tượng của lớp `System.Object` theo mặc định. Khi một đối tượng của lớp `System.Object` được trình bày bởi phương thức `Console.WriteLine()`, trước tiên đối tượng phải được tuyệt đối chuyển đổi thành một chuỗi và sau đó mới trình bày.

Như thế sẽ tốt để chỉ trình bày những giá trị cột, nhưng sẽ thế nào nếu như bạn muốn thực hiện vài kiểu tính toán nào đó với một giá trị? để làm điều này, trước tiên bạn phải ép kiểu giá trị Tới một kiểu cụ thể. Ví dụ sau đây ép kiểu đối tượng `unitPrice` tới một số thập phân (decimal) và sau đó nhân nó với 1.2

```
decimal newUnitPrice = (decimal) unitPrice * 1.2m;
```

Ghi chú Bạn thêm *m* vào cuối của một số kí tự để chỉ báo nó thộc kiểu thập phân.

Những công việc ép một đối tượng tới một kiểu cụ thể, nhưng nó không được thanh lịch lắm. Nó cũng chống lại một trong những lợi ích chính của một ngôn ngữ lập trình hiện đại: sự sử dụng sự định kiểu mạnh. sự định kiểu mạnh có nghĩa là bạn chọn đúng kiểu của một biến hay đối tượng khi khai báo nó. Lợi ích chính của sự định kiểu mạnh là bạn ít có khả năng có những lỗi trong thời gian chạy (runtime) những chương trình của bạn gây ra bởi việc sử dụng kiểu sai. Đây là nhờ trình biên tập kiểm tra mã của bạn để chắc chắn rằng ngữ cảnh của kiểu là đúng. điều cuối cùng là bạn nên cố gắng chọn cho tất cả những biến và những đối tượng của bạn với kiểu thích hợp , và chỉ sử dụng ép kiểu khi bạn không còn sự lựa chọn khác. Trong trường hợp này, bạn có một sự lựa chọn: thay vì việc ép kiểu, bạn có thể sử dụng một trong những phương thức Get\* của đối tượng SqlDataReader để lấy một giá trị cột với một kiểu thích hợp.

Ghi chú

Tôi sử dụng dấu sao trong Get\* để chỉ báo có nhiều phương thức bắt đầu với Get. Bạn có thể xem tất cả phương thức Get\* trong Bảng 9.2, trình bày trong mục trước.

Ví dụ , một trong những phương thức Get\* là GetInt32(), sẽ trả lại một giá trị cột như một giá trị int.

```
int productID =  
productsSqlDataReader.GetInt32(productIDColPos);
```

Như bạn có thể thấy, bạn xem qua số thứ tự của cột mà có giá trị bạn muốn thu được tới phương thức Get\*. Bạn đã thấy cách lấy giá trị thứ tự của cột như thế nào trong mục trước đây.

**SỬ DỤNG NHỮNG PHƯƠNG THỨC *Get\*()* ĐỂ ĐỌC NHỮNG GIÁ TRỊ CỘT:**

Trước khi tôi trình bày bạn những phương thức Get\*() đọc những giá trị cột, bạn cần biết những kiểu C# tiêu chuẩn và những giá trị mà chúng hỗ trợ. Bạn cần biết những điều này nhờ đó bạn có thể hiểu những kiểu tương thích giữa C# và SQL Server trình bày sau. Bảng 9.3 trình bày những kiểu C# tiêu chuẩn , cùng với kiểu .NET và những giá trị có thể được cất giữ trong kiểu C#.

**Bảng 9.3: Những kiểu C# Tiêu chuẩn Và. .NET**

KIỂU C#	KIỂU .NET	NHỮNG GIÁ TRỊ
bool	Boolean	Một giá trị boolean true hoặc false.
byte	Byte	một số nguyên không dấu 8 bit giữa 0 và 2 <sup>8</sup> (255).
char	Char	Một ký tự Unicode 16 bit.
DateTime	DateTime	Một ngày tháng và thời gian giữa 12: 00: 00 AM Tháng giêng 1, 0001 và 11: 59 59 Tháng mười hai 31, 9999. PM
decimal	Decimal	Một số chính xác và quy mô cố định giữa khoảng +/- 1.0* 10 <sup>-28</sup> Và +/- 7.9* 10 <sup>-28</sup> với 28 chữ số có nghĩa của độ chính xác.
double	Double	Một số dấu chấm động 64-bit giữa xấp xỉ +/- 5* 10 <sup>-324</sup> Và +/- 1.7* 10 <sup>308</sup> với 15 tới 16 chữ số có nghĩa của độ chính xác.
float	Single	Một số dấu chấm động 32-bit giữa xấp xỉ +/- 1.5* 10 <sup>-45</sup> và +/- 3.4* 10 <sup>38</sup> với 7 chữ số có nghĩa của độ chính xác.
Guid	Guid	Một giá trị số nguyên không dấu 128-bit (16 byte) mà đó là duy nhất ngang qua tất cả các máy tính và những mạng.
int	Int32	Một số nguyên 32 bit giữa -2 <sup>31</sup> (-2,147,483,648) and 2 <sup>31</sup> - 1 (2,147,483,647).

long	Int64	Một số nguyên 64 bit giữa $-2^{63}$ (-9,223,372,036,854,775,808) and $2^{63} - 1$ (9,223,372,036,854,775,807).
sbyte	SByte	Một số nguyên 8 bit giữa $-2^7$ (-128) and $2^7 - 1$ (127).
short	Int16	Một số nguyên 16 bit giữa $-2^{15}$ (-32,768) and $2^{15} - 1$ (32,767).
string	String	Một chuỗi dài -biến của những ký tự Unicode16-bit.
uint	UInt32	Một số nguyên không dấu 32-bit giữa 0 Và $2^{32} - 1$ (4,294,967,295).
ulong	UInt64	Một số nguyên không dấu 64-bit giữa 0 Và $2^{64} - 1$ (18,446,744,073,709,551,615).
ushort	UInt16	Một số nguyên không dấu 16-bit giữa 0 Và $2^{16} - 1$ (65,535).

Bảng 9.4 trình bày những kiểu dữ liệu SQL Server, những kiểu C# tiêu chuẩn thích hợp, và mỗi phương thức DataReader Get\*() trả về mỗi kiểu C#. Bạn sử dụng bảng này để chọn ra phương thức nào để truy xuất một kiểu cột cụ thể. Chẳng hạn, nếu bạn cần lấy giá trị của một cột bigint(số nguyên lớn), bạn gọi phương thức GetInt64() để trả về một giá trị long.

**Table 9.4: NHỮNG KIỂU SQL SERVER, KIỂU C# TIÊU CHUẨN THÍCH HỢP, VÀ PHƯƠNG THỨC GET\***

**KIỂU SQL SERVER      KIỂU C# TIÊU CHUẨN      PHƯƠNG THỨC GET\* ()**

binary	byte[]	GetBytes()
bigint	long	GetInt64()
bit	bool	GetBoolean()
char	string	GetString()
datetime	DateTime	GetDateTime()
decimal	decimal	GetDecimal()
float	double	GetDouble()
image	byte[]	GetBytes()
int	int	GetInt32()
money	decimal	GetDecimal()
nchar	string	GetString()
ntext	string	GetString()
nvarchar	string	GetString()
numeric	decimal	GetDecimal()
real	float	GetFloat()
smalldatetime	DateTime	GetDateTime()
smallint	short	GetInt16()
smallmoney	decimal	GetDecimal()
sql_varient	object	GetValue()
text	string	GetString()
timestamp	byte[]	GetBytes()
tinyint	byte	GetByte()
varbinary	byte[]	GetBytes()
varchar	string	GetString()
uniqueidentifier	Guid	GetGuid()

**Ghi chú** Bạn có thể xem những kiểu dữ liệu SQL Server và những giá trị được hỗ trợ bởi những kiểu này trong Bảng 2.3 của Chương 2, "Giới thiệu về những cơ sở dữ liệu."

**Ghi chú** Những phương thức Get\* được định nghĩa trong tất cả những lớp DataReader và làm việc cho tất cả các cơ sở dữ liệu.

Tiếp theo bạn sẽ thấy cách sử dụng một số những phương thức trình bày trong bảng trên Bảng 9.4.

**MỘT THÍ DỤ VỀ SỰ SỬ DỤNG NHỮNG PHƯƠNG THỨC Get\*():**

Chúng ta hãy xem xét một ví dụ đọc những cột ProductID, ProductName, UnitPrice, UnitsInStock, từ bảng Products sử dụng những phương thức Get\*() .

Để hình dung ra phải sử dụng phương thức Get\*() nào để truy xuất một kiểu dữ liệu cột SQL Server cụ thể , bạn sử dụng Bảng 9.4, được trình bày trước đó. Chẳng hạn, cột ProductID là một kiểu int trong SQL Server, và tra tìm kiểu int SQL server này trong Bảng 9.4, bạn có thể nhìn thấy bạn cần sử dụng phương thức GetInt32() để thu được giá trị cột như một C# int. Bảng 9.5 tổng kết những tên cột, những kiểu SQL Server, những phương thức Get\*(), và những kiểu trả lại C# cần thiết để truy xuất năm cột từ Bảng Products.

**Bảng 9.5: những cột của bảng, những kiểu, và những phương thức.**

TÊN CỘT	Kiểu CỘT TRONG SQL Server	PHƯƠNG THỨC GET*	Kiểu TRẢ VỀ C#
ProductID	int	GetInt32()	int
ProductName	nvarchar	GetString()	string
UnitPrice	money	GetDecimal()	decimal
UnitsInStock	smallint	GetInt16()	short
Discontinued	bit	GetBoolean()	bool

Chúng ta hãy giả thiết rằng bạn đã có một đối tượng SqlDataReader đặt tên productsSqlDataReader và nó sẽ được sử dụng để đọc năm cột từ bảng Products. Vòng lặp "while" sau đây sử dụng phương thức Get\*(), và những kiểu C# được trả lại được trình bày trong Bảng 9.5 để thu được những giá trị cột từ productsSqlDataReader:

```
while (productsSqlDataReader.Read())
{
    int productID =
    productsSqlDataReader.GetInt32(productIDColPos);
    Console.WriteLine("productID = " + productID);

    string productName =
    productsSqlDataReader.GetString(productNameColPos);
    Console.WriteLine("productName = " + productName);

    decimal unitPrice =
    productsSqlDataReader.GetDecimal(unitPriceColPos);
    Console.WriteLine("unitPrice = " + unitPrice);

    short unitsInStock =
    productsSqlDataReader.GetInt16(unitsInStockColPos);
    Console.WriteLine("unitsInStock = " + unitsInStock);

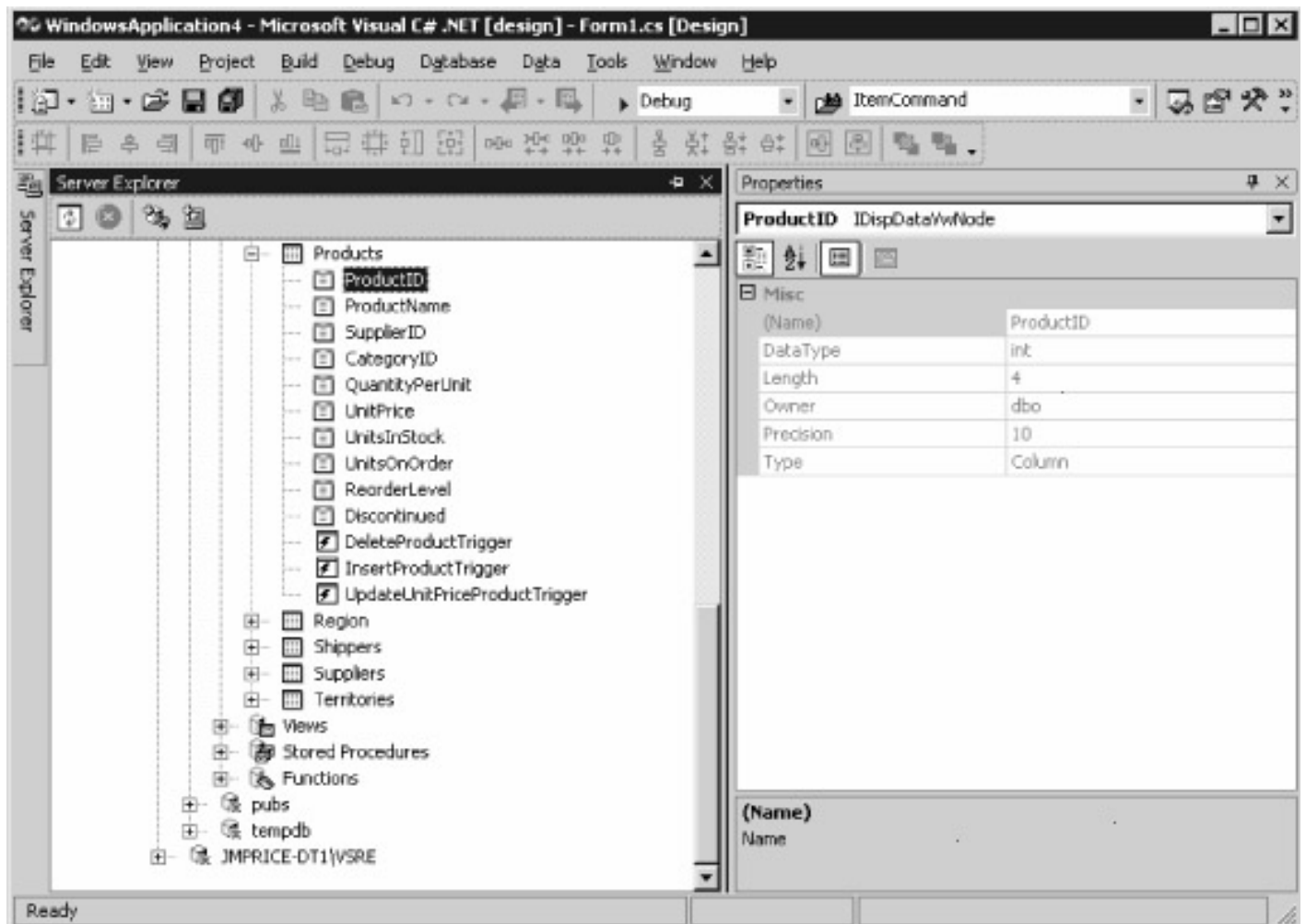
    bool discontinued =
    productsSqlDataReader.GetBoolean(discontinuedColPos);
    Console.WriteLine("discontinued = " + discontinued);
}
```



}

Như bạn thấy, làm biến của kiểu thích hợp được tạo ra trong vòng lặp "while" này, mỗi biến được dùng để cất giữ kết quả trả về từ phương thức Get\*(). Chẳng hạn, biến ProductID được dùng để cất giữ giá trị cột ProductID, và do ProductID là kiểu int SQL Server, kiểu C# thích hợp với biến ProductID là int. Để lấy giá trị cột ProductID như một C# int, Bạn gọi phương thức GetInt32(). Tương tự, biến productName là một chuỗi C# và được sử dụng để cất giữ giá trị cột ProductName. Cột này thuộc kiểu nvarchar trong SQL Server, và để lấy giá trị cột ProductName, phương thức GetString() được sử dụng.

Tất nhiên, mã này phụ thuộc vào những thông tin của bạn nắm bắt về kiểu của cột cơ sở dữ liệu. Nếu bạn không biết về kiểu của một cột, bạn có thể sử dụng trình duyệt server (Server Explorer) của Visual Studio .NET. Chẳng hạn, Hình 9.1 cho thấy những chi tiết của cột ProductID trong bảng Products. Như bạn thấy, ProductID là một int.



**Hình 9.1: truy tìm kiểu của một cột sử dụng trình duyệt server (Server Explorer) của Visual Studio .NET.**

Trước khi đóng mục này, Tôi sẽ chỉ cho bạn cách để lấy kiểu của .NET, và kiểu cơ sở dữ liệu của một cột sử dụng C#. Bạn lấy kiểu của .NET để đại diện cho một cột sử dụng phương thức GetFieldType() của đối tượng DataReader của bạn. thí dụ:

```
Console.WriteLine("ProductID .NET type = " +  
productsSqlDataReader.GetFieldType(productIDColPos));
```

Mã này sẽ hiển thị như sau:

```
ProductID .NET type = System.Int32
```

Như bạn thấy, kiểu System.Int32 .NET được dùng để đại diện cho cột ProductID. Kiểu System.Int32 .NET tương ứng tới kiểu int trong C#. Bạn có thể xem lại sự tương ứng của kiểu này Trong Bảng 9.3, chỉ ra trước đó.



Bạn có thể lấy kiểu cơ sở dữ liệu cho một cột sử dụng phương thức `GetDataTypeName()` của đối tượng `DataReader` của bạn. thí dụ:

```
Console.WriteLine("ProductID database type = " +  
productsSqlDataReader.GetDataTypeName(productIDColPos));
```

Thí dụ này hiển thị:

```
ProductID database type = int
```

Như bạn thấy, cột `ProductID` thuộc kiểu `int` trong `SQL Server`.

Danh sách 9.2 sử dụng những ví dụ mã được trình bày trong mục này.

```
/*  
StronglyTypedColumnValues.cs illustrates how to read  
column values as C# types using the Get* methods  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class StronglyTypedColumnValues  
{  
    public static void Main()  
    {  
        SqlConnection mySqlConnection =  
            new SqlConnection(  
                "server=localhost;database=Northwind;uid=sa;pwd=sa"  
            );  
  
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
  
        mySqlCommand.CommandText =  
            "SELECT TOP 5 ProductID, ProductName, UnitPrice, " +  
            "UnitsInStock, Discontinued " +  
            "FROM Products " +  
            "ORDER BY ProductID";  
  
        mySqlConnection.Open();  
  
        SqlDataReader productsSqlDataReader =  
            mySqlCommand.ExecuteReader();  
  
        int productIDColPos =  
            productsSqlDataReader.GetOrdinal("ProductID");  
        int productNameColPos =  
            productsSqlDataReader.GetOrdinal("ProductName");  
        int unitPriceColPos =  
            productsSqlDataReader.GetOrdinal("UnitPrice");  
        int unitsInStockColPos =  
            productsSqlDataReader.GetOrdinal("UnitsInStock");  
        int discontinuedColPos =  
            productsSqlDataReader.GetOrdinal("Discontinued");
```

```

// use the GetFieldType() method of the DataReader object
// to obtain the .NET type of a column
Console.WriteLine("ProductID .NET type = " +
productsSqlDataReader.GetFieldType(productIDColPos));
Console.WriteLine("ProductName .NET type = " +
productsSqlDataReader.GetFieldType(productNameColPos));
Console.WriteLine("UnitPrice .NET type = " +
productsSqlDataReader.GetFieldType(unitPriceColPos));
Console.WriteLine("UnitsInStock .NET type = " +
productsSqlDataReader.GetFieldType(unitsInStockColPos));
Console.WriteLine("Discontinued .NET type = " +
productsSqlDataReader.GetFieldType(discontinuedColPos));

// use the GetDataTypeName() method of the DataReader object
// to obtain the database type of a column
Console.WriteLine("ProductID database type = " +
productsSqlDataReader.GetDataTypeName(productIDColPos));
Console.WriteLine("ProductName database type = " +
productsSqlDataReader.GetDataTypeName(productNameColPos));
Console.WriteLine("UnitPrice database type = " +
productsSqlDataReader.GetDataTypeName(unitPriceColPos));
Console.WriteLine("UnitsInStock database type = " +
productsSqlDataReader.GetDataTypeName(unitsInStockColPos));
Console.WriteLine("Discontinued database type = " +
productsSqlDataReader.GetDataTypeName(discontinuedColPos));

// read the column values using Get* methods that
// return specific C# types
while (productsSqlDataReader.Read())
{
    int productID =
productsSqlDataReader.GetInt32(productIDColPos);
    Console.WriteLine("productID = " + productID);

    string productName =
productsSqlDataReader.GetString(productNameColPos);
    Console.WriteLine("productName = " + productName);

    decimal unitPrice =
productsSqlDataReader.GetDecimal(unitPriceColPos);
    Console.WriteLine("unitPrice = " + unitPrice);

    short unitsInStock =
productsSqlDataReader.GetInt16(unitsInStockColPos);
    Console.WriteLine("unitsInStock = " + unitsInStock);

    bool discontinued =
productsSqlDataReader.GetBoolean(discontinuedColPos);
    Console.WriteLine("discontinued = " + discontinued);
}

productsSqlDataReader.Close();
mySqlConnection.Close();
}
}

```

### Đầu ra của chương trình này như sau:

```
ProductID .NET type = System.Int32
ProductName .NET type = System.String
UnitPrice .NET type = System.Decimal
UnitsInStock .NET type = System.Int16
Discontinued .NET type = System.Boolean
ProductID database type = int
    ProductName database type = nvarchar
UnitPrice database type = money
UnitsInStock database type = smallint
Discontinued database type = bit
productID = 1
productName = Chai
unitPrice = 18
unitsInStock = 39
discontinued = False
productID = 2
productName = Chang
unitPrice = 19
unitsInStock = 17
discontinued = False
productID = 3
productName = Aniseed Syrup
unitPrice = 10
unitsInStock = 13
discontinued = False
productID = 4
productName = Chef Anton's Cajun Seasoning
unitPrice = 22
unitsInStock = 53
discontinued = False
productID = 5
productName = Chef Anton's Gumbo Mix
unitPrice = 21.35
unitsInStock = 0
discontinued = True
```

### **SỬ DỤNG NHỮNG PHƯƠNG THỨC `GetSql*()` ĐỂ ĐỌC NHỮNG GIÁ TRỊ CỘT:**

Để bổ xung thêm về sự sử dụng những phương thức `Get*` đọc những giá trị cột như những kiểu C# tiêu chuẩn, nếu bạn đang sử dụng SQL Server, bạn có thể cũng sử dụng những phương thức `GetSql*`. Những phương thức `GetSql*` trả lại những giá trị như những kiểu `Sql*`, nó tương ứng với những kiểu thực tế được sử dụng bởi SQL Server trong cơ sở dữ liệu.

Ghi chú Bạn có thể thấy tất cả những phương thức `GetSql*` trong [Bảng 9.2](#), chỉ ra trước đó.

Những phương thức `GetSql*` và những kiểu `Sql*` được định nghĩa trong không gian tên `System.Data.SqlTypes`, và chúng dành riêng cho SQL Server. Ngoài ra, Những phương thức `GetSql*` cũng đặc trưng đối với lớp `SqlDataReader`. Việc sử dụng những phương thức `GetSql*` và những kiểu `Sql*` giúp ngăn ngừa những lỗi do sự chuyển đổi kiểu gây ra bởi sự mất độ chính xác trong những giá trị số. Những phương thức `GetSql*` luôn nhanh hơn so với những phương thức `Get*()` tương ứng của chúng. Bởi vì những phương thức `GetSql*` không cần chuyển đổi giữa những kiểu SQL Server và những kiểu C# tiêu chuẩn, còn những phương thức `Get*()` lại phải thực hiện .

Mẹo nhỏ : Nếu bạn đang sử dụng SQL Server, nên luôn sử dụng những phương thức GetSql\* và những kiểu Sql\* hơn là những phương thức Get\*() và những kiểu C# tiêu chuẩn . Tôi đã trình bày với bạn những phương thức Get\*() trước đó chỉ vì chúng làm việc với những cơ sở dữ liệu không thuộc SQL Server.

Bảng 9.6 cho thấy những kiểu Sql\* và những giá trị có thể được cất giữ trong những kiểu này.

**Bảng 9.6: Sql\* TYPES**

Sql* TYPE	VALUES
SqlBinary	Một chuỗi dài biến của dữ liệu nhị phân.
SqlBoolean	Một số nguyên với một giá trị 1 hoặc 0 .
SqlByte	Một Giá trị số nguyên không dấu 8-bit giữa 0 và $2^8$ (255).
SqlDateTime	Một giá trị ngày tháng và thời gian giữa 12:00:00 AM tháng giêng 1, 1753 và 11:59:59 PM tháng chạp 31, 9999. chính xác tới mỗi 3.33 mili-giây .
SqlDecimal	một giá trị số chính xác và quy mô cố định giữa $-10^{38} + 1$ và $10^{38} - 1$ .
SqlDouble	Một số dấu chấm động 64-bit giữa -1.79769313486232E308 and 1.79769313486232E308 với 15 chữ số có nghĩa của độ chính xác.
SqlGuid	Một giá trị số nguyên mẫu (16 byte) và duy nhất ngang qua tất cả các máy tính và những mạng.
SqlInt16	Một số nguyên có dấu 16-bit giữa $-2^{15}$ (-32,768) và $2^{15}$ (32,767).
SqlInt32	Một số nguyên có dấu 32-bit giữa $-2^{31}$ (-2,147,483,648) và $2^{31}$ (2,147,483,647).
SqlInt64	Một số nguyên có dấu 64-bit giữa $-2^{63}$ (-9,223,372,036,854,775,808) và $2^{63}$ (9,223,372,036,854,775,807).
SqlMoney	Một giá trị tiền tệ giữa -922,337,203,685,477.5808 và 922,337,203,685,477.5807. chính xác tới với 1/10,000 th của một đơn vị tiền tệ.
SqlSingle	Một số dấu chấm động 32-bit giữa -3.402823E38 and 3.402823E38 với bảy chữ số có nghĩa của độ chính xác.
SqlString	Một chuỗi dài biến của những ký tự.

Bảng 9.7 cho thấy những kiểu SQL Server, Những kiểu Sql tương ứng\* , và những phương thức GetSql\* được dùng để đọc một cột như Kiểu Sql\*.

KIỂU SQL SERVER	KIỂU Sql*	PHƯƠNG THỨC GetSql*
bigint	SqlInt64	GetSqlInt64()
int	SqlInt32	GetSqlInt32()
smallint	SqlInt16	GetSqlInt16()
tinyint	SqlByte	GetSqlByte()
bit	SqlBoolean	GetSqlBoolean()
decimal	SqlDecimal	GetSqlDecimal()
numeric	SqlDecimal	GetSqlDecimal()
money	SqlMoney	GetSqlMoney()
smallmoney	SqlMoney	GetSqlMoney()

KIỂU SQL SERVER	KIỂU Sql*	PHƯƠNG THỨC GetSql*
float	SqlDouble	GetSqlDouble()
real	SqlSingle	GetSqlSingle()
datetime	SqlDateTime	GetSqlDateTime()
smalldatetime	SqlDateTime	GetSqlDateTime()
char	SqlString	GetSqlString()
varchar	SqlString	GetSqlString()
text	SqlString	GetSqlString()
nchar	SqlString	GetSqlString()
nvarchar	SqlString	GetSqlString()
ntext	SqlString	GetSqlString()
binary	SqlBinary	GetSqlBinary()
varbinary	SqlBinary	GetSqlBinary()
image	SqlBinary	GetSqlBinary()
sql_varient	object	GetSqlValue()
timestamp	SqlBinary	GetSqlBinary()
uniqueidentifier	SqlGuid	GetSqlGuid()

Tiếp theo bạn sẽ thấy cách sử dụng một số phương thức trình bày trong [Bảng 9.7](#).

### **MỘT THÍ DỤ SỬ DỤNG NHỮNG PHƯƠNG THỨC *GetSql\*()*:**

Chúng ta hãy xem xét một ví dụ mà đọc những cột ProductID, ProductName, UnitPrice, UnitsInStock, Và Discontinued từ bảng Products sử dụng những phương thức GetSql\*.

Để hình dung ra phương thức GetSql\*() nào được sử dụng để truy xuất một kiểu cột cụ thể, bạn sử dụng [Bảng 9.7](#), được chỉ ra trước đó. Chẳng hạn, cột ProductID là một kiểu int SQL Server, Và bạn tra kiểu này trong [Bảng 9.7](#), bạn có thể nhìn thấy bạn phải sử dụng phương thức GetSqlInt32() để thu được giá trị cột như một C# SqlInt32. Bảng 9.8 tổng kết những tên cột, những kiểu SQL Server, Những phương thức GetSql\* , và những kiểu trả về Sql\* cho những cột truy xuất được từ Bảng Products.

#### **Bảng 9.8: những cột của bảng Products, những kiểu, và những phương thức GetSql\***

COLUMN NAME/ SQL SERVER COLUMN TYPE/ GETSql\* METHOD/ Sql\* Return Type

ProductID	int	GetInt32()	SqlInt32
ProductName	nvarchar	GetSqlString()	SqlString
UnitPrice	money	GetSqlMoney()	SqlMoney
UnitsInStock	smallint	GetSqlInt16()	SqlInt16
Discontinued	bit	GetSqlBoolean()	SqlBoolean

Chúng ta hãy giả thiết rằng bạn đã có một đối tượng SqlDataReader có tên productsSqlDataReader và nó có thể được dùng để đọc những cột từ bảng những sản phẩm. Vòng lặp "while" sau đây sử dụng những phương thức GetSql\* và những kiểu Sql\* trả về được trình bày trước đó Trong [Bảng 9.8](#) để thu được những giá trị cột từ productsSqlDataReader:

```

while (productsSqlDataReader.Read())
{
    SqlInt32 productID =
    productsSqlDataReader.GetSqlInt32(productIDColPos);
    Console.WriteLine("productID = " + productID);

    SqlString productName =
    productsSqlDataReader.GetSqlString(productNameColPos);
    Console.WriteLine("productName = " + productName);

    SqlMoney unitPrice =
    productsSqlDataReader.GetSqlMoney(unitPriceColPos);
    Console.WriteLine("unitPrice = " + unitPrice);

    SqlInt16 unitsInStock =
    productsSqlDataReader.GetSqlInt16(unitsInStockColPos);
    Console.WriteLine("unitsInStock = " + unitsInStock);

    SqlBoolean discontinued =
    productsSqlDataReader.GetSqlBoolean(discontinuedColPos);
    Console.WriteLine("discontinued = " + discontinued);
}

```

### **Danh sách 9.3 sử dụng vòng lặp "while" này.**

```

/*
StronglyTypedColumnValuesSql.cs illustrates how to read
column values as Sql* types using the GetSql* methods
*/

using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;

class StronglyTypedColumnValuesSql
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
        new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa"
        );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        mySqlCommand.CommandText =
        "SELECT TOP 5 ProductID, ProductName, UnitPrice, " +
        "UnitsInStock, Discontinued " +
        "FROM Products " +
        "ORDER BY ProductID";

        mySqlConnection.Open();

        SqlDataReader productsSqlDataReader =
        mySqlCommand.ExecuteReader();
    }
}

```

```

        int productIDColPos =
        productsSqlDataReader.GetOrdinal("ProductID");
        int productNameColPos =
        productsSqlDataReader.GetOrdinal("ProductName");
        int unitPriceColPos =
        productsSqlDataReader.GetOrdinal("UnitPrice");
        int unitsInStockColPos =
        productsSqlDataReader.GetOrdinal("UnitsInStock");
        int discontinuedColPos =
        productsSqlDataReader.GetOrdinal("Discontinued");

        // read the column values using GetSql* methods that
        // return specific Sql* types
        while (productsSqlDataReader.Read())
        {
            SqlInt32 productID =
            productsSqlDataReader.GetSqlInt32(productIDColPos);
            Console.WriteLine("productID = " + productID);

            SqlString productName =
            productsSqlDataReader.GetSqlString(productNameColPos);
            Console.WriteLine("productName = " + productName);

            SqlMoney unitPrice =
            productsSqlDataReader.GetSqlMoney(unitPriceColPos);
            Console.WriteLine("unitPrice = " + unitPrice);

            SqlInt16 unitsInStock =
            productsSqlDataReader.GetSqlInt16(unitsInStockColPos);
            Console.WriteLine("unitsInStock = " + unitsInStock);

            SqlBoolean discontinued =
            productsSqlDataReader.GetSqlBoolean(discontinuedColPos);
            Console.WriteLine("discontinued = " + discontinued);
        }

        productsSqlDataReader.Close();
        mySqlConnection.Close();
    }
}

```

**Đầu ra của chương trình này như sau:**

```

productID = 1
productName = Chai
unitPrice = 18
unitsInStock = 39
discontinued = False
productID = 2
productName = Chang
unitPrice = 19
unitsInStock = 17
discontinued = False
productID = 3
productName = Aniseed Syrup
unitPrice = 10
unitsInStock = 13

```

```
discontinued = False
productID = 4
productName = Chef Anton's Cajun Seasoning
unitPrice = 22
unitsInStock = 53
discontinued = False
productID = 5
productName = Chef Anton's Gumbo Mix
unitPrice = 21.35
unitsInStock = 0
discontinued = True
```

## **ĐỌC NHỮNG GIÁ TRỊ NULL:**

Như bạn đã học Trong Chương 2, một cột định nghĩa là null có thể lưu trữ một giá trị null( cột trống chưa gán giá trị). Một giá trị null cho biết giá trị cột chưa được biết. Một kiểu C# tiêu chuẩn không thể cất giữ Một giá trị null, nhưng một kiểu Sql\* có thể.

Chúng ta hãy xem xét một ví dụ về việc đọc một giá trị null từ cơ sở dữ liệu. cho là bạn đã thực hiện một phát biểu SELECT để truy xuất cột UnitPrice cho một hàng từ bảng những sản phẩm, và cột UnitPrice này chứa một giá trị null. Nếu bạn thử để cất giữ giá trị null này trong một kiểu C# tiêu chuẩn (như một số thập phân) với mã sau đây:

```
decimal unitPrice =
productsSqlDataReader.GetDecimal(unitPriceColPos);
```

và bạn sẽ có ngoại lệ sau đây:

`System.Data.SqlTypes.SqlNullValueException`

Bạn sẽ cũng có ngoại lệ này nếu Bạn thử để cất giữ giá trị null trong một đối tượng, như trong ví dụ sau đây:

```
object unitPriceObj =
productsSqlDataReader["UnitPrice"];
```

Bạn có thể kiểm tra một cột có chứa một giá trị null hay không sử dụng phương thức IsDBNull() của một đối tượng DataReader. Phương pháp này trả về một Đại số Boole giá trị true hay false cho biết liệu có phải giá trị cột là null. Bạn có thể sử dụng kết quả Đại số Boole này để quyết định nên làm gì. Chẳng hạn:

```
if (productsSqlDataReader.IsDBNull(unitPriceColPos))
{
    Console.WriteLine("UnitPrice column contains a null value");
}
else
{
    unitPrice = productsSqlDataReader.GetDecimal(unitPriceColPos);
}
```

Vì productsSqlDataReader.IsDBNull(unitPriceColPos) trả về true, ví dụ này hiển thị kết quả sau:

`UnitPrice column contains a null value`



Như đã được đề cập, Một kiểu Sql\* có thể cất giữ một giá trị null. Một giá trị null được cất giữ như một giá trị ' null '. Chẳng hạn:

```
SqlMoney unitPrice =  
productsSqlDataReader.GetSqlMoney(unitPriceColPos);  
Console.WriteLine("unitPrice = " + unitPrice);
```

Những kết quả của ví dụ này:

```
unitPrice = Null
```

Mỗi kiểu trong những kiểu Sql\* cũng đều có một thuộc tính Boolean có tên IsNull là true khi Đối tượng Sql \* chứa một giá trị null. Chẳng hạn:

```
Console.WriteLine("unitPrice.IsNull = " + unitPrice.IsNull);
```

Những kết quả của ví dụ này:

```
unitPrice.IsNull = True
```

true được trình bày bởi vì unitPrice chứa một giá trị null.

## **THỰC THI NHIỀU PHÁT BIỂU SQL:**

Điện hình, chương trình C# và cơ sở dữ liệu của bạn sẽ chạy trên những máy tính khác nhau và truyền thông qua một mạng. Mỗi lần bạn thực hiện một lệnh trong chương trình của bạn, nó phải hành trình qua mạng tới cơ sở dữ liệu và được thực hiện bởi cơ sở dữ liệu, và bất kỳ kết quả nào phải được gửi trả lại ngang qua mạng đến chương trình của bạn. Đó là sự quá tải giao thông mạng! Một cách tiềm tàng để giảm bớt giao thông mạng là thực hiện nhiều câu lệnh SQL tại cùng một thời điểm.

Trong mục này, bạn sẽ thấy cách để thực thi nhiều phát biểu SELECT và truy xuất kết quả, và bạn sẽ thấy cách thực hiện nhiều phát biểu SELECT, INSERT, UPDATE , và DELETE như thế nào, và sự đặt xen kẽ các lệnh ra sao.

## **THỰC THI NHIỀU PHÁT BIỂU SELECT:**

Chúng ta hãy xem xét cách thực hiện nhiều phát biểu SELECT và truy xuất những kết quả như thế nào .

Mã sau đây đầu tiên tạo ra một đối tượng SqlCommand có tên mySqlCommand và gán thuộc tính CommandText tới ba phát biểu SELECT khác nhau :

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
mySqlCommand.CommandText =  
"SELECT TOP 5 ProductID, ProductName " +  
"FROM Products " +  
"ORDER BY ProductID;" +  
"SELECT TOP 3 CustomerID, CompanyName " +  
"FROM Customers " +  
"ORDER BY CustomerID;" +  
"SELECT TOP 6 OrderID, CustomerID " +  
"FROM Orders " +  
"ORDER BY OrderID;"
```

Chú ý tất cả những phát biểu SELECT được tách ra bởi những dấu chấm phẩy.

Cần thận chỉ truy xuất những hàng và những cột bạn thật sự cần. Đồng thời, chắc chắn rằng bạn sử dụng những phát biểu SELECT mà truy xuất những hàng từ nhiều bảng. Chẳng hạn, nếu bạn muốn xem tất cả những đơn đặt (orders) được đặt bởi khách hàng có CustomerID là ALFKI, đừng thực hiện hai phát biểu SELECT riêng biệt để truy nhập những những bảng khách hàng và đơn đặt . Thay vào đó, sử dụng một table join, như được trình bày trong phát biểu SELECT sau đây:

```
SELECT Customers.CustomerID, CompanyName, OrderID
FROM Customers, Orders
WHERE Customers.CustomerID = Orders.CustomerID
AND Customers.CustomerID = 'ALFKI';
```

---

Để chạy những câu lệnh SQL đầu, bạn gọi phương thức ExecuteReader() , nó trả về một đối tượng SqlDataReader:

```
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
```

Lệnh sẽ trả lại ba tập hợp kết quả, một cho mỗi phát biểu SELECT. Để đọc tập hợp kết quả đầu tiên , bạn gọi phương thức Read() của mySqlDataReader. Phương thức Read() trả về false khi không còn hàng cho sự Đọc. Khi bạn ở chỗ cuối của một tập hợp kết quả, bạn gọi phương thức NextResult() của mySqlDataReader trước khi đọc tập hợp kết quả kế tiếp. Phương thức NextResult sẽ chuyển mySqlDataReader về tập hợp kết quả kế tiếp và trả về false khi không còn tập hợp kết quả nào.

Mã sau đây minh họa sự sử dụng phương thức Read() Và NextResult() để đọc ba tập hợp kết quả từ những phát biểu SELECT.

```
do
{
    while (mySqlDataReader.Read())
    {
        Console.WriteLine("mySqlDataReader[0] = " +
            mySqlDataReader[0]);
        Console.WriteLine("mySqlDataReader[1] = " +
            mySqlDataReader[1]);
    }
}
while (mySqlDataReader.NextResult());
```

Chú ý sự sử dụng của vòng lặp ngoài "do ...while loop", để thử giá trị trả về từ mySqlDataReader.NextResult() tại điểm kết thúc. Vì vòng lặp "do ...while loop" kiểm tra điều kiện ở cuối vòng, điều đó có nghĩa là những phát biểu trong vòng lặp "do ...while loop" được thực hiện ít nhất một lần. Bạn muốn đặt lệnh gọi NextResult() ở cuối vì trước tiên nó thử trả mySqlDataReader tới tập hợp kết quả kế tiếp và sau đó trả lại kết quả Đại số Boole để xem liệu có phải còn có một tập kết quả khác để di chuyển Tới. Nếu bạn đặt lệnh gọi NextResult() Trong một vòng lặp "while loop" bình thường, thì mySqlDataReader sẽ bỏ qua qua tập hợp kết quả đầu tiên, và bạn không muốn làm điều đó!

#### **Danh sách 9.4 minh họa cách để thực hiện nhiều phát biểu SELECT và đọc những kết quả.**

```
/*
ExecuteMultipleSelects.cs illustrates how to execute
multiple SELECT statements using a SqlCommand object
and read the results using a SqlDataReader object
*/

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

```

class ExecuteSelect
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // set the CommandText property of the SqlCommand object to
        // the mutliple SELECT statements
        mySqlCommand.CommandText =
            "SELECT TOP 5 ProductID, ProductName " +
            "FROM Products " +
            "ORDER BY ProductID;" +
            "SELECT TOP 3 CustomerID, CompanyName " +
            "FROM Customers " +
            "ORDER BY CustomerID;" +
            "SELECT TOP 6 OrderID, CustomerID " +
            "FROM Orders " +
            "ORDER BY OrderID;";

        mySqlConnection.Open();

        SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        // read the result sets from the SqlDataReader object using
        // the Read() and NextResult() methods
        do
        {
            while (mySqlDataReader.Read())
            {
                Console.WriteLine("mySqlDataReader[0] = " +
                    mySqlDataReader[0]);
                Console.WriteLine("mySqlDataReader[1] = " +
                    mySqlDataReader[1]);
            }
            Console.WriteLine(""); // visually split the results
        } while (mySqlDataReader.NextResult());

        mySqlDataReader.Close();
        mySqlConnection.Close();
    }
}

```

**Đầu ra của chương trình này như sau:**

```

mySqlDataReader[0] = 1
mySqlDataReader[1] = Chai
mySqlDataReader[0] = 2
mySqlDataReader[1] = Chang
mySqlDataReader[0] = 3
mySqlDataReader[1] = Aniseed Syrup
mySqlDataReader[0] = 4
mySqlDataReader[1] = Chef Anton's Cajun Seasoning
mySqlDataReader[0] = 5
mySqlDataReader[1] = Chef Anton's Gumbo Mix

```

```
mySqlDataReader[0] = ALFKI  
mySqlDataReader[1] = Alfreds Futterkiste  
mySqlDataReader[0] = ANATR  
mySqlDataReader[1] = Ana Trujillo3 Emparedados y helados  
mySqlDataReader[0] = ANTON  
mySqlDataReader[1] = Antonio Moreno Taquería
```

```
mySqlDataReader[0] = 10248  
mySqlDataReader[1] = VINET  
mySqlDataReader[0] = 10249  
mySqlDataReader[1] = TOMSP  
mySqlDataReader[0] = 10250  
mySqlDataReader[1] = HANAR  
mySqlDataReader[0] = 10251  
mySqlDataReader[1] = VICTE  
mySqlDataReader[0] = 10252  
mySqlDataReader[1] = SUPRD  
mySqlDataReader[0] = 10253  
mySqlDataReader[1] = HANAR
```

## **THỰC THI NHIỀU PHÁT BIỂU SELECT, INSERT, UPDATE, VÀ DELETE:**

Bạn có thể đặt xen nhiều phát biểu SELECT, INSERT, UPDATE, và DELETE. Điều này có thể giảm thiểu giao thông mạng bởi vì bạn đã gửi nhiều câu lệnh SQL tới cơ sở dữ liệu vào một chuyến đi. Mã sau đây đầu tiên tạo ra một đối tượng SqlCommand có tên mySqlCommand và gán thuộc tính CommandText là tập hợp nhiều câu lệnh SQL được đặt xen:

```
mySqlCommand.CommandText =  
"INSERT INTO Customers (CustomerID, CompanyName) "+  
"VALUES ('J5COM', 'Jason 5 Company');" +  
"SELECT CustomerID, CompanyName " +  
"FROM Customers " +  
"WHERE CustomerID = 'J5COM';" +  
"UPDATE Customers " +  
"SET CompanyName = 'Another Jason Company' " +  
"WHERE CustomerID = 'J5COM';" +  
"SELECT CustomerID, CompanyName " +  
"FROM Customers " +  
"WHERE CustomerID = 'J5COM';" +  
"DELETE FROM Customers " +  
"WHERE CustomerID = 'J5COM';";
```

**Những câu lệnh SQL như sau :**

- Phát biểu INSERT thêm một hàng mới vào bảng khách hàng.
- Phát biểu SELECT đầu tiên truy xuất hàng mới.
- Phát biểu UPDATE sửa đổi cột CompanyName của hàng.
- Phát biểu SELECT thứ hai truy xuất hàng lần nữa.
- Phát biểu DELETE loại bỏ hàng.

Bạn có thể sử dụng vòng lặp "do ...while loop" tương tự như trình bày trong mục trước đây để truy xuất hai tập hợp kết quả được trả về bởi những sự phát biểu SELECT. Vòng lặp tương tự này cũng làm việc mặc dù thí dụ thực thi những phát biểu không phải là SELECT . nó làm việc vì chỉ những phát biểu SELECT trả lại những tập hợp kết quả, và phương thức NextResult() trả về true chỉ với những sự phát biểu SELECT; nó trả về false với những câu lệnh SQL khác. Bởi vậy, NextResult() Trả về false cho phát biểu INSERT và chuyển đến tập hợp kết quả cho phát biểu SELECT đầu tiên, vân vân.

### Danh sách 9.5 minh họa cách thực hiện nhiều câu lệnh SQL như thế nào.

```
/*
ExecuteMultipleSQL.cs illustrates how to execute
multiple SQL statements using a SqlCommand object
*/
using System;
using System.Data;
using System.Data.SqlClient;

class ExecuteMultipleSQL
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // set the CommandText property of the SqlCommand object to
        // the INSERT, UPDATE, and DELETE statements
        mySqlCommand.CommandText =
            "INSERT INTO Customers (CustomerID, CompanyName) " +
            "VALUES ('J5COM', 'Jason 5 Company');" +
            "SELECT CustomerID, CompanyName " +
            "FROM Customers " +
            "WHERE CustomerID = 'J5COM';" +
            "UPDATE Customers " +
            "SET CompanyName = 'Another Jason Company' " +
            "WHERE CustomerID = 'J5COM';" +
            "SELECT CustomerID, CompanyName " +
            "FROM Customers " +
            "WHERE CustomerID = 'J5COM';" +
            "DELETE FROM Customers " +
            "WHERE CustomerID = 'J5COM';";

        mySqlConnection.Open();

        SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        // read the result sets from the SqlDataReader object using
        // the Read() and NextResult() methods
        do
        {
            while (mySqlDataReader.Read())
            {
                Console.WriteLine("mySqlDataReader[0] = " +
```

```

        mySqlDataReader[0]);
        Console.WriteLine("mySqlDataReader[1] = " +
            mySqlDataReader[1]);
    }
    Console.WriteLine(""); // visually split the results
} while (mySqlDataReader.NextResult());

mySqlDataReader.Close();
mySqlConnection.Close();
}
}

```

**Đầu ra từ chương trình này như sau:**

```

mySqlDataReader[0] = J5COM
mySqlDataReader[1] = Jason 5 Company

mySqlDataReader[0] = J5COM
mySqlDataReader[1] = Another Jason Company

```

## **SỬ DỤNG MỘT ĐỐI TƯỢNG DATAREADER TRONG VISUAL STUDIO .NET:**

Bạn không thể tạo ra một đối tượng DataReader trực quan trong Visual Studio .NET (VS .NET); bạn chỉ có thể tạo ra chúng sử dụng những lệnh chương trình.

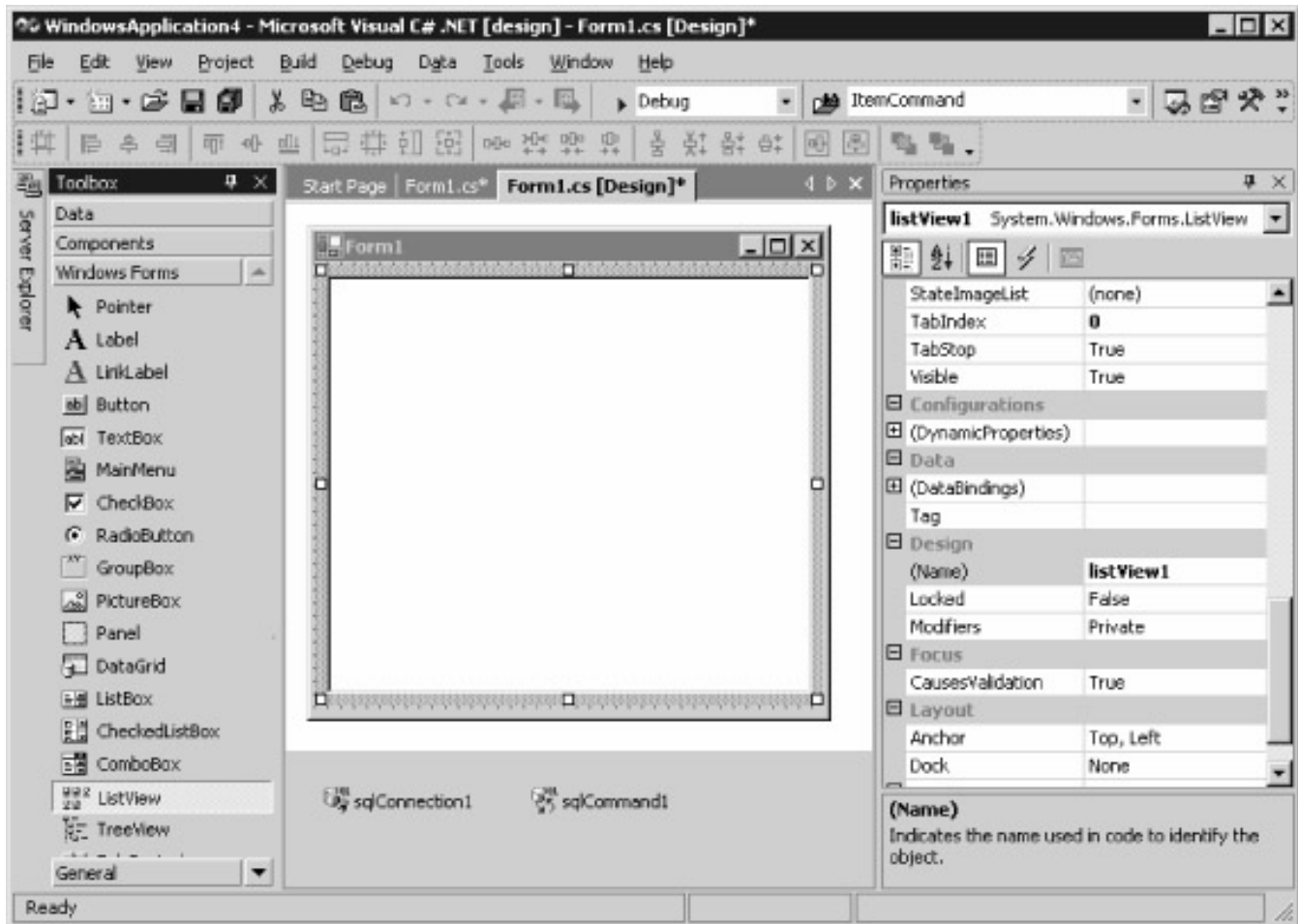
Trong mục này, bạn sẽ thấy cách tạo ra một đối tượng SqlDataReader và sử dụng nó để truy xuất tập hợp kết quả từ một đối tượng SqlCommand như thế nào, và bạn đã thấy cách tạo ra sử dụng VS .NET trong chương trước đây. Đối tượng SqlCommand chứa một phát biểu SELECT truy xuất những cột CustomerID, CompanyName, Và ContactName từ bảng khách hàng. Bạn sẽ hiểu cách để thực hiện phát biểu SELECT , đọc tập hợp kết quả sử dụng đối tượng SqlDataReader, và hiển thị tập hợp kết quả trong một điều khiển ListView như thế nào. Một điều khiển ListView cho phép bạn xem thông tin được đặt trong một lưới.

### **Ghi chú**

Bạn có thể sửa đổi dự án MyDataReader Bạn đã tạo ra Trong chương trước đây, hay nếu bạn không muốn đi theo những chỉ dẫn trong mục này, bạn có thể đơn giản mở dự án hoàn chỉnh VS .NET chứa trong thư mục DataReader. Để mở dự án được bổ sung, lựa chọn File ➤ Open ➤ Project, duyệt tới VS .NET Project \ thư mục DataReader, và mở File WindowsApplication4.csproj .

Nếu bạn đang sửa đổi ứng dụng Windows hiện có của bạn, kéo một điều khiển ListView vào Form của bạn.

Hình 9.2 cho thấy một Form với một điều khiển ListView. Chắc chắn rằng thuộc tính "Name" của ListView của bạn được gán tên là listView1 (đây là Tên mặc định, vì vậy bạn không nên thay đổi nó).



**Hình 9.2: thêm một điều khiển ListView vào Form**

Cảnh báo : nếu bạn mở dự án được bổ sung, bạn không cần phải thêm một ListView control; nó đã có sẵn trên form mẫu. Bạn sẽ cần thay đổi thuộc tính ConnectionString của đối tượng sqlConnection1, như thế nó sẽ kết nối tới cơ sở dữ liệu Northwind SQL Server của bạn. Một khi bạn đã thiết đặt ConnectionString của bạn, Bạn có thể chạy form bằng cách chọn Debug ➤ Start Without Debugging.

Tiếp theo, nhấn đúp một vùng trên form của bạn bên ngoài điều khiển ListView. thao tác này làm VS .NET hiển thị cửa sổ biên tập mã , và bạn sẽ nhìn thấy con trỏ định vị trong phương thức Form1\_Load() ; phương thức này được gọi khi form của bạn được tải ở giai đoạn thoát tiền . Điển hình, đây là phương thức mà từ đó bạn muốn thực hiện những thao tác với cơ sở dữ liệu của bạn. thiết đặt phương thức Form1\_Load() của bạn với mã sau đây:

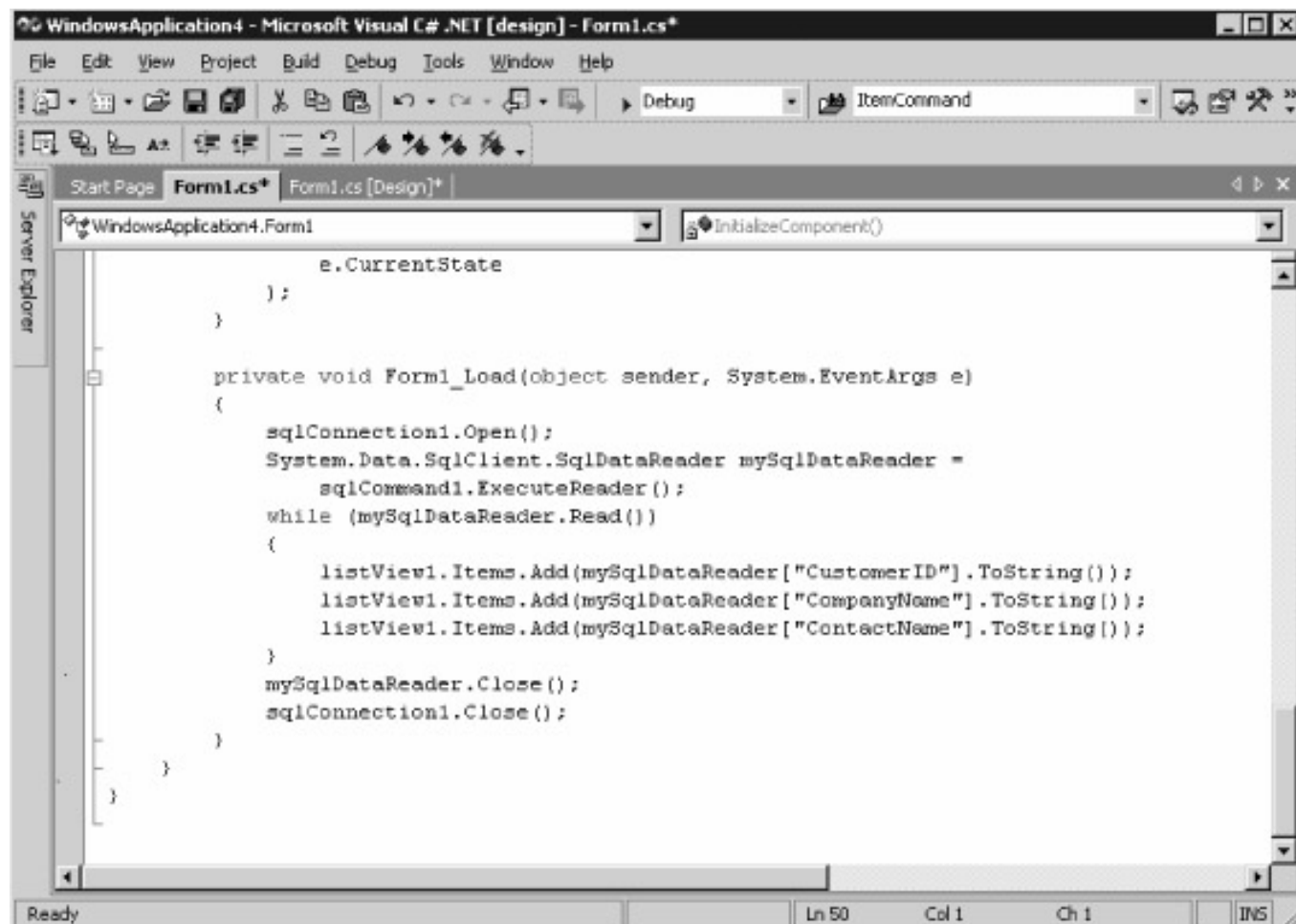
```
private void Form1_Load(object sender, System.EventArgs e)
{
    sqlConnection1.Open();
    System.Data.SqlClient.SqlDataReader mySqlDataReader =
    sqlCommand1.ExecuteReader();
    while (mySqlDataReader.Read())
    {
        listView1.Items.Add(mySqlDataReader["CustomerID"].ToString());
        listView1.Items.Add(mySqlDataReader["CompanyName"].ToString());
        listView1.Items.Add(mySqlDataReader["ContactName"].ToString());
    }
    mySqlDataReader.Close();
    sqlConnection1.Close();
}
```



Chú ý bạn thêm một mục vào điều khiển ListView sử dụng phương thức Add(), được truy nhập sử dụng thuộc tính Items. phương thức Add() chờ đợi một tham số chuỗi, và do đó bạn gọi phương thức ToString() để chuyển đổi đối tượng được trả lại bởi SqlDataReader thành một chuỗi. bạn cũng cần chú ý bao gồm không gian tên khi tham chiếu đến lớp SqlDataReader. bạn sử dụng System.Data.SqlClient.SqlDataReader khi tạo ra đối tượng SqlDataReader.

Mã trước đây mở kết nối cơ sở dữ liệu, tạo ra một đối tượng SqlDataReader, và sử dụng nó để đọc những hàng từ tập hợp kết quả được trả lại bởi đối tượng SqlCommand. Mỗi cột của tập hợp kết quả rồi được thêm vào điều khiển ListView sử dụng phương thức Add().

### Hình 9.3 cho thấy mã của phương thức Form1\_Load().

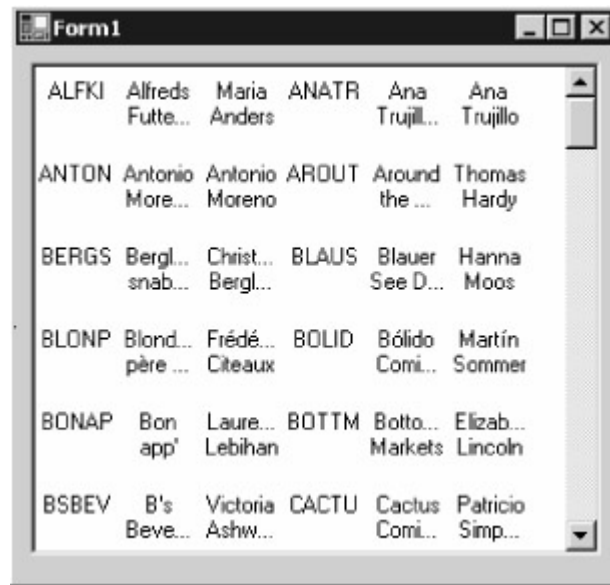


Trước khi bạn chạy form, bạn sẽ cần thêm một chuỗi con chứa mật khẩu cho sự kết nối cơ sở dữ liệu tới thuộc tính ConnectString của đối tượng SqlConnection của bạn. Cho sự cài đặt SQL Server trong máy của tôi, mật khẩu truy nhập cơ sở dữ liệu Northwind là sa, Và thuộc tính ConnectionString của tôi được đặt là :

data source=localhost;initial catalog=Northwind;persist security info=False;user  
id=sa;pwd=sa;workstation id=JMPRIce-DT1;packet size=4096

Chú ý chuỗi con pwd= sa trong chuỗi này để đặt mật khẩu.

Cuối cùng, chạy form của bạn bằng cách nhấn Ctl+ F5 trên bàn phím, hay chọn Debug ➤ Start Without Debugging. Hình 9.4 trình bày sự vận hành form.



Lưu dự án MyDataReader của bạn \_ chọn File ➤ Save All. Bạn sẽ sử dụng dự án này trong những chương sau . Nếu bạn muốn sử dụng dự án hoàn chỉnh DataReader hơn là sửa đổi dự án hiện hữu của bạn, đừng lo lắng: bạn sẽ có khả năng để sử dụng dự án hoàn chỉnh DataReader cũng trong những chương sau

## **TÓM LƯỢC:**

Trong chương này, bạn đã học cách sử dụng một đối tượng DataReader để đọc những kết quả trả lại từ cơ sở dữ liệu. Những đối tượng DataReader là bộ phận của những trình cung cấp có quản lý, và có ba lớp DataReader: SqlDataReader, OleDbDataReader, và OdbcDataReader. Bạn sử dụng một đối tượng của lớp SqlDataReader để đọc những hàng truy xuất từ một cơ sở dữ liệu SQL Server; một đối tượng của lớp OleDbDataReader để đọc những hàng từ bất kỳ cơ sở dữ liệu nào mà hỗ trợ OLE DB, như Oracle hay Access; và một đối tượng của lớp OdbcDataReader để đọc những hàng từ bất kỳ cơ sở dữ liệu nào mà hỗ trợ ODBC. Những đối tượng DataReader có thể được dùng để đọc chỉ những hàng trong một hướng xuôi, và bạn không thể sử dụng chúng sửa đổi cơ sở dữ liệu, nhưng chúng cung cấp sự truy xuất rất nhanh những hàng. Những đối tượng DataReader đóng vai trò một giải pháp đối với một đối tượng Dataset. Những đối tượng Dataset cho phép bạn lưu trữ một bản sao của những hàng từ cơ sở dữ liệu, và bạn có thể làm việc với bản sao này trong khi ngắt kết nối với cơ sở dữ liệu.

## **CHƯƠNG 10: SỬ DỤNG NHỮNG ĐỐI TƯỢNG DATASET ĐỂ LƯU TRỮ DỮ LIỆU:**

### **Tổng quan:**

Trong chương này, bạn sẽ học những chi tiết về việc sử dụng Dataset để lưu trữ những kết quả trả về từ cơ sở dữ liệu. Những đối tượng Dataset cho phép bạn lưu trữ một bản sao của thông tin từ cơ sở dữ liệu, và bạn có thể làm việc với bản sao này trong khi ngắt ra khỏi cơ sở dữ liệu. Không giống những đối tượng bộ cung cấp được quản lý Như SqlDataReader, Một Dataset là chung và bởi vậy nó làm việc với bất kỳ cơ sở dữ liệu nào. Một đối tượng Dataset cũng cho phép bạn sửa đổi những hàng và đọc những hàng trong bất kỳ trật tự nào, không giống Một SqlDataReader, chỉ cho phép bạn đọc những hàng theo một hướng xuôi tuần tự. Đây không phải nói là những đối tượng SqlDataReader là kém: như bạn đã học trong Chương 9, chúng cung cấp sự truy cập rất nhanh đến dữ liệu.

Bạn sẽ cũng học những chi tiết của việc sử dụng một đối tượng DataAdapter để đọc những hàng từ cơ sở dữ liệu vào trong một Dataset. DataAdapter là một bộ phận của những lớp cung cấp được quản lý, và có ba lớp DataAdapter: SqlDataAdapter, OleDbDataAdapter, Và OdbcDataAdapter. Bạn sử dụng một DataAdapter để sao chép những hàng từ cơ sở dữ liệu đến Dataset của bạn và cũng đẩy bất kỳ thay đổi nào Bạn thực hiện tới những hàng trong Dataset của bạn tới cơ sở dữ liệu. Bạn sẽ xem cách để thực hiện những thay đổi đến những hàng trong một Dataset và đẩy những thay đổi đó tới cơ sở dữ liệu trong Chương 11, "sử dụng những đối tượng

Dataset để Sửa đổi Dữ liệu." Trong chương này, bạn sẽ tập trung vào việc sao chép những hàng từ cơ sở dữ liệu và lưu trữ chúng trong một Dataset.

Những mục chính trong chương này:

- Lớp SqlDataAdapter
- Tạo ra một đối tượng SqlDataAdapter
  - Lớp Dataset
  - Tạo ra một đối tượng Dataset
  - Cư trú một đối tượng Dataset
  - Sử dụng phương thức Merge()\_ (Hòa trộn)
  - Viết và đọc XML sử dụng một đối tượng Dataset
  - Vẽ bản đồ những bảng và những cột
  - Tạo và sử dụng những lớp Dataset định kiểu mạnh .
  - Tạo ra một đối tượng DataAdapter và đối tượng Dataset sử dụng Visual Studio .NET

## **LỚP SQLDATAADAPTER:**

Bạn sử dụng một đối tượng của lớp SqlDataAdapter để đồng bộ hóa dữ liệu được cất giữ trong một đối tượng Dataset với một cơ sở dữ liệu SQL Server. Bạn sử dụng một đối tượng của lớp OleDbDataAdapter để đồng bộ hóa dữ liệu với một cơ sở dữ liệu hỗ trợ OLE DB, Như Oracle hay Access. Bạn sử dụng một đối tượng của lớp OdbcDataAdapter để đồng bộ hóa dữ liệu với một cơ sở dữ liệu hỗ trợ ODBC.

Ghi chú Dù lớp SqlDataAdapter được chỉ định riêng cho SQL Server, nhiều thuộc tính và phương thức trong lớp này cũng tương tự như trong lớp OleDbDataAdapter và OdbcDataAdapter.

**Bảng 10.1 trình bày một số thuộc tính của SqlDataAdapter.**

Thuộc tính	Kiểu	Mô tả
AcceptChangesDuringFill	bool	lấy hay đặt một giá trị bool chỉ định liệu phương thức AcceptChanges() có được gọi sau khi một đối tượng DataRow đã được thêm, điều chỉnh, hay loại bỏ trong một đối tượng DataTable. giá trị mặc định là true
ContinueUpdateOnError	bool	lấy hay đặt một giá trị bool chỉ định liệu có phải tiếp tục cập nhật cơ sở dữ liệu khi một lỗi xuất hiện. khi đặt là true, không có ngoại lệ nào được xuất ra khi một lỗi xuất hiện trong thời gian cập nhật một hàng. sự cập nhật của hàng được bỏ qua và thông tin lỗi được đặt trong thuộc tính RowError của DataRow mà gây ra lỗi. DataAdapter tiếp tục cập nhật những hàng kế tiếp. khi đặt là false, một ngoại lệ được ném khi một lỗi xuất hiện. giá trị mặc định là false.
DeleteCommand	SqlCommand	lấy hay gán một lệnh chứa một phát biểu SQL DELETE hoặc một lệnh gọi thủ tục lưu trữ để loại bỏ những hàng từ cơ sở dữ liệu.
InsertCommand	SqlCommand	lấy hay gán một lệnh chứa một phát biểu SQL INSERT hoặc một lệnh gọi thủ tục lưu trữ để thêm những hàng vào cơ sở dữ liệu.
MissingMappingAction	MissingMappingAction	lấy hay gán hành động để thực thi khi bảng hay cột đầu vào không có một bảng hay cột thích ứng trong tập hợp TableMappings. Những giá trị cho hành động này đến từ lớp liệt kê System.Data.MissingMappingAction với những lỗi

Thuộc tính	Kiểu	Mô tả
		<p>thành viên, Error, Ignore và Passthrough:</p> <ul style="list-style-type: none"> <li>. Lỗi có nghĩa là một SystemException được tung ra.</li> <li>. Ignore có nghĩa bảng hay cột được bỏ đi và không đọc.</li> <li>. Passthrough có nghĩa bảng hay cột được thêm vào Dataset với tên nguyên bản của nó.</li> </ul> <p>Giá trị mặc định là Passthrough.</p>
MissingSchemaAction	MissingSchemaAction	<p>lấy hay đặt hành động để thực thi khi cột đầu vào không có một cột thích ứng trong tập hợp cột của đối tượng DataTable.</p> <p>Những giá trị cho hoạt động này đến từ lớp liệt kê System.Data.MissingSchemaAction với những thành viên : Add, AddWithKey, Error, and Ignore:</p> <ul style="list-style-type: none"> <li>. Add có nghĩa cột được thêm vào DataTable.</li> <li>. AddWithKey có nghĩa thông tin cột và khóa chính được thêm vào DataTable.</li> <li>. Error có nghĩa một SystemException được ném.</li> <li>. Ignore có nghĩa cột được bỏ đi và không đọc.</li> </ul> <p>Mặc định là Add.</p>
SelectCommand	SqlCommand	lấy hay đặt một lệnh chứa một phát biểu SQL SELECT hay lệnh gọi thủ tục lưu trữ để truy xuất những hàng từ cơ sở dữ liệu.
TableMappings	DataTableMappingCollection	Lấy một tập hợp gián đồ bảng (DataTableMappingCollection) chứa ánh xạ giữa một bảng cơ sở dữ liệu và một đối tượng DataTable trong Dataset.
UpdateCommand	SqlCommand	lấy hay đặt một lệnh chứa những một phát biểu SQL UPDATE hoặc lệnh gọi thủ tục để sửa đổi những hàng trong cơ sở dữ liệu.

**Bảng 10.2 trình bày một số phương thức SqlDataAdapter.**

Phương thức	Kiểu trả về	Mô tả
Fill()	int	Quá tải . Đồng bộ hóa những hàng trong đối tượng Dataset để phù hợp với những hàng trong cơ sở dữ liệu. giá trị int được trả về bởi phương thức này là số lượng hàng được đồng bộ hóa trong Dataset với cơ sở dữ liệu.
FillSchema()	DataTable DataTable[]	Quá tải . Thêm một DataTable vào một đối tượng Dataset và định hình mô hình để phù hợp với cơ sở dữ liệu.
GetFillParameters()	IDataParameter[]	trả lại một mảng của bất kỳ tham số nào đặt cho phát biểu SQL SELECT.
Update()	int	Quá tải. gọi những phát biểu SQL tương ứng INSERT, UPDATE, hay DELETE hay lệnh gọi thủ tục lưu trữ (lưu trữ trong thuộc tính tương ứng InsertCommand, UpdateCommand, và DeleteCommand) cho mỗi hàng đã được thêm, được sửa đổi, hay được loại bỏ từ một đối tượng

Phương thức	Kiểu trả về	Mô tả
		DataTable. giá trị int được trả lại bởi phương thức này là số lượng hàng được cập nhật.

### **Bảng 10.3 trình bày một số sự kiện SqlDataAdapter.**

Sự kiện Event	EVENT HANDLER Đối tượng xử lý sự kiện	Mô tả
FillError	FillErrorEventHandler	phát ra khi một lỗi xuất hiện trong thời gian điền dữ liệu của một thao tác.
RowUpdating	RowUpdatingEventHandler	phát ra trước khi một hàng được thêm vào, điều chỉnh, hay xóa trong cơ sở dữ liệu.
RowUpdated	RowUpdatedEventHandler	Phát ra sau khi thêm, điều chỉnh, hay xóa một hàng trong cơ sở dữ liệu.

Bạn sẽ học cách sử dụng một số thuộc tính và những phương thức này để lưu trữ dữ liệu trong những đối tượng Dataset trong chương này. Bạn sẽ thấy cách sử dụng những thuộc tính, những phương thức và những sự kiện khác như thế nào trong Chương 11, trong đó bạn sẽ học sửa đổi dữ liệu trong đối tượng Dataset như thế nào, và sau đó gắn những sự sửa đổi đó tới cơ sở dữ liệu.

### **TAO MỘT ĐỐI TƯỢNG SQLDATAPDAPTER:**

Bạn tạo một đối tượng SqlDataAdapter sử dụng một trong số những bộ khởi tạo SqlDataAdapter sau đây :

- 1, SqlDataAdapter()
- 2, SqlDataAdapter(SqlCommand mySqlCommand)
- 3, SqlDataAdapter(string selectCommandString, SqlConnection mySqlConnection)
- 4, SqlDataAdapter(string selectCommandString, string connectionString)

Với

- \_ mySqlCommand đại diện đối tượng SqlCommand của các bạn.
- \_ selectCommandString chỉ định phát biểu SELECT hay lệnh gọi thủ tục lưu trữ của bạn .
- \_ mySqlConnection đại diện đối tượng kết nối (SqlConnection) của bạn.
- \_ connectionString chỉ định chuỗi kết nối của bạn để kết nối cơ sở dữ liệu.

Ví dụ sau đây sử dụng bộ khởi tạo SqlDataAdapter() để tạo ra một đối tượng SqlDataAdapter.

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
```

Trước khi sử dụng mySqlDataAdapter để lưu trữ một Dataset bạn phải gán thuộc tính SelectCommand của nó với một SqlCommand chứa một phát biểu SELECT hay lệnh gọi thủ tục lưu trữ. Ví dụ sau đây tạo ra một đối tượng SqlCommand với thuộc tính CommandText của nó được gán tới một phát biểu SELECT mà sẽ truy xuất năm hàng đầu từ bảng sản phẩm, và thuộc tính CommandText của mySqlDataAdapter được gán đến đối tượng SqlCommand vừa nói trên:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID";
mySqlDataAdapter.SelectCommand = mySqlCommand;
```

Ví dụ kế tiếp sử dụng bộ khởi tạo SqlDataAdapter(SqlCommand MySqlCommand) :

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter(mySqlCommand);
```

Ví dụ kế tiếp sử dụng bộ khởi tạo SqlDataAdapter(string SelectCommandString, SqlConnection mySqlConnection) :

```
SqlConnection mySqlConnection =  
new SqlConnection(  
"server=localhost;database=Northwind;uid=sa;pwd=sa");  
string selectCommandString =  
"SELECT TOP 10 ProductID, ProductName, UnitPrice "  
"FROM Products " +  
"ORDER BY ProductID";  
SqlDataAdapter mySqlDataAdapter =  
new SqlDataAdapter(selectCommandString, mySqlConnection);
```

Ví dụ cuối cùng sử dụng bộ khởi tạo SqlDataAdapter(string SelectCommandString, chuỗi connectionString) :

```
string selectCommandString =  
"server=localhost;database=Northwind;uid=sa;pwd=sa";  
string connectionString =  
"SELECT TOP 10 ProductID, ProductName, UnitPrice "  
"FROM Products " +  
"ORDER BY ProductID";  
SqlDataAdapter mySqlDataAdapter =  
new SqlDataAdapter(selectCommandString, connectionString);
```

### **Cảnh báo**

bộ khởi dựng này gây ra đối tượng SqlDataAdapter tạo ra một đối tượng SqlConnection riêng biệt khác. Đây điển hình là điều không ai mong muốn từ một tình huống thực thi vì việc mở một kết nối đến cơ sở dữ liệu sử dụng một đối tượng SqlConnection nắm giữ một thời gian tương đối dài. do đó bạn cần phải tránh sử dụng bộ khởi dựng SqlDataAdapter(string selectCommandString, String connectionString). Thay vào đó, bạn phải sử dụng một đối tượng SqlConnection hiện hữu với đối tượng SqlDataAdapter của các bạn.

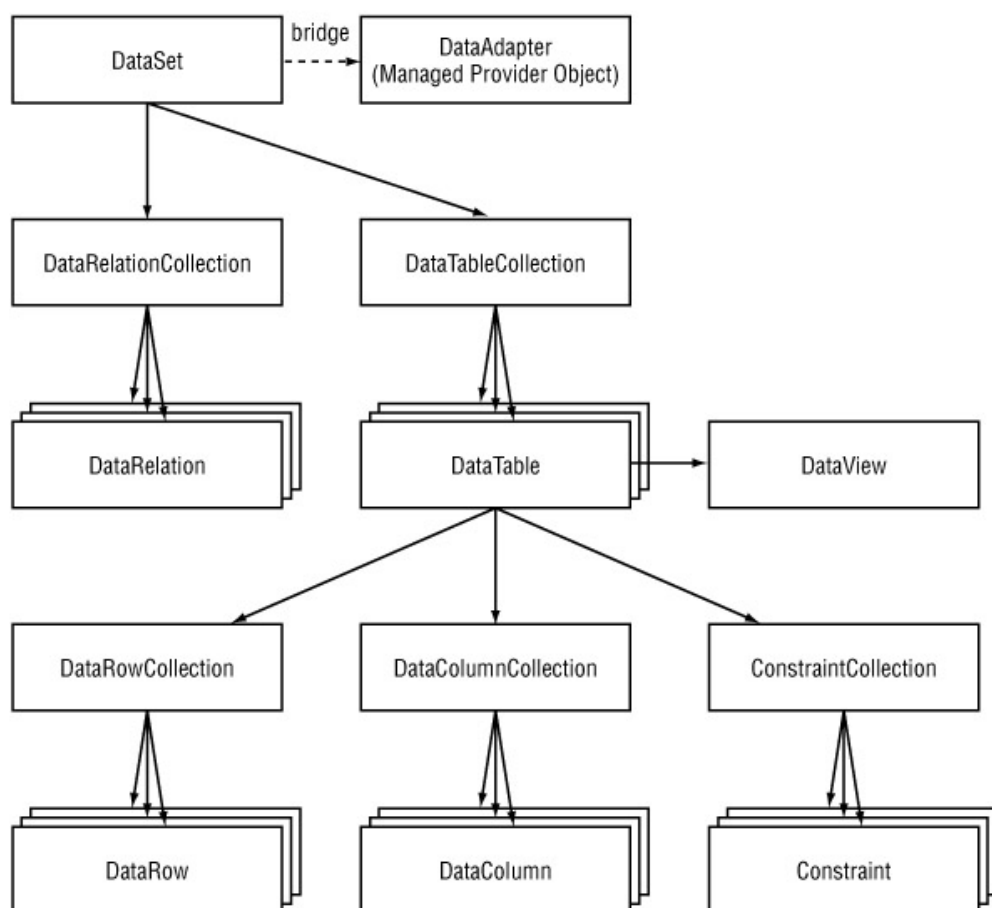
Một đối tượng DataAdapter không cất giữ những hàng: nó đơn thuần hành động như một ống dẫn giữa cơ sở dữ liệu và một đối tượng của lớp Dataset. Trong mục kế tiếp, bạn sẽ học về lớp Dataset.

## **LỚP DATASET:**

Bạn sử dụng một đối tượng của lớp Dataset để đại diện cho một bản sao cục bộ của thông tin được cất giữ trong cơ sở dữ liệu. Bạn có thể thực hiện những thay đổi tới bản sao cục bộ này trong Dataset của bạn và sau đó sau đó đồng bộ hóa những sự thay đổi đó với cơ sở dữ liệu thông qua một DataAdapter. Một Dataset có thể đại diện cho những cấu trúc cơ sở dữ liệu như những bảng, những hàng và những cột. Bạn có thể thậm chí thêm những sự ràng buộc vào những bảng cục bộ được cất giữ của bạn để giám sát những sự ràng buộc khóa chính và khóa ngoại.

Hình 10.1 trình bày Dataset và mối quan hệ của nó tới vài đối tượng bạn có thể lưu trữ bên trong nó. Như bạn có thể thấy từ hình này, bạn có thể lưu trữ nhiều đối tượng DataTable trong một Dataset, vân vân.



**Bảng 10.4: những thuộc tính Dataset**

Thuộc tính	Kiểu	Mô tả
CaseSensitive	bool	lấy hay gán một giá trị bool chỉ định liệu có phải những sự so sánh chuỗi bên trong những đối tượng DataTable thì phụ thuộc kiểu chữ.
DataSetName	string	lấy hay gán tên của đối tượng Dataset hiện thời.
DefaultViewManager	DataViewManager	lấy một view đã chỉnh lý của dữ liệu được lưu trữ trong đối tượng Dataset . Bạn sử dụng một view để lọc, tìm kiếm, và định hướng Dataset.
EnforceConstraints	bool	lấy hay gán một giá trị bool chỉ định liệu có phải những quy tắc ràng buộc được gán theo, sau khi cập nhật thông tin trong đối tượng Dataset.
ExtendedProperties	PropertyCollection	lấy một tập hợp (PropertyCollection) của thông tin người sử dụng. Bạn có thể sử dụng PropertyCollection để lưu giữ những chuỗi với bất kỳ thông tin bổ sung nào bạn muốn. Bạn sử dụng phương thức Add() thông qua ExtendedProperties để thêm một chuỗi.
HasErrors	bool	lấy một giá trị bool chỉ định liệu có phải có những lỗi trong bất kỳ hàng nào trong những bảng của đối tượng Dataset.
Locale	CultureInfo	lấy hay gán một đối tượng CultureInfo cho Dataset. một đối tượng CultureInfo chứa thông tin về một mặt đặc biệt bao gồm tên của nó, hệ thống ghi và lịch.
Namespace	string	lấy hay gán namespace cho Đối tượng Dataset. namespace là



Thuộc tính	Kiểu	Mô tả
		một chuỗi được sử dụng khi đọc và viết một tài liệu XML sử dụng những phương thức ReadXml(), WriteXml(), ReadXmlSchema(), và WriteXmlSchema() . namespace được dùng để gom nhóm những thuộc tính XML và những phần tử.
Prefix	string	lấy hay gán tiền tố XML cho namespace Dataset. Tiền tố được sử dụng trong một tài liệu XML để xác định những phần tử là thuộc về namespace của đối tượng Dataset.
Relations	DataRelationCollection	lấy một tập hợp những quan hệ (DataRelationCollection) mà cho phép sự dẫn hướng từ một bảng cha đến một bảng con. Một tập hợp DataRelationCollection chứa những đối tượng DataRelation.
Tables	DataTableCollection	lấy một tập hợp của bảng (DataTableCollection) mà chứa những đối tượng DataTable được lưu trữ trong Dataset.

**Bảng 10.5 trình bày một số phương thức của Dataset.**

Phương thức	Kiểu trả về	Mô tả
AcceptChanges()	void	Hiệu lực hóa tất cả những sự thay đổi đã thực hiện với đối tượng Dataset từ khi nó được tải hay từ lần cuối cùng phương thức AcceptChanges() được gọi.
BeginInit()	void	Sử dụng bởi trình thiết kế của Visual Studio .NET để khởi tạo một Dataset được dùng trong một form hay thành phần.
Clear()	void	loại bỏ tất cả các hàng từ tất cả các bảng trong đối tượng Dataset.
Clone()	DataSet	Sao chép cấu trúc của đối tượng Dataset và trả về bản sao này. bản sao này chứa tất cả những mô hình, những quan hệ, những sự ràng buộc và dữ liệu.
Copy()	DataSet	Copy cấu trúc và dữ liệu của đối tượng Dataset và trả lại bản sao này. bản sao chép chứa đựng tất cả những mô hình, những quan hệ, những sự ràng buộc, và dữ liệu.
EndInit()	void	sử dụng bởi trình thiết kế Visual Studio .NET để kết thúc sự khởi tạo của một Dataset được dùng trong một form hay thành phần.
GetChanges()	DataSet	Quá tải. lấy một bản sao của mọi thay đổi thực hiện tới đối tượng Dataset từ cuối sự tải hay từ lần cuối cùng phương thức AcceptChanges() được gọi .
GetXml()	string	trả về trình bày XML của dữ liệu được lưu giữ trong đối tượng Dataset.
GetXmlSchema()	string	Trả về bản trình bày XML của mô hình cho đối tượng Dataset.
HasChanges()	bool	Sự quá tải . Trả lại một giá trị bool chỉ báo liệu có phải đối tượng Dataset có thay đổi mà chưa được giao phó(cập nhật cơ sở dữ liệu).
Merge()	void	Quá tải. trộn Dataset này với đối tượng Dataset được chỉ định khác.
ReadXml()	XmlReadMode	Quá tải . tải dữ liệu từ một file XML vào trong đối tượng Dataset.
ReadXmlSchema()	void	Quá tải. nạp một mô hình từ một file XML sắp xếp vào trong đối tượng Dataset.
RejectChanges()	void	Hủy tất cả thay đổi được thực hiện với đối tượng Dataset từ khi nó được tạo ra hay từ lần cuối cùng phương thức AcceptChanges() được gọi .
Reset()	void	Đặt lại đối tượng Dataset tới trạng thái nguyên bản của nó.
WriteXml()	void	Quá tải. Viết dữ liệu từ đối tượng Dataset đến một file XML.
WriteXmlSchema()	void	Quá tải . Viết mô hình của đối tượng Dataset tới một file XML.

### **Bảng 10.6 trình bày một trong số những sự kiện Dataset.**

MergeFailed	MergeFailedEventHandler	phát ra khi một nỗ lực thêm một DataRow vào một Dataset mà giá trị khóa chính của DataRow này đã tồn tại trong Dataset.
-------------	-------------------------	---

Trong mục kế tiếp, bạn sẽ học cách tạo ra một đối tượng Dataset như thế nào.

### **TAO MỘT ĐỐI TƯỢNG Dataset:**

Bạn tạo ra một đối tượng Dataset sử dụng một trong những bộ khởi tạo Dataset sau đây:

- 1, DataSet()
- 2, DataSet(string dataSetNameString)

➤ dataSetNameString là chuỗi gán tới thuộc tính DataSetName của đối tượng dataset của bạn. Sự thiết đặt thuộc tính DataSetName là tùy chọn.

Ví dụ sau đây sử dụng bộ khởi dựng DataSet() để tạo một đối tượng Dataset.

```
DataSet myDataSet = new DataSet();
```

Ví dụ kế tiếp sử dụng bộ khởi tạo DataSet(string dataSetNameString) để tạo một đối tượng dataset.

```
DataSet myDataSet = new DataSet("myDataSet");
```

### **LƯU TRÚ MỘT ĐỐI TƯỢNG DATASET:**

Trong mục này bạn làm sẽ học cách lưu trữ một Dataset như thế nào\_ sử dụng phương thức Fill() của một đối tượng DataAdapter. Đặc biệt, bạn sẽ thấy cách lưu trữ một Dataset như thế nào, sử dụng:

- một phát biểu SELECT.
- Một phạm vi những hàng
- một thủ tục nội (stored procedure)

### **SỬ DỤNG MỘT PHÁT BIỂU SELECT:**

Trước khi bạn lưu trữ một Dataset đầu tiên bạn cần một Connection, một Command và Một DataAdapter:

```
SqlConnection mySqlConnection =  
new SqlConnection(  
"server=localhost;database=Northwind;uid=sa;pwd=sa");  
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
mySqlCommand.CommandText =  
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +  
"FROM Products " +  
"ORDER BY ProductID";  
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();  
mySqlDataAdapter.SelectCommand = mySqlCommand;  
DataSet myDataSet = new DataSet();  
mySqlConnection.Open();
```

Chú ý : Đối tượng MySqlCommand chứa một phát biểu SELECT để truy xuất ProductID, ProductName, và những cột UnitPrice của nam hàng đầu tiên từ bảng Products.

---

### Truy xuất từ nhiều bảng

Đương nhiên, bạn không được giới hạn một phát biểu SELECT mà truy xuất từ một bảng đơn. Bạn có thể sử dụng một phát biểu SELECT mà truy xuất từ nhiều bảng đang sử dụng một liên kết, tuy nhiên, Bạn cần phải diễn hình tránh thực hiện điều này vì một DataTable có nghĩa được dùng để lưu giữ những hàng từ một bảng cơ sở dữ liệu đơn.

---

Tiếp theo, để lưu trữ myDataSet với những hàng từ bảng Products, bạn gọi phương thức Fill() của mySqlDataAdapter. Chẳng hạn:

```
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Products");
```

Giá trị Int được trả lại bởi phương thức Fill() là số lượng hàng được đồng bộ hóa giữa Dataset và cơ sở dữ liệu qua DataAdapter. Trong ví dụ trước, int là số lượng hàng được sao chép từ bảng Product đến myDataSet và được gán là 5- tức số lượng hàng được truy xuất bởi phát biểu Select được trình bày trước.

Tham số đầu tiên gán tới phương thức Fill() là Dataset của bạn, và tham số thứ hai là một chuỗi chứa tên bạn muốn gán cho DataTable được tạo ra trong Dataset của bạn.

Ghi chú:

Tên bạn gán cho DataTable của bạn không phải trùng tên của bảng cơ sở dữ liệu. Bạn có thể sử dụng bất kỳ chuỗi nào, diễn hình bạn vẫn còn phải sử dụng tên giống như vậy, vì nó giúp bạn theo dõi cái mà bảng cơ sở dữ liệu đã sử dụng để lưu trữ DataTable này.

Khi bạn gọi phương thức Fill() lần đầu, những bước sau đây được thực hiện bởi ADO.NET:

1. phát biểu SELECT trong SqlCommand của bạn được thực thi.
2. một đối tượng DataTable mới được tạo ra trong Dataset của bạn.
3. DataTable của bạn được lưu trữ cùng với tập kết quả trả về bởi phát biểu SELECT.

Nếu bạn kết thúc với cơ sở dữ liệu sau khi gọi phương thức Fill() bạn cần phải đóng đối tượng kết nối của bạn sử dụng phương thức Close() .

```
mySqlConnection.Close();
```

Ghi chú:

phương thức Fill() sẽ thật sự mở và đóng Kết nối cho bạn nếu bạn không mở nó trước, tuy nhiên, tốt nhất là mở và đóng kết nối một cách tường minh bởi vì cách này là bộ thu dọn mà chương trình của bạn thực hiện. Đồng thời, nếu bạn gọi phương thức Fill() nhiều lần qua một đoạn ngắn của mã, bạn sẽ muốn giữ kết nối cơ sở dữ liệu mở và chỉ đóng nó khi bạn hoàn tất. Dataset bây giờ được lưu trữ với một DataTable tên là Products . Bạn có thể đọc Products DataTable từ myDataSet sử dụng ví dụ sau đây:

```
DataTable myDataTable = myDataSet.Tables["Products"];
```

Bạn có thể cũng đọc Products DataTable sử dụng một giá trị int :

```
DataTable myDataTable = myDataSet.Tables[0];
```

Bạn có thể trình bày những giá trị cột cho mỗi hàng ở myDataTable sử dụng vòng lặp foreach sau đây để duyệt qua những đối tượng DataRow được cất giữ trong myDataTable; chú ý sự sử dụng thuộc tính những hàng của

đối tượng myDataTable.

```
foreach (DataRow myDataRow in myDataTable.Rows)
{
    Console.WriteLine("ProductID = " + myDataRow["ProductID"]);
    Console.WriteLine("ProductName = " + myDataRow["ProductName"]);
    Console.WriteLine("UnitPrice = " + myDataRow["UnitPrice"]);
}
```

Những thuộc tính của hàng trả lại một đối tượng DataRowCollection cho phép bạn truy nhập tất cả những đối tượng DataRow cất giữ trong myDataTable. Bạn có thể đọc mỗi giá trị cột ở một DataRow sử dụng tên của cột; chẳng hạn, đọc giá trị cột ProductID bạn sử dụng myDataRow[ " ProductID "]. Bạn cũng có thể sử dụng số chỉ vị trí của cột; chẳng hạn, myDataRow[0] trả về giá trị cho cột đầu tiên . Đây là cột ProductID.

Bạn cũng có thể sử dụng mã sau đây để duyệt qua mọi DataTable, DataRow, Và DataColumn được cất giữ trong myDataSet:

```
foreach (DataTable myDataTable in myDataSet.Tables)
{
    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        foreach (DataColumn myDataColumn in myDataTable.Columns)
        {
            Console.WriteLine(myDataColumn + "= " +
                               myDataRow[myDataColumn]);
        }
    }
}
```

Chú ý bạn không cần biết những tên của những đối tượng DataTable hay DataColumn để trình bày chúng. Sự gọi phương thức WriteLine() sẽ hiển thị myDataColumn, nó trả về tên của cột, Và myDataRow[myDataColumn], sẽ trả lại giá trị cột cho dòng hiện tại.

Ghi nhớ :Bạn sẽ thấy những chi tiết của các lớp DataTable, DataRow, và DataColumn trong Chương 11.

### **Liệt kê 10.1 trình bày một chương trình sử dụng những ví dụ mã đã nêu trong mục này.**

```
/*
PopulateDataSetUsingSelect.cs illustrates how to populate a DataSet
object using a SELECT statement
*/

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

class PopulateDataSetUsingSelect
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        // create a SqlCommand object and set its CommandText property
        // to a SELECT statement that retrieves the top 5 rows from
```

```

// the Products table
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID";

// create a SqlDataAdapter object and set its SelectCommand
// property to the SqlCommand object
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;

// create a DataSet object
DataSet myDataSet = new DataSet();

// open the database connection
mySqlConnection.Open();

// use the Fill() method of the SqlDataAdapter object to
// retrieve the rows from the table, storing the rows locally
// in a DataTable of the DataSet object
Console.WriteLine("Retrieving rows from the Products table");
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Products");
Console.WriteLine("numberOfRows = " + numberOfRows);
// close the database connection
mySqlConnection.Close();

// get the DataTable object from the DataSet object
DataTable myDataTable = myDataSet.Tables["Products"];

// display the column values for each row in the DataTable,
// using a DataRow object to access each row in the DataTable
foreach (DataRow myDataRow in myDataTable.Rows)
{
    Console.WriteLine("ProductID = " + myDataRow["ProductID"]);
    Console.WriteLine("ProductName = " + myDataRow["ProductName"]);
    Console.WriteLine("UnitPrice = " + myDataRow["UnitPrice"]);
}
}
}

```

**The output from this program is as follows:**

```

Retrieving rows from the Products table
numberOfRows = 5
ProductID = 1
ProductName = Chai
UnitPrice = 18
ProductID = 2
ProductName = Chang
UnitPrice = 19
ProductID = 3
ProductName = Aniseed Syrup
UnitPrice = 10
ProductID = 4
ProductName = Chef Anton's Cajun Seasoning

```

```
UnitPrice = 22
ProductID = 5
ProductName = Chef Anton's Gumbo Mix
UnitPrice = 21.35
```

## **SỬ DỤNG MỘT DẢI NHỮNG HÀNG:**

Trong mục này bạn sẽ học cách lưu trữ một Dataset như thế nào với một phạm vi của những hàng. Bây giờ, phương thức Fill() bị quá tải và một bộ phận danh sách của những phương thức Fill() như sau:

```
int Fill(DataSet myDataSet)
int Fill(DataTable myDataTable)
int Fill(DataSet myDataSet, string dataTableName)
int Fill(DataSet myDataSet, int startRow, int numOfRows,
string dataTableName)
```

- dataTableName: chỉ định một chuỗi chứa đựng tên của DataTable để điền vào.
- startRow: là một int chỉ rõ vị trí của hàng trong tập hợp kết quả để đọc (bắt đầu tại 0).
- NumOfRows: là một int chỉ rõ số lượng hàng để đọc.

Phạm vi của những hàng từ startRow đến startRow+ numOfRows rồi được cất giữ trong DataTable. Int được trả về bởi phương thức Fill() là số lượng hàng truy xuất được từ cơ sở dữ liệu.

nếu bạn có thể thấy, phương thức Fill() cuối cùng cho phép bạn cư trú một Dataset với một phạm vi những hàng. Ví dụ sau đây cho thấy cách sử dụng của phương thức Fill() này để cất giữ một phạm vi của những hàng. Nó truy xuất năm hàng đầu tiên từ bảng Products, nhưng cất giữ chỉ ba hàng trong Products DataTable, Bắt đầu tại vị trí 1 (vì những hàng được ghi số bắt đầu tại 0 vị trí 1 tương ứng tới hàng thứ hai trong tập hợp kết quả trả về bởi phát biểu SELECT) :

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, 1, 3, "Products");
```

Biến numberOfRows được gán là 3 \_ là Số lượng hàng đã được lưu trữ với myDataSet . Một điều để nhớ là DataAdapter vẫn truy xuất tất cả năm hàng từ bảng Products, nhưng chỉ ba hàng thật sự được dùng để cư trú Dataset: còn hai hàng khác bị bỏ đi.

### **Liệt kê 10.2 cho thấy một chương trình sử dụng những ví dụ mã trình bày trong mục này.**

```
/*
PopulateDataSetUsingRange.cs illustrates how to populate a DataSet
object with a range of rows from a SELECT statement
*/

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

class PopulateDataSetUsingRange
{
```

```

public static void Main()
{
    SqlConnection mySqlConnection =
    new SqlConnection(
    "server=localhost;database=Northwind;uid=sa;pwd=sa");
    // create a SqlCommand object and set its CommandText property
    // to a SELECT statement that retrieves the top 5 rows from
    // the Products table
    SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
    mySqlCommand.CommandText =
    "SELECT TOP 5 ProductID, ProductName, UnitPrice " +
    "FROM Products " +
    "ORDER BY ProductID";
    SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
    mySqlDataAdapter.SelectCommand = mySqlCommand;
    DataSet myDataSet = new DataSet();
    mySqlConnection.Open();

    // use the Fill() method of the SqlDataAdapter object to
    // retrieve the rows from the table, storing a range of rows
    // in a DataTable of the DataSet object
    Console.WriteLine("Retrieving rows from the Products table");
    int numberOfRows = mySqlDataAdapter.Fill(myDataSet, 1, 3, "Products");
    Console.WriteLine("numberOfRows = " + numberOfRows);

    mySqlConnection.Close();

    DataTable myDataTable = myDataSet.Tables["Products"];

    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        Console.WriteLine("ProductID = " + myDataRow["ProductID"]);
        Console.WriteLine("ProductName = " + myDataRow["ProductName"]);
        Console.WriteLine("UnitPrice = " + myDataRow["UnitPrice"]);
    }
}

```

The output from this program is as follows:

```

Retrieving rows from the Products table
numberOfRows = 3
ProductID = 2
ProductName = Chang
UnitPrice = 19
ProductID = 3
ProductName = Aniseed Syrup
UnitPrice = 10
ProductID = 4
ProductName = Chef Anton's Cajun Seasoning
UnitPrice = 22

```

## **SỬ DỤNG MỘT THỦ TỤC LƯU TRỮ:**

Bạn cũng có thể cư trú một đối tượng Dataset sử dụng một thủ tục lưu trữ để trả về một tập hợp kết quả . Chẳng hạn, cơ sở dữ liệu Northwind Người phục vụ SQL chứa một thủ tục lưu trữ được gọi là CustOrderHist() nó trả



về những sản phẩm và số lượng sản phẩm được đặt bởi một khách hàng. CustomerID của khách hàng được sử dụng như một tham số cho CustOrderHist().

### **Danh sách 10.3 trình bày định nghĩa của thủ tục lưu trữ CustOrderHist().**

```
CREATE PROCEDURE CustOrderHist @CustomerID nchar(5)
AS
SELECT ProductName, Total=SUM(Quantity)
FROM Products P, [Order Details] OD, Orders O, Customers C
WHERE C.CustomerID = @CustomerID
AND C.CustomerID = O.CustomerID
AND O.OrderID = OD.OrderID
AND OD.ProductID = P.ProductID
GROUP BY ProductName
```

Ghi chú: Bạn không cần tự tạo ra thủ tục lưu trữ CustOrderHist() . Nó đã được định nghĩa trong cơ sở dữ liệu Northwind.

Việc gọi CustOrderHist() và việc lưu trữ một Dataset với tập hợp kết quả được trả về là điều dễ dàng. Chẳng hạn, mã sau đây tạo ra một đối tượng SqlCommand, đặt đối tượng CommandText của với phát biểu thực thi sự mà gọi thủ tục lưu trữ CustOrderHist(), và đặt tham số @CustomerID là ALFKI ( Những tham số nói qua trong Chương 8, " Những lệnh thực thi cơ sở dữ liệu "):

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"EXECUTE CustOrderHist @CustomerID";
mySqlCommand.Parameters.Add(
"@CustomerID", SqlDbType.NVarChar, 5).Value = "ALFKI";
```

Bạn rồi sử dụng mã tương tự như trình bày trong mục trước để lưu trữ một Dataset với tập hợp kết quả được trả về bởi CustOrderHist():

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "CustOrderHist");
mySqlConnection.Close();
```

DataTable CustOrderHist chứa bên trong myDataSet được lưu trữ với tập hợp kết quả trả về bởi thủ tục CustOrderHist() .

### **Danh sách 10.4 trình bày một chương trình sử dụng những ví dụ mã sử dụng trong mục này.**

```
/*
PopulateDataSetUsingProcedure.cs illustrates how to populate a
DataSet object using a stored procedure
*/

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

class PopulateDataSetUsingProcedure
{
```

```

public static void Main()
{
    SqlConnection mySqlConnection =
        new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa");

    // create a SqlCommand object and set its CommandText property
    // to call the CustOrderHist() stored procedure
    SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
    mySqlCommand.CommandText =
        "EXECUTE CustOrderHist @CustomerID";
    mySqlCommand.Parameters.Add(
        "@CustomerID", SqlDbType.NVarChar, 5).Value = "ALFKI";

    SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
    mySqlDataAdapter.SelectCommand = mySqlCommand;
    DataSet myDataSet = new DataSet();
    mySqlConnection.Open();
    Console.WriteLine("Retrieving rows from the CustOrderHist() Procedure");
    int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "CustOrderHist");
    Console.WriteLine("numberOfRows = " + numberOfRows);
    mySqlConnection.Close();

    DataTable myDataTable = myDataSet.Tables["CustOrderHist"];
    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        Console.WriteLine("ProductName = " + myDataRow["ProductName"]);
        Console.WriteLine("Total = " + myDataRow["Total"]);
    }
}

```

**Đầu ra chương trình này như sau:**

```

Retrieving rows from the CustOrderHist() Procedure
numberOfRows = 11
ProductName = Aniseed Syrup
Total = 6
ProductName = Chartreuse verte
Total = 21
ProductName = Escargots de Bourgogne
Total = 40
ProductName = Flotemysost
Total = 20
ProductName = Grandma's Boysenberry Spread
Total = 16
ProductName = Lakkalikööri
Total = 15
ProductName = Original Frankfurter grüne Soße
Total = 2
ProductName = Raclette Courdavault
Total = 15
ProductName = Rössle Sauerkraut
Total = 17
ProductName = Spegesild
Total = 2

```

ProductName = Vegie-spread  
Total = 20

## **LƯU TRỮ MỘT DATASET VỚI NHIỀU ĐỐI TƯỢNG DATATABLE:**

Bạn có thể lưu trữ một Dataset với nhiều đối tượng DataTable. Bạn có thể muốn làm điều đó khi bạn cần truy cập thông tin được cất giữ trong nhiều bảng trong cơ sở dữ liệu. Bạn có thể sử dụng bất kỳ kỹ thuật nào sau đây để lưu trữ một Dataset với nhiều đối tượng DataTable :

- Sử dụng nhiều phát biểu SELECT trong cùng một SelectCommand .
- Thay đổi thuộc tính CommandText của SelectCommand trước mỗi lệnh gọi phương thức Fill().
- Sử dụng nhiều đối tượng DataAdapter để lưu trữ cùng một Dataset .

Chúng ta hãy xem xét tất cả những kỹ thuật này.

## **SỬ DỤNG NHIỀU PHÁT BIỂU SELECT TRONG CÙNG MỘT SelectedCommand:**

Những ví dụ sau đây đặt thuộc tính CommandText của một SqlCommand với hai phát biểu SELECT riêng biệt

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

```
mySqlCommand.CommandText =  
"SELECT TOP 2 ProductID, ProductName, UnitPrice " +  
"FROM Products " +  
"ORDER BY ProductID;" +  
"SELECT CustomerID, CompanyName " +  
"FROM Customers " +  
"WHERE CustomerID = 'ALFKI';";
```

Chú ý: mỗi phát biểu SELECT được tách ra bởi một dấu chấm phẩy (;). Khi những phát biểu SELECT này chạy, hai tập hợp kết quả được trả về: một chứa hai hàng từ bảng Products, tập hợp kết quả thứ hai chứa một hàng từ bảng Customers. Hai tập hợp kết quả này được cất giữ trong những đối tượng DataTable riêng biệt bởi mã sau đây:

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();  
mySqlDataAdapter.SelectCommand = mySqlCommand;  
DataSet myDataSet = new DataSet();  
mySqlConnection.Open();  
int numberOfRows = mySqlDataAdapter.Fill(myDataSet);  
mySqlConnection.Close();
```

Chú ý: sự sử dụng phương thức Fill(mydataset), không chỉ định tên của DataTable sẽ được tạo ra. Thay vào đó, tên của hai đối tượng DataTable (dùng để cất giữ những tập kết quả) được tự động gán theo mặc định là Table và Table1. Table cất giữ tập hợp kết quả từ Bảng Products, và Table1 cất giữ kết quả từ những Bảng Customers.

Tên của một đối tượng DataTable được cất giữ trong thuộc tính TableName của nó, và bạn có thể thay đổi. Chẳng hạn, mã sau đây thay đổi tên của Table của Dataset thành Products và Table1 của Dataset thành Customers:

```
myDataSet.Tables["Table"].TableName = "Products";  
myDataSet.Tables["Table1"].TableName = "Customers";
```

Danh sách 10.5 trình bày một chương trình sử dụng những ví dụ mã hiện sử dụng trong mục này.

/\*

MutlipleDataTables.cs illustrates how to populate a DataSet  
with multiple DataTable objects using multiple SELECT statements  
\*/

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

class MultipleDataTables
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        // create a SqlCommand object and set its CommandText property
        // to mutliple SELECT statements
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT TOP 2 ProductID, ProductName, UnitPrice " +
            "FROM Products " +
            "ORDER BY ProductID;" +
            "SELECT CustomerID, CompanyName " +
            "FROM Customers " +
            "WHERE CustomerID = 'ALFKI'";

        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        int numberOfRows = mySqlDataAdapter.Fill(myDataSet);
        Console.WriteLine("numberOfRows = " + numberOfRows);
        mySqlConnection.Close();

        // change the TableName property of the DataTable objects
        myDataSet.Tables["Table"].TableName = "Products";
        myDataSet.Tables["Table1"].TableName = "Customers";

        foreach (DataTable myDataTable in myDataSet.Tables)
        {
            Console.WriteLine("\nReading from the " +
                myDataTable.TableName + "DataTable");
            foreach (DataRow myDataRow in myDataTable.Rows)
            {
                foreach (DataColumn myDataColumn in myDataTable.Columns)
                {
                    Console.WriteLine(myDataColumn + "=" +
                        myDataRow[myDataColumn]);
                }
            }
        }
    }
}
```

---

### Đầu ra từ chương trình này như sau

```
numberOfRows = 3
```

```
Reading from the Products DataTable
```

```
ProductID = 1
```

```
ProductName = Chai
```

```
UnitPrice = 18
```

```
ProductID = 2
```

```
ProductName = Chang
```

```
UnitPrice = 19
```

```
Reading from the Customers DataTable
```

```
CustomerID = ALFKI
```

```
CompanyName = Alfreds Futterkiste
```

### **THAY ĐỔI THUỘC TÍNH COMMANDTEXT CỦA SELECTCOMMAND:**

Bạn cũng có thể cư trú một Dataset với nhiều đối tượng DataTable bởi thay đổi thuộc tính CommandText của SelectCommand cho đối tượng DataAdapter của bạn trước trước mỗi lệnh gọi phương thức Fill(). Đầu tiên, mã sau đây cư trú một Dataset với một DataTable chứa hai hàng từ bảng Products:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 2 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Products");
```

Đối tượng myDataSet bây giờ chứa một DataTable có tên Products.

Tiếp theo, thuộc tính CommandText cho SelectCommand của mySqlDataAdapter được thay đổi tới một phát biểu SELECT để truy xuất những hàng từ bảng Customers, và phương thức Fill() được gọi lần nữa:

```
mySqlDataAdapter.SelectCommand.CommandText =
"SELECT CustomerID, CompanyName " +
"FROM Customers " +
"WHERE CustomerID = 'ALFKI'";
numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();
```

Đối tượng MyDataSet bây giờ chứa một DataTable bổ sung đặt tên Customers.

### Liệt kê 10.6 trình bày một chương trình sử dụng những ví dụ mã dùng trong mục này.

```
/*
MultipleDataTables2.cs illustrates how to populate a DataSet
object with multiple DataTable objects by changing the
CommandText property of a DataAdapter object's SelectCommand
*/

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

```

class MultipleDataTables2
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT TOP 2 ProductID, ProductName, UnitPrice " +
            "FROM Products " +
            "ORDER BY ProductID";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Products");
        Console.WriteLine("numberOfRows = " + numberOfRows);

        // change the CommandText property of the SelectCommand
        mySqlDataAdapter.SelectCommand.CommandText =
            "SELECT CustomerID, CompanyName " +
            "FROM Customers " +
            "WHERE CustomerID = 'ALFKI'";
        numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Customers");
        Console.WriteLine("numberOfRows = " + numberOfRows);

        mySqlConnection.Close();

        foreach (DataTable myDataTable in myDataSet.Tables) {
            Console.WriteLine("\nReading from the " +
                myDataTable.TableName + "DataTable");
            foreach (DataRow myDataRow in myDataTable.Rows)
            {
                foreach (DataColumn myDataColumn in myDataTable.Columns)
                {
                    Console.WriteLine(myDataColumn + " = " +
                        myDataRow[myDataColumn]);
                }
            }
        }
    }
}

```

**Đầu ra từ chương trình này như sau:**

```

numberOfRows = 2
numberOfRows = 1

```

```

Reading from the Products DataTable
ProductID = 1
ProductName = Chai
UnitPrice = 18
ProductID = 2
ProductName = Chang

```

UnitPrice = 19

Reading from the Customers DataTable

CustomerID = ALFKI

CompanyName = Alfreds Futterkiste

## **SỬ DỤNG NHIỀU ĐỐI TƯỢNG DATAADAPTER ĐỂ LƯU TRỮ CÙNG MỘT ĐỐI TƯỢNG DATASET.**

Bạn cũng có thể cư trú cùng một Dataset với nhiều đối tượng DataTable sử dụng những đối tượng DataAdapter khác nhau. Chẳng hạn, giả thiết bạn đã có một Dataset có tên myDataSet được cư trú sử dụng một SqlDataAdapter có tên mySqlDataAdapter, Và myDataSet này đang chứa một DataTable đặt tên Products. Ví dụ sau đây tạo ra SqlDataAdapter khác và sử dụng để cư trú myDataSet với DataTable khác tên Customers

```
SqlDataAdapter mySqlDataAdapter2 = new SqlDataAdapter();
mySqlDataAdapter2.SelectCommand = mySqlCommand;
mySqlDataAdapter2.SelectCommand.CommandText =
    "SELECT CustomerID, CompanyName " +
    "FROM Customers " +
    "WHERE CustomerID = 'ALFKI'";
numberOfRows = mySqlDataAdapter2.Fill(myDataSet, "Customers");
```

### **Liệt kê 10.7 trình bày một chương trình sử dụng những ví dụ mã đã nêu trong mục này**

```
/*
    utipleDataTables3.cs illustrates how to populate a DataSet
    bject with multiple DataTable objects using multiple
    ataAdapter objects to populate the same DataSet object
*/

using System;
using System.Data;
using System.Data.SqlClient;
class MultipleDataTables3

    Pblic static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT TOP 2 ProductID, ProductName, UnitPrice " +
            "FROM Products " +
            "ORDER BY ProductID";
        SqlDataAdapter mySqlDataAdapter1 = new SqlDataAdapter();
        mySqlDataAdapter1.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        int numberOfRows = mySqlDataAdapter1.Fill(myDataSet, "Products");
        Console.WriteLine("numberOfRows = " + numberOfRows);

        // create another DataAdapter object
        SqlDataAdapter mySqlDataAdapter2 = new SqlDataAdapter();
        mySqlDataAdapter2.SelectCommand = mySqlCommand;
```



```

mySqlDataAdapter2.SelectCommand.CommandText =
"SELECT CustomerID, CompanyName " +
"FROM Customers " +
"WHERE CustomerID = 'ALFKI'";
numberOfRows = mySqlDataAdapter2.Fill(myDataSet, "Customers");
Console.WriteLine("numberOfRows = " + numberOfRows);
mySqlConnection.Close();

foreach (DataTable myDataTable in myDataSet.Tables)
{
    Console.WriteLine("\nReading from the " +
myDataTable.TableName + "DataTable");
    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        foreach (DataColumn myDataColumn in myDataTable.Columns)
        {
            Console.WriteLine(myDataColumn + "=" +
myDataRow[myDataColumn]);
        }
    }
}
}

```

#### Đầu ra từ chương trình này như sau:

```

numberOfRows = 2
numberOfRows = 1

```

```

Reading from the Products DataTable
ProductID = 1
ProductName = Chai
UnitPrice = 18
ProductID = 2
ProductName = Chang
UnitPrice = 19

```

```

Reading from the Customers DataTable
CustomerID = ALFKI
CompanyName = Alfreds Futterkiste

```

### **TRỘN NHỮNG ĐỐI TƯỢNG DATAROW, DATASET, VÀ DATATABLE VÀO TRONG DATASET KHÁC:**

Trong mục này, bạn sẽ học cách sử dụng phương thức Merge() để trộn những đối tượng DataRow, Dataset, và DataTable vào trong Dataset khác như thế nào. Bạn có thể sẽ muốn làm điều này khi Bạn có nhiều nguồn dữ liệu; ví dụ bạn có nhiều dữ liệu từ nhiều văn phòng khu vực được gửi đến trụ sở chính, và bạn cần trộn tất cả dữ liệu đó vào trong một Dataset.

#### Phương thức Merge() quá tải như sau:

```

void Merge(DataRow[] myDataRows)
void Merge(DataSet myDataSet)
void Merge(DataTable myDataTable)
void Merge(DataSet myDataSet, bool preserveChanges)
void Merge(DataRow[] myDataRows, bool preserveChanges,
MissingSchemaAction myMissingSchemaAction)

```

```
void Merge(DataSet myDataSet, bool preserveChanges,
MissingSchemaAction myMissingSchemaAction)
void Merge(DataTable myDataTable, bool preserveChanges,
MissingSchemaAction myMissingSchemaAction)
```

Với:

PreserveChanges: chỉ rõ liệu có phải những sự thay đổi trong Dataset hiện thời (Dataset với phương thức Merge() được gọi) sẽ được giữ lại.

MyMissingSchemaAction : chỉ định hoạt động để nắm bắt khi Dataset hiện thời không có những bảng hay những cột giống như trong DataRow, Dataset, hay DataTable đang được trộn vào trong Dataset này.

Bạn gán myMissingSchemaAction với một trong số những hằng số được định nghĩa trong lớp liệt kê *System.Data.MissingSchemaAction*.

Bảng 10.7 cho thấy những hằng số định nghĩa trong lớp liệt kê MissingSchemaAction.

#### **Bảng 10.7: Liệt kê những phần tử MissingSchemaAction**

Hằng số	Mô tả
Add	Cột hay bảng được thêm vào Dataset hiện thời. Add là mặc định.
AddWithKey	Thông tin cột và khóa chính được thêm vào Dataset hiện thời.
Error	Một SystemException được tung ra.
Ignore	Cột hay bảng được bỏ đi và không phải đọc.

#### **Danh sách 10.8 minh họa cách sử dụng phương thức Merge().**

```
/*
Merge.cs illustrates how to use the Merge() method
*/

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

class Merge
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // populate myDataSet with three rows from the Customers table
        mySqlCommand.CommandText =
            "SELECT CustomerID, CompanyName, ContactName, Address " +
            "FROM Customers " +
            "WHERE CustomerID IN ('ALFKI', 'ANATR', 'ANTON')";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.Fill(myDataSet, "Customers");
```

```

// populate myDataSet2 with two rows from the Customers table
mySqlCommand.CommandText =
"SELECT CustomerID, CompanyName, ContactName, Address " +
"FROM Customers " +
"WHERE CustomerID IN ('AROUT', 'BERGS')";
DataSet myDataSet2 = new DataSet();
mySqlDataAdapter.Fill(myDataSet2, "Customers2");

// populate myDataSet3 with five rows from the Products table
mySqlCommand.CommandText =
"SELECT TOP 5 ProductID, ProductName, UnitPrice " +
"FROM Products " +
"ORDER BY ProductID";
DataSet myDataSet3 = new DataSet();
mySqlDataAdapter.Fill(myDataSet3, "Products");

mySqlConnection.Close();

// merge myDataSet2 into myDataSet
myDataSet.Merge(myDataSet2);

// merge myDataSet3 into myDataSet
myDataSet.Merge(myDataSet3, true, MissingSchemaAction.Add);

// display the rows in myDataSet
foreach (DataTable myDataTable in myDataSet.Tables)
{
    Console.WriteLine("\nReading from the " + myDataTable + "DataTable");
    foreach (DataRow myDataRow in myDataTable.Rows)
    {
        foreach (DataColumn myDataColumn in myDataTable.Columns)
        {
            Console.WriteLine(myDataColumn + "= " +
myDataRow[myDataColumn]);
        }
    }
}
}

```

**Đầu ra từ chương trình này như sau:**

```

Reading from the Customers DataTable
CustomerID = ALFKI
CompanyName = Alfreds Futterkiste
ContactName = Maria Anders
Address = Obere Str. 57
CustomerID = ANATR
CompanyName = Ana Trujillo3 Emparedados y helados
ContactName = Ana Trujillo
Address = Avda. de la Constitución 2222
CustomerID = ANTON
CompanyName = Antonio Moreno Taquería
ContactName = Antonio Moreno
Address = Mataderos 2312

```

Reading from the Customers2 DataTable

CustomerID = AROUT

CompanyName = Around the Horn

ContactName = Thomas Hardy

Address = 120 Hanover Sq.

CustomerID = BERGS

CompanyName = Berglunds snabbköp

ContactName = Christina Berglund

Address = Berguvsvägen 8

Reading from the Products DataTable

ProductID = 1

ProductName = Chai

UnitPrice = 18

ProductID = 2

ProductName = Chang

UnitPrice = 19

ProductID = 3

ProductName = Aniseed Syrup

UnitPrice = 10

ProductID = 4

ProductName = Chef Anton's Cajun Seasoning

UnitPrice = 22

ProductID = 5

ProductName = Chef Anton's Gumbo Mix

UnitPrice = 21.35

## **VIẾT VÀ ĐỌC XML SỬ DỤNG ĐỐI TƯỢNG DATASET:**

XML là một định dạng tiện lợi cho sự chuyển dời thông tin qua lại. Bạn có thể viết ra nội dung của những đối tượng DataTable được chứa trong một Dataset thành một file XML sử dụng phương thức WriteXml() . file XML được viết bởi phương thức này chứa đựng những tên cột và những giá trị của DataTable .

Bạn có thể viết ra mô hình của một đối tượng Dataset thành một file XML sử dụng phương thức WriteXmlSchema() . file XML được viết bởi phương pháp này chứa đựng cấu trúc của những đối tượng DataTable lưu trữ trong Dataset. Bạn cũng có thể lấy XML trong một Dataset sử dụng phương thức GetXml() , để trả về XML trong một chuỗi.

Bạn có thể đọc nội dung của những đối tượng DataTable trong một file XML vào trong một Dataset sử dụng phương thức ReadXml() . Bạn cũng có thể đọc mô hình chứa trong một file XML sử dụng phương thức ReadXmlSchema() .

### **Ghi chú**

máy chủ phục vụ SQL cũng chứa chức năng XML dựng sẵn rộng lớn, mà bạn sẽ học Trong Chương 16, "sử dụng Hỗ trợ XML của máy chủ phục vụ SQL ."

## **SỬ DỤNG PHƯƠNG THỨC WriteXml():**

Cho rằng bạn có một đối tượng Dataset đặt tên MyDataSet. Giả thiết rằng myDataSet có một DataTable chứa những cột CustomerID, CompanyName, ContactName, Và Address thuộc hai hàng đầu tiên từ bảng Customers. Mã sau đây cho thấy điều này

```

SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT TOP 2 CustomerID, CompanyName, ContactName, Address " +
    "FROM Customers " +
    "ORDER BY CustomerID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
Console.WriteLine("Retrieving rows from the Customers table");
mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();

```

Bạn có thể viết ra nội dung của myDataSet thành một file XML sử dụng phương thức WriteXml() . thí dụ:

```
myDataSet.WriteXml("myXmlFile.xml");
```

mã này viết ra một file XML có tên myXmlFile.xml, Như trình bày trong Danh sách 10.9.

#### **Danh sách 10.9: MYXMLFILE XML**

```

<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Customers>
    <CustomerID>ALFKI</CustomerID>
    <CompanyName>Alfreds Futterkiste</CompanyName>
    <ContactName>Maria Anders</ContactName>
    <Address>Obere Str. 57</Address>
  </Customers>
  <Customers>
    <CustomerID>ANATR</CustomerID>
    <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
    <ContactName>Ana Trujillo</ContactName>
    <Address>Avda. de la Constitución 2222</Address>
  </Customers>
</NewDataSet>

```

Như bạn có thể thấy, file này chứa những cột thuộc những hàng truy xuất từ bảng Customers.

#### **Phương thức WriteXml() được tải như sau:**

```

void WriteXml(Stream myStream);
void WriteXml(string fileName);
void WriteXml(TextWriter myTextWriter);
void WriteXml(XmlWriter myXmlWriter);
void WriteXml(stream myStream, XmlWriteMode myXmlWriteMode);
void WriteXml(string fileName, XmlWriteMode myXmlWriteMode);
void WriteXml(TextWriter myTextWriter, XmlWriteMode myXmlWriteMode);
void WriteXml(XmlWriter myXmlWriter, XmlWriteMode myXmlWriteMode);

```

với myXmlWriteMode là một hằng số từ lớp liệt kê System.Data.XmlWriteMode nó chỉ định viết dữ liệu và mô hình XML như thế nào

#### **Bảng 10.8 cho thấy những hằng số định nghĩa trong kiểu liệt kê XmlWriteMode.**

Hằng số	Mô tả
DiffGram	Viết ra DataSet như một DiffGram, chứa những giá trị ban đầu và những sự thay đổi tới những giá trị này để làm chúng lưu hành. Bạn có thể phát sinh một DiffGram mà chỉ chứa những sự thay đổi bởi gọi phương thức GetChanges() của DataSet của bạn, và sau đó gọi WriteXml().
IgnoreSchema	Chỉ viết ra dữ liệu trong DataSet, không viết ra mô hình. IgnoreSchema là mặc định.
WriteSchema	Viết ra mô hình trong DataSet.

Ví dụ sau đây cho thấy sự sử dụng hằng số *XmlWriteMode.WriteSchema*

```
myDataSet.WriteXml("myXmlFile2.xml", XmlWriteMode.WriteSchema);
```

Mã này ghi ra một file XML có tên myXmlFile2.xml, Như trình bày trong Danh sách 10.10.

#### **Danh sách 10.10: MYXMLFILE2. XML**

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <xsd:schema id="NewDataSet" targetNamespace="" xmlns=""
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:element name="NewDataSet" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="Customers">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="CustomerID" type="xsd:string" minOccurs="0" />
                <xsd:element name="CompanyName" type="xsd:string" minOccurs="0" />
                <xsd:element name="ContactName" type="xsd:string" minOccurs="0" />
                <xsd:element name="Address" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <Customers>
    <CustomerID>ALFKI</CustomerID>
    <CompanyName>Alfreds Futterkiste</CompanyName>
    <ContactName>Maria Anders</ContactName>
    <Address>Obere Str. 57</Address>
  </Customers>
  <Customers>
    <CustomerID>ANATR</CustomerID>
    <CompanyName>Ana Trujillo3 Emparedados y helados</CompanyName>
    <ContactName>Ana Trujillo</ContactName>
    <Address>Avda. de la Constitución 2222</Address>
  </Customers>
</NewDataSet>
```

Như bạn có thể thấy, file này chứa định nghĩa mô hình cho những cột được dùng trong phát biểu nguyên thủy SELECT, cũng như những giá trị cột cho những hàng được truy xuất.

## **SỬ DỤNG PHƯƠNG THỨC *WriteXmlSchema()*:**

Bạn có thể viết ra mô hình của myDataSet tới một file XML sử dụng phương thức WriteXmlSchema() . Chẳng hạn:

```
myDataSet.WriteXmlSchema("myXmlSchemaFile.xml");
```

Mã này viết ra một file XML có tên myXmlSchemaFile.xml, như trình bày trong Danh sách 10.11.

### **Danh sách 10.11: MYXMLSCHEMAFILE.XML**

```
<?xml version="1.0" standalone="yes"?>
<xsd:schema id="NewDataSet" targetNamespace="" xmlns=""
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xsd:element name="NewDataSet" msdata:IsDataSet="true">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="Customers">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="CustomerID" type="xsd:string" minOccurs="0" />
              <xsd:element name="CompanyName" type="xsd:string" minOccurs="0" />
              <xsd:element name="ContactName" type="xsd:string" minOccurs="0" />
              <xsd:element name="Address" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Như bạn có thể thấy, file này chứa định nghĩa mô hình cho những cột được truy xuất từ bảng Customers bởi phát biểu nguyên thủy SELECT.

## **SỬ DỤNG PHƯƠNG THỨC *ReadXml()***

Bạn có thể đọc nội dung của một file XML vào trong một đối tượng Dataset sử dụng phương thức ReadXml() . Phương thức này đọc những hàng và những cột từ file XML vào trong những đối tượng DataTable của Dataset. Chẳng hạn, phát biểu sau đây sử dụng phương thức ReadXml() để đọc file XML myXmlFile.xml được viết trước đó bởi phương thức WriteXml() :

```
myDataSet.ReadXml("myXmlFile.xml");
```

### **Phương thức ReadXml() được tải như sau:**

```
void ReadXml(Stream myStream);
void ReadXml(string fileName);
void ReadXml(TextReader myTextReader);
void ReadXml(XmlReader myXmlReader);
```



```

void ReadXml(stream myStream, XmlReadMode myXmlReadMode);
void ReadXml(string fileName, XmlReadMode myXmlReadMode);
void ReadXml(TextReader myTextReader, XmlReadMode myXmlReadMode);
void ReadXml(XmlReader myXmlReader, XmlReadMode myXmlReadMode);

```

với myXmlReadMode là một hằng số từ sự liệt kê System.Data.XmlReadMode nó chỉ định cách để đọc Dữ liệu và mô hình XML .

Bảng 10.9 cho thấy những hằng số định nghĩa trong kiểu liệt kê XmlReadMode.

**Bảng 10.9: những thành viên Liệt kê XmlReadMode**

Hằng số	Mô tả
Auto	<p>Đọc file XML trong một thái độ thích hợp:</p> <ul style="list-style-type: none"> <li>■ Nếu file XML chứa một DiffGram, thì XmlReadMode được gán tới DiffGram.</li> <li>■ Nếu Dataset đã chứa một mô hình hay file XML chứa một mô hình, thì XmlReadMode được gán là ReadSchema.</li> <li>■ Nếu DataSet không chứa một mô hình và file XML không chứa một mô hình, thì XmlReadMode được gán là InferSchema.</li> </ul> <p>Auto là mặc định</p>
DiffGram	<p>Đọc file XML như một DiffGram, chứa những giá trị ban đầu và những thay đổi tới những giá trị đó để làm chúng lưu hành. rồi những thay đổi này được ứng dụng vào Dataset của bạn . nó tương tự như gọi phương thức Merge() của một Dataset trong đó những sự thay đổi từ một Dataset được trộn với cái khác.</p>
Fragment	<p>Đọc một file XML có chứa những đoạn mô hình inline XDR như những thứ được phát sinh bởi việc thực thi những phát biểu SELECT chứa những mệnh đề FOR XML .</p>
IgnoreSchema	<p>Chỉ đọc dữ liệu trong Dataset, không đọc mô hình.</p>
InferSchema	<p>Phân tích mô hình của file XML bởi việc khảo sát dữ liệu được lưu trữ trong nó.</p>
ReadSchema	<p>Đọc mô hình từ file XML rồi ghi vào trong Dataset.</p>

Ví dụ sau đây cho thấy sự sử dụng hằng số XmlReadMode.ReadSchema:

```
myDataSet.ReadXml("myXmlFile2.xml", XmlReadMode.ReadSchema);
```

Danh sách 10.12 minh họa cách viết và đọc những file XML sử dụng ADO.NET.

**Danh sách 10.12: WRITEANDREADXML.CS**

```

/*
WriteAndReadXml.cs illustrates how to write and read XML files
*/

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

class WriteAndReadXML
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(

```

```

"server=localhost;database=Northwind;uid=sa;pwd=sa");

SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
"SELECT TOP 2 CustomerID, CompanyName, ContactName, Address " +
"FROM Customers " +
"ORDER BY CustomerID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
Console.WriteLine("Retrieving rows from the Customers table");
mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();

// use the WriteXml() method to write the DataSet out to an
// XML file
Console.WriteLine("Writing rows out to an XML file named " +
"myXmlFile.xml using the WriteXml() method");
myDataSet.WriteXml("myXmlFile.xml");

Console.WriteLine("Writing schema out to an XML file named " +
"myXmlFile2.xml using the WriteXml() method");
myDataSet.WriteXml("myXmlFile2.xml", XmlWriteMode.WriteSchema);

// use the WriteXmlSchema() method to write the schema of the
// DataSet out to an XML file
Console.WriteLine("Writing schema out to an XML file named " +
"myXmlSchemaFile.xml using the WriteXmlSchema() method");
myDataSet.WriteXmlSchema("myXmlSchemaFile.xml");

// use the Clear() method to clear the current rows in the DataSet
myDataSet.Clear();

// use the ReadXml() method to read the contents of the XML file
// into the DataSet
Console.WriteLine("Reading rows from myXmlFile.xml " +
"using the ReadXml() method");
myDataSet.ReadXml("myXmlFile.xml");

DataTable myDataTable = myDataSet.Tables["Customers"];
foreach (DataRow myDataRow in myDataTable.Rows)
{
    Console.WriteLine("CustomerID = " + myDataRow["CustomerID"]);
    Console.WriteLine("CompanyName = " + myDataRow["CompanyName"]);
    Console.WriteLine("ContactName = " + myDataRow["ContactName"]);
    Console.WriteLine("Address = " + myDataRow["Address"]);
}
}
}
}

```

**Đầu ra từ chương trình này như sau:**

```

Retrieving rows from the Customers table
Writing rows out to an XML file named myXmlFile.xml using
the WriteXml() method
Writing schema out to an XML file named myXmlFile2.xml
using the WriteXml() method

```

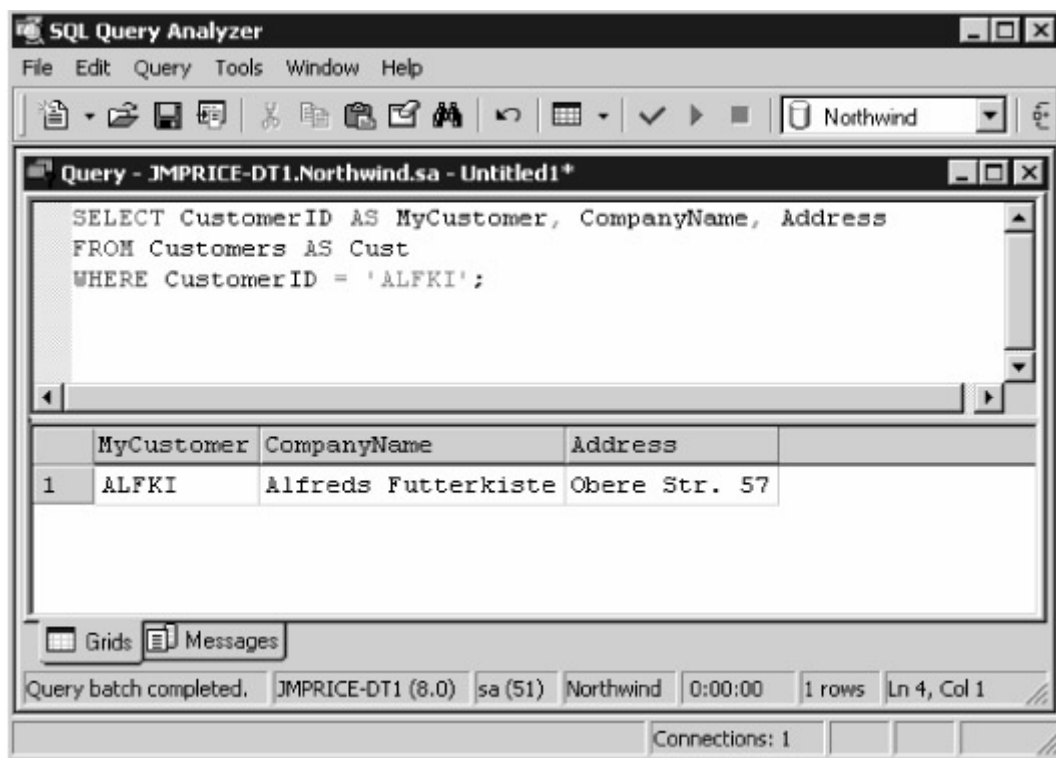
Writing schema out to an XML file named myXmlSchemaFile.xml  
using the WriteXmlSchema() method  
Reading rows from myXmlFile.xml using the ReadXml() method  
CustomerID = ALFKI  
CompanyName = Alfreds Futterkiste  
ContactName = Maria Anders  
Address = Obere Str. 57  
CustomerID = ANATR  
CompanyName = Ana Trujillo3 Emparedados y helados  
ContactName = Ana Trujillo  
Address = Avda. de la Constitución 2222

## **VỀ BẢN ĐỒ NHỮNG BẢNG VÀ CỘT:**

Trong chương 3, "Giới thiệu ngôn ngữ truy vấn có cấu trúc (SQL) ," bạn học điều này từ khóa As được dùng để chỉ định một bí danh cho một bảng hay cột. Ví dụ sau đây sử dụng từ khóa As để đặt bí danh cho cột CustomerID là MyCustomer và bí danh bảng Customers là Cust:

```
SELECT CustomerID AS MyCustomer, CompanyName, Address  
FROM Customers AS Cust  
WHERE CustomerID = 'ALFKI';
```

**Hình 10.2 cho thấy những kết quả của phát biểu SELECT này.**



Mã sau đây sử dụng phát biểu SELECT này để cấu trúc một đối tượng Dataset có tên myDataSet:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
mySqlCommand.CommandText =  
    "SELECT CustomerID AS MyCustomer, CompanyName, Address " +  
    "FROM Customers AS Cust " +  
    "WHERE CustomerID = 'ALFKI'";  
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();  
mySqlDataAdapter.SelectCommand = mySqlCommand;  
DataSet myDataSet = new DataSet();
```

```
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();
```

Chú ý phương thức Fill() chỉ định tên của DataTable như Customers, mà được biết như là tên của DataTable nguồn.

Để vẽ bản đồ một DataTable trong Dataset của bạn, bạn tạo ra một đối tượng của lớp DataTableMapping sử dụng phương thức Add(); lớp này thuộc namespace *System.Data.Common*, mà bạn cần phải tải vào trong chương trình của bạn. Ví dụ sau đây tạo ra một đối tượng DataTableMapping có tên myDataTableMapping, và chuyển giao Customers và Cust (như hai tham số) tới phương thức Add():

```
DataTableMapping myDataTableMapping =
    mySqlDataAdapter.TableMappings.Add("Customers", "Cust");
```

Chú ý rằng phương thức Add() được gọi thông qua thuộc tính TableMappings. Thuộc tính TableMappings trả lại một đối tượng của lớp TableMappingCollection. Đối tượng này là một tập hợp của những đối tượng TableMapping, và bạn sử dụng một đối tượng TableMapping để vẽ bản đồ tên nguồn này tới một tên DataTable khác, bởi vậy, ví dụ trước đây vẽ bản đồ tên nguồn của Customers tới Cust.

Bạn có thể đọc bản đồ này sử dụng những thuộc tính của SourceTable và DataSetTable của myDataTableMapping. Chẳng hạn:

```
Console.WriteLine("myDataTableMapping.SourceTable = " +
    myDataTableMapping.SourceTable);
Console.WriteLine("myDataTableMapping.DataSetTable = " +
    myDataTableMapping.DataSetTable);
```

### Ví dụ này hiển thị như sau:

```
myDataTableMapping.DataSetTable = Cust
myDataTableMapping.SourceTable = Customers
```

Bạn cũng cần phải thay đổi thuộc tính TableName của đối tượng DataTable trong Dataset của bạn để giữ cho những tên vững chắc:

```
myDataSet.Tables["Customers"].TableName = "Cust";
```

Mẹo nhỏ : điều quan trọng là bạn phải thay đổi TableName vì nếu không nó sẽ giữ tên nguyên bản của Customers, và là một chút rắc rối khi bạn đã chỉ định ánh xạ từ Customers đến Cust trước đó.

Tiếp theo, đặt bí danh cột CustomerID là MyCustomer, Bạn gọi phương thức Add() thông qua thuộc tính ColumnMappings của myDataTableMapping:

```
myDataTableMapping.ColumnMappings.Add("CustomerID", "MyCustomer");
```

Thuộc tính ColumnMappings trả lại một đối tượng của lớp DataColumnMappingCollection. Đối tượng này là một tập hợp của những đối tượng DataColumnMapping. Bạn sử dụng một đối tượng DataColumnMapping để vẽ bản đồ một tên cột từ cơ sở dữ liệu đến một tên cột dữ liệu khác, do đó, ví dụ trước đây vẽ bản đồ tên cột CustomerID từ cơ sở dữ liệu đến tên MyCustomer DataColumn.

Danh sách 10.13 minh họa cách để vẽ bản đồ những tên bảng và cột sử dụng mã trình bày trong mục này.

### Danh sách 10.13: MAPPINGS.CS

/\*

Mappings.cs illustrates how to map table and column names

\*/

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

class Mappings
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa");
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT CustomerID AS MyCustomer, CompanyName, Address " +
            "FROM Customers AS Cust " +
            "WHERE CustomerID = 'ALFKI'";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.Fill(myDataSet, "Customers");
        mySqlConnection.Close();

        // create a DataTableMapping object
        DataTableMapping myDataTableMapping =
            mySqlDataAdapter.TableMappings.Add("Customers", "Cust");

        // change the TableName property of the DataTable object
        myDataSet.Tables["Customers"].TableName = "Cust";

        // display the DataSetTable and SourceTable properties
        Console.WriteLine("myDataTableMapping.DataSetTable = " +
            myDataTableMapping.DataSetTable);
        Console.WriteLine("myDataTableMapping.SourceTable = " +
            myDataTableMapping.SourceTable);

        // map the CustomerID column to MyCustomer
        myDataTableMapping.ColumnMappings.Add("CustomerID", "MyCustomer");

        DataTable myDataTable = myDataSet.Tables["Cust"];
        foreach (DataRow myDataRow in myDataTable.Rows)
        {
            Console.WriteLine("CustomerID = " + myDataRow["MyCustomer"]);
            Console.WriteLine("CompanyName = " + myDataRow["CompanyName"]);
            Console.WriteLine("Address = " + myDataRow["Address"]);
        }
    }
}
```

Đầu ra từ chương trình này như sau:

```
myDataTableMapping.DataSetTable = Cust
myDataTableMapping.SourceTable = Customers
```

CustomerID = ALFKI  
CompanyName = Alfreds Futterkiste  
Address = Obere Str. 57

## **ĐỌC MỘT GIÁ TRỊ CỘT SỬ DỤNG NHỮNG LỚP DATASET ĐỊNH KIỂU DỮ LIỆU MẠNH:**

Một đối tượng Dataset định kiểu mạnh cho phép bạn đọc một giá trị cột sử dụng một thuộc tính cùng tên với cột. Chẳng hạn, để đọc giá trị CustomerID của một cột, bạn có thể sử dụng myDataRow.CustomerID hơn là myDataRow[ " CustomerID "]. Đây là một đặc tính hay bởi vì trình biên dịch có thể bắt bất kỳ lỗi chính tả nào trong tên cột trong thời gian biên tập hơn là thời gian chạy. Chẳng hạn, nếu bạn ghi sai CustomerID là CustimerID, và lỗi bị bắt ngay bởi trình biên dịch.

Đặc tính khác của một Dataset định kiểu dữ liệu mạnh là khi bạn làm việc với nó trong VS .NET, trình cảm ứng thông minh tự động bật ra những thuộc tính và những phương thức của Dataset này khi bạn đang gõ phím. Và Bạn có thể chọn thuộc tính hay phương thức từ danh sách, hơn là phải gõ mọi thứ vào.

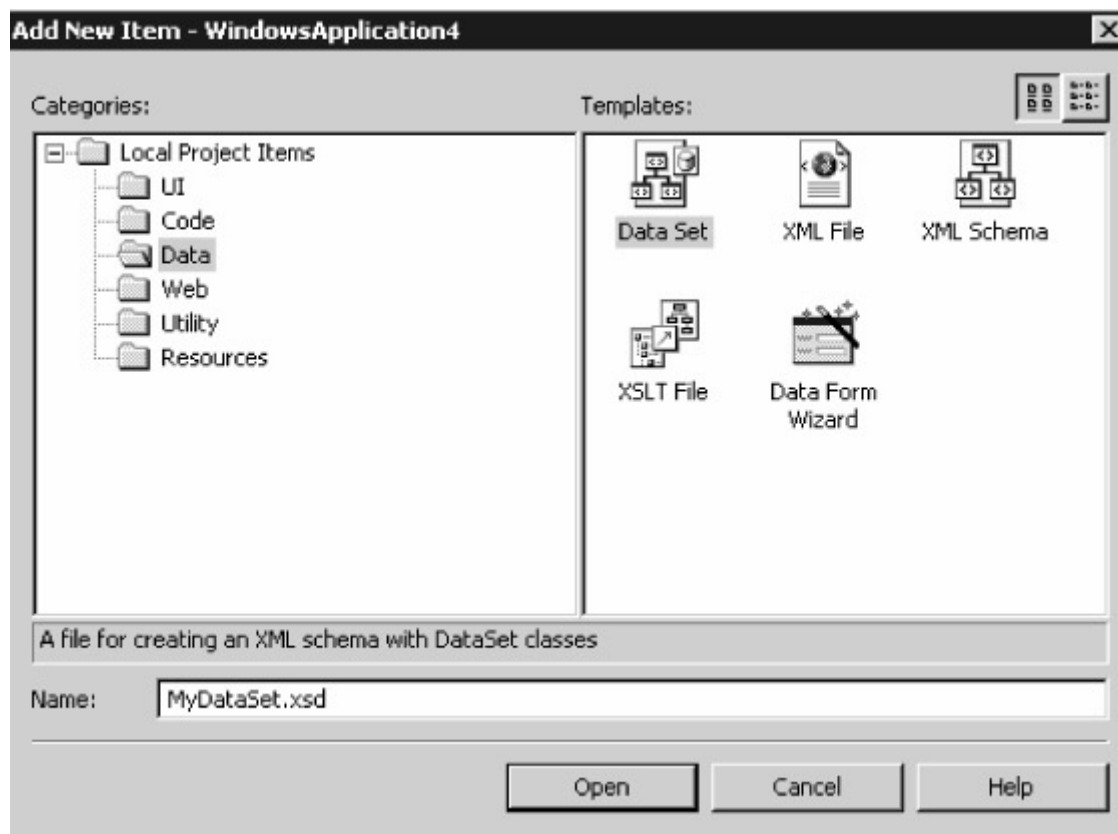
Bề sâu của việc sử dụng một Dataset định kiểu dữ liệu mạnh là bạn phải thực hiện vài công việc ban đầu nào đó để tạo ra nó trước khi bạn có thể sử dụng nó. Nếu những cột trong những bảng cơ sở dữ liệu của bạn không thường xuyên thay đổi, thì bạn cần phải cẩn trọng sử dụng những đối tượng Dataset định kiểu dữ liệu mạnh. Mặt khác, nếu những bảng cơ sở dữ liệu của các bạn thường thay đổi, bạn cần phải tránh chúng bởi vì bạn sẽ phải tạo lại Dataset định kiểu mạnh để giữ cho nó đồng bộ với định nghĩa của bảng cơ sở dữ liệu.

Ghi nhớ: Bạn sẽ tìm thấy một dự án ví dụ được bổ sung VS .NET cho mục này trong thư mục StronglyTypedDataSet. Bạn có thể mở dự án này trong VS .Net bởi chọn File \_ Open Project và mở file WindowsApplication4.csproj . Bạn sẽ cần thay đổi thuộc tínhConnectionString của đối tượng sqlConnection1 để nối tới cơ sở dữ liệu Northwind SQL Server của bạn. Bạn có thể cũng theo cùng với những chỉ dẫn trong mục này bởi việc sao chép thư mục DataReader đến thư mục khác và sử dụng dự án này như điểm xuất phát của bạn.

## **TẠO MỘT LỚP DATASET ĐỊNH KIỂU MẠNH:**

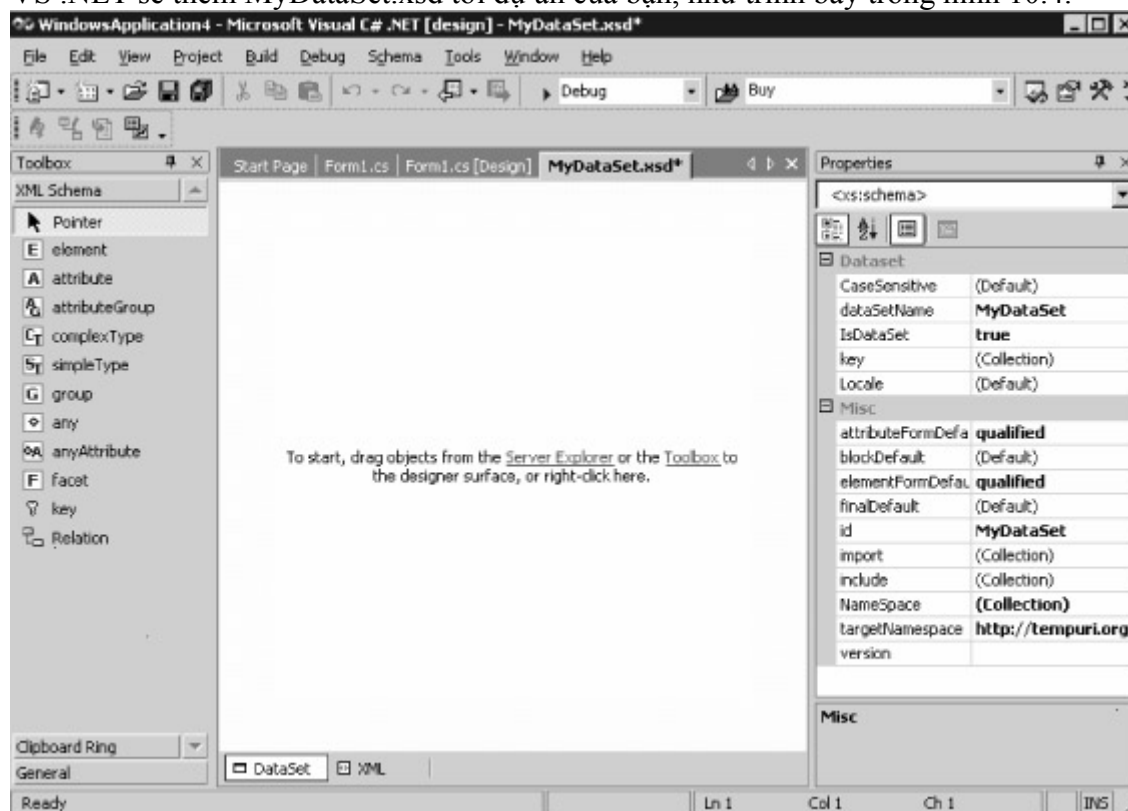
Trong mục này, bạn sẽ tạo ra một lớp Dataset định kiểu mạnh mà được dùng để truy nhập bảng Customers. Nếu bạn theo những chỉ dẫn này, mở dự án DataReader trong VS .NET và nhấn đúp Form1.cs trong cửa sổ Solution Explorer. Bạn mở cửa sổ Solution Explorer bằng cách chọn View ➤ Solution Explorer.

Tiếp theo, chọn File ➤ Add New Item. Chọn Dataset từ vùng khung mẫu và gõ vào MyDataSet.xsd, như trình bày trong hình 10.3 .



Kích nút Open để tiếp tục.

VS .NET sẽ thêm MyDataSet.xsd tới dự án của bạn, như trình bày trong hình 10.4.



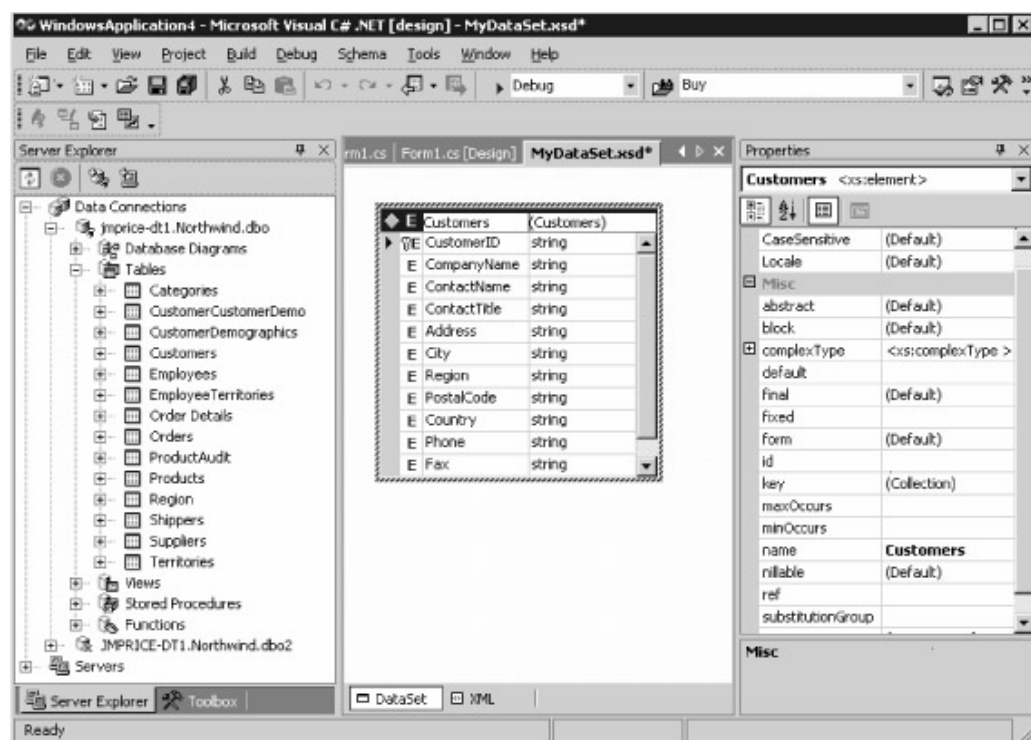
Ở dưới về phía trái của Hình 10.4, bạn chú ý hai tab: Dataset và XML. tab Dataset được trình bày theo mặc định và bạn sử dụng nó để xem visual view của Dataset của bạn. tab XML cho phép bạn xem File XML của Dataset của bạn.

Tiếp theo, hãy chắc chắn rằng bạn đã mở cửa sổ Explorer window; để mở cửa sổ, chọn View Server Explorer. Mở nút Data Connections và hoặc sử dụng một kết nối hiện hữu tới cơ sở dữ liệu Northwind hoặc tạo ra một kết nối mới bằng cách nhấn phải trên nút Data Connection và chọn Add Connection từ thực đơn sổ xuống.



Nhấn đúp đối tượng kết nối của bạn trong cửa sổ Server Explorer và tìm xuống tới bảng, View hay thủ tục lưu trữ mà bạn muốn sử dụng, và rồi kéo nó vào form của bạn , bắt đầu thực hiện \_kéo bảng Customers lên form của bạn.

### Hình 10.5 cho thấy form khi bảng Customers đã được thêm.



Ghi nhớ : Bạn có thể thêm nhiều bảng vào form của bạn và định nghĩa những quan hệ giữa chúng.

Tiếp theo, lưu công việc của bạn bởi chọn File ➤ Save All hay nhấn Ctrl+ S trên bàn phím .

Dự án của bạn bây giờ chứa đựng một File XSD có tên MyDataSet.xsd, như trình bày trong danh sách 10.14 .  
Bạn có thể xem file này bởi việc kích tab XML tại đáy của cửa sổ thiết kế XML.

### Danh sách 10.14: MYDATASET.XSD

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="MyDataSet"
targetNamespace="http://tempuri.org/MyDataSet.xsd" elementFormDefault="qualified"
attributeFormDefault="qualified"
xmlns="http://tempuri.org/MyDataSet.xsd"
xmlns:mstns="http://tempuri.org/MyDataSet.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="MyDataSet" msdata:IsDataSet="true">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Customers">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CustomerID" type="xs:string" />
              <xs:element name="CompanyName" type="xs:string" />
              <xs:element name="ContactName" type="xs:string" minOccurs="0" />
              <xs:element name="ContactTitle" type="xs:string" minOccurs="0" />
              <xs:element name="Address" type="xs:string" minOccurs="0" />
            
```

```

<xs:element name="City" type="xs:string" minOccurs="0" />
<xs:element name="Region" type="xs:string" minOccurs="0" />
<xs:element name="PostalCode" type="xs:string" minOccurs="0" />
<xs:element name="Country" type="xs:string" minOccurs="0" />
<xs:element name="Phone" type="xs:string" minOccurs="0" />
<xs:element name="Fax" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
<xs:unique name="MyDataSetKey1" msdata:PrimaryKey="true">
  <xs:selector xpath="//mstns:Customers" />
  <xs:field xpath="mstns:CustomerID" />
</xs:unique>
</xs:element>
</xs:schema>

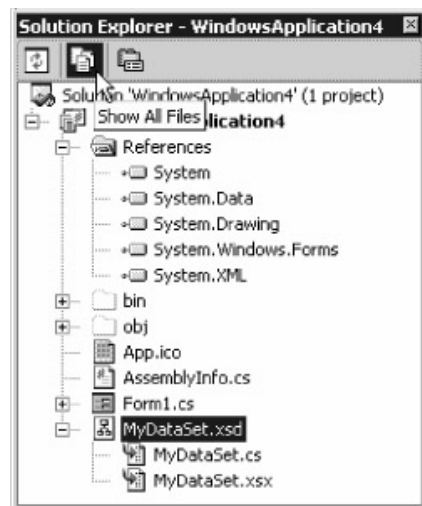
```

Chú ý: file này chứa những chi tiết của những cột trong bảng Customs.

Dự án của bạn cũng chứa một lớp mới có tên file là MyDataSet.cs, nó chứa lớp Dataset định kiểu mạnh của bạn. Bạn có xem nội dung của file này sử dụng Cửa sổ Solution Explorer . Bạn mở cửa sổ Solution Explorer bởi chọn View ➤ Solution Explorer .

Ghi chú: để xem file MyDataSet.cs, kích nút Show All Files trong cửa sổ Solution Explorer.

Tiếp theo, bung nút bên dưới MyDataSet.xsd. Bạn sẽ thấy MyDataSet.cs, như trình bày trong Hình 10.6 , và một file có tên MyDataSet.aspx, nó chứa thông tin về cách trình bày cảnh quan trực quan của Dataset của bạn.



khởi sự \_ mở MyDataSet.cs bằng cách nhấn đúp nó trong cửa sổ Solution Explorer. xem mã của form này bởi việc chọn View Code. Một trong những lớp được khai báo trong file này là MyDataSet. Lớp này bắt nguồn từ lớp Dataset. Bạn sẽ sử dụng nó trong mục tiếp theo để tạo ra một đối tượng Dataset định kiểu mạnh để truy nhập bảng Customers .

## **SỬ DỤNG LỚP DATASET ĐỊNH KIỂU MẠNH:**

Một khi bạn có lớp MyDataSet định kiểu mạnh , bạn có thể tạo ra một đối tượng của lớp này sử dụng mã sau đây:

```
MyDataSet myDataSet = new MyDataSet();
```

Bạn cũng có thể tạo ra một đối tượng bảng DataTable định kiểu mạnh sử dụng lớp MyDataSet. CustomersDataTable và cư trú nó với những hàng từ bảng Customers. ví dụ, bạn có thể đặt phương thức Form1\_Load() của form của bạn để truy xuất những giá trị cột CustomerID, CompanyName, Và Address từ bảng Customers và gán chúng vào một điều khiển ListView có tên listView1. Để làm điều này, nhấn đúp Form1.cs Trong cửa sổ Solution Explorer , xem mã, và đặt phương thức Form1\_Load() như sau:

```
private void Form1_Load(object sender, System.EventArgs e)
{
    System.Data.SqlClient.SqlCommand mySqlCommand =
    sqlConnection1.CreateCommand();
    mySqlCommand.CommandText =
    "SELECT CustomerID, CompanyName, Address " +
    "FROM Customers " +
    "WHERE CustomerID = 'ALFKI'";
    System.Data.SqlClient.SqlDataAdapter mySqlDataAdapter =
    new System.Data.SqlClient.SqlDataAdapter();
    mySqlDataAdapter.SelectCommand = mySqlCommand;
    MyDataSet myDataSet = new MyDataSet();
    sqlConnection1.Open();
    mySqlDataAdapter.Fill(myDataSet, "Customers");
    sqlConnection1.Close();
    MyDataSet.CustomersDataTable myDataTable = myDataSet.Customers;
    foreach (MyDataSet.CustomersRow myDataRow in myDataTable.Rows)
    {
        listView1.Items.Add(myDataRow.CustomerID);
        listView1.Items.Add(myDataRow.CompanyName);
        listView1.Items.Add(myDataRow.Address);
    }
}
```

Thuộc tính myDataRow.CustomerID trả lại giá trị cột CustomerID, vân vân. Biên tập và chạy form của bạn trong một bước bởi chọn Debug ➤ Start Without Debugging. Hình 10.7 cho thấy kết quả sự chạy form.



Ghi nhớ: lớp MyDataSet chứa một số phương thức cho phép bạn sửa đổi những hàng được cất giữ trong một đối tượng MyDataSet. Những phương thức này bao gồm NewCustomersRow(), AddCustomersRow(), FindByCustomerID(), Và RemoveCustomersRow(). Bạn cũng có thể kiểm tra liệu một giá trị cột chứa đựng một giá trị null sử dụng những phương thức như IsContactNameNull(), Và Bạn có thể đặt một cột với giá trị null sử dụng những phương thức như SetContactNameNull(). Bạn sẽ học cách sử dụng những phương thức này trong Chương 11.

**TAO MỘT ĐỐI TƯỢNG *DataAdapter* SỬ DỤNG VISUAL STUDIO .NET:**

Trong mục này, bạn sẽ học cách tạo một DataAdapter như thế nào sử dụng Visual Studio .NET.

Ghi nhớ: Bạn sẽ tìm thấy một dự án VS .NET cho mục này trong thư mục DataAdapter. Bạn có thể mở dự án này Trong VS .NET bởi chọn File Open Project và Mở File WindowsApplication4.csproj . Bạn sẽ cần thay đổi thuộc tínhConnectionString của đối tượng sqlConnection1 để kết nối tới cơ sở dữ liệu của bạn. Bạn có thể cũng có thể theo những chỉ dẫn trong mục này bởi việc sao chép thư mục DataReader đến thư mục khác và sử dụng dự án này như điểm xuất phát của bạn.

Mở form của bạn bằng cách nhấn đúp Form1.cs trong cửa sổ Solution Explorer. Tiếp theo, tạo một đối tượng SqlDataAdapter bằng cách kéo một đối tượng SqlDataAdapter từ tab Dữ liệu của hộp Toolbox đến form của bạn. Khi bạn kéo một đối tượng SqlDataAdapter tới form , một trình Wizard Data Adapter Configuration khởi chạy (Cấu hình Bộ tiếp hợp Dữ liệu), như trình bày trong Hình 10.8.



**Hình 10.8: trình hướng dẫn tạo Cấu hình Bộ tiếp hợp Dữ liệu**

Kích nút Next để tiếp tục.

Bạn bây giờ chọn đối tượng kết nối cơ sở dữ liệu Bạn muốn sử dụng, hay Bạn có thể tạo ra một đối tượng kết nối mới. chọn cơ sở dữ liệu cần kết nối Northwind trong danh sách comboBox( Hay tạo ra một kết nối mới nếu bạn chưa có sẵn ), Như trình bày trong Hình 10.9.



**Hình 10.9: chọn kết nối dữ liệu của bạn**

Kích nút Next để tiếp tục.

Tiếp theo, bạn đặt query type của bạn là "Use SQL statements." ( Sử dụng những câu lệnh SQL.) Như bạn có thể thấy từ Hình 10.10, bạn đặt query type của bạn để sử dụng những câu lệnh SQL, tạo ra những thủ tục lưu trữ mới, hay sử dụng những thủ tục lưu trữ hiện có. Những câu lệnh SQL hay những thủ tục lưu trữ rồi được sử dụng trong những thuộc tính SelectCommand, InsertCommand, UpdateCommand, Và DeleteCommand của đối tượng SqlDataAdapter của bạn. Bạn sẽ học ba thuộc tính sau cùng này trong Chương 11; chúng được dùng để chèn, cập nhật, và xóa những hàng.



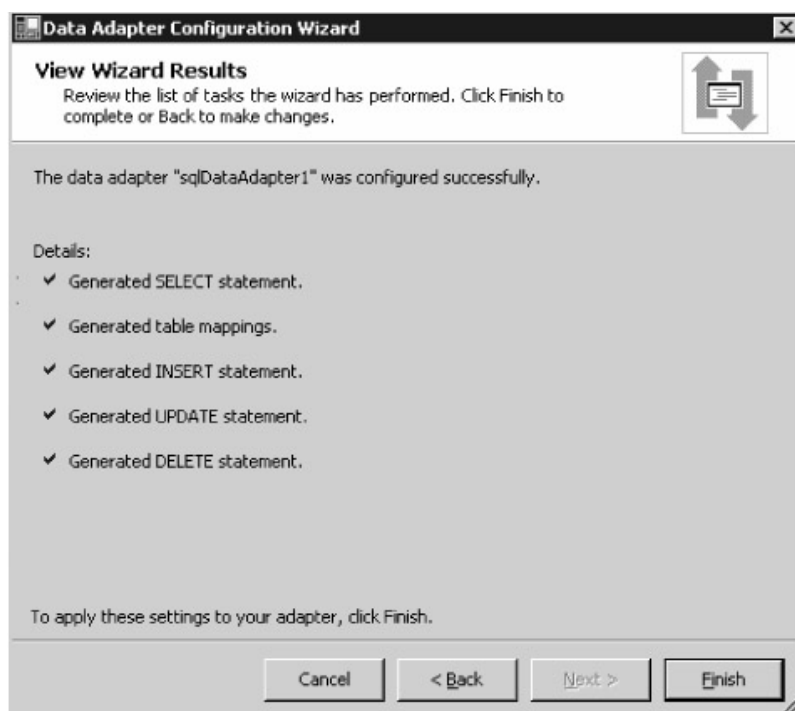
**Hình 10.10: chọn query type của các bạn**

Chắc chắn rằng bạn có chọn "Use SQL statements", và kích nút Next để tiếp tục.



Hình 10.11: phát sinh những câu lệnh SQL

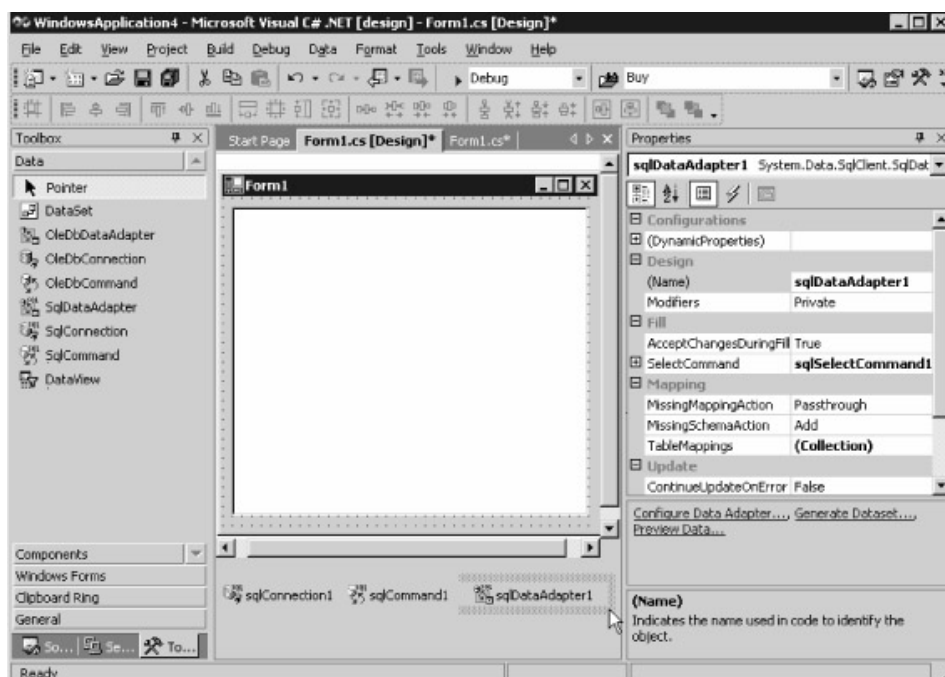
Phát biểu SELECT mà bạn nhập vào bây giờ được dùng để phát sinh những câu lệnh SQL INSERT, UPDATE, and DELETE cùng với những ánh xạ bảng. Hình 10.12 cho thấy hộp thoại cuối cùng của wizard tạo "Cấu hình Bộ tiếp hợp Dữ liệu"(Data Adapter Configuration Wizard ).



Hình 10.12:Hộp thoại Cuối cùng của wizard tạo Cấu hình Bộ tiếp hợp Dữ liệu

Kích nút Finish để hoàn thành wizard. Một đối tượng SqlDataAdapter có tên sqlDataAdapter1, bây giờ được thêm vào khay ở dưới form của bạn , như trong Hình 10.13.





Hình 10.13: đối tượng SqlDataAdapter mới trong khay

Cảnh báo: Bạn cần đặt thuộc tính Connection của SelectCommand trong đối tượng sqlDataAdapter1 của các bạn tới đối tượng Kết nối (Connection) của bạn trước khi DataAdapter có thể truy nhập cơ sở dữ liệu. Bạn thực hiện điều này sử dụng cửa sổ những thuộc tính bằng cách duyệt xuống từ SelectCommand đến Connection. rồi Bạn kích danh sách xổ xuống, chọn Existing, rồi chọn đối tượng SqlConnection của bạn , nó phải có tên là sqlConnection1 . đồng thời kiểm tra thuộc tínhConnectionString của đối tượng SqlConnection của bạn để chắc chắn rằng nó được đặt nối tới cơ sở dữ liệu Northwind của bạn.

Chú ý: ba liên kết ở đáy của cửa sổ những thuộc tính Cho sqlDataAdapter1:

**Configure Data Adapter** : mỗi liên kết này cho phép bạn ghi lại Wizard để định hình DataAdapter của các bạn.

**Generate Dataset** : mỗi liên kết này cho phép bạn phát sinh một đối tượng Dataset sử dụng tập hợp thông tin cho DataAdapter của bạn. Bạn sẽ sử dụng mỗi liên kết này trong mục kế tiếp để phát sinh một Dataset mới.

**Preview Data** : mỗi liên kết này cho phép bạn xem trước dữ liệu được trả lại bởi SelectCommand của DataAdapter của các bạn.

hãy thoải mái khảo sát mã sinh ra bởi Wizard trong form của các bạn cho đối tượng sqlDataAdapter1. Khi bạn sẵn sàng, chọn File ➤ Save All.

**Ghi nhớ** : đừng vội vàng chạy dự án của bạn ngay, bởi vì bạn sẽ còn thêm một Dataset ,nó sẽ được cư trú bởi sử dụng DataAdapter của bạn trong mục kế tiếp.

## **TAO MỘT ĐỐI TƯỢNG DATASET SỬ DỤNG Visual Studio .NET:**

Trong mục này, bạn sẽ học cách tạo ra một Dataset sử dụng Visual.Studio .NET.

Ghi nhớ: Bạn sẽ tìm thấy một dự án ví dụ được bổ sung Visual Studio .NET cho mục này trong thư mục Dataset. Bạn có thể mở dự án này trong Visual Studio .NET bởi chọn File Open Project và mở File WindowsApplication4.csproj . Bạn cũng có thể theo những chỉ dẫn trong mục này bằng cách tiếp tục sửa đổi bản sao chép của dự án DataReader mà bạn sử dụng trong mục trước .

Nếu bạn theo những chỉ dẫn này, mở bản sao của dự án DataReader mà bạn sửa đổi trong mục trước , và mở Form1.cs bằng cách nhấn đúp nó trong cửa sổ Solution Explorer . Để tạo ra một đối tượng Dataset , bạn có thể thực hiện một trong số việc sau :



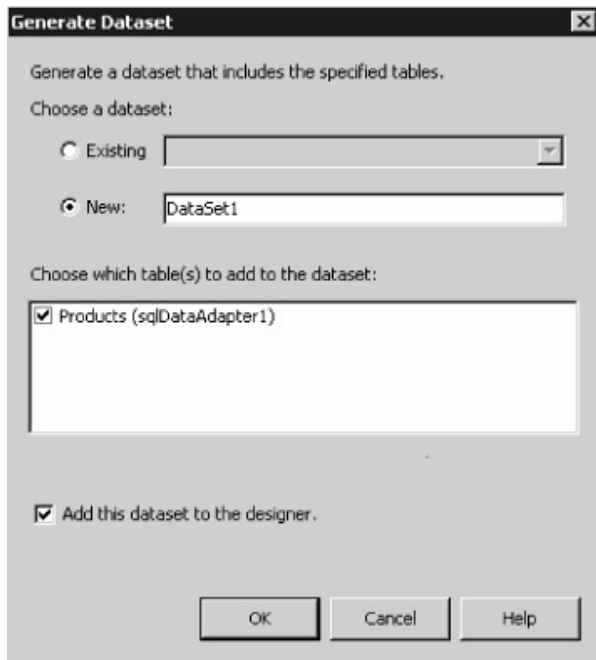
■ Kéo một đối tượng Dataset từ tab Data của Toolbox đến form của bạn, và thêm mã vào form của bạn để

Điền nó sử dụng phương thức Fill() của một đối tượng DataAdapter.

■ Kích liên kết Generate Dataset tại đáy của cửa sổ những thuộc tính của DataAdapter của bạn. Bạn có

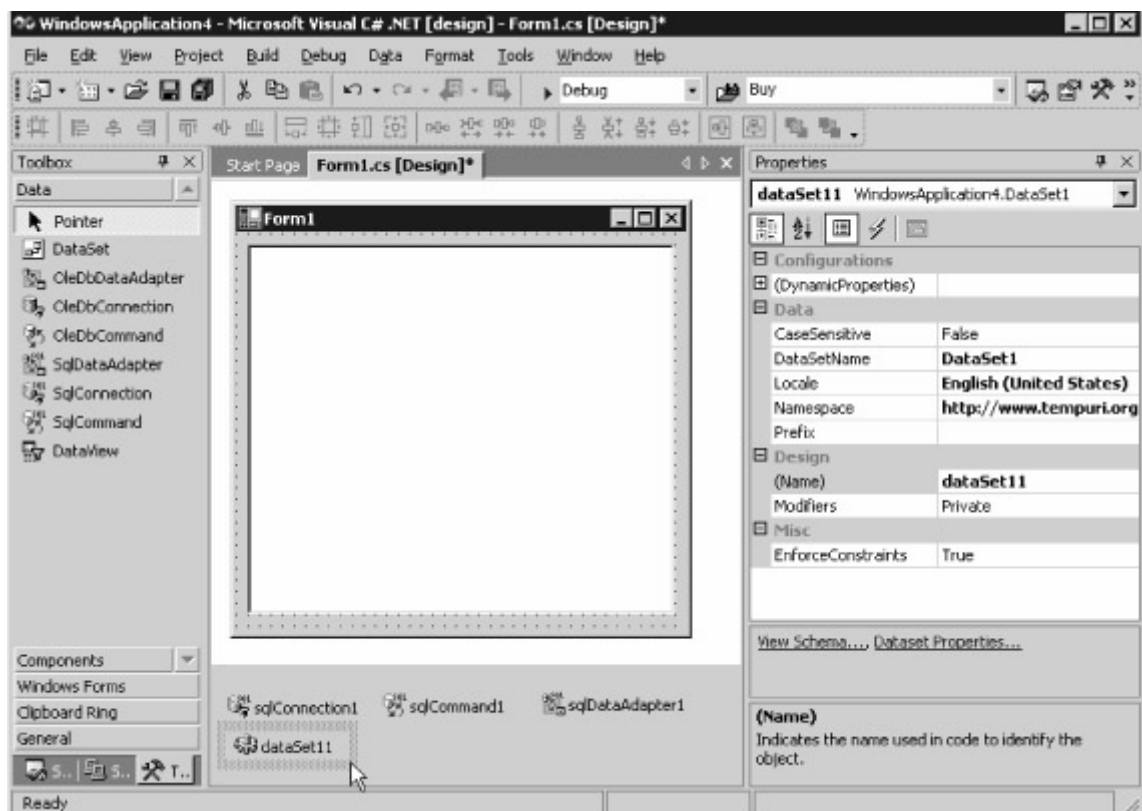
thể nhìn thấy mối liên kết này trong Hình 10.13.

Bạn sẽ sử dụng hai bước, và bắt đầu - kích liên kết Generate Dataset . hộp thoại Generate Dataset xuất hiện, như trình bày trong Hình 10.14.



Hình 10.14: hộp thoại phát sinh Dataset

Kích nút Ok để tiếp tục. Đối tượng tập dữ liệu mới có tên dataSet11 được thêm vào khay ở dưới form của bạn, như trong Hình 10.15.



### Hình 10.15: đối tượng Dataset mới trong khay

Bước tiếp theo của bạn là gán phương thức Form1\_Load() của form như sau:

```
private void Form1_Load(object sender, System.EventArgs e)
{
    sqlConnection1.Open();
    sqlDataAdapter1.Fill(dataSet11, "Products");
    sqlConnection1.Close();
    System.Data.DataTable myDataTable =
    dataSet11.Tables["Products"];
    foreach (System.Data.DataRow myDataRow in myDataTable.Rows)
    {
        listView1.Items.Add(myDataRow["ProductID"].ToString());
        listView1.Items.Add(myDataRow["ProductName"].ToString());
        listView1.Items.Add(myDataRow["UnitPrice"].ToString());
    }
}
```

#### Ghi chú:

Nhớ, để xem mã form của bạn, bạn chọn view ➤ code. Rồi bạn thay thế phương thức Form1\_Load() với mã trước đây.

và bây giờ bạn có thể biên tập và chạy form của bạn. Hình 10.16 cho thấy sự vận hành của form.



Hình 10.16: form đang chạy

### TÓM LƯỢC:

Trong chương này, bạn đã học những chi tiết về việc sử dụng những đối tượng Dataset để lưu trữ những kết quả trả lại từ cơ sở dữ liệu. Những đối tượng Dataset cho phép bạn lưu trữ một bản sao chép của những bảng và những hàng từ cơ sở dữ liệu, và bạn có thể làm việc với bản sao cục bộ này trong khi ngắt kết nối với cơ sở dữ liệu. Không giống những đối tượng bộ cung cấp được quản lý(managed provider) như những đối tượng SqlDataReader, những đối tượng Dataset là dùng chung và bởi vậy làm việc với bất kỳ cơ sở dữ liệu nào. Những đối tượng Dataset cũng cho phép bạn đọc những hàng trong bất kỳ trật tự nào và sửa đổi những hàng.

Bạn cũng đã học những chi tiết về việc sử dụng một đối tượng DataAdapter để đọc những hàng từ cơ sở dữ liệu vào trong một đối tượng Dataset. DataAdapter là bộ phận của lớp nhà cung cấp được quản lý, và có ba lớp DataAdapter: SqlDataAdapter, OleDbDataAdapter, Và OdbcDataAdapter.

Bạn sử dụng một đối tượng DataAdapter để di chuyển những hàng giữa đối tượng Dataset của bạn và cơ sở dữ liệu, và để đồng bộ hóa bất kỳ sự thay đổi nào Bạn đã làm cho những hàng lưu trữ cục bộ của bạn với cơ sở dữ liệu. Chẳng hạn, bạn có thể đọc những hàng từ cơ sở dữ liệu vào trong một Dataset thông qua một DataAdapter, Sửa đổi những hàng đó trong Dataset của bạn, và đẩy những sự thay đổi đó tới cơ sở dữ liệu thông qua đối tượng DataAdapter của bạn.

Trong Chương 11, bạn sẽ thấy cách để thực hiện những thay đổi tới những hàng trong một Dataset và sau đó đẩy những thay đổi tới cơ sở dữ liệu.

**SỬ DỤNG NHỮNG ĐỐI TƯỢNG DATASET ĐỂ SỬA ĐỔI DỮ LIỆU:**

**Tổng quan**

Trong chương 10, bạn đã thấy cách sử dụng một Dataset để cất giữ một bản sao của những hàng truy xuất từ cơ sở dữ liệu. Trong chương này, bạn sẽ học cách sửa đổi những hàng trong một Dataset, và đẩy những thay đổi này tới cơ sở dữ liệu qua một DataAdapter.

Những điều nổi bật trong chương này:

- Những lớp DataTable, DataRow, và DataColumn
- Thêm những sự hạn chế vào những đối tượng DataTable và DataColumn
- Tìm kiếm, lọc và sắp xếp những đối tượng DataRow trong một DataTable
- Sửa đổi những hàng trong một DataTable và đẩy những sự thay đổi đó tới cơ sở dữ liệu
- Sử dụng những thủ tục lưu trữ để thêm, sửa đổi, và loại bỏ những hàng từ cơ sở dữ liệu
- Sử dụng một đối tượng CommandBuilder để tự động phát sinh những câu lệnh SQL
- Duyệt tìm những biến cố của DataAdapter và DataTable
- Giải quyết những sự thất bại cập nhật (update).
- Sử dụng những giao dịch với một Tập dữ liệu
- Sửa đổi dữ liệu sử dụng một Dataset định kiểu mạnh.

**LỚP DATATABLE:**

Bạn sử dụng một đối tượng của lớp DataTable để đại diện cho một bảng. Bạn cũng có thể cất giữ nhiều đối tượng DataTable trong một Dataset. Bảng 11.1 cho thấy một số những thuộc tính DataTable.

**Bảng 11.1: những thuộc tính DataTable**

THUỘC TÍNH	KIỂU	MÔ TẢ
CaseSensitive	bool	Lấy hay gán một giá trị bool chỉ định liệu có phải những sự so sánh chuỗi bên trong những đối tượng DataTable phụ thuộc kiểu chữ.
ChildRelations	DataRelationCollection	lấy tập hợp của những quan hệ (DataRelationCollection) nó cho phép sự dẫn hướng từ một bảng cha đến một bảng con. Một DataRelationCollection gồm có những đối tượng DataRelation.
Columns	DataColumnCollection	Lấy tập hợp của những cột (DataColumnCollection) nó chứa đựng những đối tượng DataColumn đại diện cho những cột trong đối tượng DataTable .

THUỘC TÍNH	KIỂU	MÔ TẢ
Constraints	ConstraintCollection	Đọc tập hợp của những sự ràng buộc (ConstraintCollection) nó chứa đựng những đối tượng ràng buộc đại diện cho những sự ràng buộc khóa chính (UniqueConstraint) hay khóa phụ (ForeignKeyConstraint) trong đối tượng DataTable.
DataSet	DataSet	lấy Dataset mà DataTable thuộc về nó.
HasErrors	bool	trả lại Giá trị một bool mà chỉ định có những lỗi trong bất kỳ những hàng trong DataTable hay không.
PrimaryKey	DataColumn[]	Lấy hay gán một mảng của những đối tượng DataColumn là những khóa chính cho DataTable.
Rows	DataRowCollection	lấy tập hợp của những hàng (DataRowCollection) có chứa những đối tượng DataRow được cất giữ trong DataTable.
TableName	string	Lấy hay gán tên của đối tượng DataTable.

**Bảng 11.2 trình bày một số những phương thức DataTable.**

Phương thức	Kiểu trả về	Mô tả
AcceptChanges()	void	hiệu lực hóa tất cả những sự thay đổi thực hiện với đối tượng DataTable từ khi nó được tải hay từ lần cuối cùng phương thức AcceptChanges() được gọi.
Clear()	void	Loại bỏ tất cả các hàng từ đối tượng DataTable.
Clone()	DataTable	Sao chép cấu trúc của những đối tượng DataTable và sự trả về bản sao này.
Compute()	object	Tính toán biểu thức đã cho trên những hàng hiện thời thông qua điều kiện lọc.
GetChanges()	DataTable	Quá tải . Trả lại một bản sao của đối tượng DataTable từ lúc cuối cùng nó được tải hay từ lần cuối cùng phương thức AcceptChanges() được gọi .
GetErrors()	DataRow[]	Quá tải . lấy một bản sao của tất cả những đối tượng DataRow mà có những lỗi.
LoadDataRow()	DataRow	Tìm và cập nhật một đối tượng DataRow chỉ định. Nếu không tìm thấy đối tượng thích ứng , một hàng mới được tạo ra sử dụng những giá trị chỉ định.
NewRow()	DataRow	Tạo ra một đối tượng DataRow mới trong DataTable.
RejectChanges()	void	hủy tất cả những sự thay đổi thực hiện với đối tượng DataTable một khi nó được tạo ra hay từ lần cuối cùng phương thức AcceptChanges() được gọi.
Select()	DataRow[]	Quá tải. Trả lại mảng của những đối tượng DataRow được cất giữ trong DataTable mà phù hợp với chuỗi lọc được chỉ rõ. Bạn cũng có thể thông qua một chuỗi việc chứa những chi tiết về việc sắp những đối tượng DataRow.

**Bảng 11.3 trình bày một số biến cố DataTable.**

Biến cố	Bộ xử lý biến cố	Mô tả
ColumnChanging	DataColumnChangeEventHandler	phát khởi trước khi một giá trị được thay đổi của DataColumn được cập nhật ở DataRow.
ColumnChanged	DataColumnChangeEventHandler	phát khởi sau khi một giá trị DataColumn được thay đổi được cập nhật ở DataRow.
RowChanging	DataRowChangeEventHandler	phát khởi trước khi một DataRow được thay đổi được cập nhật ở một DataTable.
RowChanged	DataRowChangeEventHandler	Phát khởi sau khi một DataRow được thay đổi được cập nhật ở một DataTable.
RowDeleting	DataRowChangeEventHandler	Phát khởi trước khi một DataRow bị xóa từ một

Biến cố	Bộ xử lý biến cố	Mô tả
		DataTable.
RowDeleted	DataRowChangeEventHandler	Phát khởi sau khi một DataRow đã bị xóa từ một DataTable.

## LỚP DATAROW:

Bạn sử dụng một đối tượng của lớp DataRow để đại diện cho một hàng. Bạn cũng có thể cất giữ nhiều đối tượng DataRow ở một DataTable. Bảng 11.4 trình bày một số thuộc tính DataRow.

**Bảng 11.4: những thuộc tính DataRow**

Thuộc tính	Kiểu	Mô tả
HasErrors	bool	Trả lại một giá trị bool cho biết liệu có bất kỳ những đối tượng DataColumn nào trong DataRow có lỗi hay không.
ItemArray	object[]	Lấy hay gán tất cả đối tượng DataColumn trong DataRow.
RowState	DataRowState	Lấy thông tin tình trạng DataRow hiện thời. Trạng thái có thể là Added, Deleted, Detached (tách ra~ FT), Modified , hay Unchanged (Không thay đổi). Trạng thái phụ thuộc thao tác được thực hiện trên DataRow và liệu phương thức AcceptChanges() đã được gọi và cập nhật những sự thay đổi hay không.
Table	DataTable	lấy đối tượng DataTable mà DataRow thuộc vào.

Hàng đã được tạo ra nhưng không là bộ phận của một đối tượng DataRowCollection; một DataRow trong trạng thái này ngay lập tức sau khi nó đã được tạo ra và trước khi nó được thêm vào một tập hợp, hay nếu nó được loại bỏ khỏi một tập hợp.

**Bảng 11.5 trình bày một số phương thức của DataRow.**

Phương thức	Kiểu trả về	Mô tả
AcceptChanges()	void	Cập nhật tất cả những sự thay đổi thực hiện với đối tượng DataRow từ khi nó được tải hay từ lần cuối cùng phương thức AcceptChanges() được gọi.
BeginEdit()	void	Bắt đầu một soạn thảo cho đối tượng DataRow.
CancelEdit()	void	hủy một soạn thảo cho đối tượng DataRow và trả lại tình trạng nguyên bản của nó.
ClearErrors()	void	Xóa bỏ bất kỳ lỗi nào cho đối tượng DataRow.
Delete()	void	Xóa đối tượng DataRow.
EndEdit()	void	kết thúc một soạn thảo cho đối tượng DataRow và cập nhật sự thay đổi.
GetChildRows()	DataRow[]	Quá tải . Trả về một mảng của những đối tượng DataRow chứa những hàng con sử dụng đối tượng DataRelation được chỉ định.
GetColumnError()	string	Quá tải . Trả về một mô tả lỗi cho đối tượng DataColumn chỉ định.
GetColumnsInError()	DataColumn[]	Trả về một mảng của những đối tượng DataColumn mà có những lỗi.
GetParentRow()	DataRow	Quá tải. Trả về một đối tượng DataRow có chứa hàng cha sử dụng đối tượng DataRelation được chỉ rõ.
GetParentRows()	DataRow[]	Quá tải . Trả về một mảng của những đối tượng DataRow có chứa những hàng cha sử dụng đối tượng DataRelation được chỉ rõ.

Phương thức	Kiểu trả về	Mô tả
IsNull()	bool	Quá tải. Trả về một giá trị bool cho biết liệu đối tượng DataColumn được chỉ rõ có chứa một giá trị null hay không.
RejectChanges()	void	hủy tất cả các thay đổi thực hiện với đối tượng DataRow từ khi phương thức AcceptChanges() được gọi.
SetNull()	void	Gán một đối tượng DataColumn được chỉ rõ đến một giá trị null.
SetParentRow()	void	Quá tải. Gán hàng cha tới đối tượng DataRow được chỉ rõ.

## **LỚP DATACOLUMN:**

Bạn sử dụng một đối tượng của lớp DataColumn để đại diện cho một cột. Bạn cũng có thể cất giữ nhiều đối tượng DataColumn trong một DataRow. Bảng 11.6 cho thấy một số thuộc tính DataColumn.

**Bảng 11.6: những thuộc tính DataColumn**

Thuộc tính	Kiểu	Mô tả
AllowDBNull	bool	Lấy hay gán một giá trị bool cho biết liệu những giá trị null có được cho phép trong đối tượng DataColumn này. giá trị mặc định là true.
AutoIncrement	bool	lấy hay gán một giá trị bool cho biết liệu đối tượng DataColumn có tự động tăng dần giá trị của cột cho những hàng mới. giá trị mặc định là false.
AutoIncrementSeed	long	Lấy hay gán giá trị khởi đầu cho đối tượng DataColumn. chỉ ứng dụng khi thuộc tính AutoIncrement được gán là true. mặc định là false.
AutoIncrementStep	long	lấy hay gán bước tăng dần được dùng. chỉ áp dụng khi thuộc tính tăng tự động (AutoIncrement) được gán là true. mặc định là 1.
Caption	string	Lấy hay gán tiêu đề cho cột. Tiêu đề cho cột được hiển thị trên những form Windows. giá trị mặc định là null.
ColumnName	string	Lấy hay gán tên của đối tượng DataColumn.
ColumnMapping	MappingType	Lấy hay gán MappingType của đối tượng DataColumn. Cái này xác định cách một DataColumn được lưu giữ trong một tài liệu XML sử dụng phương thức WriteXml() như thế nào.
DataType	Type	Lấy hay gán kiểu dữ liệu .NET đại diện cho giá trị cột được cất giữ trong đối tượng DataColumn. kiểu dữ liệu này có thể là đại số Boole, Byte, Char, DateTime, Decimal(số thập phân), Double, Int16, Int32, Int64, SByte, Single, string (chuỗi), TimeSpan, UInt16, Hay UInt64.
DafaultValue	object	Lấy hay gán giá trị mặc định cho DataColumn khi những hàng mới được tạo ra. Khi thuộc tính AutoIncrement (sự tăng tự động) được gán là true, giá trị mặc định không được sử dụng.
MaxLength	int	Lấy hay gán chiều dài cực đại của text ( văn bản) được cất giữ trong một đối tượng DataColumn. giá trị mặc định là -1.
Ordinal	int	Lấy số vị trí của đối tượng DataColumn (0 là đối tượng đầu tiên).
ReadOnly	bool	Lấy hay gán một giá trị bool cho biết liệu có phải đối tượng DataColumn có thể được thay đổi một khi nó được thêm vào một DataRow. mặc định là false.
Table	DataTable	Lấy DataTable mà đối tượng DataColumn thuộc về nó.
Unique	bool	Lấy hay gán một giá trị bool cho biết liệu có phải những giá trị DataColumn trong mỗi đối tượng DataRow là duy nhất. mặc định là false.

Bạn sẽ thấy những sử dụng một số thuộc tính, phương thức và những sự kiện này sau trong chương này.

## **THÊM NHỮNG HẠN CHẾ VÀO NHỮNG ĐỐI TƯỢNG DATATABLE VÀ DATACOLUMN**

Như bạn biết, một đối tượng Dataset được dùng để cất giữ một bản sao của một tập con của cơ sở dữ liệu. Chẳng hạn, bạn có thể cất giữ một bản sao của những hàng từ cơ sở dữ liệu vào trong một Dataset, với mỗi hàng được đại diện bởi một đối tượng DataTable. Một DataTable lưu trữ những cột trong những đối tượng DataColumn.

Ngoài việc cất giữ những hàng được truy xuất từ một bảng cơ sở dữ liệu, bạn cũng có thể thêm những hạn chế vào một DataTable và những đối tượng DataColumn của nó. Điều này cho phép bạn mô hình hóa những sự hạn chế như nhau đặt lên những bảng cơ sở dữ liệu và những cột trong DataTable của bạn và những đối tượng DataColumn. Chẳng hạn, bạn có thể thêm những sự ràng buộc sau đây tới một DataTable:

- Unique (giá trị duy nhất)
- Primary key (khóa chính)
- Foreign key (khóa phụ)

Ngoài ra, bạn có thể thêm những sự hạn chế sau đây tới một DataColumn:

- Liệu có phải cột có thể chấp nhận một giá trị null mà bạn cất giữ trong thuộc tính AllowDBNull của DataColumn.

- Bất kỳ thông tin về sự tăng tự động nào mà bạn cất giữ trong thuộc tính AutoIncrement, AutoIncrementSeed, và AutoIncrementStep của DataColumn. Bạn gán những thuộc tính này khi thêm những hàng tới một DataTable với một bảng cơ sở dữ liệu tương ứng có chứa một cột khóa chính. Cột ProductID của bảng Products là một ví dụ của một cột nhận dạng.

**Ghi chú:** ADO.NET sẽ không tự động phát sinh những giá trị cho những cột khóa chính trong một hàng mới. Chỉ cơ sở dữ liệu có thể làm điều đó. Bạn phải đọc giá trị khóa chính được phát sinh cho cột từ cơ sở dữ liệu.

Bạn sẽ thấy cách làm điều đó như thế nào sau trong những mục "*truy xuất những giá trị khóa chính mới*" Và

"*sử dụng những thủ tục lưu trữ để Thêm, Sửa đổi, và Loại bỏ những hàng từ Cơ sở dữ liệu.*". Đồng thời, nếu bảng cơ sở dữ liệu của bạn chứa những cột được gán một giá trị ngầm định, bạn cần phải đọc giá trị đó từ cơ sở dữ liệu. như thế tốt hơn sự thiết đặt thuộc tính DefaultValue của một DataColumn vì nếu giá trị ngầm định gán trong những thay đổi định nghĩa bảng cơ sở dữ liệu, bạn có thể chọn giá trị mới từ cơ sở dữ liệu hơn là phải thay đổi mã của bạn.

- Chiều dài cực đại của một chuỗi hay giá trị cột ký tự mà bạn cất giữ trong thuộc tính MaxLength của DataColumn.
- Liệu có phải cột là chỉ đọc- mà bạn cất giữ trong thuộc tính ReadOnly của DataColumn.
- Liệu có phải giá trị cột là duy nhất- mà bạn cất giữ trong thuộc tính Unique của DataColumn.

Bằng cách thêm những sự hạn chế này lên phía trước, bạn ngăn ngừa dữ liệu xấu thêm vào Dataset của bạn để bắt đầu với nó. Những trợ giúp này giảm thiểu những lỗi khi thử đẩy những thay đổi trong Dataset của bạn tới cơ sở dữ liệu. Nếu một người sử dụng chương trình của bạn cố thêm dữ liệu mà xâm phạm một sự hạn chế, chúng sẽ gây ra một ngoại lệ. Bạn có thể bắt ngoại lệ này trong chương trình của bạn và hiển thị một thông báo lỗi với những chi tiết. và người sử dụng có thể thay đổi dữ liệu họ đã cố thêm vào để chỉnh định vấn đề.

Bạn cũng cần định nghĩa một khóa chính trước khi bạn có thể tìm, lọc, và phân loại những đối tượng DataRow trong một DataTable. Bạn sẽ học làm điều đó như thế nào sau trong mục "*Tìm kiếm, Lọc và phân loại những hàng trong một DataTable.*"



**Mẹo nhỏ:** Thêm những sự ràng buộc gây ra một sự giảm cấp thực thi khi bạn gọi phương thức fill() của một DataAdapter. Đây là bởi vì những hàng được truy xuất được kiểm tra chống lại những sự ràng buộc của bạn trước khi chúng được thêm vào Dataset của bạn. do đó Bạn cần phải đặt thuộc tính EnforceConstraints của Dataset của bạn là false trước khi gọi phương thức Fill() . rồi Bạn gán EnforceConstraints trả lại giá trị mặc định true sau khi gọi phương thức Fill().

- Bạn có thể sử dụng một trong những cách sau để thêm những sự hạn chế vào những đối tượng DataTable và DataColumn. Tự bạn thêm những sự hạn chế bởi việc gán những thuộc tính của những đối tượng DataTable và DataColumn của bạn. điều này dẫn đến sự thực hiện mã nhanh nhất
- Gọi phương thức FillSchema() của DataAdapter của bạn để sao chép thông tin mô hình từ cơ sở dữ liệu đến Dataset của bạn. Điều này cư trú những thuộc tính của những đối tượng DataTable và những đối tượng DataColumn của chúng một cách tự động.

**Bạn sẽ học những chi tiết của cả hai kỹ thuật này trong những mục sau .**

## **TỰ THÊM NHỮNG HẠN CHẾ BẢNG TAY:**

Bạn có thể thêm những sự hạn chế vào những đối tượng DataTable và DataColumn của bạn sử dụng những thuộc tính của đối tượng DataTable và DataColumn.

Thí dụ, giả thiết bạn có một đối tượng Dataset có tên myDataSet có chứa ba đối tượng DataTable đặt tên là Products, Orders, và Details đã được cư trú sử dụng mã sau đây:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT ProductID, ProductName " +
    "FROM Products;" +
    "SELECT OrderID " +
    "FROM Orders;" +
    "SELECT OrderID, ProductID, UnitPrice " +
    "FROM [Order Details];";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Products";
myDataSet.Tables["Table1"].TableName = "Orders";
myDataSet.Tables["Table2"].TableName = "Order Details";
```

Khóa chính cho bảng Products là cột ProductID; khóa chính cho bảng Orders là cột OrderID; và khóa chính cho bảng Order Details được tạo ra từ cả hai cột OrderID lẫn ProductID.

**Ghi chú :** Bạn phải bao gồm tất cả những cột của khóa chính của bảng cơ sở dữ liệu trong câu truy vấn của bạn nếu Bạn muốn định nghĩa một khóa chính trên những cột đó trong DataTable của bạn.

**Trong những mục sau đây, bạn sẽ hiểu làm thế nào**

- Thêm những sự ràng buộc vào những đối tượng DataTable : Products, Orders và Order Details.
- Hạn chế những giá trị được đặt trong những đối tượng DataColumn của DataTable Products.

## **THÊM NHỮNG RÀNG BUỘC VÀO NHỮNG ĐỐI TƯỢNG DATATABLE:**

Trong mục này, bạn sẽ thấy cách thêm những sự ràng buộc vào những đối tượng DataTable . Đặc biệt, bạn sẽ thấy cách để thêm những sự ràng buộc khóa chính vào những đối tượng DataTable : Products, Orders, và Order Details. Một sự ràng buộc khóa chính thật sự được thực hiện như một sự ràng buộc duy nhất. Bạn cũng sẽ thấy cách thêm những sự ràng buộc khóa phụ từ những đối tượng DataTable Order Detail đến Products và Orders .

Những sự ràng buộc được cất giữ trong một đối tượng ConstraintCollection mà cất giữ những đối tượng ràng buộc(Constraint). Bạn truy nhập ConstraintCollection sử dụng những thuộc tính ràng buộc(Constraint) của đối tượng DataTable. Để thêm một đối tượng ràng buộc mới tới ConstraintCollection, Bạn gọi phương thức Add() thông qua thuộc tính Constraints. Phương thức Add() cho phép bạn thêm những sự ràng buộc khóa chính và khóa phụ vào một DataTable. Vì một ràng buộc khóa chính được thực hiện như một ràng buộc duy nhất, bạn cũng có thể sử dụng phương thức Add() để thêm một ràng buộc khóa chính vào một DataTable. Bạn sẽ thấy cách sử dụng phương thức Add() tiếp sau đây .

Bạn có thể cũng thêm một sự ràng buộc khóa chính vào một đối tượng DataTable bởi thiết đặt thuộc tính PrimaryKey của nó, mà bạn gán tới một mảng của những đối tượng DataColumn thuộc khóa chính . Một mảng được yêu cầu bởi vì khóa chính của một bảng cơ sở dữ liệu có thể được tạo ra từ nhiều cột. Như bạn sẽ thấy trong những ví dụ, điều này đơn giản hơn so với sử dụng phương thức Add() để thêm một ràng buộc khóa chính.

---

### **Gọi phương thức Fill() của một DataAdapter nhiều hơn một lần**

phương thức Fill() truy xuất tất cả những hàng từ bảng cơ sở dữ liệu, như chỉ rõ trong thuộc tính SelectCommand của đối tượng DataAdapter của các bạn. Nếu bạn thêm một khóa chính tới DataTable của bạn, rồi gọi phương thức Fill() hơn một lần sẽ đặt những hàng được truy xuất trong DataTable của bạn và xóa bỏ bất kỳ hàng hiện hữu nào với những giá trị cột khóa chính phù hợp đã có trong DataTable của bạn.

Nếu bạn không thêm một khóa chính tới DataTable của bạn , rồi gọi phương thức Fill() nhiều hơn một lần sẽ chỉ đơn giản là thêm tất cả những hàng được truy xuất vào DataTable của bạn lần nữa, sự trùng lặp những hàng sẽ xảy ra.

và đây là lý do khác để thêm một sự ràng buộc khóa chính vào DataTable của bạn bởi vì bạn không muốn có những hàng trùng lặp.

---

## **THÊM MỘT KHÓA CHÍNH VÀO DATATABLE PRODUCTS**

Chúng ta hãy xem xét việc thêm một khóa chính vào DataTable Products. Đầu tiên, ví dụ sau đây tạo ra một đối tượng DataTable tên productsDataTable và gán nó đến DataTable Products truy xuất từ myDataSet:

```
DataTable productsDataTable = myDataSet.Tables["Products"];
```

Bây giờ, khóa chính của bảng cơ sở dữ liệu Products là cột ProductID; bởi vậy, bạn cần đặt thuộc tính PrimaryKey của productsDataTable tới một mảng chứa đối tượng DataColumn ProductID. Ví dụ sau đây cho thấy cách thực hiện điều này . Nó tạo ra một mảng những đối tượng DataColumn tên productsPrimaryKey và khởi tạo đồng thời gán nó tới cột ProductID của productsDataTable, rồi đặt thuộc tính PrimaryKey của productsDataTable tới mảng:

```
        Khởi tạo đối tượng productsPrimaryKey (cột dữ liệu)
        DataColumn[] productsPrimaryKey = new DataColumn[]
        { gán cột ProductID thuộc bảng productsDataTable vào đối tượng
productsPrimaryKey
        productsDataTable.Columns["ProductID"]
        }; gán productsPrimaryKey thuộc tính PrimaryKey của ProductsDataTable
productsDataTable.PrimaryKey = productsPrimaryKey;
```

Khi bạn đặt thuộc tính PrimaryKey của một DataTable, những thuộc tính AllowDBNull và Unique của đối

tương DataColumn tự động được thay đổi như sau:

- Thuộc tính AllowDBNull được thay đổi tới false cho biết DataColumn không thể chấp nhận một giá trị null.
- Thuộc tính Unique được thay đổi là true và cho biết giá trị DataColumn trong mỗi DataRow phải là duy nhất.

Do đó, trong ví dụ trước đây, thuộc tính AllowDBNull và Unique của cột dữ liệu ProductID tự động được thay đổi tới giá trị false và true, tương ứng.

## **THÊM MỘT KHÓA CHÍNH VÀO BẢNG DỮ LIỆU(DATATABLE) ORDERS:**

Ví dụ sau đây đặt thuộc tính PrimaryKey của DataTable -Orders tới OrderID DataColumn:

```
myDataSet.Tables["Orders"].PrimaryKey = new DataColumn[]
{
    myDataSet.Tables["Orders"].Columns["OrderID"]
};
```

Chú ý : Tôi phải sử dụng một phát biểu trong ví dụ này để cho ngắn gọn hơn ví dụ trước đây.

Bạn cũng có thể sử dụng Thêm phương thức Add() để thêm một khóa unique, primary , hay foreign vào một DataTable. phương thức Add() được tải như sau:

```
void Add(Constraint myConstraint) // adds any constraint
void Add(string constraintName, DataColumn myDataColumn,bool isPrimaryKey)
// adds a primary key or unique constraint
void Add(string constraintName, DataColumn parentColumn,DataColumn childColumn)
// adds a foreign key constraint
void Add(string constraintName, DataColumn[] myDataColumn, bool isPrimaryKey)
// adds a primary key or unique constraint
void Add(string cosntraintName, DataColumn[] parentColumns,DataColumn[] childColumns)
// adds a foreign key constraint
```

**VỚI:**

**constraintName:** là tên bạn muốn gán tới sự ràng buộc của các bạn.

**isPrimaryKey** :cho biết liệu có phải ràng buộc là một ràng buộc khóa chính hay chỉ là một ràng buộc duy nhất bình thường.

Ví dụ sau đây sử dụng phương thức Add() để thêm một sự ràng buộc khóa chính vào DataTable Products:

```
myDataSet.Tables["Orders"].Constraints.Add( "Primary key constraint",
myDataSet.Tables["Orders"].Columns["OrderID"],true );
```

Ví dụ này thực hiện tương tự như ví dụ trước đây là thêm sự ràng buộc khóa chính sử dụng thuộc tính PrimaryKey. Chú ý tham số cuối cùng tới phương thức Add() được đặt là true, nó cho biết sự ràng buộc thuộc một khóa chính.

Ngoài ra, nếu bạn có một cột không là khóa chính nhưng có giá trị là duy nhất, bạn có thể thêm một đối tượng UniqueConstraint vào ConstraintsCollection. Chẳng hạn:

```
UniqueConstraint myUC =
    new UniqueConstraint(myDataTable.Columns["myColumn"]);
myDataTable.Constraints.Add(myUC);
```

## **THÊM MỘT KHÓA CHÍNH VÀO BẢNG DỮ LIỆU ORDER DETAILS:**

Chúng ta hãy xem xét một ví dụ về thiết đặt thuộc tính PrimaryKey cho bảng dữ liệu Order Details . Khóa chính cho bảng Order Details được tạo ra từ những cột OrderID và ProductID, và ví dụ sau đây cho thấy sự thiết đặt thuộc tính PrimaryKey của bảng dữ liệu Order Details tới hai cột này:

```
myDataSet.Tables["Order Details"].PrimaryKey =  
new DataColumn[]  
{  
    myDataSet.Tables["Order Details"].Columns["OrderID"],  
    myDataSet.Tables["Order Details"].Columns["ProductID"]  
};
```

Ví dụ sau đây sử dụng phương pháp Add() để thực hiện cùng công việc:

```
myDataSet.Tables["Order Details"].Constraints.Add(  
    "Primary key constraint",  
    new DataColumn[]  
    {  
        myDataSet.Tables["Order Details"].Columns["OrderID"],  
        myDataSet.Tables["Order Details"].Columns["ProductID"]  
    },  
    true  
);
```

Một điều cần phải nhớ khi thêm những sự ràng buộc vào một DataTable là nó chỉ biết về những hàng bạn cất giữ trong nó; nó không biết về bất cứ hàng nào khác được cất giữ trong bảng cơ sở dữ liệu thực tế. Để hiểu tại sao đây lại là một vấn đề, xem xét chuỗi sự kiện sau đây nó bao gồm những khóa chính:

1. Bạn thêm một sự ràng buộc khóa chính vào một DataTable.
2. Bạn truy xuất một tập con của những hàng từ một bảng cơ sở dữ liệu và cất giữ chúng trong DataTable của bạn.
3. Bạn thêm một DataRow mới vào DataTable của bạn với một giá trị khóa chính không phải được dùng trong tập con của những hàng được truy xuất vào trong DataTable của bạn trong những bước trước - nhưng giá trị khóa chính đã được sử dụng trong một hàng trong bảng cơ sở dữ liệu. DataRow mới của bạn được thêm vào DataTable không có bất kỳ lỗi nào mặc dù bạn thêm một sự ràng buộc khóa chính vào DataTable của bạn trong bước 1. DataRow mới của bạn được thêm vào một cách thành công bởi vì DataTable chỉ biết về những hàng được cất giữ trong nó, không phải là những hàng khác được cất giữ trong bảng cơ sở dữ liệu mà không được truy xuất trong bước 2.
4. Bạn thử đẩy DataRow mới tới cơ sở dữ liệu, nhưng bạn nhận một SqlException những trạng thái Bạn đã xâm phạm sự ràng buộc khóa chính trong bảng cơ sở dữ liệu. Đây là bởi vì một hàng trong bảng cơ sở dữ liệu đã sử dụng giá trị khóa chính.

Bạn cần nhớ đến vấn đề này khi thêm những hàng tới một DataTable, mà bạn sẽ thấy cách thực hiện không lâu sau đây.

Điều đó gồm thêm những sự ràng buộc khóa chính vào những đối tượng DataTable. Tiếp theo, bạn sẽ thấy cách thêm những sự ràng buộc khóa phụ như thế nào .

## **THÊM NHỮNG RÀNG BUỘC VÀO BẢNG DỮ LIỆU ORDER DETAILS:**

Trong mục này, bạn sẽ thấy cách thêm một ràng buộc khóa phụ vào DataTable -Order Details như thế nào . Để làm điều này bạn sử dụng phương thức Add() thông qua thuộc tính Constraints của DataTable.

Ví dụ sau đây thêm một ràng buộc khóa phụ từ DataColumn(cột dữ liệu) -OrderID của DataTable(bảng dữ liệu) -Order Details đến DataColumn OrderID của DataTable- Orders :

```

ForeignKeyConstraint myFKC = new ForeignKeyConstraint(
myDataSet.Tables["Orders"].Columns["OrderID"],
myDataSet.Tables["Order Details"].Columns["OrderID"]
);
myDataSet.Tables["Order Details"].Constraints.Add(myFKC);

```

**Ghi chú:** Chú ý là DataColumn cha(OrderID của Orders) được định nghĩa trước DataColumn con (OrderID của Order Details).

Ví dụ kế tiếp thêm một ràng buộc khóa phụ từ DataColumn ProductID của DataTable Order Details đến DataColumn ProductID của DataTable Products :

```

myDataSet.Tables["Order Details"].Constraints.Add(
"Foreign key constraint to ProductID DataColumn of the " +
"Products DataTable",
myDataSet.Tables["Order Details"].Columns["ProductID"],
myDataSet.Tables["Products"].Columns["ProductID"]
);

```

Điều đó bao gồm những ràng buộc vào những đối tượng DataTable. Tiếp theo, bạn sẽ thấy cách thêm những hạn chế vào những đối tượng DataColumn như thế nào

## **THÊM NHỮNG HẠN CHẾ VÀO NHỮNG ĐỐI TƯỢNG DATACOLUMN:**

Trong mục này, bạn sẽ thấy cách thêm những hạn chế vào những đối tượng DataColumn được cất giữ ở một DataTable như thế nào. Đặc biệt, bạn sẽ thấy cách thiết đặt cho những thuộc tính AllowDBNull, AutoIncrement(Sự tăng tự động), AutoIncrementSeed, AutoIncrementStep, ReadOnly, Và Unique (duy nhất) của DataColumn - ProductID của DataTable - Products. Bạn cũng sẽ thấy cách đặt thuộc tính MaxLength của DataColumn - ProductName của DataTable - Products như thế nào.

Cột ProductID của bảng cơ sở dữ liệu Products là một cột nhận dạng (khóa chính). seed là giá trị ban đầu và step (bước tăng) là sự tăng thêm giá trị đến số sau cuối và cả hai (seed và step) được gán là 1 cho ProductID. Do đó những giá trị nhận dạng ProductID sẽ là 1, 2, 3, vân vân.

**Mẹo nhỏ:** Khi bạn gán những thuộc tính AutoIncrementSeed lẫn AutoIncrementStep cho một DataColumn tương ứng với một cột nhận dạng cơ sở dữ liệu, bạn cần phải luôn luôn gán chúng tới -1. với cách này, khi bạn gọi phương thức Fill() , ADO.NET sẽ tự động tính những giá trị để gán cho AutoIncrementSeed và AutoIncrementStep, dựa vào những giá trị truy xuất được từ cơ sở dữ liệu, và bạn không cần phải tính đến những giá trị này.

### **Mã sau đây thiết đặt những thuộc tính của DataColumn - ProductID :**

```

DataColumn productIDDataColumn =
myDataSet.Tables["Products"].Columns["ProductID"];
productIDDataColumn.AllowDBNull = false;
productIDDataColumn.AutoIncrement = true;
productIDDataColumn.AutoIncrementSeed = -1;
productIDDataColumn.AutoIncrementStep = -1;
productIDDataColumn.ReadOnly = true;
productIDDataColumn.Unique = true;

```

Ví dụ tiếp theo đặt thuộc tính MaxLength (độ dài tối đa) của DataColumn - ProductName tới 40. điều này ngăn bạn thiết đặt giá trị của cột ProductName với một chuỗi lớn hơn 40 ký tự .

```

myDataSet.Tables["Products"].Columns["ProductName"].MaxLength = 40;

```

Liệt kê 11.1 trình bày những mã ví dụ sử dụng trong mục này và một mục trước. Chú ý chương trình này cũng trình bày những thuộc tính ColumnName và DataType của những đối tượng DataColumn trong mỗi DataTable. Thuộc tính ColumnName chứa tên của DataColumn, và DataType chứa kiểu dữ liệu .NET thường đại diện cho giá trị cột được cất giữ trong DataColumn.

### **Danh sách 11.1: ADDRESTRICTIONS.CS**

```
/*
AddRestrictions.cs illustrates how to add constraints to
DataTable objects and add restrictions to DataColumn objects
*/

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

class AddRestrictions
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT ProductID, ProductName " +
            "FROM Products;" +
            "SELECT OrderID " +
            "FROM Orders;" +
            "SELECT OrderID, ProductID, UnitPrice " +
            "FROM [Order Details];";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.Fill(myDataSet);
        mySqlConnection.Close();
        myDataSet.Tables["Table"].TableName = "Products";
        myDataSet.Tables["Table1"].TableName = "Orders";
        myDataSet.Tables["Table2"].TableName = "Order Details";

        // set the PrimaryKey property for the Products DataTable
        // to the ProductID column
        DataTable productsDataTable = myDataSet.Tables["Products"];
        DataColumn[] productsPrimaryKey =
            new DataColumn[]
            {
                productsDataTable.Columns["ProductID"]
            };
        productsDataTable.PrimaryKey = productsPrimaryKey;

        // set the PrimaryKey property for the Orders DataTable
        // to the OrderID column
        myDataSet.Tables["Orders"].PrimaryKey =
```

```

new DataColumn[]
{
    myDataSet.Tables["Orders"].Columns["OrderID"]
};

// set the PrimaryKey property for the Order Details DataTable
// to the OrderID and ProductID columns
myDataSet.Tables["Order Details"].Constraints.Add(
    "Primary key constraint on the OrderID and ProductID columns",
    new DataColumn[]
    {
        myDataSet.Tables["Order Details"].Columns["OrderID"],
        myDataSet.Tables["Order Details"].Columns["ProductID"]
    },
    true
);

// add a foreign key constraint on the OrderID column
// of Order Details to the OrderID column of Orders
ForeignKeyConstraint myFKC = new ForeignKeyConstraint(
    myDataSet.Tables["Orders"].Columns["OrderID"],
    myDataSet.Tables["Order Details"].Columns["OrderID"]
);
myDataSet.Tables["Order Details"].Constraints.Add(myFKC);
// add a foreign key constraint on the ProductID column
// of Order Details to the ProductID column of Products
myDataSet.Tables["Order Details"].Constraints.Add(
    "Foreign key constraint to ProductID DataColumn of the " +
    "Products DataTable",
    myDataSet.Tables["Products"].Columns["ProductID"],
    myDataSet.Tables["Order Details"].Columns["ProductID"]
);

// set the AllowDBNull, AutoIncrement, AutoIncrementSeed,
// AutoIncrementStep, ReadOnly, and Unique properties for
// the ProductID DataColumn of the Products DataTable
DataColumn productIDDataColumn =
    myDataSet.Tables["Products"].Columns["ProductID"];
productIDDataColumn.AllowDBNull = false;
productIDDataColumn.AutoIncrement = true;
productIDDataColumn.AutoIncrementSeed = -1;
productIDDataColumn.AutoIncrementStep = -1;
productIDDataColumn.ReadOnly = true;
productIDDataColumn.Unique = true;

// set the MaxLength property for the ProductName DataColumn
// of the Products DataTable
myDataSet.Tables["Products"].Columns["ProductName"].MaxLength = 40;

// display the details of the DataColumn objects for
// the DataTable objects
foreach (DataTable myDataTable in myDataSet.Tables)
{
    Console.WriteLine("\n\nReading from the " +
        myDataTable + "DataTable:\n");
}

```



```

// display the primary key
foreach (DataColumn myPrimaryKey in myDataTable.PrimaryKey)
{
    Console.WriteLine("myPrimaryKey = " + myPrimaryKey);
}

// display some of the details for each column
foreach (DataColumn myDataColumn in myDataTable.Columns)
{
    Console.WriteLine("\nmyDataColumn.ColumnName = " +
        myDataColumn.ColumnName);
    Console.WriteLine("myDataColumn.DataType = " +
        myDataColumn.DataType);

    Console.WriteLine("myDataColumn.AllowDBNull = " +
        myDataColumn.AllowDBNull);
    Console.WriteLine("myDataColumn.AutoIncrement = " +
        myDataColumn.AutoIncrement);
    Console.WriteLine("myDataColumn.AutoIncrementSeed = " +
        myDataColumn.AutoIncrementSeed);
    Console.WriteLine("myDataColumn.AutoIncrementStep = " +
        myDataColumn.AutoIncrementStep);
    Console.WriteLine("myDataColumn.MaxLength = " +
        myDataColumn.MaxLength);
    Console.WriteLine("myDataColumn.ReadOnly = " +
        myDataColumn.ReadOnly);
    Console.WriteLine("myDataColumn.Unique = " +
        myDataColumn.Unique);
}
}
}
}

```

### **Đầu ra từ chương trình này như sau:**

Reading from the Products DataTable:

myPrimaryKey = ProductID

myDataColumn.ColumnName = ProductID  
 myDataColumn.DataType = System.Int32  
 myDataColumn.AllowDBNull = False  
 myDataColumn.AutoIncrement = True  
 myDataColumn.AutoIncrementSeed = -1  
 myDataColumn.AutoIncrementStep = -1  
 myDataColumn.MaxLength = -1  
 myDataColumn.ReadOnly = True  
 myDataColumn.Unique = True

myDataColumn.ColumnName = ProductName  
 myDataColumn.DataType = System.String  
 myDataColumn.AllowDBNull = True  
 myDataColumn.AutoIncrement = False  
 myDataColumn.AutoIncrementSeed = 0  
 myDataColumn.AutoIncrementStep = 1



```
myDataColumn.MaxLength = 40  
myDataColumn.ReadOnly = False  
myDataColumn.Unique = False
```

Reading from the Orders DataTable:

```
myPrimaryKey = OrderID  
myDataColumn.ColumnName = OrderID  
myDataColumn.DataType = System.Int32  
myDataColumn.AllowDBNull = False  
myDataColumn.AutoIncrement = False  
myDataColumn.AutoIncrementSeed = 0  
myDataColumn.AutoIncrementStep = 1  
myDataColumn.MaxLength = -1  
myDataColumn.ReadOnly = False  
myDataColumn.Unique = True
```

Reading from the Order Details DataTable:

```
myPrimaryKey = OrderID  
myPrimaryKey = ProductID
```

```
myDataColumn.ColumnName = OrderID  
myDataColumn.DataType = System.Int32  
myDataColumn.AllowDBNull = False  
myDataColumn.AutoIncrement = False  
myDataColumn.AutoIncrementSeed = 0  
myDataColumn.AutoIncrementStep = 1  
myDataColumn.MaxLength = -1  
myDataColumn.ReadOnly = False  
myDataColumn.Unique = False
```

```
myDataColumn.ColumnName = ProductID  
myDataColumn.DataType = System.Int32  
myDataColumn.AllowDBNull = False  
myDataColumn.AutoIncrement = False  
myDataColumn.AutoIncrementSeed = 0  
myDataColumn.AutoIncrementStep = 1  
myDataColumn.MaxLength = -1  
myDataColumn.ReadOnly = False  
myDataColumn.Unique = False
```

```
myDataColumn.ColumnName = UnitPrice  
myDataColumn.DataType = System.Decimal  
myDataColumn.AllowDBNull = True  
myDataColumn.AutoIncrement = False  
myDataColumn.AutoIncrementSeed = 0  
myDataColumn.AutoIncrementStep = 1  
myDataColumn.MaxLength = -1  
myDataColumn.ReadOnly = False  
myDataColumn.Unique = False
```

**Thêm những hạn chế bởi việc gọi phương thức FillSchema() của đối tượng DataAdapter**

Thay vì thêm những sự hạn chế bởi chính bạn, bạn có thể thêm chúng bằng cách gọi phương thức FillSchema() của DataAdapter của bạn. Phương thức FillSchema() thực hiện những điều sau:

- Sao chép thông tin mô hình từ cơ sở dữ liệu.
- Tạo ra những đối tượng DataTable trong Dataset của bạn nếu chúng chưa tồn tại.
- Thêm những ràng buộc vào những đối tượng DataTable.
- Đặt những thuộc tính của những đối tượng DataColumn phù hợp.

Những thuộc tính của những đối tượng DataColumn thiết đặt bởi FillSchema() bao gồm :

- Tên DataColumn -được cất giữ trong thuộc tính ColumnName.
- kiểu dữ liệu DataColumn .NET - được cất giữ trong thuộc tính DataType.
- Chiều dài cực đại của một kiểu dữ liệu độ dài biến được cất giữ trong thuộc tính MaxLength.
- Cho biết DataColumn có thể chấp nhận một giá trị null - được cất giữ trong thuộc tính AllowDBNull.
- Cho biết giá trị DataColumn phải là duy nhất - được cất giữ trong thuộc tính Unique.
- Bất kỳ thông tin tăng tự động nào - được cất giữ trong những thuộc tính AutoIncrement (tăng tự động), AutoIncrementSeed(giá trị đầu), Và AutoIncrementStep(bước tăng).

phương thức FillSchema() cũng sẽ xác định liệu có phải DataColumn là bộ phận của một khóa chính và lưu trữ thông tin này trong thuộc tính Primarykey của DataTable.

**Cảnh báo:** Phương thức FillSchema() không tự động thêm những đối tượng ForeignKeyConstraint vào những đối tượng DataTable. Không có gì khiến nó truy xuất những hàng thực tế từ cơ sở dữ liệu; nó chỉ khôi phục thông tin mô hình.

phương thức FillSchema() bị quá tải, với phiên bản thường sử dụng nhất của phương thức này như sau:

`DataTable[] FillSchema(DataSet myDataSet, SchemaType mySchemaType)`

mySchemaType : chỉ rõ bạn cách muốn xử lý bất kỳ ánh xạ mô hình hiện hữu như thế nào.

Bạn gán mySchemaType tới một trong số những hằng số được định nghĩa trong lớp liệt kê System.Data.SchemaType. Bảng 11.7 cho thấy rằng những hằng số định nghĩa trong lớp liệt kê SchemaType.

**Bảng 11.7: những thành viên Liệt kê SchemaType**

Hằng số	Mô tả
Mapped	áp dụng cho bất kỳ ánh xạ bảng hiện hữu nào vào mô hình đầu vào và định hình DataSet với mô hình biến đổi. Đây là hằng số bạn cần phải sử dụng cách điển hình.
Source	bỏ qua bất kỳ ánh xạ bảng nào và định hình Dataset không có bất kỳ sự biến đổi nào.

Chúng ta hãy xem xét một ví dụ có chứa một lệnh gọi tới phương thức FillSchema() . Chú ý lệnh gọi sử dụng hằng số SchemaType.Mapped áp dụng cho bất kỳ ánh xạ bảng hiện hữu nào:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT ProductID, ProductName " +
    "FROM Products;" +
    "SELECT OrderID " +
    "FROM Orders;" +
    "SELECT OrderID, ProductID, UnitPrice " +
    "FROM [Order Details];";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
```

```

mySqlConnection.Open();
mySqlDataAdapter.FillSchema(myDataSet, SchemaType.Mapped);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Products";
myDataSet.Tables["Table1"].TableName = "Orders";
myDataSet.Tables["Table2"].TableName = "Order Details";

```

Sự gọi tới phương thức FillSchema() sẽ sao chép thông tin mô hình từ những bảng Products, Orders, và Order Details tới myDataSet, thiết đặt thuộc tính PrimaryKey của mỗi DataTable và những thuộc tính của những đối tượng DataColumn cách phù hợp.

### **Danh sách 11.2 cho thấy sự sử dụng của phương thức FillSchema().**

```

/*
 FillSchema.cs illustrates how to read schema information
 using the FillSchema() method of a DataAdapter object
 */

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

class FillSchema
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT ProductID, ProductName " +
            "FROM Products;" +
            "SELECT OrderID " +
            "FROM Orders;" +
            "SELECT OrderID, ProductID, UnitPrice " +
            "FROM [Order Details];";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.FillSchema(myDataSet, SchemaType.Mapped);
        mySqlConnection.Close();
        myDataSet.Tables["Table"].TableName = "Products";
        myDataSet.Tables["Table1"].TableName = "Orders";
        myDataSet.Tables["Table2"].TableName = "Order Details";

        // display the details of the DataColumn objects for
        // the DataTable objects
        foreach (DataTable myDataTable in myDataSet.Tables)
        {
            Console.WriteLine("\n\nReading from the " +
                myDataTable + "DataTable:\n");

```

```

// display the primary key
foreach (DataColumn myPrimaryKey in myDataTable.PrimaryKey)
{
    Console.WriteLine("myPrimaryKey = " + myPrimaryKey);
}

// display the constraints
foreach (Constraint myConstraint in myDataTable.Constraints)
{
    Console.WriteLine("myConstraint.IsPrimaryKey = " + ((UniqueConstraint)
        myConstraint).IsPrimaryKey);
    foreach (DataColumn myDataColumn in ((UniqueConstraint)
        myConstraint).Columns)
    {
        Console.WriteLine("myDataColumn.ColumnName = " +
            myDataColumn.ColumnName);
    }
}

// display some of the details for each column
foreach (DataColumn myDataColumn in myDataTable.Columns)
{
    Console.WriteLine("\nmyDataColumn.ColumnName = " +
        myDataColumn.ColumnName);
    Console.WriteLine("myDataColumn.DataType = " +
        myDataColumn.DataType);

    Console.WriteLine("myDataColumn.AllowDBNull = " +
        myDataColumn.AllowDBNull);
    Console.WriteLine("myDataColumn.AutoIncrement = " +
        myDataColumn.AutoIncrement);
    Console.WriteLine("myDataColumn.AutoIncrementSeed = " +
        myDataColumn.AutoIncrementSeed);
    Console.WriteLine("myDataColumn.AutoIncrementStep = " +
        myDataColumn.AutoIncrementStep);
    Console.WriteLine("myDataColumn.MaxLength = " +
        myDataColumn.MaxLength);
    Console.WriteLine("myDataColumn.ReadOnly = " +
        myDataColumn.ReadOnly);
    Console.WriteLine("myDataColumn.Unique = " +
        myDataColumn.Unique);
}
}
}
}

```

**The output from this program is as follows:**

Reading from the Products DataTable:

```

myPrimaryKey = ProductID
myConstraint.IsPrimaryKey = True
myDataColumn.ColumnName = ProductID

```

```

myDataColumn.ColumnName = ProductID
myDataColumn.DataType = System.Int32

```

```
myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = True
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = -1
myDataColumn.ReadOnly = True
myDataColumn.Unique = True
```

```
myDataColumn.ColumnName = ProductName
myDataColumn.DataType = System.String
myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = False
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = 40
myDataColumn.ReadOnly = False
myDataColumn.Unique = False
```

Reading from the Orders DataTable:

```
myPrimaryKey = OrderID
myConstraint.IsPrimaryKey = True
myDataColumn.ColumnName = OrderID
```

```
myDataColumn.ColumnName = OrderID
myDataColumn.DataType = System.Int32
myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = True
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = -1
myDataColumn.ReadOnly = True
myDataColumn.Unique = True
```

Reading from the Order Details DataTable:

```
myPrimaryKey = OrderID
myPrimaryKey = ProductID
myConstraint.IsPrimaryKey = True
myDataColumn.ColumnName = OrderID
myDataColumn.ColumnName = ProductID
```

```
myDataColumn.ColumnName = OrderID
myDataColumn.DataType = System.Int32
myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = False
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = -1
myDataColumn.ReadOnly = False
myDataColumn.Unique = False
```

```
myDataColumn.ColumnName = ProductID
myDataColumn.DataType = System.Int32
```

```

myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = False
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = -1
myDataColumn.ReadOnly = False
myDataColumn.Unique = False
myDataColumn.ColumnName = UnitPrice
myDataColumn.DataType = System.Decimal
myDataColumn.AllowDBNull = False
myDataColumn.AutoIncrement = False
myDataColumn.AutoIncrementSeed = 0
myDataColumn.AutoIncrementStep = 1
myDataColumn.MaxLength = -1
myDataColumn.ReadOnly = False
myDataColumn.Unique = False

```

## **TÌM KIẾM, LỌC, PHÂN LOẠI NHỮNG HÀNG TRONG MỘT DATATABLE:**

Mỗi hàng trong một DataTable được cất giữ trong một đối tượng DataRow, và trong mục này bạn sẽ học cách tìm, lọc, và phân loại những đối tượng DataRow ở một DataTable như thế nào.

## **TÌM MỘT DATAROW TRONG MỘT DATATABLE:**

Để tìm một DataRow trong một DataTable, bạn theo những bước sau:

1. Truy xuất những hàng từ cơ sở dữ liệu vào trong DataTable của bạn.
2. Thiết đặt thuộc tính PrimaryKey của DataTable của các bạn.
3. Gọi phương thức Find() của DataTable của bạn, thông qua giá trị cột khóa chính của DataRow bạn cần tìm.

Chẳng hạn, mã sau đây thực hiện những bước 1 và 2 trong danh sách này, truy xuất 10 hàng đầu tiên từ bảng Products và thiết đặt thuộc tính Primarykey tới cột dữ liệu ProductID:

```

SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT TOP 10 ProductID, ProductName " +
    "FROM Products " +
    "ORDER BY ProductID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet, "Products");
mySqlConnection.Close();
DataTable productsDataTable = myDataSet.Tables["Products"];
productsDataTable.PrimaryKey =
    new DataColumn[] {productsDataTable.Columns["ProductID"]};

```

Tiếp theo, ví dụ sau đây thực hiện bước 3: gọi phương thức Find() để truy xuất DataRow có ProductID là 3 từ productsDataTable

```
DataRow productDataRow = productsDataTable.Rows.Find("3");
```

Chú ý : phương thức Find() được gọi thông qua thuộc tính những hàng của productsDataTable. Thuộc tính những hàng trả lại một đối tượng của lớp DataRowCollection .

Nếu khóa chính cho bảng cơ sở dữ liệu gồm có nhiều cột, thì bạn có thể thông qua một mảng của những đối tượng chuyển tới phương thức Find(). Chẳng hạn, những khóa chính của bảng Order Details được tạo ra từ những cột OrderID và ProductID. Giả thiết bạn đã thực hiện những bước 1 và 2 và truy xuất những hàng từ những bảng Order Details vào trong một đối tượng DataTable có tên orderDetailsDataTable, Rồi theo ví dụ sau đây truy xuất DataRow với một OrderID và ProductID có giá trị 10248 và 11 tương ứng.

```
object[] orderDetails = new object[] { 10248, 11 };  
DataRow orderDetailDataRow = orderDetailsDataTable.Row.Find(orderdetail);
```

## **SỰ LỌC VÀ PHÂN LOẠI NHỮNG ĐỐI TƯỢNG DATAROW Ở MỘT DATATABLE:**

Để lọc và phân loại những đối tượng DataRow trong một DataTable, bạn sử dụng phương thức Select() của DataTable của bạn. Phương thức Select() được tải như sau:

```
DataRow[] Select()  
DataRow[] Select(string filterExpression)  
DataRow[] Select(string filterExpression, string sortExpression)  
DataRow[] Select(string filterExpression, string sortExpression,  
    DataViewRowState myDataViewRowState)
```

Với

**filterExpression** : chỉ định những hàng để chọn.

**sortExpression** : chỉ rõ những hàng được lựa chọn sẽ được sắp xếp như thế nào.

**myDataViewRowState** : chỉ định trạng thái của những hàng để chọn. Bạn gán myDataViewRowState tới một trong số những hằng số được định nghĩa trong lớp liệt kê System.Data.DataViewRowState .

Bảng 11.8 trình bày những hằng số này.

**Bảng 11.8: những thành viên Liệt kê DataViewRowState**

Hằng số	Mô tả
Added	Một hàng mới.
CurrentRows	Những hàng hiện thời, mà bao gồm những hàng Unchanged (không thay đổi), Added(đã thêm vào), và ModifiedCurrent (đã thay đổi) .
Deleted	Một hàng đã xóa
ModifiedCurrent	Một hàng hiện thời đã được sửa đổi.
ModifiedOriginal	Hàng nguyên bản trước khi nó được sửa đổi.
None	Không phù hợp với bất kỳ hàng nào của những hàng trong DataTable.
OriginalRows	Những hàng nguyên bản, mà bao gồm những hàng không thay đổi và đã bị xóa .
Unchanged	Một hàng mà không được thay đổi.

Chúng ta hãy xem xét vài ví dụ sử dụng phương thức Select() .

Ví dụ sau đây gọi phương thức Select() không có những tham số, nó trả lại tất cả các hàng trong DataTable không có bất kỳ sự lọc hay sắp xếp nào

```
DataRow[] productDataRows = productsDataTable.Select();
```

Ví dụ kế tiếp cung cấp một biểu thức lọc Sort(), để trả lại những đối tượng DataRow duy nhất với những giá trị DataColumn ProductID nhỏ hơn hay bằng 5

```
DataRow[] productDataRows = productsDataTable.Select("ProductID <= 5");
```

Ví dụ sau đây cung cấp cả hai : một biểu thức lọc và một biểu thức phân loại nó sắp xếp những đối tượng DataRow với những giá trị ProductID giảm dần:

```
DataRow[] productDataRows = productsDataTable.Select("ProductID <= 5", "ProductID DESC");
```

Như bạn có thể thấy từ những ví dụ trước đây, những biểu thức lọc và phân loại thì tương tự như mệnh đề WHERE và ORDER BY trong phát biểu SELECT. Do đó bạn có thể sử dụng những biểu thức mạnh mẽ trong những lệnh gọi của bạn tới phương thức Sort(). ví dụ, bạn có thể sử dụng những toán tử so sánh AND, OR, NOT, IN, LIKE, những toán tử số học, những ký tự đại diện, và những hàm tổng thể trong những biểu thức lọc của bạn.

**Ghi chú :** cho những chi tiết đầy đủ về cách sử dụng những biểu thức lọc, tham chiếu tới thuộc tính DataColumn.Expression trong tài liệu trực tuyến .NET.

Ví dụ sau đây sử dụng toán tử LIKE và ký tự đại diện (%) - phù hợp với bất kỳ số lượng ký tự nào - để lọc những hàng có ProductName bắt đầu với 'cha'. Ví dụ cũng phân loại và sắp xếp những hàng với giá trị ProductID giảm dần và ProductName tăng dần .

```
productDataRows =  
productsDataTable.Select("ProductName LIKE 'Cha%'",  
"ProductID DESC, ProductName ASC");
```

**Chú ý:** chuỗi Cha% được đặt trong những trích dẫn đơn ('Cha%'), mà bạn phải thực hiện cho tất cả các ký tự chuỗi.

**Ghi nhớ :** Bạn cũng có thể sử dụng một đối tượng DataView để lọc và phân loại những hàng, và bạn sẽ học cách làm trong Chương 13, " cách sử dụng những đối tượng DataView ."

Danh sách 11.3 trình bày một chương trình thực hiện việc tìm , lọc, và sắp xếp những đối tượng DataRow .

### Danh sách 11.3: FINDFILTERANDSORTDATAROWS.CS

```
/*  
FindFilterAndSortDataRows.cs illustrates how to find, filter,  
and sort DataRow objects  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class FindFilterAndSortDataRows  
{  
    public static void Main()  
    {  
        SqlConnection mySqlConnection =  
            new SqlConnection(  
                "server=localhost;database=Northwind;uid=sa;pwd=sa"  
            );  
  
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
        mySqlCommand.CommandText =  
            "SELECT TOP 10 ProductID, ProductName "  
            "FROM Products " +
```



```

"ORDER BY ProductID;" +
"SELECT TOP 10 OrderID, ProductID, UnitPrice, Quantity " +
"FROM [Order Details] " +
"ORDER BY OrderID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Products";
myDataSet.Tables["Table1"].TableName = "Order Details";

// set the PrimaryKey property for the Products DataTable
// to the ProductID column
DataTable productsDataTable = myDataSet.Tables["Products"];
productsDataTable.PrimaryKey =
new DataColumn[]
{
    productsDataTable.Columns["ProductID"]
};

// set the PrimaryKey property for the Order Details DataTable
// to the OrderID and ProductID columns
DataTable orderDetailsDataTable = myDataSet.Tables["Order Details"];
orderDetailsDataTable.Constraints.Add(
    "Primary key constraint on the OrderID and ProductID columns",
new DataColumn[] {orderDetailsDataTable.Columns["OrderID"],
    orderDetailsDataTable.Columns["ProductID"]},true);
// find product with ProductID of 3 using the Find() method
// to locate the DataRow using its primary key value
Console.WriteLine("Using the Find() method to locate DataRow object " +
    "with a ProductID of 3");
DataRow productDataRow = productsDataTable.Rows.Find("3");
foreach (DataColumn myDataColumn in productsDataTable.Columns)
{
    Console.WriteLine(myDataColumn + " = " + productDataRow[myDataColumn]);
}

// find order with OrderID of 10248 and ProductID of 11 using
// the Find() method
Console.WriteLine("Using the Find() method to locate DataRow object " +
    "with an OrderID of 10248 and a ProductID of 11");
object[] orderDetails = new object[] {10248,11};
DataRow orderDetailDataRow = orderDetailsDataTable.Rows.Find(orderDetails);
foreach (DataColumn myDataColumn in orderDetailsDataTable.Columns)
{
    Console.WriteLine(myDataColumn + " = " + orderDetailDataRow[myDataColumn]);
}

// filter and sort the DataRow objects in productsDataTable
// using the Select() method
Console.WriteLine("Using the Select() method to filter and sort DataRow objects");
DataRow[] productDataRows =
productsDataTable.Select("ProductID <= 5", "ProductID DESC",
    DataViewRowState.OriginalRows);

```

```

foreach (DataRow myDataRow in productDataRows)
{
    foreach (DataColumn myDataColumn in productsDataTable.Columns)
    {
        Console.WriteLine(myDataColumn + " = " + myDataRow[myDataColumn]);
    }
}

// filter and sort the DataRow objects in productsDataTable
// using the Select() method
Console.WriteLine("Using the Select() method to filter and sort DataRow objects");
productDataRows =
    productsDataTable.Select("ProductName LIKE 'Cha*'",
        "ProductID ASC, ProductName DESC");
foreach (DataRow myDataRow in productDataRows)
{
    foreach (DataColumn myDataColumn in productsDataTable.Columns)
    {
        Console.WriteLine(myDataColumn + " = " + myDataRow[myDataColumn]);
    }
}
}

```

**The output from this program is as follows:**

```

Using the Find() method to locate DataRow object with a ProductID of 3
ProductID = 3
ProductName = Aniseed Syrup
Using the Find() method to locate DataRow object with an OrderID of 10248 and a
ProductID of 11
OrderID = 10248
ProductID = 11
UnitPrice = 14
Quantity = 12
Using the Select() method to filter and sort DataRow objects
ProductID = 5
ProductName = Chef Anton's Gumbo Mix
ProductID = 4
ProductName = Chef Anton's Cajun Seasoning
ProductID = 3
ProductName = Aniseed Syrup
ProductID = 2
ProductName = Chang
ProductID = 1
ProductName = Chai
Using the Select() method to filter and sort DataRow objects
ProductID = 1
ProductName = Chai
ProductID = 2
ProductName = Chang

```

**SỬ ĐỔI NHỮNG HÀNG TRONG MỘT DATATABLE:**

Trong mục này, bạn sẽ thấy những bước được yêu cầu để thêm, sửa đổi, và loại bỏ những đối tượng DataRow từ một DataTable và sau đó đẩy những thay đổi đến cơ sở dữ liệu. Những ví dụ trong mục này chỉ ra cách để thêm, sửa đổi, và xóa những hàng trong bảng cơ sở dữ liệu Customers.

**Ghi nhớ:** Bạn sẽ tìm thấy một chương trình đầy đủ có tên AddModifyAndRemoveDataRows.cs trong thư mục ch11, nó minh họa cách sử dụng những phương thức đã trình bày trong mục này. Chương trình này bị bỏ qua trong sách này để cho ngắn gọn.

## **THIẾT ĐẶT MỘT *DataAdapter* ĐỂ ĐẨY NHỮNG THAY ĐỔI TỚI CƠ SỞ DỮ LIỆU:**

Trong chương 10, bạn đã thấy trước khi bạn gọi phương thức Fill() của DataAdapter của bạn để đọc những hàng từ cơ sở dữ liệu, đầu tiên bạn cần thiết đặt thuộc tính SelectCommand của DataAdapter của bạn. Ví dụ:

```
SqlCommand mySelectCommand = mySqlConnection.CreateCommand();
mySelectCommand.CommandText =
    "SELECT CustomerID, CompanyName, Address " +
    "FROM Customers " +
    "ORDER BY CustomerID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySelectCommand;
```

Phát biểu SELECT rồi được chạy khi bạn gọi phương thức Fill() của đối tượng mySqlDataAdapter để truy xuất những hàng từ bảng Customers vào trong một dataset.

Tương tự, trước khi bạn có thể đẩy những thay đổi tới cơ sở dữ liệu, đầu tiên bạn phải thiết lập DataAdapter của bạn với những đối tượng Command chứa những phát biểu SQL INSERT, UPDATE, và DELETE tương ứng. Bạn cất giữ những đối tượng lệnh này trong những thuộc tính InsertCommand, UpdateCommand, Và DeleteCommand của đối tượng DataAdapter của bạn .

Bạn đẩy những thay đổi từ Dataset của bạn đến cơ sở dữ liệu sử dụng phương thức Update() của DataAdapter của bạn. Khi bạn thêm, điều chỉnh, hay loại bỏ những đối tượng DataRow từ Dataset của bạn và sau đó gọi là phương thức Update() của DataAdapter, những lệnh InsertCommand, UpdateCommand, hay DeleteCommand thích hợp sẽ được chạy để đẩy những sự thay đổi của bạn đến cơ sở dữ liệu.

Chúng ta hãy xem xét cách thiết đặt những thuộc tính InsertCommand, UpdateCommand, Và DeleteCommand của một DataAdapter.

## **ĐẶT THUỘC TÍNH INSERTCOMMAND CỦA MỘT *DataAdapter*:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myInsertCommand chứa một phát biểu INSERT:

```
SqlCommand myInsertCommand = mySqlConnection.CreateCommand();
myInsertCommand.CommandText =
    "INSERT INTO Customers (" +
    " CustomerID, CompanyName, Address" +
    ") VALUES (" +
    " @CustomerID, @CompanyName, @Address" +
    ")";
myInsertCommand.Parameters.Add("@CustomerID", SqlDbType.NChar,
    5, "CustomerID");
myInsertCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar,
    40, "CompanyName");
myInsertCommand.Parameters.Add("@Address", SqlDbType.NVarChar,
    60, "Address");
```

### **Bổn tham số tới phương thức Add() như sau:**

- Tên của tham số

- Kiểu tham số .NET.
- Chiều dài cực đại của chuỗi mà có thể gán tới giá trị của tham số
- Tên của cột cơ sở dữ liệu tương ứng mà tham số được buộc vào

**Ghi nhớ:** những lệnh và những tham số được nói qua trong Chương 8, " thực thi những lệnh Cơ sở dữ liệu."

Như bạn có thể thấy trong mã trước đây, những tham số @CustomerID, @CompanyName, và @Address được buộc kết tới những cột CustomerID, CompanyName, và Address trong cơ sở dữ liệu.

Tiếp theo, ví dụ sau đây gán thuộc tính InsertCommand của mySqlDataAdapter tới myInsertCommand:

```
mySqlDataAdapter.InsertCommand = myInsertCommand;
```

### **THIẾT ĐẶT THUỘC TÍNH UpdateCommand CỦA MỘT DataAdapter:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myUpdateCommand có chứa một phát biểu UPDATE và thiết đặt thuộc tính UpdateCommand của mySqlDataAdapter tới myUpdateCommand:

```
myUpdateCommand.CommandText =
    "UPDATE Customers " +
    "SET " +
    " CompanyName = @NewCompanyName, " +
    " Address = @NewAddress " +
    "WHERE CustomerID = @OldCustomerID " +
    "AND CompanyName = @OldCompanyName " +
    "AND Address = @OldAddress";
myUpdateCommand.Parameters.Add("@NewCompanyName", SqlDbType.NVarChar,
    40, "CompanyName");
myUpdateCommand.Parameters.Add("@NewAddress", SqlDbType.NVarChar,
    60, "Address");
myUpdateCommand.Parameters.Add("@OldCustomerID", SqlDbType.NChar,
    5, "CustomerID");
myUpdateCommand.Parameters.Add("@OldCompanyName", SqlDbType.NVarChar,
    40, "CompanyName");
myUpdateCommand.Parameters.Add("@OldAddress", SqlDbType.NVarChar,
    60, "Address");
myUpdateCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
myUpdateCommand.Parameters["@OldCompanyName"].SourceVersion =
    DataRowVersion.Original;
myUpdateCommand.Parameters["@OldAddress"].SourceVersion =
    DataRowVersion.Original;
mySqlDataAdapter.UpdateCommand = myUpdateCommand;
```

### **Có hai điều cần chú ý về mã này:**

- Mệnh đề WHERE của phát biểu UPDATE chỉ rõ những tham số cho những cột CompanyID, CompanyName, và Address. Sự tranh chấp lạc quan những sử dụng này, mà bạn sẽ học về nó sau .
- Một thuộc tính có tên SourceVersion cho những tham số @OldCustomerID, @OldCompanyName và @OldAddress được gán tới DataRowVersion.Original. điều này gây ra những giá trị cho những tham số này được thiết lập tới những giá trị cột DataRow nguyên bản trước lúc bạn thay đổi chúng.

Những tiết mục này xác định sự tranh chấp của UPDATE, mà bạn sẽ học ngay sau đây.

## **SỰ TRANH CHẤP:**

Sự tương tranh quyết định những sự thay đổi của nhiều người sử dụng tới cùng một hàng được xử lý như thế nào. Có hai kiểu tương tranh ứng dụng cho một Dataset:

**Sự tương tranh lạc quan:** với sự tương tranh lạc quan, bạn có thể sửa đổi một hàng trong một bảng cơ sở dữ liệu chỉ khi không ai khác cùng sửa đổi chính hàng đó từ khi bạn tải nó vào trong Dataset của bạn. Đây điển hình là kiểu tốt nhất của sự tương tranh để sử dụng, bởi vì bạn không muốn ghi đè lên những thay đổi của một người nào đó.

**Sự tương tranh " cái sau cùng thắng " ("Last One Wins" Concurrency)** với sự tương tranh " cái sau cùng thắng " , bạn có thể luôn luôn sửa đổi một hàng- và những sự thay đổi của các bạn ghi đè lên những sự thay đổi của bất cứ ai . Điển hình bạn luôn muốn tránh sử dụng sự tương tranh " một sự chiến thắng cuối cùng " .

Để sử dụng sự tương tranh lạc quan, bạn phải thực hiện như sau trong mệnh đề WHERE của phát biểu UPDATE hay DELETED của bạn:

1. Bao gồm tất cả những cột được dùng trong SELECT nguyên bản.
2. Gán những giá trị cột này tới những giá trị ban đầu được truy xuất từ hàng trong bảng trước lúc bạn thay đổi những giá trị.

Khi bạn thực hiện hai điều này trong sự mệnh đề WHERE của phát biểu UPDATE hay DELETED của bạn , đầu tiên phát biểu kiểm tra xem hàng nguyên bản vẫn còn tồn tại không trước khi cập nhật hay xóa hàng. Cách này, bạn có thể chắc chắn những sự thay đổi của bạn không ghi đè lên những sự thay đổi của bất cứ ai . Tất nhiên, nếu hàng nguyên bản đã được xóa bởi người sử dụng khác, và phát biểu cập nhật hay xóa bỏ của bạn sẽ thất bại.

Để sử dụng sự tranh chấp " cái sau cùng thắng " , bạn phải bao gồm khóa chính và giá trị của nó trong mệnh đề WHERE của phát biểu UPDATE hay DELETE của bạn . Vì phát biểu UPDATE của bạn không kiểm tra những giá trị ban đầu, nó đơn giản ghi đè lên những thay đổi của bất cứ ai nếu như hàng vẫn còn tồn tại. Đồng thời, một phát biểu DELETE đơn giản xóa hàng cho dù người sử dụng khác đã sửa đổi hàng đó.

Trở lại ví dụ mã trước đây, nó thiết đặt thuộc tính UpdateCommand của mySqlDataAdapter, Bạn có thể thấy tất cả những cột đều được bao gồm trong mệnh đề WHERE của phát biểu UPDATE . Điều đó thỏa mãn yêu cầu đầu tiên của việc sử dụng sự tương tranh lạc quan được chỉ ra trước đó.

Yêu cầu thứ hai là bạn đặt cột trong mệnh đề WHERE tới những giá trị hàng nguyên thủy. Bạn làm điều này bởi thiết đặt thuộc tính SourceVersion của những tham số @OldCustomerID, @OldCompanyName, Và @OldAddress tới DataRowVersion.Original. Khi thực hiện, điều này này kéo những giá trị nguyên thủy từ những đối tượng DataColumn trong DataRow trước khi bạn thay đổi chúng và đặt chúng vào mệnh đề WHERE của phát biểu UPDATE.

Original - chính là một trong những phần tử của lớp liệt kê (enumeration) System.Data.DataRowVersion ; những cái khác được trình bày trong Bảng 11.9.

**Bảng 11.9: những thành viên Liệt kê DataRowVersion**

Hằng số	Mô tả
Current	Giá trị cột hiện tại
Default	Giá trị mặc định của cột
Original	Giá trị cột nguyên bản.

Hàng số	Mô tả
Proposed	Giá trị cột được đề xướng, nó được gán khi bạn soạn thảo một DataRow sử dụng phương thức BeginEdit() .

## **THIẾT ĐẶT THUỘC TÍNH DeleteCommand CỦA MỘT DataAdapter:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myDeleteCommand có chứa một phát biểu DELETE và thiết đặt thuộc tính DeleteCommand của mySqlDataAdapter tới myDeleteCommand:

```
SqlCommand myDeleteCommand = mySqlConnection.CreateCommand();
myDeleteCommand.CommandText =
    "DELETE FROM Customers " +
    "WHERE CustomerID = @OldCustomerID " +
    "AND CompanyName = @OldCompanyName " +
    "AND Address = @OldAddress";
myDeleteCommand.Parameters.Add("@OldCustomerID", SqlDbType.NChar,
    5, "CustomerID");
myDeleteCommand.Parameters.Add("@OldCompanyName", SqlDbType.NVarChar,
    40, "CompanyName");
myDeleteCommand.Parameters.Add("@OldAddress", SqlDbType.NVarChar,
    60, "Address");
myDeleteCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
myDeleteCommand.Parameters["@OldCompanyName"].SourceVersion =
    DataRowVersion.Original;
myDeleteCommand.Parameters["@OldAddress"].SourceVersion =
    DataRowVersion.Original;
mySqlDataAdapter.DeleteCommand = myDeleteCommand;
```

Chú ý rằng phát biểu DELETE cũng sử dụng sự tương tranh lạc quan.

điều này hoàn thành sự cài đặt đối tượng DataAdapter.

## **THÊM MỘT DATAROW VÀO MỘT DATATABLE:**

Trong mục này, bạn sẽ học cách thêm một DataRow vào một DataTable như thế nào. Trước khi bạn học điều này, chúng ta hãy cư trú một Dataset với những hàng từ bảng Customers. Mã sau đây tạo ra một đối tượng Dataset có tên myDataSet và cư trú nó bởi gọi phương thức mySqlDataAdapter.Fill():

```
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numOfRows =
    mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();
```

Giá trị Int trả lại bởi phương thức Fill() là số hàng được truy xuất từ cơ sở dữ liệu. Đối tượng myDataSet bây giờ chứa một DataTable có tên Customers, nó chứa những hàng được truy xuất bởi phát biểu SELECT mà đã được thiết đặt trước đó trong thuộc tính SelectCommand của mySqlDataAdapter :

```
SELECT CustomerID, CompanyName, Address
FROM Customers
ORDER BY CustomerID
```

Để thêm một hàng mới vào một đối tượng DataTable, bạn sử dụng những bước sau đây:

1. Sử dụng phương thức `NewRow()` của `DataTable` của bạn để tạo ra một `DataRow` mới.
2. Gán giá trị cho những đối tượng `DataColumn` của `DataRow` mới của bạn. Ghi chú: bạn có thể gán một giá trị `DataColumn` tới null sử dụng phương thức `SetNull()` của một `DataRow`. Bạn cũng có thể kiểm tra liệu một `DataColumn` có chứa giá trị null không sử dụng phương thức `IsNull()` của một `DataRow`.
3. Sử dụng phương thức `Add()` thông qua thuộc tính `rows` của `DataTable` của bạn để thêm `DataRow` mới vào `DataTable` của bạn.
4. Sử dụng phương thức `Update()` của `DataAdapter` của bạn để đẩy hàng mới tới cơ sở dữ liệu.

Phương pháp sau đây, có tên `AddDataRow()`, Sử dụng những bước này để thêm một hàng mới vào một `DataTable`:

```
public static void AddDataRow(  
    DataTable myDataTable,  
    SqlDataAdapter mySqlDataAdapter,  
    SqlConnection mySqlConnection  
)  
{  
    Console.WriteLine("\nIn AddDataRow()");  
  
    // step 1: use the NewRow() method of the DataTable to  
    // create a new DataRow  
    Console.WriteLine("Calling myDataTable.NewRow()");  
    DataRow myNewDataRow = myDataTable.NewRow();  
    Console.WriteLine("myNewDataRow.RowState = " +  
        myNewDataRow.RowState);  
  
    // step 2: set the values for the DataColumn objects of  
    // the new DataRow  
    myNewDataRow["CustomerID"] = "J5COM";  
    myNewDataRow["CompanyName"] = "J5 Company";  
    myNewDataRow["Address"] = "1 Main Street";  
  
    // step 3: use the Add() method through the Rows property  
    // to add the new DataRow to the DataTable  
    Console.WriteLine("Calling myDataTable.Rows.Add()");  
    myDataTable.Rows.Add(myNewDataRow);  
    Console.WriteLine("myNewDataRow.RowState = " +  
        myNewDataRow.RowState);  
  
    // step 4: use the Update() method to push the new  
    // row to the database  
    Console.WriteLine("Calling mySqlDataAdapter.Update()");  
    mySqlConnection.Open();  
    int numOfRows = mySqlDataAdapter.Update(myDataTable);  
    mySqlConnection.Close();  
    Console.WriteLine("numOfRows = " + numOfRows);  
    Console.WriteLine("myNewDataRow.RowState = " +  
        myNewDataRow.RowState);  
  
    DisplayDataRow(myNewDataRow, myDataTable);  
}
```

Bạn chú ý, tôi gọi phương thức `Open()` và `Close()` của `mySqlConnection` xung quanh phương thức `Update()`. Bạn không cần phải làm điều này vì phương thức `Update()`- cũng như phương thức `Fill()` - sẽ tự động mở và sau đó đóng `mySqlConnection` nếu như hiện thời nó đang đóng. Đó là thực hành lập trình tốt, tuy nhiên, để rõ



ràng bao gồm những lệnh gọi Open() và Close() là để bạn có thể nhìn thấy chính xác cái mà chúng ta đang tiếp tục.

**Ghi chú:** Trong ADO.NET kiểu truy nhập dữ liệu ngắt kết nối, bạn cần phải điển hình giữ kết nối tới cơ sở dữ liệu mở càng ngắn càng tốt. Tất nhiên, nếu bạn đang thực hiện nhiều lệnh gọi tới phương thức Update() hay Fill() trong một thời gian ngắn, bạn đã có thể giữ cho kết nối mở và sau đó đóng nó khi bạn kết thúc. Bằng cách này, mã của bạn sẽ thực thi tốt hơn. Bạn có lẽ cần thử nghiệm những chương trình của mình để tìm ra sự cân bằng đúng.

Phương thức Update() bị quá tải như sau:

```
int Update(DataRow[] myDataRows)
int Update(DataSet myDataSet)
int Update(DataTable myDataTable)
int Update(DataRow[] myDataRows, DataTableMapping myDataTableMapping)
int Update(DataSet myDataSet, string dataTableName)
```

với

**dataTableName:** là một chuỗi chứa tên của DataTable để cập nhật. giá trị Int được trả lại bởi phương thức Update() là số lượng hàng được cập nhật thành công trong cơ sở dữ liệu.

Quay trở lại phương thức AddDataRow() trước đây, bạn cũng nên chú ý sự bao gồm của những lệnh gọi Console.WriteLine() nó hiển thị thuộc tính RowState của myNewDataRow. Thuộc tính RowState được gán tới một trong số những hằng số định nghĩa trong lớp liệt kê System.Data.DataViewRowState. Bảng 11.10 cho thấy rằng những hằng số định nghĩa trong lớp liệt kê DataRowState.

**Bảng 11.10: những thành viên Liệt kê DataRowState**

Hằng số	Mô tả
Added	DataRow đã được thêm vào DataRowCollection của DataTable.
Deleted	DataRow đã được loại bỏ từ DataTable.
Detached	DataRow không là phần tử của DataTable.
Modified	DataRow đã được sửa đổi.
Unchanged	DataRow không được sửa đổi.

AddDataRow() Gọi một phương thức có tên DisplayDataRow(), nó hiển thị những giá trị DataColumn cho DataRow được gọi qua như tham số đầu tiên. DisplayDataRow() Được định nghĩa như sau:

```
public static void DisplayDataRow(
    DataRow myDataRow,
    DataTable myDataTable)
{
    Console.WriteLine("\nIn DisplayDataRow()");
    foreach (DataColumn myDataColumn in myDataTable.Columns)
    {
        Console.WriteLine(myDataColumn + " = " +
            myDataRow[myDataColumn]);
    }
}
```

Trong phương thức AddDataRow() trước đây, bạn thấy rằng nó trình bày nhiều điểm của thuộc tính RowState của myNewDataRow. Đầu ra từ AddDataRow() và những lệnh gọi tới DisplayDataRow() như sau:

In AddDataRow()

```

        Calling myDataTable.NewRow()
        myNewDataRow.RowState = Detached
        Calling myDataTable.Rows.Add()
        myNewDataRow.RowState = Added
        Calling mySqlDataAdapter.Update()
        numOfRows = 1
        myNewDataRow.RowState = Unchanged
        In DisplayDataRow()
        CustomerID = J5COM
        CompanyName = J5 Company
        Address = 1 Main Street

```

Chúng ta hãy khảo sát chi tiết quá trình chạy mã này :

- Sau khi myDataTable.NewRow() được gọi để tạo ra myNewDataRow, RowState của nó là Detached (tách ra), nó chỉ định myNewDataRow đã không còn là phần tử của myDataTable.
- Tiếp theo, myDataTable.Rows.Add() được gọi để thêm myNewDataRow vào myDataTable. điều này gây ra RowState của myNewDataRow thay đổi về Added, nó cho biết myNewDataRow bây giờ là phần tử của myDataTable.
- Cuối cùng, mySqlDataAdapter.Update() được gọi để đẩy hàng mới tới cơ sở dữ liệu. điều này gây ra RowState của myNewDataRow thay đổi tới Unchanged.

Đằng sau ngữ cảnh, phương thức Update() chạy phát biểu INSERT trong thuộc tính mySqlDataAdapter.InsertCommand để thêm hàng mới tới bảng Customers. giá trị Int được trả lại bởi phát biểu Update() là số lượng hàng bị ảnh hưởng bởi việc gọi phương thức. Trong ví dụ này, 1 được trả về vì một hàng đã được thêm vào.

## **SỬA ĐỔI MỘT DATAROW TRONG MỘT DATATABLE:**

Để sửa đổi một DataRow trong một DataTable, Bạn theo những bước sau đây

1. Thiết đặt thuộc tính PrimaryKey của DataTable của các bạn. Bạn cần thiết đặt nó để tìm kiếm DataRow trong bước tiếp theo.
2. Sử dụng phương thức Find() để định vị DataRow mà bạn muốn sửa đổi trong DataTable của bạn. Bạn định vị DataRow sử dụng giá trị cột khóa chính của nó.
3. Thay đổi những giá trị DataColumn cho DataRow của bạn.
4. Sử dụng phương thức Update() của đối tượng DataAdapter của bạn để đẩy hàng được sửa đổi tới cơ sở dữ liệu.

Phương thức sau đây, có tên ModifyDataRow(), Sử dụng những bước này để sửa đổi hàng được thêm vào trước đó bởi phương thức AddDataRow() :

```

public static void ModifyDataRow(
    DataTable myDataTable,
    SqlDataAdapter mySqlDataAdapter,
    SqlConnection mySqlConnection)
{
    Console.WriteLine("\nIn ModifyDataRow()");

    // step 1: set the PrimaryKey property of the DataTable
    myDataTable.PrimaryKey =
        new DataColumn[] {myDataTable.Columns["CustomerID"]};

    // step 2: use the Find() method to locate the DataRow
    // in the DataTable using the primary key value
    DataRow myEditDataRow = myDataTable.Rows.Find("J5COM");

```

```

// step 3: change the DataColumn values of the DataRow
myEditDataRow["CompanyName"] = "Widgets Inc.";
myEditDataRow["Address"] = "1 Any Street";
Console.WriteLine("myEditDataRow.RowState = " +
    myEditDataRow.RowState);
Console.WriteLine("myEditDataRow[\" CustomerID\", \" +
    \"DataRowVersion.Original] = \" +
    myEditDataRow[\"CustomerID\", DataRowVersion.Original]);
Console.WriteLine("myEditDataRow[\" CompanyName\", \" +
    \"DataRowVersion.Original] = \" +
    myEditDataRow[\"CompanyName\", DataRowVersion.Original]);
Console.WriteLine("myEditDataRow[\" Address\", \" +
    \"DataRowVersion.Original] = \" +
    myEditDataRow[\"Address\", DataRowVersion.Original]);
Console.WriteLine("myEditDataRow[\" CompanyName\", \" +
    \"DataRowVersion.Current] = \" +
    myEditDataRow[\"CompanyName\", DataRowVersion.Current]);
Console.WriteLine("myEditDataRow[\" Address\", \" +
    \"DataRowVersion.Current] = \" +
    myEditDataRow[\"Address\", DataRowVersion.Current]);

// step 4: use the Update() method to push the modified
// row to the database
Console.WriteLine("Calling mySqlDataAdapter.Update()");
mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Update(myDataTable);
mySqlConnection.Close();
Console.WriteLine("numOfRows = \" + numOfRows);
Console.WriteLine("myEditDataRow.RowState = \" +
    myEditDataRow.RowState);

    DisplayDataRow(myEditDataRow, myDataTable);
}

```

Đặt khóa chính trong bước 1 không phải được thực hiện trong phương thức ModifyDataRow(). Bạn đã có thể, chẳng hạn, đặt khóa chính ngay sau khi gọi phương thức Fill() trong phương thức Main() của chương trình AddModifyAndRemoveDataRows.cs. Lý do Tôi đặt khóa chính trong ModifyDataRow() là để bạn có thể cùng nhìn thấy tất cả những bước trong phương thức này.

Chú ý trong bước 3 của phương thức này giá trị nguyên thủy của đối tượng DataColumn : CustomerID, CompanyName, Và Address được trình bày sử dụng hằng số DataRowVersion. Original. Đây là những giá trị nguyên thủy của DataColumn trước khi chúng được thay đổi. Những giá trị hiện tại của những đối tượng DataColumn : CompanyName và Address cũng được hiển thị sử dụng hằng số DataRowVersion.Current. Đây là những giá trị DataColumn sau khi chúng được thay đổi.

Đầu ra từ ModifyDataRow() và những lệnh gọi tới DisplayDataRow() như sau:

```

In ModifyDataRow()
myEditDataRow.RowState = Modified
myEditDataRow[\"CustomerID\", DataRowVersion.Original] = J5COM
myEditDataRow[\"CompanyName\", DataRowVersion.Original] = J5 Company
myEditDataRow[\"Address\", DataRowVersion.Original] = 1 Main Street
myEditDataRow[\"CompanyName\", DataRowVersion.Current] = Widgets Inc.
myEditDataRow[\"Address\", DataRowVersion.Current] = 1 Any Street
Calling mySqlDataAdapter.Update()

```

```
numOfRows = 1  
myEditDataRow.RowState = Unchanged
```

```
In DisplayDataRow()  
CustomerID = J5COM  
CompanyName = Widgets Inc.  
Address = 1 Any Street
```

Chú ý : Đối tượng DataColumn -CompanyName và Address của myEditDataRow được thay đổi. thuộc tính RowState của myEditDataRow thay đổi tới Modified sau khi CompanyName và Address của nó được thay đổi, và sau đó tới Unchanged mySqlDataAdapter.Update() được gọi .

## **ĐÁNH DẤU NHỮNG SỰ SỬA ĐỔI CỦA BẠN:**

Bạn có thể sử dụng phương thức BeginEdit() để đánh dấu sự bắt đầu của một sự sửa đổi tới một DataRow. Chẳng hạn:

```
myEditDataRow.BeginEdit();  
myEditDataRow["CompanyName"] = "Widgets Inc.";  
myEditDataRow["Address"] = "1 Any Street";
```

Rồi bạn sử dụng cả hai phương thức EndEdit() hay CancelEdit() để đánh dấu kết thúc sự sửa đổi tới DataRow. EndEdit() giao phó sự sửa đổi; CancelEdit() loại bỏ sự sửa đổi và khôi phục DataRow tới phát biểu nguyên bản của nó trước trước lúc soạn thảo bắt đầu.

Ví dụ sau đây gọi phương thức EndEdit() của myEditDataRow để giao phó những thay đổi đã thực hiện trong ví dụ trước.

```
myEditDataRow.EndEdit();
```

## **XÓA MỘT DATAROW TỪ MỘT DATATABLE:**

Để loại bỏ một DataRow từ một DataTable, bạn sử dụng những bước sau đây:

1. Đặt thuộc tính PrimaryKey cho đối tượng DataTable của bạn.
2. Sử dụng phương thức Find() để định vị DataRow của bạn.
3. Sử dụng phương thức Delete() để loại bỏ DataRow của bạn.
4. Sử dụng phương thức Update() để đẩy lệnh xóa tới cơ sở dữ liệu.

Phương thức sau đây, có tên RemoveDataRow(), sử dụng những bước này để loại bỏ DataRow mà trước đó được sửa đổi bởi phương thức ModifyDataRow() :

```
public static void RemoveDataRow(  
    DataTable myDataTable,  
    SqlDataAdapter mySqlDataAdapter,  
    SqlConnection mySqlConnection)  
{  
    Console.WriteLine("\nIn RemoveDataRow()");  
  
    // step 1: set the PrimaryKey property of the DataTable  
    myDataTable.PrimaryKey =  
        new DataColumn[] {myDataTable.Columns["CustomerID"]};  
  
    // step 2: use the Find() method to locate the DataRow  
    DataRow myRemoveDataRow = myDataTable.Rows.Find("J5COM");
```

```
// step 3: use the Delete() method to remove the DataRow
Console.WriteLine("Calling myRemoveDataRow.Delete()");
myRemoveDataRow.Delete();
Console.WriteLine("myRemoveDataRow.RowState = " +
    myRemoveDataRow.RowState);

// step 4: use the Update() method to remove the deleted
// row from the database
Console.WriteLine("Calling mySqlDataAdapter.Update()");
mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Update(myDataTable);
mySqlConnection.Close();
Console.WriteLine("numOfRows = " + numOfRows);
Console.WriteLine("myRemoveDataRow.RowState = " +
    myRemoveDataRow.RowState);
}
```

### **Đầu ra từ RemoveDataRow() như sau:**

```
In RemoveDataRow()
Calling myRemoveDataRow.Delete()
myRemoveDataRow.RowState = Deleted
Calling mySqlDataAdapter.Update()
numOfRows = 1
myRemoveDataRow.RowState = Detached
```

### **Chú ý:**

Thuộc tính RowState của myRemoveDataRow được đặt tới Deleted sau khi myRemoveData.Delete() được gọi, và sau đó tới Detached sau khi mySqlDataAdapter.Update() được gọi - có nghĩa là myRemoveDataRow không còn là phần tử của DataTable.

Ghi nhớ : Bạn sẽ tìm thấy một chương trình đầy đủ có tên AddModifyAndRemoveDataRows.cs trong thư mục ch11 nó minh họa cách sử dụng của những phương thức AddDataRow(), ModifyDataRow(), và RemoveDataRow(). Danh sách chương trình này được bỏ qua trong sách này cho ngắn gọn.

## **TRUY XUẤT NHỮNG GIÁ TRỊ CỘT NHẬN DẠNG (KHÓA CHÍNH):**

Cột ProductID của bảng Products là một cột "căn cước"(cột khóa chính). Trong mục này bạn sẽ thấy cách chèn một hàng mới vào bảng Products và truy xuất giá trị mới được phát sinh bởi cơ sở dữ liệu cho cột khóa chính ProductID như thế nào.

**Ghi nhớ:** Bạn sẽ tìm thấy một chương trình đầy đủ có tên UsingIdentityColumn.cs trong thư mục ch11 nó minh họa sự sử dụng của những phương thức trình bày trong mục này. Danh sách chương trình này được bỏ qua trong sách này để cho ngắn gọn.

Trong những ví dụ, giả thiết bạn có một DataTable có tên productsDataTable mà được lưu trữ với những hàng được truy xuất bởi phát biểu SELECT sau đây.

```
SELECT
ProductID, ProductName, UnitPrice
FROM Products
ORDER BY ProductID
```

Ví dụ sau đây thiết đặt thuộc tính PrimaryKey của productsDataTable:

```
productsDataTable.PrimaryKey =
    new DataColumn[]
    {
        productsDataTable.Columns["ProductID"]
    };
```

Những ví dụ kế tiếp thiết đặt thuộc tính AutoIncrement, AutoIncrementSeed, AutoIncrementStep, ReadOnly, và Unique cho DataColumn ProductID của productsDataTable:

```
DataColumn productIDDataColumn =
    productsDataTable.Columns["ProductID"];
productIDDataColumn.AllowDBNull = false;
productIDDataColumn.AutoIncrement = true;
productIDDataColumn.AutoIncrementSeed = -1;
productIDDataColumn.AutoIncrementStep = -1;
productIDDataColumn.ReadOnly = true;
productIDDataColumn.Unique = true;
```

Vì những sự thiết đặt này, khi bạn thêm một DataRow mới vào productsDataTable, ProductID DataColumn của DataRow mới của bạn thoát tiên sẽ có giá trị -1.

Như trong mục trước , " Sửa đổi những hàng trong một DataTable, " Bạn cần đặt thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của đối tượng DataAdapter của bạn những đối tượng lệnh thích hợp. Thuộc tính CommandText của đối tượng Command được dùng trong thuộc tính UpdateCommand như sau:

```
myUpdateCommand.CommandText =
    "UPDATE Products " +
    "SET " +
    " ProductName = @NewProductName, " +
    " UnitPrice = @NewUnitPrice " +
    "WHERE ProductID = @OldProductID " +
    "AND ProductName = @OldProductName " +
    "AND UnitPrice = @OldUnitPrice";
```

Thuộc tính CommandText của đối tượng Command được dùng trong thuộc tính DeleteCommand như sau:

```
myDeleteCommand.CommandText =
    "DELETE FROM Products " +
    "WHERE ProductID = @OldProductID " +
    "AND ProductName = @OldProductName " +
    "AND UnitPrice = @OldUnitPrice";
```

Chú ý: CommandText của hai đối tượng Command này không khác biệt đáng kể với những gì được trình bày trong mục trước , chỉ có điều nó đi ngược lại bảng Products hơn là bảng Customers.

Sự khác nhau thực sự là trong CommandText của đối tượng Command sử dụng trong thuộc tính InsertCommand - phải truy xuất giá trị ProductID phát sinh bởi cơ sở dữ liệu cho hàng mới. để thực hiện điều này, bạn có thể sử dụng mã sau đây có chứa một phát biểu INSERT để thêm một hàng mới, cùng với một phát biểu SELECT để truy xuất giá trị ProductID sử dụng một lệnh gọi tới hàm SQL Server:SCOPE\_IDENTITY() :

```
myInsertCommand.CommandText =
    "INSERT INTO Products (" +
    " ProductName, UnitPrice " +
    ") VALUES (" +
    " @MyProductName, @MyUnitPrice" +
    ");" +
```

```

"SELECT @MyProductID = SCOPE_IDENTITY()";
myInsertCommand.Parameters.Add(
"@MyProductName", SqlDbType.NVarChar, 40, "ProductName");
myInsertCommand.Parameters.Add(
"@MyUnitPrice", SqlDbType.Money, 0, "UnitPrice");
myInsertCommand.Parameters.Add("@MyProductID", SqlDbType.Int,
0, "ProductID");
myInsertCommand.Parameters["@MyProductID"].Direction =
ParameterDirection.Output;

```

SCOPE\_IDENTITY() hàm trả lại giá trị nhận dạng (khóa chính) được thêm sau cùng vào trong bất kỳ bảng nào được thực hiện bên trong phiên cơ sở dữ liệu hiện thời và store procedure, trigger, function, hay batch. Chẳng hạn, việc gọi SCOPE\_IDENTITY() trong ví dụ trước trả lại giá trị nhận dạng (khóa chính) được thêm sau cùng vào trong bảng Products, chính là ProductID của hàng mới.

Ghi chú: cho những chi tiết về hàm SCOPE\_IDENTITY() , tham chiếu tới Chương 4, " giới thiệu về lập trình Transact-SQL ."

Khi bạn thêm một DataRow mới vào productsDataTable, DataColumn ProductID của DataRow mới của bạn thoát tiên sẽ có giá trị -1. Khi bạn gọi phương thức Update() của SqlDataAdapter của bạn để đẩy hàng mới tới cơ sở dữ liệu, những bước sau đây sẽ xuất hiện:

1. DataRow mới của bạn được đẩy tới cơ sở dữ liệu sử dụng phát biểu INSERT được thiết đặt trong myInsertCommand, với cột ProductID của bảng Products đang được gán tới một giá trị "căn cước" mới phát sinh bởi cơ sở dữ liệu.
2. Giá trị căn cước ProductID được truy xuất bởi phát biểu SELECT gán trong myInsertCommand.
3. Cột dữ liệu ProductID trong DataRow của bạn được gán tới giá trị "căn cước" được truy xuất.

Cứ tự nhiên khảo sát, biên tập, và chạy chương trình UsingIdentityColumn.cs được lưu trữ trong thư mục ch11. Chương trình này thực hiện những hoạt động cấp cao sau đây:

1. Truy xuất những hàng từ bảng Products vào trong một DataTable có tên productsDataTable.
2. Thêm một DataRow vào productsDataTable.
3. Sửa đổi DataRow mới.
4. Xóa DataRow mới.

Khi bạn chạy chương trình này bạn sẽ chú ý sự thay đổi của giá trị cột dữ liệu ProductID cho một DataRow mới thêm vào từ -1 đến một giá trị thực truy xuất từ cơ sở dữ liệu.

## **SỬ DỤNG NHỮNG THỦ TỤC PROCEDURE ĐỂ THÊM, SỬA, XÓA NHỮNG HÀNG TỪ CƠ SỞ DỮ LIỆU**

Bạn có thể dùng một đối tượng DataAdapter để gọi những thủ tục lưu trữ để thêm, sửa đổi, và loại bỏ những hàng từ cơ sở dữ liệu. Những thủ tục này được gọi thay cho những phát biểu INSERT, UPDATE, và DELETE mà bạn đã được thấy cách để thiết đặt các thuộc tính InsertCommand, UpdateCommand, và DeleteCommand cho một đối tượng DataAdapter .

Khả năng để gọi những thủ tục lưu trữ sử dụng một DataAdapter là một bổ sung rất mạnh tới ADO.NET. ví dụ bạn có thể sử dụng một thủ tục lưu trữ để thêm một hàng vào một bảng có chứa một cột khóa chính, và truy xuất giá trị mới cho cột này được phát sinh bởi cơ sở dữ liệu. Bạn cũng có thể thực hiện việc bổ sung trong một thủ tục lưu trữ như chèn một hàng vào trong một bảng kiểm toán khi một hàng được sửa đổi. Bạn sẽ thấy những ví dụ của cả hai kịch bản này trong mục này.

**Mẹo nhỏ :** sử dụng những thủ tục lưu trữ thay vì những phát biểu Chèn, sự Cập nhật, và Xóa có thể cũng cải thiện sự thực thi. Bạn cần phải sử dụng những thủ tục lưu trữ nếu cơ sở dữ liệu của bạn hỗ trợ chúng. SQL Server và Oracle hỗ trợ những thủ tục lưu trữ. những thủ tục lưu trữ Oracle được viết trong



PL/ SQL.

Cột ProductID của bảng Products là một cột khóa chính, và bạn thấy một số thủ tục lưu trữ trong Chương 4, " giới thiệu về lập trình giao dịch- SQL " mà đã thêm một hàng vào những bảng sản phẩm và trả lại ProductID.

Trong mục này, bạn sẽ thấy cách để

- Tạo ra những thủ tục lưu trữ được yêu cầu trong cơ sở dữ liệu Northwind.
- thiết lập một DataAdapter để gọi những thủ tục lưu trữ.
- Thêm, sửa đổi, và loại bỏ một DataRow từ một DataTable.

Những phương thức C# được trình bày trong mục này đi theo những bước tương tự như trình bày trong mục trước , " Sửa đổi những hàng ở một DataTable."

**Ghi nhớ:** Bạn sẽ tìm thấy một chương trình đầy đủ có tên PushChangesUsingProcedures.cs trong thư mục ch11 minh họa sự sử dụng của những phương thức trình bày trong mục này. Danh sách cho chương trình này được bỏ qua trong sách này cho ngắn gọn.

## **TAO STORED PROCEDURES TRONG CƠ SỞ DỮ LIỆU:**

Bạn sẽ tạo ra ba thủ tục lưu trữ trong cơ sở dữ liệu Northwind như sau:

- AddProduct4(), mà thêm một hàng vào những bảng sản phẩm.
- UpdateProduct(), nó cập nhật một hàng trong bảng Products.
- DeleteProduct(), để xóa một hàng từ những bảng sản phẩm.

Chúng ta hãy xem xét những thủ tục này.

## **THỦ TỤC LƯU TRỮ AddProduct4()**

AddProduct4() thêm một hàng vào bảng Products. Nó sử dụng số 4 vì những chương trước đã sử dụng những thủ tục có tên AddProduct(), AddProduct2(), Và AddProduct3().

Danh sách 11.4 trình bày file AddProduct4.sql mà bạn sử dụng để tạo ra thủ tục AddProduct4() . Hãy tham chiếu tới Chương 4 nếu bạn cần một nước giải khát trên Ngôn ngữ Giao dịch- SQL (Transact-SQL language) hay nếu bạn cần tìm hiểu cách chạy script này để tạo ra thủ tục trong cơ sở dữ liệu.

### **Danh sách 11.4: ADDPRODUCT4. SQL**

```
/*
    AddProduct4.sql creates a procedure that adds a row to the
    Products table using values passed as parameters to the
    procedure. The procedure returns the ProductID of the new row
    using a RETURN statement
*/

CREATE PROCEDURE AddProduct4
    @MyProductName nvarchar(40),
    @MyUnitPrice money
AS

    -- declare the @MyProductID variable
    DECLARE @MyProductID int

    -- insert a row into the Products table
    INSERT INTO Products (
```

```

    ProductName, UnitPrice
) VALUES (
    @MyProductName, @MyUnitPrice
)

```

```

-- use the SCOPE_IDENTITY() function to get the last
-- identity value inserted into a table performed within
-- the current database session and stored procedure,
-- so SCOPE_IDENTITY returns the ProductID for the new row
-- in the Products table in this case
SET @MyProductID = SCOPE_IDENTITY()

```

```

RETURN @MyProductID

```

**Ghi chú:** Bạn sẽ tìm thấy AddProduct4.sql trong thư mục ch11.

## **THỦ TỤC UpdateProduct()**

```

/*
    UpdateProduct.sql creates a procedure that modifies a row
    in the Products table using values passed as parameters
    to the procedure
*/

```

```

CREATE PROCEDURE UpdateProduct
    @OldProductID int,
    @NewProductName nvarchar(40),
    @NewUnitPrice money,
    @OldProductName nvarchar(40),
    @OldUnitPrice money
AS

-- update the row in the Products table
UPDATE Products
SET
    ProductName = @NewProductName,
    UnitPrice = @NewUnitPrice
WHERE ProductID = @OldProductID
AND ProductName = @OldProductName
AND UnitPrice = @OldUnitPrice

```

Vì mệnh đề WHERE chứa những giá trị cột cũ trong phát biểu Update của thủ tục này, UPDATE sử dụng sự tương tranh lạc quan được mô tả trước đó. Có nghĩa là một người sử dụng không ghi đè lên những thay đổi của người sử dụng khác.

## **THỦ TỤC DeletProduct()**

DeleteProduct() xóa một hàng từ bảng Products. Danh sách 11.6 trình bày tệp tin DeleteProduct.sql mà bạn sử dụng để tạo ra thủ tục DeleteProduct() .

### **Danh sách 11.6: DELETEDPRODUCT.SQL**

```

/*
    DeleteProduct.sql creates a procedure that removes a row
    from the Products table

```

\*/

```
CREATE PROCEDURE DeleteProduct
    @OldProductID int,
    @OldProductName nvarchar(40),
    @OldUnitPrice money
AS

-- delete the row from the Products table
DELETE FROM Products
WHERE ProductID = @OldProductID
AND ProductName = @OldProductName
AND UnitPrice = @OldUnitPrice
```

## **SỬ DỤNG SET NOCOUNT ON TRONG STORED PROCEDURES**

Trong Chương 4, " giới thiệu về Lập trình Giao dịch- SQL " bạn đã thấy bạn sử dụng command NET NOCOUNT ON để ngăn ngừa Transact- SQL về việc trả lại số lượng hàng bị ảnh hưởng. Điển hình, bạn phải tránh sử dụng lệnh này trong thủ tục lưu trữ của bạn bởi vì DataAdapter sử dụng số lượng hàng được ảnh hưởng được trả lại để biết liệu có phải sự cập nhật thành công.

Có một tình huống khi bạn phải sử dụng SET NOCOUNT ON : Khi thủ tục lưu trữ của bạn thực hiện một phát biểu Chèn, sự Cập nhật hay Xóa sự mà ảnh hưởng đến bảng khác không kể một chính bạn đang đẩy một sự thay đổi Tới. Chẳng hạn, nói DeleteProduct() Thủ tục cũng thực hiện một sự phát biểu Chèn để thêm một hàng vào bảng ProductAudit (được mô tả trong Chương 4) ghi sự nỗ lực để xóa hàng từ những bảng sản phẩm. Trong ví dụ này, bạn phải sử dụng Tập hợp NOCOUNT Trên việc trước đây thực hiện sự Chèn vào trong bảng ProductAudit, như được Đưa vào liệt kê 11.7.

### **Danh sách 11.7: DELETEPRODUCT2. SQL**

```
/*
DeleteProduct2.sql creates a procedure that removes a row
from the Products table
*/

CREATE PROCEDURE DeleteProduct2
    @OldProductID int,
    @OldProductName nvarchar(40),
    @OldUnitPrice money
AS
-- delete the row from the Products table
DELETE FROM Products
WHERE ProductID = @OldProductID
AND ProductName = @OldProductName
AND UnitPrice = @OldUnitPrice

-- use SET NOCOUNT ON to suppress the return of the
-- number of rows affected by the INSERT statement
SET NOCOUNT ON

-- add a row to the Audit table
IF @@ROWCOUNT = 1
    INSERT INTO ProductAudit (
        Action
    ) VALUES (
        'Product deleted with ProductID of ' +
```

```

        CONVERT(nvarchar, @OldProductID)
    )
ELSE
    INSERT INTO ProductAudit (
        Action
    ) VALUES (
        'Product with ProductID of ' +
        CONVERT(nvarchar, @OldProductID) +
        ' was not deleted'
    )

```

Do việc sử dụng SET NOCOUNT ON trước lệnh INSERT, chỉ có số lượng hàng ảnh hưởng bởi phát biểu DELETE được trả về, và do đó DataAdapter lấy được giá trị đúng.

Transact- SQL cũng có một Command SET NOCOUNT ON để trả về số lượng hàng bị ảnh hưởng. Bạn đã có thể sử dụng một sự kết hợp của SET NOCOUNT OFF và SET NOCOUNT ON nếu như Bạn cần thực hiện một phát biểu Chèn, Cập nhật, hay Xóa trước phát biểu SQL chính trong thủ tục lưu trữ của các bạn.

### **THIẾT LẬP MỘT *DataAdapter* ĐỂ GỌI NHỮNG THỦ TỤC LƯU TRỮ:**

Như đã được đề cập trong mục trước - " Sửa đổi những hàng trong một DataTable, " Bạn cần tạo ra một đối tượng DataAdapter và gán những thuộc tính SelectCommand, InsertCommand, UpdateCommand, và DeleteCommand của nó với những đối tượng Command thích hợp . Tuy nhiên, lúc này, những thuộc tính InsertCommand, UpdateCommand, và DeleteCommand sẽ chứa những đối tượng Command gọi những thủ tục lưu trữ được trình bày trước đó.

Đầu tiên, ví dụ sau đây tạo ra một đối tượng SqlCommand chứa một phát biểu SELECT và thiết đặt những thuộc tính SelectCommand của một SqlDataAdapter tới SqlCommand này:

```

SqlCommand mySelectCommand = mySqlConnection.CreateCommand();
mySelectCommand.CommandText =
    "SELECT " +
    " ProductID, ProductName, UnitPrice " +
    "FROM Products " +
    "ORDER BY ProductID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySelectCommand;

```

Phát biểu SELECT được chạy khi bạn gọi phương thức Fill() của đối tượng mySqlDataAdapter để truy xuất những hàng từ bảng Product vào trong một Dataset.

Trước khi bạn có thể đẩy những thay đổi tới cơ sở dữ liệu, bạn phải gán những thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của DataAdapter của bạn với những đối tượng Command. Những đối tượng Command này sẽ chứa những lệnh gọi tới những thủ tục lưu trữ AddProduct4(), UpdateProduct(), Và DeleteProduct() mà bạn tạo ra trước đó. Rồi bạn thêm, điều chỉnh, hay loại bỏ những đối tượng DataRow từ Dataset của bạn, và sau đó gọi phương thức Update() của DataAdapter của bạn , Thủ tục lưu trữ thích hợp được chạy đẩy những sự thay đổi của bạn tới cơ sở dữ liệu.

Chúng ta hãy xem xét cách thiết đặt những thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của DataAdapter của bạn.

### **THIẾT ĐẶT THUỘC TÍNH *InsertCommand* CỦA *DataAdapter*:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myInsertCommand chứa một lệnh gọi tới thủ tục lưu trữ AddProduct4() :

```

SqlCommand myInsertCommand = mySqlConnection.CreateCommand();
myInsertCommand.CommandText =
    "EXECUTE @MyProductID = AddProduct4 @MyProductName, @MyUnitPrice";
myInsertCommand.Parameters.Add(
    "@MyProductID", SqlDbType.Int, 0, "ProductID");
myInsertCommand.Parameters["@MyProductID"].Direction =
    ParameterDirection.Output;
myInsertCommand.Parameters.Add(
    "@MyProductName", SqlDbType.NVarChar, 40, "ProductName");
myInsertCommand.Parameters.Add(
    "@MyUnitPrice", SqlDbType.Money, 0, "UnitPrice");

```

Như bạn có thể thấy từ mã trước đây, hướng của tham số @MyProductID được gán là ParameterDirection.Output, nó chỉ định tham số này là một tham số đầu ra. Đồng thời, chiều dài cực đại của những tham số @MyProductID và @MyUnitPrice được gán là 0 trong tham số thứ ba của phương thức Add(). Việc gán chúng tới 0 sẽ tốt hơn bởi vì chiều dài cực đại không ứng dụng với những kiểu độ dài cố định như những con số, mà chỉ ứng dụng với những kiểu như chuỗi.

Tiếp theo, ví dụ sau đây đặt thuộc tính InsertCommand của mySqlDataAdapter Tới myInsertCommand:

```
mySqlDataAdapter.InsertCommand = myInsertCommand;
```

### **THIẾT ĐẶT THUỘC TÍNH UpdateCommand CỦA MỘT DataAdapter:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myUpdateCommand chứa một lệnh gọi tới thủ tục lưu trữ UpdateProduct() và gán thuộc tính UpdateCommand của mySqlDataAdapter là myUpdateCommand:

```

SqlCommand myUpdateCommand = mySqlConnection.CreateCommand();
myUpdateCommand.CommandText =
    "EXECUTE UpdateProduct @OldProductID, @NewProductName, " +
    "@NewUnitPrice, @OldProductName, @OldUnitPrice";
myUpdateCommand.Parameters.Add(
    "@OldProductID", SqlDbType.Int, 0, "ProductID");
myUpdateCommand.Parameters.Add(
    "@NewProductName", SqlDbType.NVarChar, 40, "ProductName");
myUpdateCommand.Parameters.Add(
    "@NewUnitPrice", SqlDbType.Money, 0, "UnitPrice");
myUpdateCommand.Parameters.Add(
    "@OldProductName", SqlDbType.NVarChar, 40, "ProductName");
myUpdateCommand.Parameters.Add(
    "@OldUnitPrice", SqlDbType.Money, 0, "UnitPrice");
myUpdateCommand.Parameters["@OldProductID"].SourceVersion =
    DataRowVersion.Original;
myUpdateCommand.Parameters["@OldProductName"].SourceVersion =
    DataRowVersion.Original;
myUpdateCommand.Parameters["@OldUnitPrice"].SourceVersion =
    DataRowVersion.Original;
mySqlDataAdapter.UpdateCommand = myUpdateCommand;

```

### **THIẾT ĐẶT THUỘC TÍNH DeleteCommand CỦA MỘT DataAdapter:**

Ví dụ sau đây tạo ra một đối tượng SqlCommand có tên myDeleteCommand chứa một lệnh gọi tới thủ tục lưu trữ DeleteProduct() và gán thuộc tính DeleteCommand của mySqlDataAdapter là myDeleteCommand:

```

SqlCommand myDeleteCommand = mySqlConnection.CreateCommand();
myDeleteCommand.CommandText =

```

```

"EXECUTE DeleteProduct @OldProductID, @OldProductName, @OldUnitPrice";
myDeleteCommand.Parameters.Add(
    "@OldProductID", SqlDbType.Int, 0, "ProductID");
myDeleteCommand.Parameters.Add(
    "@OldProductName", SqlDbType.NVarChar, 40, "ProductName");
myDeleteCommand.Parameters.Add(
    "@OldUnitPrice", SqlDbType.Money, 0, "UnitPrice");
myDeleteCommand.Parameters["@OldProductID"].SourceVersion =
    DataRowVersion.Original;
myDeleteCommand.Parameters["@OldProductName"].SourceVersion =
    DataRowVersion.Original;
myDeleteCommand.Parameters["@OldUnitPrice"].SourceVersion =
    DataRowVersion.Original;
mySqlDataAdapter.DeleteCommand = myDeleteCommand;

```

Đoạn mã này hoàn thành việc cài đặt của đối tượng DataAdapter.

### **THÊM MỘT DataRow VÀO MỘT DataTable:**

Trong mục này, bạn sẽ học cách để thêm một DataRow vào một DataTable . Đầu tiên, mã sau đây tạo ra một đối tượng Dataset có tên myDataSet và cư trú nó bởi việc gọi mySqlDataAdapter. Fill():

```

DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numOfRows =
    mySqlDataAdapter.Fill(myDataSet, "Products");
mySqlConnection.Close();

```

Giá trị Int trả về bởi phương thức Fill() là số lượng hàng được truy xuất từ cơ sở dữ liệu và sao chép tới myDataSet. Đối tượng myDataSet bây giờ chứa một DataTable có tên Products, nó chứa những hàng được truy xuất bởi phát biểu SELECT đã thiết lập trước đó trong thuộc tính SelectCommand của mySqlDataAdapter :

```

SELECT ProductID, ProductName, UnitPrice
FROM Products
ORDER BY ProductID

```

Để thêm một hàng mới vào một đối tượng DataTable , bạn sử dụng bốn bước tương tự như trình bày trước đó trong mục " sửa đổi một DataRow ở một DataTable." Phương thức sau đây, có tên AddDataRow(), sử dụng những bước này để thêm một hàng mới vào một DataTable:

```

public static int AddDataRow(
    DataTable myDataTable,
    SqlDataAdapter mySqlDataAdapter,
    SqlConnection mySqlConnection
)
{
    Console.WriteLine("\nIn AddDataRow()");

    // step 1: use the NewRow() method of the DataTable to
    // create a new DataRow
    Console.WriteLine("Calling myDataTable.NewRow()");
    DataRow myNewDataRow = myDataTable.NewRow();
    Console.WriteLine("myNewDataRow.RowState = " +
        myNewDataRow.RowState);

    // step 2: set the values for the DataColumn objects of

```

```

// the new DataRow
myNewDataRow["ProductName"] = "Widget";
myNewDataRow["UnitPrice"] = 10.99;

// step 3: use the Add() method through the Rows property
// to add the new DataRow to the DataTable
Console.WriteLine("Calling myDataTable.Rows.Add()");
myDataTable.Rows.Add(myNewDataRow);
Console.WriteLine("myNewDataRow.RowState = " +
    myNewDataRow.RowState);
// step 4: use the Update() method to push the new
// row to the database
Console.WriteLine("Calling mySqlDataAdapter.Update()");
mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Update(myDataTable);
mySqlConnection.Close();
Console.WriteLine("numOfRows = " + numOfRows);
Console.WriteLine("myNewDataRow.RowState = " +
    myNewDataRow.RowState);

DisplayDataRow(myNewDataRow, myDataTable);

// return the ProductID of the new DataRow
return (int) myNewDataRow["ProductID"];
}

```

Chú ý không có giá trị nào cho cột dữ liệu ProductID được thiết đặt trong bước 2. Bởi vì ProductID tự động được phát sinh bởi cơ sở dữ liệu khi hàng mới được đẩy tới cơ sở dữ liệu bởi phương thức Update() trong bước 4.

Khi phương thức Update() được gọi, thủ tục lưu trữ AddProduct4() được chạy để thêm hàng mới vào bảng Products. Rồi cơ sở dữ liệu phát sinh một ProductID mới cho hàng, giá trị này được trả về bởi thủ tục lưu trữ AddProduct4(). Và bạn có thể đọc ProductID mới bởi sử dụng myNewDataRow["ProductID"], mà bây giờ chứa ProductID mới. ProductID này được trả về vào thời điểm cuối cùng của thực thi phương thức AddDataRow().

Đầu ra từ AddDataRow() và lệnh gọi của nó tới DisplayDataRow() như sau:

```

In AddDataRow()
Calling myDataTable.NewRow()
myNewDataRow.RowState = Detached
Calling myDataTable.Rows.Add()
myNewDataRow.RowState = Added
Calling mySqlDataAdapter.Update()
numOfRows = 1
myNewDataRow.RowState = Unchanged

In DisplayDataRow()
ProductID = 180
ProductName = Widget
UnitPrice = 10.99

```

Như bạn có thể thấy, sau khi myDataTable.NewRow() được gọi để tạo ra myNewDataRow, RowState của nó là Detached, cho biết myNewDataRow không còn là phần tử của myDataTable.

Tiếp theo, myDataTable.Row.Add() được gọi để thêm myNewDataRow vào myDataTable. Điều này gây ra



RowState của myDataTable thay đổi thành Added, cho biết myNewDataRow đã được thêm vào myDataTable. Cuối cùng, mySqlDataAdapter.Update() được gọi để đẩy hàng mới tới cơ sở dữ liệu. Thủ tục lưu trữ AddProduct4() được chạy để thêm hàng mới vào bảng Products, và RowState của myNewDataRow thay đổi thành Unchanged.

### **THAY ĐỔI MỘT DataRow TRONG MỘT DataTable:**

Phương thức sau đây, có tên ModifyDataRow(), sử dụng bốn bước để sửa đổi một DataRow trong một đối tượng DataTable. Chú ý rằng ProductID để chỉnh sửa được gọi qua như một tham số.

```
public static void ModifyDataRow(
    DataTable myDataTable,
    int productID,
    SqlDataAdapter mySqlDataAdapter,
    SqlConnection mySqlConnection
)
{
    Console.WriteLine("\nIn ModifyDataRow()");

    // step 1: set the PrimaryKey property of the DataTable
    myDataTable.PrimaryKey =
        new DataColumn[]
        {
            myDataTable.Columns["ProductID"]
        };

    // step 2: use the Find() method to locate the DataRow
    // in the DataTable using the primary key value
    DataRow myEditDataRow = myDataTable.Rows.Find(productID);

    // step 3: change the DataColumn values of the DataRow
    myEditDataRow["ProductName"] = "Advanced Widget";
    myEditDataRow["UnitPrice"] = 24.99;
    Console.WriteLine("myEditDataRow.RowState = " +
        myEditDataRow.RowState);
    Console.WriteLine("myEditDataRow[\" ProductID\", \" +
        \"DataRowVersion.Original\"] = \" +
        myEditDataRow[\"ProductID\", DataRowVersion.Original]);
    Console.WriteLine("myEditDataRow[\" ProductName\", \" +
        \"DataRowVersion.Original\"] = \" +
        myEditDataRow[\"ProductName\", DataRowVersion.Original]);
    Console.WriteLine("myEditDataRow[\" UnitPrice\", \" +
        \"DataRowVersion.Original\"] = \" +
        myEditDataRow[\"UnitPrice\", DataRowVersion.Original]);
    Console.WriteLine("myEditDataRow[\" ProductName\", \" +
        \"DataRowVersion.Current\"] = \" +
        myEditDataRow[\"ProductName\", DataRowVersion.Current]);
    Console.WriteLine("myEditDataRow[\" UnitPrice\", \" +
        \"DataRowVersion.Current\"] = \" +
        myEditDataRow[\"UnitPrice\", DataRowVersion.Current]);

    // step 4: use the Update() method to push the update
    // to the database
    Console.WriteLine("Calling mySqlDataAdapter.Update()");
    mySqlConnection.Open();
    int numOfRows = mySqlDataAdapter.Update(myDataTable);
}
```

```

mySqlConnection.Close();
Console.WriteLine("numOfRows = " + numOfRows);
Console.WriteLine("myEditDataRow.RowState = " +
    myEditDataRow.RowState);

DisplayDataRow(myEditDataRow, myDataTable);
}

```

Chú ý phương thức này trình bày những giá trị ban đầu của những đối tượng DataColumn : ProductID, ProductName, và UnitPrice sử dụng DataRowVersion.Original.Constant . Đây là những giá trị DataColumn trước khi chúng được thay đổi. Phương thức cũng trình bày những giá trị hiện tại cho những đối tượng cột dữ liệu ProductName và UnitPrice sử dụng hằng số DataRowVersion.Current . Đây là những giá trị DataColumn sau khi chúng được thay đổi. Khi phương thức Update() được gọi trong bước 4, thủ tục lưu trữ UpdateProduct() được chạy phía sau ngữ cảnh để thực hiện sự cập nhật.

Đầu ra từ ModifyDataRow() và lệnh gọi của tới DisplayDataRow() như sau:

```

In ModifyDataRow()
myEditDataRow.RowState = Modified
myEditDataRow["ProductID", DataRowVersion.Original] = 180
myEditDataRow["ProductName", DataRowVersion.Original] = Widget
myEditDataRow["UnitPrice", DataRowVersion.Original] = 10.99
myEditDataRow["ProductName", DataRowVersion.Current] = Advanced Widget
myEditDataRow["UnitPrice", DataRowVersion.Current] = 24.99
Calling mySqlDataAdapter.Update()
numOfRows = 1
myEditDataRow.RowState = Unchanged

In DisplayDataRow()
ProductID = 180
ProductName = Advanced Widget
UnitPrice = 24.99

```

Chú ý rằng thuộc tính RowState của myEditDataRow thay đổi thành Modified sau khi dữ liệu của nó được thay đổi, và rồi thành Unchange sau khi mySqlDataAdapter.Update() được gọi .

### **XÓA BỎ MỘT DataRow TỪ MỘT DataTable:**

Phương thức sau đây, có tên RemoveDataRow(), sử dụng bốn bước để loại bỏ một DataRow từ một DataTable. Chú ý ProductID để điều chỉnh được gọi qua như một tham số.

```

public static void RemoveDataRow(
    DataTable myDataTable,
    int productID,
    SqlDataAdapter mySqlDataAdapter,
    SqlConnection mySqlConnection
)
{
    Console.WriteLine("\nIn RemoveDataRow()");

    // step 1: set the PrimaryKey property of the DataTable
    myDataTable.PrimaryKey =
        new DataColumn[]
        {
            myDataTable.Columns["ProductID"]
        };
};

```

```
// step 2: use the Find() method to locate the DataRow
DataRow myRemoveDataRow = myDataTable.Rows.Find(productID);

// step 3: use the Delete() method to remove the DataRow
Console.WriteLine("Calling myRemoveDataRow.Delete()");
myRemoveDataRow.Delete();
Console.WriteLine("myRemoveDataRow.RowState = " +
    myRemoveDataRow.RowState);

// step 4: use the Update() method to push the delete
// to the database
Console.WriteLine("Calling mySqlDataAdapter.Update()");
mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Update(myDataTable);
mySqlConnection.Close();
Console.WriteLine("numOfRows = " + numOfRows);
Console.WriteLine("myRemoveDataRow.RowState = " +
    myRemoveDataRow.RowState);
}
```

Đầu ra từ RemoveDataRow() như sau:

```
In RemoveDataRow()
Calling myRemoveDataRow.Delete()
myRemoveDataRow.RowState = Deleted
Calling mySqlDataAdapter.Update()
numOfRows = 1
myRemoveDataRow.RowState = Detached
```

Chú ý rằng thuộc tính RowState của myRemoveDataRow được gán là Deleted sau khi myRemoveData.Delete() được gọi, và chuyển thành Detached sau khi mySqlDataAdapter.Update() được gọi. Khi phương thức Update() được gọi trong bước 4, Thủ tục lưu trữ DeleteProduct() được chạy đằng sau ngữ cảnh để thực hiện việc xóa.

**Ghi nhớ** Bạn sẽ tìm thấy một chương trình đầy đủ có tên PushChangesUsingProcedures.cs trong thư mục ch11 minh họa sự sử dụng của những phương thức AddDataRow(), ModifyDataRow(), Và RemoveDataRow(). Danh sách này được bỏ qua trong sách này cho ngắn gọn.

### **TỰ ĐỘNG PHÁT SINH NHỮNG PHÁT BIỂU SQL:**

Như bạn đã thấy trong những mục trước đây, việc cung cấp những phát biểu INSERT, UPDATE, và DELETE hay những thủ tục lưu trữ của bạn để đẩy những sự thay đổi từ Dataset của bạn đến cơ sở dữ liệu có nghĩa bạn phải bỏ công viết nhiều mã. Bạn có thể tránh viết mã này bởi việc sử dụng một đối tượng CommandBuilder, mà có thể tự động phát sinh những lệnh (Command) INSERT, UPDATE, và DELETE bằng đơn lẻ, mà đẩy những sự thay đổi bạn thực hiện với một đối tượng Dataset đến cơ sở dữ liệu. Rồi những lệnh này được gán vào những thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của đối tượng DataAdapter của bạn.

Mặc dù bạn có tiết kiệm thời gian ghi một vài dòng mã nhờ sử dụng CommandBuilder, Bạn phải nhớ những giới hạn sau đây khi sử dụng một CommandBuilder:

- Thuộc tính SelectCommand của DataAdapter của bạn chỉ có thể truy xuất những hàng từ một bảng đơn.
- Bảng cơ sở dữ liệu được dùng trong SelectCommand của bạn phải chứa đựng một khóa chính.
- Khóa chính của bảng phải được bao gồm trong SelectCommand của các bạn.
- CommandBuilder mất một số lượng nhất định thời gian để phát sinh những lệnh bởi vì nó phải khảo sát cơ sở dữ liệu.

**Cảnh báo:** Bởi vì một CommandBuilder làm chậm sự thực thi của chương trình của bạn, bạn cần phải

tránh sử dụng chúng. chúng được dự định cho sự sử dụng bởi những người phát triển mà không quen thuộc với SQL hay thủ tục lưu trữ. Để thực hiện tốt nhất, bạn sử dụng những thủ tục lưu trữ.

Có ba lớp nhà cung cấp được quản lý CommandBuilder : SqlCommandBuilder, OleDbCommandBuilder, và OdbcCommandBuilder. Bạn sẽ thấy sự sử dụng một đối tượng SqlCommandBuilder trong mục này, nó làm việc với một cơ sở dữ liệu người phục vụ SQL. Những kiểu đối tượng khác làm việc trong cùng một cách .

Đầu tiên, bạn cần thiết đặt thuộc tính SelectCommand của một đối tượng SqlDataAdapter. Phát biểu SELECT được dùng trong lệnh (Command) này chỉ có thể truy xuất những hàng từ một bảng đơn, và trong ví dụ sau đây bảng Customers được sử dụng

```
SqlCommand mySelectCommand = mySqlConnection.CreateCommand();
mySelectCommand.CommandText =
    "SELECT CustomerID, CompanyName, Address " +
    "FROM Customers " +
    "ORDER BY CustomerID";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySelectCommand;
```

Tiếp theo, ví dụ sau đây tạo ra một đối tượng SqlCommandBuilder, thông qua mySqlDataAdapter đến Bộ khởi tạo.

```
SqlCommandBuilder mySqlCommandBuilder =
    new SqlCommandBuilder(mySqlDataAdapter);
```

SqlCommandBuilder sẽ phát sinh những lệnh (Command) chứa phát biểu Chèn, Cập nhật, và Xóa dựa vào phát biểu SELECT trước đó được gán trong thuộc tính SelectCommand của đối tượng mySqlDataAdapter.

Bạn có thể thu được được những lệnh phát sinh - sử dụng những phương thức GetInsertCommand(), GetUpdateCommand(), và GetDeleteCommand() của mySqlCommandBuilder. Chẳng hạn:

```
Console.WriteLine(
    "mySqlCommandBuilder.GetInsertCommand().CommandText =\n" +
    mySqlCommandBuilder.GetInsertCommand().CommandText
);
Console.WriteLine(
    "mySqlCommandBuilder.GetUpdateCommand().CommandText =\n" +
    mySqlCommandBuilder.GetUpdateCommand().CommandText
);
Console.WriteLine(
    "mySqlCommandBuilder.GetDeleteCommand().CommandText =\n" +
    mySqlCommandBuilder.GetDeleteCommand().CommandText
);
```

Mã này hiển thị đầu ra sau đây ( Tôi có thêm vài khoảng trắng để làm cho mã dễ đọc hơn)

```
mySqlCommandBuilder.GetInsertCommand().CommandText =
    INSERT INTO Customers(CustomerID , CompanyName , Address)
    VALUES (@p1 , @p2 , @p3)
mySqlCommandBuilder.GetUpdateCommand().CommandText =
    UPDATE Customers
    SET CustomerID = @p1 , CompanyName = @p2 , Address = @p3
    WHERE (CustomerID = @p4 AND CompanyName = @p5 AND Address = @p6)
mySqlCommandBuilder.GetDeleteCommand().CommandText =
    DELETE FROM Customers
    WHERE (CustomerID = @p1 AND CompanyName = @p2 AND Address = @p3)
```

Như bạn có thể thấy, những lệnh này tương tự như những lệnh đã trình bày trước đó trong mục " Sửa đổi những hàng ở một DataTable." Những câu lệnh SQL được dùng trong những lệnh này sử dụng sự tương tranh lạc quan.

Bây giờ bạn có thể cư trú và thực hiện những thay đổi tới một DataTable chứa những hàng từ bảng Customers, rồi đẩy những thay đổi đó tới cơ sở dữ liệu sử dụng phương thức Update(). Bạn có thể sử dụng phương thức AddDataRow(), ModifyDataRow(), Và RemoveDataRow() như trình bày trong mục trước đó, " Sửa đổi những hàng ở một DataTable."

**Ghi nhớ :** Bạn sẽ tìm thấy một chương trình đầy đủ có tên UsingCommandBuilder.cs trong thư mục ch11 minh họa sự sử dụng đối tượng CommandBuilder trình bày trong mục này. Danh sách này được bỏ qua trong sách này cho ngắn gọn.

## **KHÁM PHÁ *DataAdapter* VÀ NHỮNG SỰ KIỆN *DataTable*:**

Bạn sẽ tìm thấy tất cả những mã ví dụ được trình bày trong mục này trong chương trình UsingEvents.cs trong thư mục ch11. Danh sách chương trình này được bỏ qua trong sách này cho ngắn gọn. Nếu bạn biên dịch và chạy chương trình này, bạn sẽ thấy trình tự mà trong đó những sự cố phát khởi khi bạn thêm, sửa đổi, và loại bỏ một hàng từ một DataTable có chứa những hàng được truy xuất từ bảng Customers.

## **NHỮNG SỰ KIỆN *DataAdapter*:**

Những sự cố được phát khởi bởi một đối tượng SqlDataAdapter được trình bày trong Bảng 11.11.

Bảng 11.11: những sự kiện SqlDataAdapter

SỰ KIỆN	EVENT HANDLER	MÔ TẢ
FillError	FillErrorHandler	Phát khởi khi một lỗi xuất hiện trong thời gian một lệnh gọi tới phương thức Fill() .
RowUpdating	RowUpdatingEventHandler	Phát khởi <i>trước khi</i> một hàng được bổ sung, điều chỉnh, hay xóa trong cơ sở dữ liệu như một kết quả của việc gọi phương thức Update() .
RowUpdated	RowUpdatedEventHandler	Phát khởi <i>sau khi</i> một hàng được bổ sung, điều chỉnh, hay xóa trong cơ sở dữ liệu như một kết quả của việc gọi phương pháp Update() .

## **SỰ KIỆN *FillError*:**

Sự kiện FillError phát khởi khi bạn gọi phương thức Fill() mà một lỗi xảy ra. Sau đây là một cặp tình huống sẽ gây ra một lỗi :

- Một nỗ lực để thêm một số từ cơ sở dữ liệu đến một DataColumn mà vốn không thể chuyển đổi được vào kiểu dữ liệu .NET của DataColumn này mà không mất sự chính xác.
- Một nỗ lực để thêm một hàng từ cơ sở dữ liệu đến một DataTable mà xâm phạm một sự ràng buộc trong DataTable này.

ví dụ về Bộ xử lý sự kiện (event handler) sau đây, có tên FillErrorHandler(), kiểm tra lỗi chuyển đổi chính xác:

```
public static void FillErrorHandler(  
    object sender, FillEventArgs myFEEA  
)  
{  
    if (myFEEA.Errors.GetType() == typeof(System.OverflowException))  
    {
```

```

        Console.WriteLine("A loss of precision occurred");
        myFEEA.Continue = true;
    }
}

```

Tham số đầu tiên là một đối tượng của lớp System.Object , và nó đại diện cho đối tượng phát khởi sự kiện. Tham số thứ hai là một đối tượng của lớp FillEventArgs, nó giống như tất cả những lớp EventArgs, được dẫn xuất từ lớp System.EventArgs. Lớp EventArgs là lớp cơ sở cho dữ liệu sự kiện và đại diện cho những chi tiết của sự kiện. Bảng 11.12 cho thấy những thuộc tính FillEventArgs.

**Bảng 11.12: những thuộc tính FillEventArgs**

Thuộc tính	Kiểu	Mô tả
Continue	bool	Lấy hay gán một giá trị bool cho biết liệu có phải bạn muốn tiếp tục điền đầy Dataset của bạn mặc dù có một lỗi xảy ra. mặc định là false.
DataTable	DataTable	Lấy DataTable mà đã được điền đầy khi lỗi xuất hiện.
Errors	Exception	Lấy ngoại lệ đang chứa lỗi xảy ra.
Values	object[]	Lấy những giá trị DataColumn của DataRow mà trong đó lỗi xảy ra. Những giá trị này được trả lại trong một mảng đối tượng.

Bạn chỉ định mySqlDataAdapter sẽ gọi phương thức FillEventHandler() khi sự kiện FillError khởi phát sử dụng mã sau đây:

```

mySqlDataAdapter.FillError +=
new FillEventHandler(FillEventHandler);

```

### **SỰ KIỆN RowUpdating:**

Sự kiện RowUpdating phát khởi trước khi một hàng được cập nhật trong cơ sở dữ liệu như một kết quả của phương thức Update() của DataAdapter của bạn mà bạn đã gọi. Sự kiện này phát khởi một lần cho mỗi DataRow bạn bổ sung, sửa đổi, hay đã xóa ở một DataTable.

Những hoạt động sau đây được thực hiện phía sau ngữ cảnh cho mỗi DataRow khi bạn gọi phương thức Update() của DataAdapter của bạn :

1. Những giá trị trong DataRow của các bạn được sao chép tới những giá trị tham số của lệnh(Command) thích hợp trong thuộc tính InsertCommand, UpdateCommand, hay DeleteCommand của DataAdapter của bạn.
2. Sự kiện RowUpdating của DataAdapter của các bạn khởi phát .
3. Lệnh ( Command) được chạy để thay đổi tới cơ sở dữ liệu.
4. Mọi tham số đầu ra từ Command được trả về.
5. Sự kiện RowUpdated của DataAdapter của bạn khởi phát .
6. Phương thức AcceptChanges() của DataRow của bạn được gọi .

Tham số thứ hai tới bất kỳ phương thức bộ xử lý sự kiện nào bạn viết để xử lý sự kiện RowUpdating của một đối tượng SqlDataAdapter đều thuộc lớp SqlRowUpdatingEventArgs, và Bảng 11.13 cho thấy những thuộc tính của lớp này.

**Bảng 11.13: những thuộc tính SqlRowUpdatingEventArgs**

Thuộc tính	Kiểu	Mô tả
Command	SqlCommand	Lấy hay gán SqlCommand được chạy khi phương thức Update() được gọi.
Errors	Exception	Lấy ngoại lệ cho bất kỳ lỗi nào xuất hiện.
Row	DataRow	Lấy DataRow để gởi tới cơ sở dữ liệu thông qua phương thức Update() .

Thuộc tính	Kiểu	Mô tả
StatementType	StatementType	Lấy kiểu của phát biểu SQL sẽ được chạy. StatementType là một liệt kê trong namespace System.Data có chứa những thành viên sau đây: <ul style="list-style-type: none"> <li>■ Delete</li> <li>■ Insert</li> <li>■ Select</li> <li>■ Update</li> </ul>
Status	UpdateStatus	Lấy hay gán UpdateStatus của đối tượng Command. UpdateStatus là một liệt kê trong namespace System.Data chứa những thành viên sau đây: <ul style="list-style-type: none"> <li>■ <b>Continue</b> :cho biết DataAdapter sẽ tiếp tục việc xử lý những hàng.</li> <li>■ <b>ErrorsOccurred</b>: cho biết sự cập nhật này sẽ được xử lý như một lỗi.</li> <li>■ <b>SkipAllRemainingRows</b>: cho biết hàng hiện tại và tất cả các hàng còn lại đều sẽ được bỏ qua và không cập nhật.</li> <li>■ <b>SkipCurrentRow</b>: cho biết hàng hiện thời này sẽ được bỏ qua và không cập nhật. mặc định là Continue (tiếp tục).</li> </ul>
TableMapping	DataTableMapping	Lấy đối tượng DataTableMapping được gửi tới phương thức Update() . Một đối tượng DataTableMapping chứa một sự mô tả về một mối quan hệ được mô hình hóa giữa một bảng nguồn và một DataTable ( xem Chương 10, " Sử dụng những đối tượng Dataset để lưu trữ Dữ liệu ").

Ví dụ về Bộ xử lý sự kiện sau đây, có tên RowUpdatingEventHandler(), ngăn ngừa bất kỳ hàng mới nào được thêm vào cơ sở dữ liệu với một CustomerID có trị là J5COM:

```
public static void RowUpdatingEventHandler(
    object sender, SqlRowUpdatingEventArgs mySRUEA
)
{
    Console.WriteLine("\nIn RowUpdatingEventHandler()");
    if ((mySRUEA.StatementType == StatementType.Insert) &&
        (mySRUEA.Row["CustomerID"] == "J5COM"))
    {
        Console.WriteLine("Skipping current row");
        mySRUEA.Status = UpdateStatus.SkipCurrentRow;
    }
}
```

Bạn chỉ định mySqlDataAdapter sẽ gọi phương thức RowUpdatingEventHandler() khi sự kiện RowUpdating khởi phát sử dụng mã sau đây

```
mySqlDataAdapter.RowUpdating +=
    new SqlRowUpdatingEventHandler(RowUpdatingEventHandler);
```

Nếu bạn gọi phương thức AddDataRow() được trình bày trước đó để thử thêm một hàng vào bảng Customers , sự kiện RowUpdating sẽ khởi phát và gọi phương thức RowUpdatingEventHandler() . Phương thức này làm cho hàng mới bị bỏ qua và ngăn ngừa nó thêm vào bảng cơ sở dữ liệu Customers.

### **BIẾN CỐ RowUpdate:**

Sự kiện RowUpdated khởi phát sau khi một hàng được cập nhật trong cơ sở dữ liệu như một kết quả của việc gọi phương thức Update() của DataAdapter của bạn. Sự kiện này phát khởi một lần cho mỗi DataRow bạn bổ sung, sửa đổi, hay đã bị xóa ở một DataTable.

Tham số thứ hai tới bất kỳ phương thức nào mà bạn viết để xử lý sự kiện RowUpdated của một đối tượng SqlDataAdapter đều thuộc lớp SqlRowUpdatedEventArgs . Những thuộc tính của lớp này cũng giống như



những thuộc tính đã trình bày trước đây trong bảng 11.13 , cộng với một thuộc tính bổ sung trình bày trong Bảng 11.14.

**Bảng 11.14: Thuộc tính SqlRowUpdatedEventArgs Bổ sung**

Thuộc tính	Kiểu	Mô tả
RecordsAffected	int	Lấy một int chứa số lượng hàng đã được thêm vào, sửa đổi, hay loại bỏ khi lệnh (Command) thích hợp được chạy bởi phương thức Update() .

ví dụ về Bộ xử lý sự kiện sau đây, có tên RowUpdatedEventHandler(), nó hiển thị số lượng bản ghi bị ảnh hưởng bởi Command sau đây:

```
public static void RowUpdatedEventHandler(  
    object sender, SqlRowUpdatedEventArgs mySRUEA  
)  
{  
    Console.WriteLine("\nIn RowUpdatedEventHandler()");  
    Console.WriteLine("mySRUEA.RecordsAffected = " +  
        mySRUEA.RecordsAffected);  
}
```

Bạn chỉ định mySqlDataAdapter sẽ gọi phương thức RowUpdatedEventHandler() khi sự kiện RowUpdated khởi phát sử dụng mã sau đây:

```
mySqlDataAdapter.RowUpdated +=  
    new SqlRowUpdatedEventHandler(RowUpdatedEventHandler);
```

### **NHỮNG SỰ KIỆN DataTable:**

Những sự kiện được phát khởi từ một đối tượng DataTable được trình bày trong Bảng 11.15.

**Bảng 11.15: những sự kiện DataTable**

Sự kiện	EVENT HANDLER	Mô tả
ColumnChanging	DataColumnChangeEventHandler	Phát khởi trước khi một giá trị đã được thay đổi của DataColumn được giao phó ở một DataRow.
ColumnChanged	DataColumnChangeEventHandler	Phát khởi sau khi một giá trị đã được thay đổi của DataColumn được giao phó ở một DataRow.
RowChanging	DataRowChangeEventHandler	Phát khởi trước khi một DataRow đã được thay đổi, được giao phó ở một DataTable.
RowChanged	DataRowChangeEventHandler	Phát khởi sau khi một DataRow đã được thay đổi, được giao phó ở một DataTable.
RowDeleting	DataRowChangeEventHandler	Phát khởi trước khi một DataRow bị xóa từ một DataTable.
RowDeleted	DataRowChangeEventHandler	Phát khởi trước khi một DataRow bị xóa từ một DataTable.

### **NHỮNG SỰ KIỆN ColumnChanging VÀ ColumnChanged:**

Những sự kiện ColumnChanging phát khởi trước khi một sự thay đổi tới một giá trị DataColumn được giao phó ở một DataRow. Tương tự, những sự kiện ColumnChanged phát khởi sau khi một sự thay đổi tới một giá trị DataColumn được giao phó ở một DataRow. Hai sự kiện này luôn được phát khởi trước những sự kiện RowChanging và RowChanged.

Từ " Commit " trong văn cảnh này có nghĩa là gì? Nếu bạn đơn giản đặt một giá trị mới cho một DataColumn, thì sự thay đổi được tự động được giao phó trong DataRow. Tuy nhiên, nếu bạn bắt đầu sự thay đổi tới DataRow sử dụng phương thức BeginEdit() , thì sự thay đổi chỉ được giao phó khi bạn gọi phương thức EndEdit() của DataRow này. Bạn cũng có thể loại bỏ sự thay đổi tới DataRow sử dụng phương thức CancelEdit() .

Tham số thứ hai tới bất kỳ bộ xử lý sự kiện nào mà bạn viết để xử lý những sự kiện ColumnChanging hay ColumnChanged của một đối tượng DataTable điều thuộc lớp DataColumnChangeEventArgs. Bảng 11.16 cho thấy những thuộc tính (của) lớp này.

**Bảng 11.16: những thuộc tính DataColumnChangeEventArgs**

Thuộc tính	Kiểu	Mô tả
Column	DataColumn	Lấy DataColumn với giá trị mà đang thay đổi.
ProposedValue	object	Lấy hay gán giá trị mới cho DataColumn.
Row	DataRow	Lấy DataRow chứa DataColumn với giá trị mà đang thay đổi.

Ví dụ về những bộ xử lý phương thức sau đây, có tên ColumnChangingEventHandler() và ColumnChangedEventHandler(), trình bày những thuộc tính Column và ProposedValue:

```
public static void ColumnChangingEventHandler(  
    object sender,  
    DataColumnChangeEventArgs myDCCEA  
)  
{  
    Console.WriteLine("\nIn ColumnChangingEventHandler()");  
    Console.WriteLine("myDCCEA.Column = " + myDCCEA.Column);  
    Console.WriteLine("myDCCEA.ProposedValue = " + myDCCEA.ProposedValue);  
}  
  
public static void ColumnChangedEventHandler(  
    object sender,  
    DataColumnChangeEventArgs myDCCEA  
)  
{  
    Console.WriteLine("\nIn ColumnChangedEventHandler()");  
    Console.WriteLine("myDCCEA.Column = " + myDCCEA.Column);  
    Console.WriteLine("myDCCEA.ProposedValue = " + myDCCEA.ProposedValue);  
}
```

Ví dụ tiếp theo tạo ra một DataTable có tên customersDataTable và thêm hai phương thức trước đây tới những sự kiện ColumnChanging và ColumnChanged của customersDataTable:

```
DataTable customersDataTable = myDataSet.Tables["Customers"];  
  
customersDataTable.ColumnChanging +=  
    new DataColumnChangeEventHandler(ColumnChangingEventHandler);  
  
customersDataTable.ColumnChanged +=  
    new DataColumnChangeEventHandler(ColumnChangedEventHandler);
```

### **SỰ KIỆN RowChanging VÀ RowChanged:**

Sự kiện RowChanging khởi phát trước khi một sự thay đổi tới một DataRow được giao phó ở một DataTable. Tương tự, sự kiện RowChanged khởi phát sau khi một sự thay đổi tới một giá trị DataRow được giao phó ở một DataTable.

Tham số thứ hai tới bất kỳ bộ xử lý sự kiện nào bạn viết để xử lý những sự kiện RowChanging hay RowChanged của một đối tượng DataTable đều thuộc lớp DataRowChangeEventArgs. Bảng 11.17 cho thấy những thuộc tính (của) lớp này.

**Bảng 11.17: những thuộc tính DataRowChangeEventArgs**

Thuộc tính	Kiểu	Mô tả
Action	DataRowAction	Lấy DataRowAction xuất hiện cho DataRow. Sự liệt kê DataRowAction được định nghĩa trong namespace System.Data và chứa những thành viên sau đây: <ul style="list-style-type: none"><li>● Add: cho biết DataRow đã được thêm vào DataTable.</li><li>● Change: cho biết DataRow đã được sửa đổi.</li><li>● Commit: cho biết DataRow đã được "giao phó" trong DataTable.</li><li>● Delete: cho biết DataRow đã được loại bỏ từ DataTable.</li><li>● Nothing: cho biết DataRow không thay đổi.</li><li>● Rollback: cho biết sự thay đổi tới DataRow đã được hủy.</li></ul>
Row	DataRow	Lấy DataRow chứa DataColumn với giá trị đang thay đổi.

ví dụ về những bộ xử lý sự kiện sau đây, có tên RowChangingEventHandler() và RowChangedEventHandler(), hiển thị thuộc tính Action:

```
public static void RowChangingEventHandler(  
    object sender,  
    DataRowChangeEventArgs myDRCEA  
)  
{  
    Console.WriteLine("\nIn RowChangingEventHandler()");  
    Console.WriteLine("myDRCEA.Action = " + myDRCEA.Action);  
}  
  
public static void RowChangedEventHandler(  
    object sender,  
    DataRowChangeEventArgs myDRCEA  
)  
{  
    Console.WriteLine("\nIn RowChangedEventHandler()");  
    Console.WriteLine("myDRCEA.Action = " + myDRCEA.Action);  
}
```

Ví dụ tiếp theo thêm hai phương thức trước đây tới những sự kiện RowChanging và RowChanged của customersDataTable:

```
customersDataTable.RowChanging +=  
    new DataRowChangeEventHandler(RowChangingEventHandler);  
  
customersDataTable.RowChanged +=  
    new DataRowChangeEventHandler(RowChangedEventHandler);
```

### **SỰ KIỆN RowDeleting VÀ RowDeleted:**

Sự kiện RowDeleting phát khởi trước khi một DataRow bị xóa từ một DataTable. Tương tự, Sự kiện RowDeleted khởi phát sau khi một DataRow bị xóa từ một DataTable.

Tham số thứ hai tới bất kỳ bộ xử lý sự kiện nào bạn viết để xử lý những sự kiện RowDeleting hay RowDeleted của một DataTable đều thuộc lớp DataRowChangeEventArgs, và Bảng 11.17 trước cho thấy những thuộc tính

của lớp này.

Ví dụ về những " bộ điều khiển phương thức" sau đây, có tên RowDeletingEventHandler() và RowDeletedEventHandler(), trình bày thuộc tính Action:

```
public static void RowDeletingEventHandler(  
    object sender,  
    DataRowChangeEventArgs myDRCEA  
)  
{  
    Console.WriteLine("\nIn RowDeletingEventHandler()");  
    Console.WriteLine("myDRCEA.Action = " + myDRCEA.Action);  
}  
  
public static void RowDeletedEventHandler(  
    object sender,  
    DataRowChangeEventArgs myDRCEA  
)  
{  
    Console.WriteLine("\nIn RowDeletedEventHandler()");  
    Console.WriteLine("myDRCEA.Action = " + myDRCEA.Action);  
}
```

Ví dụ kế tiếp thêm hai phương thức trước tới những sự kiện RowDeleting và RowDeleted của customersDataTable:

```
customersDataTable.RowDeleting +=  
    new DataRowChangeEventHandler(RowDeletingEventHandler);  
  
customersDataTable.RowDeleted +=  
    new DataRowChangeEventHandler(RowDeletedEventHandler);
```

Ghi nhớ: Bạn sẽ tìm thấy tất cả mã những ví dụ trình bày trong mục này trong chương trình UsingEvents.cs định vị trong thư mục ch11. Danh sách chương trình này được bỏ qua trong sách này cho ngắn gọn.

### **GIẢI QUẾT NHỮNG SỰ THẤT BẠI CẬP NHẬT:**

Cho đến lúc này, những ví dụ bạn đã thấy đều giả thiết rằng những sự cập nhật đẩy tới cơ sở dữ liệu bởi phương thức Update() đều thành công. Trong mục này, bạn sẽ thấy những gì xảy ra khi những sự cập nhật thất bại - và những gì bạn có thể làm về điều này.

Ghi nhớ : Bạn sẽ tìm thấy tất cả mã ví dụ trình bày trong mục này trong tệp tin HandlingUpdateFailures.cs được định vị trong thư mục ch11. Danh sách chương trình này được bỏ qua trong sách này cho ngắn gọn.

Trong những ví dụ trong mục này, giả thiết rằng thuộc tính CommandText của một UpdateCommand của đối tượng SqlDataAdapter được thiết đặt như sau:

```
UPDATE Customers  
SET  
    CompanyName = @NewCompanyName,  
    Address = @NewAddress  
WHERE CustomerID = @OldCustomerID  
AND CompanyName = @OldCompanyName  
AND Address = @OldAddress
```

Phát biểu Update này sử dụng sự "tương tranh lạc quan" bởi vì những cột được cập nhật bao gồm trong mệnh đề WHERE.

### **MỘT TÌNH HUỐNG THẤT BẠI CẬP NHẬT:**

Xem xét tình huống sau đây trình bày một sự thất bại cập nhật:

1. Người sử dụng 1 truy xuất những hàng từ bảng Customers vào trong một DataTable có tên customersDataTable.
2. Người sử dụng 2 cũng truy xuất cùng hàng này.
3. Người sử dụng 1 cập nhật cột dữ liệu CustomerName của DataRow với cột khóa chính CustomerID là J5COM và đẩy sự thay đổi tới cơ sở dữ liệu. Cho là người sử dụng 1 thay đổi CustomerName từ J5 Company đến Updated Company.
4. Người sử dụng 2 cũng cập nhật cùng DataRow này và thay đổi CompanyName từ J5 Company đến Widgets Inc. và nỗ lực để đẩy sự thay đổi này tới cơ sở dữ liệu. như vậy người sử dụng 2 gây cho một đối tượng DBConcurrency- Exception sẽ được ném ra và sự cập nhật của họ thất bại. (Ngoại lệ giống như vậy xuất hiện nếu người sử dụng 2 thử cập nhật hay xóa một hàng mà đã bị xóa bởi người sử dụng 1.)

Tại sao sự cập nhật trong bước 4 bị thất bại ? Lý do là với sự “ tương tranh lạc quan”, cột *CompanyName* được sử dụng trong mệnh đề WHERE của phát biểu *Update*. Bởi vì điều này, hàng nguyên bản được tải bởi người sử dụng 2 không thể tìm thấy nữa - và do đó phát biểu cập nhật thất bại. Hàng không thể tìm được vì người sử dụng 1 đã thay đổi cột CompanyName từ *J5 Company* đến *Updated Company* trong bước 2.

Đây là vấn đề với sự "tương tranh lạc quan", nhưng điều gì mà Bạn có thể làm như một người lập trình ? Bạn có thể báo cáo sự cố tới người sử dụng 2, làm tươi lại những hàng của họ sử dụng phương thức Fill(), để họ có thể thực hiện sự thay đổi của họ lần nữa - tuy nhiên, nếu người sử dụng 2 đã thực rất nhiều lần sự thay đổi và họ cũng không thể lưu trữ bất kỳ thứ gì của họ, họ có lẽ sẽ rất bất mãn về chương trình của bạn.

May mắn thay, bạn có thể đặt thuộc tính ContinueUpdateOnError của DataAdapter của bạn là true để tiếp tục cập nhật bất kỳ đối tượng DataRow nào dù một lỗi xuất hiện. với cách này, khi người sử dụng 2 lưu những sự thay đổi của họ , họ có thể ít nhất lưu được những hàng mà không gây ra bất kỳ lỗi nào. Chúng ta hãy xem xét cách để thiết đặt thuộc tính ContinueUpdateOnError như thế nào.

### **THIẾT ĐẶT THUỘC TÍNH ContinueUpdateOnError:**

Ví dụ sau đây đặt thuộc tính ContinueUpdateOnError tới true cho mySqlDataAdapter:

```
mySqlDataAdapter.ContinueUpdateOnError = true;
```

Khi bạn gọi mySqlDataAdapter.Update(), nó sẽ đẩy tất cả những sự thay đổi mà không gây ra những lỗi tới cơ sở dữ liệu.và Bạn có thể kiểm tra những lỗi sau đó sử dụng thuộc tính HasErrors của một Dataset hay thuộc tính HasErrors của những đối tượng DataTable riêng lẻ, những lỗi mà Bạn sẽ thấy không lâu sau đây trong mục " Sự kiểm tra những lỗi."

### **LẬP TRÌNH MỘT VÍ DỤ CẬP NHẬT THẤT BẠI:**

Chúng ta hãy lập trình một ví dụ về một sự cập nhật bị thất bại. Ví dụ này sẽ mô phỏng những sự cập nhật gây ra bởi người sử dụng 1 và người sử dụng 2 mô tả trước đó. Tôi sẽ sử dụng phương thức sau đây, có tên ModifyRowsUsingUPDATE(), để mô phỏng sự cập nhật thực hiện bởi người sử dụng 1 trong bước 3 mô tả trước đó

```
public static void ModifyRowUsingUPDATE(  
    SqlConnection mySqlConnection  
)  
{  
    Console.WriteLine("\n\n ModifyDataRowUsingUPDATE()");
```

```

Console.WriteLine("Updating CompanyName to 'Updated Company' for J5COM");

SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "UPDATE Customers " +
    "SET CompanyName = 'Updated Company' " +
    "WHERE CustomerID = 'J5COM'";

mySqlConnection.Open();
int numberOfRows = mySqlCommand.ExecuteNonQuery();
Console.WriteLine("Number of rows updated = " +
    numberOfRows);
mySqlConnection.Close();
}

```

Chú ý rằng CompanyName được gán là Update Company cho hàng với CustomerID là J5COM.

```

public static void ModifyDataRow(
    DataTable myDataTable,
    SqlDataAdapter mySqlDataAdapter,
    SqlConnection mySqlConnection
)
{
    Console.WriteLine("\nIn ModifyDataRow()");

    // step 1: set the PrimaryKey property of the DataTable
    myDataTable.PrimaryKey =
        new DataColumn[]
        {
            myDataTable.Columns["CustomerID"]
        };

    // step 2: use the Find() method to locate the DataRow
    // in the DataTable using the primary key value
    DataRow myDataRow = myDataTable.Rows.Find("J5COM");

    // step 3: change the DataColumn values of the DataRow
    myDataRow["CompanyName"] = "Widgets Inc.";
    Console.WriteLine("myDataRow.RowState = " +
        myDataRow.RowState);

    // step 4: use the Update() method to push the modified
    // row to the database
    Console.WriteLine("Calling mySqlDataAdapter.Update()");
    mySqlConnection.Open();
    int numOfRows = mySqlDataAdapter.Update(myDataTable);
    mySqlConnection.Close();
    Console.WriteLine("numOfRows = " + numOfRows);
    Console.WriteLine("myDataRow.RowState = " +
        myDataRow.RowState);

    DisplayDataRow(myDataRow, myDataTable);
}

```

Ví dụ kế tiếp gọi ModifyRowUsingUPDATE() để thực hiện sự cập nhật đầu tiên của hàng với CustomerID là J5COM:

`ModifyRowUsingUPDATE(mySqlConnection);`

Tiếp theo, thuộc tính `ContinueUpdateOnError` của `mySqlDataAdapter` được gán là `true` để tiếp tục cập nhật bất kỳ đối tượng `DataRow` nào dù một lỗi xảy ra

`mySqlDataAdapter.ContinueUpdateOnError = true;`

Tiếp theo, `ModifyDataRow()` được gọi để thử sửa đổi cùng một hàng như `ModifyRowsUsingUPDATE()`:

`ModifyDataRow(customersDataTable, mySqlDataAdapter, mySqlConnection);`

Thông thường, điều này sẽ ném ra một ngoại lệ vì hàng này không thể tìm thấy được, nhưng do thuộc tính `ContinueUpdateOnError` của `mySqlDataAdapter` được thiết lập tới `true`, không có ngoại lệ được ném ra. Bằng cách này, nếu `myDataTable` có những hàng được cập nhật khác, chúng sẽ vẫn được đẩy tới cơ sở dữ liệu bởi sự gọi tới phương thức `Update()`.

### **KIỂM TRA NHỮNG LỖI:**

Khi thuộc tính `ContinueUpdateOnError` của `mySqlDataAdapter` được gán tới `true`, sẽ không có ngoại lệ được ném khi một lỗi xuất hiện. Thay vào đó, bạn có thể kiểm tra những lỗi trong một `Dataset` hay `DataTable` riêng rẽ hay `DataRow` sử dụng thuộc tính `HasErrors`. rồi Bạn có thể hiển thị cho người sử dụng, những chi tiết của lỗi, sử dụng thuộc tính `RowError` của `DataRow`, Cùng với nguyên bản và những giá trị hiện tại cho những đối tượng `DataColumn` trong `DataRow` đó. ví dụ:

```
if (myDataSet.HasErrors)
{
    Console.WriteLine("\nDataSet has errors!");
    foreach (DataTable myDataTable in myDataSet.Tables)
    {
        // check the HasErrors property of myDataTable
        if (myDataTable.HasErrors)
        {
            foreach (DataRow myDataRow in myDataTable.Rows)
            {
                // check the HasErrors property of myDataRow
                if (myDataRow.HasErrors)
                {
                    Console.WriteLine("Here is the row error:");
                    Console.WriteLine(myDataRow.RowError);
                    Console.WriteLine("Here are the column details in the DataSet:");
                    foreach (DataColumn myDataColumn in myDataTable.Columns)
                    {
                        Console.WriteLine(myDataColumn + "original value = " +
                            myDataRow[myDataColumn, DataRowVersion.Original]);
                        Console.WriteLine(myDataColumn + "current value = " +
                            myDataRow[myDataColumn, DataRowVersion.Current]);
                    }
                }
            }
        }
    }
}
```

### **CHỈNH LÝ LỖI:**



Hiện thị lỗi cho người sử dụng chỉ là nửa công việc. Bạn có thể làm gì để chỉnh định vấn đề ? Một giải pháp là gọi phương thức Fill() lần nữa để đồng bộ hóa những hàng trong Dataset của bạn với cơ sở dữ liệu. bằng cách này, hàng được cập nhật bởi ModifyRowsUsingUPDATE() trong bảng Customers sẽ được kéo từ bảng Customers và thay thế hàng nguyên bản J5COM trong Dataset. Chẳng hạn:

```
mySqlConnection.Open();
numOfRows =
mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();
```

Bạn có lẽ đã mong đợi numOfRows sẽ là 1, bởi vì Bạn đang thay thế chỉ một hàng, đúng ? sai : numOfRows sẽ chứa tổng số lượng hàng trong bảng Customers . Lý do cho điều này là phương thức Fill() thật sự kéo tất cả những hàng từ bảng Customers và đặt chúng trong DataTable Customers của myDataSet, nó vứt bỏ bất kỳ hàng hiện hữu nào có giá trị cột khóa chính phù hợp đã có trong DataTable Customers.

**Cảnh báo:** Nếu bạn đã không thêm một khóa chính vào DataTable Customers của bạn , thì sự gọi tới phương thức Fill() chỉ đơn giản là thêm tất cả những hàng từ bảng Customers đến DataTable Customers lần nữa- sự trùng lặp những hàng xảy ra ở đây.

Tiếp theo, bạn gọi ModifyDataRow() lần nữa để điều chỉnh hàng J5COM :

```
ModifyDataRow(customersDataTable, mySqlDataAdapter, mySqlConnection);
```

Lần này sự cập nhật thành công vì ModifyDataRow() tìm kiếm hàng J5COM trong customersDataTable mà vừa được kéo về từ bảng Customers.

Ghi nhớ: nếu một người sử dụng cố cập nhật một hàng đã bị xóa từ bảng cơ sở dữ liệu, thì điều duy nhất bạn có thể làm là làm mới (refresh) lại những hàng và yêu cầu người sử dụng cập nhật lại hàng của họ lần nữa.

### **SỬ DỤNG Transaction(NHỮNG GIAO DỊCH) VỚI MỘT Dataset(SQL):**

Trong Chương 3, " giới thiệu Ngôn ngữ truy vấn có cấu trúc", bạn đã thấy cách nhóm những câu lệnh SQL cùng nhau như thế nào vào trong transactions (giao dịch).rồi transaction được giao phó hay hồi phục như một đơn vị. Chẳng hạn, trong trường hợp của một giao dịch ngân hàng bạn có lẽ đã muốn rút tiền từ một tài khoản và chuyển vào trong tài khoản khác. rồi Bạn giao phó cả hai sự thay đổi này như một đơn vị, hoặc nếu xảy ra một vấn đề, hồi phục lại cả hai thay đổi. Trong Chương 8, " thực hiện những lệnh Cơ sở dữ liệu ", bạn đã thấy cách sử dụng một đối tượng transaction (Giao dịch) như thế nào để đại diện cho một Giao dịch.

Như bạn biết, một Dataset không có một sự nối kết trực tiếp tới cơ sở dữ liệu. Thay vào đó, bạn sử dụng những phương thức Fill() và Update() tương ứng của một DataAdapter để kéo và đẩy những hàng từ và tới cơ sở dữ liệu vào Dataset của bạn . Thật ra, một Dataset không có chút " kiến thức " nào về cơ sở dữ liệu . Một Dataset đơn giản cất giữ một bản sao không kết nối của dữ liệu. Vì điều này, một Dataset không có bất kỳ chức năng gắn sẵn nào để xử lý những giao dịch.

rồi Bạn sử dụng những giao dịch với một Dataset như thế nào? Câu trả lời là bạn phải sử dụng thuộc tính Transaction của những đối tượng Command được cất giữ trong một DataAdapter.

### **SỬ DỤNG THUỘC TÍNH TRANSACTION CỦA ĐỐI TƯỢNG DataAdapter Command**

Một DataAdapter cất giữ bốn đối tượng Command (Lệnh) mà khi bạn truy nhập sử dụng những thuộc tính SelectCommand, InsertCommand, UpdateCommand, và DeleteCommand. Khi bạn gọi phương thức Update() của một DataAdapter, nó sẽ chạy InsertCommand, UpdateCommand, hay DeleteCommand thích hợp.

Bạn có thể tạo ra một đối tượng Giao dịch (Transaction) và thuộc tính Transaction của những đối tượng Command trong DataAdapter của bạn được gán với Transaction này. rồi khi bạn sửa đổi Dataset của bạn và đẩy những sự thay đổi tới cơ sở dữ liệu sử dụng phương thức Update() của DataAdapter của bạn, Những sự thay đổi sẽ sử dụng cùng một Transaction.

Ví dụ sau đây tạo ra một đối tượng SqlTransaction có tên mySqlTransaction và gán thuộc tính Transaction của toàn bộ những đối tượng Command trong mySqlDataAdapter tới mySqlTransaction:

```
SqlTransaction mySqlTransaction =  
    mySqlConnection.BeginTransaction();  
mySqlDataAdapter.SelectCommand.Transaction = mySqlTransaction;  
mySqlDataAdapter.InsertCommand.Transaction = mySqlTransaction;  
mySqlDataAdapter.UpdateCommand.Transaction = mySqlTransaction;  
mySqlDataAdapter.DeleteCommand.Transaction = mySqlTransaction;
```

Toàn bộ những đối tượng Command trong mySqlDataAdapter bây giờ sẽ sử dụng mySqlTransaction.

Chúng ta hãy cho là bạn thêm, điều chỉnh, và loại bỏ một số hàng từ một DataTable được chứa trong một Dataset có tên myDataSet. Bạn có thể đẩy những sự thay đổi này tới cơ sở dữ liệu sử dụng ví dụ sau đây:

```
mySqlDataAdapter.Update(myDataSet);
```

Tất cả những sự thay đổi của các bạn tới myDataSet được đẩy tới cơ sở dữ liệu như bộ phận của giao dịch trong mySqlTransaction. Bạn có thể giao phó những sự thay đổi đó sử dụng phương thức Commit() của mySqlTransaction:

Bạn có thể cũng phục nguyên những sự thay đổi đó sử dụng phương thức Rollback() của mySqlTransaction.

### Ghi chú:

Một giao dịch là phục nguyên theo mặc định; bởi vậy, bạn cần phải luôn luôn chỉ định rõ ràng là giao phó hay phục nguyên giao dịch của bạn, sử dụng Commit() hay Rollback() để nó thực hiện đúng cái mà chương trình của bạn dự định làm.

## **SỬA ĐỔI DỮ LIỆU SỬ DỤNG MỘT *Dataset* ĐỊNH KIỂU DỮ LIỆU MẠNH**

Trong [Chương 10](#), bạn đã thấy cách tạo và sử dụng một lớp Dataset định kiểu dữ liệu mạnh có tên MyDataSet như thế nào. Bạn có thể sử dụng những đối tượng của lớp này để đại diện cho bảng Customers và những hàng từ bảng này. Trong mục này, bạn sẽ thấy cách sửa đổi dữ liệu như thế nào sử dụng một đối tượng định kiểu mạnh của lớp MyDataSet.

### Ghi chú

Một trong số những đặc tính của một đối tượng Dataset định kiểu mạnh cho phép bạn đọc một giá trị cột sử dụng một thuộc tính có cùng tên với cột. Chẳng hạn, để đọc CustomerID của một cột bạn có thể sử dụng myDataRow.CustomerID thay vì myDataRow["CustomerID"]. Xem [Chương 10 cho nhiều chi tiết về đọc giá trị cột hơn](#).

Ba phương thức sau đây trong lớp MyDataSet cho phép bạn sửa đổi những hàng được cất giữ trong một đối tượng MyDataSet gồm : NewCustomersRow(), AddCustomersRow(), và RemoveCustomersRow(). Bạn có thể tìm thấy một hàng sử dụng phương thức FindByCustomerID(). Bạn có thể kiểm tra liệu một giá trị cột chứa

đựng một giá trị null sử dụng những phương thức như IsContactNameNull(), và Bạn có thể đặt một cột tới null sử dụng những phương thức như SetContactNameNull(). Bạn sẽ thấy cách sử dụng phương thức này không lâu say đây.

Phương thức Form1\_Load() của Form trong dự án ví dụ trình bày cách để thêm, sửa đổi, và loại bỏ một hàng tới một đối tượng Dataset kiểu dữ liệu mạnh có tên myDataSet1. Bạn sẽ thấy những bước để hoàn thành những nhiệm vụ này trong phương thức Form1\_Load() sau :

```
private void Form1_Load(object sender, System.EventArgs e)
{
    // populate the DataSet with the CustomerID, CompanyName,
    // and Address columns from the Customers table
    sqlConnection1.Open();
    sqlDataAdapter1.Fill(myDataSet1, "Customers");

    // get the Customers DataTable
    MyDataSet.CustomersDataTable myDataTable =
        myDataSet1.Customers;

    // create a new DataRow in myDataTable using the
    // NewCustomersRow() method of myDataTable
    MyDataSet.CustomersRow myDataRow =
        myDataTable.NewCustomersRow();

    // set the CustomerID, CompanyName, and Address of myDataRow
    myDataRow.CustomerID = "J5COM";
    myDataRow.CompanyName = "J5 Company";
    myDataRow.Address = "1 Main Street";

    // add the new row to myDataTable using the
    // AddCustomersRow() method
    myDataTable.AddCustomersRow(myDataRow);

    // push the new row to the database using
    // the Update() method of sqlDataAdapter1
    sqlDataAdapter1.Update(myDataTable);

    // find the row using the FindByCustomerID()
    // method of myDataTable
    myDataRow = myDataTable.FindByCustomerID("J5COM");
    // modify the CompanyName and Address of myDataRow
    myDataRow.CompanyName = "Widgets Inc.";
    myDataRow.Address = "1 Any Street";

    // push the modification to the database
    sqlDataAdapter1.Update(myDataTable);

    // display the DataRow objects in myDataTable
    // in the listView1 object
    foreach (MyDataSet.CustomersRow myDataRow2 in myDataTable.Rows)
    {
        listView1.Items.Add(myDataRow2.CustomerID);
        listView1.Items.Add(myDataRow2.CompanyName);

        // if the Address is null, set Address to "Unknown"
```

```

        if (myDataRow2.IsAddressNull() == true)
        {
            myDataRow2.Address = "Unknown";
        }
        listView1.Items.Add(myDataRow2.Address);
    }

    // find and remove the new row using the
    // FindByCustomerID() and RemoveCustomersRow() methods
    // of myDataTable
    myDataRow = myDataTable.FindByCustomerID("J5COM");
    myDataTable.RemoveCustomersRow(myDataRow);

    // push the delete to the database
    sqlDataAdapter1.Update(myDataTable);

    sqlConnection1.Close();
}

```

Hãy thoải mái biên tập và chạy form ví dụ.

### **TÓM TẮT:**

Trong chương này, bạn học cách sửa đổi những hàng trong một Dataset và sau đó đẩy những sự thay đổi đó tới cơ sở dữ liệu qua một DataAdapter như thế nào.

Bạn đã thấy cách thêm những sự hạn chế vào một DataTable và những đối tượng DataColumn của nó như thế nào. Điều này cho phép bạn mô hình những sự hạn chế tương tự dựa trên những bảng cơ sở dữ liệu và những cột trong những đối tượng DataTable và DataColumn của bạn. Bởi việc thêm những sự hạn chế lên trên về phía trước, bạn ngăn ngừa dữ liệu xấu được thêm vào Dataset của bạn, và những trợ giúp này giảm bớt những lỗi khi thử đẩy những thay đổi trong Dataset của bạn tới cơ sở dữ liệu.

Mỗi hàng trong một DataTable được cất giữ trong một đối tượng DataRow, và bạn thấy cách để tìm, lọc, và phân loại những đối tượng DataRow ở một DataTable sử dụng phương thức Find() của một DataTable như thế nào. Bạn cũng học cách lọc và phân loại những đối tượng DataRow ở một DataTable như thế nào sử dụng phương thức Select() .

Bạn đã thấy những bước được yêu cầu để thêm, sửa đổi, và loại bỏ những đối tượng DataRow từ một DataTable và sau đó đẩy những thay đổi này đến cơ sở dữ liệu. Để làm điều này đầu tiên bạn phải thiết lập DataAdapter của bạn với những đối tượng Command chứa những phát biểu SQL INSERT, UPDATE, và DELETE tương ứng. Bạn cất giữ những đối tượng Command này trong những thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của đối tượng DataAdapter của bạn. Bạn đẩy những thay đổi từ Dataset của bạn đến cơ sở dữ liệu sử dụng phương thức Update() của DataAdapter của bạn. Khi bạn thêm, điều chỉnh, hay loại bỏ những đối tượng DataRow từ Dataset của bạn và sau đó gọi phương thức Update() của DataAdapter của bạn, InsertCommand, UpdateCommand, hay DeleteCommand thích hợp được chạy để đẩy những thay đổi của bạn tới cơ sở dữ liệu.

Sự Tương tranh xác định những sự sửa đổi của nhiều người sử dụng tới cùng hàng được xử lý như thế nào. Với sự tương tranh lạc quan, bạn có thể sửa đổi một hàng trong một bảng cơ sở dữ liệu chỉ khi không ai khác đã sửa đổi chính hàng đó từ khi bạn tải nó vào trong Dataset của các bạn. Đây là kiểu tương tranh điển hình tốt nhất để sử dụng bởi vì bạn không muốn ghi đè những sự thay đổi của một người khác. Với sự tương tranh " người

trước thắng ", bạn luôn luôn có thể sửa đổi một hàng- và những sự thay đổi của bạn ghi đè lên những sự thay đổi bất cứ ai khác. Bạn điền hình muốn tránh sử dụng sự tương tranh " người trước thắng " .

Bạn có thể có một đối tượng DataAdapter để gọi những thủ tục lưu trữ để thêm, sửa đổi, và loại bỏ những hàng từ cơ sở dữ liệu. Những thủ tục này được gọi (thay cho những phát biểu Chèn, Cập nhật, và Xóa) và bạn được thấy cách thiết đặt những thuộc tính InsertCommand, UpdateCommand, Và DeleteCommand của một đối tượng DataAdapter .

Việc cung cấp những phát biểu Chèn, Cập nhật và Xóa hay những thủ tục được lưu trữ của bạn để đẩy những thay đổi từ Dataset của bạn đến cơ sở dữ liệu có nghĩa là bạn phải viết nhiều mã. Bạn có thể tránh viết mã này bởi việc sử dụng một đối tượng CommandBuilder, nó có thể tự động phát sinh những phát biểu Chèn , Cập nhật, và xóa bằng đơn để đẩy những thay đổi Bạn làm với một đối tượng Dataset tới cơ sở dữ liệu. Những lệnh này sẽ được gán trong thuộc tính InsertCommand, UpdateCommand, và DeleteCommand của đối tượng DataAdapter của bạn .

Bạn cũng đã thấy cách xử lý những sự thất bại cập nhật và sử dụng những giao dịch với một Dataset như thế nào, và trong mục cuối cùng của chương này Bạn đã thấy cách cập nhật những hàng được cất giữ trong một Dataset kiểu dữ liệu mạnh như thế nào.

Trong [Chương 12](#), bạn sẽ học cách định hướng và sửa đổi dữ liệu liên quan như thế nào.

## **CHƯƠNG 12: ĐỊNH HƯỚNG VÀ SỬA ĐỔI DỮ LIỆU LIÊN HỆ:**

### **Tổng quan**

Trong [chương 2](#), " Giới thiệu về những cơ sở dữ liệu, " Bạn đã thấu suốt những bảng cơ sở dữ liệu liên quan lẫn nhau thông qua những khóa ngoại. Chẳng hạn, cột OrderID của bảng Orders liên quan đến cột CustomerID của bảng Customers thông qua một khóa ngoại. Vì bảng Orders phụ thuộc vào bảng Customers, Bảng Orders được hiểu như bảng con và bảng Customers như bảng cha . Khóa ngoại được nói tới để định nghĩa một mối quan hệ cha con giữa hai bảng.

Trong chương này, bạn sẽ đi sâu vào trong những chi tiết của những đối tượng UniqueConstraint, ForeignKeyConstraint, và DataRelation . Bạn sẽ cũng xem xét cách định hướng những hàng trong những đối tượng DataTable liên quan, thực hiện những thay đổi trong những đối tượng DataTable liên quan , và cuối cùng đẩy những sự thay đổi đó tới cơ sở dữ liệu.

### **Mẹo nhỏ:**

Để cải thiện sự thực hiện khi tải một DataTable với một lượng lớn đối tượng DataRow , bạn cần phải đặt thuộc tính EnforceConstraints của Dataset của bạn là false cho khoảng thời gian của sự tải. Điều này ngắt bỏ những sự ràng buộc bắt buộc. Nhớ gán EnforceConstraints trở về giá trị mặc định true ở cuối quá trình tải.

Những chủ điểm trong chương này:

- Lớp UniqueConstraint
- [Tạo ra một đối tượng UniqueConstraint](#)
- Lớp ForeignKeyConstraint
- [Tạo ra một đối tượng ForeignKeyConstraint](#)
- Lớp DataRelation
- Tạo và sử dụng một đối tượng DataRelation

- Thêm, cập nhật, và xóa những hàng liên quan
- Những vấn đề khi cập nhật khóa chính của một hàng cha
- Mạng XML
- Định nghĩa một môi quan hệ sử dụng Visual Studio .NET

### LỚP *UniqueConstraint*:

Bạn sử dụng một đối tượng của lớp *UniqueConstraint* để bảo đảm là giá trị một *DataColumn* , hay sự kết hợp của những giá trị *DataColumn*, là duy nhất cho mỗi *DataRow* ở một *DataTable*. Lớp *UniqueConstraint* được bắt nguồn từ lớp *System.Data.Constraint* . [Bảng 12.1 cho thấy những thuộc tính \*UniqueConstraint\*.](#)

**Bảng 12.1: những thuộc tính *UniqueConstraint***

Thuộc tính	kiểu dữ liệu	Mô tả
Columns	<i>DataColumn</i> []	Lấy mảng của những đối tượng <i>DataColumn</i> cho <i>UniqueConstraint</i> .
ConstraintName	string	Lấy tên của <i>UniqueConstraint</i> .
ExtendedProperties	<i>PropertyCollection</i>	Lấy đối tượng <i>PropertyCollection</i> mà bạn có thể sử dụng để cất giữ những chuỗi của thông tin bổ sung.
IsPrimaryKey	bool	Lấy một giá trị bool cho biết liệu có phải <i>UniqueConstraint</i> là một khóa chính.
Table	<i>DataTable</i>	Lấy <i>DataTable</i> trong đó <i>UniqueConstraint</i> được tạo ra.

### TAO MỘT ĐỐI TƯỢNG *UniqueConstraint*:

Trong mục này, bạn sẽ học cách tạo ra một đối tượng *UniqueConstraint* như thế nào. Bộ khởi tạo *UniqueConstraint* được tải như sau:

```

UniqueConstraint(DataColumn myDataColumn)
UniqueConstraint(DataColumn[] myDataColumns)
UniqueConstraint(DataColumn myDataColumn, bool isPrimaryKey)
UniqueConstraint(DataColumn[] myDataColumns, bool isPrimaryKey)
UniqueConstraint(string constraintName, DataColumn myDataColumn)
UniqueConstraint(string constraintName, DataColumn[] myDataColumns)
UniqueConstraint(string constraintName, DataColumn myDataColumn, bool isPrimaryKey)
UniqueConstraint(string constraintName, DataColumn[] myDataColumns,
    bool isPrimaryKey)
UniqueConstraint(string constraintName, string[] columnNames,
    bool isPrimaryKey)

```

với:

**myDataColumn** và **myDataColumns** là những đối tượng *DataColumn* mà bạn muốn bảo đảm duy nhất.

**constraintName** là tên bạn muốn gán tới thuộc tính *ConstraintName* của *UniqueConstraint* của bạn.

**isPrimaryKey** cho biết liệu có phải UniqueConstraint của bạn thuộc một khóa chính.

Trước khi tạo ra và thêm một UniqueConstraint tới một DataTable, đầu tiên Bạn cần một DataTable. Ví dụ sau đây tạo ra và cư trú hai đối tượng DataTable có tên customersDT và ordersDT (Đối tượng ordersDT sẽ được sử dụng sau đó trong mục "[Tạo ra một đối tượng ForeignKeyConstraint](#)"):

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT CustomerID, CompanyName " +
    "FROM Customers " +
    "WHERE CustomerID = 'ALFKI'" +
    "SELECT OrderID, CustomerID " +
    "FROM Orders " +
    "WHERE CustomerID = 'ALFKI'";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Customers";
myDataSet.Tables["Table1"].TableName = "Orders";
DataTable customersDT = myDataSet.Tables["Customers"];
DataTable ordersDT = myDataSet.Tables["Orders"];
```

Ví dụ sau đây tạo ra một đối tượng UniqueConstraint trên DataColumn CustomerID của customersDT DataTable, Và sau đó thêm UniqueConstraint này vào customersDT; chú ý tham số thứ ba tới Bộ khởi dựng UniqueConstraint được gán là true, cho biết sự ràng buộc là cho một khóa chính:

```
UniqueConstraint myUC =
    new UniqueConstraint(
        "UniqueConstraintCustomerID",
        customersDT.Columns["CustomerID"],
        true
    );
customersDT.Constraints.Add(myUC);
```

### **Ghi chú**

Để thêm một cách thành công một UniqueConstraint vào DataColumn của một DataTable, giá trị DataColumn trong mỗi đối tượng DataRow trong DataTable phải là duy nhất.

Ví dụ cuối cùng truy xuất sự ràng buộc vừa thêm vào customerDT và hiển thị những thuộc tính của nó:

```
myUC =
    (UniqueConstraint) customersDT.Constraints["UniqueConstraintCustomerID"];
Console.WriteLine("Columns:");
foreach (DataColumn myDataColumn in myUC.Columns)
{
    Console.WriteLine("'" + myDataColumn);
}
Console.WriteLine("myUC.ConstraintName = " + myUC.ConstraintName);
```



```
Console.WriteLine("myUC.IsPrimaryKey = " + myUC.IsPrimaryKey);
Console.WriteLine("myUC.Table = " + myUC.Table);
```

Ví dụ này trình bày đầu ra sau đây

```
Columns:
  CustomerID
myUC.ConstraintName = UniqueConstraintCustomerID
myUC.IsPrimaryKey = True
myUC.Table = Customers
```

Thuộc tính IsPrimaryKey là true bởi vì sự ràng buộc thuộc một khóa chính; điều này được chỉ rõ trong bộ khởi tạo khi sự ràng buộc được tạo ra trước đó.

## **LỚP Foreignkey Constraint:**

Bạn sử dụng một đối tượng của lớp ForeignKeyConstraint để đại diện cho một sự ràng buộc khóa ngoại giữa hai đối tượng DataTable. Điều này bảo đảm rằng mỗi DataRow trong DataTable con có một DataRow thích ứng trong DataTable cha của bạn. Lớp ForeignKeyConstraint được bắt nguồn từ lớp System.Data.Constraint. [Bảng 12.2 cho thấy những thuộc tính ForeignKeyConstraint.](#)

**Bảng 12.2: những thuộc tính ForeignKeyConstraint**

Thuộc tính	Kiểu dữ liệu	Mô tả
AcceptRejectRule	AcceptRejectRule	Lấy hay gán AcceptRejectRule cho biết hoạt động sẽ xảy ra khi phương thức AcceptChanges() của DataTable được gọi. Những thành viên của Sự liệt kê System.Data.AcceptRejectRule là: <b>Cascade:</b> cho biết những sự thay đổi tới những đối tượng DataRow trong DataTable cha cũng được làm trong DataTable con <b>None :</b> cho biết không có hoạt động nào xảy ra. Mặc định là None.
Columns	DataColumn[]	Lấy mảng của những đối tượng DataColumn từ DataTable con.
ConstraintName	string	Lấy tên của đối tượng UniqueConstraint.
DeleteRule	Rule	Lấy hay gán quy tắc cho biết hoạt động sẽ xảy ra khi một DataRow trong DataTable cha bị xóa. Những thành viên của sự liệt kê System.Data.Rule là: ▪ <b>Cascade</b> , cho biết sự xóa hay cập nhật tới những đối tượng DataRow trong DataTable cha cũng được thực hiện trong DataTable con. ▪ <b>None:</b> cho biết không có hoạt động nào xảy ra. ▪ <b>SetDefault:</b> cho biết rằng những giá trị DataColumn trong DataTable con sẽ được thiết lập tới giá trị trong thuộc tính DefaultValue của DataColumn. ▪ <b>SetNull:</b> cho biết những giá trị DataColumn trong DataTable con sẽ được thiết lập tới DBNull.

		giá trị mặc định là Cascade.
ExtendedProperties	PropertyCollection	Lấy đối tượng PropertyCollection mà bạn có thể sử dụng để lưu trữ những chuỗi của thông tin bổ sung.
RelatedColumns	DataColumn[]	Lấy mảng của những đối tượng DataColumn trong DataTable cha cho UniqueConstraint.
RelatedTable	DataTable	Lấy DataTable cha DataTable cho UniqueConstraint.
Table	DataTable	Lấy DataTable con mà UniqueConstraint thuộc về.
UpdateRule	Rule	Lấy hay gán quy tắc cho biết hoạt động sẽ xảy ra khi một DataRow trong DataTable cha được cập nhật. Xem thuộc tính DeleteRule về những thành viên của sự liệt kê quy tắc (Rule enumeration) . mặc định là Cascade.

### **TAO MỘT ĐỐI TƯỢNG *ForeignKeyConstraint*:**

Bộ khởi dựng ForeignKeyConstraint được quá tải như sau:

```

ForeignKeyConstraint(DataColumn parentDataColumn, DataColumn childDataColumn)
ForeignKeyConstraint(DataColumn[] parentDataColumns,
DataColumn[] childDataColumns)
ForeignKeyConstraint(string constraintName, DataColumn parentDataColumn,
DataColumn childDataColumn)
ForeignKeyConstraint(string constraintName,
DataColumn[] parentDataColumns, DataColumn[] childDataColumns)
ForeignKeyConstraint(string constraintName, string parentDataTableName,
string[] parentDataColumnNames, string[] childDataColumnNames,
AcceptRejectRule acceptRejectRule, Rule deleteRule, Rule updateRule)

```

Với :

- . **parentDataColumn** và **parentDataColumns** là những đối tượng DataColumn trong DataTable cha .
- . **childDataColumn** và **childDataColumns** là những đối tượng DataColumn trong DataTable con .
- . **constraintName** là tên bạn muốn gán tới thuộc tính ConstraintName của ForeignKeyConstraint của các bạn.
- . **parentDataTableName** là tên của DataTable cha.
- . **parentDataColumnNames** và **childDataColumnNames** chứa tên của những đối tượng DataColumn trong những đối tượng DataTable cha và con.
- . **acceptRejectRule**, **deleteRule**, và **updateRule** là những quy tắc cho ForeignKey- Constraint.

Trước đó trong mục "[Tạo ra một Đối tượng UniqueConstraint](#)," bạn đã thấy một ví dụ mã tạo ra hai đối tượng DataTable có tên customersDT và ordersDT. Ví dụ sau đây tạo ra một đối tượng ForeignKeyConstraint trên DataColumn CustomerID của ordersDT tới DataColumn CustomerID của customersDT:

```

ForeignKeyConstraint myFKC =
new ForeignKeyConstraint(
"ForeignKeyConstraintCustomersOrders",

```

```

        customersDT.Columns["CustomerID"],
        ordersDT.Columns["CustomerID"]
    );
    ordersDT.Constraints.Add(myFKC);

```

**Chú ý:** ForeignKeyConstraint được thêm vào ordersDT sử dụng phương thức Add().

#### **Ghi chú:**

Để thêm một cách thành công một ForeignKeyConstraint vào một DataTable, mỗi giá trị DataColumn trong DataTable con phải có một giá trị DataColumn thích ứng trong DataTable cha.

ví dụ kế tiếp truy xuất sự ràng buộc vừa thêm vào ordersDT và trình bày những thuộc tính của nó :

```

myFKC =
    (ForeignKeyConstraint)
        ordersDT.Constraints["ForeignKeyConstraintCustomersOrders"];
Console.WriteLine("myFKC.AcceptRejectRule = " + myFKC.AcceptRejectRule);
Console.WriteLine("Columns:");
foreach (DataColumn myDataColumn in myFKC.Columns)
{
    Console.WriteLine("'" + myDataColumn);
}
Console.WriteLine("myFKC.ConstraintName = " + myFKC.ConstraintName);
Console.WriteLine("myFKC.DeleteRule = " + myFKC.DeleteRule);
Console.WriteLine("RelatedColumns:");
foreach (DataColumn relatedDataColumn in myFKC.RelatedColumns)
{
    Console.WriteLine("'" + relatedDataColumn);
}
Console.WriteLine("myFKC.RelatedTable = " + myFKC.RelatedTable);
Console.WriteLine("myFKC.Table = " + myFKC.Table);
Console.WriteLine("myFKC.UpdateRule = " + myFKC.UpdateRule);

```

Ví dụ này trình bày đầu ra sau đây:

```

myFKC.AcceptRejectRule = None
Columns:
    CustomerID
myFKC.ConstraintName = ForeignKeyConstraintCustomersOrders
myFKC.DeleteRule = Cascade
RelatedColumns:
    CustomerID
myFKC.RelatedTable = Customers
myFKC.Table = Orders
myFKC.UpdateRule = Cascade

```

#### **Ghi chú**

Bạn sẽ tìm thấy tất cả mã những ví dụ trình bày trong mục này và [mục trước đây](#), "Tạo ra một đối tượng UniqueConstraint, " trong chương trình AddConstraints.cs . chương trình này bị bỏ qua trong sách này cho ngắn gọn.

### **LỚP DataRelation:**

Bạn sử dụng một đối tượng của lớp `DataRelation` để đại diện cho một mối quan hệ giữa hai đối tượng `DataTable`. Bạn sử dụng một đối tượng `DataRelation` để mô hình những mối quan hệ cha - con giữa hai bảng cơ sở dữ liệu. Theo mặc định, khi bạn tạo ra một `DataRelation`- một `UniqueConstraint` và `ForeignKeyConstraint` được tự động thêm vào những đối tượng `DataTable` cha và con của các bạn . [Bảng 12.3 cho thấy những thuộc tính `DataRelation`.](#)

**Bảng 12.3: những thuộc tính `DataRelation`**

Thuộc tính	Kiểu dữ liệu	Mô tả
<code>ChildColumns</code>	<code> DataColumn[] </code>	Lấy mảng của những đối tượng <code>DataColumn</code> con.
<code>ChildKeyConstraint</code>	<code>ForeignKeyConstraint</code>	Gets the <code>ForeignKeyConstraint</code> object for the <code>DataRelation</code> . lấy đối tượng <code>ForeignKeyConstraint</code> cho <code>DataRelation</code> .
<code>ChildTable</code>	<code>DataTable</code>	Lấy đối tượng <code>DataTable</code> con.
<code>DataSet</code>	<code>DataSet</code>	Lấy <code>Dataset</code> mà <code>DataRelation</code> thuộc về.
<code>ExtendedProperties</code>	<code>PropertyCollection</code>	Lấy đối tượng <code>PropertyCollection</code> mà bạn có thể dùng để cất giữ những chuỗi của thông tin bổ sung.
<code>Nested</code>	<code>bool</code>	Lấy hay gán một giá trị <code>bool</code> cho biết liệu có phải những đối tượng <code>DataRelation</code> được lồng vào.Điều này hữu ích khi định nghĩa những mối quan hệ có thứ bậc trong XML. Mặc định là <code>false</code> .
<code>ParentColumns</code>	<code> DataColumn[] </code>	Lấy mảng của những đối tượng <code>DataColumn</code> cha .
<code>ParentKeyConstraint</code>	<code>UniqueConstraint</code>	Lấy đối tượng <code>UniqueConstraint</code> mà bảo đảm rằng những giá trị <code>DataColumn</code> trong <code>DataTable</code> cha là duy nhất.
<code>ParentTable</code>	<code>DataTable</code>	Lấy đối tượng <code>DataTable</code> cha .
<code>RelationName</code>	<code>string</code>	Lấy tên của đối tượng <code>DataRelation</code> .

### **TAO VÀ SỬ DỤNG MỘT ĐỐI TƯỢNG `DataRelation`:**

Trong mục này, bạn sẽ học cách tạo ra một đối tượng `DataRelation` để định nghĩa một mối quan hệ giữa hai đối tượng `DataTable` mà giữ một số hàng từ bảng `Customers` và `Orders` . Như bạn biết, cột `CustomerID` của bảng `Orders` con là một khóa ngoại liên kết tới cột `CustomerID` của bảng `Customers` cha.

Một khi bạn đã tạo ra một `DataRelation`, bạn có thể sử dụng phương thức `GetChildRows()` của một đối tượng `DataRow` trong `DataTable` cha để thu được những đối tượng `DataRow` tương ứng từ `DataTable` con . Với từ " tương ứng ", Tôi định nghĩa những hàng phù hợp với những giá trị trong những đối tượng `DataColumn` khóa ngoại. Bạn có thể cũng sử dụng phương thức `GetParentRow()` của một `DataRow` trong `DataTable` con để thu được `DataRow` tương ứng trong `DataTable` cha .

Trước khi tạo ra và thêm một `DataRelation` vào một `Dataset`, đầu tiên bạn cần một `Dataset`. Ví dụ theo sau tạo ra và cư trú một `Dataset` với hai đối tượng `DataTable` đặt tên `customersDT` và `ordersDT`; chú ý hai hàng đầu tiên từ bảng `Customers` cùng với những hàng tương ứng từ bảng `Orders` được truy xuất:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT TOP 2 CustomerID, CompanyName " +
    "FROM Customers " +
```

```

"ORDER BY CustomerID;" +
"SELECT OrderID, CustomerID " +
"FROM Orders " +
"WHERE CustomerID IN (" +
" SELECT TOP 2 CustomerID " +
" FROM Customers " +
" ORDER BY CustomerID" +
");
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Customers";
myDataSet.Tables["Table1"].TableName = "Orders";
DataTable customersDT = myDataSet.Tables["Customers"];
DataTable ordersDT = myDataSet.Tables["Orders"];

```

Bạn sẽ thấy cách tạo ra một `DataRelation` định nghĩa một mối quan hệ giữa những đối tượng `DataTable`: `customersDT` và `ordersDT` như thế nào .

### Ghi chú:

Bạn sẽ tìm thấy tất cả những mã ví dụ trình bày trong mục này trong chương trình `CreateDataRelation.cs` .

### **TAO RA MỘT *DataRelation*:**

Bộ khởi dựng `DataRelation` được quá tải như sau:

```

DataRelation(string dataRelationName,
    DataColumn parentDataColumn, DataColumn childDataColumn)
DataRelation(string dataRelationName,
    DataColumn[] parentDataColumns, DataColumn[] childDataColumns)
DataRelation(string dataRelationName,
    DataColumn parentDataColumn, DataColumn childDataColumn, bool createConstraints)
DataRelation(string dataRelationName,
    DataColumn[] parentDataColumns, DataColumn[] childDataColumns,
    bool createConstraints)
DataRelation(string dataRelationName,
    string parentDataTableName, string childDataTableName,
    string[] parentDataColumnNames, string[] childDataColumnNames,
    bool nested)

```

### Với :

- . **dataRelationName** là tên bạn muốn gán tới thuộc tính `RelationName` của `DataRelation` của bạn.
- . **parentDataColumn** và đối tượng **DataColumn** **parentDataColumns**are trong `DataTable` cha .
- . **childDataColumn** và **childDataColumns** là những đối tượng `DataColumn` trong `DataTable` con .
- . **createConstraints** cho biết liệu bạn có muốn một `UniqueConstraint` được thêm vào `DataTable` cha và một `ForeignKeyConstraint` được thêm vào `DataTable` con một cách tự động (mặc định là `true`)
- . **parentDataTableName** và **childDataTableName** là những tên của những đối tượng `DataTable` cha và con.
- . **parentDataColumnNames** và **childDataColumnNames** chứa tên của những đối tượng `DataColumn` trong những đối tượng `DataTable` cha và con .

- **nested** : cho biết liệu có phải những mối quan hệ được lồng vào.

Ví dụ sau đây tạo ra một đối tượng `DataRelation` có tên `customersOrdersDataRel`:

```
DataRelation customersOrdersDataRel =  
    new DataRelation(  
        "CustomersOrders",  
        customersDT.Columns["CustomerID"],  
        ordersDT.Columns["CustomerID"]  
    );
```

Tên được gán tới thuộc tính `RelationName` của `customersOrdersDataRel` là `CustomersOrders`, `DataColumn` cha là `customersDT.Columns[ " CustomerID " ]`, và `DataColumn` con là `ordersDT.Columns[ " CustomerID " ]`.

Tiếp theo, `customersOrdersDataRel` phải được thêm vào `myDataSet`. Bạn truy nhập những đối tượng `DataRelation` trong một đối tượng `Dataset` thông qua thuộc tính `Relationships` của nó. Thuộc tính `Relationships` trả lại một đối tượng của lớp `DataRelationCollection`, là một tập hợp của những đối tượng `DataRelation`. Để thêm một đối tượng `DataRelation` vào đối tượng `DataRelationCollection` của một `Dataset`, bạn gọi phương thức `Add()` thông qua thuộc tính `Relationships` của `Dataset` của bạn.

Ví dụ sau đây sử dụng phương thức `Add()` để thêm `customersOrdersDataRel` tới `myDataSet`:

```
myDataSet.Relations.Add(  
    customersOrdersDataRel  
);
```

Phương thức `Add()` được quá tải, và bạn cũng có thể sử dụng phiên bản sau của phương thức `Add()` để tạo ra và thêm một đối tượng `DataRelation` tới `myDataSet`:

```
myDataSet.Relations.Add(  
    "CustomersOrders",  
    customersDT.Columns["CustomerID"],  
    ordersDT.Columns["CustomerID"]  
);
```

Ví dụ này thực hiện cùng công việc như với hai ví dụ ví dụ trước. Tham số đầu tiên tới phương thức `Add()` là một chuỗi chứa tên bạn muốn gán tới thuộc tính `RelationName` của `DataRelation`. Những tham số thứ hai và ba của mối quan hệ là những đối tượng `DataColumn` từ những đối tượng `DataTable` cha và con.

### **KHẢO SÁT NHỮNG SỰ RÀNG BUỘC TẠO RA BỞI `DataRelation`:**

Theo mặc định, khi bạn tạo ra một `DataRelation`, một `UniqueConstraint` và `ForeignKeyConstraint` được tự động thêm vào những đối tượng `DataTable` cha và con của bạn. Bạn có thể lấy `UniqueConstraint` từ một `DataRelation` sử dụng thuộc tính `ParentKeyConstraint` của nó. Chẳng hạn:

```
UniqueConstraint myUC =  
    customersOrdersDataRel.ParentKeyConstraint;
```

Bạn có thể xem những thuộc tính của đối tượng `myUC UniqueConstraint` sử dụng mã sau đây:

```

Console.WriteLine("Columns:");
foreach (DataColumn myDataColumn in myUC.Columns)
{
    Console.WriteLine("'" + myDataColumn);
}
Console.WriteLine("myUC.ConstraintName = " + myUC.ConstraintName);
Console.WriteLine("myUC.IsPrimaryKey = " + myUC.IsPrimaryKey);
Console.WriteLine("myUC.Table = " + myUC.Table);

```

Đầu ra từ mã này như sau:

```

Columns:
    CustomerID
myUC.ConstraintName = Constraint1
myUC.IsPrimaryKey = False
myUC.Table = Customers

```

Bạn có thể lấy ForeignKeyConstraint từ một DataRelation sử dụng thuộc tính ChildKeyConstraint của nó. Chẳng hạn:

```

ForeignKeyConstraint myFKC =
    customersOrdersDataRel.ChildKeyConstraint;

```

Bạn có thể xem những thuộc tính của myFKC sử dụng mã sau đây :

```

Console.WriteLine("myFKC.AcceptRejectRule = " + myFKC.AcceptRejectRule);
Console.WriteLine("Columns:");
foreach (DataColumn myDataColumn in myFKC.Columns)
{
    Console.WriteLine("'" + myDataColumn);
}
Console.WriteLine("myFKC.ConstraintName = " + myFKC.ConstraintName);
Console.WriteLine("myFKC.DeleteRule = " + myFKC.DeleteRule);
Console.WriteLine("RelatedColumns:");
foreach (DataColumn relatedDataColumn in myFKC.RelatedColumns)
{
    Console.WriteLine("'" + relatedDataColumn);
}
Console.WriteLine("myFKC.RelatedTable = " + myFKC.RelatedTable);
Console.WriteLine("myFKC.Table = " + myFKC.Table);
Console.WriteLine("myFKC.UpdateRule = " + myFKC.UpdateRule);

```

Đầu ra từ mã này như sau:

```

myFKC.AcceptRejectRule = None
Columns:
    CustomerID
myFKC.ConstraintName = CustomersOrders
myFKC.DeleteRule = Cascade
RelatedColumns:
    CustomerID
myFKC.RelatedTable = Customers
myFKC.Table = Orders
myFKC.UpdateRule = Cascade

```



Thuộc tính DeleteRule và UpdateRule được gán tới Cascade theo mặc định. Vì DeleteRule được gán tới Cascade, khi bạn xóa một DataRow trong DataTable cha, thì bất kỳ đối tượng DataRow tương ứng nào trong DataTable con đều cũng bị xóa. Vì UpdateRule được gán tới Cascade, nên khi bạn thay đổi DataColumn trong DataTable cha trong đó ForeignKeyConstraint được tạo ra, thì sự thay đổi giống như vậy cũng được thực hiện trong bất kỳ đối tượng DataRow tương ứng nào trong DataTable con. Bạn sẽ học nhiều hơn về điều này sau đây trong mục " Vấn đề xảy ra khi cập nhật khóa chính của một hàng cha"

## **ĐỊNH HƯỚNG NHỮNG ĐỐI TƯỢNG DataRow TRONG NHỮNG ĐỐI TƯỢNG DataTable CHA VÀ CON**

Để định hướng những đối tượng DataRow trong những đối tượng DataTable liên quan, bạn sử dụng những phương thức GetChildRows() và GetParentRows() của một DataRow.

### **SỬ DỤNG PHƯƠNG THỨC GetChildRow():**

Bạn sử dụng phương thức GetChildRows() để lấy những đối tượng DataRow con liên quan từ DataRow cha. Chẳng hạn, mã sau đây trình bày những đối tượng DataRow cha từ DataTable customersDT và những đối tượng DataRow con liên quan của chúng từ DataTable ordersDT:

```
foreach (DataRow customerDR in customersDT.Rows)
{
    Console.WriteLine("\nCustomerID = " + customerDR["CustomerID"]);
    Console.WriteLine("CompanyName = " + customerDR["CompanyName"]);

    DataRow[] ordersDRs = customerDR.GetChildRows("CustomersOrders");
    Console.WriteLine("This customer placed the following orders:");
    foreach (DataRow orderDR in ordersDRs)
    {
        Console.WriteLine("OrderID = " + orderDR["OrderID"]);
    }
}
```

Đầu ra từ mã này như sau:

```
CustomerID = ALFKI
CompanyName = Alfreds Futterkiste
This customer placed the following orders:
OrderID = 10643
OrderID = 10692
OrderID = 10702
OrderID = 10835
OrderID = 10952
OrderID = 11011

CustomerID = ANATR
CompanyName = Ana Trujillo Emparedados y helados
This customer placed the following orders:
OrderID = 10308
OrderID = 10625
OrderID = 10759
OrderID = 10926
```

## **SỬ DỤNG PHƯƠNG THỨC *GetParentRow()*:**

Bạn sử dụng phương thức `GetParentRow()` để lấy `DataRow` cha từ `DataRow` con. Chẳng hạn, mã sau đây hiển thị `DataRow` con đầu tiên từ `ordersDT` và `DataRow` cha liên quan của nó từ `customersDT`:

```
DataRow parentCustomerDR = ordersDT.Rows[0].GetParentRow("CustomersOrders");
Console.WriteLine("\nOrder with OrderID of " + ordersDT.Rows[0]["OrderID"] +
    " was placed by the following customer:");
Console.WriteLine("CustomerID = " + parentCustomerDR["CustomerID"]);
```

Đầu ra từ mã này như sau:

```
Order with OrderID of 10643 was placed by the following customer:
CustomerID = ALFKI
```

## **THÊM, CẬP NHẬT VÀ XÓA NHỮNG HÀNG LIÊN QUAN:**

Trong mục này, bạn sẽ học cách thực hiện những thay đổi trong những đối tượng `DataTable` cất giữ những hàng từ những bảng `Customers` và `Orders`. Những bảng này có quan hệ thông qua khóa ngoại `CustomerID`. Như bạn sẽ thấy, bạn phải đẩy những thay đổi tới cơ sở dữ liệu nằm bên dưới theo một trật tự chỉ định. Nếu bạn không thực hiện, chương trình của bạn sẽ phát sinh một ngoại lệ.

### **Ghi chú:**

Bạn sẽ tìm thấy tất cả mã những ví dụ trình bày trong mục này trong chương trình `ModifyingRelatedData.cs`.

## **THIẾT LẬP NHỮNG ĐỐI TƯỢNG *DataAdapter*:**

Bạn sẽ cần hai đối tượng `DataAdapter`:

- . Một để làm việc với bảng `Customers`, mà sẽ được đặt tên `customersDA`.
- . Một để làm việc với bảng `Orders`, mà sẽ được đặt tên `ordersDA`.

Chúng ta hãy xem xét việc thiết lập hai đối tượng `DataAdapter` này.

## **THIẾT LẬP *DataAdapter* `customersDA`:**

Mã sau đây tạo ra và thiết lập một `DataAdapter` có tên `customersDA` chứa phát biểu cần thiết `SELECT`, `INSERT`, `UPDATE`, và `DELETE` để truy nhập bảng `Customers`:

```
SqlDataAdapter customersDA = new SqlDataAdapter();

// create a SqlCommand object to hold the SELECT
SqlCommand customersSelectCommand = mySqlConnection.CreateCommand();
customersSelectCommand.CommandText =
    "SELECT CustomerID, CompanyName " +
    "FROM Customers";

// create a SqlCommand object to hold the INSERT
SqlCommand customersInsertCommand = mySqlConnection.CreateCommand();
customersInsertCommand.CommandText =
    "INSERT INTO Customers (" +
    " CustomerID, CompanyName " +
    ") VALUES (" +
```

```

" @CustomerID, @CompanyName" +
    ");";
customersInsertCommand.Parameters.Add("@CustomerID", SqlDbType.NChar,
    5, "CustomerID");
customersInsertCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar,
    40, "CompanyName");

// create a SqlCommand object to hold the UPDATE
SqlCommand customersUpdateCommand = mySqlConnection.CreateCommand();
customersUpdateCommand.CommandText =
    "UPDATE Customers " +
    "SET " +
    " CompanyName = @NewCompanyName " +
    "WHERE CustomerID = @OldCustomerID " +
    "AND CompanyName = @OldCompanyName";
customersUpdateCommand.Parameters.Add("@NewCompanyName",
    SqlDbType.NVarChar, 40, "CompanyName");
customersUpdateCommand.Parameters.Add("@OldCustomerID",
    SqlDbType.NChar, 5, "CustomerID");
customersUpdateCommand.Parameters.Add("@OldCompanyName",
    SqlDbType.NVarChar, 40, "CompanyName");
customersUpdateCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
customersUpdateCommand.Parameters["@OldCompanyName"].SourceVersion =
    DataRowVersion.Original;

// create a SqlCommand object to hold the DELETE
SqlCommand customersDeleteCommand = mySqlConnection.CreateCommand();
customersDeleteCommand.CommandText =
    "DELETE FROM Customers " +
    "WHERE CustomerID = @OldCustomerID " +
    "AND CompanyName = @OldCompanyName";
customersDeleteCommand.Parameters.Add("@OldCustomerID",
    SqlDbType.NChar, 5, "CustomerID");
customersDeleteCommand.Parameters.Add("@OldCompanyName",
    SqlDbType.NVarChar, 40, "CompanyName");
customersDeleteCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
customersDeleteCommand.Parameters["@OldCompanyName"].SourceVersion =
    DataRowVersion.Original;

// set the customersDA properties
// to the SqlCommand objects previously created
customersDA.SelectCommand = customersSelectCommand;
customersDA.InsertCommand = customersInsertCommand;
customersDA.UpdateCommand = customersUpdateCommand;
customersDA.DeleteCommand = customersDeleteCommand;

```

Chú ý rằng phát biểu UPDATE chỉ sửa đổi giá trị cột CompanyName; nó không sửa đổi giá trị cột khóa chính CustomerID. Bạn sẽ học về những vấn đề liên quan với việc cập nhật một giá trị cột khóa chính sau trong mục " Những vấn đề liên quan khi cập nhật khóa chính của một hàng cha ."

Chương trình ModifyingRelatedData.cs chứa một phương thức có tên SetupCustomersDA() thực hiện mã trước đây.

## **THIẾT LẬP *DataAdapter ordersDA*:**

Mã sau đây tạo ra và thiết lập một đối tượng `DataAdapter` có tên `ordersDA` chứa những phát biểu cần thiết `SELECT`, `INSERT`, `UPDATE`, và `DELETE` để truy nhập bảng `Orders`; chú ý `ordersInsertCommand` chứa cả hai gồm một phát biểu `INSERT` và một phát biểu `SELECT` để truy xuất cột `OrderID` mới, cột này là một cột khóa chính có một giá trị được tự động sinh ra bởi cơ sở dữ liệu

```
SqlDataAdapter ordersDA = new SqlDataAdapter();

// create a SqlCommand object to hold the SELECT
SqlCommand ordersSelectCommand = mySqlConnection.CreateCommand();
ordersSelectCommand.CommandText =
    "SELECT OrderID, CustomerID, ShipCountry " +
    "FROM Orders";

// create a SqlCommand object to hold the INSERT
SqlCommand ordersInsertCommand = mySqlConnection.CreateCommand();
ordersInsertCommand.CommandText =
    "INSERT INTO Orders (" +
    " CustomerID, ShipCountry " +
    ") VALUES (" +
    " @CustomerID, @ShipCountry" +
    ");" +
    "SELECT @OrderID = SCOPE_IDENTITY();";
ordersInsertCommand.Parameters.Add("@CustomerID", SqlDbType.NChar,
    5, "CustomerID");
ordersInsertCommand.Parameters.Add("@ShipCountry", SqlDbType.NVarChar,
    15, "ShipCountry");
ordersInsertCommand.Parameters.Add("@OrderID", SqlDbType.Int,
    0, "OrderID");
ordersInsertCommand.Parameters["@OrderID"].Direction =
    ParameterDirection.Output;

// create a SqlCommand object to hold the UPDATE
SqlCommand ordersUpdateCommand = mySqlConnection.CreateCommand();
ordersUpdateCommand.CommandText =
    "UPDATE Orders " +
    "SET " +
    " ShipCountry = @NewShipCountry " +
    "WHERE OrderID = @OldOrderID " +
    "AND CustomerID = @OldCustomerID " +
    "AND ShipCountry = @OldShipCountry";
ordersUpdateCommand.Parameters.Add("@NewShipCountry",
    SqlDbType.NVarChar, 15, "ShipCountry");
ordersUpdateCommand.Parameters.Add("@OldOrderID",
    SqlDbType.Int, 0, "OrderID");
ordersUpdateCommand.Parameters.Add("@OldCustomerID",
    SqlDbType.NChar, 5, "CustomerID");
ordersUpdateCommand.Parameters.Add("@OldShipCountry",
    SqlDbType.NVarChar, 15, "ShipCountry");
ordersUpdateCommand.Parameters["@OldOrderID"].SourceVersion =
    DataRowVersion.Original;
ordersUpdateCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
ordersUpdateCommand.Parameters["@OldShipCountry"].SourceVersion =
```

```

DataRowVersion.Original;

// create a SqlCommand object to hold the DELETE
SqlCommand ordersDeleteCommand = mySqlConnection.CreateCommand();
ordersDeleteCommand.CommandText =
    "DELETE FROM Orders " +
    "WHERE OrderID = @OldOrderID " +
    "AND CustomerID = @OldCustomerID " +
    "AND ShipCountry = @OldShipCountry";
ordersDeleteCommand.Parameters.Add("@OldOrderID", SqlDbType.Int,
    0, "OrderID");
ordersDeleteCommand.Parameters.Add("@OldCustomerID",
    SqlDbType.NChar, 5, "CustomerID");
ordersDeleteCommand.Parameters.Add("@OldShipCountry",
    SqlDbType.NVarChar, 15, "ShipCountry");
ordersDeleteCommand.Parameters["@OldOrderID"].SourceVersion =
    DataRowVersion.Original;
ordersDeleteCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;
ordersDeleteCommand.Parameters["@OldShipCountry"].SourceVersion =
    DataRowVersion.Original;

// set the ordersDA properties
// to the SqlCommand objects previously created
ordersDA.SelectCommand = ordersSelectCommand;
ordersDA.InsertCommand = ordersInsertCommand;
ordersDA.UpdateCommand = ordersUpdateCommand;
ordersDA.DeleteCommand = ordersDeleteCommand;

```

Chương trình ModifyingRelatedData.cs chứa một phương thức có tên SetupOrdersDA() thực hiện mã trước đây.

### **TAO VÀ CƯ TRÚ MỘT Dataset:**

Tiếp theo, ví dụ sau tạo ra và cư trú một Dataset đặt tên myDataSet với những hàng từ những bảng Customers và Orders sử dụng customersDA Và ordersDA:

```

DataSet myDataSet = new DataSet();
mySqlConnection.Open();
customersDA.Fill(myDataSet, "Customers");
ordersDA.Fill(myDataSet, "Orders");
mySqlConnection.Close();
DataTable customersDT = myDataSet.Tables["Customers"];
DataTable ordersDT = myDataSet.Tables["Orders"];

```

Chú ý những đối tượng DataTable có tên là customersDT và ordersDT .

Những ví dụ sau đây gán những thuộc tính PrimaryKey của customersDT Và ordersDT:

```

customersDT.PrimaryKey =
    new DataColumn[]
    {
        customersDT.Columns["CustomerID"]
    }

```

```

};

ordersDT.PrimaryKey =
    new DataColumn[]
    {
        ordersDT.Columns["OrderID"]
    };

```

Ví dụ sau đây thiết lập DataColumn OrderID của ordersDT như một khóa chính:

```

ordersDT.Columns["OrderID"].AllowDBNull = false;
ordersDT.Columns["OrderID"].AutoIncrement = true;
ordersDT.Columns["OrderID"].AutoIncrementSeed = -1;
ordersDT.Columns["OrderID"].AutoIncrementStep = -1;
ordersDT.Columns["OrderID"].ReadOnly = true;
ordersDT.Columns["OrderID"].Unique = true;

```

Ví dụ cuối cùng thêm một DataRelation vào myDataSet nó chỉ rõ một mối quan hệ giữa customersDT và ordersDT sử dụng CustomerID DataColumn:

```

DataRelation customersOrdersDataRel =
    new DataRelation(
        "CustomersOrders",
        customersDT.Columns["CustomerID"],
        ordersDT.Columns["CustomerID"]
    );
myDataSet.Relations.Add(
    customersOrdersDataRel
);

```

Chương trình ModifyingRelatedData.cs thực hiện mã trước đây trong phương thức Main().

### **THÊM NHỮNG ĐỐI TƯỢNG DataRow VÀO customersDT VÀ ordersDT:**

Ví dụ sau đây thêm một DataRow có tên customerDR vào customersDT; chú ý CustomerID được gán giá trị J6COM:

```

DataRow customerDR = customersDT.NewRow();
customerDR["CustomerID"] = "J6COM";
customerDR["CompanyName"] = "J6 Company";
customersDT.Rows.Add(customerDR);

```

Ví dụ kế tiếp thêm một DataRow có tên orderDR vào ordersDT; chú ý CustomerID cũng được gán giá trị là J6COM, cho biết đây là DataRow con thuộc DataRow trước đây trong customersDT:

```

DataRow orderDR = ordersDT.NewRow();
orderDR["CustomerID"] = "J6COM";
orderDR["ShipCountry"] = "England";
ordersDT.Rows.Add(orderDR);

```

Bởi vì DataColumn OrderID của ordersDT được thiết lập như một khóa chính, nó sẽ tự động được gán giá trị ban đầu là -1. Khi DataRow này được đẩy tới cơ sở dữ liệu, phát biểu SELECT trong ordersDA sẽ gán OrderID tới giá trị duy nhất được phát sinh bởi cơ sở dữ liệu cho hàng mới trong bảng Orders. Bạn sẽ thấy cách đẩy những sự thay đổi tới cơ sở dữ liệu không lâu nữa trong mục "[Đẩy những sự thay đổi ở customersDT và ordersDT tới Cơ sở dữ liệu.](#)"

Chương trình ModifyingRelatedData.cs thực hiện mã trước đây trong phương thức Main().

### **CẬP NHẬT NHỮNG ĐỐI TƯỢNG DataRow TỪ customersDT VÀ ordersDT:**

Ví dụ sau đây cập nhật CompanyName ở customerDR tới Widgets Inc.:

```
customerDR["CompanyName"] = "Widgets Inc.";
```

Ví dụ kế tiếp cập nhật ShipCountry ở orderDR tới USA

```
orderDR["ShipCountry"] = "USA";
```

Chương trình ModifyingRelatedData.cs thực hiện mã trước đây trong phương thức Main().

### **XÓA NHỮNG ĐỐI TƯỢNG DataRow TỪ customersDT VÀ ordersDT:**

Ví dụ sau đây xóa DataRow customerDR từ customersDT DataTable:

```
customerDR.Delete();
```

Trước đó trong mục "[Khảo sát những sự ràng buộc được tạo ra bởi DataRelation](#)," bạn thấy một ForeignKeyConstraint được thêm vào DataTable con theo mặc định khi một đối tượng DataRelation được thêm vào một Dataset. Bạn cũng nhìn thấy thuộc tính DeleteRule của đối tượng ForeignKeyConstraint được gán là Cascade theo mặc định. Điều này có nghĩa là khi DataRow trong DataTable cha bị xóa, do đó những đối tượng DataRow tương ứng trong DataTable con cũng bị xóa. Bởi vậy, trong ví dụ trước đây, khi customerDR bị xóa từ customersDT, cũng tương tự như orderDR trong ordersDT.

Chương trình ModifyingRelatedData.cs thực hiện mã trước đây trong phương thức Main().

### **ĐẨY NHỮNG THAY ĐỔI TRONG customersDT VÀ ordresDT TỚI CƠ SỞ DỮ LIỆU:**

Trong mục này, bạn sẽ học cách để đẩy những thay đổi đã làm trước đó trong những đối tượng DataTable : customersDT và ordersDT tới cơ sở dữ liệu. Khi đẩy những sự thay đổi tới cơ sở dữ liệu bạn phải áp dụng chúng trong một trật tự thỏa mãn những sự ràng buộc khóa ngoài trong những bảng liên quan.

Chẳng hạn, một hàng trong bảng Customers với một CustomerID là J6COM phải tồn tại trước khi một hàng với CustomerID đó có thể được thêm vào bảng Orders. Tương tự, bạn không thể xóa hàng có một CustomerID là J6COM trong khi có những hàng với CustomerID này trong bảng Orders. Cuối cùng, tất nhiên, bạn chỉ có thể cập nhật những hàng đã tồn tại trong một bảng.



Đi theo những bước này khi đẩy những sự thay đổi từ customersDT và ordersDT đến cơ sở dữ liệu:

1. Đẩy những đối tượng DataRow được thêm vào customersDT tới bảng Customers.
2. Đẩy những đối tượng DataRow được thêm vào ordersDT tới bảng Orders.
3. Đẩy những đối tượng DataRow được cập nhật ở customersDT tới bảng Customers.
4. Đẩy những đối tượng DataRow được cập nhật ở ordersDT tới bảng Orders.
5. Xóa những đối tượng DataRow được loại bỏ từ ordersDT từ bảng Orders.
6. Xóa những đối tượng DataRow được loại bỏ từ customersDT từ bảng Customers.

Để lấy những đối tượng DataRow đã được thêm, được cập nhật, hay đã bị xóa, bạn sử dụng phương thức Select() của một DataTable. Phương thức Select() được nói qua trong [chương trước đây](#), và một trong số những phiên bản quá tải của phương thức này:

```
DataRow[] Select(string filterExpression, string sortExpression,  
DataViewRowState myDataViewRowState)
```

#### Với:

- . filterExpression : chỉ định những hàng để chọn.
- . sortExpression : chỉ định những hàng được lựa chọn sẽ được sắp xếp như thế nào.
- . myDataViewRowState : chỉ rõ trạng thái của những hàng để chọn. Bạn có thể thấy những thành viên của lớp liệt kê DataRowState trong [Bảng 11.8 trong chương trước đây](#).

Để lấy những đối tượng DataRow đã được thêm vào DataTable customersDT , Bạn có thể sử dụng mã sau đây gọi phương thức Select() :

```
DataRow[] newCustomersDRArray =  
customersDT.Select("", "", DataViewRowState.Added);
```

Chú ý sự sử dụng hằng số Added từ liệt kê DataRowState.Điều này cho biết chỉ những đối tượng DataRow mới bổ sung ở customersDT sẽ được trả lại và lưu trữ trong newCustomersDRArray.

Và rồi bạn có thể đẩy những đối tượng DataRow trong newCustomersDRArray tới bảng Customers trong cơ sở dữ liệu sử dụng lệnh gọi sau đây tới phương thức Update() của DataAdapter customersDA:

```
int numOfRows = customersDA.Update(newCustomersDRArray);
```

Giá trị Int : numOfRows là số lượng hàng được thêm vào bảng Customers.

Mã sau đây sử dụng sáu bước được trình bày trước để đẩy tất cả những sự thay đổi tới cơ sở dữ liệu; chú ý những hằng số khác nhau được dùng từ bộ đếm DataRowState để lấy những đối tượng DataRow được yêu cầu .

```
mySqlConnection.Open();  
  
// push the new rows in customersDT to the database  
Console.WriteLine("Pushing new rows in customersDT to database");  
DataRow[] newCustomersDRArray =  
customersDT.Select("", "", DataViewRowState.Added);  
int numOfRows = customersDA.Update(newCustomersDRArray);
```

```

Console.WriteLine("numOfRows = " + numOfRows);

// push the new rows in ordersDT to the database
Console.WriteLine("Pushing new rows in ordersDT to database");
DataRow[] newOrdersDRArray =
    ordersDT.Select("", "", DataRowViewState.Added);
numOfRows = ordersDA.Update(newOrdersDRArray);
Console.WriteLine("numOfRows = "+ numOfRows);

// push the modified rows in customersDT to the database
Console.WriteLine("Pushing modified rows in customersDT to database");
DataRow[] modifiedCustomersDRArray =
    customersDT.Select("", "", DataRowViewState.ModifiedCurrent);
numOfRows = customersDA.Update(modifiedCustomersDRArray);
Console.WriteLine("numOfRows = " + numOfRows);

// push the modified rows in ordersDT to the database
Console.WriteLine("Pushing modified rows in ordersDT to database");
DataRow[] modifiedOrdersDRArray =
    ordersDT.Select("", "", DataRowViewState.ModifiedCurrent);
numOfRows = ordersDA.Update(modifiedOrdersDRArray);
Console.WriteLine("numOfRows = " + numOfRows);
// push the deletes in ordersDT to the database
Console.WriteLine("Pushing deletes in ordersDT to database");
DataRow[] deletedOrdersDRArray =
    ordersDT.Select("", "", DataRowViewState.Deleted);
numOfRows = ordersDA.Update(deletedOrdersDRArray);
Console.WriteLine("numOfRows = " + numOfRows);

// push the deletes in customersDT to the database
Console.WriteLine("Pushing deletes in customersDT to database");
DataRow[] deletedCustomersDRArray =
    customersDT.Select("", "", DataRowViewState.Deleted);
numOfRows = customersDA.Update(deletedCustomersDRArray);
Console.WriteLine("numOfRows = " + numOfRows);

mySqlConnection.Close();

```

Chương trình `ModifyingRelatedData.cs` chứa một phương pháp có tên `PushChangesToDatabase()` sử dụng mã trước đây.

Một điều bạn cần chú ý về `ModifyingRelatedData.cs` là nó gọi `PushChangesToDatabase()` ngay lập tức sau khi thực hiện theo những bước sau trong phương thức `Main()`.

1. Thêm những đối tượng `DataRow` vào `customersDT` và `ordersDT`.
2. Cập nhật những đối tượng `DataRow` mới.
3. Xóa những đối tượng `DataRow` mới.

`PushChangesToDatabase()` ngay lập tức được gọi sau mỗi bước này do đó bạn có thể nhìn thấy hoạt động cơ sở dữ liệu khi chương trình tiến triển. Tôi có thể đơn giản gọi `PushChangesToDatabase()` một lần tại thời điểm kết thúc của ba bước này \_nhưng rồi bạn không thấy được bất kỳ sự thay đổi nào tới cơ sở dữ liệu, vì những hàng mới đã được xóa trong bước thứ 3 trước `PushChangesToDatabase()` được gọi .

**Mẹo nhỏ:** Trong những chương trình của mình, bạn thích việc thêm, cập nhật, và xóa nhiều hàng khác nhau trong những đối tượng `DataTable` của bạn, và vì thế đẩy những sự thay đổi một lần ở thời

điểm cuối sẽ hiệu quả hơn.

## **NHỮNG VẤN ĐỀ LIÊN QUAN KHI CẬP NHẬT KHÓA CHÍNH CỦA MỘT HÀNG CHA:**

Trong mục này, bạn sẽ học về những vấn đề liên quan khi thử cập nhật khóa chính trong một DataTable cha , và sau đó đẩy sự cập nhật tới bảng cơ sở dữ liệu nằm bên dưới. Những vấn đề xuất hiện khi bảng cơ sở dữ liệu con đã chứa những hàng có khóa chính bạn muốn thay đổi trong bảng cha .

Những ví dụ trong mục này sẽ sử dụng những bảng Customers và Orders, mà có liên quan thông qua khóa ngoại trên cột CustomerID của bảng Orders tới cột CustomerID của bảng Customers.

Như bạn sẽ học, tốt hơn nhiều bạn rời bỏ, không cho phép những sự thay đổi tới cột khóa chính của một bảng. Nếu bạn cho phép thay đổi tới cột khóa chính, thì như bạn sẽ thấy không lâu sau đây, bạn có thể gặp những vấn đề khi đẩy những thay đổi tới cơ sở dữ liệu. Thay vào đó, bạn cần phải gán thuộc tính ReadOnly là true cho DataColumn khóa chính trong DataTable cha của bạn, và cũng đặt ReadOnly là true cho DataColumn khóa ngoại trong DataTable con của bạn. Điều đó ngăn ngừa những sự thay đổi tới những giá trị trong những đối tượng DataColumn này.

Nếu bạn thật sự cần thay đổi những giá trị khóa chính và khóa ngoại, bạn cần phải xóa và sau đó tái tạo những hàng trong cơ sở dữ liệu với những giá trị khóa chính và khóa ngoại mới .

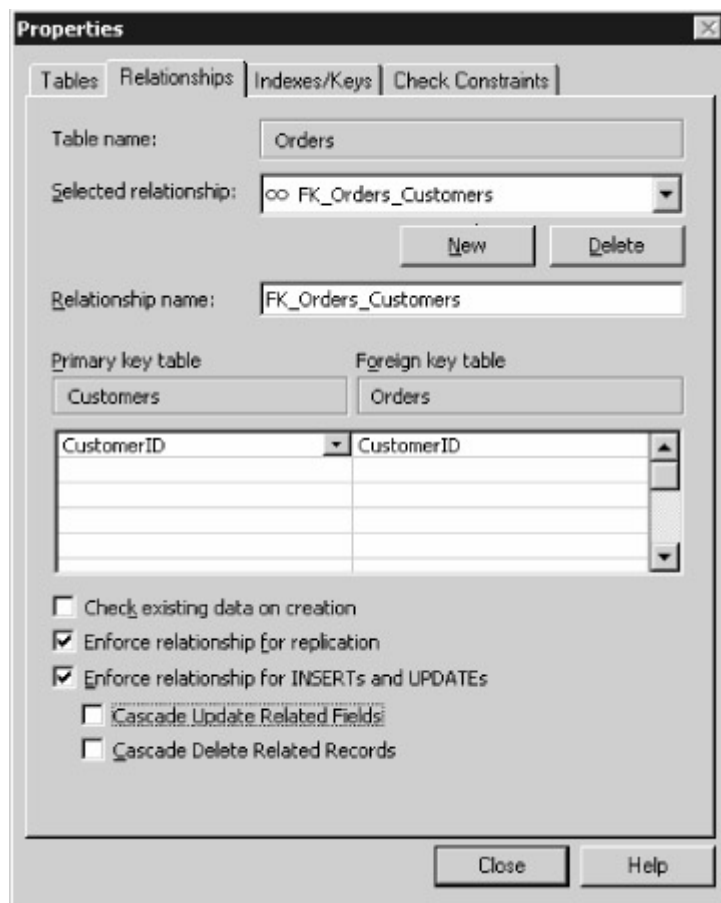
Bạn có thể điều khiển cách cập nhật và xóa được thực hiện như thế nào sử dụng những thuộc tính của khóa ngoại trong cơ sở dữ liệu SQL Server và cả những thuộc tính UpdateRule và DeleteRule của một đối tượng ForeignKeyConstraint nữa . Bạn sẽ khám phá cả hai tiết mục này trong những mục sau đây.

## **SỬ LÝ CẬP NHẬT VÀ XÓA SỬ DỤNG SQL SERVER:**

Bạn có thể điều khiển cách cập nhật và xóa được thực hiện như thế nào sử dụng SQL Server bằng cách thiết đặt những thuộc tính của khóa ngoại . Bạn thiết đặt những thuộc tính này sử dụng tab Relationships của hộp thoại "những thuộc tính của bảng cơ sở dữ liệu". Bạn mở hộp thoại này trong Enterprise Manager cho bảng Orders bởi thực hiện những bước sau đây:

1. Click phải bảng Orders tìm thấy trong những nút bảng của Enterprise Manager.
2. Lựa chọn bảng Design từ danh sách sổ ra.
3. Click nút Manage Relationships trong thanh công cụ của hộp thoại Design Table.
4. Chọn khóa ngoại bạn muốn khảo sát trong danh sách thả xuống Select relationship .

[Hình 12.1](#) hiển thị tab Relationships cho khóa ngoại có tên FK\_Orders\_Customers nó chứa những chi tiết của khóa ngoại giữa những bảng Orders và Customers. Như bạn có thể thấy, hai bảng này có liên quan thông qua một khóa ngoại trên cột CustomerID.



Hộp check box "Cascade Update Related Fields" cho biết liệu có phải một thay đổi tới một giá trị trong cột khóa chính của bảng khóa chính (bảng cha) cũng được thực hiện tới cột khóa ngoại của những hàng tương ứng của bảng khóa ngoại (bảng con). Chẳng hạn, giả thiết hộp này được chọn và bạn đã thay đổi CustomerID trong hàng của bảng Customers từ ALFKI đến ANATR; Điều này cũng gây cho cột CustomerID thay đổi từ ALFKI đến ANATR trong những hàng của bảng Orders.

Tương tự, hộp kiểm "Cascade Delete Related Records" chỉ định liệu việc xóa một hàng trong bảng khóa chính cũng xóa bất kỳ hàng liên quan nào từ bảng khóa ngoại. Chẳng hạn, giả thiết hộp kiểm này được chọn và bạn xóa hàng có CustomerID là ANTON từ bảng Customers; điều này gây ra những hàng với CustomerID là ANTON cũng bị xóa từ bảng Orders.

#### Ghi chú:

Diễn hình, bạn cần phải để cả hai hộp kiểm trong trạng thái không chọn mặc định của chúng. Nếu bạn chọn chúng, cơ sở dữ liệu sẽ làm thay đổi tới những hàng trong bảng con phía sau ngữ cảnh và như bạn sẽ thấy không lâu sau đây, bạn sẽ gặp những vấn đề khi đây những sự thay đổi từ dataset của các bạn đến cơ sở dữ liệu.

### Kiểm soát cập nhật và xóa sử dụng những thuộc tính *UpdateRule* và *DeleteRule* của một Đối tượng *ForeignKeyConstraint*

Bạn có thể cũng kiểm soát những sự cập nhật và xóa sử dụng những thuộc tính *UpdateRule* và *DeleteRule* của một đối tượng *ForeignKeyConstraint*. Những thuộc tính này thuộc kiểu liệt kê *System.Data.Rule*; những thành viên của kiểu này được trình bày trong [Bảng 12.4](#).

Bảng 12.4: những thành viên liệt kê- Rule (quy tắc)

Hằng số	Mô tả
Cascade	Chỉ định việc xóa hay cập nhật tới những đối tượng DataRow trong DataTable cha cũng được

Hàng số	Mô tả
	thực hiện trong DataTable con. Đây là mặc định.
None	cho biết không có hoạt động nào xảy ra.
SetDefault	cho biết những giá trị DataColumn trong DataTable cha sẽ được thiết lập tới giá trị trong thuộc tính DefaultValue của DataColumn.
SetNull	cho biết những giá trị DataColumn trong DataTable con sẽ được gán tới DBNull (giá trị null)

Theo mặc định, UpdateRule được gán là cascade; bởi vậy, khi bạn thay đổi DataColumn trong DataTable cha trong đó ForeignKeyConstraint được tạo ra, thì sự thay đổi giống như vậy cũng được thực hiện trong bất kỳ đối tượng DataRow tương ứng nào trong DataTable con. Bạn cần phải gán UpdateRule tới None trong chương trình của bạn; ngoài ra, như bạn sẽ học trong [mục kế tiếp](#), bạn sẽ gặp những vấn đề khi đẩy những thay đổi từ Dataset của bạn đến cơ sở dữ liệu.

Theo mặc định, DeleteRule được gán là Cascade ; do đó, khi bạn xóa một DataRow trong DataTable cha, thì bất kỳ đối tượng DataRow tương ứng nào trong DataTable con đều cũng bị xóa. Điều này hữu ích, miễn là bạn nhớ đẩy những sự xóa tới bảng con trước khi bạn đẩy những sự xóa tới bảng cha.

## **Cập nhật khóa chính của một Bảng cha và đẩy sự thay đổi tới Cơ sở dữ liệu**

Trong mục này bạn sẽ học những gì sẽ xảy ra nếu bạn thử cập nhật khóa chính trong một bảng cha khi có những hàng tương ứng trong bảng con. Giả thiết như sau:

- Có một hàng trong bảng Customers với một CustomerID là J6COM. Một bản sao của hàng này được cất giữ trong một DataTable có tên customersDT.
- Có một hàng trong bảng Orders cũng có một CustomerID là J6COM. Một bản sao của hàng này được cất giữ trong một DataTable có tên ordersDT.
- Đối tượng DataTable - CustomersDT và ordersDT liên hệ lẫn nhau sử dụng DataRelation sau đây :

```
DataRelation customersOrdersDataRel =
    new DataRelation(
        "CustomersOrders",
        customersDT.Columns["CustomerID"],
        ordersDT.Columns["CustomerID"]
    );
myDataSet.Relations.Add(
    customersOrdersDataRel
);
```

Bây giờ, hai sự thiết đặt cho hộp kiểm "Cascade Update Related Fields" (Đồng bộ cập nhật những trường quan hệ) cho FK\_Orders\_Customers là:

- Bỏ chọn , có nghĩa là những sự thay đổi đối với giá trị khóa chính CustomerID trong bảng Customers không được buộc kết tới bảng Orders. Đây là mặc định.
- Chọn hộp kiểm, có nghĩa là những sự thay đổi đối với giá trị khóa chính CustomerID trong bảng Customers được buộc kết tới bảng Orders.

Ngoài ra, những sự thiết đặt về quyền hạn cho thuộc tính UpdateRule của đối tượng ForeignKeyConstraint được thêm vào khi DataRelation trước đó được tạo ra

- Cascade , có nghĩa là những sự thay đổi đối với DataColumn CustomerID của customersDT được buộc kết tới ordersDT. Đây là mặc định.

- None, nghĩa là những sự thay đổi tới DataColumn CustomerID của customersDT không được buộc kết tới ordersDT.

Hãy khảo sát ba trường hợp quan trọng nhất là: thay đổi việc chọn kiểm của hộp kiểm "Cascade Update Related Fields" và việc thiết đặt thuộc tính UpdateRule đến Cascade và sau đó là None.

#### **Ghi chú:**

Bạn có thể sử dụng chương trình ModifyingRelatedData2.cs như cơ sở để thử ba trường hợp mô tả trong mục này.

### **TRƯỜNG HỢP THỨ NHẤT:**

Giả thiết như sau:

Hộp kiểm "Cascade Update Related Fields" được chọn kiểm.

UpdateRule được gán tới Cascade .

Nếu bạn thay đổi giá trị DataColumn CustomerID từ J6COM thành J7COM và đẩy sự thay đổi tới cơ sở dữ liệu, và sự thay đổi được thực hiện cách thành công trong những đối tượng DataTable customersDT và ordersDT và cũng trong những bảng Customers và Orders.

Nó làm việc tốt miễn là bạn chỉ sử dụng cột OrderID trong mệnh đề WHERE của đối tượng Command trong thuộc tính UpdateCommand của DataAdapter của bạn. Chẳng hạn:

```
ordersUpdateCommand.CommandText =  
    "UPDATE Orders " +  
    "SET " +  
    " CustomerID = @NewCustomerID " +  
    "WHERE OrderID = @OldOrderID";
```

Cập nhật này sử dụng sự tương tranh " người sau cùng thắng " vì chỉ cột khóa chính OrderID được sử dụng trong mệnh đề WHERE ( cột CustomerID cũ được bỏ ra khỏi mệnh đề WHERE ). Như được đề cập trong [chương trước đây](#), sự tương tranh "người sau cùng hắng " rất tệ bởi vì một người sử dụng có thể ghi đè lên một sự thay đổi được thực hiện bởi người sử dụng khác.

Nếu thay vào đó bạn cũng bao gồm giá trị cột CustomerID cũ trong mệnh đề WHERE của sự Cập nhật, như trình bày trong ví dụ sau đây,

```
ordersUpdateCommand.CommandText =  
    "UPDATE Orders " +  
    "SET " +  
    " CustomerID = @NewCustomerID " +  
    "WHERE OrderID = @OldOrderID " +  
    "AND CustomerID = @OldCustomerID";
```

Và rồi sự đẩy thay đổi tới cơ sở dữ liệu sẽ thất bại vì hàng nguyên thủy trong bảng Orders không được tìm thấy do CustomerID đã được thay đổi từ J6COM thành J7COM trong bảng Orders một cách tự động bởi cơ sở dữ liệu do hộp kiểm "Cascade Update Related Fields " được chọn kiểm cho khóa ngoại trong bảng Orders, nhưng trong ordersDT giá trị CustomerID được gán là J6COM. Bởi vậy, Sự thêm của OrderID = @OldOrderID trong mệnh đề WHERE cản trở hàng được tìm thấy. lệnh UPDATE gây ra một DBConcurrencyException được ném ra.

### **TRƯỜNG HỢP THỨ HAI:**

Giả thiết như sau:

- Hộp kiểm "Cascade Update Related Fields" không được chọn.
- UpdateRule được gán tới Cascade.

- Thuộc tính CommandText của đối tượng Command trong thuộc tính UpdateCommand của DataAdapter được thiết đặt như sau:

```
ordersUpdateCommand.CommandText =  
"UPDATE Orders " +  
"SET " +  
" CustomerID = @NewCustomerID " +  
"WHERE OrderID = @OldOrderID";
```

Nếu bạn thay đổi CustomerID từ J6COM tới J7COM trong customersDT và đẩy sự thay đổi tới cơ sở dữ liệu, và sự cập nhật sẽ ném một SQLException. Đây là vì bảng con Orders hiện đang chứa một hàng với CustomerID là J6COM, và bởi vì khóa ngoại bạn không thể thay đổi CustomerID trong bảng cha Customers. Mặc dù bạn thử thay đổi CustomerID trong ordersDT trước và cố gắng đẩy sự thay đổi tới cơ sở dữ liệu, bạn sẽ gặp ngoại lệ giống như vậy.

### **TRƯỜNG HỢP THỨ BA:**

Giả thiết như sau:

Cascade Update Related Fields không được kiểm chọn.

UpdateRule được gán tới None.

CommandText của đối tượng Command trong UpdateCommand của DataAdapter cũng giống như trong trường hợp thứ hai.

Những mã sau đây gán UpdateRule của ChildKeyConstraint đến None :

```
myDataSet.Relations["CustomersOrders"].ChildKeyConstraint.UpdateRule =  
Rule.None;
```

Nếu bạn thử thay đổi CustomerID từ J6COM tới J7COM trong customersDT, thì bạn sẽ gây ra một InvalidConstraintException. Đây là vì DataTable con ordersDT hiện thời chứa một DataRow với CustomerID là J6COM, và vì khóa ngoại bạn không thể thay đổi CustomerID trong DataTable cha customersDT. Mặc dù bạn thử thay đổi CustomerID trong ordersDT trước, bạn sẽ gặp ngoại lệ giống như vậy.

### **Kết luận:**

Sự cường bức của những sự ràng buộc trong ba ví dụ trước đây là đúng, và chúng cho thấy một sự nhức đầu khi thay đổi những giá trị cột khóa chính có thể đem đến. trường hợp đầu tiên là một trường hợp duy nhất hoạt động, và mặc dù bạn đã phải sử dụng đến sự tương tranh " người sau cùng thắng " trong phát biểu UPDATE, mà điển hình bạn cần phải tránh.

Bạn sẽ làm gì nếu như bạn muốn thay đổi giá trị cột khóa chính và áp dụng sự thay đổi giống như vậy tới những hàng con ? Cách dễ dàng nhất là đơn giản xóa những hàng trong bảng con trước, thay đổi giá trị khóa chính trong bảng cha, và tái tạo những hàng trong bảng con với giá trị khóa chính mới.

### **MANG LƯỚI XML:**

Như được đề cập trong [Chương 10](#), " Sử dụng những đối tượng Dataset để lưu trữ dữ liệu, " một Dataset chứa hai phương thức gửi ra nội dung của những đối tượng DataRow như XML:

- . GetXml() trả về một dạng trình bày XML của dữ liệu cất giữ trong Dataset như một chuỗi.
- . WriteXml() viết dữ liệu từ đối tượng Dataset ra thành một file dưới dạng XML.



Một DataRelation chứa đựng một thuộc tính có tên Nested, từ nó có thể lấy hay gán một giá trị bool cho biết liệu có phải những đối tượng DataRelation được kết vào. Điều này hữu ích khi định nghĩa những mối quan hệ có thứ bậc trong XML.

Đặc biệt, khi bạn gán Nested là true, những hàng con sẽ được kết vào bên trong những hàng cha trong bất kỳ file XML nào mà bạn gửi ra - sử dụng những phương thức GetXml() và WriteXml() . Tương tự, bạn có thể đọc những hàng được kết vào khi gọi phương thức ReadXml() của một DataSet để đọc một file XML.

Thuộc tính Nested của một đối tượng DataRelation được gán tới true như trong ví dụ sau đây:

```
myDataSet.Relations["CustomersOrders"].Nested = true;
```

Điều này được trình bày trong [Danh sách 12.1](#). Chú ý chương trình này viết ra hai file XML có tên nonNestedXmlFile.xml và nestedXmlFile.xml. nonNestedXmlFile.xml chứa những hàng không được lồng vào theo mặc định, và nestedXmlFile.xml chứa những hàng được lồng vào sau khi thuộc tính Nested của đối tượng DataRelation được gán tới true .

#### Danh sách 12.1: NESTEDXML.CS

```
/*
NestedXml.cs illustrates how setting the Nested property
of a DataRelation to true causes the the child rows to be nested within the
parent rows in the output XML
*/

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

class NestedXml
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );

        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT TOP 2 CustomerID, CompanyName " +
            "FROM Customers " +
            "ORDER BY CustomerID;" +
            "SELECT OrderID, CustomerID, ShipCountry " +
            "FROM Orders " +
            "WHERE CustomerID IN (" +
            " SELECT TOP 2 CustomerID " +
            " FROM Customers " +
            " ORDER BY CustomerID " +
            ")";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
```

```

mySqlConnection.Open();
int numberOfRows = mySqlDataAdapter.Fill(myDataSet);
Console.WriteLine("numberOfRows = " + numberOfRows);
mySqlConnection.Close();
DataTable customersDT = myDataSet.Tables["Table"];
DataTable ordersDT = myDataSet.Tables["Table1"];

// create a DataRelation object named customersOrdersDataRel
DataRelation customersOrdersDataRel =
    new DataRelation(
        "CustomersOrders",
        customersDT.Columns["CustomerID"],
        ordersDT.Columns["CustomerID"]
    );
myDataSet.Relations.Add(
    customersOrdersDataRel
);

// write the XML out to a file
Console.WriteLine("Writing XML out to file nonNestedXmlFile.xml");
myDataSet.WriteXml("nonNestedXmlFile.xml");

// set the DataRelation object's Nested property to true
// (causes child rows to be nested in the parent rows of the
// XML output)
myDataSet.Relations["CustomersOrders"].Nested = true;

// write the XML out again (this time the child rows are nested
// within the parent rows)
Console.WriteLine("Writing XML out to file nestedXmlFile.xml");
myDataSet.WriteXml("nestedXmlFile.xml");
}
}

```

**Danh sách 12.2** hiện thị file nonNestedXmlFile.xml được xuất ra bởi chương trình. Chú ý những hàng cha từ bảng Customers được liệt kê đầu tiên, theo sau là những hàng con từ bảng Orders. Những hàng con không được lồng bên trong những hàng cha .

#### Danh sách 12.2: NONNESTEDXMLFILE.XML

```

<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Table>
    <CustomerID>ALFKI</CustomerID>
    <CompanyName>Alfreds Futterkiste</CompanyName>
  </Table>
  <Table>
    <CustomerID>ANATR</CustomerID>
    <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
  </Table>
  <Table1>
    <OrderID>10308</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>

```

```

    <OrderID>10625</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10643</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10692</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10702</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10759</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10835</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10926</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10952</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>11011</OrderID>
    <CustomerID>ALFKI</CustomerID>
    <ShipCountry>Germany</ShipCountry>
  </Table1>
</NewDataSet>

```

**Danh sách 12.3** hiển thị file nestedXmlFile.xml được xuất ra bởi chương trình. Chú ý lúc này những hàng con từ bảng Orders được lồng bên trong những hàng cha từ bảng Customers.

### Danh sách 12.3: NESTEDXMLFILEL.CS

```

<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Table>
    <CustomerID>ALFKI</CustomerID>
    <CompanyName>Alfreds Futterkiste</CompanyName>

```

```
<Table1>
  <OrderID>10643</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
<Table1>
  <OrderID>10692</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
<Table1>
  <OrderID>10702</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
<Table1>
  <OrderID>10835</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
<Table1>
  <OrderID>10952</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
<Table1>
  <OrderID>11011</OrderID>
  <CustomerID>ALFKI</CustomerID>
  <ShipCountry>Germany</ShipCountry>
</Table1>
</Table>
<Table>
  <CustomerID>ANATR</CustomerID>
  <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>
  <Table1>
    <OrderID>10308</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10625</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10759</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
  <Table1>
    <OrderID>10926</OrderID>
    <CustomerID>ANATR</CustomerID>
    <ShipCountry>Mexico</ShipCountry>
  </Table1>
</Table>
</NewDataSet>
```

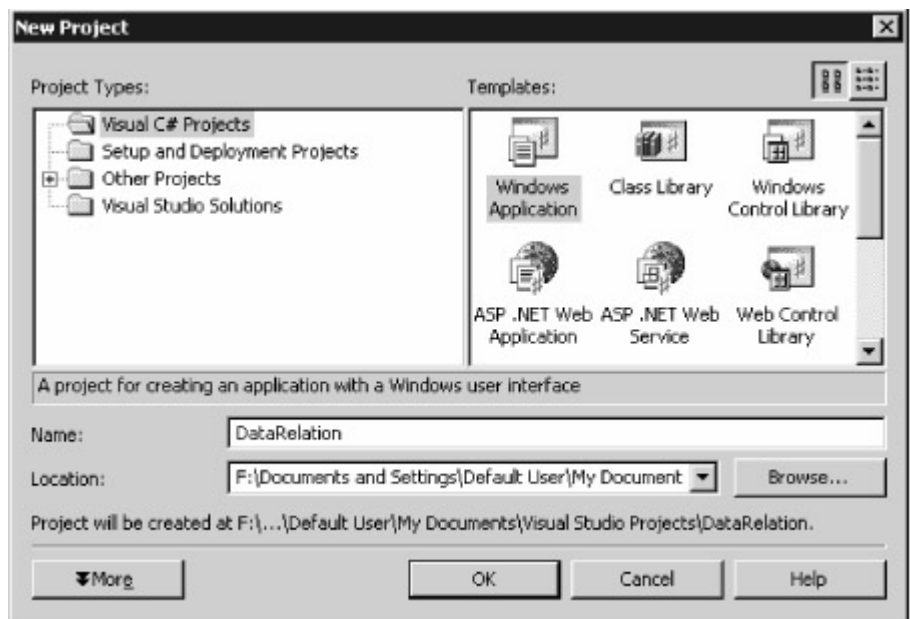
## **ĐỊNH NGHĨA MỘT MỐI QUAN HỆ SỬ DỤNG *Visual Studio .NET***

Trong mục này, bạn sẽ thấy cách tạo ra một ứng dụng Windows trong Visual Studio .NET (VS .NET) với một Dataset chứa hai đối tượng DataTable như thế nào. Những đối tượng DataTable này sẽ tham chiếu những bảng Customers và Orders trong cơ sở dữ liệu. Bạn sẽ học cách định nghĩa một mối quan hệ giữa những hai đối tượng DataTable trong mô hình XML như thế nào.

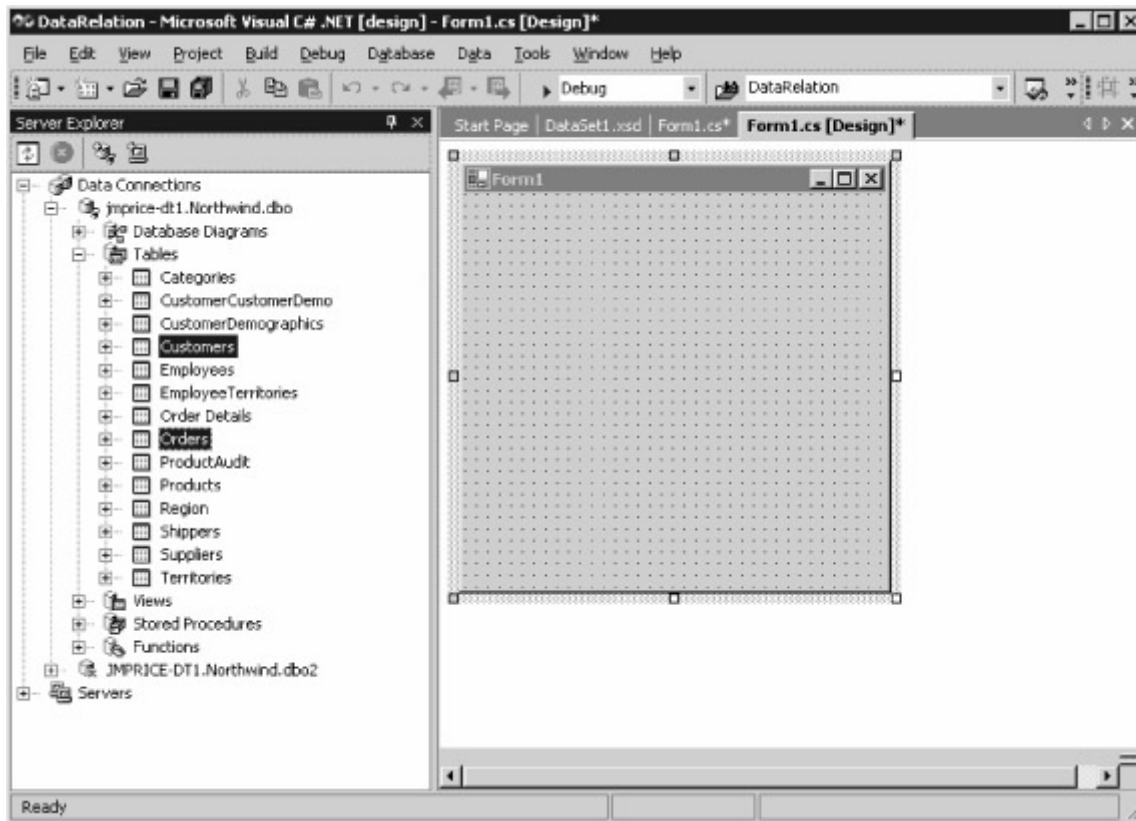
### **TAO MỘT TRÌNH ỨNG DỤNG WINDOWS:**

Thực hiện những bước sau đây để tạo ứng dụng Windows:

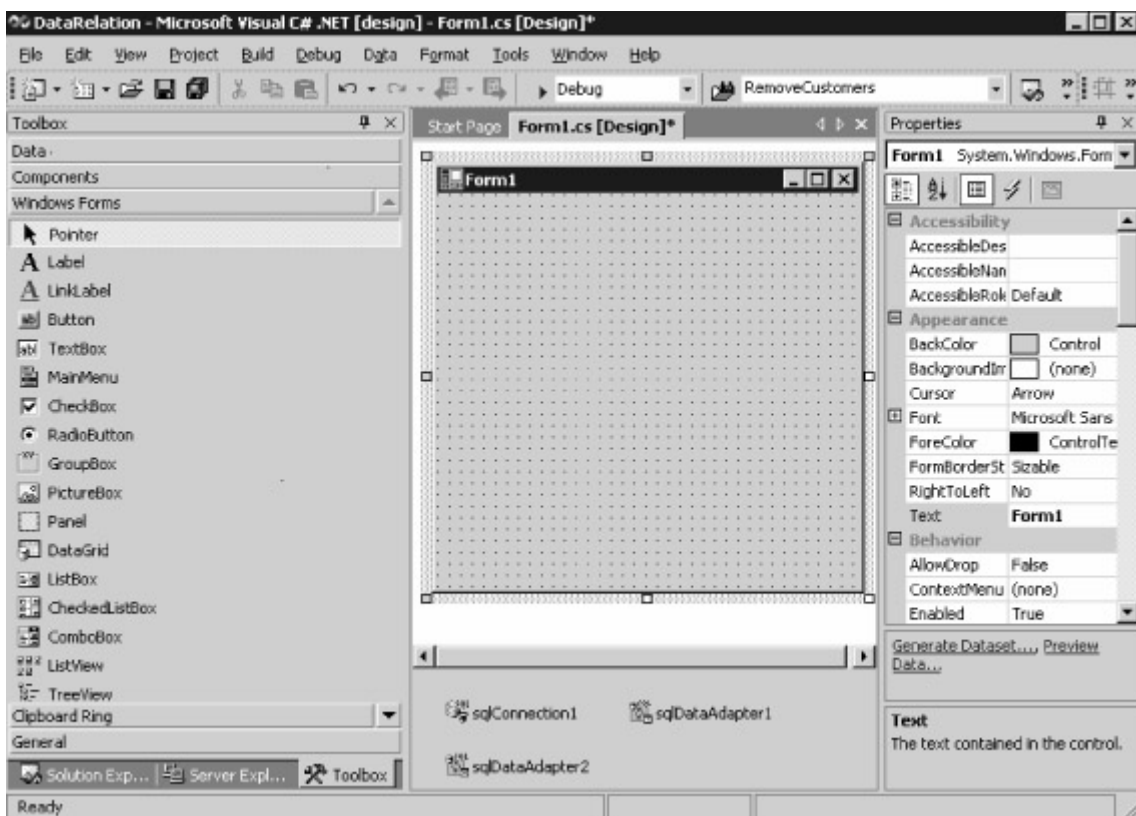
1. Mở VS .NET và chọn File ➤ New ➤ Project và tạo ra một ứng dụng Windows mới. Nhập tên của dự án là DataRelation, như trình bày trong [Hình 12.2](#).



2. Kích nút Ok để tiếp tục.
3. Mở Server Explorer và kết nối tới cơ sở dữ liệu Northwind - sử dụng kết nối bạn đã dùng trong những chương trước đây. Bạn có thể mở Server Explorer bởi chọn View ➤ Server Explorer. Mở rộng những nút bằng trong cây thư mục và chọn cả hai bảng Customers và Orders bằng cách nhấn Ctrl + kích nút trái để chọn các bảng, như trình bày trong [Hình 12.3](#).

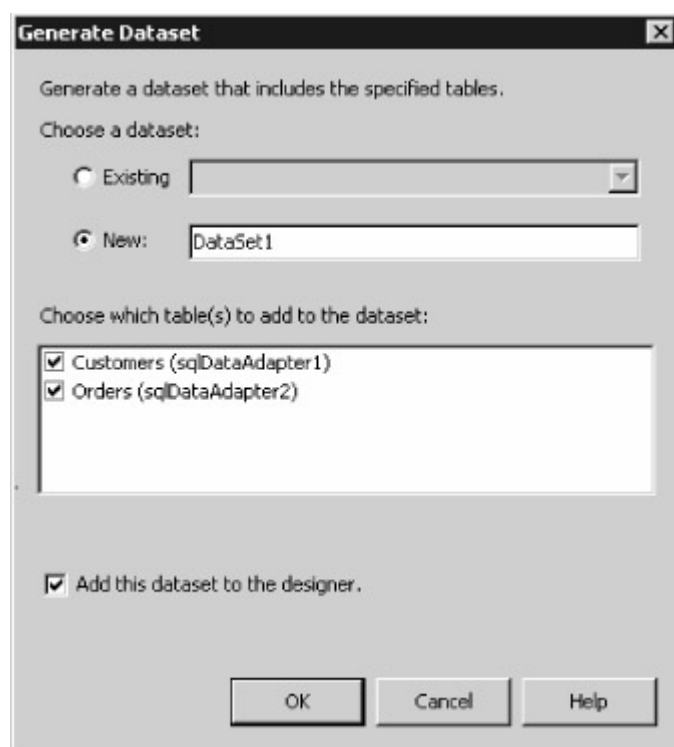


4. Kéo những bảng Customers và Orders tới form của bạn. VS .NET sẽ tự động tạo ra ba đối tượng trong khay bên dưới form của bạn. Những đối tượng này có tên sqlConnection1 (dùng để truy cập cơ sở dữ liệu Northwind) sqlDataAdapter1 (dùng để xử lý truy cập tới bảng Customers), và sqlDataAdapter2 (dùng để xử lý sự truy cập tới bảng Orders). Những đối tượng này được trình bày trong [Hình 12.4](#).



5. Tiếp theo, bạn cần một đối tượng Dataset chứa những đối tượng DataTable để lưu trữ những hàng từ những bảng Customers và Orders. Để tạo ra một đối tượng Dataset, kích vào khoảng trống trên form của bạn và sau đó kích vào liên kết Generate Dataset ở đáy của cửa sổ Properties thuộc form đã trình bày trong hình [Hình 12.4](#) trước.

điều này làm hiển thị hộp thoại "Generate Dataset" , như trong [Hình 12.5](#). Để nguyên tất cả những sự thiết đặt trong hội thoại này trong trạng thái mặc định của chúng.



Chú ý một Dataset mới sẽ được tạo ra, và những bảng Customers và Orders được sử dụng trong Dataset mới. Dataset mới sẽ cũng được thêm vào bộ thiết kế, như được chỉ định bởi hộp kiểm trong hộp thoại. Kích nút Ok để tiếp tục.

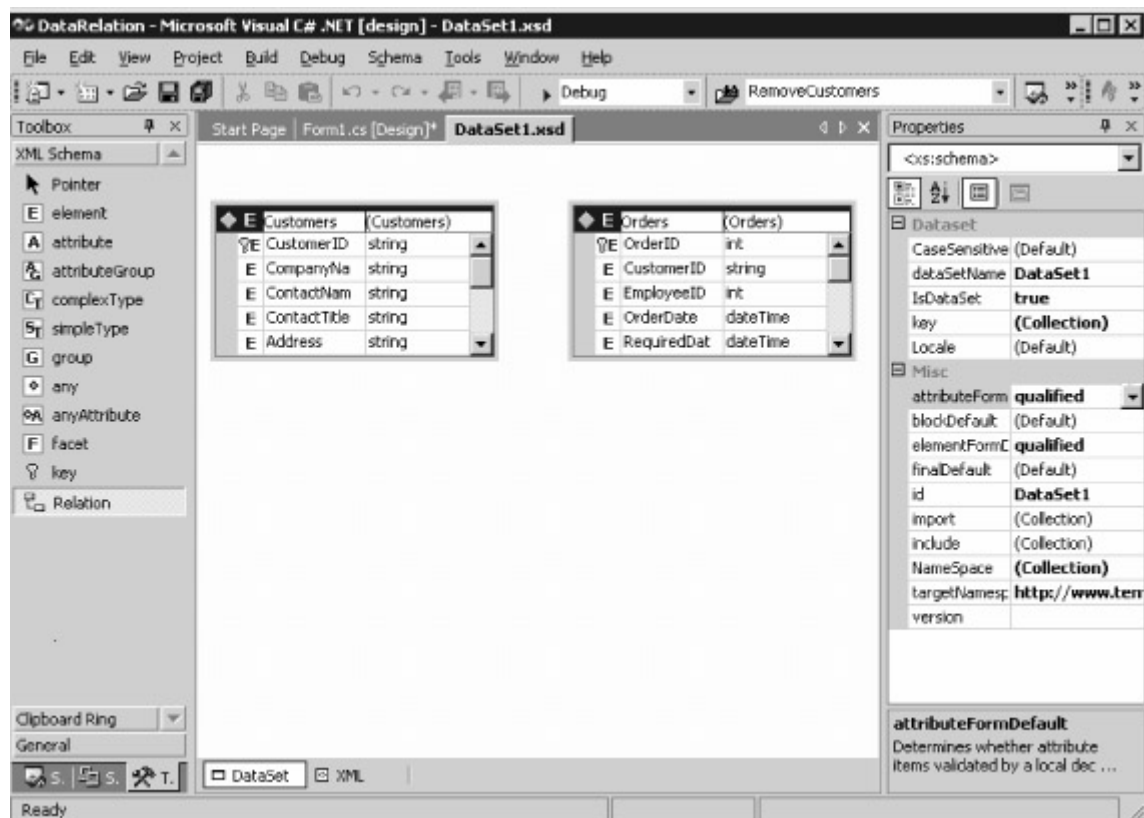
Một Dataset mới có tên dataSet11 sẽ được thêm vào khay bên dưới form của bạn.

### **Thêm một quan hệ vào mô hình XML của Dataset**

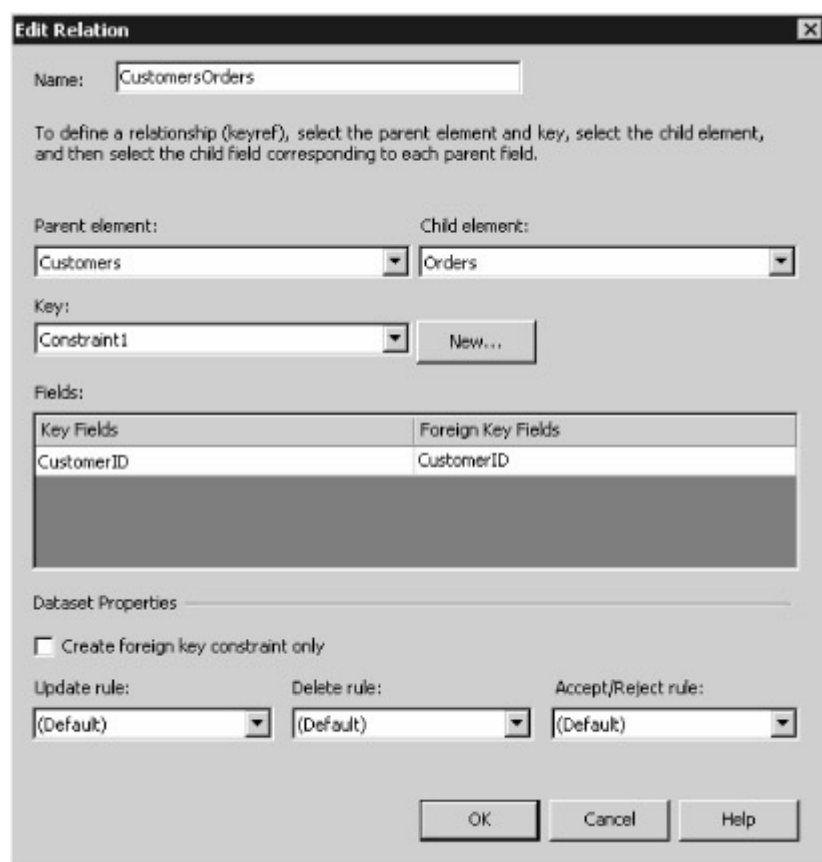
Thực hiện những bước sau đây để thêm một quan hệ vào mô hình XML của Dataset của bạn:

1. Chọn Dataset mới của bạn trong khay và kích liên kết "View Schema" trong cửa sổ những thuộc tính. việc này hiển thị cửa sổ "Schema Editor" (bộ biên tập mô hình), như trong [Hình 12.6](#).





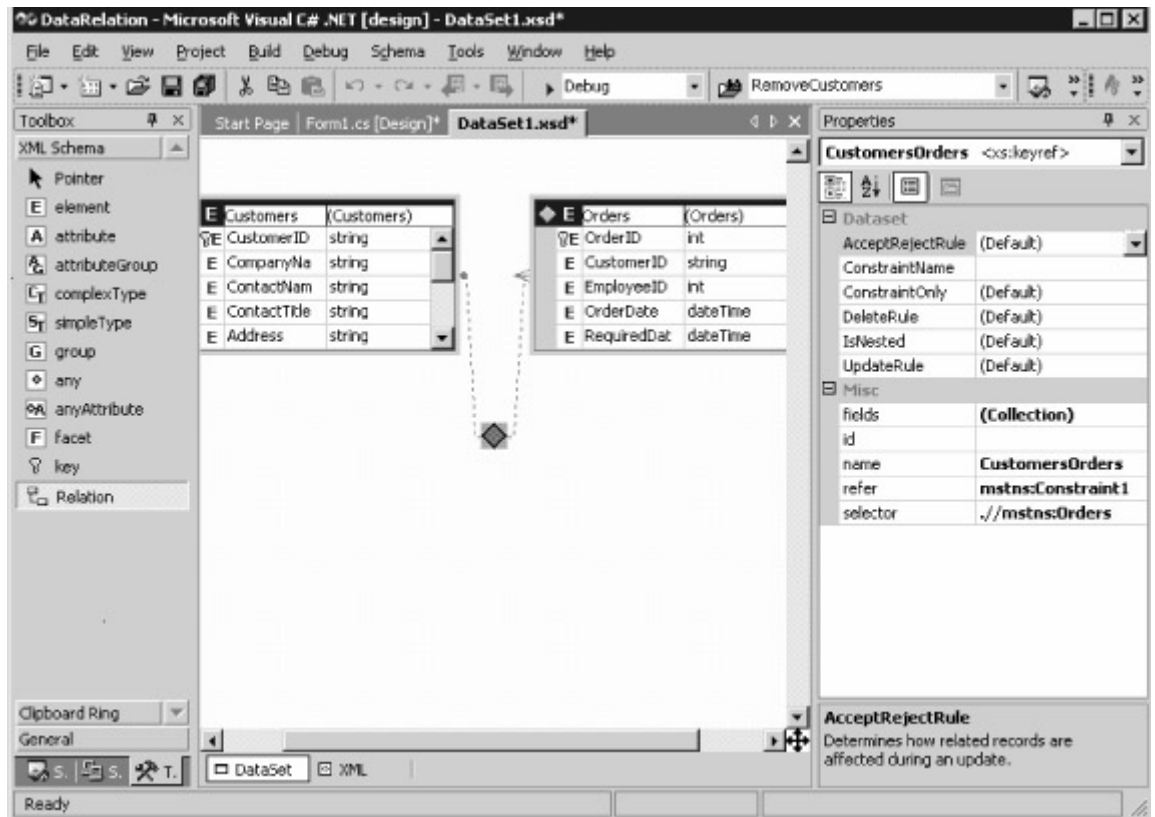
Kéo một relation từ tab "XML Schema" trong Toolbox đến thực thể bảng Orders. Việc này mở hộp thoại "Edit Relation" (Soạn thảo quan hệ) , như trong [Hình 12.7](#).



Như bạn có thể thấy từ [Hình 12.7](#), bạn có thể gán quan hệ cùng với những chi tiết khác cho những bảng cha và con , khóa chính và khóa ngoại .

3. Để nguyên những sự thiết đặt trong hộp thoại Edit Relation trong trạng thái mặc định của chúng và kích nút Ok để tiếp tục.

Việc này thêm quan hệ mới vào những thực thể Customers và Orders trong mô hình XML. Bạn có thể kích vào biểu tượng "kim cương" giữa những thực thể Customers và Orders để xem những thuộc tính của quan hệ, như trình bày trong [Hình 12.8](#).



**Ghi chú** Bạn có thể thêm một quan hệ vào một Dataset định kiểu mạnh trong cùng cách với mục được mô tả trong phần này.

## **Tóm lược**

Trong chương này, bạn đã đi sâu vào những chi tiết của những đối tượng UniqueConstraint và ForeignKeyConstraint. Bạn cũng đã học cách định nghĩa một mối quan hệ giữa những đối tượng DataTable sử dụng một đối tượng DataRelation như thế nào. Đối tượng quan hệ này theo mặc định tự động tạo ra một những đối tượng UniqueConstraint và ForeignKeyConstraint cho Bạn. Đối tượng UniqueConstraint được thêm vào DataTable cha của bạn và ForeignKeyConstraint được thêm vào DataTable con của bạn.

Bạn cũng đã thấy cách để định hướng những hàng trong những đối tượng DataTable liên quan như thế nào, thực hiện những sự thay đổi trong những đối tượng DataTable liên quan, và cuối cùng đẩy những sự thay đổi đó tới cơ sở dữ liệu.

Trong [chương kế tiếp](#), bạn sẽ học cách sử dụng những đối tượng DataView như thế nào.

## **CHƯƠNG 13: SỬ DỤNG NHỮNG ĐỐI TƯỢNG DataView:**

### **Tổng quan**

Trong [chương 11](#), "Sử dụng những đối tượng Dataset để sửa đổi Dữ liệu," Bạn đã thấy bạn có thể lọc và phân loại những đối tượng DataRow ở một DataTable sử dụng phương thức Select(). Trong chương này, bạn sẽ học sử dụng những đối tượng DataView cũng để lọc và phân loại những hàng. Lợi thế của một DataView là bạn có

thể kết buộc nó tới một thành phần trực quan như một điều khiển DataGridView, và bạn sẽ cũng sẽ thấy cách làm điều đó trong chương này.

Một DataView lưu trữ những bản sao của những hàng trong một DataTable như những đối tượng DataRowView. Những đối tượng DataRowView cung cấp sự truy cập tới những đối tượng DataRow nằm bên dưới trong một DataTable. Bởi vậy, khi bạn khảo sát và sửa đổi nội dung của một DataRowView, Bạn thật sự đang làm việc với DataRow nằm bên dưới. hãy nhớ điều này khi đọc chương này.

**Chủ điểm trong chương này:**

- Lớp DataView
- Tạo ra và sử dụng một đối tượng DataView
- Sử dụng giải thuật phân loại mặc định
- Thực hiện sự lọc nâng cao
- **Lớp DataRowView**
- Tìm những đối tượng DataRowView trong một DataView
- **Thêm, điều chỉnh, và loại bỏ những đối tượng DataRowView từ một DataView**
- Tạo những đối tượng DataView con
- Lớp DataViewManager
- **Tạo ra và sử dụng một đối tượng DataViewManager**
- Việc tạo một DataView sử dụng Visual Studio .NET

**LỚP DATAVIEW:**

Bạn sử dụng một đối tượng của lớp DataView để xem chỉ những hàng đặc biệt trong một đối tượng DataTable sử dụng một bộ lọc. Bạn cũng có thể phân loại những hàng được xem bởi một DataView. Bạn có thể thêm, sửa đổi, và loại bỏ những hàng từ một DataView, và những sự thay đổi đó sẽ cũng được ứng dụng vào DataTable nằm bên dưới mà DataView đọc được; bạn sẽ học nhiều hơn về điều này sau trong mục "**Thêm, điều chỉnh, và loại bỏ những đối tượng DataRowView từ một DataView.**" **Bảng 13.1 cho thấy một số những thuộc tính DataView, và bảng 13.2 cho thấy một số những phương thức DataView.**

Bảng 13.1: những thuộc tính DataView

**Thuộc tính      Kiểu dữ liệu      Mô tả**

AllowDelete	bool	Lấy hay gán một giá trị bool cho biết liệu có phải sự xóa của những đối tượng DataRowView từ DataView của bạn được cho phép. Mặc định là true.
AllowEdit	bool	Lấy hay gán một giá trị bool cho biết liệu có phải sự sửa đổi của những đối tượng DataRowView trong DataView của bạn được cho phép. Mặc định là true.
AllowNew	bool	Lấy hay gán một giá trị bool cho biết liệu có phải việc thêm những đối tượng DataRowView mới vào DataView của bạn được cho phép. Mặc định là true.
ApplyDefaultSort	bool	Lấy hay gán một giá trị bool cho biết liệu có phải sử dụng giải thuật phân loại mặc định để phân loại những hàng trong DataView của bạn. Khi gán là true, phân loại mặc định được sử dụng và được gán tới "thứ tự tăng dần" của thuộc tính PrimaryKey của DataTable nằm bên dưới (nếu PrimaryKey được thiết đặt). Mặc định là false.
Count	int	Lấy số lượng hàng hiển thị đối với DataView của bạn.

DataViewManager	DataViewManager	Lấy DataViewManager có liên hệ với DataView của bạn. Bạn sẽ học về những đối tượng DataViewManager sau trong mục " <a href="#">Tạo và sử dụng một Đối tượng DataViewManager</a> ."
RowFilter	string	Lấy hay gán biểu thức được dùng để lọc những hàng trong DataView của bạn.
RowStateFilter	DataViewRowState	Lấy hay gán biểu thức được dùng để lọc những hàng dựa vào những hằng số từ liệt kê DataViewRowState. Những giá trị được trình bày trong <a href="#">Bảng 13.3</a> .
Sort	string	Lấy hay gán một biểu thức cho biết những cột cần phân loại bởi một trật tự sắp xếp và tùy chọn cho những hàng trong DataView của bạn. Biểu thức chuỗi này chứa tên cột theo sau là ASC (sắp xếp tăng) hay DESC (sắp xếp giảm). Một cột được sắp xếp tăng dần theo mặc định. Bạn phân chia nhiều cột bằng những dấu phẩy trong chuỗi. Ví dụ: CustomerID ASC, CompanyName DESC.
Table	DataTable	Lấy hay gán DataTable nằm bên dưới mà DataView của bạn liên hệ đến.

**Bảng 13.2: những phương thức của DataView**

Phương thức	Kiểu trả về	Mô tả
AddNew()	DataViewRow	Thêm một DataRowView mới vào DataView của bạn, và do đó thêm một DataRow mới vào DataTable nằm bên dưới.
BeginInit()	void	Bắt đầu sự khởi tạo runtime của DataView của bạn trong một form hay thành phần.
CopyTo()	void	Sao chép những hàng từ DataView của bạn vào trong một mảng. Phương thức này chỉ dành cho những giao diện Web form.
Delete()	void	Xóa DataRowView có chỉ số được chỉ định ra khỏi DataView của bạn. Sự xóa của DataRow nằm bên dưới không được duy trì cho đến khi bạn gọi phương thức AcceptChanges() của DataTable của bạn. Bạn có thể hủy lệnh xóa bằng cách gọi phương thức RejectChanges() của DataTable của bạn.
EndInit()	void	Kết thúc sự khởi tạo runtime của DataView của bạn trong một form hay thành phần.
Find()	int	Bị quá tải. Tìm và trả về chỉ số của DataRowView với khóa chính được chỉ rõ trong DataView của bạn. Giá trị Int được trả lại bởi phương thức này là chỉ số của DataRowView nếu được tìm thấy; nếu không -1 được trả về. trước tiên bạn phải đặt thuộc tính Sort của DataView của bạn để phân loại trên khóa chính. Chẳng hạn, nếu bạn muốn tìm một DataRowView đặt cơ sở trên CustomerID, Bạn phải đặt Sort tới CustomerID, CustomerID ASC, hay CustomerID DESC.
FindRows()	DataRowView[]	Bị quá tải. Tìm và trả lại một mảng của những đối tượng DataRowView có những cột thích ứng với khóa chính chỉ định. Như với phương thức Find() , bạn phải gán thuộc tính Sort của DataView của bạn cho khóa chính trước khi gọi phương thức FindRows() .

**Bảng 13.2: những phương thức của DataView**

Phương thức	Kiểu trả về	Mô tả
GetEnumerator()	IEnumerator	Trả lại một bộ đếm cho DataView của bạn.
ToString()	string	Trả lại một chuỗi đại diện cho DataView của bạn.

**Bảng 13.3: những thành viên Liệt kê DataRowState**

HÀNG SỐ	MÔ TẢ
Added	Một hàng mới thêm
CurrentRows	Những hàng hiện thời, bao gồm không thay đổi, hàng được thêm, và những hàng đã sửa đổi.
Deleted	một hàng bị xóa
ModifiedCurrent	một hàng đã được sửa đổi
ModifiedOriginal	hàng nguyên thủy trước khi sửa đổi
None	Không phù hợp với bất kỳ hàng nào trong DataTable.
OriginalRows	Những hàng nguyên bản, bao gồm những hàng không thay đổi và đã bị xóa .
Unchanged	Một hàng mà không bị sửa đổi.

Một trong số những sự kiện DataView là ListChanged. Nó phát khởi khi danh sách được quản lý bởi DataView của bạn thay đổi. Bộ xử lý sự kiện của nó là ListChangedEventHandler.

[Bảng 13.3 cho thấy những thành viên liệt kê của System.Data.DataViewRowState . Sự liệt kê này được sử dụng với thuộc tính RowState của một DataTable; thuộc tính này được dùng để chỉ rõ những hàng được xem bởi DataView được lọc bởi DataRowState của chúng.](#)

### **TAO VÀ SỬ DỤNG MỘT ĐỐI TƯỢNG DATAVIEW:**

Trong mục này, bạn sẽ học cách lọc và phân loại những hàng như thế nào với một đối tượng DataView. Bạn tạo ra một đối tượng DataView sử dụng một trong số những bộ khởi dựng sau đây :

```
DataView()  
DataView(DataTable myDataTable)  
DataView(DataTable myDataTable, string filterExpression, string sortExpression,  
DataRowState rowState)
```

#### **VỚI:**

- myDataTable chỉ định DataTable mà DataView của bạn liên hệ đến. DataView của bạn sẽ đọc những hàng từ DataTable này. Thuộc tính Table của DataView của bạn được gán tới myDataTable.
- filterExpression chỉ định một chuỗi chứa biểu thức bạn muốn dùng để lọc những hàng . Thuộc tính RowFilter của DataView của bạn gán tới filterExpression.
- sortExpression chỉ định một chuỗi chứa biểu thức bạn muốn dùng để phân loại những hàng . Thuộc tính Sort của DataView của bạn được gán tới sortExpression.

- rowState chỉ định một bộ lọc bổ sung ứng dụng vào những hàng; Ví dụ sau đây tạo ra và cư trú một DataTable có tên customersDT chứa những hàng từ bảng Customers:

Trước khi bạn tạo ra một DataView, trước tiên bạn cần một DataTable từ đó để đọc những hàng. Ví dụ sau đây tạo ra và cư trú một DataTable có tên customersDT chứa những hàng từ bảng Customers:

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT CustomerID, CompanyName, Country " +
    "FROM Customers";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet, "Customers");
mySqlConnection.Close();
DataTable customersDT = myDataSet.Tables["Customers"];
```

Chúng ta hãy cho là bạn muốn lọc những hàng ở CustomersDT để xem chỉ những khách hàng trong UK. Biểu thức chuỗi lọc của các sẽ là

```
string filterExpression = "Country = 'UK'";
```

**Ghi chú:**

Sắp xếp ASC : thứ tự tăng dần. sắp xếp DESC theo thứ tự giảm dần.

Cuối cùng, chúng ta hãy cho là bạn chỉ muốn xem những hàng nguyên bản trong DataView; do đó bạn gán bộ lọc trạng thái hàng của bạn tới DataViewRowState.OriginalRows:

```
DataViewRowState rowStateFilter = DataViewRowState.OriginalRows;
```

**Ghi chú:**

Theo mặc định là DataViewRowState.CurrentRows, bao gồm những hàng trong DataView của bạn mà DataViewRowState là không thay đổi, được thêm vào, và đang được sửa đổi.

Ví dụ sau đây tạo ra một đối tượng DataView có tên customersDV và chuyển customersDT, filterExpression, sortExpression, và rowStateFilter đến bộ khởi dựng DataView:

```
DataView customersDV =
    new DataView(
        customersDT, filterExpression, sortExpression, rowStateFilter
    );
```

You can also create a DataView and set the Table, RowFilter, Sort, and RowStateFilter properties individually. For example:

```
DataView customersDV = new DataView();
customersDV.Table = customersDT;
customersDV.RowFilter = filterExpression;
customersDV.Sort = sortExpression;
customersDV.RowStateFilter = rowStateFilter;
```



Một DataView cất giữ những hàng như những đối tượng DataRowView, và những hàng được đọc từ những đối tượng DataRow được cất giữ trong DataTable nằm bên dưới. Ví dụ sau đây sử dụng một vòng foreach để hiển thị những đối tượng DataRowView trong DataView customersDV :

```
foreach (DataRowView myDataRowView in customersDV)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}
```

Ghi nhớ rằng myDataRowView[count] trả về giá trị số chỉ vị trí của cột được chỉ rõ bởi bộ đếm. Chẳng hạn, myDataRowView[0] Trả lại giá trị của chỉ vị trí cột CustomerID. Bạn sẽ học nhiều hơn về lớp DataRowView sau trong mục "[Lớp DataRowView](#)."

[Liệt kê 13.1 trình bày một chương trình sử dụng những ví dụ mã trước đây.](#)

#### Danh sách 13.1: USINGDATAVIEW.CS

```
/*
    UsingDataView.cs illustrates the use of a DataView object to
    filter and sort rows
*/

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

class UsingDataView
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT CustomerID, CompanyName, Country " +
            "FROM Customers";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.Fill(myDataSet, "Customers");
        mySqlConnection.Close();
        DataTable customersDT = myDataSet.Tables["Customers"];
        // set up the filter and sort expressions
        string filterExpression = "Country = 'UK'";
        string sortExpression = "CustomerID ASC, CompanyName DESC";
        DataViewRowState rowStateFilter = DataViewRowState.OriginalRows;
```



```

// create a DataView object named customersDV
DataView customersDV = new DataView();
customersDV.Table = customersDT;
customersDV.RowFilter = filterExpression;
customersDV.Sort = sortExpression;
customersDV.RowStateFilter = rowStateFilter;

// display the rows in the customersDV DataView object
foreach (DataRowView myDataRowView in customersDV)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}
}
}

```

Chú ý rằng những hàng trong customersDV đã được lọc với Country là UK, và những hàng kết quả sẽ được sắp xếp theo CustomerID. Đầu ra chương trình này như sau:

```

AROUT
Around the Horn
UK

BSBEV
B's Beverages
UK

CONSH
Consolidated Holdings
UK

EASTC
Eastern Connection
UK

ISLAT
Island Trading
UK
NORTS
North/South
UK

SEVES
Seven Seas Imports
UK

```

## **Sử dụng giải thuật sắp xếp mặc định**

Nếu bạn muốn phân nhóm những đối tượng DataRowView trong DataView của bạn dựa vào khóa chính DataTable của bạn , Bạn có thể sử dụng một phím tắt. Thay vì gán thuộc tính Sort của DataView của bạn , Bạn

gán thuộc tính `PrimaryKey` của `DataTable` và sau đó gán thuộc tính `ApplyDefaultSort` của `DataView` của bạn tới `true`.

Thuộc tính `Sort` của `DataView` của bạn rồi sẽ tự động được gán tới khóa chính của `DataTable` của bạn. Điều này gây ra những đối tượng `DataRowView` trong `DataView` của bạn sẽ được sắp xếp trong thứ tự tăng dần dựa vào giá trị cột khóa chính .

Chúng ta hãy xem xét một ví dụ. bộ mã sau đây gán những thuộc tính `PrimaryKey` của `DataTable` - `customersDT` tới `DataColumn` - `CustomerID` :

```
customersDT.PrimaryKey =  
    new DataColumn[]  
    {  
        customersDT.Columns["CustomerID"]  
    };
```

Ví dụ kế tiếp đặt thuộc tính `ApplyDefaultSort` của `customersDV` tới `true` :

```
customersDV.ApplyDefaultSort = true;
```

Thuộc tính `Sort` của `customersDV` rồi được gán tới `CustomerID`, nó gây ra những đối tượng `DataRowView` sẽ được sắp xếp theo giá trị tăng dần của `CustomerID`.

**Ghi chú** Bạn sẽ tìm thấy những ví dụ mã trong mục này trong chương trình `UsingDefaultSort.cs` . Danh sách bị bỏ qua trong sách này cho ngắn gọn.

## **Thực hiện sự lọc nâng cao:**

Thuộc tính `RowFilter` của một `DataView` tương tự với mệnh đề `WHERE` trong một sự phát biểu `SELECT`. Do đó bạn có thể sử dụng những biểu thức lọc rất mạnh trong `DataView` của bạn. Ví dụ, bạn có thể sử dụng những chức năng tổng thể `AND`, `OR`, `NOT`, `IN`, `LIKE`, những toán tử so sánh , những toán tử số học, những ký tự đại diện ( `*` và `%` ) .

**Ghi chú** : Chi tiết về cách sử dụng những biểu thức bộ lọc như vậy trong những đối tượng `DataView` của bạn , tham chiếu tới thuộc tính `DataColumn.Expression` trong tài liệu trực tuyến .NET.

Đây là một ví dụ đơn giản sử dụng toán tử `LIKE` và ký tự đại diện (%) phần trăm để lọc những hàng với `CustomerName` bắt đầu với `Fr`:

```
string filterExpression = "CompanyName LIKE 'Fr%'";  
customersDV.RowFilter = filterExpression;
```

Chú ý : chuỗi `Fr %` được đặt trong những lời trích dẫn đơn- mà bạn phải làm cho tất cả các kí tự chuỗi. Khi mã này thay thế mã hiện hữu trong chương trình `UsingDataView.cs` được trình bày trước trong Danh sách 13.1, đầu ra như sau:

```
FRANK  
Frankenversand  
Germany  
  
FRANR  
France restauration  
France
```

**Ghi nhớ :** tôi có thực hiện sự thay đổi này trong chương trình UsingDataView2.cs ( Danh sách được bỏ qua trong sách này cho ngắn gọn). hãy thoải mái tự do khảo sát và chạy chương trình này

### **LỚP DATAROWVIEW:**

Những hàng trong một đối tượng DataView được cất giữ như những đối tượng của lớp DataRowView. Một đối tượng DataRowView cung cấp sự truy cập đối tượng DataRow nằm bên dưới trong DataTable. Khi bạn khảo sát và soạn thảo nội dung của một DataRowView, Bạn thật sự đang làm việc với DataRow nằm bên dưới. Bạn phải nhớ đến điều này khi làm việc với những đối tượng DataRowView. Bảng 13.4 cho thấy một số những thuộc tính DataRowView, và Bảng 13.5 cho thấy một số những phương pháp DataRowView.

**Bảng 13.4: những thuộc tính DataRowView**

Thuộc tính	Kiểu dữ liệu	Mô tả
DataRowView	DataRowView	Lấy DataRowView mà DataRowView thuộc về.
IsEdit	bool	lấy một giá trị bool cho biết liệu DataRowView ( Và do đó DataRow nằm bên dưới) đang trong chế độ soạn thảo.
IsNew	bool	Lấy một giá trị bool cho biết liệu có phải DataRowView vừa mới được thêm vào.
Row	DataRow	Lấy DataRow nằm bên dưới mà đang được xem từ DataTable.
RowVersion	DataRowVersion	Lấy DataRowVersion của DataRow nằm bên dưới . Những thành viên của liệt kê <a href="#">System.Data.DataRowVersion</a> là: <ul style="list-style-type: none"><li>■ Current : chỉ DataRow chứa những giá trị hiện thời.</li><li>■ Default: chỉ DataRow chứa những giá trị mặc định.</li><li>■ Original: chỉ DataRow chứa những giá trị nguyên bản.</li><li>■ Proposed: chỉ DataRow chứa những giá trị dự định.</li></ul>

**Bảng 13.5: những phương thức của DataRowView**

Phương thức	Kiểu trả về	Mô tả
BeginEdit()	void	Bắt đầu sự soạn thảo của DataRowView trong DataView của bạn, Và do đó cũng bắt đầu sự soạn thảo của DataRow bên dưới trong DataTable của bạn. Rồi Bạn soạn thảo DataRow này thông qua DataRowView.
CancelEdit()	void	hủy bỏ sự soạn thảo của DataRowView trong DataView của bạn, Và bởi vậy cũng hủy bỏ soạn thảo của DataRow nằm bên dưới trong cơ sở dữ liệu.
CreateChildView()	DataRowView	Bị quá tải. trả về một DataRowView cho DataTable con, nếu có.
Delete()	void	Xóa DataRowView trong DataView của bạn. Sự xóa của DataRow nằm bên dưới không được giao phó trong DataTable cho đến khi bạn gọi phương thức AcceptChanges() của DataTable của bạn. Bạn có thể hủy bỏ sự xóa bởi gọi phương thức RejectChanges() của DataTable của bạn, và cũng hủy bất kỳ sự thêm không giao phó hay những sự sửa đổi nào.
EndEdit()	void	Kết thúc sự soạn thảo của một DataRowView.

### **SỰ TÌM KIẾM NHỮNG ĐỐI TƯỢNG DataRowView TRONG DataView:**

Bạn có thể tìm chỉ số của một DataRowView trong một DataView sử dụng phương thức Find() của một DataView. Bạn cũng có thể lấy một mảng của những đối tượng DataRowView sử dụng phương thức FindRows() của một DataView. Bạn sẽ học cách sử dụng những phương thức Find() và FindRows() trong mục này.

### **TÌM CHỈ SỐ CỦA MỘT DataRowView SỬ DỤNG PHƯƠNG THỨC Find():**

Phương thức tìm kiếm trả lại chỉ số của DataRowView với khóa chính được chỉ rõ trong DataView của bạn. giá trị Int được trả lại bởi phương thức này là chỉ số của DataRowView nếu được tìm thấy; nếu không -1 được trả về.

Để tìm thấy chỉ số đúng, đầu tiên bạn phải đặt thuộc tính Sort của DataView của bạn tới Sort trên khóa chính. Chẳng hạn, nếu bạn muốn tìm một DataRowView đặt cơ sở trên CustomerID, Bạn phải đặt thuộc tính Sort của DataView của bạn tới CustomerID, CustomerID ASC, hay CustomerID DESC:

```
string sortExpression = "CustomerID";  
customersDV.Sort = sortExpression;
```

Giả thiết rằng những đối tượng DataRowView được lọc trong customersDV như sau:

```
AROUT  
Around the Horn  
UK  
  
BSBEV  
B's Beverages  
UK  
  
CONSH  
Consolidated Holdings  
UK  
  
EASTC  
Eastern Connection  
UK  
  
ISLAT  
Island Trading  
UK  
  
NORTS  
North/South  
UK  
  
SEVES  
Seven Seas Imports  
UK
```

Ví dụ sau đây gọi phương pháp Find() để tìm chỉ số của DataRowView trong customersDV với một CustomerID là BSBEV:

```
int index = customersDV.Find("BSBEV");
```

Bởi vì BSBEV xuất hiện tại vị trí có chỉ số 1, phương thức Find() trả lại 1.

Ghi chú: Những đối tượng DataRowView trong một DataView khởi đầu tại chỉ số 0. Bởi vậy, BSBEV xuất

hiện tại chỉ số 1.

## ***Tìm những đối tượng DataRowView sử dụng phương thức FindRows():***

Phương thức FindRows() của một DataView tìm và trả lại một mảng của những đối tượng DataRowView mà có cột khóa chính phù hợp với khóa chính trong DataView của bạn. Nếu không có hàng nào được tìm thấy, thì mảng trả về sẽ không có phần tử nào, và thuộc tính Length (chiều dài) của mảng sẽ là 0.

Để tìm những đối tượng DataRowView sử dụng phương thức FindRows(), đầu tiên bạn phải đặt thuộc tính Sort của DataView của bạn là lọc trên khóa chính. Chẳng hạn, nếu bạn muốn tìm những đối tượng DataRowView đặt cơ sở trên CustomerID, Bạn phải đặt thuộc tính Sort của DataView của bạn là CustomerID, CustomerID ASC, hay CustomerID DESC:

```
string sortExpression = "CustomerID";
customersDV.Sort = sortExpression;
```

Ví dụ sau đây gọi phương thức FindRows() để tìm DataRowView có CustomerID là BSBEV:

```
DataRowView[] customersDRVs = customersDV.FindRows("BSBEV");
```

Vì chỉ có một đối tượng phù hợp, mảng customersDRVs sẽ chứa đựng một DataRowView.

Danh sách 13.2 trình bày một chương trình sử dụng những phương thức Find() Và FindRows()

### Danh sách 13.2: FINDINGDATAROWVIEWS.CS

```
/*
    FindingDataRowViews.cs illustrates the use of the Find() and
    FindRows() methods of a DataView to find DataRowView objects
*/

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

class FindingDataRowViews
{
    public static void Main()
    {
        SqlConnection mySqlConnection =
            new SqlConnection(
                "server=localhost;database=Northwind;uid=sa;pwd=sa"
            );
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
        mySqlCommand.CommandText =
            "SELECT CustomerID, CompanyName, Country " +
            "FROM Customers";
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        DataSet myDataSet = new DataSet();
        mySqlConnection.Open();
        mySqlDataAdapter.Fill(myDataSet, "Customers");
        mySqlConnection.Close();
        DataTable customersDT = myDataSet.Tables["Customers"];
```

```

// set up the filter and sort expressions
string filterExpression = "Country = 'UK'";
string sortExpression = "CustomerID";
DataRowViewState rowStateFilter = DataRowViewState.OriginalRows;

// create a DataView object named customersDV
DataView customersDV = new DataView();
customersDV.Table = customersDT;
customersDV.RowFilter = filterExpression;
customersDV.Sort = sortExpression;
customersDV.RowStateFilter = rowStateFilter;

// display the rows in the customersDV DataView object
foreach (DataRowView myDataRowView in customersDV)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}

// use the Find() method of customersDV to find the index of
// the DataRowView whose CustomerID is BSBEV
int index = customersDV.Find("BSBEV");
Console.WriteLine("BSBEV found at index " + index + "\n");

// use the FindRows() method of customersDV to find the DataRowView
// whose CustomerID is BSBEV
DataRowView[] customersDRVs = customersDV.FindRows("BSBEV");
foreach (DataRowView myDataRowView in customersDRVs)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}
}
}

```

Mẹo nhỏ: Nếu bạn đang sử dụng một phiên bản trước của NET. SDK, Bạn có lẽ đã gặp lỗi biên dịch sau đây khi biên dịch chương trình này

*FindingDataRowViews .cs(59, 35): lỗi CS0117: 'System.Data.DataView' không chứa  
chứa một định nghĩa cho ' FindRows'*

Nếu bạn gặp lỗi này, biên tập chương trình với Visual Studio .NET

Đầu ra từ chương trình này như sau:

```

AROUT
Around the Horn
UK

```

```

BSBEV
B's Beverages

```

UK

CONSH

Consolidated Holdings

UK

EASTC

Eastern Connection

UK

ISLAT

Island Trading

UK

NORTS

North/South

UK

SEVES

Seven Seas Imports

UK

BSBEV found at index 1

BSBEV

B's Beverages

UK

### **THÊM, SỬA ĐỔI, LOẠI BỎ NHỮNG ĐỐI TƯỢNG *DataRowView* TỪ MỘT *DataView*:**

Sẽ là quan trọng khi biết những đối tượng *DataRowView* trong một *DataView* cung cấp những sự truy cập tới đối tượng *DataRow* nằm bên dưới trong một *DataTable*. Bởi vậy, khi bạn khảo sát và soạn thảo nội dung của một *DataRowView*, Bạn thật sự đang làm việc với *DataRow* nằm bên dưới. Tương tự, khi bạn loại bỏ một *DataRowView*, tức bạn đang loại bỏ *DataRow* nằm bên dưới.

### **THÊM MỘT *DataRowView* VÀO MỘT *DataView*:**

Để thêm một *DataRowView* mới vào một *DataView*, Bạn gọi phương thức *AddNew()* của *DataView* của bạn. Phương thức *AddNew()* trả về một đối tượng *DataRowView* mà bạn gán những giá trị cột cho hàng mới. Ví dụ sau đây gọi phương thức *AddNew()* của *DataView* *customersDV*:

```
DataRowView customerDRV = customersDV.AddNew();  
customerDRV["CustomerID"] = "J7COM";  
customerDRV["CompanyName"] = "J7 Company";  
customerDRV["Country"] = "UK";  
customerDRV.EndEdit();
```

Chú ý sự sử dụng của phương thức *EndEdit()* của *DataRowView* *customerDRV* để kết thúc sự soạn thảo. phương thức *EndEdit()* tạo ra một *DataRow* mới trong *DataTable* nằm bên dưới. Những đối tượng *DataColumn* trong *DataRow* mới sẽ chứa những giá trị cột được chỉ rõ trong mã trước đây.

**Ghi nhớ** Bạn có thể huỷ bỏ sự thêm bởi gọi phương thức *CancelEdit()* của một *DataRowView*.

Bạn có thể lấy *DataRow* nằm bên dưới được thêm vào *DataTable* sử dụng thuộc tính *Row* của một *DataRowView*. Chẳng hạn:



`DataRow customerDR = customerDRV.Row;`

## **Sửa đổi một DataRowView hiện hữu:**

Để bắt đầu sửa đổi một DataRowView hiện hữu trong một DataView, Bạn gọi phương thức BeginEdit() của DataRowView trong DataView của bạn. Ví dụ sau đây gọi phương thức BeginEdit() cho DataRowView đầu tiên trong customersDV:

```
customersDV[0].BeginEdit();
```

Ghi chú: Nhớ rằng những đối tượng DataRowView trong một DataView bắt đầu tại chỉ số 0, và bởi vậy customersDV[0] là DataRowView đầu tiên trong customersDV.

Và rồi Bạn có thể sửa đổi một DataColumn trong DataRow nằm bên dưới thông qua DataRowView.

Ví dụ sau đây đặt DataColumn CompanyName là Widgets Inc.:

```
customersDV[0]["CompanyName"] = "Widgets Inc.";
```

Một khi bạn đã thực hiện xong những sửa đổi của bạn, bạn gọi phương thức EndEdit() để làm cho những sự sửa đổi của bạn bền vững trong DataTable nằm bên dưới. Chẳng hạn:

```
customersDV[0].EndEdit();
```

**Ghi nhớ:** Bạn có thể huỷ bỏ sự sửa đổi bởi gọi phương thức CancelEdit() của một DataRowView.

## **Loại bỏ một DataRowView hiện hữu**

Để loại bỏ một DataRowView hiện hữu từ một DataView, Bạn có thể gọi phương thức Delete() của DataView hoặc DataRowView. Khi gọi phương thức Delete() của một DataView, Bạn thông qua chỉ số của DataRowView mà bạn muốn loại bỏ. Ví dụ sau đây loại bỏ DataRowView thứ hai từ customersDV:

```
customersDV.Delete(1);
```

Khi gọi phương thức Delete() của một DataRowView, Bạn đơn giản gọi phương thức này của DataRowView trong DataView của bạn. ví dụ sau đây loại bỏ DataRowView thứ ba từ customersDV:

```
customersDV[2].Delete();
```

Với bất kỳ trong ba phương thức Delete() này, sự xóa không được giao phó trong DataTable nằm bên dưới cho đến khi bạn gọi phương thức AcceptChanges() của DataTable của bạn. Chẳng hạn:

```
customersDT.AcceptChanges();
```

**Ghi chú:** Bạn có thể gọi phương thức RejectChanges() của một DataTable để huỷ bỏ những sự xóa. Phương thức này sẽ cũng huỷ bỏ bất kỳ sự thêm không giao phó và những sự sửa đổi của những hàng.

Danh sách 13.3 cho thấy một chương trình mà thêm, điều chỉnh, và xóa những đối tượng DataRowView từ một DataView. Chương trình này cũng trình bày những thuộc tính IsNew và IsEdit của những đối tượng DataRowView. Nó cho biết liệu có phải DataRowView là mới và đang được sửa đổi hay không.

**Danh sách 13.3: ADDMODIFYANDREMOVEDATAROWVIEWS.CS**

/\*

AddModifyAndRemoveDataRowViews.cs illustrates how to

```
add, modify, and remove DataRowView objects from a DataView
*/
```

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

```
class AddModifyAndRemoveDataRowViews
{
    public static void DisplayDataRow(
        DataRow myDataRow,
        DataTable myDataTable
    )
    {
        Console.WriteLine("\nIn DisplayDataRow()");
        foreach (DataColumn myDataColumn in myDataTable.Columns)
        {
            Console.WriteLine(myDataColumn + " = " +
                myDataRow[myDataColumn]);
        }
    }
}

public static void Main()
{
    SqlConnection mySqlConnection =
        new SqlConnection(
            "server=localhost;database=Northwind;uid=sa;pwd=sa"
        );
    SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
    mySqlCommand.CommandText =
        "SELECT CustomerID, CompanyName, Country " +
        "FROM Customers";
    SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
    mySqlDataAdapter.SelectCommand = mySqlCommand;
    DataSet myDataSet = new DataSet();
    mySqlConnection.Open();
    mySqlDataAdapter.Fill(myDataSet, "Customers");
    mySqlConnection.Close();
    DataTable customersDT = myDataSet.Tables["Customers"];

    // set up the filter expression
    string filterExpression = "Country = 'UK'";

    // create a DataView object named customersDV
    DataView customersDV = new DataView();
    customersDV.Table = customersDT;
    customersDV.RowFilter = filterExpression;

    // add a new DataRowView (adds a DataRow to the DataTable)
    Console.WriteLine("\nCalling customersDV.AddNew()");
    DataRowView customerDRV = customersDV.AddNew();
    customerDRV["CustomerID"] = "J7COM";
    customerDRV["CompanyName"] = "J7 Company";
    customerDRV["Country"] = "UK";
    Console.WriteLine("customerDRV[\" CustomerID\"] = " +
        customerDRV["CustomerID"]);
}
```

```

Console.WriteLine("customerDRV[" CompanyName"] = " +
    customerDRV["CompanyName"]);
Console.WriteLine("customerDRV[" Country"] = " +
    customerDRV["Country"]);
Console.WriteLine("customerDRV.IsNew = " + customerDRV.IsNew);
Console.WriteLine("customerDRV.IsEdit = " + customerDRV.IsEdit);
customerDRV.EndEdit();
// get and display the underlying DataRow
DataRow customerDR = customerDRV.Row;
DisplayDataRow(customerDR, customersDT);

// modify the CompanyName of customerDRV
Console.WriteLine("\nSetting customersDV[0][" CompanyName"] to Widgets Inc.");
customersDV[0].BeginEdit();
customersDV[0]["CompanyName"] = "Widgets Inc.";
Console.WriteLine("customersDV[0][" CustomerID"] = " +
    customersDV[0]["CustomerID"]);
Console.WriteLine("customersDV[0][" CompanyName"] = " +
    customersDV[0]["CompanyName"]);
Console.WriteLine("customersDV[0].IsNew = " + customersDV[0].IsNew);
Console.WriteLine("customersDV[0].IsEdit = " + customersDV[0].IsEdit);
customersDV[0].EndEdit();

// display the underlying DataRow
DisplayDataRow(customersDV[0].Row, customersDT);

// remove the second DataRowView from customersDV
Console.WriteLine("\ncustomersDV[1][" CustomerID"] = " +
    customersDV[1]["CustomerID"]);
Console.WriteLine("\nCalling customersDV.Delete(1)");
customersDV.Delete(1);
Console.WriteLine("customersDV[1].IsNew = " + customersDV[1].IsNew);
Console.WriteLine("customersDV[1].IsEdit = " + customersDV[1].IsEdit);

// remove the third DataRowView from customersDV
Console.WriteLine("\ncustomersDV[2][" CustomerID"] = " +
    customersDV[2]["CustomerID"]);
Console.WriteLine("\nCalling customersDV[2].Delete()");
customersDV[2].Delete();

// call the AcceptChanges() method of customersDT to
// make the deletes permanent in customersDT
customersDT.AcceptChanges();

// display the rows in the customersDV DataView object
Console.WriteLine("\nDataRowView objects in customersDV:\n");
foreach (DataRowView myDataRowView in customersDV)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}
}
}
}

```

Đầu ra từ chương trình này như sau:

```
Calling customersDV.AddNew()
customerDRV["CustomerID"] = J7COM
customerDRV["CompanyName"] = J7 Company
customerDRV["Country"] = UK
customerDRV.IsNew = True
customerDRV.IsEdit = True
```

```
In DisplayDataRow()
CustomerID = J7COM
CompanyName = J7 Company
Country = UK
```

```
Setting customersDV[0]["CompanyName"] to Widgets Inc.
customersDV[0]["CustomerID"] = AROUT
customersDV[0]["CompanyName"] = Widgets Inc.
customersDV[0].IsNew = False
customersDV[0].IsEdit = True
```

```
In DisplayDataRow()
CustomerID = AROUT
CompanyName = Widgets Inc.
Country = UK
```

```
customersDV[1]["CustomerID"] = BSBEV
```

```
Calling customersDV.Delete(1)
customersDV[1].IsNew = False
customersDV[1].IsEdit = False
```

```
customersDV[2]["CustomerID"] = EASTC
```

```
Calling customersDV[2].Delete()
```

DataRowView objects in customersDV:

```
AROUT
Widgets Inc.
UK
CONSH
Consolidated Holdings
UK
```

```
ISLAT
Island Trading
UK
```

```
NORTS
North/South
UK
```

```
SEVES
Seven Seas Imports
UK
```

```
J7COM
```

### **TAO NHỮNG ĐỐI TƯỢNG *DataView* CON:**

Bạn có thể tạo ra một *DataView* con từ một *DataRowView* cha sử dụng phương thức *CreateChildView()*. Và rồi Bạn có thể xem những đối tượng *DataRowView* *DataView* con. Để gọi phương thức *CreateChildView()*, đầu tiên bạn phải thêm một *DataRelation* vào *Dataset* mà định nghĩa một mối quan hệ giữa hai đối tượng *DataTable* nằm bên dưới. (xem Chương 12, " định hướng và sửa đổi Dữ liệu liên quan ", để có thông tin về những đối tượng *DataRelation*.)

Chúng ta hãy xem xét một ví dụ. Giả thiết bạn có hai đối tượng *DataTable* có tên *customersDT* và *ordersDT* . Cũng giả thiết bạn có thêm *DataRelation* sau đây vào *Dataset* mà định nghĩa một mối quan hệ giữa *customersDT* và *ordersDT*:

```
DataRelation customersOrdersDataRel =  
    new DataRelation(  
        "CustomersOrders",  
        customersDT.Columns["CustomerID"],  
        ordersDT.Columns["CustomerID"]  
    );  
myDataSet.Relations.Add(  
    customersOrdersDataRel  
);
```

Cuối cùng, giả thiết bạn có một *DataView* có tên *customersDV* để xem những khách hàng có một cột *Country* là UK. Và rồi Bạn có thể gọi phương thức *CreateChildView()* từ một *DataRowView* trong *customersDV* để tạo ra một *DataView* con; chú ý rằng tên của *DataRelation* (*CustomersOrders*) được chuyển cho phương thức *CreateChildView()*:

```
DataView ordersDV = customersDV[0].CreateChildView("CustomersOrders");
```

*DataView* *ordersDV* cho phép bạn truy cập những hàng con từ *DataTable* *ordersDT*. Đối tượng cha trong ví dụ này là *DataRowView* đầu tiên từ *customersDV* có một *CustomerID* là AROUT. *DataView* *ordersDV* con chứa những đối tượng *DataRowView* với những chi tiết của đơn đặt cho khách hàng AROUT.

Ghi chú : Phương thức *CreateChildView()* bị quá tải. Phiên bản khác của phương thức này chấp nhận một đối tượng *DataRelation* như tham số.

liệt kê 13.4 trình bày một chương trình ví dụ đầy đủ

#### Danh sách 13.4: CREATECHILDDATAVIEW.CS

```
/*  
    CreateChildDataView.cs illustrates how to create a  
    child DataView  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class CreateChildDataView  
{  
    public static void Main()
```

```

{
SqlConnection mySqlConnection =
    new SqlConnection(
        "server=localhost;database=Northwind;uid=sa;pwd=sa"
    );
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "SELECT CustomerID, CompanyName, Country " +
    "FROM Customers;" +
    "SELECT OrderID, CustomerID " +
    "FROM Orders;";
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
mySqlDataAdapter.Fill(myDataSet);
mySqlConnection.Close();
myDataSet.Tables["Table"].TableName = "Customers";
myDataSet.Tables["Table1"].TableName = "Orders";
DataTable customersDT = myDataSet.Tables["Customers"];
DataTable ordersDT = myDataSet.Tables["Orders"];
// add a DataRelation object to myDataSet
DataRelation customersOrdersDataRel =
    new DataRelation(
        "CustomersOrders",
        customersDT.Columns["CustomerID"],
        ordersDT.Columns["CustomerID"]
    );
myDataSet.Relations.Add(
    customersOrdersDataRel
);

// create a DataView object named customersDV
DataView customersDV = new DataView();
customersDV.Table = customersDT;
customersDV.RowFilter = "Country = 'UK'";
customersDV.Sort = "CustomerID";

// display the first row in the customersDV DataView object
Console.WriteLine("Customer:");
for (int count = 0; count < customersDV.Table.Columns.Count; count++)
{
    Console.WriteLine(customersDV[0][count]);
}

// create a child DataView named ordersDV that views
// the child rows for the first customer in customersDV
DataView ordersDV = customersDV[0].CreateChildView("CustomersOrders");

// display the child rows in the customersDV DataView object
Console.WriteLine("\nOrderID's of the orders placed by this customer:");
foreach (DataRowView ordersDRV in ordersDV)
{
    Console.WriteLine(ordersDRV["OrderID"]);
}
}

```

}

Đầu ra từ chương trình này như sau:

Customer:  
AROUT  
Around the Horn  
UK

OrderID's of the orders placed by this customer:

10355  
10383  
10453  
10558  
10707  
10741  
10743  
10768  
10793  
10864  
10920  
10953  
11016

### **LỚP *DataViewManager*:**

Một *DataViewManager* cho phép bạn tập trung quản lý nhiều đối tượng *DataView* trong một *Dataset*. Một *DataViewManager* cũng cho phép bạn tạo ra những đối tượng *DataView* trong thời gian chạy. Bảng 13.6 cho thấy một số những thuộc tính *DataViewManager*.

**Bảng 13.6: những thuộc tính *DataViewManager***

Thuộc tính	Kiểu dữ liệu	Mô tả
<i>DataSet</i>	<i>DataSet</i>	lấy hay gán <i>Dataset</i> được dùng bởi <i>DataViewManager</i> của bạn.
<i>DataViewSettings</i>	<i>DataViewSettingCollection</i>	Lấy <i>DataViewSettingCollection</i> cho mỗi <i>DataTable</i> trong <i>Dataset</i> của bạn. Một <i>DataViewSettingCollection</i> giúp bạn truy cập đến những thuộc tính của <i>DataView</i> cho mỗi <i>DataTable</i> .

Một trong số những phương thức của *DataViewManager* là *CreateDataView()*. Nó tạo ra một *DataView* mới cho *DataTable* được chỉ định. *DataTable* được chuyển qua như một tham số tới phương thức *CreateDataView()*. Kiểu trả về của nó là *DataView*.

Một trong số những sự kiện *DataViewManager* là *ListChanged*. Nó phát khởi khi danh sách được quản lý bởi một *DataView* trong *DataViewManager* của bạn thay đổi. Bộ xử lý sự kiện của nó là *ListChangedEventHandler*.

### **Tạo ra và sử dụng một Đối tượng *DataViewManager***

Tạo ra một *DataViewManager*, Bạn sử dụng một trong số bộ khởi dựng sau đây:

*DataViewManager*()  
*DataViewManager*(*DataSet myDataSet*)

Với **myDataSet** chỉ định *Dataset* được dùng bởi đối tượng *DataViewManager*. Nó gán thuộc tính *Dataset* của



đối tượng DataViewManager mới tới myDataSet.

Chúng ta hãy xem xét một ví dụ về tạo ra và sử dụng một DataViewManager. Giả thiết bạn có một DataSet có tên myDataSet, có chứa một DataTable được lưu trữ với những hàng từ bảng Customers. Ví dụ sau đây tạo ra một đối tượng DataViewManager có tên myDVM, MyDataSet chuyển qua tới bộ khởi dựng:

```
DataViewManager myDVM = new DataViewManager(myDataSet);
```

Ví dụ kế tiếp thiết đặt những thuộc tính Sort và RowFilter mà sẽ được sử dụng sau đó khi một DataView cho DataTable Customers được tạo ra :

```
myDVM.DataViewSettings["Customers"].Sort = "CustomerID";  
myDVM.DataViewSettings["Customers"].RowFilter = "Country = 'UK'";
```

**Ghi nhớ** : mã trước đây không thật sự tạo ra một DataView; nó đơn thuần gán những thuộc tính của bất kỳ DataView nào được tạo ra trong tương lai, mà xem những hàng từ DataTable Customers.

Ví dụ sau đây thật sự tạo ra một DataView bởi sự gọi phương thức CreateDataView() của DataViewManager myDVM, chuyển DataTable customersDT tới CreateDataView():

```
DataView customersDV = myDVM.CreateDataView(customersDT);
```

Thuộc tính Sort và RowFilter của DataView customersDV được gán tới CustomerID và Country = ' UK' tương ứng. Đây là những sự thiết đặt tương tự như những thiết đặt trước đó trong thuộc tính DataViewSettings.

Danh sách 13.4 Trình bày một ví dụ đầy đủ về tạo và sử dụng DataViewManager đã học trong mục này.

[Danh sách 13.4 Một : USINGDATAVIEWMANAGER.CS](#)

```
/*  
UsingDataViewManager.cs illustrates the use of a  
DataViewManager object  
*/  
  
using System;  
using System.Data;  
using System.Data.SqlClient;  
  
class UsingDataViewManager  
{  
    public static void Main()  
    {  
        SqlConnection mySqlConnection =  
            new SqlConnection(  
                "server=localhost;database=Northwind;uid=sa;pwd=sa"  
            );  
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();  
        mySqlCommand.CommandText =  
            "SELECT CustomerID, CompanyName, Country " +  
            "FROM Customers";  
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();  
        mySqlDataAdapter.SelectCommand = mySqlCommand;  
        DataSet myDataSet = new DataSet();  
        mySqlConnection.Open();  
        mySqlDataAdapter.Fill(myDataSet, "Customers");  
        mySqlConnection.Close();  
        DataTable customersDT = myDataSet.Tables["Customers"];
```

```

// create a DataViewManager object named myDVM
DataViewManager myDVM = new DataViewManager(myDataSet);

// set the Sort and RowFilter properties for the Customers DataTable
myDVM.DataViewSettings["Customers"].Sort = "CustomerID";
myDVM.DataViewSettings["Customers"].RowFilter = "Country = 'UK'";

// display the DataViewSettingCollectionString property of myDVM
Console.WriteLine("myDVM.DataViewSettingCollectionString = " +
    myDVM.DataViewSettingCollectionString + "\n");

// call the CreateDataView() method of myDVM to create a DataView
// named customersDV for the customersDT DataTable
DataView customersDV = myDVM.CreateDataView(customersDT);

// display the rows in the customersDV DataView object
foreach (DataRowView myDataRowView in customersDV)
{
    for (int count = 0; count < customersDV.Table.Columns.Count; count++)
    {
        Console.WriteLine(myDataRowView[count]);
    }
    Console.WriteLine("");
}
}
}

```

Đầu ra của chương trình này như sau:

```

myDVM.DataViewSettingCollectionString =
<DataViewSettingCollectionString>
<Customers Sort="CustomerID" RowFilter="Country = 'UK'"
RowStateFilter="CurrentRows"/>
</DataViewSettingCollectionString>

```

AROUT

Around the Horn

UK

BSBEV

B's Beverages

UK

CONSH

Consolidated Holdings

UK

EASTC

Eastern Connection

UK

ISLAT

Island Trading

UK

NORTS

North/South

UK

SEVES

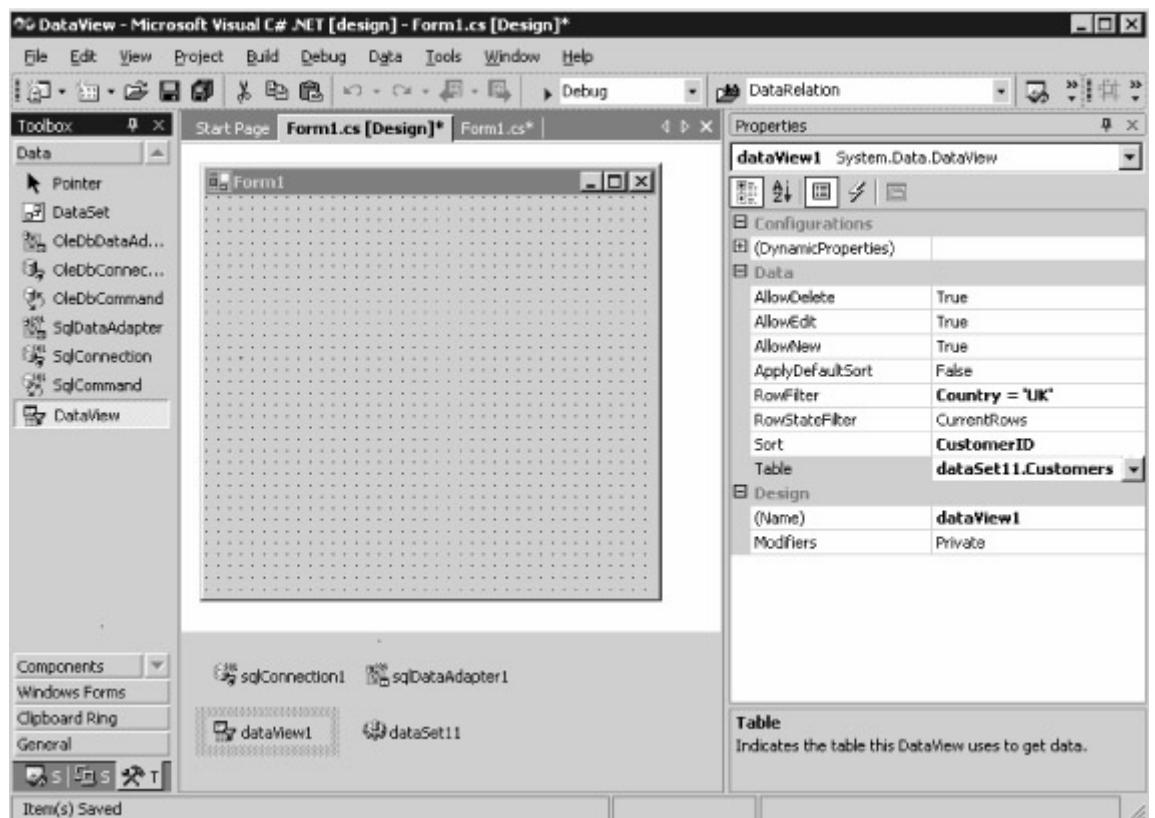
Seven Seas Imports

UK

## **TAO MỘT DataView SỬ DỤNG Visual Studio .NET:**

Trong mục này, bạn sẽ học cách tạo ra một DataView như thế nào sử dụng Visual Studio .NET (VS .NET). Bạn có thể theo những bước được mô tả trong mục này:

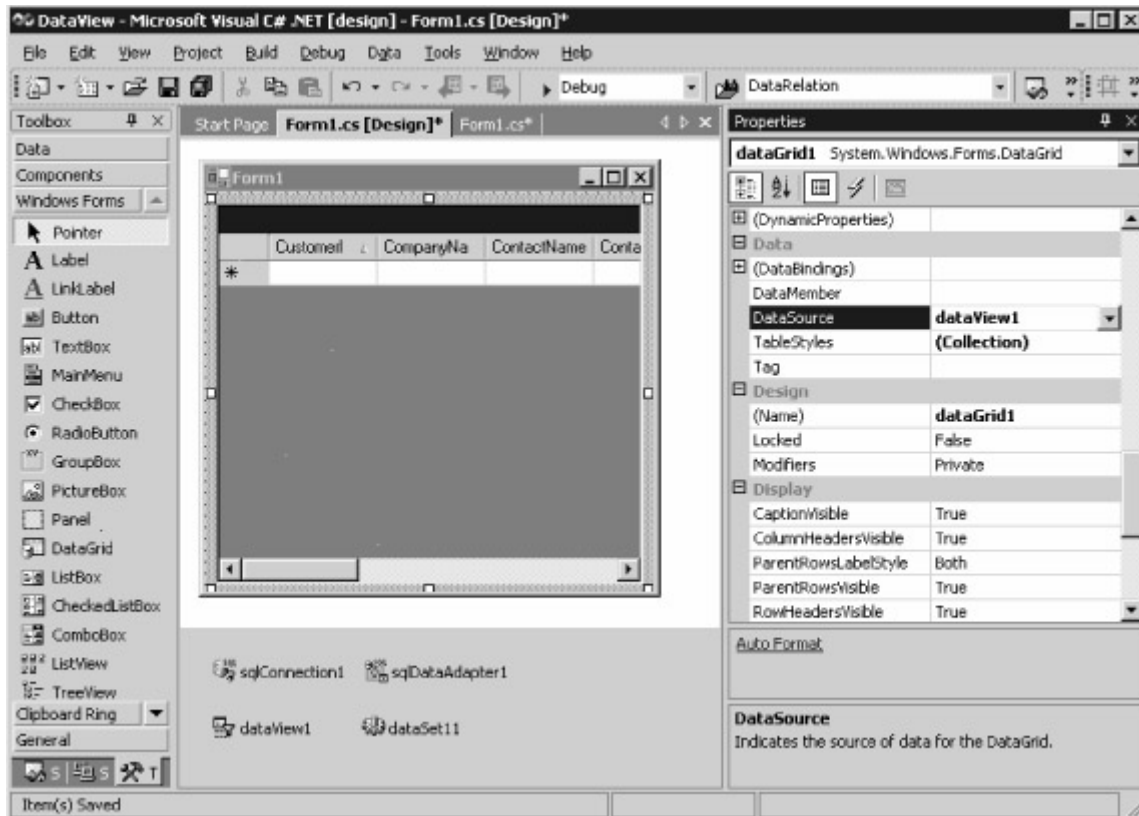
1. Mở VS .NET và tạo ra một ứng dụng Windows mới đặt tên myDataView.
2. hiển thị Server Explorer, kết nối tới cơ sở dữ liệu Northwind của bạn, và sự kéo bảng Customers tới form của bạn. Việc này tạo ra một đối tượng SqlConnection có tên sqlConnection1 và một đối tượng SqlDataAdapter có tên sqlDataAdapter1. Những đối tượng này được đặt trong khay bên dưới form của bạn.
3. Thay đổi thuộc tínhConnectionString của sqlConnection1 để kết nối tới cơ sở dữ liệu Northwind của bạn. Nhớ thêm một chuỗi con chứa mật khẩu ( Pwd= sa, hay tương tự).
4. Kích vào đối tượng sqlDataAdapter1 trong form của bạn, và rồi kích liên kết Generate Dataset tại đáy của cửa sổ những thuộc tính cho sqlDataAdapter1. Chấp nhận những mặc định trong hộp thoại, và kích nút Ok để tạo ra một đối tượng Dataset có tên dataSet11.
5. Kéo một đối tượng DataView từ tab Data của Toolbox đến form của bạn. Việc này tạo ra một đối tượng DataView có tên dataView1.
6. Gán thuộc tính Table của đối tượng dataView1 của bạn tới dataSet11.Customers sử dụng danh sách sổ xuống ở bên phải của thuộc tính Table; đặt thuộc tính RowFilter tới Country =' UK'; và đặt thuộc tính Sort tới CustomerID. xem [Hình 13.1](#).



Thiết đặt những thuộc tính của DataView1

7. Kéo một điều khiển DataGrid từ những tab Windows Form của Toolbox đến form của bạn. Việc này tạo ra một đối tượng DataGrid có tên dataGrid1.

8. Gán thuộc tính DataSource của dataGrid1 tới dataView1 sử dụng danh sách sổ xuống ở bên phải của thuộc tính DataSource, như trong [Hình 13.2](#). Điều này kết buộc dữ liệu được cất giữ trong dataView1 vào dataGrid1 và cho phép dataGrid1 truy cập bất kỳ dữ liệu nào được cất giữ trong dataView1.



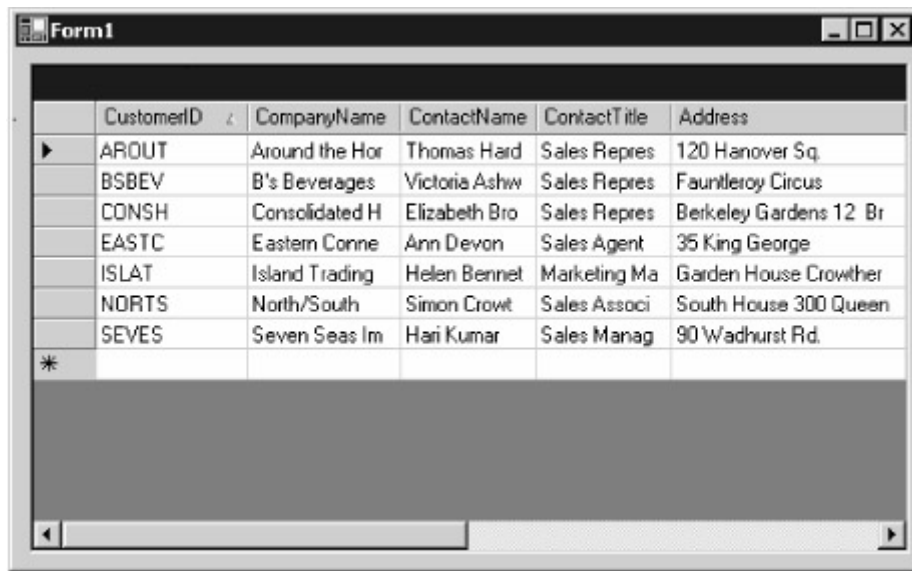
[Hình 13.2: gán những thuộc tính của dataGrid1](#)

9. Chọn View - Code và gán phương thức Form1() của form của bạn với mã sau

```
public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();

    // call the Fill() method of sqlDataAdapter1
    // to populate dataSet11 with a DataTable named
    // Customers
    sqlDataAdapter1.Fill(dataSet11, "Customers");
}
```

Biên dịch và chạy form của bạn bằng cách nhấn Ctrl + F5. [Hình 13.3 cho thấy sự vận hành của form. Chú ý thông tin trong form đến từ DataView mà bạn đã tạo ra.](#)



	CustomerID	CompanyName	ContactName	ContactTitle	Address
▶	AROUT	Around the Hor	Thomas Hard	Sales Repres	120 Hanover Sq.
	BSBEV	B's Beverages	Victoria Ashw	Sales Repres	Fauntleroy Circus
	CONSH	Consolidated H	Elizabeth Bro	Sales Repres	Berkeley Gardens 12 Br
	EASTC	Eastern Conne	Ann Devon	Sales Agent	35 King George
	ISLAT	Island Trading	Helen Bennet	Marketing Ma	Garden House Crowther
	NORTS	North/South	Simon Crowt	Sales Associ	South House 300 Queen
	SEVES	Seven Seas Im	Hari Kumar	Sales Manag	90 Wadhurst Rd.
*					

### **TÓM TẮT:**

Trong chương này, bạn đã học cách sử dụng những đối tượng DataView để lọc và sắp xếp những hàng như thế nào. Lợi thế của một DataView là bạn có thể kết buộc nó tới một thành phần trực quan như một điều khiển DataGrid.

Một DataView cất giữ những bản sao của những hàng trong một DataTable như những đối tượng DataRowView. Những đối tượng DataRowView cung cấp sự truy cập tới những đối tượng DataRow nằm bên dưới trong một DataTable. Bởi vậy, khi bạn khảo sát và sửa đổi nội dung của một DataRowView, tức là bạn thật sự đang làm việc với DataRow nằm bên dưới.

Thuộc tính RowFilter của một DataView tương tự như một mệnh đề WHERE trong một phát biểu SELECT. Do đó bạn có thể sử dụng những biểu thức lọc rất mạnh trong DataView của bạn. Ví dụ, bạn có thể sử dụng AND, OR, NOT, IN, LIKE, những toán tử so sánh, những toán tử số học, những ký tự đại diện (\* và %) và những chức năng tổng thể.

Bạn có thể tìm thấy chỉ số của một DataRowView trong một DataView sử dụng phương thức Find() của một DataView. Bạn cũng có thể lấy một mảng của những đối tượng DataRowView sử dụng phương thức FindRows() của một DataView.

Một DataViewManager cho phép bạn tập trung quản lý nhiều đối tượng DataView trong một Dataset. Một DataViewManager cũng cho phép bạn tạo ra những đối tượng DataView trong thời gian chạy.