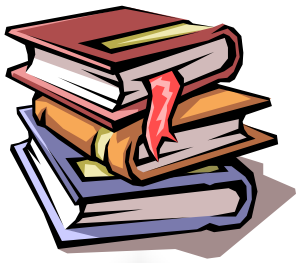
BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ...

KHOA ...



Mastering C# Database Programming

Jason Price

Lời nói đầu

Trước khi bạn đọc tài liệu này, tôi cũng xin giới thiệu chút ít về nguồn gốc của nó. Đây là một tài liệu free trên mạng tựa đề “Mastering C# Database Programming” của Jason Price , nội dung tập trung chủ yếu vào lập trình cơ sở dữ liệu với ngôn ngữ lập trình C#, sásh trình bày chi tiết cặn kẽ và có kèm theo những ví dụ sinh động giúp người đọc dễ dàng nắm bắt các khía cạnh của vấn đề.

Có lẽ ý đồ của tác giả khi viết sách này là để dành riêng cho những người đã có hiểu biết về ngôn ngữ lập trình nói chung và nhất là ngôn ngữ lập trình C# nói riêng. Nếu bạn chưa biết gì về ngôn ngữ lập trình C# ,có lẽ bạn nên nghiên cứu nó trước khi đọc tài liệu này.

Tài liệu về ngôn ngữ lập trình C# viết bằng tiếng việt có vài cuốn trên mạng, nhưng những tài liệu chuyên sâu về lập trình Windows Form, lập trình cơ sở dữ liệu với C# dường như quá hiếm hoi. Trong khi những tài này viết bằng tiếng anh thì rất nhiều, đối với những bạn yêu thích lập trình mà kém ngọai ngữ thì đành bó tay.

Đây là một tài liệu viết bằng tiếng anh , tuy rằng trình độ anh ngữ của tôi cũng có giới hạn, nhưng vì “đam mê lập trình” nên tôi cũng quyết định dịch tài liệu này, một là để đóng góp một chút công sức cho những bạn có cùng đam mê như tôi, hai là rất mong mỏi những bậc thầy trong làng IT chỉ giáo cho những gì chưa rõ ràng hoặc có sai xót trong bản dịch để anh em cùng nhau học hỏi

Chân thànhcảm ơn
CVL

@@

Chương 1: Giới thiệu về Lập trình Cơ sở dữ liệu với ADO.NET

Tổng quan

Một Cơ sở dữ liệu là một tập hợp có tổ chức của thông tin được phân chia vào trong những bảng. Mỗi bảng lại được chia thành những hàng và những cột ; những cột này lưu trữ những thông tin thực tế. Bạn truy nhập một cơ sở dữ liệu sử dụng ngôn ngữ truy vấn có cấu trúc (SQL), là một ngôn ngữ tiêu chuẩn được hỗ trợ bởi đa số phần mềm cơ sở dữ liệu bao gồm SQL Server, Access, và Oracle.

Trong chương này, bạn sẽ thấy một chương trình C# kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL, truy xuất và hiển thị nội dung cất giữ trong những cột của một hàng từ một bảng, rồi ngắt kết nối với cơ sở dữ liệu. Bạn cũng sẽ thấy những chương trình kết nối tới những cơ sở dữ liệu Access và Oracle .

Bạn cũng sẽ học về công cụ phát triển nhanh ứng dụng của Microsoft (RAD) , Visual Studio .NET (VS .NET). VS .NET cho phép bạn phát triển, chạy, và gỡ lỗi những chương trình trong một môi trường phát triển tích hợp. Môi trường này sử dụng tất cả những đặc tính lớn của Windows, như con chuột và những thực đơn nhanh, và tăng năng suất của bạn như một lập trình viên.

Trong những mục cuối cùng của chương này, bạn sẽ thấy cách sử dụng tài liệu Microsoft rộng lớn đi cùng với công cụ phát triển phần mềm .NET(SDK). và VS .NET. Bạn sẽ tìm thấy tài liệu này vô giá khi bạn trở nên một chuyên gia với ADO.NET và C#. Bạn cũng sẽ học cách sử dụng tài liệu SQL Server như thế nào.

Những đặc trưng trong chương này:

- Phần mềm cần thiết
- Phát triển chương trình ADO.NET đầu tiên của bạn.
- Sự kết nối tới những cơ sở dữ liệu Access và Oracle
- Giới thiệu về Visual Studio .NET
- Sử dụng tài liệu .NET.
- Sử dụng tài liệu SQL Server

Phần mềm cần thiết

Trước khi bạn có thể phát triển những chương trình C#, bạn sẽ cần cài đặt .NET Software Development Kit (SDK) hoặc VS .NET. Bạn có thể tải .NET SDK. tại http: // msdn.microsoft.com / downloads (tìm kiếm công cụ phát triển phần mềm khung .NET của Microsoft). Bạn có thể mua một một bản thử hay bản copy đầy đủ của VS .NET từ Microsoft tại http: // msdn.microsoft.com / vstudio.

Để cài đặt .NET SDK, Chạy tập tin khả thi bạn tải xuống và theo những chỉ dẫn trên màn ảnh để cài đặt nó trên máy tính các bạn. Để cài đặt VS .NET, chạy file setup.exe trên đĩa và theo những chỉ dẫn trên màn ảnh.

Bạn cũng sẽ cần một bàn sao của phần mềm cơ sở dữ liệu SQL Server. Vào thời điểm đang ghi bạn có thể tải xuống một phiên bản thử của SQL Server từ Microsoft tại http: // www.microsoft.com / sql. Bạn có thể cũng mua một phiên bản thử hay bản sao đầy đủ của SQL Server từ trang web của Microsoft.

Sách này sử dụng phần mềm Phiên bản người phát triển của SQL Server 2000 (Developer Edition of the SQL Server 2000) và sử dụng một cơ sở dữ liệu tên Northwind. Cơ sở dữ liệu này chứa đựng thông tin cho Công ty Northwind tưởng tượng, bán những sản phẩm thức ăn tới những khách hàng. Northwind là một trong số những cơ sở dữ liệu ví dụ mà bạn có thể cài đặt với SQL Server. Thông tin khách hàng trong cơ sở dữ liệu Northwind là những khách hàng có tên được lưu trữ trong một bảng; bạn sẽ thấy cách sử dụng bảng này trong chương trình ví dụ sau trong chương này.

Nếu bạn không muốn tải hay mua một phiên bản thử của SQL Server, .NET SDK (và VS .NET) đi cùng với một bộ dịch vụ cơ sở dữ liệu máy để bàn độc lập được biết đến như Microsoft SQL Server 2000 (MSDE 2000). MSDE 2000 có một phiên bản của cơ sở dữ liệu Northwind mà bạn có thể sử dụng thay cho cơ sở dữ liệu Northwind SQL Server mặc dầu bạn sẽ không có được tất cả những công cụ quản trị đồ thị có trong SQL Server. Nếu bạn đang sử dụng .NET SDK. và muốn cài đặt MSDE 2000, chọn Start ➢ Microsoft. NET Framwork SDK ➢ Samples and QuickStart Tutorials . Nếu bạn đang sử dụng VS .NET và muốn cài đặt MSDE 2000 , chạy chương trình setup.exe mà bạn dùng để cài đặt VS .NET và chọn MSDE 2000 như một tính năng mới để cài đặt.

Ghi chú bạn có thể học nhiều hơn về MSDE 2000 tại http: // www.microsoft.com / sql/ techinfo/ development/ 2000/ msde2000.asp.

Phát triển chương trình ADO.NET đầu tiên của bạn

Trong mục này bạn sẽ bắt tay vào việc lập trình ADO.NET và sẽ thấy một chương trình C# thực hiện những tác vụ sau đây:

1. Kết nối tới cơ sở dữ liệu Northwind của máy chủ phục vụ SQL
2. Truy xuất một hàng từ bảng những khách hàng (Customers)
3. Trình bày những cột của hàng
4. Đóng kết nối cơ sở dữ liệu

Bạn sẽ được giới thiệu tới nhiều khái niệm trong mục này , và nó sẽ hoàn toàn được khám phá trong những chương sau . Đừng quá lo lắng về tất cả những chi tiết của những khái niệm trong giai đoạn này; bạn sẽ học những chi tiết đó trong những chương sau.

Danh sách 1.1 cho thấy chương trình ví dụ , nó được chứa trong file FirstExample.cs.

Danh sách 1.1: FIRSTEXAMPLE.CS

```
/*
  FirstExample.cs minh họa cách:
  1. kết nối với cơ sở dữ liệu Northwind SQL Server .
  2. truy xuất một hàng từ bảng Customers sử dụng
     một phát biểu SELECT SQL.
  3. hiển thị những cột từ một hàng.
  4. đóng kết nối dữ liệu.
*/
```

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

class FirstExample
{
 public static void Main()
 {
  try
  {
   // bước 1: tạo một đối tượng SqlConnection để kết nối tới
   // cơ sở dữ liệu Northwind SQL Server
   SqlConnection mySqlConnection =
    new SqlConnection( "server=localhost;database=Northwind;uid=sa;pwd=sa");

   // bước 2: tạo một đối tượng SqlCommand
   SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

   // bước 3: gán thuộc tính CommandText của đối tượng SqlCommand tới
   // một phát biểu SELECT SQL để truy xuất một hàng từ bảng Customers
   mySqlCommand.CommandText =
    "SELECT CustomerID, CompanyName, ContactName, Address "+
    "FROM Customers "+
    "WHERE CustomerID = 'ALFKI'";

   // bước 4: mở kết nối cơ sở dữ liệu sử dụng
   // phương thức Open() của đối tượng SqlConnection
   mySqlConnection.Open();
   // bước 5: tạo một đối tượng SqlDataReader và gọi phương thức ExecuteReader()
   // của đối tượng SqlCommand để chạy phát biểu SELECT
   SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

   // bước 6: đọc những hàng từ đối tượng SqlDataReader sử dụng
   // phương thức Read()
   mySqlDataReader.Read();

   // bước 7: hiển thị giá trị những cột
   Console.WriteLine("mySqlDataReader[\" CustomerID\"] = "+
    mySqlDataReader["CustomerID"]);
   Console.WriteLine("mySqlDataReader[\" CompanyName\"] = "+
    mySqlDataReader["CompanyName"]);
   Console.WriteLine("mySqlDataReader[\" ContactName\"] = "+
    mySqlDataReader["ContactName"]);
   Console.WriteLine("mySqlDataReader[\" Address\"] = "+
    mySqlDataReader["Address"]);

   // bước 8: đóng đối tượng SqlDataReader sử dụng phương thức Close()
   mySqlDataReader.Close();

   // bước 9: đóng đối tượng SqlConnection sử dụng phương thức Close()
   mySqlConnection.Close();
  }
  catch (SqlException e)
  {
   Console.WriteLine("A SqlException was thrown");
   Console.WriteLine("Number = "+ e.Number);
   Console.WriteLine("Message = "+ e.Message);
```

```
            Console.WriteLine("StackTrace:\n" + e.StackTrace);
        }
    }
}
```

Ghi chú

Bạn có thể tải tất cả những tập tin nguồn cho những chương trình đặc trưng trong sách này từ trang web của Sybex tại www.sybex.com. Bạn sẽ tìm thấy những chỉ dẫn về việc tải những tập tin này trong phần giới thiệu của sách này. Một khi bạn đã tải xuống những tập tin này, bạn có thể theo những ví dụ không cần phải nhập những mã chương trình.

Chúng ta hãy duyệt qua những hàng trong FirstExample.cs. Tập hợp những hàng đầu tiên là một chú thích cho biết cái mà chương trình làm

```
/*
  FirstExample.cs illustrates how to:
  1. Connect to the SQL Server Northwind database.
  2. Retrieve a row from the Customers table using
     a SQL SELECT statement.
  3. Display the columns from the row.
  4. Close the database connection.
*/
```

Hai hàng tiếp theo cho biết những ***namespaces*** được tham chiếu trong chương trình với phát biểu ***using***:

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

Namespace "System" là namespace gốc và được tham chiếu để chúng ta có thể sử dụng cách đơn giản những lệnh gọi Console .WriteLine() trong chương trình, thay vì phải ghi lệnh gọi hợp cách đầy đủ như : System.Console.WriteLine() . Không gian tên (namespace) chứa những lớp ADO.NET cho sự sử dụng với SQL Server, bao gồm các lớp SqlConnection, SqlCommand, và SqlDataReader được sử dụng sau đó trong chương trình. Bạn sẽ được giới thiệu tới những lớp này không lâu nữa, và Bạn sẽ học những chi tiết đầy đủ của những lớp ADO.NET khi bạn tiến triển thông qua sách này.

Bạn xử lý những ngoại lệ mà có lẽ đã được tung ra trong mã của bạn bởi việc đặt mã bên trong một khối try/ catch . Bạn sẽ chú ý chín bước được đặt vào một khối try/ catch trong phương thức Main() , với khối catch xử lý một đối tượng SqlException mà có lẽ đã được ném ra bởi mã trong khối try. Bạn sẽ học nhiều hơn về điều này sau trong mục " Xử lý những ngoại lệ " sau khi tôi bàn luận về chín bước trong những mục sau đây.

Bước 1: Tạo ra một đối tượng SqlConnection để kết nối tới Cơ sở dữ liệu

Bạn sử dụng một đối tượng của lớp SqlConnection để kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL.

Bước 1: trong phương thức Main() tạo ra một đối tượng SqlConnection đặt tên mySqlConnection để kết nối tới cơ sở dữ liệu Northwind máy chủ phục vụ SQL:

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

Chuỗi được chuyển đến bộ khởi dựng SqlConnection được biết như connection string và chứa những phần tử sau đây:

Server: chỉ rõ tên của máy tính trên đó SQL Server đang chạy- trong ví dụ này là localhost ; localhost là một tên chung tham chiếu tới máy tính mà trên đó chương trình của bạn chạy. Nếu cơ sở dữ liệu của bạn đang chạy lưu trú trên một máy tính khác với máy tính mà chương trình hiện thời bạn đang chạy, thì bạn sẽ cần phải thay thế localhost với tên của máy tính đó.

Database: chỉ rõ tên của cơ sở dữ liệu - trong ví dụ này là Northwind.

uid : chỉ rõ tên tài khoản của người sử dụng cơ sở dữ liệu : trong ví dụ này là sa ; sa là một tài khoản của người dùng cơ sở dữ liệu chung được quản lý bởi người quản trị cơ sở dữ liệu "database administrator" (DBA). Bạn có thể sử dụng bất kỳ tài khoản người sử dụng cơ sở dữ liệu nào miễn là nó có quền truy cập

tới cơ sở dữ liệu Northwind.

pwd: chỉ rõ mật khẩu cho người sử dụng. Mật khẩu cho người sử dụng "sa" trong cơ sở dữ liệu của tôi cũng là sa. Bạn sẽ cần thay đổi pwd thành mật khẩu cho tài khoản sa của bạn, hay bất cứ tài khoản nào bạn chỉ định ở uid.

Bạn sẽ cần thay đổi những sự thiết đặt của một số hay tất cả những phần tử trước đây trong chuỗi kết nối của bạn. Bạn có thể đã cần liên hệ với DBA (người quản trị cơ sở dữ liệu) của bạn để lấy thông tin về những phần tử tạo ra chuỗi kết nối của bạn. Một khi bạn có những giá trị đúng, bạn cần phải thực hiện những sự thay đổi tới chuỗi kết nối trong bản sao của FirstExample.cs của bạn với những giá trị đúng này.

Ghi nhớ: Một người quản trị cơ sở dữ liệu (DBA) chịu trách nhiệm thực hiện những nhiệm vụ như cài đặt phần mềm cơ sở dữ liệu, sao lưu những cơ sở dữ liệu, vân vân.

Bước 2: Tạo ra một đối tượng SqlCommand

Bước 2: tạo ra một đối tượng SqlCommand đặt tên mySqlCommand được sử dụng sau đó để gửi một phát biểu SELECT cho cơ sở dữ liệu cho sự thực thi.

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

Bước 3: Thiết đặt thuộc tính CommandText của Đối tượng SqlCommand

Bạn sử dụng SQL để làm việc với thông tin được cất giữ trong một cơ sở dữ liệu. SQL là một ngôn ngữ tiêu chuẩn công nghiệp được hỗ trợ bởi máy chủ phục vụ SQL, Access, và Oracle. Bạn sử dụng phát biểu SELECT SQL để truy xuất thông tin từ một cơ sở dữ liệu. Bạn sẽ học những cơ sở SQL trong phần giới thiệu ở Chương 3, " Giới thiệu ngôn ngữ truy vấn có cấu trúc."

Bước 3: gán thuộc tính CommandText của mySqlCommand tạo ra trong bước trước đây tới một phát biểu SELECT. Phát biểu này sẽ truy xuất những cột CustomerID, CompanyName, ContactName, và Address từ hàng trong bảng những khách hàng có CustomerID là ALFKI:

```
mySqlCommand.CommandText =
  "SELECT CustomerID, CompanyName, ContactName, Address "+
  "FROM Customers "+
  "WHERE CustomerID = 'ALFKI'";
```

Bước 4: Mở đối tượng SqlConnection

Bước 4: mở kết nối cơ sở dữ liệu sử dụng phương thức Open() của đối tượng SqlConnection tạo ra trong Bước 1:

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

Một khi kết nối tới cơ sở dữ liệu được mở, bạn có thể gửi những lệnh cho cơ sở dữ liệu cho sự thực thi.

Bước 5: Chạy phát biểu SELECT

Bạn chạy phát biểu SELECT trước đó được gán trong mySqlCommand bởi việc gọi phương thức ExecuteReader() . Phương thức này trả về một đối tượng SqlDataReader và rồi bạn sẽ dùng để đọc dữ liệu hàng được trả về bởi phát biểu SELECT.

Bước 5: tạo ra một đối tượng SqlDataReader và gọi phương thức ExecuteReader() của đối tượng mySqlCommand để chạy phát biểu SELECT.

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

Bước 6: Đọc hàng sử dụng đối tượng SqlDataReader

Bước 6: đọc hàng trong mySqlDataReader sử dụng phương thức Read():

```
mySqlDataReader.Read();
```

Bước 7: Trình bày những giá trị cột từ đối tượng SqlDataReader

Bạn có thể đọc giá trị cho một cột từ mySqlDataReader bằng cách chuyển qua tên của cột trong cặp dấu ngoặc vuông. Chẳng hạn, mySqlDataReader[" CustomerID "] trả về giá trị của cột CustomerID.

Bước 7: trình bày những giá trị cột cho CustomerID, CompanyName, ContactName, và Address:

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

Bước 8: Đóng đối tượng SqlDataReader

Khi bạn kết thúc việc đọc những hàng từ một đối tượng SqlDataReader, đóng nó sử dụng phương thức Close().
Bước 8: gọi phương thức Close() để đóng mySqlDataReader:

```
mySqlDataReader.Close();
```

Bước 9: Đóng đối tượng SqlConnection

Khi bạn kết thúc truy cập cơ sở dữ liệu, đóng đối tượng SqlConnection của bạn sử dụng phương thức Close().
Bước 9: gọi phương thức Close() để đóng mySqlConnection:

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

Xử lý những ngoại lệ

Bạn xử lý những ngoại lệ mà có lẽ đã được tung ra trong mã của bạn bên trong một khối try/catch. Bạn sẽ chú ý tới chín bước được đặt bên trong một khối try/catch, với khối catch xử lý một đối tượng SqlException mà có lẽ đã được tung ra bởi mã trong khối try. Lớp SqlException đặc biệt dành cho sự sử dụng của mã truy cập một cơ sở dữ liệu máy chủ phục vụ SQL.

Ví dụ sau đây trình bày cách để xây dựng một khối try/catch

```
try
{
  /* đoạn mã có thể gay ra một lỗi SqlException */
}
catch (SqlException e)
{
  Console.WriteLine("A SqlException was thrown");
  Console.WriteLine("Number = "+ e.Number);
```

```
    Console.WriteLine("Message = "+ e.Message);
    Console.WriteLine("StackTrace:\n" + e.StackTrace);
  }
```

Những thuộc tính được trình bày trong khối catch như sau:

Number: Số đặc trưng cho lỗi
Message: Một chuỗi chứa một mô tả về lỗi
StakTrace: Một chuỗi chứa tên của lớp và phương thức từ đối tượng mà lỗi được tung ra.

Hai ví dụ chung nhất về khi nào một SqlException được ném ra như sau:

- Đối tượng SqlConnection của bạn không thể kết nối tới cơ sở dữ liệu. Nếu điều này xảy ra, bạn cần phải kiểm tra chuỗi kết nối chỉ định sự kết nối tới cơ sở dữ liệu của bạn như thế nào.

- Phát biểu SELECT của bạn chứa một lỗi chính tả trong tên một bảng hay cột.

Đầu ra ví dụ sau đây cho thấy những gì xảy ra khi đối tượng SqlConnection trong FirstExample.cs không thể kết nối tới cơ sở dữ liệu bởi vì cơ sở dữ liệu hiện thời đang suy giảm .

```
A SqlException was thrown
Một lỗi về kết nối đã tung ra
Number = -2
Mã lỗi = -2
Message = Timeout expired. Possible reasons: the timeout period elapsed prior
Thông báo= kết thúc thời gian chờ kết nối. nguyên nhân có thể: giai đoạn thời gian chờ kết nối
 to completion of the operation, the server is not responding,
đã qua trước khi hoàn tất thao tác, máy chủ không đáp ứngkết nối
 or the maximum pool size was exceeded.
Hoặc đã tới mức cực đại vủa bể chứa
 Please see the documentation for further details.
StackTrace:
   at System.Data.SqlClient.SqlConnection.Open()
   at FirstExample.Main()
```

Bạn có thể sử dụng đầu ra từ khối catch để xác định vấn đề. Nếu cơ sở dữ liệu đang suy giảm, liên hệ với DBA (người quản lý dữ liệu) của bạn.

Ghi nhớ: Để cho ngắn gọn, chương trình chỉ sử dụng một khối try/catch trong sách này là FirstExample.cs. Bạn cần phải sử dụng những khối try/catch trong những chương trình của mình để bắt những ngoại lệ . Để biết thêm nhiều chi tiết về xử lý ngoại lệ hơn, Tôi xin giới thiệu bạn sách "Mastering Visual C# .NET from Sybex (2002).".

Trong mục kế tiếp bạn sẽ thấy cách biên dịch FirstExample.cs và chạy nó.

Biên dịch và chạy file FirstExample.cs

Bạn có thể biên dịch chương trình FirstExample.cs sử dụng công cụ command-line đi cùng với .NET SDK. hay VS .NET. Trong mục này, bạn sẽ thấy cách sử dụng phiên bản dòng lệnh (command-line) của trình biên dịch cho chương trình FirstExample.cs như thế nào. Sau đó trong chương này, trong mục " Giới thiệu về Visual Studio .NET", Bạn sẽ thấy cách sử dụng VS .NET để biên dịch và chạy một chương trình như thế nào.

Bạn chạy phiên bản dòng lệnh (command-line) của trình biên dịch bởi việc nhập csc vào trong công cụ dấu nhắc lệnh (Command Prompt), theo sau là tên của tập tin nguồn chương trình của bạn. Chẳng hạn, để biên dịch FirstExample.cs, Bạn nhập lệnh sau đây vào trong công cụ dấu nhắc lệnh:

```
csc FirstExample.cs
```

Nếu bạn muốn theo cùng với những ví dụ, khởi động công cụ dấu nhắc lệnh bằng cách chọn Start > Programs > Command Prompt.

> **Ghi nhớ**: nếu bạn đang sử dụng Windows XP thay vì Windows 2000, khởi động công cụ dấu nhắc lệnh bằng cách chọn Start > All Programs > Accessories > Command Prompt.

Tiếp theo, bạn cần thay đổi những thư mục tới nơi mà bạn đã sao chép file FirstExample.cs . Để làm điều này, đầu tiên bạn nhập vào phân vùng (partition) trên đĩa cứng của các bạn nơi bạn đã lưu tập tin. Chẳng hạn, chúng ta hãy cho là bạn đã lưu tập tin trong thư mục ADO.NET\book\ch01\programs trong đĩa C của bạn. Để truy cập đĩa C , bạn nhập hàng sau đây vào trong dấu nhắc lệnh và sau đó bạn nhấn phím Enter

```
C:
```

Tiếp theo, chuyển tới thư mục ADO.NET\book\ch01\programs , bạn nhập cd theo sau là ADO.NET\book\ch01\programs:

```
cd ADO.NET\book\ch01\programs
```

Để biên tập FirstExample.cs sử dụng csc, Bạn nhập lệnh sau đây:

```
csc FirstExample.cs
```

> **Chú ý:** tên của tập tin nguồn chương trình theo sau csc; trong trường hợp này, nó là FirstExample.cs.

Nếu bạn nhận một lỗi khi chạy csc, Bạn sẽ cần thêm thư mục nơi bạn cài đặt SDK vào biến môi trường đường dẫn của bạn. Biến môi trường đường dẫn chỉ rõ một danh sách của những thư mục chứa những chương trình có thể thực thi. Bất cứ khi nào mà bạn chạy một chương trình từ dấu nhắc lệnh, những thư mục trong biến đường dẫn được tìm kiếm cho chương trình bạn muốn chạy. Thư mục hiện hành của bạn cũng được tìm kiếm. Để gán biến môi trường đường dẫn ,bạn thực hiện như sau:

1. Chọn Start > Settings > Control Panel. Rồi nhấn đúp System và chọn thẻ Advanced .
2. Kích nút Environment Variables và nhấn đúp Path từ vùng system variables ở đáy.
3. Thêm thư mục nơi bạn cài đặt SDK vào biến môi trường Path của bạn.
4. Kích OK để lưu sự thay đổi của bạn, và kích OK lần nữa trên hộp thoại kế tiếp.
5. Khởi động lại Command Prompt, như thế sự thay đổi của bạn được chọn. và Bạn có thể chạy csc một cách thành công.

Trình biên dịch đọc file FirstExample.cs và biên dịch thành một tập tin khả thi có tên FirstExample.exe. Tập tin loại EXE. chứa những chỉ lệnh mà một máy tính có thể chạy được, và phần mở rộng tập tin .exe cho biết tập tin này là một tập tin khả thi.

Bạn chạy một tập tin khả thi sử dụng công cụ Command Prompt (dấu nhắc lệnh) bằng cách nhập tên của tập tin khả thi này. Chẳng hạn, để chạy file FirstExample.exe , bạn nhập hàng sau đây trong dấu nhắc lệnh và sau đó bạn nhấn phím Enter

```
FirstExample
```

Khi bạn chạy chương trình, nếu thành công ,bạn phải thấy văn bản sau đây được trình bày trong cửa sổ dấu nhắc lệnh của bạn:

```
mySqlDataReader["CustomerID"] = ALFKI
mySqlDataReader["CompanyName"] = Alfreds Futterkiste
mySqlDataReader["ContactName"] = Maria Anders
mySqlDataReader["Address"] = Obere Str. 57
```

Nếu bạn gặp một ngoại lệ- như chương trình của bạn không thể kết nối tới cơ sở dữ liệu- bạn cần phải kiểm tra chuỗi kết nối thiết lập trong bước 1 của FirstExample.cs, và liên hệ với DBA (người quan trị dữ liệu) của bạn nếu cần thiết .

Kết nối tới những cơ sở dữ liệu Access và Oracle

Trong mục này bạn sẽ thấy những ví dụ của việc kết nối tới một cơ sở dữ liệu Access lẫn Oracle. Để tương tác với những cơ sở dữ liệu này trong chương trình của bạn, bạn sử dụng những lớp từ không gian tên System.Data.OleDb. Không gian tên này chứa những lớp cho sự sử dụng với những cơ sở dữ liệu hỗ trợ sự liên kết và sự nhúng đối tượng cho những cơ sở dữ liệu (OLE DB) như Access và Oracle. Bạn sẽ học nhiều hơn về không gian tên System.Data.OleDb trong Chương 5, "Tổng quan về những lớp ADO.NET ."

Kết nối tới một cơ sở dữ liệu Access

Bạn kết nối tới một cơ sở dữ liệu Access sử dụng một đối tượng OleDbConnection thay vì một đối tượng SqlConnection - với một chuỗi kết nối của định dạng sau đây:

```
provider=Microsoft.Jet.OLEDB.4.0;data source=databaseFile
```

Với databaseFile là đường dẫn và tên file của cơ sở dữ liệu Access của bạn. Chú ý bạn chỉ rõ provider (nhà cung cấp) trong chuỗi kết nối, nó được gán là Microsoft.Jet.OLEDB.4.0.

Ví dụ sau đây tạo một chuỗi có tên connectionString với định dạng thích hợp để kết nối tới cơ sở dữ liệu Northwind Access được lưu trữ trong file Northwind.mdb :

```
string connectionString =
  "provider=Microsoft.Jet.OLEDB.4.0;" +
  "data source=F:\\Program Files\\Microsoft
  Office\\Office\\Samples\\Northwind.mdb";
```

Ghi chú:

Chú ý sử dụng của hai ký tự dấu sổ ngược (\\) trong phần data source của chuỗi kết nối. Dấu sổ ngược đầu tiên được dùng để chỉ định dấu sổ ngược thứ hai sẽ được đối sử như ký tự; bởi vậy \\ được xem xét như \ trong chuỗi kết nối. Bạn sẽ cần định vị file Northwind.mdb trên đĩa cứng của bạn và thiết lập chuỗi kết nối của bạn cho phù hợp.

Giả thiết namespace *System.Data.OleDb* đã được tham chiếu (imported), ví dụ sau đây tạo một đối tượng *OleDbConnection*, thông qua *connectionString* (thiết đặt trong dòng mã trước đây) tới bộ khởi dựng:

```
OleDbConnection myOleDbConnection =
  new 01eDbConnection(connectionString);
```

Danh sách 1.2 minh họa cách kết nối tới cơ sở dữ liệu Access Northwind sử dụng một đối tượng OleDbConnection và truy xuất một hàng từ bảng những khách hàng như thế nào. Chú ý bạn sử dụng một đối tượng OleDbCommand và OleDbDataReader để chạy một câu lệnh SQL và đọc những kết quả được trả về từ một cơ sở dữ liệu Access.

Danh sách 1.2: OLEDBCONNECTIONACCESS.CS

```
/*
  OleDbConnectionAccess.cs illustrates how to use an
  OleDbConnection object to connect to an Access database
*/

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

class OleDbConnectionAccess
{
  public static void Main()
  {
```

```
            // formulate a string containing the details of the
            // database connection
            string connectionString =
              "provider=Microsoft.Jet.OLEDB.4.0;" +
              "data source=F:\\Program Files\\Microsoft
        Office\\Office\\Samples\\Northwind.mdb";

            // create an OleDbConnection object to connect to the
            // database, passing the connection string to the constructor
            OleDbConnection myOleDbConnection =
              new OleDbConnection(connectionString);

            // create an OleDbCommand object
            OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();

            // set the CommandText property of the OleDbCommand object to
            // a SQL SELECT statement that retrieves a row from the Customers table
            myOleDbCommand.CommandText =
              "SELECT CustomerID, CompanyName, ContactName, Address "+
              "FROM Customers "+
              "WHERE CustomerID = 'ALFKI'";
            // open the database connection using the
            // Open() method of the OleDbConnection object
            myOleDbConnection.Open();

            // create an OleDbDataReader object and call the ExecuteReader()
            // method of the OleDbCommand object to run the SELECT statement
            OleDbDataReader myOleDbDataReader = myOleDbCommand.ExecuteReader();

            // read the row from the OleDbDataReader object using
            // the Read() method
            myOleDbDataReader.Read();

            // display the column values
            Console.WriteLine("myOleDbDataReader[\" CustomerID\"] = "+
              myOleDbDataReader["CustomerID"]);
            Console.WriteLine("myOleDbDataReader[\" CompanyName\"] = "+
              myOleDbDataReader["CompanyName"]);
            Console.WriteLine("myOleDbDataReader[\" ContactName\"] = "+
              myOleDbDataReader["ContactName"]);
            Console.WriteLine("myOleDbDataReader[\" Address\"] = "+
              myOleDbDataReader["Address"]);

            // close the OleDbDataReader object using the Close() method
            myOleDbDataReader.Close();

            // close the OleDbConnection object using the Close() method
            myOleDbConnection.Close();
          }
        }
```

Đầu ra từ chương trình này như sau:

```
myOleDbDataReader["CustomerID"] = ALFKI
myOleDbDataReader["CompanyName"] = Alfreds Futterkiste
myOleDbDataReader["ContactName"] = Maria Anders
myOleDbDataReader["Address"] = Obere Str. 57
```

Kết nối tới một cơ sở dữ liệu Oracle

Bạn kết nối tới một cơ sở dữ liệu Oracle sử dụng một đối tượng OleDbConnection với một chuỗi kết nối với khuôn dạng sau đây:

provider=MSDAORA;data source=*OracleNetServiceName*;user id=*username*;password=*password*

với:

OracleNetServiceName: chỉ rõ tên dịch vụ mạng cơ sở dữ liệu Oracle . Oracle Net là một thành phần phần mềm cho phép bạn kết nối tới một cơ sở dữ liệu qua một mạng. Bạn sẽ cần liên hệ với DBA (người quản trị cơ sở dữ liệu) của bạn để lấy tên dịch vụ Mạng Oracle.

username: chỉ rõ tên của người sử dụng cơ sở dữ liệu .

Password: chỉ rõ mật khẩu cho người sử dụng cơ sở dữ liệu.

Ví dụ sau đây tạo ra một chuỗi kết nối có tên connectionString với định dạng chính sác để kết nối tới một cơ sở dữ liệu Oracle:

```
string connectionString =
  "provider=MSDAORA;data source=ORCL;user id=SCOTT;password=TIGER";
```

Ghi chú:

ID người sử dụng là SCOTT với một mật khẩu là TIGER là mặc định cho sự truy cập một trong số những cơ sở dữ liệu ví dụ đi cùng Oracle. Cơ sở dữ liệu này chứa một bảng gọi là emp chứa dữ liệu người làm thuê mẫu.

Giả thiết namespace System.Data.OleDb đã được tham chiếu, ví dụ sau đây tạo ra một đối tượng OleDbConnection, thông qua connectionString tới bộ khởi dựng:

```
OleDbConnection myOleDbConnection =
  new OleDbConnection(connectionString);
```

Danh sách 1.3 minh họa cách kết nối tới một cơ sở dữ liệu Oracle sử dụng một đối tượng OleDbConnection và truy xuất một hàng từ bảng emp như thế nào. Chú ý bạn sử dụng một đối tượng OleDbCommand và OleDbDataReader để chạy một câu lệnh SQL và đọc những kết quả được trả về từ một cơ sở dữ liệu Oracle.

Danh sách 1.3: OLEDBCONNECTIONORACLE.CS

```
/*
  OleDbConnectionOracle.cs illustrates how to use an
  OleDbConnection object to connect to an Oracle database
*/

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

class OleDbConnectionOracle
{
  public static void Main()
  {
    // formulate a string containing the details of the
    // database connection
    string connectionString =
      "provider=MSDAORA;data source=ORCL;user id=SCOTT;password=TIGER";
    // create an OleDbConnection object to connect to the
    // database, passing the connection string to the constructor
```

```
        OleDbConnection myOleDbConnection =
          new OleDbConnection(connectionString);

        // create an OleDbCommand object
        OleDbCommand myOleDbCommand = myOleDbConnection.CreateCommand();

        // set the CommandText property of the OleDbCommand object to
        // a SQL SELECT statement that retrieves a row from the emp table
        myOleDbCommand.CommandText =
          "SELECT empno, ename, sal "+
          "FROM emp "+
          "WHERE empno = 7369";

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

        // create an OleDbDataReader object and call the ExecuteReader()
        // method of the OleDbCommand object to run the SELECT statement
        OleDbDataReader myOleDbDataReader = myOleDbCommand.ExecuteReader();

        // read the row from the OleDbDataReader object using
        // the Read() method
        myOleDbDataReader.Read();

        // display the column values
        Console.WriteLine("myOleDbDataReader[\" empno\"] = "+
          myOleDbDataReader["empno"]);
        Console.WriteLine("myOleDbDataReader[\" ename\"] = "+
          myOleDbDataReader["ename"]);
        Console.WriteLine("myOleDbDataReader[\" sal\"] = "+
          myOleDbDataReader["sal"]);

        // close the OleDbDataReader object using the Close() method
        myOleDbDataReader.Close();

        // close the OleDbConnection object using the Close() method
        myOleDbConnection.Close();
      }
    }
```

Đầu ra từ chương trình này như sau:

```
myOleDbDataReader[ " empno "]= 7369
myOleDbDataReader[ " ename "]= Smith
myOleDbDataReader[ " muối "]= 800
```

Giới thiệu Visual Studio .NET

Trong những mục trước đây, bạn đã thấy chương trình mà kết nối tới nhiều cơ sở dữ liệu, truy xuất một hàng từ một bảng, và hiển thị những giá trị cột cho hàng này trên màn hình máy tính của bạn. Kiểu chương trình này được biết như một ứng dụng console vì nó trình bày đầu ra trực tiếp trên màn ảnh trên chương trình nào đang chạy.

Bạn có thể sử dụng Visual Studio .NET (VS .NET) để tạo ra những ứng dụng console, cũng như những kiểu ứng dụng sau đây:

Windows Applications : là những ứng dụng nắm lợi thế của những điều khiển trực quan cung cấp bởi hệ điều hành windows, như những thực đơn, những nút, và những hộp văn bản soạn thảo. Windows Explorer, mà bạn thường dùng để dẫn hướng tập tin của máy tính của bạn, là một ví dụ. Bạn sẽ học về lập trình Windows trong Chương 6, "Giới thiệu những ứng dụng Windows và ADO.NET."

ASP.NET Applications : Những ứng dụng này chạy qua Internet. Bạn truy cập một ứng dụng ASP.NET sử dụng một bộ duyệt web, như Internet Explorer. Những ví dụ của những ứng dụng ASP.NET là công việc ngân hàng trực tuyến, thương mại cổ phần hay những hệ thống bán đấu giá. Bạn sẽ học về lập trình ASP.NET trong Chương 15, "Giới thiệu về những ứng dụng: ASP.NET."

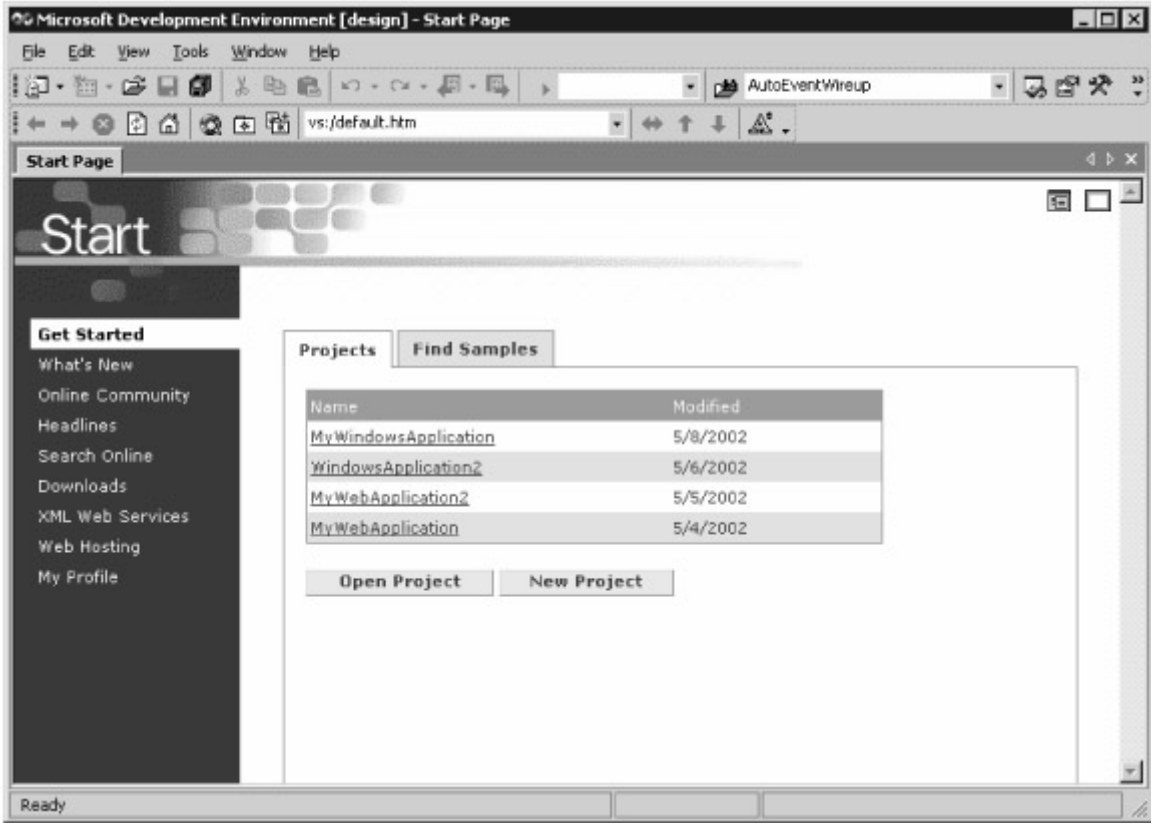
ASP.NET Web Services: Những dịch này cũng chạy qua Internet. Được biết như những dịch vụ mạng XML, sự khác nhau là bạn có thể sử dụng chúng để cung cấp một dịch vụ có thể được dùng trong một hệ thống phân tán của những dịch vụ liên kết với nhau. Chẳng hạn, dịch vụ mạng hộ chiếu của Microsoft đưa ra sự nhận dạng và sự thẩm định quền hạn của những người sử dụng Web và rồi bạn có thể rồi sử dụng trong ứng dụng Mạng của mình. Bạn sẽ học về những dịch vụ mạng trong Chương 17, "Những dịch vụ Mạng."

Đây không phải là một danh sách toàn diện của những kiểu ứng dụng bạn có thể phát triển với VS .NET, nhưng nó đưa cho bạn hương vị cho phạm vi rộng về những khả năng của VS.NET .

Trong phần còn lại của mục này bạn sẽ thấy cách phát triển và chạy một ứng dụng console sử dụng VS .NET. Nếu bạn có cài đặt VS .NET trên máy tính của bạn, bạn sẽ có khả năng thực hiện những ví dụ. Nếu bạn không có VS .NET, đừng lo lắng; bạn sẽ vẫn còn có khả năng để nhìn thấy những gì đang diễn tiến từ những hình ảnh được cung cấp sau đây.

Khởi động Visual Studio .NET và tạo một Dự án

Tất cả công việc của bạn trong VS .NET được tổ chức vào trong những dự án. Những dự án chứa nguồn và những tập tin khả thi cho chương trình của bạn, trong số những tiết mục khác. Nếu bạn có cài đặt VS .NET, khởi động nó bằng cách chọn Start > Programes > Microsoft Visual Studio .NET . Một khi VS .NET bắt đầu, bạn sẽ nhìn thấy trang Bắt đầu "Start page" (xem Hình 1.1).



Hình 1.1: trang Bắt đầu

Từ trang Start page, bạn có thể nhìn thấy bất kỳ dự án hiện hữu nào mà bạn đã tạo ra. Bạn có thể mở và tạo những dự án sử dụng những nút Open Project và New Project tương ứng. Bạn sẽ tạo ra một dự án mới không lâu sau đây.

Sử dụng những liên kết VS .NET

Như bạn có thể thấy từ Hình 1.1, VS .NET chứa một số mối liên kết ở bên trái trên trang Start page. Một số trong những mối liên kết này cung cấp sự truy nhập tới thông tin hữu ích trên Internet về .NET; những mối liên kết này như sau:

Get started: mở trang Start page.

What's New : xem bất kỳ cập nhật nào về VS .NET hay Windows. Bạn cũng có thể xem những sự kiện huấn luyện gần đây và những hội nghị.

Online Community (Cộng đồng trực tuyến) giao tiếp với những thành viên khác của Cộng đồng Mạng... Bao gồm những mối liên kết tới những trang web và những nhóm tin tức.

Headlines : xem tin tức mới nhất trên .NET.

Search Online : Tìm kiếm thư viện trực tuyến MSDN cho vật chất kỹ thuật như những bài báo được xuất bản trên Mạng.

Downloads : Tải xuống những ứng dụng thử và những chương trình ví dụ từ những trang web đặc trưng ở đây.

XML Web Services : Tìm những dịch vụ mạng XML đã đăng ký mà bạn có thể sử dụng trong những chương trình của mình. Những dịch vụ mạng XML cũng được biết như những dịch vụ mạng ASP.NET. Bạn sẽ học nhiều hơn về những dịch vụ mạng trong Chương 17.

Web Hosting: Một công ty chủ một web site có thể nắm giữ chương trình của bạn và chạy nó cho bạn. Nó chú ý đến những máy tính mà trên đó chương trình của bạn chạy. Sử dụng mối liên kết Web Hosting để xem những công ty cung cấp những dịch vụ này.

My Profile : thiết đặt những tiết mục như sơ đồ bàn phím yêu cầu và cách trình bày cửa sổ của bạn .

Kích những mối liên kết này và khám phá thông tin được cung cấp. Như bạn sẽ nhìn thấy, ở đó là nhiều thông tin quanh. Mạng(lưới) trên Internet.

Tạo ra một dự án mới

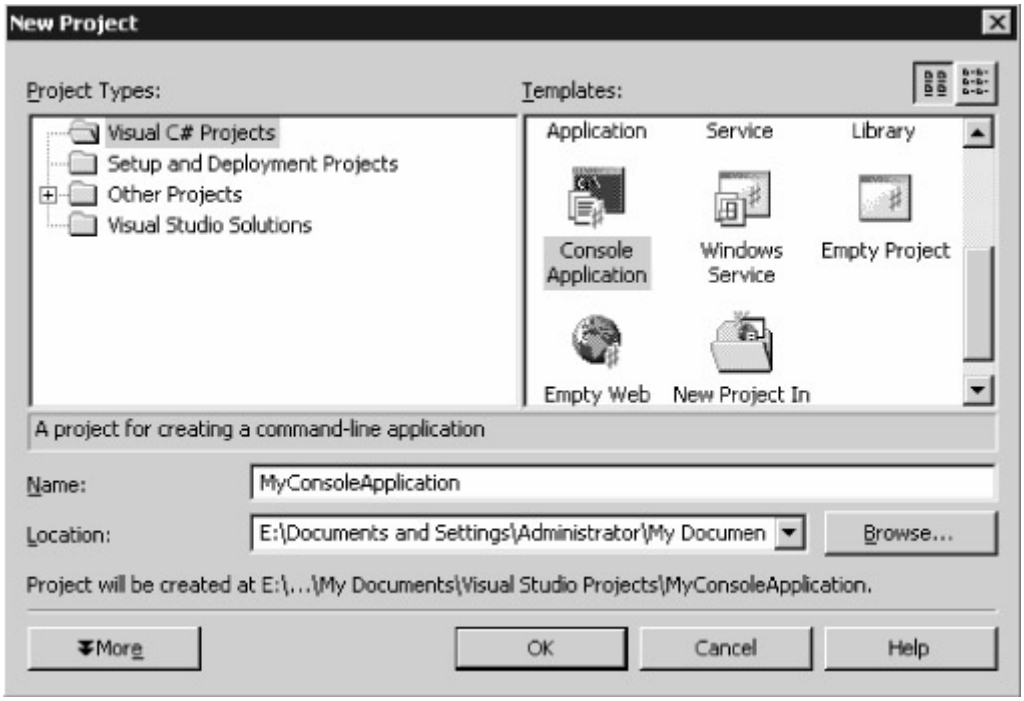
Khi bạn kết thúc khảo sát thông tin trong những mối liên kết trước đây, tạo một dự án mới bởi kích nút New Project trên trang Get Started .

Ghi chú:

Bạn có thể cũng tạo ra một dự án mới bởi việc chọn File > New > Project , hay bởi việc nhấn Ctrl+ Shift+ N trên bàn phím của bạn.

Khi bạn tạo ra một dự án mới, VS .NET trình bày hộp thoại New Project, mà bạn sử dụng để chọn kiểu dự án bạn muốn tạo ra. Bạn cũng nhập tên và vị trí của dự án mới của bạn; Vị trí là thư mục nơi bạn muốn lưu trữ những tập tin cho dự án của bạn.

Vì bạn đang chuẩn bị tạo ra một ứng dụng console C#, chọn Visual C# Projects từ khu vực Project Types về bên trái hộp thoại New Project , và chọn Console Application từ khu vực Templates (khuôn mẫu) bên trái. Nhập MyConsoleApplication trong trường Name, và giữ thư mục mặc định trong trường Location. Hình 1.2 cho thấy hộp thoại Dự án Mới đầy đủ với những sự thiết đặt này.

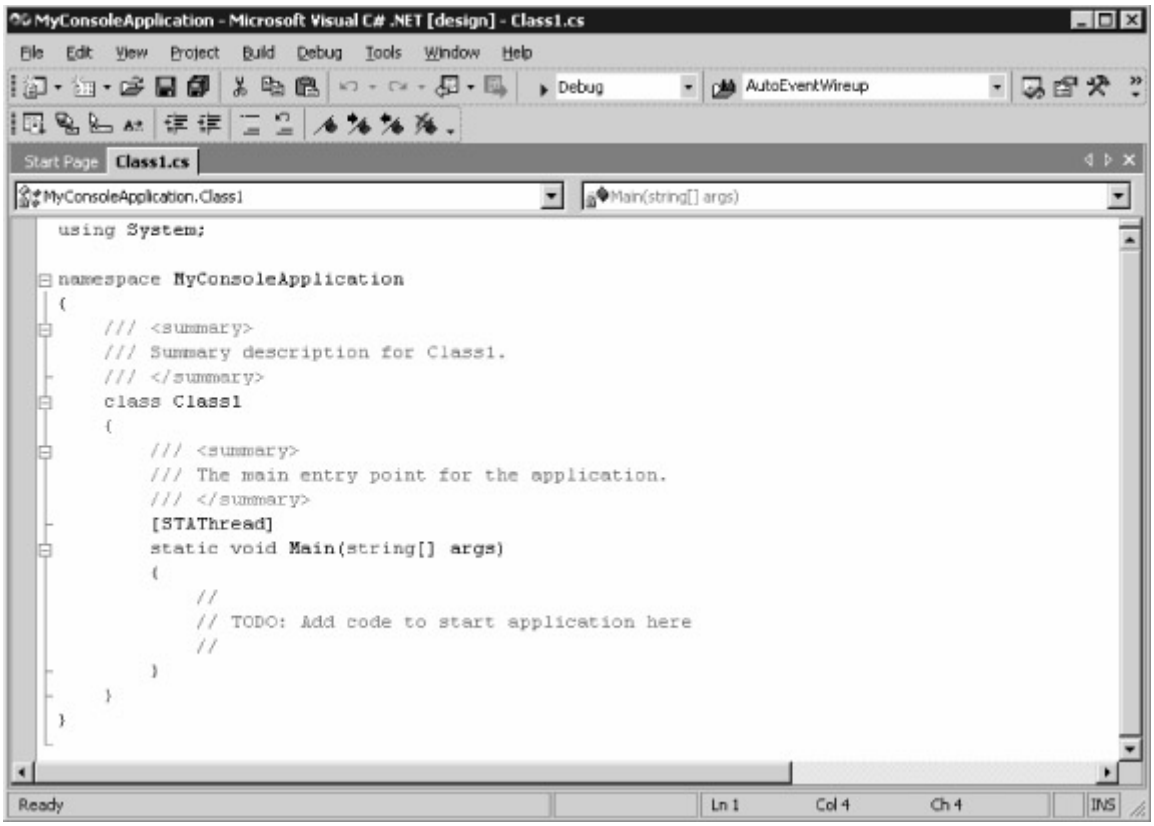


Hình 1.2: hộp thoại New Project với những sự thiết đặt thích hợp cho một ứng dụng console C#

Kích nút Ok để tạo ra dự án mới.

Làm việc trong môi trường VS .NET

Một khi bạn tạo ra một dự án mới, màn hình phát triển chính được trình bày (xem Hình 1.3). Màn hình này là môi trường trong đó bạn sẽ phát triển dự án của bạn. Như bạn có thể thấy, VS .NET đã tạo đoạn mã bắt đầu cho bạn. Mã này là một khung sườn cho chương trình của bạn; bạn sẽ thấy cách sửa đổi nó như thế nào không lâu nữa. Trong mục này, Tôi sẽ cho bạn một sự mô tả ngắn gọn của những phần khác nhau của môi trường VS .NET.



Hình 1.3: Môi trường VS .NET

Ghi chú:

Phụ thuộc vào những sự thiết đặt của bạn cho VS .NET, màn hình của bạn có thể khác với trình bày trong Hình 1.3.

Thực đơn (menu) VS .NET chứa những mục sau đây:

- **File** Open, close, và save project files.
- **Edit** Cut, copy, và paste text from the Clipboard. Clipboard là một vùng nhớ tạm thời trong ram.
- **View** Hide và show different windows như Solution Explorer (nó cho phép bạn xem những file tao thành dự án của bạn), Class View (nó cho phép bạn xem những lớp và những đối tượng trong dự án của bạn), Server Explorer (cho phép bạn duyệt qua những mục như Databases), và Properties window (cho phép bạn gán thuộc tính cho đối tượng, như kích cỡ của nút). Bạn cũng có thể sử dụng View menu để chọn những thanh công cụ bạn muốn hiển thị.
- **Project** thêm những tập tin lớp vào dự án của bạn và thêm những form cửa sổ và những điều khiển .
- **Build** Biên dịch những file nguồn trong dự án của bạn
- **Debug:** chạy chương trình của bạn với hoặc không có sự chỉnh lý. Sự chỉnh lý cho bạn bước qua từng dòng chương trình của bạn, để tìm kiếm những lỗi.
- **Tools :** kết nối tới một cơ sở dữ liệu và tùy biến những sự thiết đặt cho VS .NET . Chẳng hạn, gán những màu được dùng cho những bộ phận khác nhau của những dòng chương trình của bạn hay thiết đặt cho trang ban đầu trình bày bởi VS .NET khi bạn chạy nó.
- **Window :** Chuyển đổi giữa những tập tin bạn đã mở và ẩn những cửa sổ.
- **Help :** Mở tài liệu trên. Mạng. Bạn sẽ học sử dụng tài liệu này sau trong chương này trong mục "Using the .NET Documentation <LiB0011.html>."

Thanh công cụ VS .NET chứa một loạt những nút đóng vai trò như những phím tắt tới một số những tùy chọn thực đơn. Chẳng hạn, bạn có thể lưu một file hay tất cả các file, cắt và dán văn bản từ Clipboard, và khởi động một chương trình sử dụng trình gỡ rối. Bạn sẽ học sử dụng một số những đặc tính này sau đó trong chương này .

Mã trình bày trong cửa sổ (ở dưới thanh công cụ) với tiêu đề Class1.cs là mã được tự động phát sinh bởi VS .NET, Và trong mục kế tiếp bạn sẽ cần sửa đổi mã này.

Điều chỉnh mã được phát sinh bởi VS .NET

Một khi VS .NET đã tạo ra dự án của bạn, nó sẽ trình bày đoạn mã khởi đầu nào đó ứng dụng console với một tên lớp là Class1.cs. Bạn có thể sử dụng mã này như sự khởi đầu cho chương trình của mình. Hình 1.3, trước, trình bày đoạn mã khởi đầu tạo bởi VS .NET.

Phương thức Main() tạo bởi VS .NET như sau:

```
static void Main(string[] args)
{
  //
  // TODO: Add code to start application here
  //
}
```

Như bạn có thể thấy, mã này chứa những chú thích cho biết nơi để bạn thêm mã của mình. Thay thế phương thức Main() với mã sau đây được lấy từ phương thức Main() trong FirstExample.cs, Được chỉ ra trước đó trong Danh sách 1.1:

```
public static void Main()
{
  try
  {
    // step 1: create a SqlConnection object to connect to the
    // SQL Server Northwind database
    SqlConnection mySqlConnection =
      new SqlConnection(
        "server=localhost;database=Northwind;uid=sa;pwd=sa"
```

```
      );

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

    // step 3: set the CommandText property of the SqlCommand object to
    // a SQL SELECT statement that retrieves a row from the Customers table
    mySqlCommand.CommandText =
      "SELECT CustomerID, CompanyName, ContactName, Address "+
      "FROM Customers "+
      "WHERE CustomerID = 'ALFKI'";

    // step 4: open the database connection using the
    // Open() method of the SqlConnection object
    mySqlConnection.Open();

    // step 5: create a SqlDataReader object and call the ExecuteReader()
    // method of the SqlCommand object to run the SELECT statement
    SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

    // step 6: read the row from the SqlDataReader object using
    // the Read() method
    mySqlDataReader.Read();

    // step 7: display the column values
    Console.WriteLine("mySqlDataReader[\" CustomerID\"] = "+
      mySqlDataReader["CustomerID"]);
    Console.WriteLine("mySqlDataReader[\" CompanyName\"] = "+
      mySqlDataReader["CompanyName"]);
    Console.WriteLine("mySqlDataReader[\" ContactName\"] = "+
      mySqlDataReader["ContactName"]);
    Console.WriteLine("mySqlDataReader[\" Address\"] = "+
      mySqlDataReader["Address"]);

    // step 8: close the SqlDataReader object using the Close() method
    mySqlDataReader.Close();

    // step 9: close the SqlConnection object using the Close() method
    mySqlConnection.Close();
  }
  catch (SqlException e)
  {
    Console.WriteLine("A SqlException was thrown");
    Console.WriteLine("Number = "+ e.Number);
    Console.WriteLine("Message = "+ e.Message);
    Console.WriteLine("StackTrace:\n" + e.StackTrace);
  }
}
```

Ghi chú:

Bạn cũng sẽ cần thêm dòng sau đây cạnh đọan khởi đầu của lớp của bạn:

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

Một khi bạn đã thêm mã vào, những bước tiếp theo của bạn là biên dịch và chạy chương trình .

Biên dịch và chạy Chương trình sử dụng VS .NET

Như thông thường, đầu tiên bạn phải biên dịch chương trình trước khi có thể chạy nó. Vì những chương trình trong VS .NET được tổ chức trong những dự án, bạn phải biên dịch dự án; việc này cũng được hiểu như xây dựng dự án . Để xây dựng dự án của bạn, chọn Build > Build Solution. nó biên dịch file nguồn Class1.cs thành một tập tin khả thi .exe.

Mẹo nhỏ:

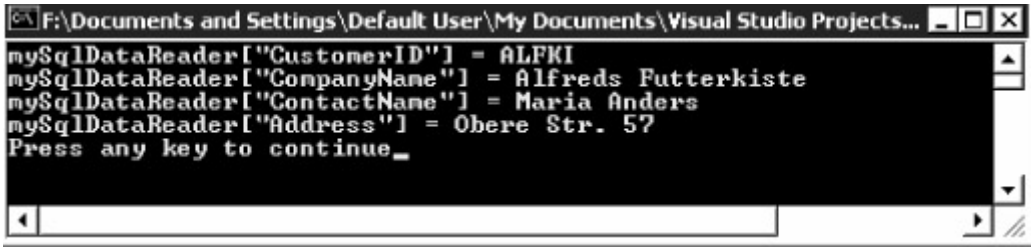
Bạn cũng có thể nhấn Ctrl+ Shift+ B trên bàn phím để xây dựng dự án của bạn.

Cuối cùng, bây giờ bạn có thể chạy chương trình của bạn. Chọn Debug □ Start Without Debugging . Khi bạn chọn Start Without Debugging, chương trình sẽ tạm ngừng ở chỗ cuối, cho phép bạn xem đầu ra.

Mẹo nhỏ:

Bạn có thể cũng nhấn Ctrl+ F5 trên bàn phím để chạy chương trình .

Khi bạn chạy chương trình , VS .NET sẽ chạy chương trình trong một cửa sổ dấu nhắc lệnh mới, như trong Hình 1.4. Chương trình của bạn được chạy trong cửa sổ này bởi vì đó là một ứng dụng console.



Hình 1.4: chương trình đang chạy

Để kết thúc chương trình, nhấn bất kỳ khóa nào. Điều này cũng sẽ đóng cửa sổ dấu nhắc lệnh.

Bạn chỉ vừa được khái quát bề mặt của VS .NET trong mục này. Bạn sẽ khám phá một số những đặc tính khác của VS .NET sau trong sách này. Trong mục kế tiếp, bạn sẽ học cách sử dụng tài liệu rộng lớn đi cùng với .NET.

Sử dụng Tài liệu Mạng.

Cả hai NET SDK. và VS .NET đến cùng với tài liệu rộng lớn, bao gồm tham khảo đầy đủ tới tất cả những lớp trong NET. Như khi bạn trở thành chuyên gia với C#, Bạn sẽ tìm thấy tài liệu tham khảo này là vô giá.

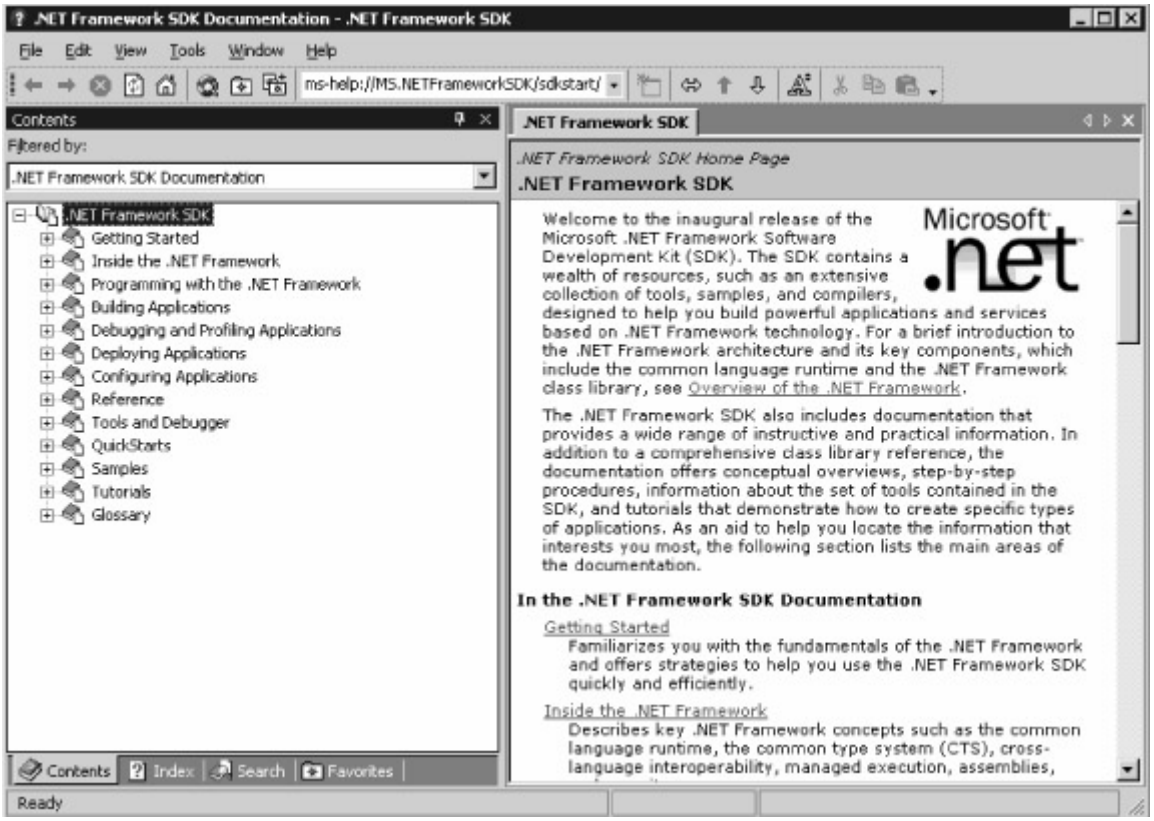
Trong những mục sau đây, bạn sẽ thấy cách truy cập và tìm kiếm tài liệu Mạng, và xem một số nội dung của tài liệu như thế nào. Phụ thuộc vào bạn đang sử dụng NET SDK. hay VS .NET, bạn truy cập tài liệu với một cách hơi khác nhau. Bạn sẽ hiểu cách sử dụng cả hai cách để truy cập tài liệu trong mục này như thế nào.

Ghi chú:

Tài liệu đi cùng với NET SDK. là một tập con của tài liệu đi cùng VS .NET.

Truy cập Tài liệu sử dụng Net SDK.

Nếu bạn đang sử dụng .NET SDK, Bạn truy cập tài liệu bằng cách chọn Start □ Programs □ Microsoft .NET Framework SDK □ Documentation . Hình 1.5 trình bày NET Framework SDK document home page; đây là trang bắt đầu cho tài liệu.



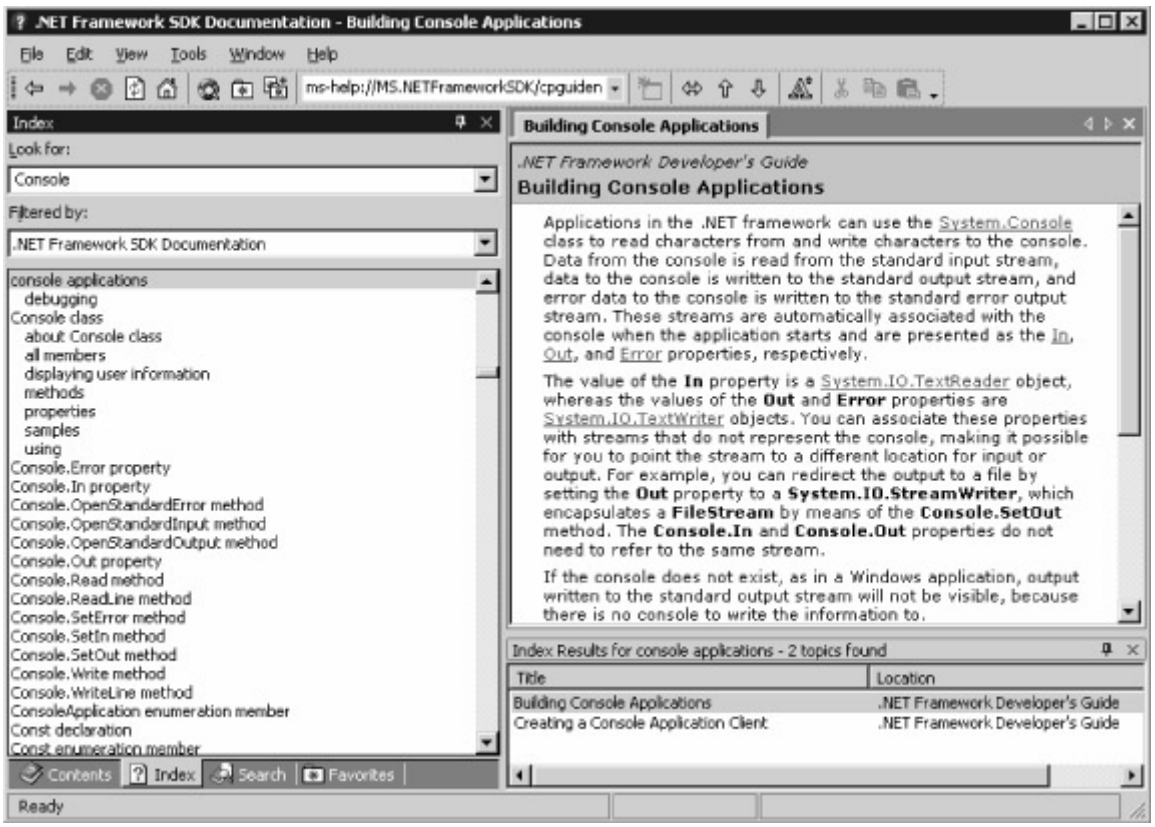
Hình 1.5: trang chủ tài liệu

Ở bên trái của trang, bạn có thể nhìn thấy nhiều mục nó tạo ra nội dung của tài liệu. Bạn có thể xem chỉ số của tài liệu bởi việc chọn thẻ Index (chỉ số) ở đáy trang.

Mẹo nhỏ:

Bạn cũng có thể xem cửa sổ Index bởi việc chọn Help □ Index, hay bởi việc nhấn Ctrl+ Alt+ F2 trên bàn phím .

Bạn có thể tìm kiếm chỉ số bởi việc nhập một từ trong trường **Look For** của thẻ Index. Hình 1.6 cho thấy những kết quả của việc tìm kiếm Console. Hình 1.6 cũng trình bày văn bản cho những chi tiết về việc xây dựng ứng dụng Console ở bên phải màn hình. Tôi đã mở tổng quan này bởi nhấn đúp liên kết Building Console Applications trong những kết quả chỉ số ở đáy phải của màn hình.



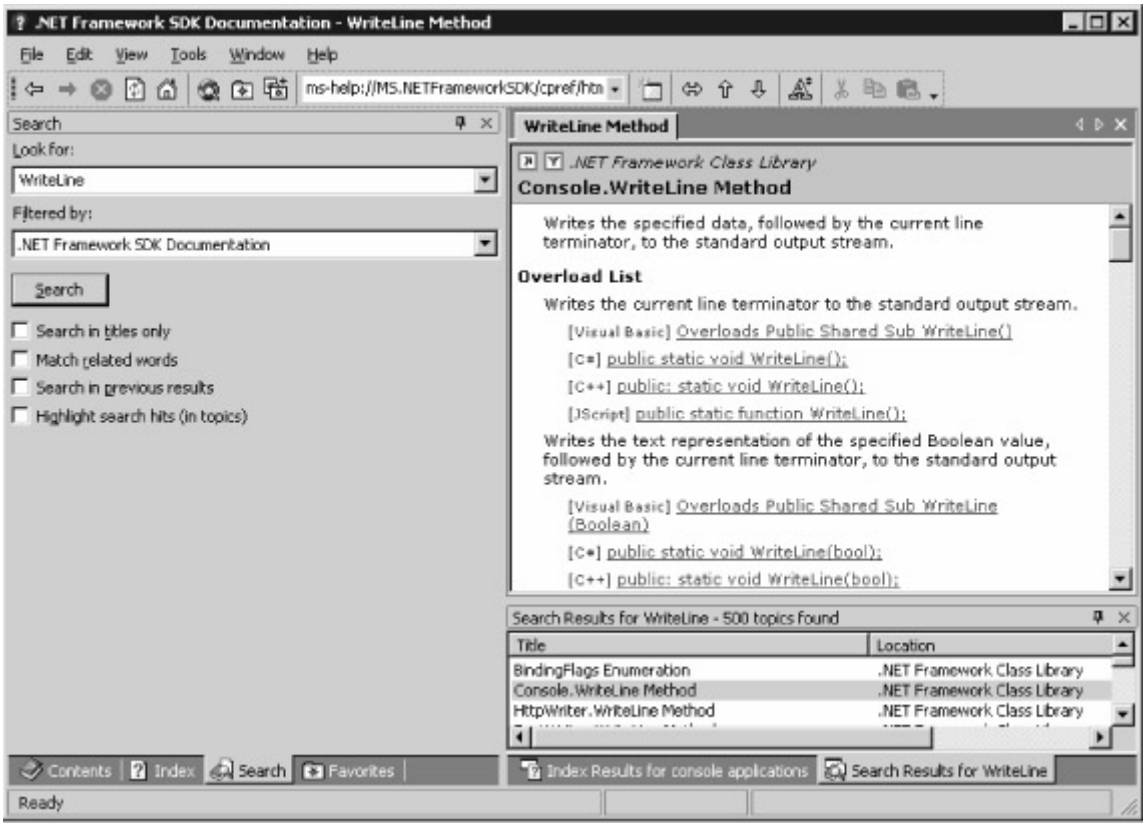
Hình 1.6: tìm kiếm chỉ số cho từ console

Bạn có thể cũng tìm kiếm tất cả các trang trong tài liệu sử dụng thẻ Search . Bạn trình bày thẻ Search bởi chọn nó ở đáy của màn hình.

Mẹo nhỏ:

Bạn cũng có thể xem cửa sổ Search bởi chọn Help ➢ Search , hay bởi nhấn trl+ Alt+ F3 trên bàn phím.

Bạn nhập những từ Bạn muốn tìm kiếm trong trường Look For của cửa sổ Search. Hình 1.7 cho thấy trang Search và kết quả tìm kiếm trả về bởi một sự tìm kiếm cho WriteLine. Khi bạn chạy sự tìm kiếm, những tên của những trang chứa những từ được yêu cầu của bạn được trình bày trong cửa sổ kết quả Tìm kiếm xuất hiện ở đáy màn hình (Bạn có thể xem cửa sổ này trong Hình 1.7).



Hình 1.7: Tìm kiếm tất cả tài liệu cho từ "WriteLine"

Mẹo nhỏ:

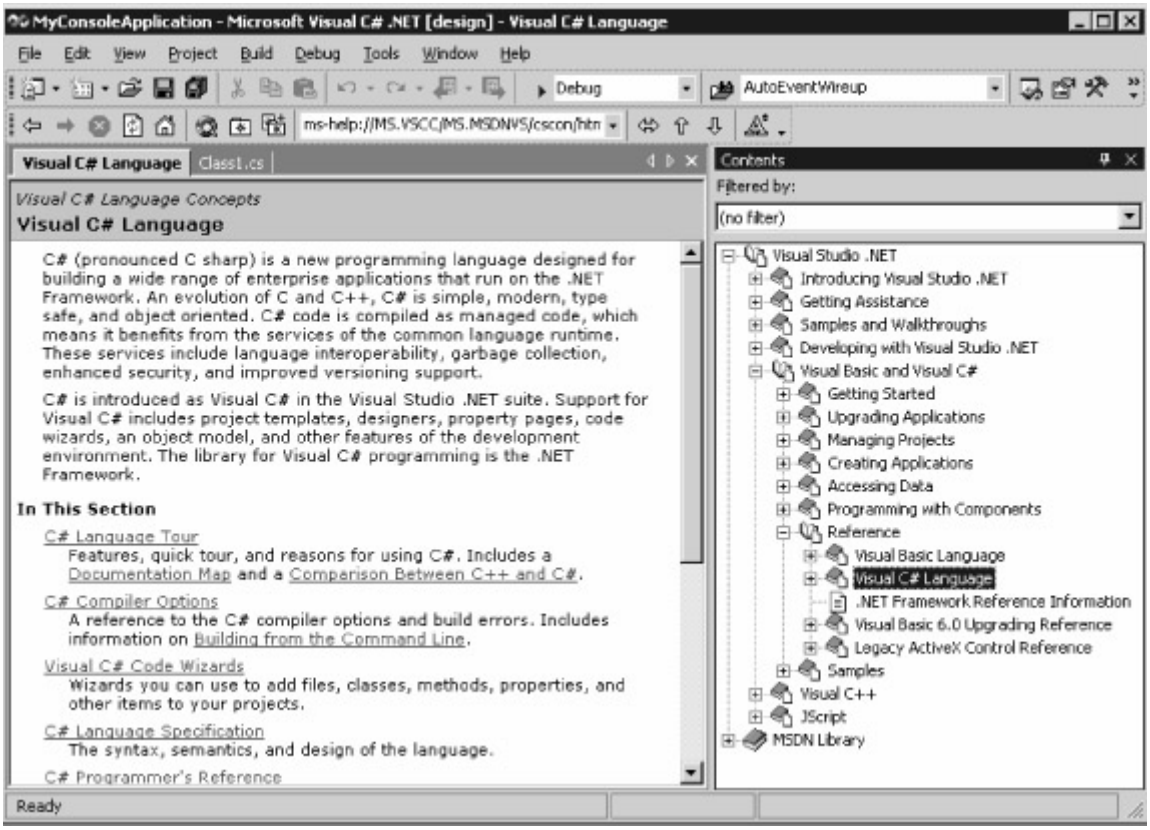
Bạn cũng có thể xem cửa sổ kết quả tìm kiếm bởi việc chọn Help ➢ Search , hay bởi nhấn Shift+ Alt+ F3 trên bàn phím .

Bạn xem nội dung của một trang đặc biệt được trình bày trong cửa sổ những kết quả tìm kiếm bởi việc nhấn đúp dòng thích hợp. Chẳng hạn, Trong Hình 1.7, chúng tôi nhấn đúp dòng thứ hai trong cửa sổ kết quả tìm kiếm. Dòng này chứa trang với tiêu đề "Console.WriteLine Method," và như bạn có thể thấy, trang này được trình bày trong cửa sổ bên trên "Search Results" Hình 1.7.

Trong mục kế tiếp, bạn sẽ thấy cách truy cập tài liệu - sử dụng VS .NET.

Việc truy cập Tài liệu sử dụng VS .NET

Nếu Bạn đang sử dụng VS .NET, Bạn truy cập tài liệu sử dụng thực đơn Help. Để truy cập nội dung của tài liệu, bạn chọn Help ➢ Contents. Hình 1.8 cho thấy nội dung được trình bày trong VS .NET. Chú ý tài liệu được trình bày trực tiếp trong VS .NET, thay vì trong một cửa sổ riêng biệt, như nó đã làm khi xem tài liệu với NET SDK..



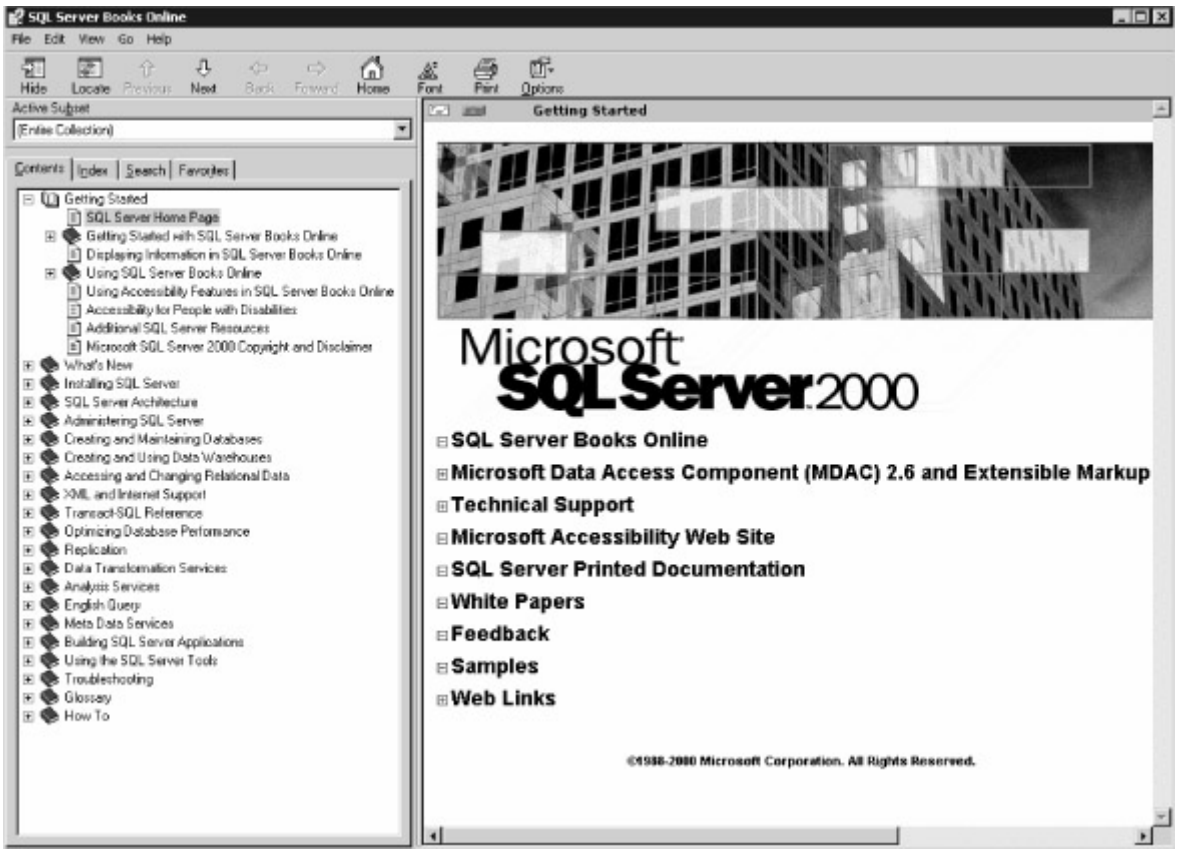
Hình 1.8: nội dung tài liệu được xem trong VS .NET

Ghi chú Những phím tắt bàn phím giống như vậy được trình bày trong mục trước đây cũng ứng dụng cho VS .NET.

Thực đơn Help cũng cung cấp sự truy cập tới chỉ số (Index) và của sổ tìm kiếm tương tự khi bạn xem thấy trong mục trước đây.

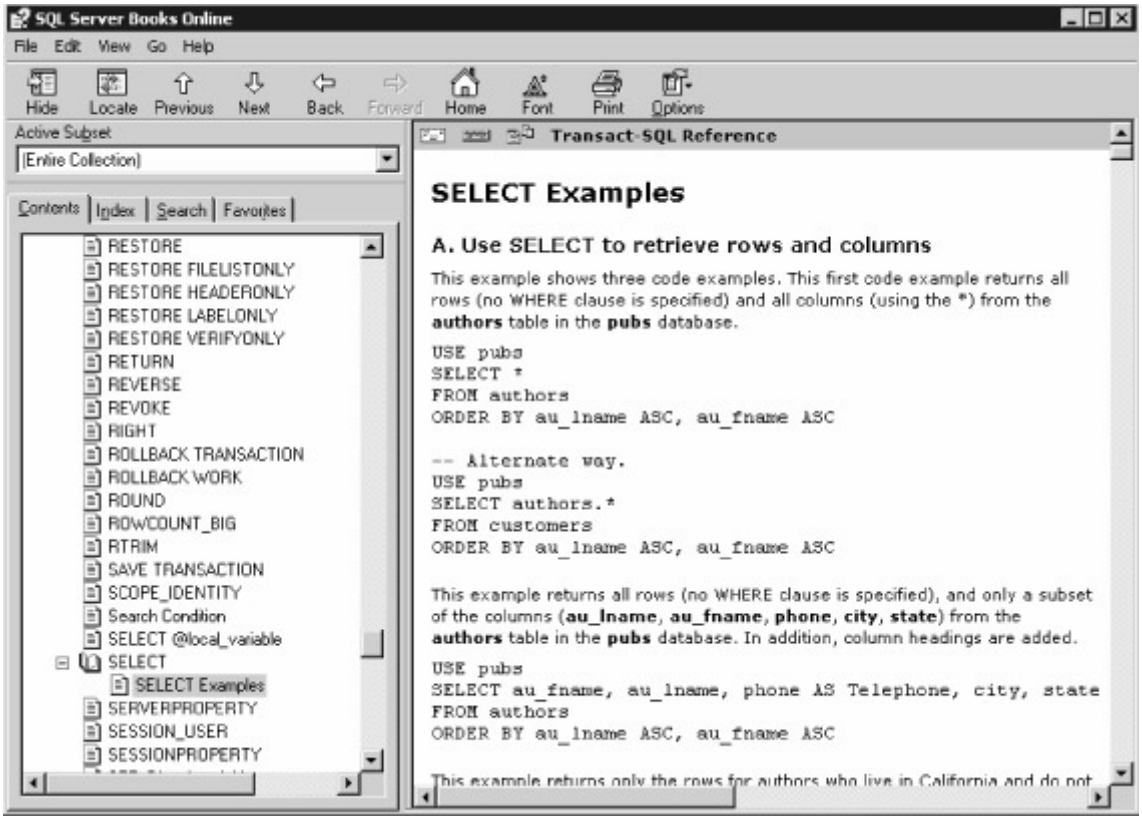
Sử dụng Tài liệu SQL Server

Máy chủ phục vụ SQL cũng đi cùng với tài liệu điện tử rộng lớn. Để truy cập tài liệu này, bạn chọn Start ➢ Programs ➢ Microsoft SQL Server ➢ Books Online. Hình 1.9 cho thấy trang chủ tài liệu Máy chủ phục vụ SQL.



Hình 1.9: trang chủ tài liệu Máy chủ phục vụ SQL

Bạn có thể duyệt những sách trực tuyến sử dụng thẻ Nội dung (Contents), và Bạn có thể tìm kiếm thông tin đặc biệt sử dụng chỉ số (Index) và những thẻ Search. Hình 1.10 cho thấy một số thông tin cho phát biểu SELECT, được định vị trong sách tham khảo Giao dịch-SQL.



Hình 1.10: Tài liệu về những ví dụ SELECT

Ghi chú : Giao dịch- SQL là sự thực thi đầy đủ của Microsoft của SQL và chứa những mở rộng lập trình. Bạn sẽ học về lập trình Giao dịch-SQL trong Chương 4.

Bạn có thể tự xem thông tin trong Hình 1.10 bởi mở Contents ➢ Transact-SQL Reference ➢ SELECT ➢ SELECT Examples.

Tóm lược

Một cơ sở dữ liệu là một tập hợp được tổ chức của thông tin được chia vào trong những bảng. Mỗi bảng lại được chia vào trong những hàng và những cột ; những cột này lưu trữ thông tin thực tế. Bạn truy cập một cơ sở dữ liệu sử dụng ngôn ngữ truy vấn có cấu trúc (SQL), là một ngôn ngữ tiêu chuẩn được hỗ trợ bởi đa số phần mềm cơ sở dữ liệu bao gồm SQL Server, Access, và Oracle.

Bạn đã thấy một chương trình C# kết nối tới một cơ sở dữ liệu Máy chủ phục vụ SQL, đã truy xuất và trình bày nội dung được lưu trữ trong những cột của một hàng từ một bảng, và rồi ngắt kết nối với cơ sở dữ liệu. Bạn cũng nhìn thấy những chương trình kết nối tới một cơ sở dữ liệu Access và một Oracle.

Công cụ phát triển nhanh ứng dụng của Microsoft là Visual Studio .NET(VS .NET). VS .NET cho phép bạn phát triển và chạy những chương trình trong một môi trường phát triển tích hợp. Môi trường này sử dụng tất cả những đặc tính lớn của Windows, như con chuột và những thực đơn trực quan, và tăng năng suất của bạn như một lập trình viên.

Trong những mục cuối cùng của chương này, bạn đã thấy cách sử dụng tài liệu rộng lớn từ Microsoft đi cùng với Đồ dùng phát triển phần mềm .NET (SDK). và VS .NET. Bạn cũng đã thấy cách sử dụng tài liệu Máy chủ phục vụ SQL như thế nào.

Trong chương kế tiếp, bạn sẽ học nhiều hơn về những cơ sở dữ liệu.

$$

Chương 2: Giới thiệu về những cơ sở dữ liệu

Tổng quan

Trong chương này, bạn sẽ học về cơ sở của những cơ sở dữ liệu: cấu trúc của những cơ sở dữ liệu như thế nào, Làm sao để tạo ra và liên hệ những bảng, Và làm sao để xây dựng những câu truy vấn để truy xuất thông tin. Chương này cũng chỉ cho bạn làm cách nào để sử dụng một cơ sở dữ liệu Máy chủ phục vụ SQL có tên Northwind. Cơ sở dữ liệu này chứa thông tin cho một Công ty tưởng tượng Northwind, bán những sản phẩm thức ăn. Cơ sở dữ liệu này là một trong số những cơ sở dữ liệu ví dụ được thiết đặt với SQL Server điển hình. Bạn có thể thu được một phiên bản thử của Máy chủ phục vụ SQL từ website của Microsoft tại www.microsoft.com <http://www.microsoft.com>.

> **Ghi chú:**
>
> Trong thời gian ghi, bạn có thể tải phiên bản thử của máy chủ phục vụ SQL từ website của Microsoft. Nếu kết nối Internet của bạn quá chậm, bạn cũng có thể mua một đĩa CD-ROM chứa phiên bản thử.

Tôi sử dụng phiên bản phát triển của Máy chủ phục vụ SQL khi chuẩn bị sách này. Khi chạy một hệ thống sản xuất, bạn cần phải sử dụng cách đặc trưng phiên bản Doanh nghiệp của máy chủ phục vụ SQL. Bạn có thể xem những sự khác nhau giữa những kiểu Máy chủ phục vụ SQL ở Microsoft website.

Đặc trưng trong chương này:

- Giới thiệu những cơ sở dữ liệu
- Sử dụng Máy chủ phục vụ SQL
- Khám phá cơ sở dữ liệu Northwind
- Xây dựng câu truy vấn sử dụng Enterprise Manager (Quản trị doanh nghiệp)
- Tạo ra một bảng

Giới thiệu những cơ sở dữ liệu

Một cơ sở dữ liệu là một tập hợp được tổ chức của thông tin. Một cơ sở dữ liệu quan hệ (relational database) là một tập hợp của thông tin liên quan đã được tổ chức vào trong những cấu trúc được biết như những bảng. Mỗi bảng chứa những hàng (rows) được sắp xếp vào trong những cột (columns). Bạn cần phải đã quen thuộc với thông tin được tham chiếu trong form của một bảng với những cột. Chẳng hạn, Bảng 2.1 cho thấy những chi tiết của một số sản phẩm được bán bởi Công ty Northwind. Bảng 2.1 liệt kê ID của sản phẩm, tên, số lượng trên đơn vị, và đơn giá cho 10 sản phẩm đầu tiên;

Bảng 2.1: Một số hàng từ Bảng những sản phẩm

PRODUCT ID (mã sản phẩm)	NAME (tên sản phẩm)	QUANTITY PER UNIT (số lượng trên đơn vị)	Unit Price (đơn giá)
1	Chai	10 boxes x 20 bags	$18
2	Chang	24-12oz bottles	$19
3	Aniseed Syrup	12-550ml bottles	$10
4	Chef Anton's Cajun Seasoning	48-6oz jars	$22
5	Chef Anton's Gumbo Mix	36 boxes	$21.35
6	Grandma's Boysenberry Spread	12-8oz jars	$25
7	Uncle Bob's Organic Dried Pears	12-1lb pkgs.	$30
8	Northwoods Cranberry Sauce	12-12oz jars	$40
9	Mishi Kobe Niku	18-500g pkgs.	$97
10	Ikura	12-200ml jars	$31

Bạn có thể lưu trữ thông tin trong một cơ sở dữ liệu trên giấy trong một tủ đựng hồ sơ hay trong định dạng điện tử được lưu trữ trong bộ nhớ và hệ thống tập tin của một máy tính. Hệ thống được dùng để quản lý thông tin trong cơ sở dữ liệu là hệ quản trị cơ sở dữ liệu (database management system). Trong trường hợp của một cơ sở dữ liệu điện tử, hệ quản trị cơ sở dữ liệu là phần mềm quản lý thông tin trong những bộ nhớ và hồ sơ của máy tính. Một ví dụ về phần mềm như vậy là Máy chủ phục vụ SQL (Đây là hệ quản trị cơ sở dữ liệu có quan hệ, hay RDBMS, sử dụng trong sách này). Những thí dụ khác về phần mềm RDBMS bao gồm Oracle và DB2.

> **Ghi chú:**
>
> Bạn phải cẩn thận để phân biệt giữa một cơ sở dữ liệu và một hệ quản trị cơ sở dữ liệu. Một cơ sở dữ liệu là một tập hợp được tổ chức của thông tin, và một hệ quản trị cơ sở dữ liệu là phần mềm cất giữ và cung cấp những công cụ để thao tác thông tin được cất giữ. Sự phân biệt này bị làm mờ trong những ngày gần đây, vì vậy thuật ngữ cơ sở dữ liệu thường được dùng để tham chiếu tới phần mềm.

Vấn đề khác bạn cần thông thuộc là một mô hình cơ sở dữ liệu, đó là một sự trình bày của cấu trúc dữ liệu, và bao gồm định nghĩa của những bảng và những cột tạo thành cơ sở dữ liệu.

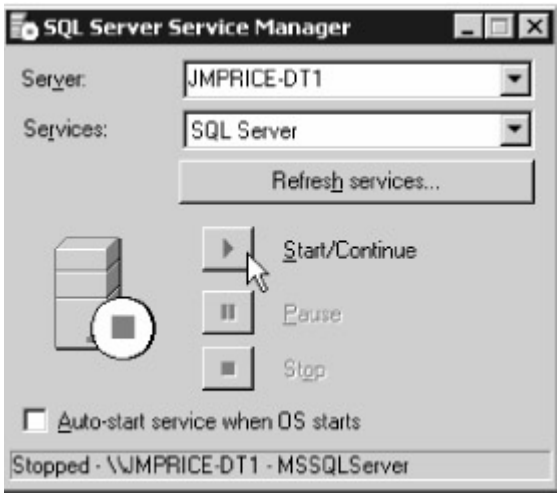
Trong mục kế tiếp, bạn sẽ khám phá SQL Server.

Sử dụng SQL Server

Trong mục này, bạn sẽ khám phá vài công cụ mà bạn sử dụng để quản lý SQL Server. Đặc biệt, bạn sẽ học cách khởi động và dừng Người phục vụ SQL sử dụng Trình quản lý dịch vụ (Service Manager) và sử dụng Trình quản lý doanh nghiệp (Enterprise Manager) để điều hành SQL Server.

Khởi chạy và dừng SQL Server

Để khởi chạy và dừng Trình phục vụ SQL, bạn sử dụng công cụ Service Manager . Để mở Service Manager, bạn chọn Start □ Programs □ Microsoft SQL Server □ Service Manager. Service Manager được trình bày trong Hình 2.1.

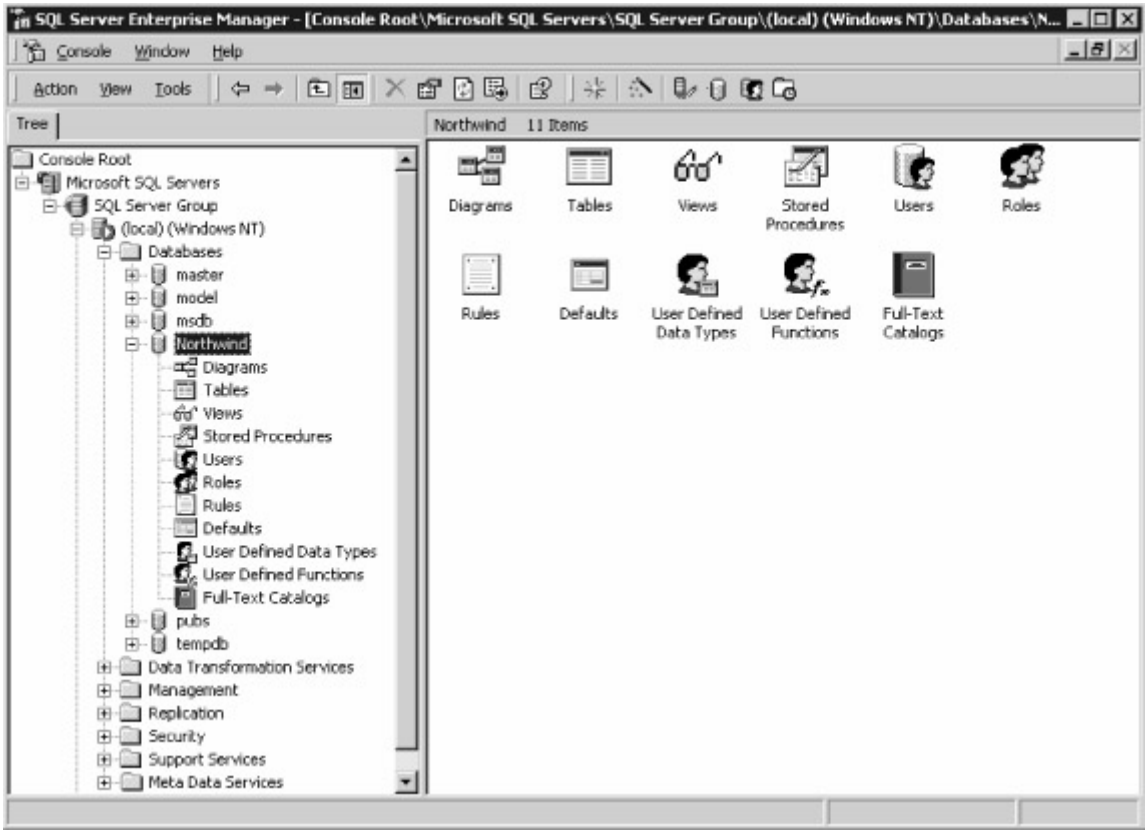


Hình 2.1: cửa sổ trình quản trị dịch vụ

Bạn chọn tên của máy tính người phục vụ trên đó Trình phục vụ SQL đang chạy trong hộp danh sách thả xuống của cửa sổ trình phục vụ. Để khởi động Trình phục vụ SQL, bạn kích nút Start/Contrinue. Để dừng Trình phục vụ SQL, bạn kích nút Stop. Bạn cũng có thể sử dụng Service Manager để tạm dừng trình phục vụ SQL , và chọn nếu bạn muốn tự động khởi động trình phục vụ SQL khi hệ điều hành khởi động.

Sử dụng Enterprise Manager (trình quản trị doanh nghiệp)

Để điều hành một cơ sở dữ liệu, bạn sử dụng công cụ Enterprise Manager. Bạn có thể sử dụng Enterprise Manager để tạo những cơ sở dữ liệu, tạo và soạn thảo những bảng, tạo và soạn thảo những người sử dụng, vân vân. để mở Enterprise Manager, bạn chọn Start □ Programs □ Microsoft SQL Server □ Enterprise Manager. Enterprise Manager được trình bày trong Hình 2.2.



Hình 2.2: Enterprise Manager

Trong khung bên trái của Enterprise Manager , bạn sẽ thấy một cây trình bày những sự cài đặt Máy chủ phục vụ SQL có thể tiếp cận. Nội dung của khung bên phải của Enterprise Manager trình bày thông tin khác nhau dựa vào những gì mà bạn chọn trong khung bên trái. Chẳng hạn, Tôi đã chọn thư mục những cơ sở dữ liệu và cơ sở dữ liệu North-wind trong khung bên trái khi chuẩn bị Hình 2.2. Như bạn có thể thấy, khung bên phải trình bày những biểu tượng cho phép bạn soạn thảo những tiết mục được lưu trữ trong cơ sở dữ liệu này.

Mỗi sự cài đặt Máy chủ phục vụ SQL chứa bảy thư mục sau được trình bày trong khung bên trái:

Databases: chứa những công cụ cho phép bạn truy cập những cơ sở dữ liệu được quản lý bởi Người phục vụ SQL.

Data Transformation Services : cung cấp những truy cập tới công cụ cho phép bạn di chuyển Dữ liệu từ cơ sở dữ liệu này sang cơ sở dữ liệu khác. Bạn cũng có thể chuyển đổi dữ liệu một cách tự động khi nó được di chuyển. Chẳng hạn, bạn có lẽ muốn di chuyển dữ liệu từ cơ sở dữ liệu Người phục vụ SQL đến một cơ sở dữ liệu Oracle, hay ngược lại.

Management: chứa những công cụ cho phép bạn sao lưu những cơ sở dữ liệu của bạn, theo dõi hoạt động cơ sở dữ liệu hiện thời, và những nhiệm vụ khác.

Replication: cung cấp truy cập những công cụ cho phép bạn sao chép thông tin từ cơ sở dữ liệu này sang cơ sở dữ liệu khác trong thời gian thực sử dụng một quá trình được biết như bản sao (replication). Chẳng hạn, bạn có lẽ muốn di chuyển dữ liệu từ một cơ sở dữ liệu đang chạy tại một chi nhánh của một công ty đến một cơ sở dữ liệu tại trụ sở chính.

Security: chứa những công cụ cho phép bạn quản lý những đăng nhập và những vai trò gắn sẵn chứa những sự cho phép. Bạn cũng có thể quản lý những Máy chủ phục vụ và những Máy chủ phục vụ liên kết từ xa. Những Máy chủ phục vụ được liên kết là những cơ sở dữ liệu mà bạn có thể truy cập qua một mạng. Những cơ sở dữ liệu này không phải là những cơ sở dữ liệu Máy chủ phục vụ SQL; chẳng hạn,chúng cũng có thể là những cơ sở dữ liệu Oracle. Sự giới hạn duy nhất là phải có một trình cung cấp DB OLE (sự liên kết và nhúng đối tượng cho những cơ sở dữ liệu) cho cơ sở dữ liệu này. Những máy chủ phục vụ từ xa là những cơ sở dữ liệu SQL Server mà bạn có thể truy cập qua một mạng và chạy những thủ tục lưu trữ trên đó.

Support Services: Cung cấp sự truy cập tới những công cụ cho phép bạn quản lý Trình phối hợp Giao dịch Phân tán (Distributed Transaction Coordinator), sự tìm kiếm toàn bộ văn bản và những dịch vụ bưu chính SQL. Dịch vụ phối hợp giao dịch phân phối cho phép bạn quản lý những giao dịch sử dụng nhiều cơ sở dữ liệu. Dịch vụ tìm kiếm toàn văn cho phép bạn thực hiện tìm kiếm những mệnh đề thông qua

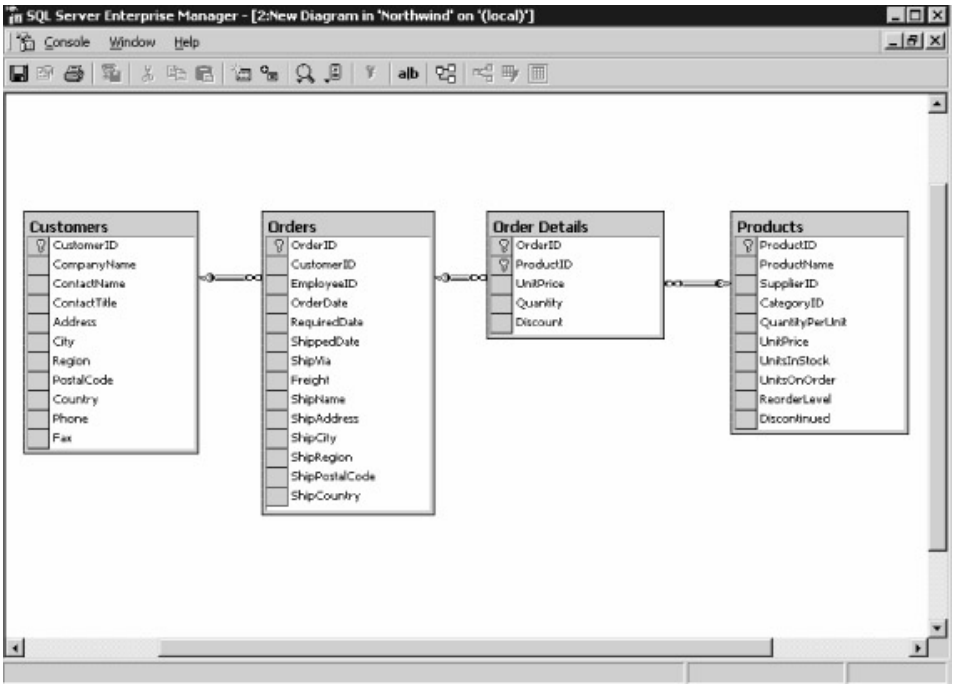
những số lượng lớn văn bản. Dịch vụ bưu chính SQL cho phép bạn gửi thư điện tử từ máy chủ phục vụ SQL.

Meta Data Services : chứa những công cụ cho phép bạn quản lý thông tin được lưu trữ trong kho chứa địa phương. Thông tin này chứa những chi tiết về những cơ sở dữ liệu, những người sử dụng, những bảng, những cột, những bảng view, những thủ tục lưu trữ vân vân. Thông tin này chủ yếu được sử dụng bởi những ứng dụng kho dữ liệu.

Ghi nhớ: Vì đây là một sách về lập trình cơ sở dữ liệu, Tôi sẽ không bao trùm quá nhiều chi tiết về quản trị cơ sở dữ liệu; Tôi sẽ chỉ tập trung vào thư mục những cơ sở dữ liệu . Điển hình, tổ chức của bạn sẽ có một người quản trị cơ sở dữ liệu, hay DBA, là người chăm sóc điều hành những cơ sở dữ liệu của bạn và sẽ sử dụng những thư mục khác để thực hiện những nhiệm vụ của họ. Nếu bạn cần nhiều chi tiết về điều hành Máy chủ phục vụ SQL hơn, Tôi giới thiệu bạn sách "Mastering SQL Server 2000 by Mike Gunderloy and Joseph L. Jorden (Sybex, 2000)".

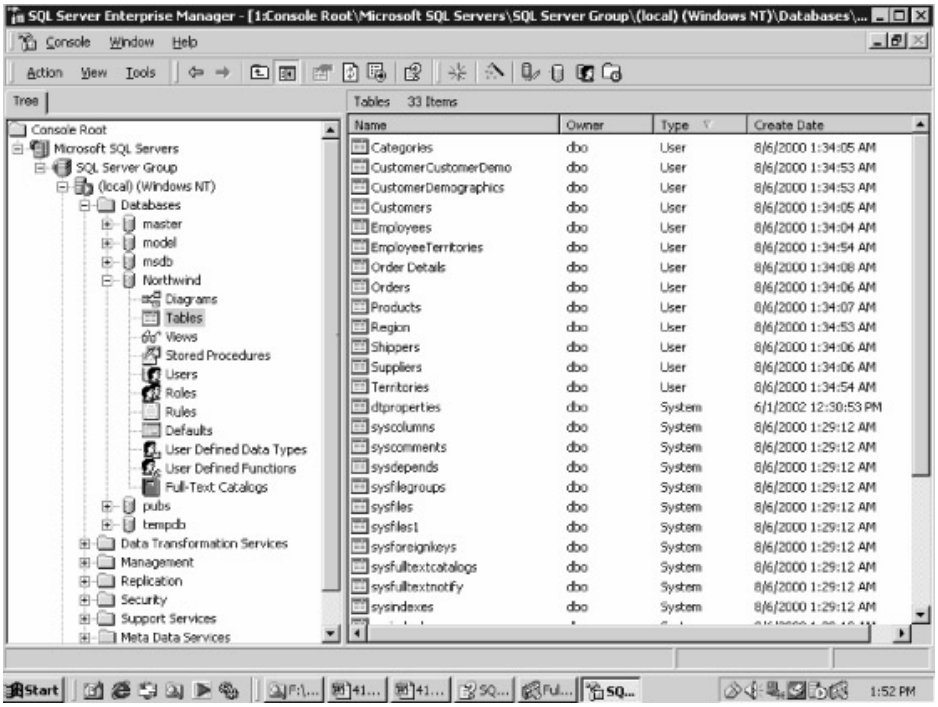
Chúng ta xem sét kỹ hơn tại thư mục những cơ sở dữ liệu, chứa những cơ sở dữ liệu được quản lý bởi một sự cài đặt trình máy chủ phục vụ SQL đặc biệt. Chẳng hạn, sự cài đặt máy chủ phục vụ SQL của tôi quản lý sáu cơ sở dữ liệu có tên master, model, msdb, Northwind, pubs, và tempdb. Khi bạn mở rộng thư mục những cơ sở dữ liệu cho một cơ sở dữ liệu, bạn sẽ nhìn thấy những nút sau đây:

Diagrams : Bạn sử dụng một sơ đồ để lưu trử một sự trình bày trực quan của những bảng trong một cơ sở dữ liệu. Chẳng hạn, cơ sở dữ liệu Northwind chứa nhiều bảng, bốn bảng có tên sau: Customers, Orders, Order Details, và Products. Hình 2.3 minh họa mối liên quan giữa những bảng này như thế nào. Những cột thuộc mỗi bảng được trình bày bên trong mỗi hộp trong sơ đồ. Chẳng hạn, bảng những khách hàng chứa 11 cột: CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, và Fax. Như bạn sẽ học trong những mục " những mối quan hệ bảng và những khóa ngoại ", những đường nối giữa những Bảng cho thấy những mối quan hệ giữa nhiều Bảng.



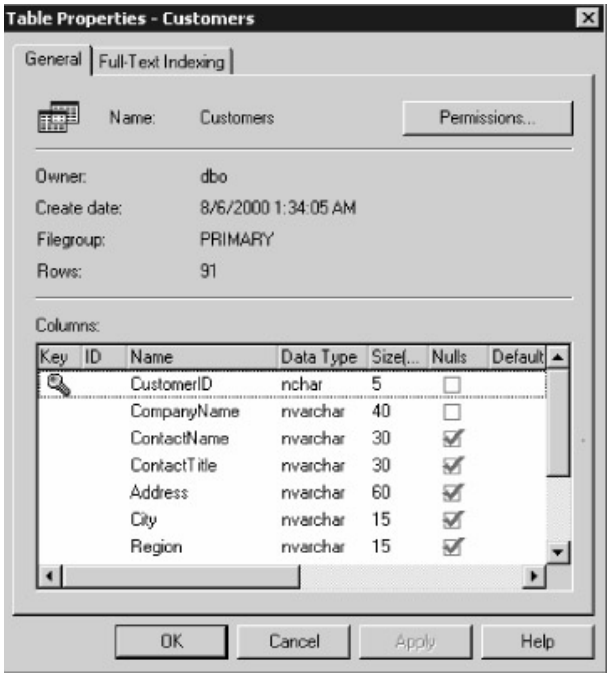
Hình 2.3: những bảng Customers, Orders, Order Details, và Products

Tables: Bạn sử dụng một bảng để lưu trữ những hàng được chia vào trong những cột. Hình 2.4 cho thấy một danh sách của những bảng được cất giữ trong cơ sở dữ liệu Northwind.



Hình 2.4: những bảng của cơ sở dữ liệu Northwind

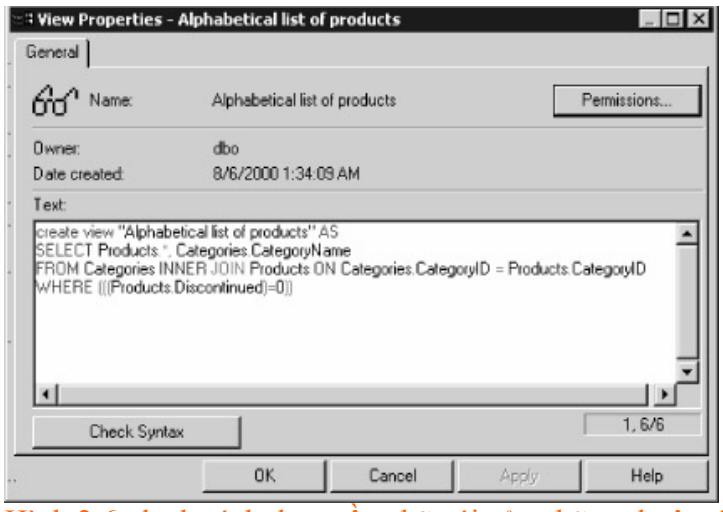
Bạn có thể tạo ra những bảng mới, xem những thuộc tính của một bảng, và truy vấn những hàng cất giữ trong một bảng. Bạn sẽ học cách tạo ra một bảng mới như thế nào sau trong mục "Creating a Table" . Để xem những thuộc tính của một bảng, bạn chọn bảng từ danh sách trong khung bên phải, kích nút chuột phải, và chọn những thuộc tính từ thực đơn nhạy ngữ cảnh sổ ra. Bạn cũng có thể nhấn đúp bảng để hiển thị những thuộc tính, và Hình 2.5 cho thấy những thuộc tính của bảng những khách hàng. Bạn sẽ học ý nghĩa của những thuộc tính này khi chương này tiến triển.



Hình 2.5: những thuộc tính của bảng những khách hàng

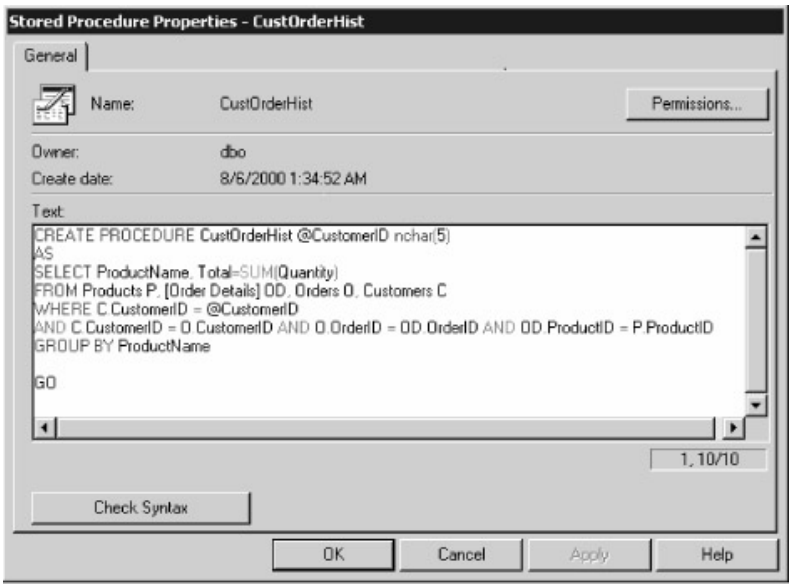
Views : Bạn sử dụng một view để truy xuất một tập hợp của những cột từ một hoặc nhiều bảng. Bạn có thể hiểu một view như một cách khảo sát linh hoạt hơn những hàng được cất giữ trong những bảng . Chẳng hạn, một trong số những views của cơ sở dữ liệu Northwind truy xuất một danh sách xếp theo vần chữ cái của những sản phẩm, và truy xuất tên sản phẩm và tên loại, trong số những cột khác. Thông tin này đến từ cả bảng Products lẫn bảng Categories. Bạn cũng có thể tạo những view mới, khảo sát những thuộc tính của một view, và truy vấn những hàng thông qua một view. Để khảo sát những thuộc tính của một view, bạn chọn view này, kích nút chuột phải, và chọn Properties. Bạn có thể cũng nhấn đúp view này để khảo sát những thuộc tính. Hình 2.6 cho thấy danh sách những thuộc tính xếp theo vần chữ cái của bảng view những sản phẩm . Văn bản của view được

viết trong SQL, bạn sẽ học nhiều hơn trong Chương 3, cùng với cách sử dụng view trong Chương này như thế nào.



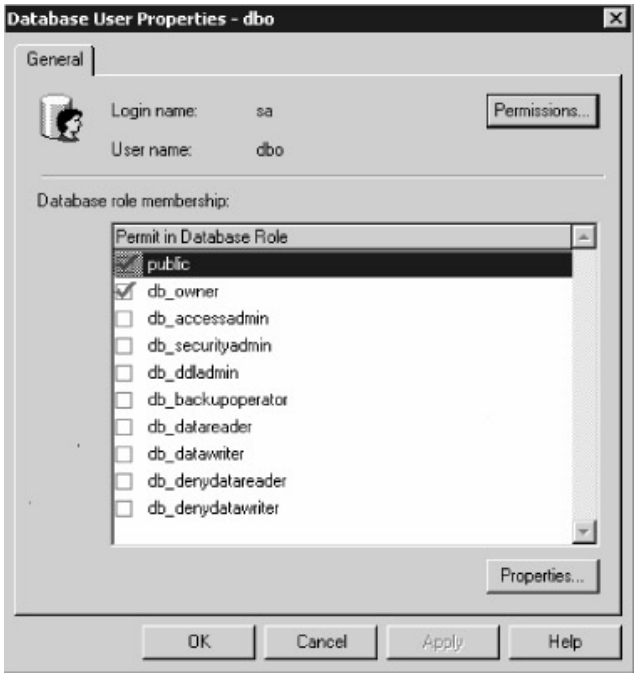
Hình 2.6: danh sách theo vần chữ cái của những thuộc tính bảng view những sản phẩm

Stored Procedures: Bạn sử dụng một thủ tục lưu trữ để chạy một chuỗi của những sự phát biểu trong cơ sở dữ liệu. Trong Máy chủ phục vụ SQL, những thủ tục lưu trữ được viết với Transact- SQL, mà bạn sẽ học về nó trong Chương 4. Những thủ tục lưu trữ được cất giữ trong cơ sở dữ liệu, và điển hình được sử dụng khi bạn cần thực hiện một tác vụ ,tác vụ này sử dụng cơ sở dữ liệu với cường độ cao, hay bạn muốn tập trung một hàm trong cơ sở dữ liệu mà bất kỳ người sử dụng nào có thể gọi hơn là mỗi người sử dụng phải viết chương trình riêng của mình để thực hiện tác vụ giống như vậy. Chẳng hạn, một trong số những thủ tục lưu trữ trong cơ sở dữ liệu Northwind có tên CustOrdHist, nó trả về tên sản phẩm và tổng số lượng của những sản phẩm được đặt mua bởi một khách hàng riêng biệt, người mà được gởi qua như một tham số tới thủ tục. Hình 2.7 cho thấy rằng những thuộc tính của thủ tục lưu trữ CustOrdHist .



Hình 2.7: những thuộc tính của thủ tục lưu trữ CustOrdHist

Users: Mỗi khi bạn truy cập cơ sở dữ liệu, bạn kết nối tới một tài khoản người sử dụng riêng biệt trong cơ sở dữ liệu. Mỗi cơ sở dữ liệu SQL Server đi cùng với hai người sử dụng mặc định có tên dbo và guest. Người sử dụng dbo sở hữu cơ sở dữ liệu và có những quền hạn để làm bất cứ điều gì trong cơ sở dữ liệu, như tạo ra những bảng mới, sửa đổi bảng vân vân. Người sử dụng guest có những sự giới hạn về quyền hạn - cho phép sự truy cập tới nội dung của những bảng, nhưng không thể tạo ra hay sửa đổi những bảng, vân vân. Hình 2.8 cho thấy những thuộc tính của người sử dụng dbo. Bạn chú ý là người sử dụng dbo được coi như có hai vai trò public và db_owner. Bạn sẽ học về những vai trò tiếp theo. Bạn có thể xem tất cả những quyền hạn được gán tới người sử dụng dbo bởi kích nút Permissions.



Hình 2.8: những thuộc tính người sử dụng dbo

Roles : Một vai trò là một tập hợp có tên của những quền hạn mà bạn có thể gán cho một người sử dụng. Nó hữu ích khi bạn cần gán tập hợp giống như vậy của những quền hạn tới nhiều người sử dụng. Cách này, nếu bạn cần thay đổi tập hợp của những quền hạn, bạn chỉ cần thay đổi những quền hạn được gán tới vai trò, hơn là những quền hạn gán tới mỗi người sử dụng. Chẳng hạn, bạn thấy trong hình trước đây người sử dụng dbo được coi là có vai trò Public và db_owner. Hình 2.9 cho thấy những thuộc tính của vai trò public . Bạn chú ý là vai trò public cũng được coi là người sử dụng khách (guest). Nếu không có vai trò public nào được sử dụng, thì tập hợp của những quền hạn phải được bổ sung bằng tay tới cả dbo lẫn những người sử dụng khách (guest).

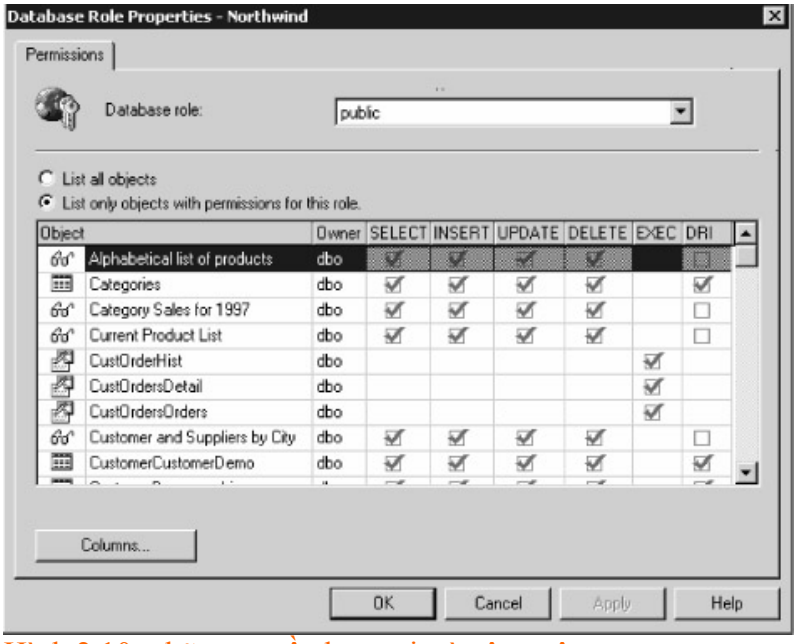
Hình 2.9: những thuộc tính vai trò công cộng (public)

Bạn có thể xem những quền hạn được gán cho một vai trò bởi kích nút Permissions . Hình 2.10 cho thấy những thuộc tính được gán tới vai trò Public, và những danh sách 2.2 liệt kê những ý nghĩa của những quyền hạn sẵn có.

Bảng 2.2: ý nghĩa của những quyền hạn sẵn có

Quyền hạn	Ý nghĩa
SELECT	Cho phép truy xuất những hàng từ một bảng hay view.

INSERT	Cho phép thêm những hàng vào trong một bảng hay view.
UPDATE	Cho phép sủa đổi những hàng trong một bảng hay view.
DELETE	Cho phép loại bỏ những hàng từ một bảng hay view.
EXEC	Cho phép thực thi một thủ tục lưu trữ.
DRI	Cho phép thêm hay loại bỏ những ràng buộc, sự toàn vẹn, liên hệ, sự khai báo(DRI) tới một bảng. Những sự ràng buộc bảo đảm rằng những hoạt động thích hợp được sử lý khi thêm, điều chỉnh, hay loại bỏ những giá trị khóa ngoại. Những khóa ngoại chỉ rõ một cột trong một bảng liên quan đến một cột trong bảng khác. Bạn sẽ học về những khóa ngoại nhiều hơn trong những mục " Những mối quan hệ và những khóa ngoại ".



Hình 2.10: những quyền hạn vai trò công cộng

Rules : Một quy tắc là một biểu thức mà định giá trị tới true hay false và xác định liệu bạn có thể gán một giá trị riêng biệt tới một cột không. Chẳng hạn, bạn có lẽ đã định nghĩa một quy tắc chỉ rõ một phạm vi của những giá trị, và nếu một giá trị được cung cấp bên ngoài phạm vi này, thì bạn không thể gán giá trị tới cột này. Những quy tắc được cung cấp cho tính tương thích với những phiên bản cũ hơn của SQL Server và bây giờ được thay thế bởi những ràng buộc. Bạn sẽ học về những ràng buộc nhiều hơn trong mục " Tạo ra một ràng buộc " tiếp sau .

Defaults: Một giá trị mặc định là một giá trị ban đầu được gán khi bạn thêm một hàng mới vào một bảng. Những mặc định được cung cấp cho tính tương thích với những phiên bản cũ hơn của SQL Server và bây giờ được thay thế bởi giá trị mặc định của một cột. Bạn sẽ học nhiều hơn về những giá trị mặc định trong mục " Tạo ra một Bảng " sau .

User-Defined Data Types: những kiểu dữ liệu do người dùng định nghĩa cho phép bạn tạo ra những kiểu của mình dựa vào những kiểu SQL Server hiện hữu. Chẳng hạn, cho là bạn muốn lưu trữ một mã ZIP code Mỹ trong vài bảng của cơ sở dữ liệu của bạn; bạn có thể tạo ra một kiểu để lưu trữ một chuỗi năm ký tự. Và rồi nếu bạn muốn tăng chiều dài từ năm đến tám để lưu trữ một mã ZIP code mở rộng, thì tất cả những gì bạn cần làm là sửa đổi kiểu của bạn và sự thay đổi sẽ được phản ánh trong tất cả những bảng nơi bạn sử dụng kiểu này.

Full-Text Catalogs : những danh mục văn bản đầy đủ cho phép bạn tạo ra một chỉ số toàn bộ văn bản, mà cho phép bạn thực hiện tìm kiếm những mệnh đề thông qua những số lượng lớn văn bản.

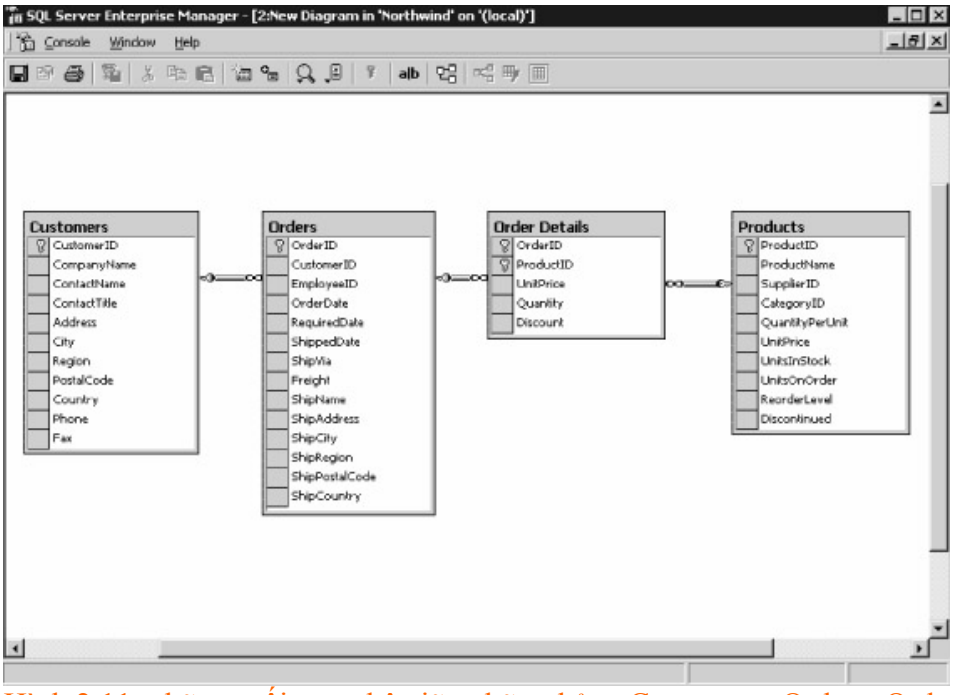
Trong chương kế tiếp, bạn sẽ thấy Trình duyệt Server của Visual Studio .NET cũng cho phép bạn sử dụng nhiều đặc trưng giống nhau chứa trong thư mục những cơ sở dữ liệu của Enterprise Manager. Đặc biệt, Trình duyệt Server cho phép bạn xem, tạo ra, và sửa đổi những mục sau đây: những sơ đồ cơ sở dữ liệu (database diagrams), những bảng, những views , những thủ tục lưu trữ (stored procedures), và những hàm do người dùng

định nghĩa.

Trong mục sau đây, bạn sẽ về ý nghĩa của thuật ngữ relational (có quan hệ) trong ngữ cảnh của một cơ sở dữ liệu quan hệ, và bạn sẽ khám phá một số những bảng trong cơ sở dữ liệu Northwind.

Khám phá Cơ sở dữ liệu Northwind

Một cơ sở dữ liệu có thể có nhiều bảng, Một số chúng có liên hệ lẫn nhau. Chẳng hạn, cơ sở dữ liệu North-wind chứa nhiều bảng, bốn trong số đó có tên: Customers, Orders, Order Details, và Products. Hình 2.11 là một sự lặp lại sơ đồ được trình bày trước đó minh họa mối liên hệ của những bảng này.



Hình 2.11: những mối quan hệ giữa những bảng Customers, Orders, Order Details, và Products

Những cột cho mỗi bảng được trình bày bên trong mỗi hộp. Chẳng hạn, bảng khách hàng chứa đựng 11 cột:

- CustomerID
- CompanyName
- ContactName
- ContactTitle
- Address
- City
- Region
- PostalCode
- Country
- Phone
- Fax

Trong vài mục kế tiếp, bạn sẽ học một số lý thuyết về cơ sở dữ liệu, rồi bạn sẽ học mỗi cột trong những cột trước đây được định nghĩa trong bảng những khách hàng như thế nào. Bạn cũng sẽ khám phá những bảng Orders, Order Details, và Products

Những khóa chính

Điển hình, mỗi bảng trong một cơ sở dữ liệu có một hoặc nhiều cột là duy nhất để xác định mỗi hàng trong bảng. Cột này được biết như khóa chính cho bảng. Một khóa chính có thể bao gồm nhiều cột. Trong trường hợp này, đó là một khóa được biết như một khóa tổ hợp.

Ghi nhớ : giá trị cho khóa chính trong mỗi hàng của một bảng phải là duy nhất (không trùng với bất cứ giá trị khóa chính của hàng nào khác).

Trong trường hợp của bảng những khách hàng, khóa chính là cột CustomerID. Biểu tượng chìa khóa bên trái của cột CustomerID trong Hình 2.11 chỉ định cột này là khóa chính cho bảng những khách hàng. Tương tự, khóa chính cho bảng Orders là OrderID. Khóa chính cho bảng Order Details được tổ hợp từ hai cột: OrderID và ProductID. Khóa chính cho bảng Products (những sản phẩm) là ProductID.

Mối quan hệ Bảng và những khóa ngoại

Những đường nối những bảng trong Hình 2.11, trình bày trước đó, cho thấy những mối quan hệ giữa những bảng. Dấu vô cực (∞) ở cuối của mỗi đường chỉ định một mối quan hệ một- nhiều giữa hai bảng có nghĩa là một hàng trong một bảng có thể liên quan đến một hoặc nhiều hàng trong bảng khác.

Chẳng hạn, bảng khách hàng (Customers) có một mối quan hệ một- nhiều với bảng Orders (đơn đặt) . Mỗi khách hàng có thể đặt nhiều đơn đặt. Tương tự, mối quan hệ một- nhiều giữa những đơn đặt (orders) và bảng những chi tiết đơn đặt (Order Details) có nghĩa là mỗi đơn đặt có thể bao gồm nhiều chi tiết đơn đặt (bạn có thể hiểu một chi tiết đơn đặt như một hàng trong một đơn liệt kê những món hàng , với mỗi hàng tham chiếu tới một sản phẩm riêng biệt được đặt mua). Cuối cùng, mối quan hệ một- nhiều giữa bảng những sản phẩm (Products) và những chi tiết đơn đặt (Order Details) có nghĩa là mỗi sản phẩm có thể xuất hiện trong nhiều chi tiết đơn đặt (Order Details).

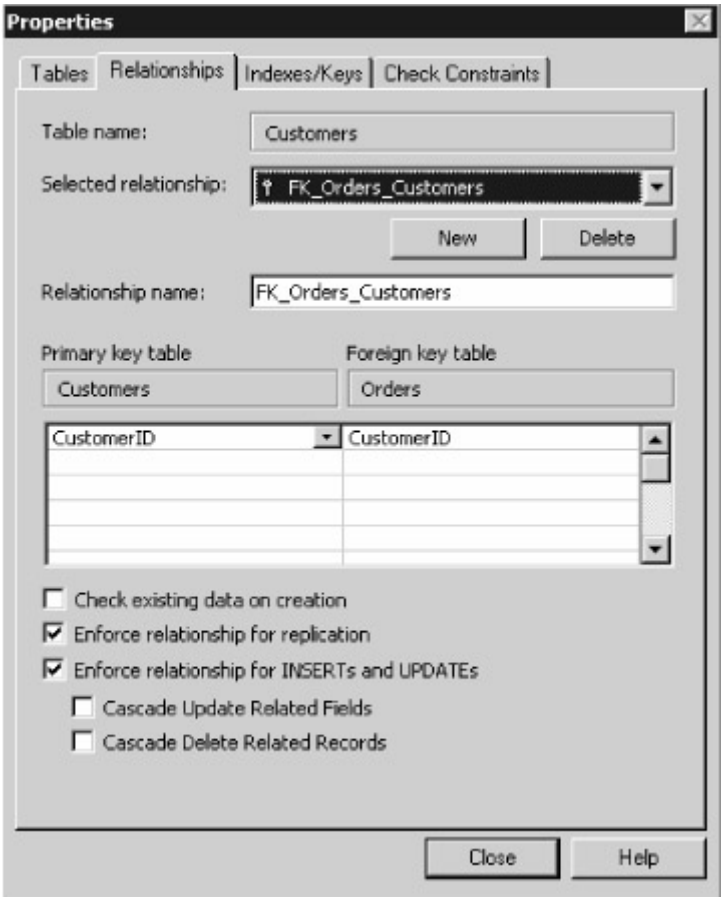
Những mối quan hệ một- nhiều được mô hình hóa sử dụng những khóa ngoại. Chẳng hạn, bảng những đơn đặt có một cột tên là CustomerID. Cột này liên quan đến cột CustomerID trong bảng những khách hàng thông qua một khóa ngoại. Điều này có nghĩa là mọi hàng trong bảng những đơn đặt phải có một hàng tương ứng trong bảng những khách hàng với một giá trị tương tự như cột CustomerID. Chẳng hạn, nếu một hàng trong bảng những đơn đặt (Orders) có một CustomerID là ALFKI, thì cũng phải có một hàng trong bảng những khách hàng (Customers) với một CustomerID là ALFKI. Do mối quan hệ giữa những khách hàng và những đon đặt là một- nhiều, điều này có nghĩa là có thể có nhiều hàng trong những đơn đặt với cột CustomerID giống như vậy. Dựa trên khái niệm này, Bạn có thể hiểu khóa ngoại như một con trỏ từ bảng những đơn đặt đến bảng những khách hàng.

Thường thường, bảng chứa khóa ngoại được hiểu như bảng con, và bảng chứa cột được tham chiếu bởi khóa ngoại được hiể như bảng cha. Chẳng hạn, bảng những đơn đặt là bảng con, và bảng những khách hàng là bảng cha. Những mối quan hệ khóa ngoại thường được hiểu như những mối quan hệ cha con.

Ghi chú:

Thuật ngữ *relational* (có quan hệ) từ *relational database* (cơ sở dữ liệu quan hệ) đến từ thực tế là những bảng có thể liên quan lẫn nhau thông qua những khóa ngoại.

Bạn có thể quản lý những mối quan hệ cho một bảng với *Enterprise Manager* (trình quản lý doanh nghiệp) bằng cách chọn *Table* từ nút *Tables* , kích nút chuột phải, và chọn *Design Table* (Thiết kế bảng). Rồi bạn kích nút *Manage Relationships* (Quản lý những mối quan hệ) trên thanh công cụ của table designer (cửa sổ thiết kế bảng). Chẳng hạn, Hình 2.12 cho thấy mối quan hệ giữa bảng những khách hàng và những đơn đặt.



Hình 2.12: Mối quan hệ giữa bảng Customers và Orders

Những bảng Customers và Orders có liên quan với nhau thông qua cột CustomerID. Cột CustomerID trong bảng những đơn đặt là khóa ngoại. Mối quan hệ giữa hai bảng được gán tên là "FK_Orders_Customers" .

Những giá trị Null

Những cơ sở dữ liệu cũng phải cung cấp những khả năng xử lý những giá trị chưa được gán vào, hay nói cách khác là chưa được biết. Những giá trị chưa được biết được gọi là những giá trị Null (null values), và một cột được định nghĩa là: cho phép hay không cho phép những giá trị null. Khi một cột được cho phép có những giá trị null, thì cột này được gán giá trị là null; ngược lại nó được định nghĩa là not-null. Một cột not-null trong một hàng phải luôn có một giá trị trong đó. Nếu bạn thử thêm một hàng nhưng không cung cấp một giá trị tới một cột đã được định nghĩa là not-null, thì cơ sở dữ liệu sẽ hiện ra một thông báo lỗi và sự thêm hàng mới của bạn thất bại.

Những chỉ số (Indexs)

Khi tìm kiếm một đề tài riêng biệt trong một cuốn sách, bạn có thể duyệt qua toàn bộ cuốn sách để tìm kiếm đề tài của các bạn, hay có thể sử dụng chỉ mục của sách để tìm trực tiếp vị trí chính xác của đề tài. Một chỉ số cho một bảng cơ sở dữ liệu cũng tương tự như khái niệm chỉ mục của sách, chỉ có điều những chỉ số cơ sở dữ liệu được dùng để tìm những hàng riêng biệt trong một bảng. Downside (mặt tiềm ẩn?) của những chỉ số là khi một hàng được thêm vào bảng, cần thiết một thời gian bổ xung để cập nhật chỉ số cho hàng mới.

Nói chung, bạn chỉ cần phải tạo ra một chỉ số trên một cột khi bạn thấy là bạn đang truy xuất một số ít hàng từ một bảng chứa nhiều hàng. Một kinh nghiệm tốt là một chỉ số hữu ích khi bạn mong đợi bất kỳ câu truy vấn đơn nào để truy xuất 10 phần trăm hoặc ít hơn so với tổng số hàng trong một bảng. điều này có nghĩa là cột thích hợp cho một chỉ số cần phải được dùng để lưu trữ một phạm vi rộng những giá trị. Một ứng cử viên tốt cho sự chỉ số hóa là một cột chứa một con số duy nhất xác định cho mỗi bản ghi (hàng), trong khi một ứng cử

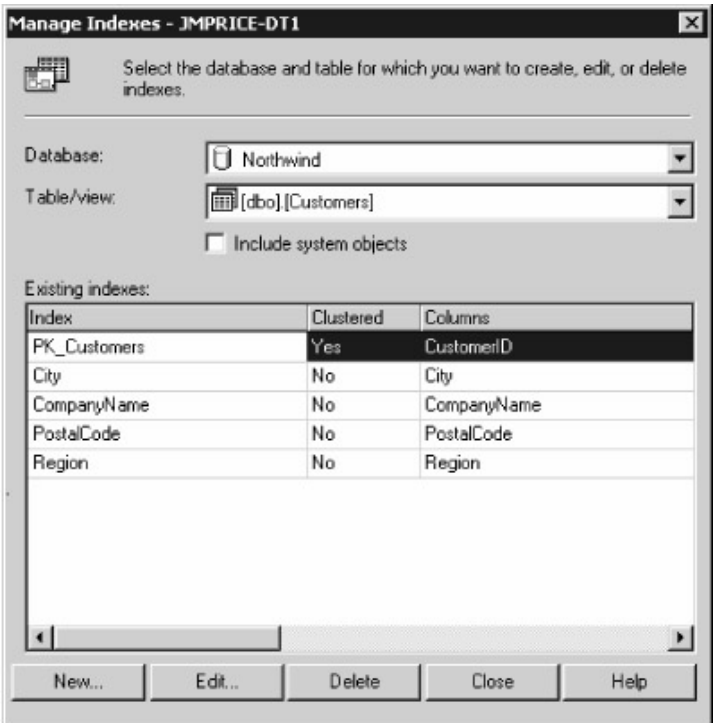
viên kém cho sự chỉ số hóa là một cột chỉ chứa một phạm vi nhỏ những mã số như 1, 2, 3, hay 4. Sự xuy xét này ứng dụng cho tất cả các kiểu cơ sở dữ liệu, không phải chỉ riêng cho những con số.

Ghi chú:

SQL Server tự động tạo ra một chỉ số cho cột khóa chính của một bảng.

Bình thường, DBA chịu trách nhiệm tạo ra những chỉ số, nhưng là một người phát triển ứng dụng, bạn chắc chắn biết nhiều về ứng dụng của bạn hơn DBA và sẽ có khả năng chọn ra những cột nào là những ứng cử viên tốt cho sự chỉ số hóa.

Bạn có thể quản lý những chỉ số cho một bảng với *Enterprise Manager* bằng cách chọn bảng từ (node) nút Tables, kích nút phải chuột, và chọn *All Tasks* *Manage Indexes*. Chẳng hạn, Hình 2.13 cho thấy những chỉ số cho bảng những khách hàng. Bạn cũng có thể quản lý những chỉ số từ *table designer* bởi kích nút *Manage Indexes/Keys.*



Hình 2.13: những chỉ số cho bảng những khách hàng

Bảng Customers có năm chỉ số: mỗi cái trên mỗi cột CustomerID, City, CompanyName, PostalCode, và Region columns.

Bạn sẽ học cách thêm một chỉ số vào một bảng như thế nào trong mục " Tạo một chỉ số " sau.

Những kiểu Cột

Mỗi cột trong một bảng có một kiểu cơ sở dữ liệu cụ thể. Kiểu này tương tự như kiểu một biến trong C#, ngoại trừ một kiểu cơ sở dữ liệu ứng dụng vào kiểu của giá trị mà bạn có thể lưu trữ trong một cột của bảng. Bảng 2.3 liệt kê những kiểu trong cơ sở dữ liệu SQL Server.

Bảng 2.3: những kiểu trong cơ sở dữ liệu SQL Server

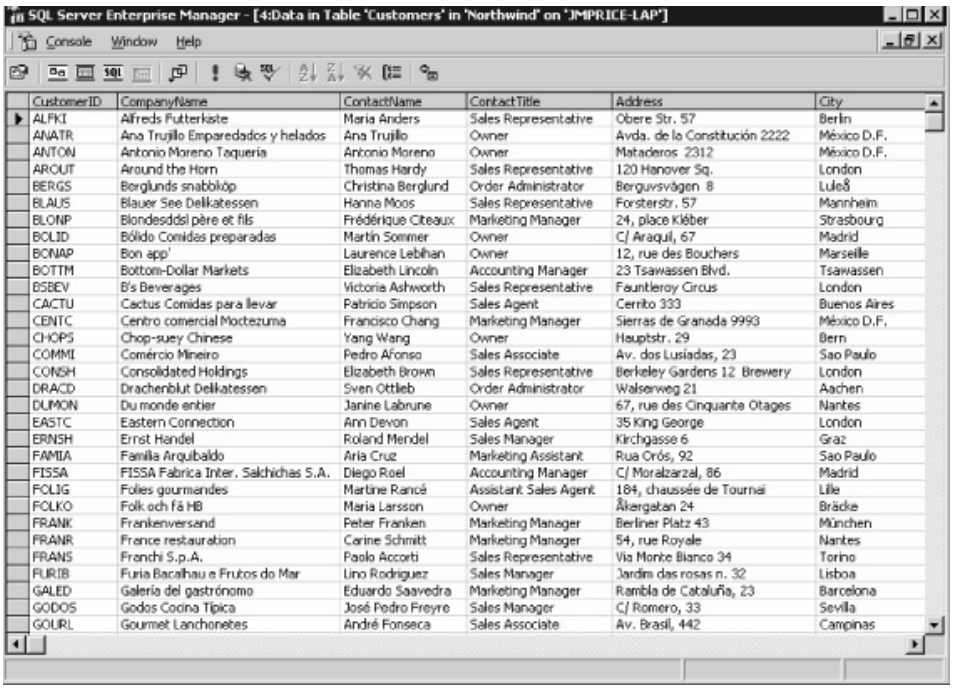
Kiểu dữ liệu	Mô tả
bigint	Giá trị số nguyên từ -2^{63} (-9,223,372,036,854,775,808) to 2^{63}-1 (9,223,372,036,854,775,807).
int	Giá trị số nguyên từ -2^{31} (-2,147,483,648) to 2^{31}-1 (2,147,483,647).
smallint	Giá trị số nguyên từ 2^{15} (-32,768) to 2^{15}-1 (32,767).
tinyint	Giá trị số nguyên từ 0 to 255.
bit	Giá trị số nguyên hoặc 1 hoặc 0.

decimal	giá trị số thập phân có độ chính xác cố định từ -10^{38} to 10^{38}.
numeric	Tương tự decimal.
money	Giá trị dữ liệu tiền tệ từ -2^{63} (-922,337,203,685,477.5808) to $2^{63}-1$ (922,337,203,685,477.5807), với độ chính xác tới một phần mười nghìn của một đơn vị tiền tệ.
smallmoney	Giá trị dữ liệu tiền tệ từ -214,748.3648 to 214,748.3647, với độ chính xác tới một phần mười nghìn của một đơn vị tiền tệ.
float	Giá trị kiểu dấu chấm động từ -1.79E+308 to 1.79E+308.
real	Giá trị kiểu dấu chấm động từ -3.40E + 38 to 3.40E + 38.
datetime	Giá trị ngày và giờ từ tháng giêng 1, 1753, đến tháng mười hai 31, 9999, với độ chính xác tới 3% của giây (3.33 milli giây).
smalldatetime	Giá trị ngày tháng và thời gian từ Tháng giêng 1, 1900 đến Tháng sáu 6, 2079 với độ chính xác tới một phút.
char	Những ký tự không phải Unicode có chiều dài cố định với chiều dài cực đại 8.000 ký tự.
varchar	Những ký tự không phải Unicode với chiều dài cực đại 8.000 ký tự.
text	Những ký tự không phải Unicode với chiều dài cực đại 2^{31} (2,147,483,647).
nchar	Những ký tự Unicode chiều dài cố định với một chiều dài cực đại 4.000 ký tự.
nvarchar	Những ký tự Unicode chiều dài biến đổi ,chiều dài cực đại 4.000 ký tự.
ntext	Những ký tự Unicode chiều dài thay đổi, chiều dài cực đại 2^{30} (1,073,741,823) ký tự.
binary	Dữ liệu nhị phân chiều dài cố định, chiều dài cực đại 8.000 bytes.
varbinary	Dữ liệu nhị phân chiều dài thay đổi, chiều dài cực đại 8.000 bytes.
image	Dữ liệu nhị phân chiều dài thay đổi, chiều dài cực đại 2^{31} (2,147,483,647) bytes.
cursor	Sự Tham khảo tới một con trỏ, được gán tới những hàng.
sql_variant	Có thể lưu trữ những giá trị của nhiều kiểu dữ liệu SQL server ngoại trừ text, ntext, timestamp, và Sql_variant.
table	Lưu trữ một tập những hàng
timestamp	Số nhị phân duy nhất được cập nhật mỗi khi bạn sửa đổi một hàng. Bạn có thể chỉ định nghĩa một cột timestamp trong một bảng.
uniqueidentifier	Định danh toàn cục duy nhất (GUID).

Tốt, đầy đủ lý thuyết! Chúng ta hãy xem xét kỹ hơn ở những bảng khách hàng, những đơn đặt, những chi tiết đơn đặt, và những sản phẩm.

Bảng những khách hàng (Customers)

Bảng những khách hàng chứa những hàng mà lưu trữ những chi tiết của một công ty đã đặt những đơn đặt với Công ty Northwind. Hình 2.14 cho thấy một số những hàng và những cột được cất giữ trong bảng những khách hàng.



CustomerID	CompanyName	ContactName	ContactTitle	Address	City
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim
BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993	México D.F.
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 23	Sao Paulo
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkeley Gardens 12 Brewery	London
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	Walserweg 21	Aachen
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinquante Otages	Nantes
EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George	London
ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6	Graz
FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assistant	Rua Orós, 92	Sao Paulo
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accounting Manager	C/ Moralzarzal, 86	Madrid
FOLIG	Folies gourmandes	Martine Rancé	Assistant Sales Agent	184, chaussée de Tournai	Lille
FOLKO	Folk och fä HB	Maria Larsson	Owner	Åkergatan 24	Bräcke
FRANK	Frankenversand	Peter Franken	Marketing Manager	Berliner Platz 43	München
FRANR	France restauration	Carine Schmitt	Marketing Manager	54, rue Royale	Nantes
FRANS	Franchi S.p.A.	Paolo Accorti	Sales Representative	Via Monte Bianco 34	Torino
FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Sales Manager	Jardim das rosas n. 32	Lisboa
GALED	Galería del gastrónomo	Eduardo Saavedra	Marketing Manager	Rambla de Cataluña, 23	Barcelona
GODOS	Godos Cocina Típica	José Pedro Freyre	Sales Manager	C/ Romero, 33	Sevilla
GOURL	Gourmet Lanchonetes	André Fonseca	Sales Associate	Av. Brasil, 442	Campinas

Hình 2.14: những hàng từ bảng những khách hàng

Như bạn có thể thấy, hàng đầu tiên được trình bày cho một khách hàng có tên Alfreds Futterkiste; tên này được cất giữ trong cột CompanyName của bảng những khách hàng.

CustomerID cho hàng đầu tiên là ALFKI, và như bạn có thể thấy, CustomerID là duy nhất cho mỗi hàng. Như được đề cập trước đó, khóa chính cho bảng những khách hàng là cột CustomerID. Nếu bạn thử thêm một hàng mới với một khóa chính đã được dùng trong trong một hàng trong bảng này, thì cơ sở dữ liệu sẽ loại bỏ hàng mới của bạn. Chẳng hạn, nếu bạn thử thêm một hàng vào những bảng khách hàng với một CompanyID là ALFKI, thì hàng này bị loại bỏ bởi vì ALFKI đã được sử dụng bởi hàng đầu tiên trong bảng.

> **Mẹo nhỏ:** Bạn có thể tự xem những hàng từ một bảng bởi chọn bảng trong Enterprise Manager, kích chuột phải, và chọn Open Table □ Return all rows. Bạn sẽ học nhiều hơn về xem những hàng từ những bảng sau trong mục " xây dựng những câu truy vấn ".

Định nghĩa của Bảng những khách hàng

Bảng 2.4 cho thấy định nghĩa cho những cột của Bảng những khách hàng. Bảng này cho thấy tên cột, kiểu cơ sở dữ liệu, chiều dài, và liệu cột cho phép những giá trị null hay không.

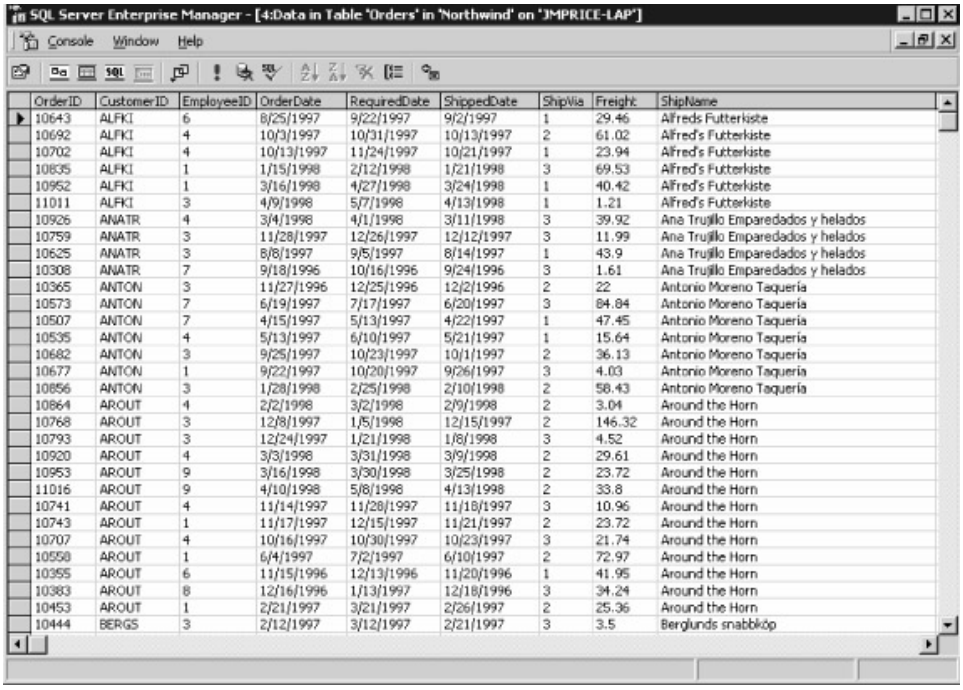
Bảng 2.4: Định nghĩa cho những cột của Bảng những khách hàng

Tên cột	Kiểu cơ sở dữ liệu	Độ dài	Cho phép giá trị Null ?
CustomerID	nchar	5	No
CompanyName	nvarchar	40	No
ContactName	nvarchar	30	Yes
ContactTitle	nvarchar	30	Yes
Address	nvarchar	60	Yes
City	nvarchar	15	Yes
Region	nvarchar	15	Yes
PostalCode	nvarchar	10	Yes
Country	nvarchar	15	Yes
Phone	nvarchar	24	Yes
Fax	nvarchar	24	Yes

Trong mục kế tiếp, bạn sẽ học về bảng Đơn đặt (Orders) .

Bảng những đơn đặt

Bảng Orders chứa những hàng cất giữ những đơn đặt được đặt bởi khách hàng. Hình 2.15 cho thấy một số hàng và những cột được cất giữ trong bảng Orders.



OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
10643	ALFKI	6	8/25/1997	9/22/1997	9/2/1997	1	29.46	Alfreds Futterkiste
10692	ALFKI	4	10/3/1997	10/31/1997	10/13/1997	2	61.02	Alfred's Futterkiste
10702	ALFKI	4	10/13/1997	11/24/1997	10/21/1997	1	23.94	Alfred's Futterkiste
10835	ALFKI	1	1/15/1998	2/12/1998	1/21/1998	3	69.53	Alfred's Futterkiste
10952	ALFKI	1	3/16/1998	4/27/1998	3/24/1998	1	40.42	Alfred's Futterkiste
11011	ALFKI	3	4/9/1998	5/7/1998	4/13/1998	1	1.21	Alfred's Futterkiste
10926	ANATR	4	3/4/1998	4/1/1998	3/11/1998	3	39.92	Ana Trujillo Emparedados y helados
10759	ANATR	3	11/28/1997	12/26/1997	12/12/1997	3	11.99	Ana Trujillo Emparedados y helados
10625	ANATR	3	8/8/1997	9/5/1997	8/14/1997	1	43.9	Ana Trujillo Emparedados y helados
10308	ANATR	7	9/18/1996	10/16/1996	9/24/1996	3	1.61	Ana Trujillo Emparedados y helados
10365	ANTON	3	11/27/1996	12/25/1996	12/2/1996	2	22	Antonio Moreno Taquería
10573	ANTON	7	6/19/1997	7/17/1997	6/20/1997	3	84.84	Antonio Moreno Taquería
10507	ANTON	7	4/15/1997	5/13/1997	4/22/1997	1	47.45	Antonio Moreno Taquería
10535	ANTON	4	5/13/1997	6/10/1997	5/21/1997	1	15.64	Antonio Moreno Taquería
10682	ANTON	3	9/25/1997	10/23/1997	10/1/1997	2	36.13	Antonio Moreno Taquería
10677	ANTON	1	9/22/1997	10/20/1997	9/26/1997	3	4.03	Antonio Moreno Taquería
10856	ANTON	3	1/28/1998	2/25/1998	2/10/1998	2	58.43	Antonio Moreno Taquería
10864	AROUT	4	2/2/1998	3/2/1998	2/9/1998	2	3.04	Around the Horn
10768	AROUT	3	12/8/1997	1/5/1998	12/15/1997	2	146.32	Around the Horn
10793	AROUT	3	12/24/1997	1/21/1998	1/8/1998	3	4.52	Around the Horn
10920	AROUT	4	3/3/1998	3/31/1998	3/9/1998	2	29.61	Around the Horn
10953	AROUT	9	3/16/1998	3/30/1998	3/25/1998	2	23.72	Around the Horn
11016	AROUT	9	4/10/1998	5/8/1998	4/13/1998	2	33.8	Around the Horn
10741	AROUT	4	11/14/1997	11/28/1997	11/18/1997	3	10.96	Around the Horn
10743	AROUT	1	11/17/1997	12/15/1997	11/21/1997	2	23.72	Around the Horn
10707	AROUT	4	10/16/1997	10/30/1997	10/23/1997	3	21.74	Around the Horn
10558	AROUT	1	6/4/1997	7/2/1997	6/10/1997	2	72.97	Around the Horn
10355	AROUT	6	11/15/1996	12/13/1996	11/20/1996	1	41.95	Around the Horn
10383	AROUT	8	12/16/1996	1/13/1997	12/18/1996	3	34.24	Around the Horn
10453	AROUT	1	2/21/1997	3/21/1997	2/26/1997	2	25.36	Around the Horn
10444	BERGS	3	2/12/1997	3/12/1997	2/21/1997	3	3.5	Berglunds snabbköp

Hình 2.15: những hàng từ bảng Orders

Khóa chính cho bảng Orders là cột OrderID, có nghĩa là giá trị cho cột này phải là duy nhất cho mỗi hàng. Nếu bạn nhìn kỹ sáu hàng đầu tiên trong bảng Orders, bạn sẽ thấy cột CustomerID đều là ALFKI, giống như gái trị cột CustomerID của hàng đầu tiên trong bảng những khách hàng trình bày trước đó trong Hình 2.12.

Bây giờ bạn có thể hiểu thông tin liên hệ của những khóa ngoại như thế nào . Cột CustomerID của bảng Orders là một khóa ngoại nó tham chiếu cột CustomerID của bảng những khách hàng. Trong ví dụ này, bảng Orders là bảng con, và bảng những khách hàng là bảng cha . Bạn có thể hiểu khóa ngoại như một con trỏ từ bảng Orders đến bảng những khách hàng. Bảng 2.5 cho thấy định nghĩa cho những cột của bảng Orders.

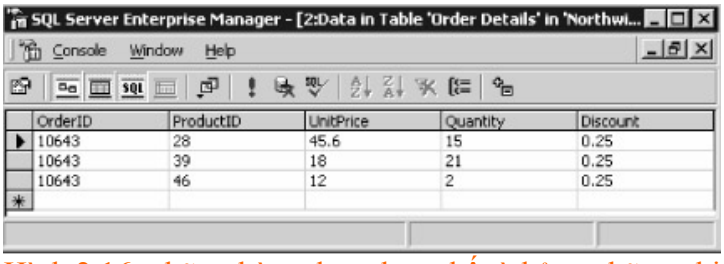
Bảng 2.5: Định nghĩa cho những cột của bảng Orders

Tên cột	Kiểu csdl	Chiều dài	Cho phép giá trị null ?
OrderID	int	4	No
CustomerID	nchar	5	Yes
EmployeeID	int	4	Yes
OrderDate	datetime	8	Yes
RequiredDate	datetime	8	Yes
ShippedDate	datetime	8	Yes
ShipVia	int	4	Yes
Freight	money	8	Yes
ShipName	nvarchar	40	Yes
ShipAddress	nvarchar	60	Yes
ShipCity	nvarchar	15	Yes
ShipRegion	nvarchar	15	Yes

ShipPostalCode	nvarchar	10	Yes
ShipCountry	nvarchar	15	Yes

Bảng chi tiết đơn đặt

Bảng những chi tiết đơn đặt chứa những hàng lưu giữ những chi tiết của mỗi đơn đặt. Trong hình 2.16, tôi đã hạn chế những hàng được truy xuất từ bảng những chi tiết đơn đặt với những hàng có cột OrderID bằng 10643 (nó giống như cột OrderID của hàng đầu tiên trong bảng Orders được trình bày trước đó trong Hình 2.15).



Hình 2.16: những hàng được hạn chế từ bảng những chi tiết đơn đặt

Khóa chính cho bảng những chi tiết đơn đặt là tổ hợp của những cột OrderID và CustomerID, có nghĩa là sự tổ hợp của những giá trị trong hai cột này phải là duy nhất cho mỗi hàng.

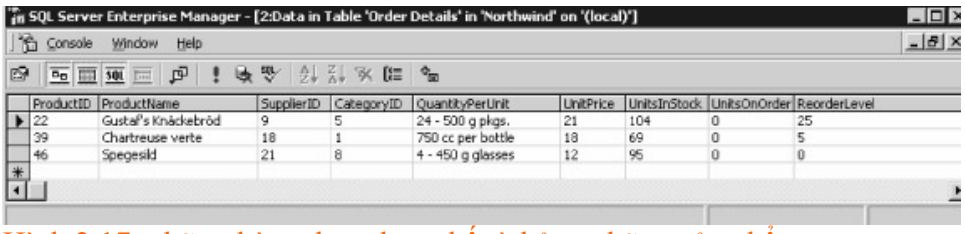
Đồng thời, cột OrderID của bảng Order Details (những chi tiết đơn đặt) là một khóa ngoại nó tham chiếu cột OrderID của bảng Ordres. Cột ProductID của bảng Order Details (chi tiết đơn đặt) là một khóa ngoại nó tham chiếu đến cột ProductID của bảng những sản phẩm. Bảng 2.6 cho thấy định nghĩa cho những cột của Bảng những chi tiết đơn đặt . Bạn sẽ học về những bảng sản phẩm tiếp theo.

Bảng 2.6: Định nghĩa cho những cột của Bảng những chi tiết đơn đặt

Tên cột	Kiểu cơ sở dữ liệu	Chiều dài	Cho phép giá trị Null ?
OrderID	int	4	Yes
ProductID	int	4	Yes
UnitPrice	money	8	Yes
Quantity	smallint	2	Yes
Discount	real	4	Yes

Bảng những sản phẩm

Bảng những sản phẩm chứa những hàng lưu trữ những chi tiết của mỗi sản phẩm được bán bởi Công ty Northwind. trong Hình 2.17, Tôi có hạn chế những hàng được truy xuất từ bảng những sản phẩm với cột ProductID bằng 22, 39, và 46 (chúng tương tự như những giá trị cho cột ProductID cho những hàng trong bảng Order Details được chỉ ra trước đó trong Hình 2.16).



Hình 2.17: những hàng được hạn chế từ bảng những sản phẩm

Khóa chính cho bảng những sản phẩm là cột ProductID. Cột CategoryID của bảng những sản phẩm là một khóa ngoại ,nó tham chiếu tới cột CategoryID của bảng Categories . Bảng Categories chứa nhiều hạng loại của những sản phẩm.

Cột SupplierID của bảng những sản phẩm là một khóa ngoại nó tham chiếu tới cột SupplierID của bảng Suppliers (nhà cung ứng) Bảng Suppliers chứa những nhà cung ứng (Suppliers) những sản phẩm cho Công ty Northwind. Bảng 2.7 cho thấy định nghĩa cho những cột của Bảng những sản phẩm .

Bảng 2.7: Định nghĩa cho những cột của Bảng những sản phẩm

Tên cột	Kiểu dữ liệu	Chiều dài	Cho phép giá trị Null ?
ProductID	int	4	No
ProductName	nvarchar	40	No
SupplierID	int	4	Yes
CategoryID	int	4	Yes
QuantityPerUnit	nvarchar	20	Yes
UnitPrice	money	8	Yes
UnitsInStock	smallint	2	Yes
UnitsOnOrder	smallint	2	Yes
ReorderLevel	smallint	2	Yes
Discontinued	bit	1	Yes

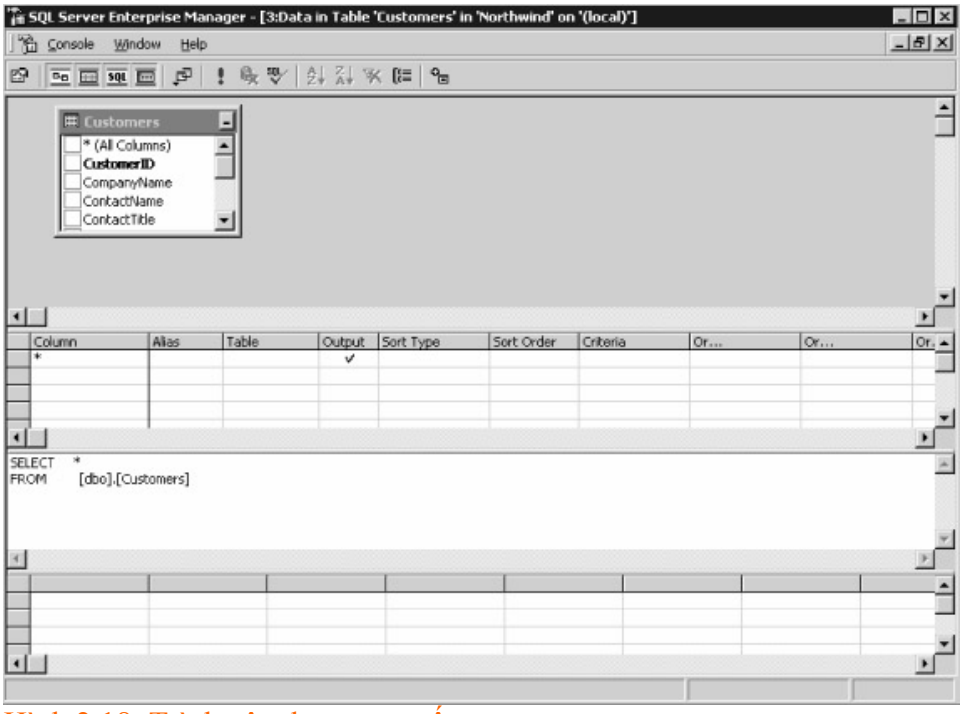
Trong mục kế tiếp bạn sẽ học xây dựng những câu truy vấn để truiy xuất những hàng từ những bảng như thế nào.

Xây dựng những câu truy vấn sử dụng Enterprise Manager

Bạn có thể xây dựng những câu truy vấn của mình để khảo sát những hàng trong những bảng sử dụng Enterprise Manager (Trình quản trị doanh nghiệp). Trong mục này, bạn sẽ học xây dựng và chạy một câu truy vấn để xem những đơn đặt của khách hàng với một CustomerID là ALFKI, Cùng với những chi tiết đơn đặt và những sản phẩm cho đơn đặt với một OrderID là 10643. Đặc biệt, bạn sẽ lựa chọn những cột sau đây:

- Những cột CustomerID và cột CompanyName từ bảng những khách hàng
- Những cột OrderID và cột OrderDate từ bảng Orders
- Những cột ProductID và Quantity (số lượng) từ bảng những chi tiết đơn đặt

Bắt đầu xây dựng câu truy vấn, chọn bảng những khách hàng trong Enterprise Manager từ nút Tables của thư mục Databases thuộc cơ sở dữ liệu Northwind. Kích chuột phải và chọn Open Table ➢ Query. việc này mở query builder (trình xây dựng truy vấn), như trong Hình 2.18.



Hình 2.18: Trình xây dựng truy vấn

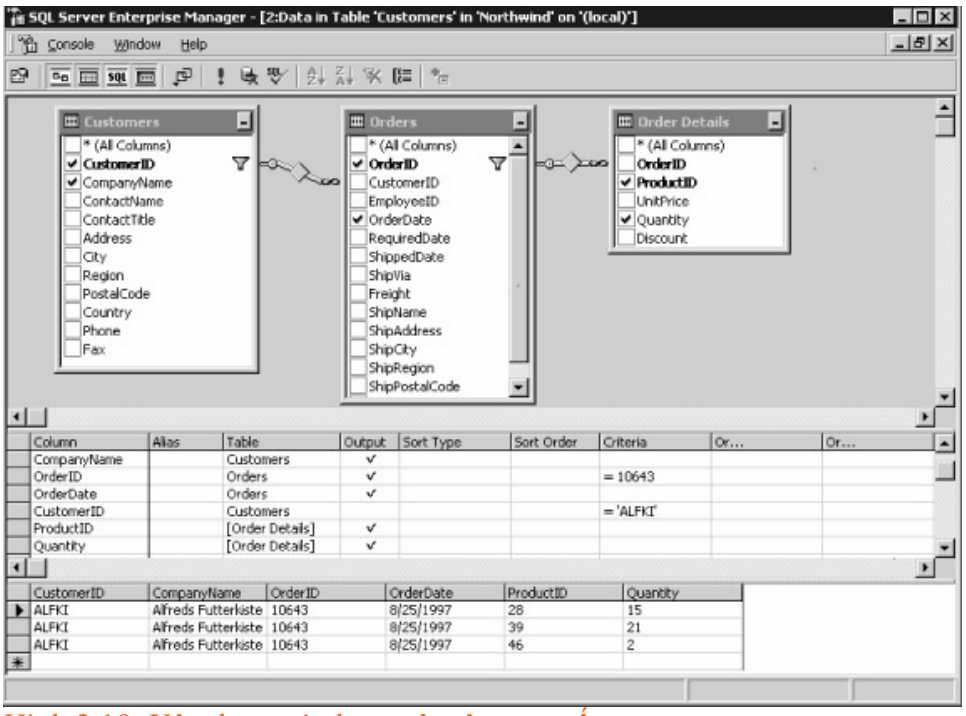
Khung bên trên được gọi là khung sơ đồ (Diagram Pane), và nó cho thấy những bảng được sử dụng trong câu truy vấn . Như bạn có thể thấy, bảng Customers được thoạt tiên hiện ra trong khung Diagram Pane (sơ đồ). Khung bên dưới được gọi là Grid Pane (khung lưới), và nó trình bày những chi tiết cho những cột và những hàng được truy xuất từ những bảng. Thoạt tiên, tất cả các hàng đều sẽ được truy xuất từ bảng những khách hàng , như biểu thị bởi dấu sao (*) trong Grid Pane (khung lưới). bên dưới khung lưới là khung SQL, và nó trình bày câu lệnh SQL cho câu truy vấn.

> **Ghi chú**: SQL là một ngôn ngữ trên nền văn bản để truy cập một cơ sở dữ liệu, và bạn sẽ học mọi thứ về SQL trong chương kế tiếp. Bây giờ, bạn có thể kích nút SQL trên thanh công cụ để ẩn khung SQL- trừ phi bạn muốn xem câu lệnh SQL được xây dựng bởi query builder (trình xây dựng truy vấn).

Bên dưới khung SQL là khung chứa những kết quả, nó trình bày bất kỳ hàng nào truy xuất bởi câu truy vấn. khung này thoạt đầu trống rỗng bởi vì chưa có câu truy vấn được chạy. Sử dụng những bước sau đây để xây dựng câu truy vấn:

1. Loại bỏ dấu sao (*) từ Grid Pane (khung lưới) bởi kích chuột phải trên hộp ở bên trái của hàng chứa dấu sao và chọn Delete (Xóa). Việc này dừng tất cả việc truy xuất các cột từ bảng những khách hàng.
2. Kích nút phải chuột trong Diagram Pane (khung sơ đồ), và chọn Add Table. Thêm những bảng Orders và Order Details như thế bạn có thể truy vấn những bảng này. Bạn cũng có thể kích nút Add table trên thanh công cụ để thêm những bảng. Bạn chú ý sau khi thêm những bảng, chúng xuất hiện trong Diagram Pane (khung sơ đồ) cùng với những đường nối những bảng cha và con thông qua khóa ngoại. Chẳng hạn, những bảng Customers và bảng Orders được nối thông qua cột CustomerID. Tương tự, những bảng Orders và Order Details được nối thông qua cột OrderID.
3. Chọn những cột CustomerID và CompanyName từ bảng Customers bởi chọn những hộp kiểm ở bên trái của column names (những tên cột) trong Diagram Pane (khung sơ đồ).
4. Chọn những cột OrderID và OrderDate từ bảng Orders.
5. Chọn những cột ProductID và Quantity (số lượng) từ bảng Order Details (nhưng chi tiết đơn đặt).
6. Trong Grid Pane (khung lưới), gán tiêu chuẩn (Criteria) cho cột CustomerID là =' ALFKI'. Việc này gây ra câu truy vấn chỉ truy xuất những hàng từ bảng những khách hàng có cột CustomerID là ALFKI.
7. Trong khung Lưới, gán tiêu chuẩn cho OrderID là = 10643. việc này gây ra câu truy vấn chỉ truy xuất những hàng từ bảng Orders có cột OrderID bằng 10643.
8. Chạy câu truy vấn bởi kích nút Run trên thanh công cụ.

Hình 2.19 trình bày kết quả cuối cùng của sự xây dựng và chạy câu truy vấn.



Hình 2.19: Xây dựng và chạy một câu truy vấn

Như bạn sẽ thấy trong chương kế tiếp, bạn cũng có thể xây dựng và chạy những câu truy vấn sử dụng Visual Studio .NET. Trong mục kế tiếp, bạn sẽ học cách tạo ra một bảng như thế nào sử dụng Enterprise Manager.

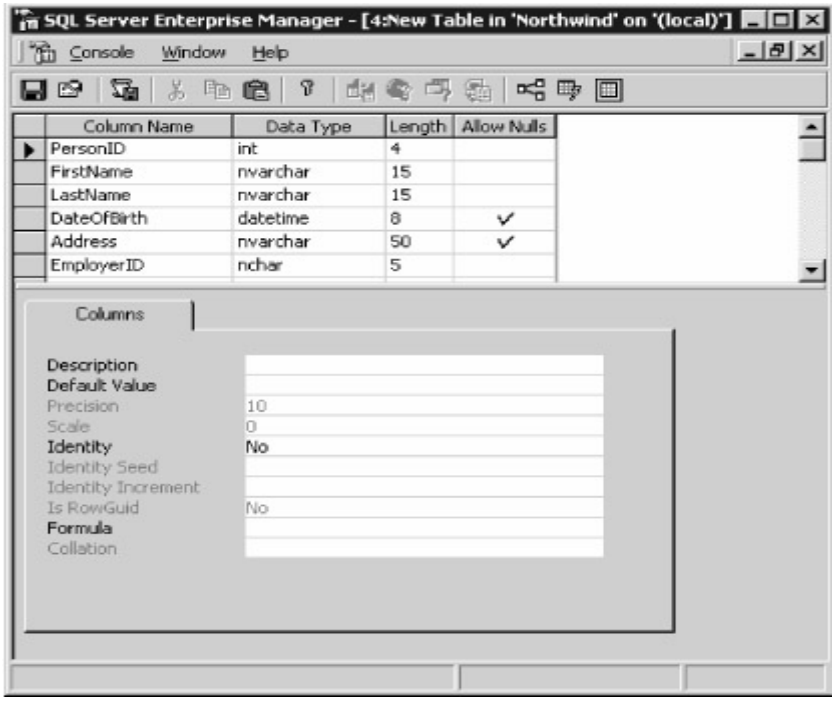
Tạo ra một Bảng

Bạn có thể sử dụng Enterprise Manager để thêm một bảng vào một cơ sở dữ liệu. Trong mục này, bạn sẽ thêm một bảng vào cơ sở dữ liệu Northwind để lưu những chi tiết của một người. Bảng này sẽ được gọi là Persons, và sẽ chứa những cột được trình bày trong Bảng 2.8.

Bảng 2.8: Định nghĩa cho những cột của Bảng Persons

COLUMN NAME	DATABASE TYPE	LENGTH	ALLOWS NULL VALUES?
PersonID	int	4	No
FirstName	nvarchar	15	No
LastName	nvarchar	15	No
DateOfBirth	datetime	8	Yes
Address	nvarchar	50	Yes
EmployerID	nchar	5	No

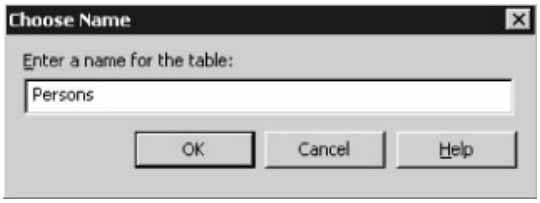
Để tạo ra một bảng trong cơ sở dữ liệu Northwind, bạn chọn nút Tables của cơ sở dữ liệu Northwind trong Enterprise Manager và chọn Action □ New Table. Rồi bạn sẽ thấy table designer (trình thiết kế bảng). Thêm những cột như trình bày trong bảng 2.8 thuộc bảng được trình bày trong Hình 2.20.



Hình 2.20: thêm một bảng mới

Ghi nhớ: chiều dài một số những kiểu dữ liệu là cố định. Chẳng hạn, kiểu int luôn luôn sử dụng 4 bytes không gian lưu trữ, vì vậy bạn không thể thay đổi chiều dài một cột int khác 4. Tương tự, kiểu datetime luôn luôn sử dụng 8 bytes không gian lưu trữ. Bạn có thể thay đổi chiều dài của những cột nchar và nvarchar bởi vì những kiểu này được thiết kế để lưu trữ dữ liệu có chiều dài thay đổi.

Kích nút Save trên thanh công cụ để lưu bảng. Trong hộp thoại Choose Name , Nhập Persons như một tên, và kích OK để lưu bảng của bạn, như trình bày trong Hình 2.21.



Hình 2.21: Nhập tên của bảng

Ghi nhớ: Một khi bạn đã lưu bảng của bạn, bạn có thể trở lại table designer vào bất cứ lúc nào bởi chọn bảng này trong node "Tables "của Enterprise Manager, nhấp phải bảng này, và chọn Design Table.

Trong phần còn lại của chương này, bạn sẽ học cách:

- Lấy thông tin bổ sung về những cột trong một bảng sử dụng thẻ Columns.
- Thiết đặt khóa chính của một bảng.
- Thiết đặt những quyền hạn cho phép sự truy cập tới nội dung của một bảng.
- Tạo ra một mối quan hệ giữa những bảng.
- Tạo ra một chỉ số để cho phép truy cập tới thông tin nhanh hơn trong một bảng.
- Tạo ra một sự ràng buộc để hạn chế những giá trị có thể được lưu trữ trong một cột.

Thẻ Columns

Trong vùng bên dưới lưới, bạn chú ý một thẻ có tên Columns. Thẻ Columns này chứa thông tin bổ sung về cột đang được chọn trong lưới, và Hình 2.20, được trình bày trước đó, trình bày thông tin trên cột PersonID. Khi

bạn thay đổi cột được chọn của bạn, thông tin trong thẻ Columns sẽ thay đổi.

Bạn có thể nhập một sự mô tả tùy chọn cho một cột trong trường Description của thẻ Columns. Trường Default Value (giá trị mặc định) cho phép bạn cung cấp một giá trị ban đầu khi một hàng mới được thêm vào bảng; tất nhiên bạn có thể cung cấp giá trị bạn muốn cho một cột và sẽ ghi đè lên giá trị mặc định.

Trường Precision (độ chính xác) trình bày số lượng chữ số cực đại có thể được dùng để lưu trữ một con số, bao gồm cả những chữ số được cất giữ ở bên phải của dấu phẩy ở số thập phân. Trường Scale cho thấy số lượng chữ số cực đại ở bên phải của một dấu phẩy ở số thập phân. Chẳng hạn, precision và Scale của một cột int là 10 và 0, có nghĩa là cột int này có thể cất giữ tới 10 chữ số, và không có chữ số nào ở bên phải dấu phẩy ở số thập phân- nó không có chữ số nào ở bên phải vì một int là một số nguyên. Precision và Scale cho một cột tiền tệ là 19 và 4, có nghĩa là cột tiền tệ này có thể cất giữ tới 19 chữ số, trong đó có 4 chữ số ở bên phải của dấu phẩy ở phần thập phân.

Trường Identity (mã nhận dạng) cho phép bạn chỉ định liệu có phải SQL Server cần phải tự động gán một giá trị tới một trường. Nếu bạn đặt trường Identity là true, thì bạn cũng có thể chỉ định những giá trị cho những trường Identity Seed (mã nhận dạng khởi đầu) và Identity Increment (mã nhận dạng tăng dần) . Bạn sử dụng trường Identity Seed (mã nhận dạng khởi đầu) để gán giá trị ban đầu cho cột, và bạn sử dụng trường Identity Increment (mã nhận dạng tăng dần) để chỉ rõ sự tăng dần giá trị. Chẳng hạn, nếu bạn đặt Identity Seed là 1 và Identity Increment tới 1, thì giá trị cho cột đầu tiên là 1, tiếp theo là 2, vân vân. Cột ProductID của những bảng Products là một ví dụ của một cột sử dụng một identity để đặt giá trị cho nó.

Trường IsRowGuid chỉ rõ liệu có phải một cột uniqueidentifier là một định danh toàn cục duy nhất được biết đến như một GUID.

Mẹo nhỏ : SQL Server không tự động cung cấp một giá trị cho một GUID. Nếu bạn muốn SQL Server sinh ra một GUID, Bạn có thể sử dụng hàm NEWID() của SQL Server. Hàm NEWID() luôn luôn trả lại một giá trị khác nhau. Và bạn có thể sử dụng giá trị được gửi ra từ hàm này như giá trị mặc định cho cột uniqueidentifier (cột khóa chính) của bạn. Chẳng hạn, bạn đặt trường Default Value (giá trị mặc định) tới [NEWID()]. Bạn sẽ học nhiều hơn về những hàm SQL Servers trong chương kế tiếp.

Trường Formula (Công thức) cho phép bạn thiết đặt một công thức được dùng để gán một giá trị tới một cột.

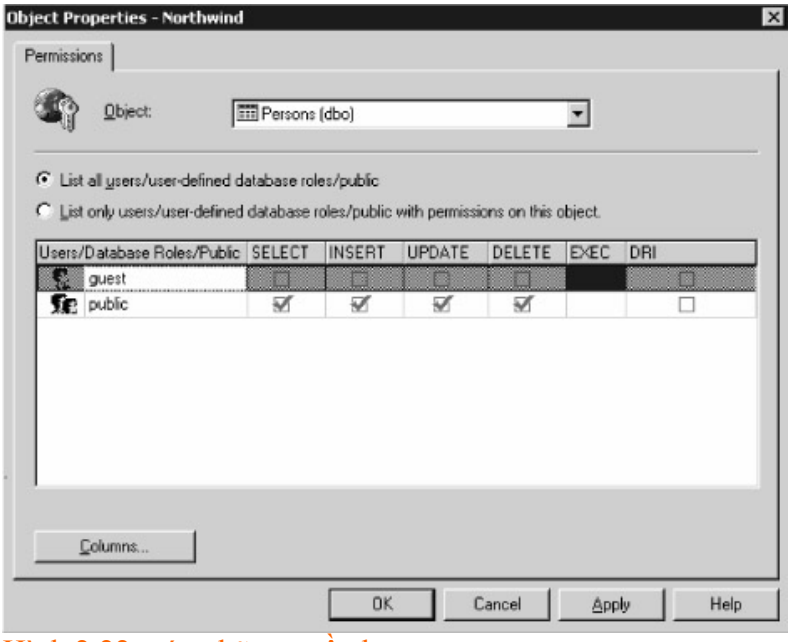
Trường Collation chỉ rõ những quy tắc được dùng để sắp xếp và so sánh những ký tự. Bạn có lẽ cần thiết đặt trường này khi làm việc với những ngoại ngữ. Những chi tiết hơn, tham khảo những sách tài liệu trực tuyến SQL Server.

Đặt khóa chính

Tiếp theo, bạn sẽ đặt khóa chính cho bảng Persons tới PersonID. Để làm điều này, kích vào hàng đầu tiên trong lưới đang chứa cột PersonID, và kích nút Set primary key trên thanh công cụ. Một khi bạn làm điều này, bạn sẽ thấy một biểu tượng chìa khóa nhỏ bên trái của PersonID.

Gán những quền hạn

Để đặt những quền hạn cho bảng của bạn, kích nút Show permissions trên thanh công cụ của table designer. Cấp phát quền hạn SELECT, INSERT, UPDATE, and DELETE tới vai trò public, như như trình bày trong Hình 2.22. Những quền hạn này cho phép những người sử dụng công cộng (public) truy xuất, thêm, sửa đổi, và loại bỏ những hàng từ bảng Persons.

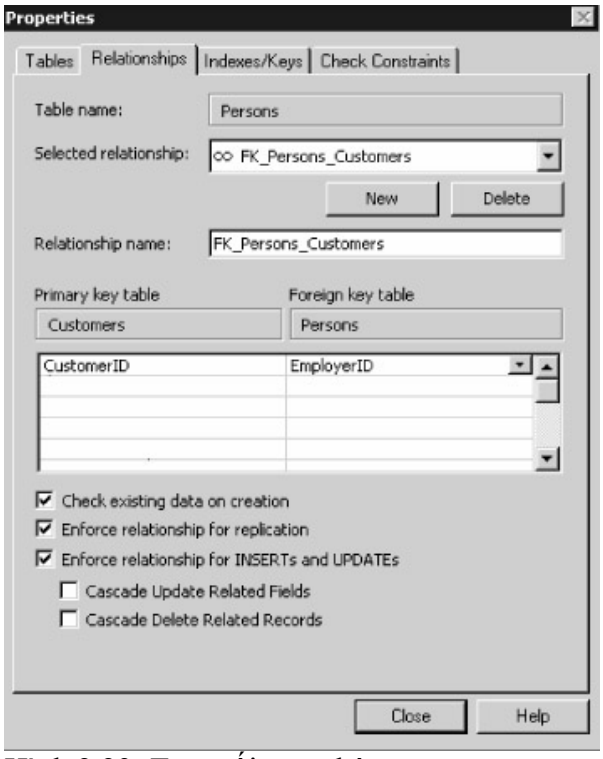


Hình 2.22: gán những quền hạn

Kích OK để tiếp tục.

Tạo mối quan hệ

Bạn sẽ tạo ra một mối quan hệ giữa bảng Persons của bạn và bảng Customers. Để xem màn hình những mối quan hệ, kích nút Manage Relationships (Quản lý những mối quan hệ) trên thanh công cụ của table designer (bộ thiết kế bảng). Kích New để bắt đầu tạo mối quan hệ. Chọn bảng Customers như bảng khóa chính và chọn cột CustomerID từ bảng này. Đảm bảo rằng Persons được chọn như bảng khóa ngoại, và chọn cột EmployerID từ bảng này. Hình 2.23 cho thấy điều này. Bạn chú ý là tên mối quan hệ được tự động gán tới FK_Persons_Customers.



Hình 2.23: Tạo mối quan hệ

Những hộp kiểm tại đáy Trang như sau:

Check existing data on creation (Kiểm tra dữ liệu hiện hữu trên sự tạo thành): Điều này áp dụng sự ràng buộc của bạn tới dữ liệu mà có lẽ đã tồn tại trong cơ sở dữ liệu khi bạn thêm mối quan hệ của bạn vào bảng

khóa ngoại .

Enforce relationship for replication (thi hành mối quan hệ cho bản sao): Bản sao cho phép bạn sao chép thông tin tới một cơ sở dữ liệu khác. Khi bạn cho phép thực thi mối quan hệ cho bản sao, sự ràng buộc của bạn được ứng dụng vào bảng khóa ngoại khi bảng này được sao chép tới một cơ sở dữ liệu khác nhau trong thời gian sao chép.

Enforce relationship for INSERTs and UPDATEs (kết buộc mối quan hệ cho những sự chèn và những sự cập nhật): Điều này áp dụng sự ràng buộc của bạn tới nhữnghàng được thêm vào, được sửa đổi, hay được loại bỏ từ bảng khóa ngoại .

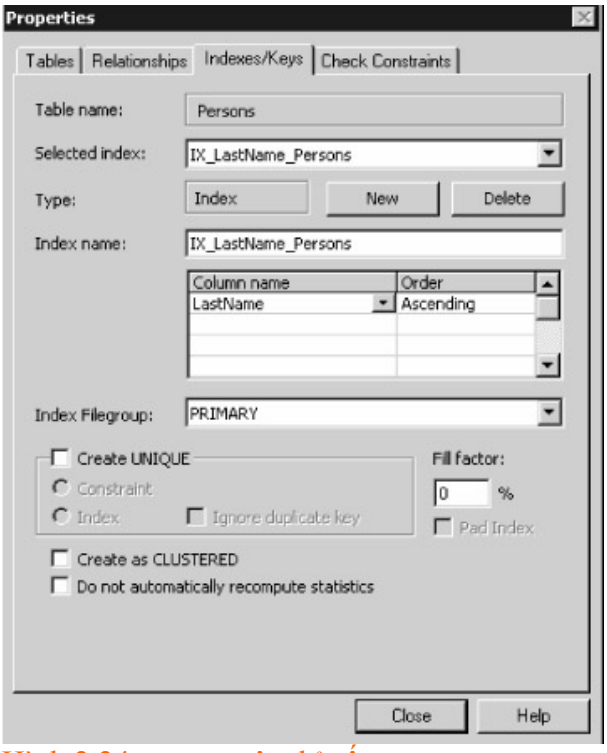
Cascade Update Related Fields (Cập nhật dây chuyền những trường quan hệ): Điều này gây cho SQL Server tự động cập nhật những giá trị khóa ngoại của mối quan hệ của bạn khi giá trị khóa chính được sửa đổi.

Cascade Delete Related Fields (Xóa dây chuyền những trường quan hệ) điều này gây cho SQL Server tự động loại bỏ những hàng từ bảng khóa ngoại bất cứ khi nào hàng được tham chiếu trong bảng khóa chính bị loại bỏ. Kích Close để tiếp tục.

Tạo ra một chỉ số

Một chỉ số cho phép cơ sở dữ liệu nhanh chóng định vị một hàng khi bạn yêu cầu lấy lại hàng này dựa vào một giá trị cột đặc biệt. Trong mục này, bạn sẽ tạo ra một chỉ số trên cột LastName của bảng Persons của bạn.

Để xem những chỉ số cho bảng Persons của bạn, kích nút Manage Indexes/Keys (Quản lý những chỉ số/ những khóa) trên thanh công cụ của (bộ thiết kế bảng). Kích new để bắt đầu tạo ra một chỉ số mới. Đặt tên của chỉ số là IX_LastName_Persons, chọn cột LastName và gán order là ascending (tăng dần). Hình 2.24 cho thấy điều này.



Hình 2.24: tạo ra một chỉ số

Bạn sẽ không thay đổi bất kỳ những trường và hộp kiểm nào khác khi tạo ra chỉ số của bạn, hãy để chúng y như những gì chúng có, dưới đây là ý nghĩa của những trường:

Index Filegroup: Chỉ số nhóm tập tin là filegroup mà trong đó bạn muốn cất giữ chỉ số của bạn. Một filegroup được tạo ra từ một hoặc nhiều file vật lý trên đĩa cứng của máy tính. SQL Server sử dụng filegroups để cất giữ

thông tin thực tế tạo thành một cơ sở dữ liệu.

Create UNIQUE : Tùy chọn Create UNIQUE (tạo mã duy nhất) cho phép bạn tạo ra một ràng buộc duy nhất hay chỉ số cho bảng cơ sở dữ liệu được lựa chọn. Bạn chỉ định bạn đang tạo ra một sự ràng buộc duy nhất hay chỉ số bởi chọn nút rađiô Constraint hoặc Index .
Ignore duplicate key (bỏ qua khóa trùng lặp): nếu bạn tạo ra một chỉ số duy nhất, bạn có thể chọn tùy chọn này để lờ đi những giá trị khóa trùng lặp.

Fill factor (Hệ số lấp đầy): Trừ phi bạn là một người sử dụng SQL Server cao cấp, nếu không bạn cần phải để hệ số lấp đầy trong sự thiết đặt mặc định. Đơn vị nhỏ nhất của bộ lưu trữ trong một cơ sở dữ liệu SQL Server là một trang nhớ, nó có thể lưu trữ tới 8,096. bytes dữ liệu. Dữ liệu cho những bảng và những chỉ số được cất giữ trong những trang nhớ. Bạn có thể chỉ định dung lượng mỗi trang chỉ số bởi thiết đặt fill factor (hệ số lập đầy) . Chẳng hạn, nếu bạn gán hệ số lấp đầy tới 60 phần trăm, thì trang sẽ chứa tới 60 phần trăm dữ liệu và 40 phần trăm không gian trống . Số lượng của không gian trống trên một trang chỉ số là quan trọng vì khi một trang chỉ số được lấp đầy, SQL Server phải tách trang ra từng phần để làm nơi lưu trữ cho dữ liệu chỉ số mới. Bằng cách giảm bớt hệ số lấp đầy, nhờ đó, bạn có thể tăng sự thực thi của cơ sở dữ liệu của bạn bởi vì SQL Server sẽ không phải chia những trang ra thường xuyên. Tuy nhiên, việc giảm bớt hệ số lấp đầy, cũng gây cho chỉ số nắm giữ không gian đĩa cứng hơn bởi vì sẽ có nhiều không gian trống hơn trong mỗi trang. Nếu bạn không chỉ định một hệ số lấp đầy, thì hệ số lấp đầy mặc định của cơ sở dữ liệu được sử dụng.

Pad Index (Chỉ số đệm): trừ phi bạn là một người sử dụng SQL Server cao cấp, nếu không bạn không nên cho phép tùy chọn chỉ số đệm (Pad Index) . Nếu bạn chỉ định một hệ số lấp đầy (fill factor) nhiều hơn 0 phần trăm và bạn đang tạo ra một chỉ số duy nhất (unique index) thông qua tùy chọn (Create UNIQUE) , thì bạn có thể cho phép tùy chọn Pad index (Chỉ số đệm). Điều này thông báo Máy chủ phục vụ SQL nó sẽ sử dụng số phần trăm như bạn chỉ rõ trong trường fill factor (hệ số lấp đầy) làm khoảng trống bỏ ngõ trên mỗi nhánh node của cây nhị phân tạo ra chỉ số. Bạn có thể học nhiều hơn tùy chọn này trong những sách tài liệu trực tuyến SQL Server .

Create as CLUSTERED : (tạo nhóm) Bạn sử dụng tùy chọn Create as CLUSTERED để chỉ định chỉ số của bạn được tạo nhóm. Một chỉ số được tạo nhóm là một chỉ số chứa những hàng của bảng thực tế, thay vì những con trỏ tới những hàng của bảng . Những chỉ số được tạo nhóm cho phép truy xuất nhanh chóng hơn những hàng, nhưng yêu cầu nhiều thời gian hơn khi chèn những hàng mới. Bạn có thể học nhiều hơn tùy chọn này trong những sách tài liệu trực tuyến SQL Server.

Do not automatically recompute statistics (Không tự động tính toán lại thống kê): Bạn điển hình không nên sử dụng tùy chọn này vì nó có thể làm giảm yếu sự thực thi. Khi bạn tạo ra một chỉ số, SQL Server tự động lưu trữ thông tin thống kê về sự phân phối của những giá trị trong những cột được chỉ số hóa của bạn. SQL Server sử dụng thống kê này để đánh giá chi phí của việc sử dụng chỉ số cho một câu truy vấn. Bạn sử dụng tùy chọn "Do not automatically recompute statistics" để chỉ định SQL Server cần phải sử dụng thống kê đã tạo ra trước đó , có nghĩa là thống kê không nhất thiết được cập và cho phép giảm thiểu sự thực thi. Bạn có thể học nhiều hơn tùy chọn này trong những sách tài liệu trực tuyến SQL Server.

Kích Close để tiếp tục.

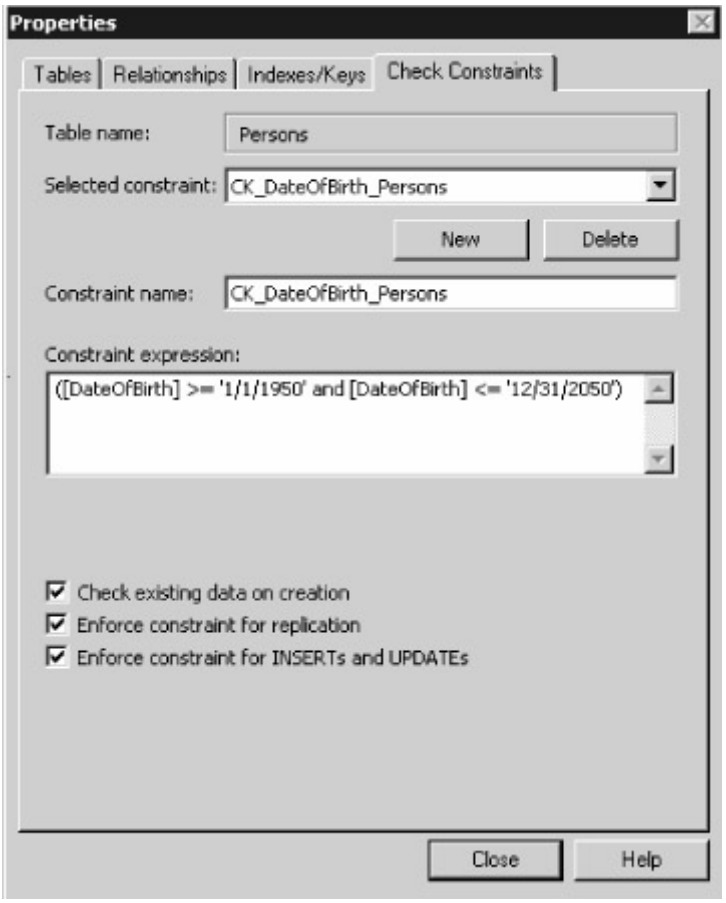
Tạo ra một sự ràng buộc

Một sự ràng buộc cho phép bạn định nghĩa một giới hạn trên giá trị có thể được cất giữ trong một cột. Trong mục này, bạn sẽ tạo ra một sự ràng buộc trên cột DateOfBirth của bảng Persons của bạn. Sự ràng buộc này sẽ bảo đảm rằng bạn có thể đặt chỉ những ngày tháng giữa tháng giêng 1, 1950, và tháng mười hai 31, 2050, trong cột DateOfBirth.

Để xem những sự ràng buộc cho bảng Persons của bạn, kích nút Manage Constraints (Quản lý những ràng buộc) trên thanh công cụ của table designer. Kích New để bắt đầu tạo ra một sự ràng buộc mới. Đặt biểu thức ràng buộc như sau:

```
([DateOfBirth] >= '1/1/1950' and [DateOfBirth] <= '12/31/2050')
```

Gán tên ràng buộc là CK_DateOfBirth_Persons. Hình 2.25 cho thấy điều này.



Hình 2.25: tạo ra một sự ràng buộc

Bạn sẽ không thay đổi bất kỳ hộp kiểm tra nào khi tạo ra sự ràng buộc của các bạn, hảy để nguyên như những gì chúng có, dưới đây là ý nghĩa những trường :

Check existing data on creation (Kiểm tra dữ liệu hiện hữu trên sự tạo thành): Sử dụng tùy chọn này để bảo đảm rằng dữ liệu đang tồn tại trong bảng thỏa mãn sự ràng buộc của bạn.

Enforce constraint for replication (kết buộc sự ràng buộc cho bản sao): sử dụng tùy chọn này để kết buộc sự ràng buộc của bạn khi bảng của bạn được sao chép tới cơ sở dữ liệu khác thông qua bản sao.

Enforce constraint for INSERTs and UPDATEs (kết buộc sự ràng buộc cho những sự chèn và cập nhật) Sử dụng tùy chọn này để kết buộc sự ràng buộc của bạn khi những hàng được thêm vào hay được sửa đổi trong bảng.

Kích Close để tiếp tục. lưu bảng và đóng table designer.

Tóm lược

Trong chương này, bạn đã học nền tảng của những cơ sở dữ liệu và SQL Server. Một cơ sở dữ liệu là một tập hợp có tổ chức của thông tin. Một cơ sở dữ liệu quan hệ là một tập hợp của thông tin liên quan mà đã được tổ chức vào trong những cấu trúc được biết đến như những bảng. Mỗi bảng chứa đựng những hàng và thêm nữa được tổ chức vào trong những cột.

Hệ thống được dùng để quản lý thông tin trong cơ sở dữ liệu được biết như hệ quản trị cơ sở dữ liệu (database management system). Trong trường hợp của một cơ sở dữ liệu điện tử trong một máy tính, hệ quản trị cơ sở dữ liệu là phần mềm quản lý thông tin trong những bộ nhớ và những tập tin của máy tính. Một ví dụ về phần mềm này là SQL Server. Bạn đã thấy cách khởi chạy một cơ sở dữ liệu SQL Server như thế nào, và làm sao để sử dụng Enterprise Manager để khám phá cơ sở dữ liệu Northwind.

Điển hình, mỗi bảng trong một cơ sở dữ liệu có một hoặc nhiều cột mà xác định mỗi hàng duy nhất trong bảng. Cột này được biết như khóa chính cho bảng. Những bảng có thể liên quan lẫn nhau thông qua những khóa ngoại. Bạn đã học cách truy vấn những hàng trong một bảng như thế nào và cách tạo ra một bảng mới sử dụng Enterprise Manager.

Trong chương kế tiếp, bạn sẽ học sử dụng ngôn ngữ truy vấn có cấu trúc như thế nào.

Chương 3: Giới thiệu về ngôn ngữ truy vấn có cấu trúc (SQL)

Tổng quan

Trong chương này, bạn sẽ học cách sử dụng ngôn ngữ truy vấn có cấu trúc (SQL) để truy nhập một cơ sở dữ liệu , sử dụng hai công cụ để nhập và chạy những câu truy vấn : Query Analyzer (bộ phân tích truy vấn) và Visual Studio .NET. Chương này chỉ cho bạn cách sử dụng cơ sở dữ liệu Northwind SQL Serever, chứa thông tin về một công ty tưởng tượng Northwind . Bạn sẽ thấy cách sử dụng SQL để tương tác với cơ sở dữ liệu Northwind như thế nào để truy xuất và thao tác thông tin và tạo ra, sửa đổi, xóa những bảng trong cơ sở dữ liệu này.

Đặc trưng trong chương này:

- Sử dụng SQL
- Việc truy nhập một cơ sở dữ liệu sử dụng Visual Studio .NET

SỬ DỤNG SQL :

SQL (sự xuy luận rỏ ràng) là một ngôn ngử tiêu chuẩn về truy cập những cơ sở dữ liệu quan hệ. Như bạn thấy trong chương này, SQL rất dễ học và dễ sử dụng. với SQL , bạn có thể cho cơ sở dữ liệu biết dữ liệu mà bạn cần truy cập, và phần mềm quảnlý cơ sở dữ liệu sẽ tính tóan chính xác làm thế nào để lấy dữ liệu đó.

Có rất nhiều kiểu phát biểu SQL, nhưng những kiểu thông dụng nhất của phát biểu SQL như dưới đây:

- Những phát biểu của Ngôn ngữ thao tác dữ liệu (Data Manipulation Language_DML):
- Những phát biểu của Ngôn ngữ định nghĩa dữ liệu (Data Definition Language (DDL))

Những phát biểu DML hco phép bạn truy xuất, thêm, sửa đổi, xóa những dòng chứa trong cơ sở dữ liệu. Những phát biểu DDL cho phép bạn tạo những cấu trúc cơ sở dữ liệu như bảng v v.

Trước khi bạn học những điều cơ bản về những phát biểu DML, bạn cần biết cách nhập và chạy những phát biểu SQL. Sử dụng những công cụ phân tích truy vấn (Query Analyzer tool).

Chú ý: như bạn sẽ thấy trong mục " Truy cập một cơ sở dữ liệu sử dụng Visual Studio .NET" ở nh74ng chương sau, bạn cũng có thể sử dụng Visual Studio .NET để tạo những phát biểu SQL. Visual Studio .NET còn cho phép bạn những phát biểu SQL một cách trực quan, hay nhập vào bằng tay.

SỬ DỤNG BỘ PHÂN TÍCH TRUY VẤN (Query Analyzer):

Bạn sử dụng Query Analyzer để nhập và chạy những phát biểu SQL :
chọn start ⏩ Microsoft SQL Server ⏩ Query Analyzer.
Trong những mục dưới đây , bạn sẽ học cách kết nối với một instance SQLserver, nhập vào và chạy một phát biểu SQL, lưu một phát biểu SQL, và tải lên một SQL Statement.

KẾT NỐI VỚI MỘT INSTANCE SQL SEVER:

Khi bạn khởi động Query Analyzer , một hộp thoại "Connect to SQL Server " sẽ hiển thị đầu tiên, trong hộp comboBox với nhãn SQL Server , bạn nhập vào tên của SQL Server muốn kết nối. bạn có thể click vào mũi tên để kiển thị danh sách sổ xuống và chon tên của đối tượng muốn kết nối, hoặc nút ba chấm …để hiển thị danh sách SQL server đang chạy trên mạng.

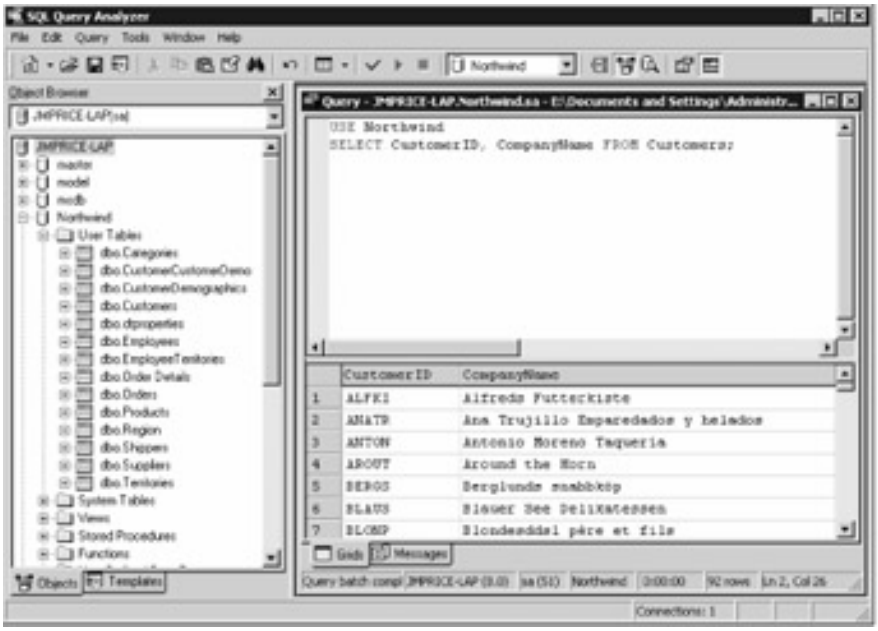
Hình 3.1: Kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL

Nếu bạn chọn nút radio _Windows authentication, thì SQL Server sẽ sử dụng những thông tin người dùng trong Windows 2000/NT để hiệu lực hóa yêu cầu kết nối với SQL Server của bạn. nếu bạn chọn nút riodio_ SQL Server authentication , bạn sẽ phải nhập mã tài khoản và password.

Trong hình trên ,bạn đã nhập " localhost" vào trường SQL Server, tương ứng với thể hiện SQL Server được cài đặt trên máy cục bộ. tôi cũng đã chọn nút radio_SQL Server authentication , và nhập **sa** vào trường **Login Name**, và **sa** vào trường **Password** (đây là Password mà tôi sử dụng khi cài đặt SQL Server). Những chi tiết này rồi lại được sử dụng để kết nối với SQL Server. Nếu bạn có một instance của SQL Server chạy trên máy cục bộ hopặc trên mạng , bạn có thể nhập vào những chi tiết liên quan và click OK để kết nối với SQL Server. Bây giờ bạn đã hiểu cách kết nối với cơ sở dữ liệu, hảy tiếp tục xem xét làm thế nào để nhập và chạy một phát biểu SQL.

NHẬP VÀ CHẠY MỘT PHÁT BIỂU SQL:

Một khi bạn đã kết nối được với SQL Server sử dụng trình phân tích Query (Query Analyzer),bạn có thể sử dụng Object Browser để xem những thành phần của cơ sở dữ liệu, và bạn nhập vào và chạy phát biểu SQL sử dụng một cửa sổ Query (Query window) .hình 3.2 trình bày của sổ Object Browser và một thí dụ Query windows , cùng với những kết quả từ việc truy xuất các cột CustomerID và Companyname từ bảng Customers.



Hình 3.2 : xem những mục cơ sở dữ liệu sử dụng Object Brower và thực thi phát biểu SELECT sử dụng Query Windows

Như bạn tháy ở hình 3.2 , bạn nhập vào những phát biểu SQL vào phần trên của cửa sổ query, và kết quả truy xuất từ cơ sở dữ liệu hiển thị trong phần dưới của cửa sổ query. Bạn chỉ định cơ sở dữ liệu để truy xuất với phát biểu USE , và bạn truy xuất những hàng từ cơ sở dữ liệu sử dụng phát biểu SELECT.

Mẹo nhỏ: bạn cũng có thể chỉ định một cơ sở dữ liệu để đăng nhập sử dụng danh sách sổ xuống trên Toolbar.

Nếu bạn muốn đi theo cùng với thí dụ này, bắt đầu nhập phát biểu USE sau đây vào trong cửa sổ Query của bạn:

```
USE Northwind
```

Phát biểu USE này chỉ định rằng bạn muốn sử dụng cơ sở dữ liệu Northwind. Tiếp theo trên hàng khác, nhập vào phát biểu SELECT bhư dưới đây:

```
SELECT CustomerID, CompanyName FROM Customers;
```

Phát biểu SELECT này cho biết bạn muốn truy xuất các cột CustomerID và Companyname từ bảng Customers. Chú ý: SELECT và FROM là những từ khóa SQL. Mặc dù SQL không phải là “case sensitive”,tôi sử dụng chữ hoa khi chỉ định những từ khóa SQLvà kiểu lạc đà khi chỉ định những cột và tên bảng. bạn có thể sử dụng dấu chấm phẩy(;) để kết thúc một phát biểu SQL,

Bạn có thể chạy một phát biểu SQL đã nhập trong cửa sổ Query theo năm cách:

- chọn Execute từ query menue
- click nút Execute Query trên toolbar
- nhấn F5 trên bàn phím
- nhấn Ctrl+E trên bàn phím
- nhấn Alt+X trên bàn phím

Một khi bạn đã chạy phát biểu SQL, phát biểu của bạn được gởi đến cơ sở dữ liệu để thự thi. Cơ sở dữ liệu sẽ chạy phát biểu của bạn và gởi kết quả về . và kết quả này hiển thị ở phần dưới của cửa sổ query windows

LƯU VÀ TẢI MỘT PHÁT BIỂU SQL:

Bạn có thể lưu một phát biểu SQL đã nhập vào Query Analyzer thành một file Text . sau đ1o bạn ccó thể tải lên và chạy phát biểu SQL đã lưu thành file Text đó. Bạn có thể lưu phát biểu SQL bằng cách:

- chọn Save hoặc Save As từ File menu
- click nút Save Query/Result trên toolbar
- nhấn Ctrl+S trên bàn phím

Khi bạn thự hiện một cách nào đó trên đây, trình Query Analyzer mở ra một hộp thoại “Open Query File”. Chúng ta hãy nói bạn mở CustomerSelect.sql.một khi bạn đã mở một File Query, bạn có thể sử dụng một trong những kỹ thuật được mô tả trước đây để chạy nó.

TÌM HIỂU NHỮNG PHÁT BIỂU CỦA NGÔN NGỮ SỬ LÝ DỮ LIỆU (DML):

Như được đề cập trước, những phát biểu DML cho phép bạn truy xuất, thêm, sửa đổi, và xóa những hàng được lưu trữ trong các bảng của cơ sở dữ liệu. có bốn kiểu phát biểu DML:

SELECT truy xuất những hàng từ một hay nhiều bảng.
INSERT thêm một hay nhiều hàng mới vào một bảng.
UPDATE sửa đổi một hoạc nhiều hàng trong một bảng.

DELETE xóa một hoặc nhi6èu hàng khỏi một bảng.

Bạn sẽ học cách sử dụng bốn phát biểu này trong những phần tiếp sau.

TRUY XUẤT NHỮNG HÀNG TỪ MỘT BẢNG ĐƠN:

Bạn sử dụng phát biểu SELECT để truy xuất những hàng từ những bảng. phát biểu SELECT có rất nhiều khuôn Mẫu, và phiên bản đơn giản nhất cho phép bạn chỉ định một danh sách những tên cột và bảng. thí dụ , phát biểu SELECT dưới đây truy xuất những cột CustomerID, CompanyName, ContactName, and Address từ bảng Customers:

```
SELECT CustomerID, CompernyName, ContactName, Address
FROM Customers;
```

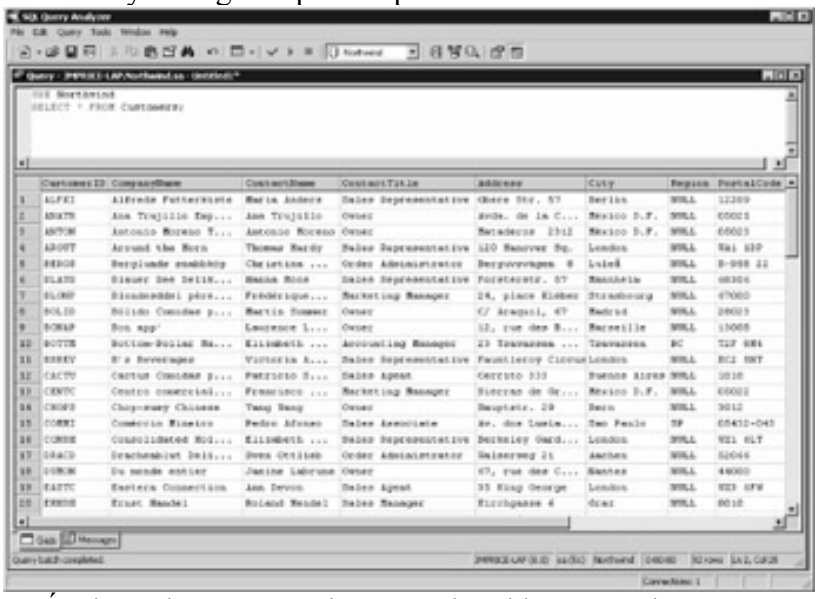
Những cột cần truy cập được ghi sau từ khóa SELECT , và bảng chỉ định được ghisau từ khóa FROM.
Nếu bạn muốn truy xuất tất cả các cột từ bảng, bạn chỉ định ký tự (*) ngay sau từ khóa SELECT.

Mẹo nhỏ: để tránh truy xuất những thông tin không cần thiết , thay vì sử dụng (*) ,chỉ nên liệt kê những cột mà bạn thực sự cần.

Thí dụ: phát biểu SELECT dưới đây sẽ truy xuất tất cả các cột từ bảng Customers sử dụng ký tự (*).

```
SELECT *
FROM Customers;
```

Hình 3.3 trình bày những kết quả của phát biểu SELECT.



Để truy xuất những hàng từ một bảng có chứa khỏang cách trongtên của nó, bạn đặt tên của bảng này trong một cặp ngoặc vuông. Thí dụ , phát biểu SELECT dưới đây truy xuất những hàng từ bảng Order Details:

```
SELECT *
FROM [Order Details];
```

Chú thích: bạn cũng có thể sử dụng cặp ngoặc vuông với tên cột có khỏang trắng.

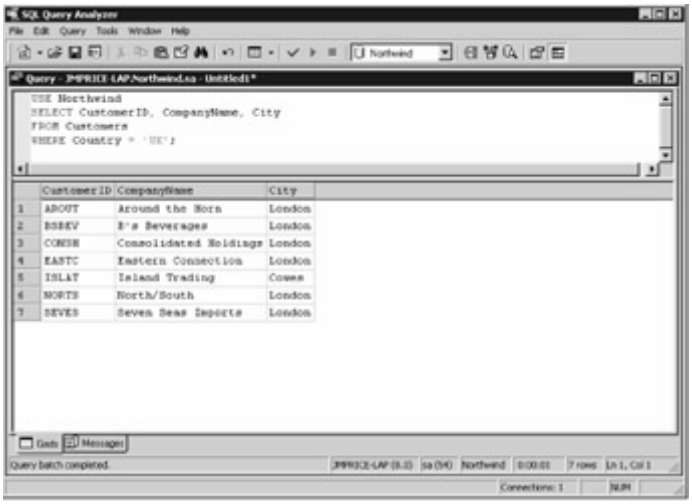
HẠN CHẾ NHỮNG HÀNG TRUY XUẤT:

Bạn sử dụng mệnh đề WHERE để hạn chế những hàng được truy xuất bởi phát biểu SELECT. Thí dụ: phát biểu SELECT dưới đây sử dụng mệnh đề WHERE để hạn chế những hàng được truy xuất từ bảng Customer Với những hàng nào có cột Column là ‘UK’:

```
SELECT CustomerID, CompanyName, City
```

```
FROM Customers
WHERE Country = 'UK';
```

Hình 3.4 : trình bày những kết quả của biểu thức SELECT



Phát biểu SELECT tiếp theo sử dụng mệnh đề WHERE để hạn chế những hàng truy xuất từ bảng Product quy cho dòng có ProductID = 10:

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID = 10;
```

Tóan tử bằng không phải là tóan tử duy nhất mà bạn có thể sử dụng trong mệnh đề where .
Bảng 3.1 trình bày những tóan tử tóan học mà bạn có thể sử dụng.

Table 3.1: SQL MATHEMATICAL OPERATORS

OPERATOR	DESCRIPTION
=	Equal
<> or !=	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

Phát biểu SELECT sau đây trình bày toán tử nhỏ hơn hoặc bằng (<=) để truy xuất những hàng từ bảng Products Mà giá trị cột ProductID nhỏ hơn hay bằng 10:

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID <= 10;
```

phát biểu SELECT tiếp theo sử dụng tóan tử không bằng (<>) để truy cập những hàng tử bảng Product có giá trị cột ProductID không bằng mười :

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID <> 10;
```

THỰC THI SỰ THÍCH ỨNG VỚI MẪU:

Bạn sử dụng toán tử LIKE trong mênh đề WHERE để thực hiện sự thích ứng với mẫu. bạn chỉ một hay nhiều Ký tự đại diện để sử dụng trong chuỗi thực hiện sự thích ứng mẫu của bạn.

Bảng 3.2: những ký tự đại diện

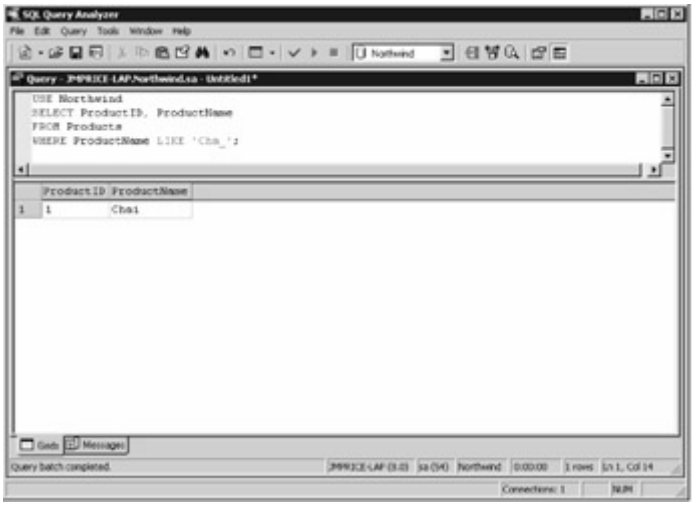
Table 3.2: NHỮNG KÍ TỰ ĐẠI DIÊN

KÍ TỰ	MÔ TẢ
_	Đại diện cho bất cứ một kí tự nào. Thí dụ, J_y phù hợp với Joy và Jay.
%	Phù hợp với mọi số lượng kí tự. thí dụ, %wind phù hợp với Northwind và Southwind; %fire% phù hợp với starfire, firestarter,và fireman.
[ ]	Phù hợp với bất cứ kí tự nào trong ngoặc vuông. Thí dụ, [sm]ay phù hợp với say và may.
[^]	Phù hợp với bất cứ kí tự nào không nằm trong ngoặc vuông. Thí dụ, [^a] phù hợp với bất cứ kí tự nào ngòai a.
[-]	Phù hợp với một dãy những kí tự. thí dụ, [a-c]bc phù hợp với abc, bbc,và cbc.
#	Phù hợp với bất cứ một số nào. Thí dụ, A# phù hợp với A1 đến A9.

Hãy xem một vài thí dụ sử dụng một vài kí tự đại diện trình bày trong bảng 3.2 . Phát biểu SELECT sau đây sử dụng tóan tử LIKE để truy xuất từ bảng Products những dòng có cột ProductName giống như 'Cha_'

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE 'Cha_';
```

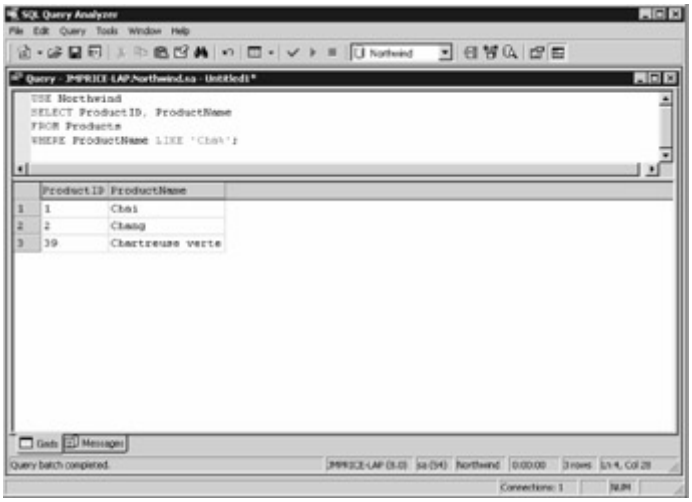
Bảng 3.5 trình bày những kết quả của phát biểu SELECT . LIKE 'Cha_' phù hợp với tên những sản phẩm khởi đầu bằng từ Cha và kết thúc với bất cứ kí tự nào.



Phát biểu SELECT tiếp theo sử dụng tóan tử LIKE để truy xuất những sản phẩm có cột ProductsName giống 'Cha%':

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE 'Cha%';
```

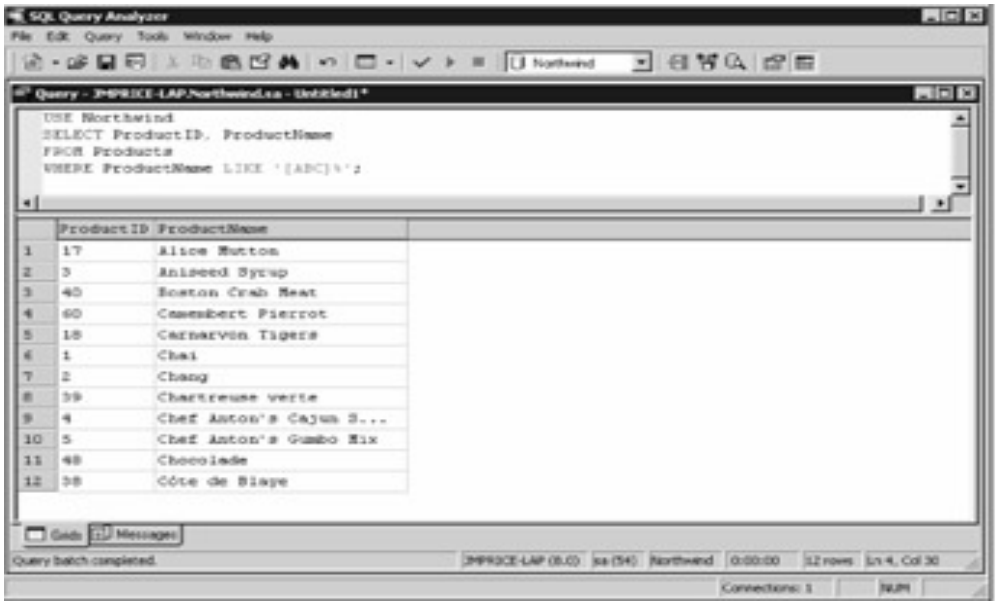
Hình 3.6 trình bày những kết quả của phát biểu SELECT . LIKE ‘Cha%’ phù hợp với những sản phẩm có tên bắt đầu với “Cha” và kết thúc với số lượng kí tự bất kì.



Phát biểu SELECT tiếp theo sử dụng tóan tử LIKE để truy xuất sản phẩm có cột ProductsName giống ‘[ABC]%’ :

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE '[ABC]%';
```

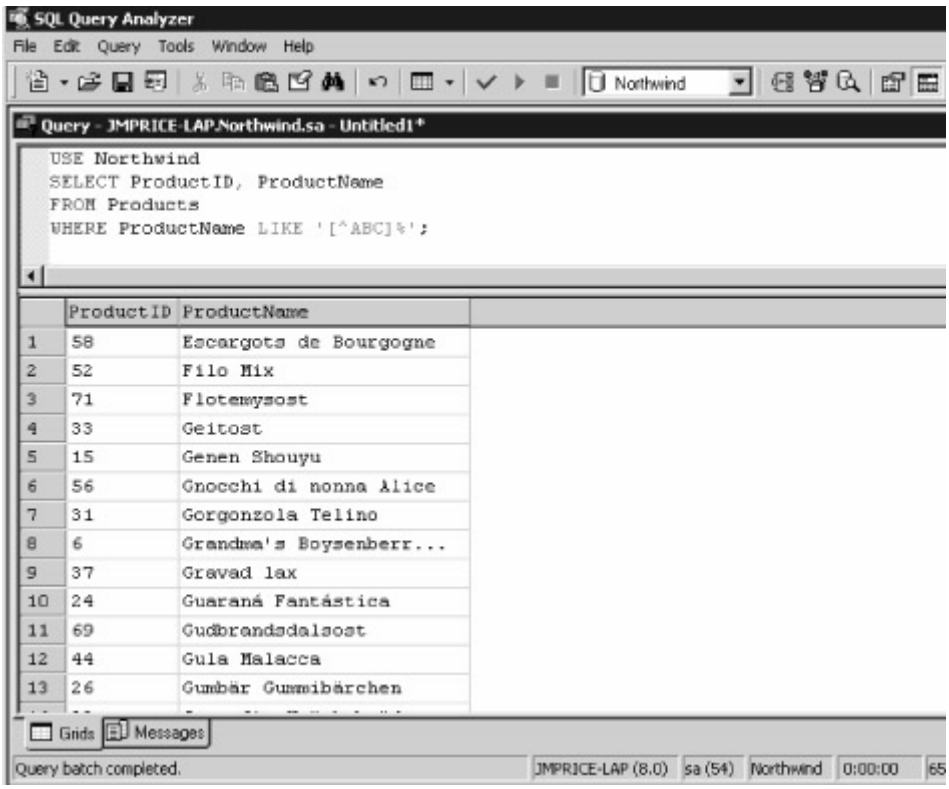
Hình 3.7 trình bày mhững kết quả của phát biểu SELECT . LIKE ‘[ABC]%’ phù hợp với những sản phẩm với tên bắt đầu với một trong các kí tự trong ngoặc vuông : A,B hoặc C và kết thúc với số lượng bất kỳ ký tự



Phát biểu SELECT tiếp theo sử dụng toán tử LIKE để truy xuất những sản phẩm có cột ProductName giống ‘[^ABC]%’ :

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE '[^ABC]%';
```

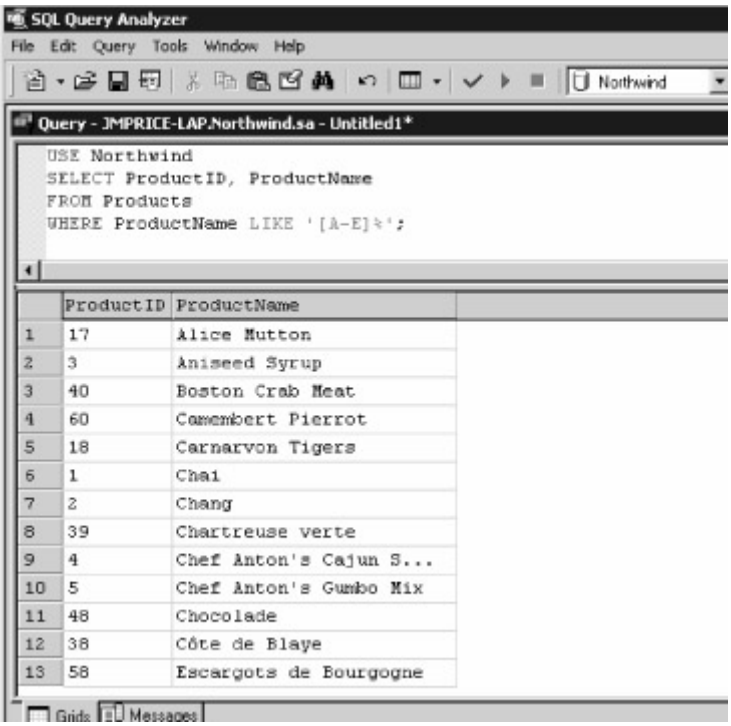
Hình 3.8 trình bày những kết quả của phát biểu SELECT .LIKE ‘[^ABC]%’ phù hợp với những sản phẩm có tên không bắt đầu bằng bất cứ ký tự nào trong ngoặc vuông :A, B hoặc C và kết thúc với số lượng bất kì kí tự



Phát biểu SELECT tiếp theo sử dụng tóan tử LIKE để truy xuất những sản phẩm có cột ProductName giống '[A-E]%':

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE '[A-E]%';
```

Figure 3.9 trình bày kết quả của phát biểu SELECT này. LIKE '[A-E]%' phù hợp với những sản phẩm có tên bắt đầu bằng một kí tự bất kì nào trong dải kí tự từ A đến E, và kết thúc với bất kì kí tự số.



Hình 3.9: những sản phẩm có ProductName giống '[A-E]%'

CHỈ ĐỊNH MỘT DANH SÁCH GIỚI HẠN NHỮNG GIÁ TRỊ:

Bạn sử dụng tóan tử IN trong một mệnh đề WHERE để truy xuất những hàng có những cột chứa những giá trị có trong một danh sách chỉ định. Thí dụ, phát biểu SELECT sau sử dụng tóan tử IN để truy xuất những sản phẩm có ProductID là 1,2,5,20,45 hay 50:

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID IN (1, 2, 5, 15, 20, 45, 50);
```

SỮ DỤNG SUBQUERY ĐỂ LẤY MỘT DANH SÁCH GIÁ TRỊ GIỚI HẠN:

đây là ví dụ khác trình bày cột OrderID từ bảng Orders của những hàng có cột CustomerID có cùng giá trị có trong danh sách truy xuất được từ một truy vấn phụ;truy vấn phụ này truy xuất cột CustomerID từ bảng Customers nơi CompanyName giống **'Fu%':**

```
SELECT OrderID
FROM Orders
WHERE CustomerID IN (
SELECT CustomerID
FROM Customers
WHERE CompanyName LIKE 'Fu%'
);
```

The results of the subquery are used in the outer query.

CHỈ ĐỊNH MỘT DÃY GIÁ TRỊ GIỚI HẠN:

Bạn sử dụng toán tử BETWEEN trong một mệnh đề WHERE để truy xuất những hàng với những cột chứa những giá trị trong một phạm vi được chỉ định. Chẳng hạn, phát biểu SELECT sau sử dụng tóan tử BETWEEN để truy xuất những sản phẩm với ProductID **giữa 1 và 12:**

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID BETWEEN 1 AND 12;
```

Đây là một ví dụ khác hiển thị cột OrderID cho những hàng từ bảng Orders có OrderDate trong khoảng '1996-07-04' và '1996-07-08':

```
SELECT OrderID
FROM Orders
WHERE OrderDate BETWEEN '1996-07-04' AND '1996-07-08';
```

ĐẢO NGƯỢC Ý NGHĨA CỦA MỘT TÓAN TỬ:

Bạn sử dụng từ khóa NOT trước một toán tử trong một mệnh đề WHERE để đảo ngược ý nghĩa của toán tử này. Chẳng hạn, phát biểu SELECT sau sử dụng từ khóa NOT để đảo ngược ý nghĩa của toán tử BETWEEN:

```
SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID NOT BETWEEN 1 AND 12;
```

Chú thích: bạn có thể sử dụng từ khóa NOT để đảo ngược ý nghĩa của những tóan tử khác, thí dụ , NOT LIKE, NOT IN.

TRUY XUẤT NHỮNG HÀNG VỚI CỘT CÓ GIÁ TRỊ NULL:

Trước đó, Tôi đã đề cập đến những cột này có thể chứa những giá trị NULL. Một giá trị NULL khác với ột chuỗi trống hay số không: Một giá trị NULL đại diện cho một giá trị mà chưa được thiết lập, hay chưa được biết. Bạn có thể sử dụng toán tử IS NULL trong một mệnh đề WHERE để xác định một cột có chứa một giá trị NULL hay không. Chẳng hạn, phát biểu SELECT sử dụng toán tử IS NULL để truy xuất những khách hàng có cột Fax chứa một giá trị NULL:

```
SELECT CustomerID, CompanyName, Fax
FROM Customers
WHERE Fax IS NULL;
```

Hình 3.10 cho thấy những kết quả của phát biểu SELECT này .

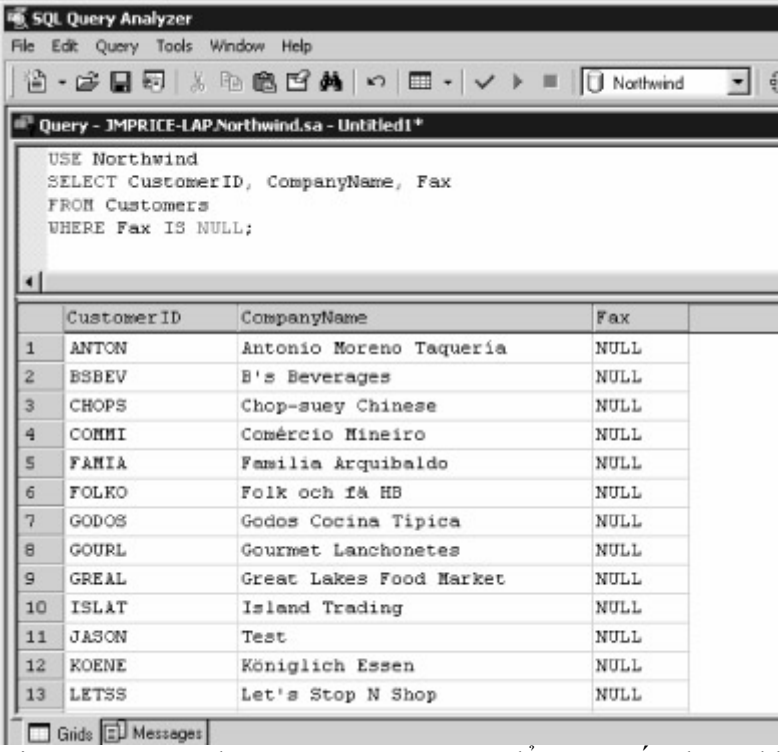


Figure 3.10: Sử dụng toán tử IS NULL để truy xuất những khách hàng có cột Fax chứa một giá trị NULL

Như bạn có thể thấy, những giá trị NULL được trình bày là NULL trong Query Analyzer.

CHỈ ĐỊNH NHỮNG ĐIỀU KIỆN:

Bạn có thể sử dụng những toán tử lôgic được trình bày trong Bảng 3.3 để chỉ rõ những điều kiện trong một mệnh đề WHERE.

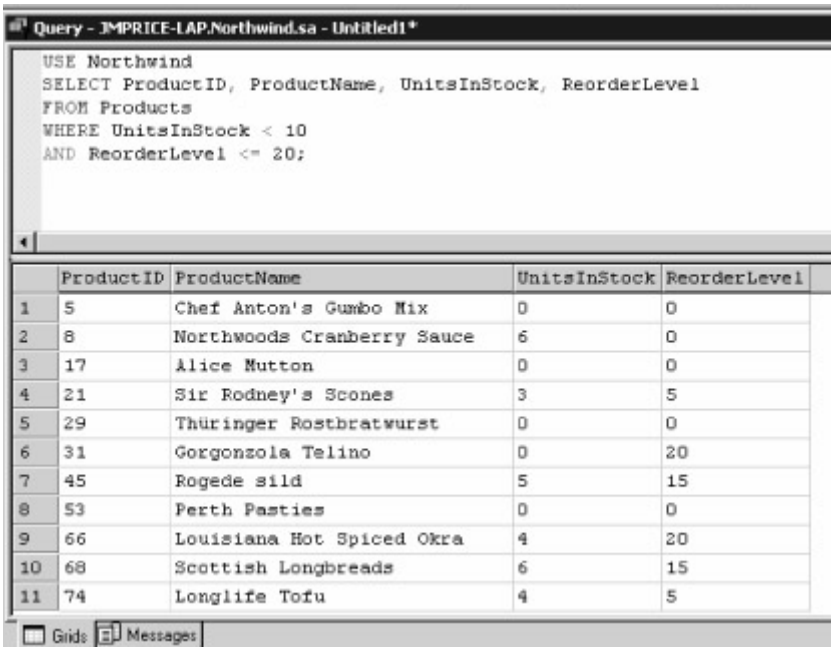
Bảng 3.3: những toán tử lôgic

Toán tử	Mô tả
a AND b	Đặt giá trị tới true khi cả hai a và b đều đúng
a OR b	Đặt giá trị tới true khi một trong hai a hoặc b là true
NOT a	Gán giá trị tới true nếu a là false, và false nếu a là true

Chẳng hạn, phát biểu SELECT sau sử dụng toán tử AND để truy xuất những sản phẩm có cột UnitsInStock nhỏ hơn 10 và cột ReorderLevel cột nhỏ hơn hay bằng 20

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE UnitsInStock < 10
AND ReorderLevel <= 20;
```

Hình 3.11 cho thấy những kết quả của phát biểu SELECT.



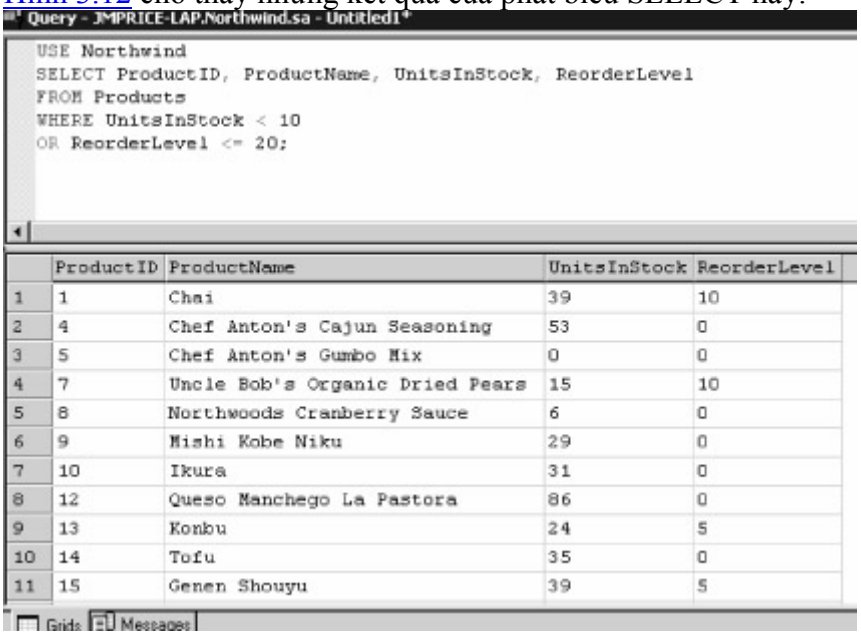
	ProductID	ProductName	UnitsInStock	ReorderLevel
1	5	Chef Anton's Gumbo Mix	0	0
2	8	Northwoods Cranberry Sauce	6	0
3	17	Alice Mutton	0	0
4	21	Sir Rodney's Scones	3	5
5	29	Thüringer Rostbratwurst	0	0
6	31	Gorgonzola Telino	0	20
7	45	Rogede sild	5	15
8	53	Perth Pasties	0	0
9	66	Louisiana Hot Spiced Okra	4	20
10	68	Scottish Longbreads	6	15
11	74	Longlife Tofu	4	5

Hình 3.11: Sử dụng toán tử AND để truy xuất những sản phẩm có UnitsInStock nhỏ hơn 10 và ReorderLevel nhỏ hơn hay bằng 20

Trong ví dụ kế tiếp,Phát biểu SELECT sử dụng **toán tử OR** để truy xuất những sản phẩm có cột UnitsInStock nhỏ hơn 10 hoặc cột ReorderLevel nhỏ hơn hay bằng 20

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE UnitsInStock < 10
OR ReorderLevel <= 20;
```

Hình 3.12 cho thấy những kết quả của phát biểu SELECT này.



	ProductID	ProductName	UnitsInStock	ReorderLevel
1	1	Chai	39	10
2	4	Chef Anton's Cajun Seasoning	53	0
3	5	Chef Anton's Gumbo Mix	0	0
4	7	Uncle Bob's Organic Dried Pears	15	10
5	8	Northwoods Cranberry Sauce	6	0
6	9	Mishi Kobe Niku	29	0
7	10	Ikura	31	0
8	12	Queso Manchego La Pastora	86	0
9	13	Konbu	24	5
10	14	Tofu	35	0
11	15	Genen Shouyu	39	5

Hình 3.12: Sử dụng toán tử OR để truy xuất những sản phẩm có UnitsInStock nhỏ hơn 10 hoặc ReorderLevel nhỏ hơn hay bằng 20

Phát biểu SELECT tiếp theo sử dụng **toán tử NOT** để truy xuất những sản phẩm có cột UnitsInStock không nhỏ hơn 10

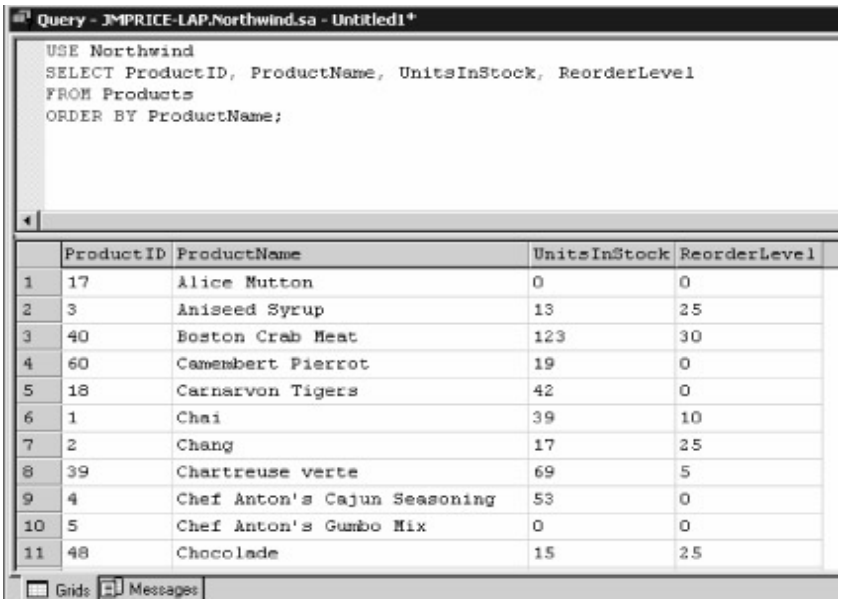
```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE NOT (UnitsInStock < 10);
```

SẮP XẾP NHỮNG HÀNG:

Bạn có thể sử dụng mệnh đề **ORDER BY** để phân loại sắp xếp những hàng truy xuất được từ cơ sở dữ liệu. Bạn chỉ định cột (hay những cột) để phân loại sắp xếp trong mệnh đề ORDER BY. Theo mặc định, những hàng được phân loại trong thứ tự tăng dần. Chẳng hạn, phát biểu SELECT sau đây sắp xếp những hàng sử dụng cột ProductName :

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
ORDER BY ProductName;
```

Hình 3.13 cho thấy những kết quả của phát biểu SELECT này. Như bạn có thể thấy, những hàng được sắp xếp theo thứ tự tăng dần sử dụng cột ProductName .



	ProductID	ProductName	UnitsInStock	ReorderLevel
1	17	Alice Mutton	0	0
2	3	Aniseed Syrup	13	25
3	40	Boston Crab Meat	123	30
4	60	Camembert Pierrot	19	0
5	18	Carnarvon Tigers	42	0
6	1	Chai	39	10
7	2	Chang	17	25
8	39	Chartreuse verte	69	5
9	4	Chef Anton's Cajun Seasoning	53	0
10	5	Chef Anton's Gumbo Mix	0	0
11	48	Chocolade	15	25

Hình 3.13: Sử dụng mệnh đề ORDER BY để sắp xếp những sản phẩm theo giá trị tăng dần của ProductName

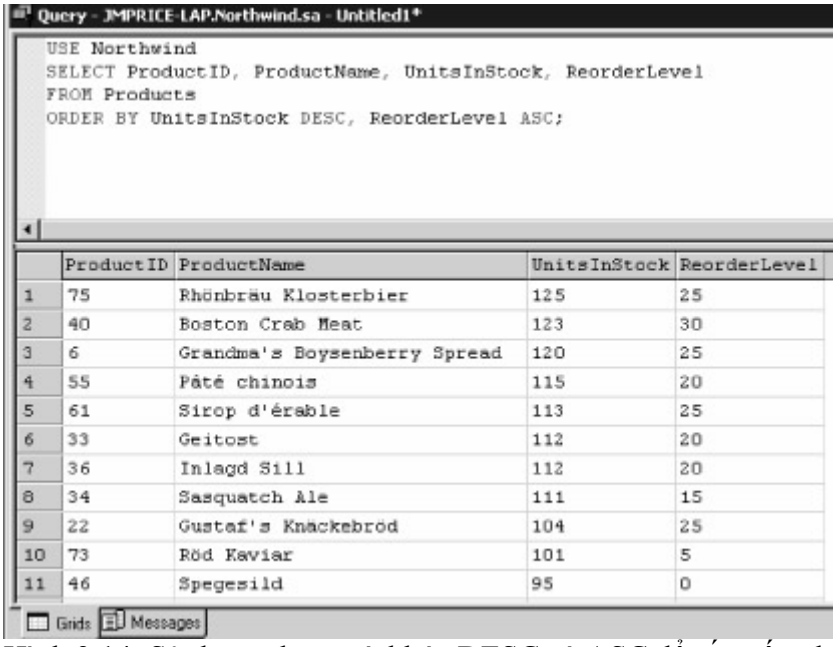
Bạn có thể phát biểu rõ ràng sự phân loại sắp xếp cho một cột sử dụng từ khóa ASC hay DESC. ASC sắp xếp những cột theo thứ tự tăng dần (tiết mục nhỏ nhất đầu tiên), và DESC sắp xếp những cột theo thứ tự giảm dần (tiết mục lớn nhất đầu tiên). Chẳng hạn, phát biểu SELECT sau đây sắp xếp những sản phẩm theo thứ tự giảm dần sử dụng cột ProductName :

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
ORDER BY ProductName DESC;
```

Bạn có thể chỉ định nhiều cột trong một mệnh đề ORDER BY. Chẳng hạn, phát biểu SELECT sau đây sắp xếp những hàng sử dụng cả hai cột UnitsInStock và ReorderLevel :

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
ORDER BY UnitsInStock DESC, ReorderLevel ASC;
```

Hình 3.14 cho thấy những kết quả của phát biểu SELECT này. Như bạn có thể nhìn thấy, những hàng được sắp xếp cột UnitsInStock trước tiên (theo thứ tự giảm dần), và sau đó là cột ReorderLevel (theo thứ tự tăng dần).



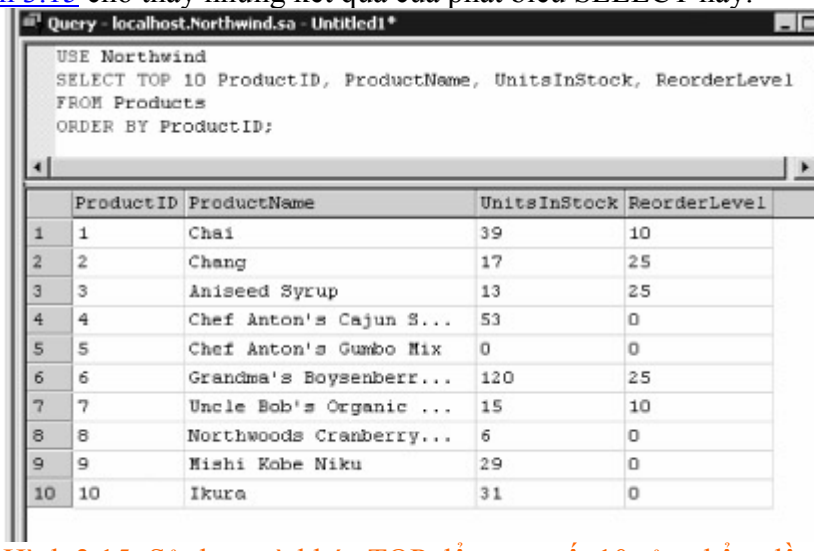
Hình 3.14: Sử dụng những từ khóa DESC và ASC để sắp xếp những sản phẩm với sự giảm dần UnitsInStock và tăng lên ReorderLevel

TRUY XUẤT N DÒNG ĐẦU TIÊN:

Bạn sử dụng từ khóa TOP để chỉ truy xuất N hàng đầu tiên từ một phát biểu SELECT. Chẳng hạn, phát biểu SELECT sau sử dụng từ khóa TOP để truy xuất 10 hàng đầu tiên từ bảng những sản phẩm, được sắp xếp bởi cột ProductID:

```
SELECT TOP 10 ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
ORDER BY ProductID;
```

Hình 3.15 cho thấy những kết quả của phát biểu SELECT này.



Hình 3.15: Sử dụng từ khóa TOP để truy xuất 10 sản phẩm đầu tiên bởi ProductID

LOẠI TRỪ NHỮNG HÀNG TRÙNG LẶP:

Bạn sử dụng từ khóa DISTINCT để loại trừ những hàng trùng lặp được truy xuất bởi một phát biểu SELECT. Chẳng hạn, phát biểu sau đây sử dụng từ khóa DISTINCT để truy xuất những giá trị cột Country phân biệt từ bảng những khách hàng :

```
SELECT DISTINCT Country
FROM Customers;
```

Hình 3.16 cho thấy những kết quả của phát biểu SELECT này.

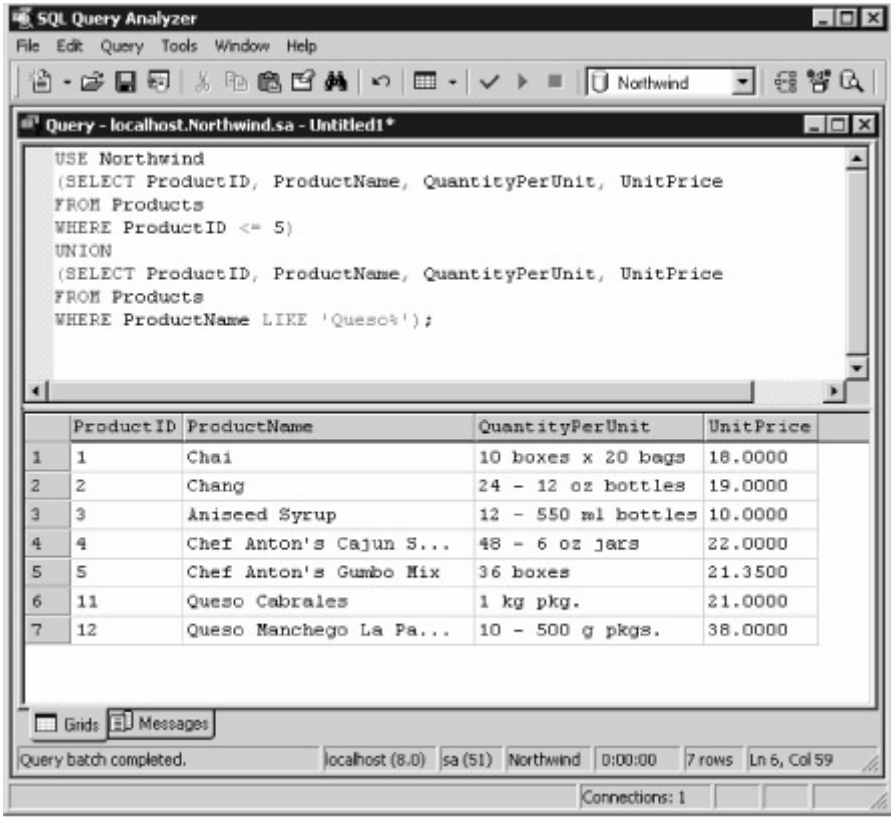
Figure 3.16: Using the DISTINCT keyword to retrieve distinct Country column values

Như bạn có thể thấy, Phát biểu SELECT chỉ trình bày những giá trị cột Country có ý nghĩa riêng biệt và duy nhất: những giá trị trùng lặp khác được loại trừ. Nếu bạn không bao gồm từ khóa DISTINCT, thì tất cả những giá trị cột Country sẽ được hiển thị bao gồm cả những giá trị trùng lặp.

SỰ KẾT HỢP NHỮNG HÀNG ĐƯỢC TRUY XUẤT TỪ PHÁT BIỂU SELECT:

Bạn sử dụng tóan tử UNION để kết hợp những hàng tuy xuất được từ phát biểu SELECT vào trong một bộ những dòng.
Thí dụ : phát biểu SELECT dưới đây sử dụng tóan tử UNION để kết hợp những hàng truy xuất được từ phát biểu SELECT
Sử dụng tóan tử UNION để kết hợp những hàng truy xuất được từ bảng Products ; những hàng truy xuất lần thứ nhất có ProducID nhỏ hơn hay bằng 5, và những cột truy xuất lần hai có ProductName khởi đầu với “Queso”:

```
(SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductID <= 5)
UNION
(SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice
FROM Products
WHERE ProductName LIKE 'Queso%');
```



CHIA NHỮNG HÀNG TRUY XUẤT ĐƯỢC VÀO TRONG NHỮNG KHỐI:

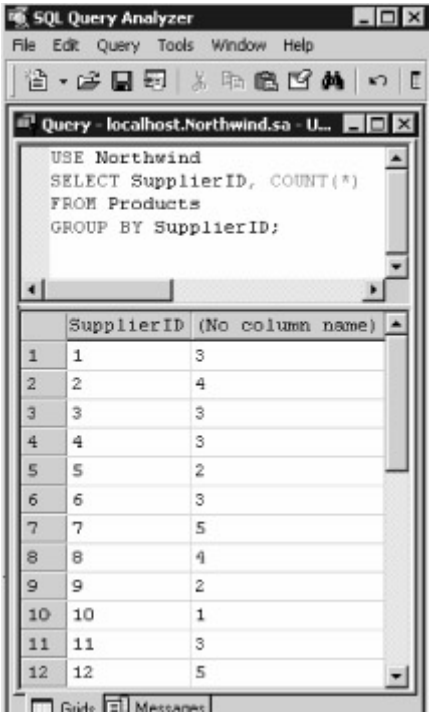
Bạn sử dụng mệnh đề GROUP BY để chia những hàng truy xuất được vào trong những khối. bạn có thể xem một block như là một nhóm những hàng được cô đọng vào trong một hàng . thí dụ , bạn nhóm cột mã nhà cung ứng (SupplierID) của những hàng trong bảng sản phẩm(Products). Bạn sẽ được một hàng cho mỗi hàng có cột SupplierID cùng giá trị.

```
SELECT SupplierID
FROM Products
GROUP BY SupplierID;
```

Phát biểu SELECT này hiển thị một hàng cho mỗi nhóm những hàng có cùng giá trị SupplierID. Bạn có thể lấy được số lượng hàng trong mỗi block nhờ sử dụng hàm COUNT(). COUNT() là một trong những hàm đi cùng với SQL Server, và được biết như một hàm tổng thể bởi vì nó có thể thao tác trên nhiều dòng trong cùng một lúc. Bạn có thể sử dụng COUNT() để lấy thông tin về số lượng hàng , như trình bày trong thí dụ dưới đây:

```
SELECT SupplierID, COUNT(*)
FROM Products
GROUP BY SupplierID;
```

Hình 3.18 trình bày kết quả của phát biểu SELECT này

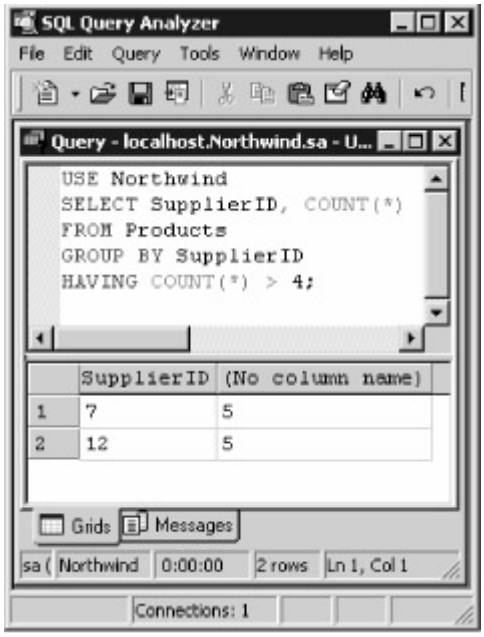


HẠN CHẾ NHỮNG NHÓM CỦA NHỮNG HÀNG ĐƯỢC TRUY XUẤT:

Bạn sử dụng mệnh đề HAVING để hạn chế những nhóm của những hàng truy xuất được bởi mệnh đề GROUP BY. Thí dụ :
Phát biểu SELECT sau đây sử dụng mệnh đề HAVING để hạn chế nhóm những hàng trả về có hơn bốn hàng trong một nhóm:

```
SELECT SupplierID, COUNT(*)
FROM Products
GROUP BY SupplierID
HAVING COUNT(*) > 4;
```

Hình 3.19 trình bày kết quả của phát biểu SELECT này

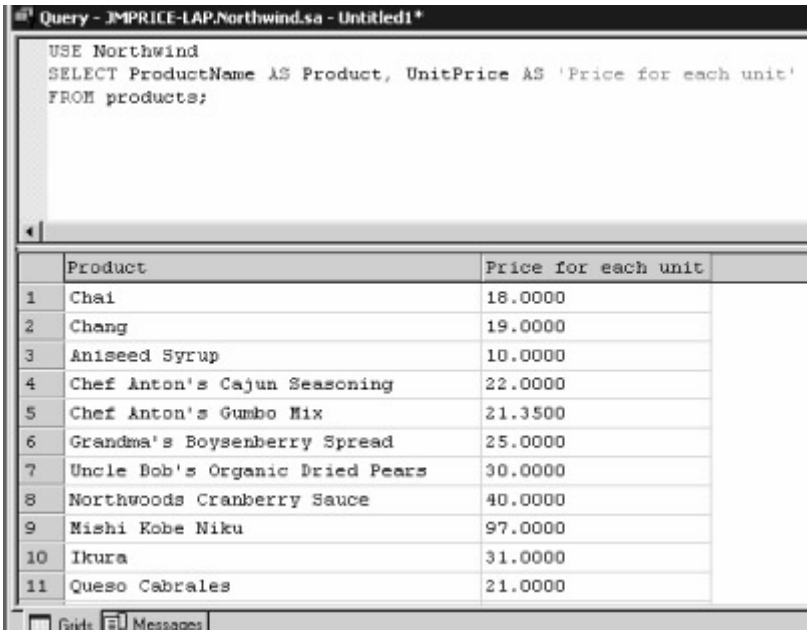


CHỈ ĐỊNH HIỂN THỊ TÊN MỘT CỘT VÀ ĐỊNH BIỆT DANH CHO MỘT BẢNG:

Bạn có thể sử dụng mệnh đề AS để định tên cho một cột khi nó được hiển thị trong bảng trả về từ phát biểu SELECT. Bạn có thể muốn thực hiện điều này khi cần hiển thị những tên thân thiện hay mô tả tên những cột. thí dụ , phát biểu SELECT sau đây sử dụng mệnh đề AS để định tên hiển thị của cột ProductName thành Product, and cột UnitPrice thành Price cho mỗi đơn vị:

```
SELECT ProductName AS Product, UnitPrice AS 'Price for each unit'
FROM products;
```

Hình 3.20 trình bày những kết quả của phát biểu này



Bạn cũng có thể sử dụng mệnh đề AS để đặt tên đại diện cho bảng. bạn có thể muốn thực hiện điều này nếu như tên bảng của bạn dài. Thí dụ sau đây sử dụng mệnh đề AS để đặt tên đại diện cho bảng Customers và Orders tương ứng với Cust và Ord

```
SELECT Cust.CustomerID, CompanyName, Address, OrderID, ShipAddress
FROM Customers AS Cust, Orders AS Ord
WHERE Cust.CustomerID = Ord.CustomerID
AND Cust.CustomerID = 'ALFKI';
```

THỰC HIỆN NHỮNG TÍNH TÓAN DỰA VÀO NHỮNG CỘT CHỨA GIÁ TRỊ:

Bạn điển hình sử dụng những cột được tính tóan để thực hiện những tính toán dựa trên những giá trị của cột . thí dụ , bạ có thể muốn sử dụng một trường được tính tóan để tính tóan kết quả của việc tăng giá trị của cột UnitPrice trong bảng Products lên 20%. Thí dụ:

```
SELECT UnitPrice * 1.20
FROM Products
WHERE ProductID = 1;
```

Kết quả trả về là 21.600000 (18000000*1.2)

Thí dụ tiếp theo liên kết giá trị cột ContactName và ContactTitle từ bảng Customers nơi dòng nào có CustomerID bằng ALFKI:

```
SELECT ContactName + ', ' + ContactTitle
FROM Customers
WHERE CustomerID = 'ALFKI';
```

Kết quả trả về Maria Anders, Sales Representive

TRUY XUẤT NHỮNG HÀNG TỪ NHIỀU BẢNG:

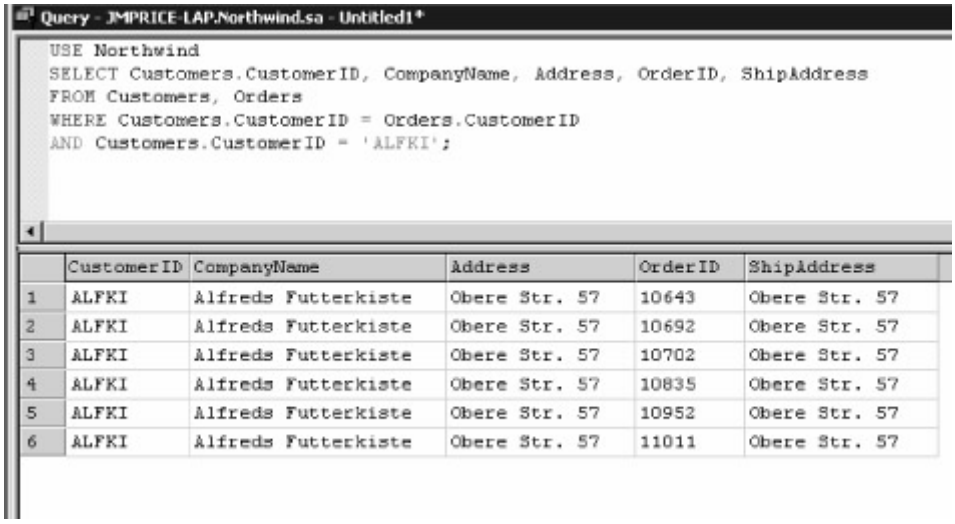
Cho đến lúc này bạn đã được nhìn thấy những phát biểu SELECT truy xuất những hàng từ chỉ một bảng cho một lần thực thi. Bạn sẽ luôn luôn cần truy xuất những hàng từ nhiều bảng sử dụng cùng mộtphát biểu SELECT.

Thí dụ: bạn có thể muốn thấy tất cả những đơn đặt của một khách hàng. Để thực kiện điều này ,bạn phải chỉ định cả hai bảng Customers và Orders sau từ khóa FROM trong phát biểu SELECT và sử dụng *a table join* trong mệnh đề WHERE.
Bạn cũng phải chỉ định tên của bảng khi tham chiếu đến những cột cùng tên trong cả hai bảng. phát biểu SELCET
Sau đây trình bày và truy xuất những đơn đặt của khách hàng với CustomerID là ALFKI:

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

Chú ý rằng bảng Customers và Orders được chỉ định sau từ khóa FROM, bởi vì cả hai bảng đều chứa cột có tên là CustomerID, tên bảng được đặt trước mỗi tham chiếu đến cột tương ứng trong mỗi bảng. sự nối bảng được thự hiện trên cột CustomerID cho mỗi bảng (Customers.CustomerID = Orders.SustomerID).

Hình 3.21 trình bày những kết quả của phát biểu SELECT này.



	CustomerID	CompanyName	Address	OrderID	ShipAddress
1	ALFKI	Alfreds Futterkiste	Obere Str. 57	10643	Obere Str. 57
2	ALFKI	Alfreds Futterkiste	Obere Str. 57	10692	Obere Str. 57
3	ALFKI	Alfreds Futterkiste	Obere Str. 57	10702	Obere Str. 57
4	ALFKI	Alfreds Futterkiste	Obere Str. 57	10835	Obere Str. 57
5	ALFKI	Alfreds Futterkiste	Obere Str. 57	10952	Obere Str. 57
6	ALFKI	Alfreds Futterkiste	Obere Str. 57	11011	Obere Str. 57

Phát biểu SELECT trước sử dụng định dạng tiêu chuẩn SQL cho việc liên kết những bảng. với SQL Server, bạn cũng có thể sử dụng từ khóa JOIN cho việc liên kế những bảng. lợi thế của từ khóa JOIN là bạn có thể sử dụng nó để thực hiện những kết nối ngòai. Bạn sẽ học đến ngay sau đây. Đây là một thí dụ nó viết lại những phát biểu SELECT trước sử dụng từ khóa JOIN :

```
SELECT Customers.CustomerID, CompanyName, Address, OrderID, ShipAddress
FROM Customers
JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
AND Customers.CustomerID = 'ALFKI';
```

Phát biểu SELECT này trả về cùng một kết quả như thí dụ trước.

Sự bất lợi của hai phát biểu SELECT trước là chúng trả về những hàng chỉ khi cả hai cột liên kết có một giá trị. Nghĩa là ,
Không một cột nào chứa giá trị NULL. Đều này có thể gây ra một lỗi nếu bạn có những hàng có một giá trị NULL trong
cả những cột sử dụng trong liên kết và bạn muốn thực sự truy xuất những hàng đó .những liên kết ngòai (Outer Jions)
giải quyết vấn đề này .sau đây là ba kiểu Outer Joins :

LEFT OUTER JOIN: (luôn được rút ngắn tới LIÊN KẾT TRÁI) trả về tất cả những hàng từ bảng phía bên trái của liên kết, bao gồm cả những hàng có một cột chứa giá trị NULL.

RIGHT OUTER JOIN : (luôn được rút ngắn tới LIÊN KẾT PHẢI) trả về tất cả những hàng từ một bảng phía bên phải liên kết, bao gồm cả những hàng có một cốt chứa giá trị NULL.

FULL OUTER JOIN : (luôn được rút ngắn tới LIÊN KẾT ĐẦY) trả về tất cả những hàng từ những bảng phía phải và cả phía trái cua liên kết, bao gồm những hàng có một cột có giá trị NULL.

chúng ta hãy xem cặp thí dụ dưới đây. Đầu tiên thực hiện phát biểu INSERT để thêm một hàng vào bảng Products:

```
INSERT INTO Products (ProductName, SupplierID)
 VALUES ('DVD Player', NULL);
```

Chú thích : bạn sẽ học chi tiết về phát biểu INSERT sau cũng trong chương này.

Bạn không cần phải chỉ định cột ProducID bởi vì SQL Server sẽ tự động cung cấp một giá trị sử dụng một phương thức đặt mã khoá tự động. Sự nhận danh này được thành lập khi bảng Products được tạo ra, và phương thức đặt mã khoá tự động sẽ phát sinh một loạt những giá trị ,giá trị được phát sinh lần đầu tiên khi được gọi là 1 và được tăng lên một đơn vị với mội lần gọi tiếp theo. Thí dụ ,cột ProductID từ lúc đầu nhập liệu đến giờ đã chứa một loạt những giá trị từ 1 đến 77 (77 hàng) do đó phát biểu INSERT tiếp theo sẽ thêm một hàng vào bảng Products phương thức đặt mã khoá tự động sẽ đặt giá trị cho cột ProductID của dòng mới này là 78 .

Bạn chú ý là cột SupplierID trong phát biểu INSERT này là NULL. Nếu bây giờ bạn thực hiện phát biểu SELECT tiếp sau, bạn sẽ không nhìn thấy hàng mới vừa INSERT bởi vì cột SupplierID của hàng mới này là NULL và do đó từ khoá JOIN trong phát biểu không thực hiện được :

```
SELECT ProductID
FROM Products
JOIN Suppliers
ON Products.SupplierID = Suppliers.SupplierID;
```

Để thấy được hàng mới này , bạn sử dụng LEFT JOIN trong phát biểu SELECT để truy xuất tất cả những hàng từ bảng bên trái từ khoá LEFT JOIN (trong trường hợp này , là bảng Products):

```
SELECT ProductID
FROM Products
LEFT JOIN Suppliers
ON Products.SupplierID = Suppliers.SupplierID;
```

Bạn cũng có thể sử dụng LEFT JOIN với từ khoá IS NULL trong phát biểu SELECT để chỉ truy xuất hàng mới này:

```
SELECT ProductID
FROM Products
LEFT JOIN Suppliers
```

```
ON Products.SupplierID = Suppliers.SupplierID
WHERE Products.SupplierID IS NULL;
```

TRUY XUẤT NHỮNG HÀNG TỪ MỘT BẢNG VIEW:

Bạn sử dụng một bảng VIEW để truy xuất một bộ những cột từ một hoặ nhiều bảng. bạn có thể xemmột View như một cách linh hoạt hơn để xem xét những hàng lưu trữ trong những bảng. thí dụ , một trong những bảng view của cơ sở dữ liệu NorthWind truy xuất một danh sách những sản phẩm (products) xếp theo thứ tự abc. Và truy xuất tên sản phẩm
(product name) , loại sản phẩm (Categories name), trong số những cột khác. Những thông tin này được rút từ cả hai bảng Products và Categories. Bảng view này được đặt tên là Alphabetical list of products và phát biểu SELECT tạo nên bảng View này như sau:

```
SELECT Products.*, Categories.CategoryName
FROM Categories INNER JOIN Products ON
Categories.CategoryID = Products.CategoryID
WHERE (((Products.Discontinued)=0));
```

Bạn có thể truy xuất tất cả các cột và hàng từ những bảng được tham chiếu bởi View này bằng cách sữ dụng phát biểu SELECT dưới đây:

```
SELECT *
FROM [Alphabetical list of products];
```

Bạn cũng có thể truy xuất những cột riêng biệt từ một view. Thí dụ :

```
SELECT ProductName, CategoryName
FROM [Alphabetical list of products];
```

THÊM MỘT HÀNG MỚI VÀO MỘT BẢNG:

Bạn dử dụng phát biểu INSERT để thêm một hàng mới vào một bảng. khi thêm một hàng mới , bạn chỉ định tên của bảng, tên cuả những cột tuỳ chọn , và những giá trị cho những cột này . thí dụ :

```
INSERT INTO Customers (
CustomerID, CompanyName, ContactName, ContactTitle, Address,
City, Region, PostalCode, Country, Phone, Fax)
VALUES ( 'JPCOM', 'Jason Price Company', 'Jason Price', 'Owner', '1 Main Street',
 'New York', NULL, '12345', 'USA', '(800)-555-1212', NULL);
```

Cột CustomerID là khoá chính của bảng Customers, do đó hàng mới phải chứa một giá trị duy nhất cho cột này. Bạn chú ý rằng phát biểu INSERT chỉ định một giá trị NULL cho cột Region và Fax (chỉ định này sử dụng từ khoá NULL).

Bạn có thể sử dụng Query Analyzer và nhập những phát biểu INSERT. Hình 3.22 trinh bày phát biểu INSERT trước và theo sau là một phát biểu SELECT truy xuất hàng mới INSERT.

Chú ý : bạn phải cung cấp giá trị cho tất cả các cột mà đã được định nghiã là NOT NULL trong bảng. cũng như số lượng trong INSERT và danh sách VALUES phải phù hợp, và cả kiểu dữ liệu của mỗi cột cũng phải phù hợp.

Khi cung cấp những giá trị cho tất cả các cột của trong một hàng, bạn có thể bỏ qua những tên cột và chỉ cung cấp những giá trị cho mỗi cột. thí dụ:

```
INSERT INTO Customers VALUES (
'CRCOM', 'Cynthia Red Company', 'Cynthia Red', 'Owner', '2 South Street',
'New York', NULL, '12345', 'USA', '(800)-555-1212', NULL
);
```

SỬA ĐỔI NHỮNG HÀNG TRONG MỘT BẢNG:

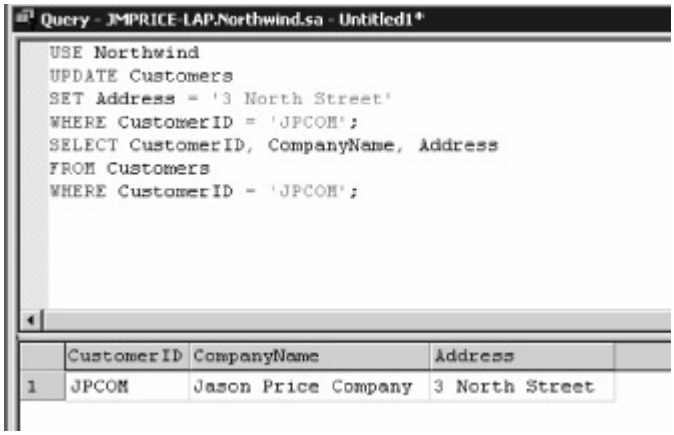
Bạn sử dụng phát biểu UPDATE để cập nhật những hàng trong bảng. khi thực hiện cập nhật một hàng, bạn chỉ định tên cuả bảng , tên những cột cần cập nhật, và những giá trị mới cho những cột.

Cảnh báo: một cách điển hình , bạn cũng phải sử dụng mệnh đề WHERE để hạn chế những hàng sẽ được cập nhật. nếu bạn không cung cấp mệnh đề WHERE , thì tất cả những hàng trong bảng chỉ định đều sẽ bị cập nhật. trong rất nhiều trường hợp bạn bạn sẽ phải chỉ định giá trị cho khoá chính trong mệnh đề WHERE.

Phát biểu UPDATE sau sửa đổi cột đia chỉ cho hàng trong bảng Customers với CustomerID = JPCOM :

```
UPDATE Customers
SET Address = '3 North Street'
WHERE CustomerID = 'JPCOM';
```

hình 2.23 trình bày phát biểu UPDATE , và theo sau với một phát biểu SELECT truy xuất hàng được sưả đổi.



Bạn có thể sử dụng phát biểu UPDATE để sửa đổi nhiều cột. thí dụ, phát biểu sau sửa đổi giá trị các cột Address và ContactTitle :

```
UPDATE Customers
SET Address = '5 Liberty Street', ContactTitle = 'CEO'
WHERE CustomerID = 'JPCOM';
```

XOÁ NHỮNG HÀNG TỪ MỘT BẢNG:

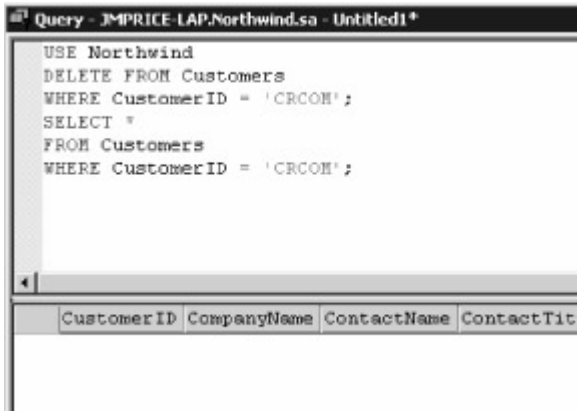
Bạn sử dụng phát biểu DELETE để xoá những hàng từ một bảng. khi thực hiện xoá một hàng, bạn chỉ định tên bảng và những hàng cần xoá sử dụng mệnh đề WHERE .

Cảnh báo : nếu bạn bỏ qua mệnh đề WHERE trong phát biểu DELETE , tất cả những hàng trong bảng đều sẽ bị xoá . bảo đảm rằng bạn có cung cấp mệnh đề WHERE nếu như bạn không muốn xoá hết hàng trong bảng.

Phát biểu DELETE sau xoá những hàng trong Customers với những hàng có CustomerID = CRCOM :

```
DELETE FROM Customers
WHERE CustomerID = 'CRCOM';
```

Hình 3.24 trình bày phát biểu DELETE này , cùng với một phát biểu SELECT chứng minh rằng hàng trên đã bị xoá.



Trong phần tiếp theo chúng ta sẽ học về phần mềm cơ sở dữ liệu duy trì tính toàn vẹn của những thông tin lưu trữ trongcơ sở như thế nào.

SỰ DUY TRÌ TÍNH TOÀN VẸN CHO CƠ SỞ DỮ LIỆU:

Phần mềm cơ sở dữ liệu bảo đảm rằng những thông tin lưu trữ trong các bảng là nhất quán . trong những điều kiện kỷ thuật, nó duy trì tính toàn vẹn của thông tin. Hai thí dụ về điều này như sau :

- khoá chính cuả một hàng luôn luôn chứa một diá trị duy nhất.
- khoá phu cuả một hàng trong bảng con luôn tham chiếu một giá trị hiện hữu trong bảng cha.

Hãy xem những gì sảy ra khi bạn cố gán một hàng vào một bảng với một khoá chính đã tồn tại trong bảng. phát biểu INSERT sau đang cố thử thêm một hàng vào bảng Customers với CustomerID = ALFKI (khoá này đã tồn tại trong bảng Customers)

```
INSERT INTO Customers (
CustomerID, CompanyName, ContactName, ContactTitle, Address,
City, Region, PostalCode, Country, Phone, Fax
) VALUES (
'ALFKI', 'Jason Price Company', 'Jason Price', 'Owner', '1 Main Street',
```

```
'New York', NULL, '12345', 'USA', '(800)-555-1212', NULL
);
```

Nếu bạn cố thử chạy phát biểu INSERT này, bạn sẽ nhận một thông báo lỗi từ co8 sở dữ liệu :

Violation of PRIMARY KEY constraint 'PK_Customers'.
Cannot insert duplicate key in object 'Customers'.
The statement has been terminated.

Hãy xem những gì sẽ sảy ra khi bạn cố sửa đổi một khoá chính ở bảng cha có một giá trị đã được tham chiếu từ một khoá phụ trong bảng con. Phát biểu UPDATE sau cố sửa đổi CustomerID từ ALFKI thành ALFKZ trong bảng cha Customers (hàng này được tham chiếu bởi những hàng trong bảng con Orders) :

```
UPDATE Customers
SET CustomerID = 'ALFKZ'
WHERE CustomerID = 'ALFKI';
```

Nếu bạn cố chạy thử phát biểu UPDATE này , bạn sẽ nhận thông báo lỗi như sau:

Phát biểu UPDATE này xung đột với cột tham chiếu ràng buộc 'FK_Orders_Customers'. Sự xung đột này sảy ra trong cơ sở dữ liệu ‘NorthWind’, bàng ‘Orders’, cột ‘CustomerID’. Phát biểu này đã bị giới hạn.

Phát biểu UPDATE này bị hỏng vì hàng chứa giá trị khoá chính là ALFKI được tham chiếu bởi những han2g trong bàng Orders. Thông báo cho bạn biết giá trị mới cho cột CustomerID đã vi phạm sự ràng buộc khoá phụ trên cột CustomerID của bàng Orders. Ràng buộc nà có tên FK_Orders_Customers.

Bạn cũng không thể xoá một hàng từ một bảng cha được tham chiếu bởi một hàng trong bảng con. Thí dụ, phát biểu SELECT dưới đây cố thử xoá hàng từ bảng Customers có cột CustomerID bằng ALFKI (hàng này được tham chiếu bởi những hàng trong bảng Orders) :

```
DELETE FROM Customers
WHERE CustomerID = 'ALFKI';
```

Nếu bạn cố chạy thử phát biểu DELETE này ,bạn sẽ nhận được thông báo lỗi tương tự như trên.

GOM NHÓM NHỮNG CÂU LỆNH SQL:

Theo mặc định , khi bạn chạy một phát biểu INERT, UPDATE, hay DELETE , SQL Server luôn ghi lại những kết quả của phát biểu vào cơ sở dữ liệu . đây có thể không phải luôn luôn là kết quả mong muốn của bạn. thí dụ , trong trường hợp một giao dịch ngân hàng, bạn có thể muốn rút tiền từ một tài khoản và giửi sang một tài khoản khác. Nếu bạn có hai phát biểu UPDATE riêng biệt thực hiện việc rút và gởi, và bạn muốn thực hiện kết quả cho mỗi phát biểu UPDATE thường xuyên chỉ như một đơn vị. Nếu như sự cập nhật bị thất bại do một lí do nào đó, và bạn muốn huỷ những kết quả của cả hai phát biểu UPDATE này.

Chú thích: luôn luôn thực hiện việc ghi lại những kết quả của những phát biểu SQL được hiểu như là một cam kết, hay sự giao phó cho những câu lệnh SQL. Việc huỷ những kết quả của những câu lệnh SQL được hiểu như một sự trả lại tình trạng cũ (như khi chưa thực thi UPDATE).

Bạn có thể nhóm các câu lệnh SQL trong một *trasaction.* Và sau đó bạn có thể thực hiện hoặc quay trở lại những câu lệnh SQL trong *transaction* đó như một đơn vị. thí dụ , hai phát biểu trước trong thí dụ về ngân hàng có thể được đặt vào trong một *transaction* , và rồi bạn có thể giao phó sự thực hiện hay quay ngược giao dic5ch này như một đơn vị,
Tuỳ thuộc vào hai phát biểu này có thành công hay không.

Bạn khởi chạy một giao dịch sử dụng phát biểu BEGIN TRANSASTION hay phiên bản tốc ký ,BEGIN TRAN. Sau đó bạn có thể thực hiện những phát biểu SQL đã tạo thành *transaction* này. Để giao phó sự thực thi cho

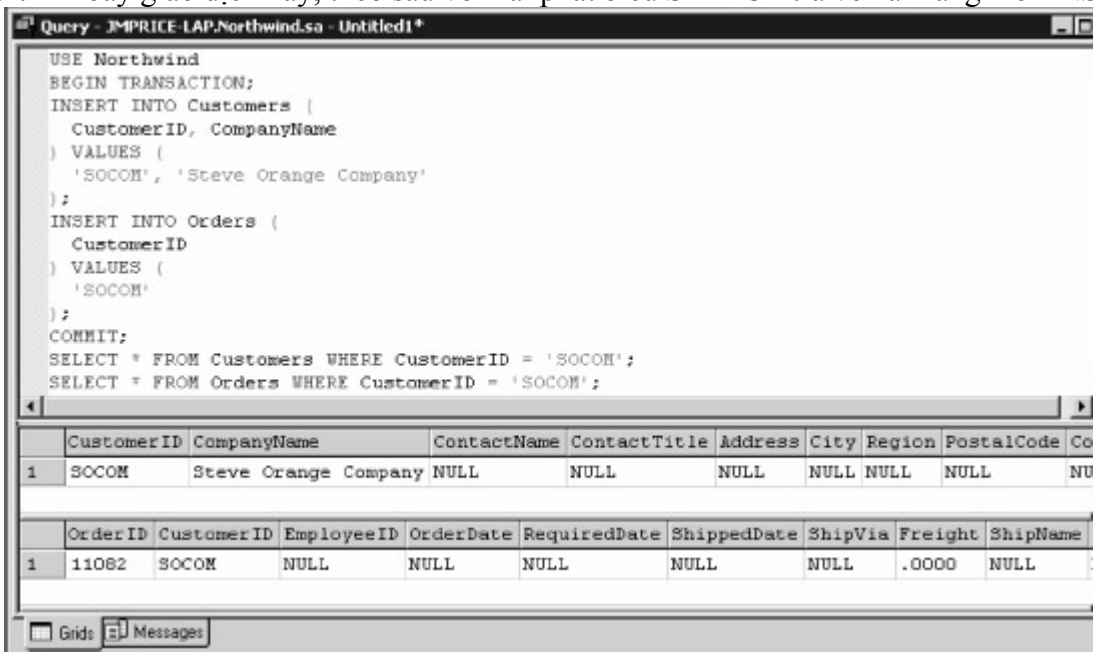
transaction bạn thực thi một phát biểu COMMIT TRANSACTION hoặc một trong số trong phiên bản tốc ký, COMMIT TRAN hoặc COMMIT. Để roll back một transaction ,bạn thực hiện một phát biểu ROLLBACK TRANSACTION hoặc một trong số những phiên bản tốc ký , ROLLBACK TRANS hay ROLLBACK.

Chú ý : theo mặc định , transaction là phục hồi (rollback) . bạn phải luôn luôn chỉ định rõ ràng commit hay roll back một transaction để cho biết bạn muốn làm gì.

Hãy xem một thí dụ , transaction dưới đây bao gồm hai phát biểu INSERT : phát biểu thứ nhất thêm một hàng vào bảng Customers, và phát biểu thứ hai thêm một hàng vào bảng Orders. Cuối cùng , giao dịch này được giao phó sử dụng phát biểu COMMIT :

```
BEGIN TRANSACTION;
INSERT INTO Customers (
CustomerID, CompanyName
) VALUES (
'SOCOM', 'Steve Orange Company'
);
INSERT INTO Orders (
CustomerID
) VALUES (
'SOCOM'
);
COMMIT;
```

Hình 3.25 trình bày giao dịch này, theo sau với hai phát biểu SELECT trả về hai hàng mới INSERT



Transaction tiếp theo bao gồm phát biểu INSERT tương tự, ngoại trừ lúc này transaction là phục hồi sử dụng phát biểu
ROLLBACK.

```
BEGIN TRANSACTION;
INSERT INTO Customers (
CustomerID, CompanyName
) VALUES (
'SYCOM', 'Steve Yellow Company'
);
INSERT INTO Orders (
CustomerID
) VALUES (
```

```
'SYCOM'
);
ROLLBACK;
```

Bởi vì transaction là phục hồi (rolled back) , hai hàng thêm vào bởi hai phát biểu INSERT được cuộn lui trở lại(hủy).
Bạn nên kiểm tra lỗi của một transaction trước khi quyết định thực hiện một COMMIT hay ROLLBACK bởi vì những lỗi không phải luôn luôn đình chỉ việc thực thi hàng tiếp theo. Để làm điều này trong SQL Server, bạn sử dụng hàm @@ERROR. Hàmnày trả lại số khôngbất cứ khi nào một phát biểu được thự thi và không gây ra lỗi. nếu @@ERROR trả về một giá trị khác không, bạn biết đã sảy ra một lỗi. Nếu @@ERROR trả về 0, bạn thực hiện một COMMIT, ngược lại bạn thực hiện một ROLLBACK.

Bạn cũng có thể gán một tên cho transaction của bạn trong phát biểu BEGIN TRANSACTION. Điều này hữu ich như nó cho biết transaction nào bạn đang làm việc .

Thí dụ sau đây trình bày việc gán tên cho một transaction, cùng với sự sử dụng hàm @@ERROR để xác định nên thực hiện một COMMIT hay một ROLLBACK :

```
BEGIN TRANSACTION MyTransaction;
INSERT INTO Customers (
CustomerID, CompanyName
) VALUES (
'SYCOM', 'Steve Yellow Company'
);
INSERT INTO Orders (CustomerID
) VALUES (
'SYCOM'
);
IF @@Error = 0
COMMIT TRANSACTION MyTransaction;
ELSE
ROLLBACK TRANSACTION MyTransaction;
```

Chú ý : tên của transaction là MyTransaction, và tên này được sử dụng trong phát biểu COMMIT và ROLLBACK.

Chú thích:bạn sử dụng phát biểu IF để định điều kiện thực thi một phát biểu (câu lệnh SQL). Bạn sẽ học nhiều hơn về điều này ở chương 4, "Introduction to Transact-SQL Programming."

GIỚI THIỆU VỀ NHỮNG PHÁT BIỂU NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU:

Như đã đề cập trước, những phát biểu DDL cho phép bạn tạo những cấu trúc cơ sở dữ liệu như các bảng và các index .

TẠO MỘT BẢNG:

Bạn tạo một bảng sử dụng phát biểu CREATE TABLE. Thí dụ , hãy cho là bạn muốn lưu trữ những chi tiết cho một số người trong cơ sở dữ liệu. giả sử bạn muốn lưu trử họ, tên và ngày sinh của một người. hãy gọi bảng này là Persons.
Bạn cũng muốn nhận dạng duy nhất mỗi hàng trong bảng Persons sử dụng một ID dạng số, nó hoạt động như một khóa chính trong bảng. phát biểu CREATE TABLE sau đây tạo một bảng Persons :

```
CREATE TABLE Persons (
PersonID int CONSTRAINT PK_Persons PRIMARY KEY,
FirstName nvarchar(15) NOT NULL,
LastName nvarchar(15) NOT NULL,
```

```
    DateOfBirth datetime
);
```

Bạn sử dụng mệnh đề CONSTRAINT (ràng buộc) để hạn chế những giá trị lưu trữ trong bảng hay cột. bạn chú ý rằng mệnh đề CONSTRAINT được sử dụng để chỉ định khóa chính của bảng sử dụng từ khóa PRIMARY KEY. Khóa chính này là
Cột PersonID, và ràng buộc này có tên là PK_Persons. Cột khóa này có kiểu int, có nghĩa là nó lưu trữ những số intergers.
Mỗi hàng trong bảng Persons phải có một số duy nhất cho cột PersonsID.

Cột FirstName và LastName là những cột nvarchar có thể chứa đến 15 kí tự. cả hai cột này được định nghĩa sử dụng ràng buộc NOT NULL. NOT NULL chỉ định là bạn phải cung cấp một giá trị cho cột. theo mặc định là NULL, có nghĩa là bạn không cần phải cung cấp một giá trị cho cột.

Chú thích: những khóa chính luôn luôn đòi hỏi một giá trị, và do đó tuyệt đối không d8ược bỏ trống.

Cột DateOfBirth thuộc kiểu datetime, có nghĩa nó có thể dự trữ giá trị ngày và giờ, phút, giây. Cột này không được định nghĩa ràng buộc NOT NULL, do đó sẽ sữ dụng giá trị mặ định là NULL.

SỬA ĐỔI MỘT BẢNG:

Bạn thay đổi một bảng đang tồn tại sử dụng phát biểu ALTER TABLE. Bạn có thể thêm hoặc xóa một cột , thêm hoặc xóa một ràng buộc sử dụng phát biểu ALTER TABLE. Thí dụ, phát biểu ALTER TABLE sau đây thêm nột cột tên Address vào bảng Persons :

```
ALTER TABLE Persons
ADD Address nvarchar(50);
```

Cột Address là một nvarchar có thể chứ a trên 50 kí tự.

Thí dụ tiếp theo xóa cột Adrress từ bảng Persons:

```
ALTER TABLE Persons
DROP COLUMN Address;
```

Thí dụ tiếp theo thêm một cột tên EmployerID vào bảng Persons, ghi tên công ty mà một người làm việc:

```
ALTER TABLE Persons
ADD EmployerID nchar(5) CONSTRAINT FK_Persons_Customers REFERENCES
Customers(CustomerID);
```

Cột EmployerID là một khóa phụ cho cột CustomerID của bảng Customers. Ràng buộc này có tên FK_Persons_Customers.

XÓA MỘT BẢNG:

Bạn thêm một index(chỉ số) vào một bảng sử dụng phát biểu CREATE INDEX. Một index cho phép bạn . một index cho phép bạn tìm một hàng một cách nhanh chóng hơn khi bạn sử dụng index(chỉ số) trong mệnh đề WHERE.
Thí dụ , phát biểu CREATE INDEX dưới đây thêm một index vào cột LastName của bảng Persons.

```
    CREATE INDEX LastNameIndex
ON Persons(LastName);
```

Mẹo nhỏ : nếu bạn thường xuyên sử dụng một cột trong mệnh đề WHERE , bạn cần xem xet thêm một chỉ số (index) vào cột đó.

Nói chung, bạn cần phải tạo một chỉ số trên một cột chỉ khi bạn tìm thấy là bạn đang truy xuất một số ít hàng từ một bảng đang chứa rất nhiều hàng. Một kinh nghiệm tốt là một chỉ số sẽ hữu ích khi bạn chờ đợi bất kỳ truy vấn đơn nào truy xuất 10 phần trăm hoặc ít so với tổng số những hàng trong một bảng. điều này có nghiã là cột thích hợp cho một chỉ số (index) phải được sử dụng để lưu trữ một dãi rộng của những giá trị. Cột thích hợp nhất cho sư chỉ số hoá là một cột có một số duy nhất cho mỗi mẫu tin(cột koá chính), những cột không thích hợp cho sự chỉ số hoá là những cột chỉ chứa một dãy nhỏ những số mã ,thí dụ 1,2,3 hoặc 4. sự xem xét này áp dụng cho tất cả các kiểu dữ liệu không chỉ dành cho kiểu số.

Thộng thường ,DBA chịu trách nhiệm tạo ra những chỉ số, nhưng là một người phát triển ứng dụngbạn có lẽ đã biết nhiều về trỉnh ứng dụng hơn là DBA và có khả năng nhận biết cột nào làthích hợp nhất cho sự chỉ số hoá.

XOÁ MỘT INDEX:

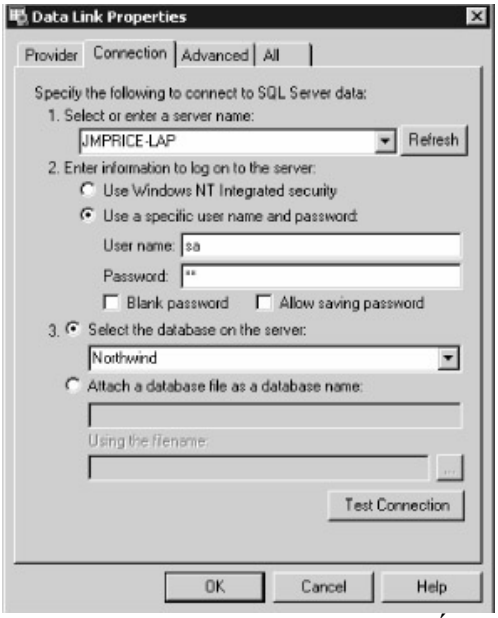
Bạn xoá một index trong một bảng sử dụng phát biểu DROP INDEX . thí dụ, phát biểu DROP INDEX sau xoá LastNameIndex từ bảng Persons.

```
DROP INDEX Persons.LastNameIndex;
```

Việc truy nhập một Cơ sở dữ liệu sử dụng Visual Studio .NET

Trình duyệt server của Visual Studio .NET cho phép bạn sử dụng một tập con của những đặc tính được chứa trong những thư mục cơ sở dữ liệu của Enterprise Manager. Đặc biệt, trình duyệt Server cho phép bạn xem, tạo ra, và soạn thảo những cơ sở dữ liệu, những sơ đồ cơ sở dữ liệu, những bảng, những view, những thủ tục lưu trữ, và những hàm do người dùng định ra. Trong mục này, bạn sẽ được giới thiệu về trình duyệt Server và một số chức năng của nó. Như bạn sẽ thấy, trình duyệt Server vận hành một cách tương tự với Enterprise Manager, nó đã được bao trùm trong Chương 2. Bởi vì sự giống nhau giữa trình duyệt Server và Enterprise Manager, Tôi sẽ chỉ tóm tắt bao trùm trình duyệt Server ở đây.

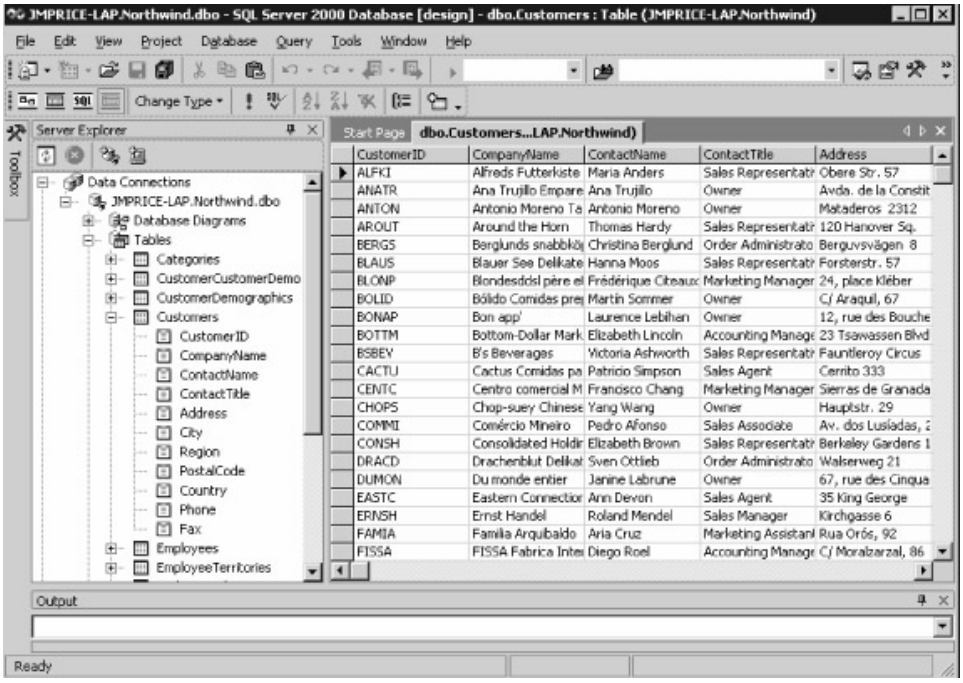
Bước đầu tiên của bạn là kết nối tới một cơ sở dữ liệu. Để làm điều này, bạn chon *Tools ➢ Connect To Database*. Việc này hiển thị hộp *Data Link Properties* . Hình 3.26 trình bày hộp thoại này với những mục thích hợp để kết nối tới cơ sở dữ liệu Northwind chạy trên máy tính JMPRICE- LAP.



Hình 3.26: việc nhập những chi tiết cơ sở dữ liệu sử dụng hộp thoại Data Link Properties

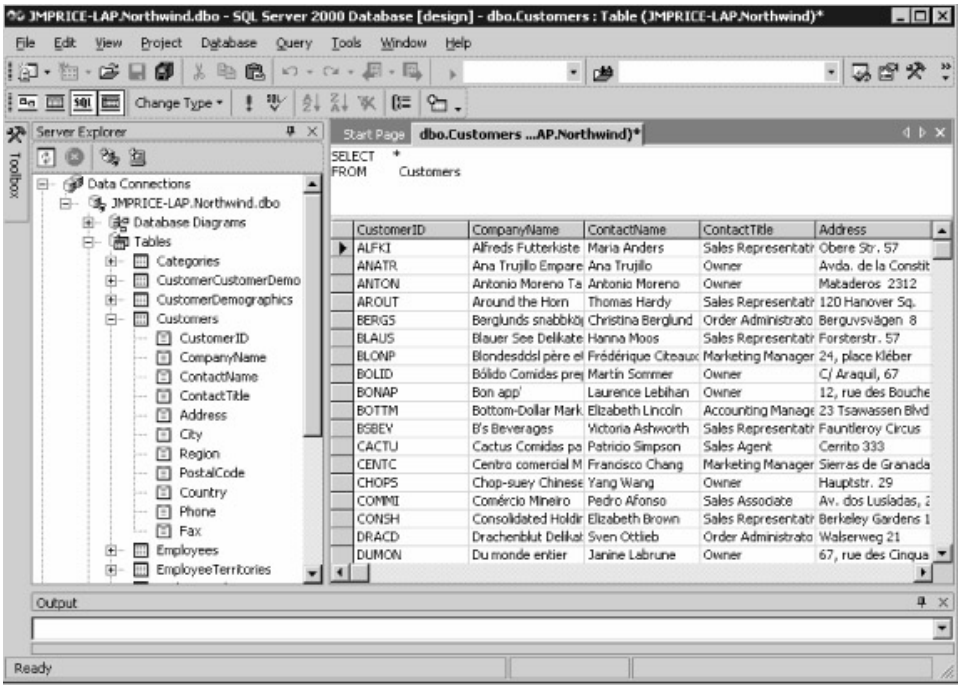
Một khi bạn đã nhập vào những chi tiết cơ sở dữ liệu của bạn, bước thứ hai của bạn là kích nút Test Connection để xác minh những chi tiết kết nối cơ sở dữ liệu. Kích nút Ok một khi sự thử kết nối thành công.

Một khi bạn đã có kết nối tới cơ sở dữ liệu, bạn có thể xem những thứ như những bảng. Bạn cũng có thể truy xuất và sửa đổi những hàng trong những bảng. Bạn có thể khoan sâu xuống những bảng trong cơ sở dữ liệu bởi kích Add icon (thêm biểu tượng) trong cây trong trình duyệt Server, và Bạn có thể truy xuất những hàng từ một bảng bởi kích chuột phải trên bảng trong cây và chọn Retrieve Data From Table trong cửa sổ sổ ra. Hình 3.27 cho thấy những hàng từ bảng những khách hàng .



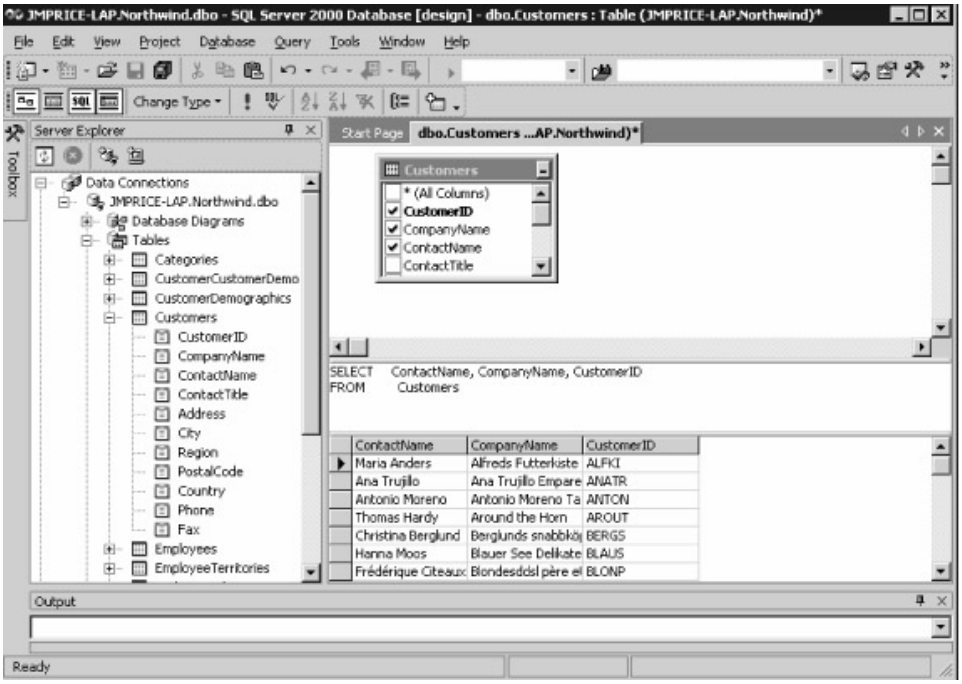
Hình 3.27: xem những hàng trong bảng những khách hàng sử dụng trình duyệt Server

Bạn có thể nhập những câu lệnh SQL bởi kích nút Show SQL Pane trong thanh công cụ, như trong Hình 3.28.



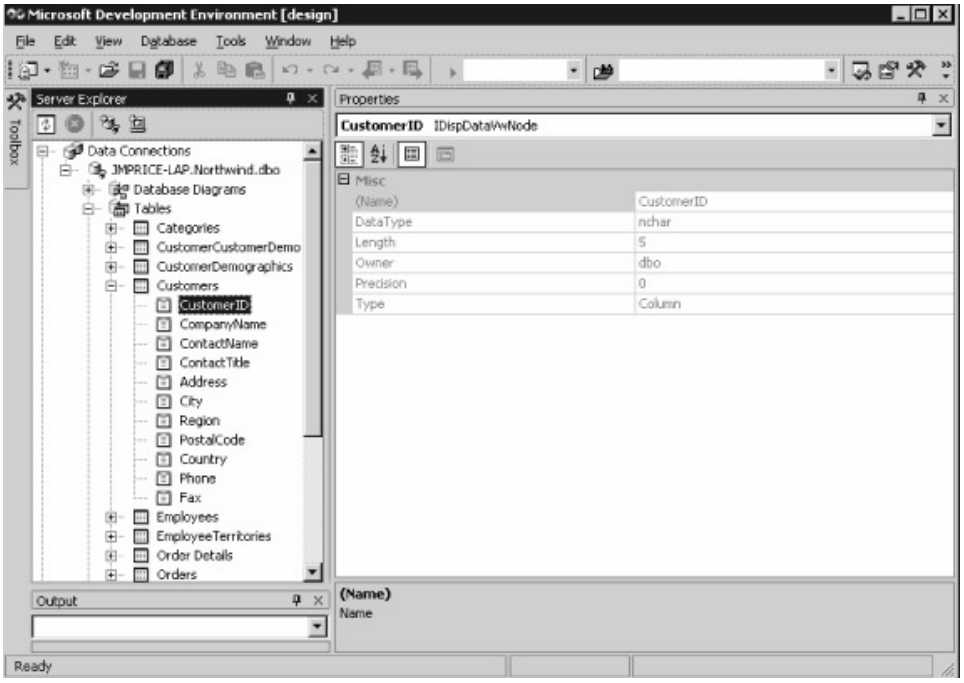
Hình 3.28: việc nhập một câu lệnh SQL

Bạn có thể xây dựng những câu lệnh SQL một cách trực quan bởi kích nút Show Diagram trong thanh công cụ và chọn những cột từ bảng, như trình bày trong Hình 3.29. bạn có thể thấy, Tôi đã chọn những cột ContactName, CompanyName, và CustomerID từ bảng những khách hàng.



Hình 3.29: xây dựng một câu lệnh SQL cách trực quan

Bạn có thể xem những thuộc tính của một cột trong một bảng bởi kích chuột phải trên cột và lựa chọn Properties từ cửa sổ bung ra. Hình 3.30 cho thấy những thuộc tính của cột CustomerID của bảng những khách hàng .



Hình 3.30: những thuộc tính của những cột CustomerID

Bạn chỉ mới sơ lược qua Server Explorer (trình duyệt Server) trong mục này. Nếu bạn có VS .NET, bạn cần phải thoải mái thử nghiệm trình duyệt Server này- đó là cách tốt nhất để học về nó.

Tóm lược

Trong chương này, bạn đã học sử dụng SQL (sự xuy diễn rõ ràng) để truy nhập một cơ sở dữ liệu như thế nào. SQL là ngôn ngữ tiêu chuẩn để truy cập những cơ sở dữ liệu quan hệ. Với SQL, Bạn nói với cơ sở dữ liệu dữ liệu nào bạn muốn truy cập, và phần mềm cơ sở dữ liệu tính tóan chính xác cách để lấy dữ liệu này. Bạn có thể nhập và chạy những câu lệnh SQL tiếp cận một cơ sở dữ liệu máy chủ phục vụ SQL sử dụng công cụ bộ phân tích truy vấn (Query Analyzer) .

Có hai kiểu chính thuộc những câu lệnh SQL: những phát biểu ngôn ngữ xử lý dữ liệu (DML) "Data Manipulation Language", và những phát biểu ngôn ngữ định nghĩa dữ liệu (DDL) "Data Definition Language". Những phát biểu DML cho phép bạn truy xuất, thêm, sửa đổi, và xóa những hàng được cất giữ trong cơ sở dữ liệu. Những phát biểu DDL cho phép bạn tạo ra những cấu trúc cơ sở dữ liệu như những bảng và những chỉ số.

Bạn sử dụng một phát biểu SELECT SQL để truy xuất những hàng, một phát biểu INSERT để thêm những hàng, một phát biểu UPDATE để điều chỉnh những hàng, và một phát biểu xóa để loại bỏ những hàng.

Bạn khảo sát trình công cụ duyệt Server Visual Studio .NET'. nó cho phép bạn kết nối tới một cơ sở dữ liệu. Trình duyệt Server chứa một tập con của chức năng được đề xuất bởi Enterprise Manager.

Trong chương kế tiếp, bạn sẽ được giới thiệu lập trình Transact-SQL.

Chương 4: Giới thiệu về lập trình giao dịch- SQL (Transact-SQL)

Tổng quan

Transact-SQL (Giao dịch- SQL) là sự thực thi đầy đủ của Microsoft của SQL, Và nó chứa những cấu trúc lập trình bổ sung . (Nó thường được viết tắt là T- SQL, một quy ước mà bạn sẽ thấy trong chương này.) T- SQL cho phép bạn viết những chương trình chứa những câu lệnh SQL, cùng với tiêu chuẩn lập trình cấu trúc như những biến, lôgic có điều kiện, những vòng lặp, những thủ tục, và những hàm.

Đặc trưng trong chương này:

- Những nền tảng của Transact- SQL
- Sử dụng những hàm
- Tạo những hàm do người dùng định nghĩa
- Giới thiệu về những thủ tục lưu trữ
- Giới thiệu về những trigơ (triggers)

Những nền tảng của Transact-SQL (Giao dịch- SQL)

Trong mục này, bạn sẽ học một số yếu tố cần thiết về lập trình xây dựng sẵn có trong T- SQL. Đặc biệt, bạn sẽ thấy cách sử dụng những biến, những chú thích, những lôgic có điều kiện . Bạn cũng sẽ thấy cách sử dụng một số phát biểu cho phép bạn thực hiện nhảy và những vòng lặp như thế nào. Cuối cùng, bạn sẽ khảo sát những con trỏ, cho phép bạn xử lý những hàng được trả lại từ cơ sở dữ liệu một tại một thời điểm.

Chúng ta hãy bắt đầu với việc khảo sát những biến.

Sử dụng những biến (Variables)

Một biến cho phép bạn cất giữ một giá trị trong bộ nhớ của một máy tính. Mỗi biến có một kiểu chỉ định loại của giá trị sẽ được cất giữ trong biến này. Bạn có thể sử dụng bất kỳ những kiểu được trình bày trước đó trong bảng 2.3 trong chương 2 " Giới thiệu về cơ sở dữ liệu ."

Bạn khai báo một biến sử dụng phát biểu DECLARE, theo sau là tên biến và kiểu của nó. Bạn đặt một ký tự (@) trước tên của biến . Cú pháp sau đây minh họa sự sử dụng phát biểu DECLARE :

```
DECLARE @name type
```

Với *name* là tên của biến , và *type* là kiểu củabiến .

Chẳng hạn, những phát biểu sau đây khai báo hai biến có tên MyProductName Và MyProductID:

```
DECLARE @MyProductName nvarchar(40)
DECLARE @MyProductID int
```

Như bạn có thể thấy, MyProductName thuộc kiểu nvarchar, và MyProductID thuộc kiểu int.

Bạn có thể đặt nhiều khai báo biến trên cùng một hàng. Chẳng hạn:

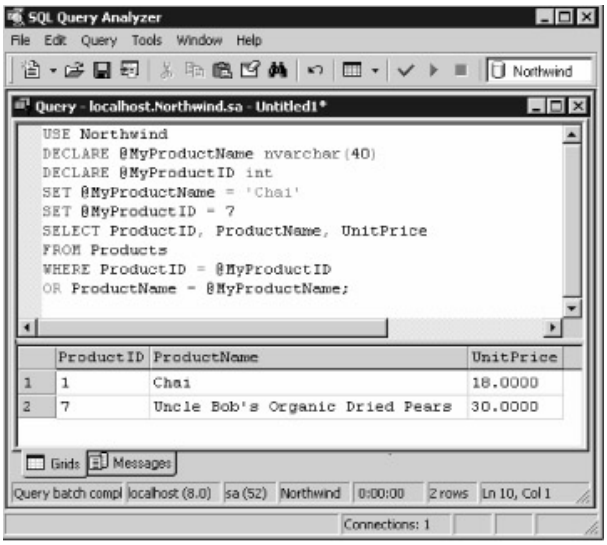
```
DECLARE @MyProductName nvarchar(40), @MyProductID int
```

Những biến thoạt tiên được gán tới null. Bạn gán một giá trị của biến sử dụng sự phát biểu SET. Chẳng hạn, những phát biểu sau đây đặt MyProductName là chai và MyProductID là 7:

Phát biểu SELECT sau đây sử dụng những biến này trong mệnh đề WHERE:

```
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE ProductID = @MyProductID
OR ProductName = @MyProductName;
```

Bạn có thể thực thi T- SQL sử dụng Query Analyzer, và hình 4.1 cho thấy đầu ra từ những ví dụ trình bày trong mục này.



Hình 4.1: Thực thi T- SQL sử dụng Query Analyzer

Sử dụng những chú thích

Bạn thêm những chú thích để giải thích mã của bạn, làm chúng dễ hiểu hơn cho cả chính bạn lẫn những lập trình viên khác. Bạn có thể đã nghĩ rằng bạn hiểu mã của mình dễ dàng, nhưng sáu tháng sau , khi bạn trở lại nó để bảo trì , Bạn có lẽ đã quên những điều phức tạp do chính bạn hình thành ! Điểm chính là bạn cần phải thêm những chú thích vào mã của bạn để giúp bạn hiểu dễ dàng hơn, nhưng đừng nghĩ rằng bạn phải chú thích mọi hàng. nên sử dụng những chú thích cách thận trọng.

Bạn cần đánh dấu những chú thích của bạn với những đặc trưng riêng sao cho máy chủ phục vụ SQL lờ đi chúng và không sử lý tới .

Có hai kiểu chú thích : hàng đơn và nhiều hàng. Một chú thích hàng đơn sử dụng hai dấu trừ (--) và có thể trải trên chỉ một hàng, như trình bày dưới đây:

```
-- A single-line comment may only span one line.
```

-- Báo SQL Server lờ đi mọi thứ cho tới kết thúc của hàng này.

Một chú thích nhiều hàng bắt đầu với một dấu (/*) mở đầu chú thích và những kết thúc với một dấu (*/) kết thúc chú thích:

```
/* A multi-line comment
   may span more than one
   line. */
```

Dấu /* báo SQL Server lờ đi mọi thứ trong dòng này cho tới dấu */ kế tiếp, không có vấn đề gì về số lương hàng chú thích được bao trong cặp dấu này. Nếu bạn sử dụng những chú thích hàng đơn trong ví dụ này, bạn phải thêm những ký tự -- ở đầu của mọi dòng chú thích.

Những chú thích nhiều hàng tất nhiên cũng có thể dùng cho chỉ một hàng

```
/* Another comment */
```

Sử dụng Lôgic có điều kiện

Lôgic có điều kiện cho phép bạn thực thi những nhánh khác nhau của mã dựa trên giá trị Boolean của một biểu thức đã cho là true hay false . Chẳng hạn, bạn có thể muốn kiểm tra nếu một điều kiện lỗi là true thì trình bày một thông báo. Bạn sử dụng IF và tùy chọn những từ khóa ELSE để thực hiện lôgic có điều kiện. Cú pháp sau đây minh họa sự sử dụng lôgic có điều kiện

```
IF condition
  statement1
[ELSE
    statement2]
```

Với condition là một biểu thức Đại số Boole nó đó định giá tới true hay false. Nếu điều kiện là true, thì statement1 được thực hiện, trường hợp khác statement2 được thực hiện.

Ghi nhớ: Bạn có thể thay thế một phát biểu đơn với nhiều phát biểu bằng cách sắp xếp những phát biểu này bên trong những phát biểu BEGIN và END. Quy tắc này ứng dụng cho tất cả cấu trúc lập trình T- SQL.

Cú pháp sau đây cho thấy sự thay thế những phát biểu đơn với một khối những sự phát biểu được đặt bên trong BEGIN và END.

```
IF condition
BEGIN
  statements1
END
ELSE
BEGIN
  statements2
END
```

Với statements1 và statements2 là những phát biểu. Bạn cũng có thể sử dụng một phát biểu ELSE tùy chọn để thực thi một nhánh mã khác nếu điều kiện là false.

Ghi chú:

Bạn có thể lồng những phát biểu IF vào nhau tới độ sâu bất kỳ .

Ví dụ sau đây trình bày những cột ProductID, ProductName, và UnitPrice cho bất kỳ hàng nào từ bảng Products có một UnitPrice nhỏ hơn 5 USD. Bạn chú ý sự sử dụng lệnh in (PRINT) để gửi ra một hàng trong ví dụ này.

```
IF (SELECT COUNT(*) FROM Products WHERE UnitPrice < 5) > 0
BEGIN
```

```
  PRINT 'The following products have a UnitPrice of less than $5:'
  SELECT ProductID, ProductName, UnitPrice
  FROM Products
  WHERE UnitPrice < 5
END
ELSE
BEGIN
  PRINT 'There are no products that have a UnitPrice of less than $5'
END
```

Sử dụng những phát biểu CASE

Bạn sử dụng phát biểu CASE để so sánh một giá trị với một danh sách những giá trị và thực hiện một hoặc nhiều phát biểu khi một giá trị phù hợp được tìm thấy. Chẳng hạn, phát biểu CASE sau đây trả lại Massachusetts

```
CASE 'MA'
  WHEN 'CA' THEN 'California'
  WHEN 'MA' THEN 'Massachusetts'
  WHEN 'NY' THEN 'New York'
END
```

Ví dụ kế tiếp sử dụng một phát biểu SELECT để truy xuất giá trị Massachusetts được trả lại bởi phát biểu CASE:

```
DECLARE @State nchar(2)
SET @State = 'MA'
DECLARE @StateName nvarchar(15)
SELECT CASE @State
  WHEN 'CA' THEN 'California'
  WHEN 'MA' THEN 'Massachusetts'
  WHEN 'NY' THEN 'New York'
END
```

Bạn có thể cất giữ giá trị được truy xuất bởi phát biểu SELECT trong một biến, như trong ví dụ kế tiếp

```
DECLARE @State nchar(2)
SET @State = 'MA'
DECLARE @StateName nvarchar(15)
SELECT @StateName =
  CASE @State
    WHEN 'CA' THEN 'California'
    WHEN 'MA' THEN 'Massachusetts'
    WHEN 'NY' THEN 'New York'
  END
PRINT @StateName
```

Đầu ra từ ví dụ này như sau:

```
Massachusetts
```

Bạn cũng có thể so sánh một giá trị cột trong một phát biểu CASE. Chẳng hạn:

```
SELECT Price =
CASE
```

```
  WHEN UnitPrice IS NULL THEN 'Unknown'
  WHEN UnitPrice < 10 THEN 'Less than $10'
  WHEN UnitPrice = 10 THEN '$10'
  ELSE 'Greater than $10'
END
FROM Products
```

Bạn chú ý: từ ví dụ này bạn cũng có thể cung cấp một điều kiện ELSE catchall trong một phát biểu CASE.

Sử dụng những vòng lặp "WHILE"

Bạn sử dụng một vòng lặp "WHILE" để chạy một hoặc nhiều phát biểu nhiều lần. Một vòng lặp "WHILE" chạy cho đến khi một điều kiện chỉ định chuyển giá trị thành false. Cú pháp cho một vòng lặp WHILE như sau:

```
WHILE condition
  statement
```

Ví dụ sau đây cho thấy một vòng lặp "WHILE"

```
DECLARE @count int
SET @count = 5
WHILE (@count > 0)
BEGIN
  PRINT 'count = ' + CONVERT(nvarchar, @count)
  SET @count = @count -1
END
```

Vòng lặp này được chạy cho đến khi biến đếm count trở thành 0, và đầu ra từ mã này như sau:

```
count = 5
count = 4
count = 3
count = 2
count = 1
```

Hàm *CONVERT()* (chuyển đổi) được dùng để chuyển đổi một giá trị từ kiểu này sang kiểu khác. Chẳng hạn, *CONVERT (nvarchar, @count)* chuyển đổi biến đếm count tới kiểu nvarchar, rồi nó có thể được sử dụng với lệnh PRINT (in).

Phát biểu CONTINUE (tiếp tục)

Bạn sử dụng phát biểu CONTINUE để nhảy tới lần lặp tiếp theo của một vòng lặp "WHILE" bỏ qua qua bất kỳ mã còn lại nào trong lần lặp hiện thời. phát biểu CONTINUE gây ra sự thực hiện nhảy trở lại điểm khởi đầu của lần lặp kế tiếp.

Ví dụ sau đây cho thấy một vòng lặp "WHILE" sự sử dụng phát biểu CONTINUE để bắt đầu lần lặp tiếp theo của vòng lặp nếu biến đếm count bằng 2:

```
DECLARE @count int
SET @count = 5
WHILE (@count > 0)
BEGIN
  PRINT 'count = ' + CONVERT(nvarchar, @count)
  SET @count = @count -1
  IF (@count = 2)
  BEGIN
```

```
    SET @count = @count -1
    CONTINUE
  END
END
```

Đầu ra từ mã này như sau:

```
count = 5
count = 4
count = 3
count = 1
```

Bạn chú ý kết quả không hiển thị biến đếm count = 2 . Đây là vì phát biểu CONTINUE thực thi sự nhảy qua lần lặp này.

Phát biểu BREAK

Bạn sử dụng phát biểu BREAK để kết thúc một vòng lặp "WHILE" ngay lập tức. Phát biểu BREAK gây ra sự thực hiện nhảy khỏi vòng lặp và tiếp tục thực hiện bất kỳ phát biểu nào sau vòng lặp này.

Ví dụ sau đây cho thấy một vòng lặp "WHILE" sử dụng phát biểu BREAK để kết thúc vòng lặp khi biến đếm count bằng 2

```
DECLARE @count int
SET @count = 5
WHILE (@count > 0)
BEGIN
  PRINT 'count = ' + CONVERT(nvarchar, @count)
  SET @count = @count -1
  IF (@count = 2)
  BEGIN
    BREAK
  END
END
```

Đầu ra từ mã này như sau:

```
count = 5
count = 4
count = 3
```

Sử dụng những nhãn và Phát biểu GOTO

Bạn sử dụng phát biểu GOTO để nhảy tới một nhãn được chỉ định trong mã của bạn; bạn sử dụng một nhãn để xác định một phát biểu trong mã của bạn. Bạn phải định nghĩa nhãn trước khi phát hành GOTO tới nhãn này. Trước khi tôi trình bày những chi tiết của phát biểu GOTO, bạn cần phải ý thức được sự sử dụng nó được coi là một thực hành lập trình nghèo nàn, và bạn cần phải tránh sử dụng nó nếu có thể được.Thông thường luôn có cấu trúc mã để bạn không cần sử dụng phát biểu GOTO. Như đã nói, tôi bao gồm nó trong chương này chỉ có tính chất bổ sung.

Như đã được đề cập, phát biểu GOTO yêu cầu bạn tạo ra một nhãn trong chương trình của bạn. Bạn làm điều này bằng cách đặt một định danh chứa tên nhãn trong mã của bạn, theo sau là một dấu hai chấm (:) . Ví dụ sau đây tạo ra một nhãn có tên myLabel:

```
myLabel:
```

Rồi bạn có thể sử dụng phát biểu GOTO để nhảy tới nhãn này, chẳng hạn:

```
GOTO myLabel
```

Ví dụ sau đây cho thấy sự sử dụng một nhãn và phát biểu GOTO:

```
DECLARE @count int
SET @count = 5
myLabel:
PRINT 'count = ' + CONVERT(nvarchar, @count)
SET @count = @count -1
IF (@count > 0)
BEGIN
  GOTO myLabel
END
```

Đầu ra từ mã này như sau:

```
count = 5
count = 4
count = 3
count = 2
count = 1
```

Sử dụng những phát biểu RETURN

Bạn sử dụng phát biểu RETURN để thoát ra khỏi một thủ tục lưu trữ hay nhóm những phát biểu. Bất kỳ phát biểu nào ở sau RETURN của bạn đều không được thực hiện. Bạn có thể cũng trả về một giá trị sử dụng phát biểu RETURN .

Cú pháp cho phát biểu RETURN như sau:

```
RETURN [int_expression]
```

Với ***int_expression*** là bất kỳ biểu thức nào có kết quả là một giá trị ***int***.

Ghi chú:

Bạn có thể trả về một giá trị chỉ khi sử dụng phát biểu RETURN với một thủ tục lưu trữ. Bạn sẽ thấy một ví dụ về điều này sau trong mục "Giới thiệu về những thủ tục lưu trữ ".

Ví dụ sau đây cho thấy sự sử dụng phát biểu RETURN:

```
DECLARE @count int
SET @count = 5
WHILE (@count > 0)
BEGIN
  PRINT 'count = ' + CONVERT(nvarchar, @count)
  SET @count = @count -1
  IF (@count = 2)
  BEGIN
    RETURN
  END
END
```

Đầu ra từ mã này như sau:

```
count = 5
count = 4
count = 3
```

Sử dụng những phát biểu WAITFOR

Có những lần khi bạn muốn chương trình của bạn tạm dừng trước khi chạy mã nào đó để thực hiện một tác vụ cụ thể, như chạy một đợt chương trình vào ban đêm để cập nhật những bản ghi khách hàng. Bạn sử dụng sự phát biểu WAITFOR để chỉ rõ một khoảng thời gian hay thời gian để đợi cho đến khi tiếp tục sự thực hiện của mã.

Cú pháp cho phát biểu WAITFOR như sau:

```
WAITFOR {DELAY 'time interval' | TIME 'actual time'}
```

Bạn có thể chỉ rõ khoảng thời gian đợi sử dụng từ khóa DELAY , hay bạn có thể chỉ rõ thời gian thực tế để đợi cho đến sử dụng từ khóa TIME. Bạn có thể chỉ rõ một khoảng thời gian hay một thời gian thực tế trong định dạng HH:MM:SS, với HH là giờ (trong định dạng 24), MM là phút, và SS là giây.

Đây là một số ví dụ:

- WAITFOR DELAY '00: 00: 05' đợi trong khoảng thời gian 5 giây.
- WAITFOR DELAY '23: 10: 25' đợi trong khoảng thời gian 23 giờ, 10 phút, và 25 giây.
- WAITFOR TIME '20: 15: 10' đợi cho đến 10 giây sau 10: 15 PM.

Những ví dụ sau đây in ra một thông báo sau 5 giây trôi qua

```
BEGIN
  WAITFOR DELAY '00:00:05'
  PRINT '5 seconds have elapsed'
END
```

Sử dụng những phát biểu RAISERROR

Bạn sử dụng sự phát biểu RAISERROR để phát sinh một thông báo lỗi. Bạn điển hình sẽ muốn làm điều này nếu một lỗi xuất hiện trong một trong số những thủ tục lưu trữ của bạn, Bạn sẽ thấy cách sử dụng sau đó trong mục " Tạo những thủ tục lưu trữ ."

Cú pháp được đơn giản hóa cho phát biểu RAISERROR như sau:

```
RAISERROR ({number | description} {, severity, state})
```

Với number là số đặc trưng cho lỗi, giá trị của nó nằm giữa 50,001 và 2,147,483,648. description là một thông báo mà không được vượt quá 400 ký tự. severity là mức độ của lỗi và phải nằm trong khoảng từ 0 đến 18 (18 là lỗi nặng nhất). state là một giá trị bất kỳ nằm giữa 1 và 127, và mô tả thông tin về trạng thái yêu cầu của lỗi .

Những ví dụ sau đây cho thấy sự sử dụng phát biểu RAISERROR:

```
RAISERROR (50001, 15, 1)
RAISERROR ('No row with that ProductID was found', 10, 1)
```

Sử dụng những con trỏ

Khi bạn thực hiện một phát biểu SELECT, tất cả những hàng được trả về trong một lần truy cập. Điều này có lẽ không luôn luôn thích hợp. Chẳng hạn, bạn có thể đã muốn sử lý hoạt động nào đó dựa vào những giá trị cột được truy xuất cho một hàng cụ thể. Để làm điều này, bạn có thể sử dụng một con trỏ (cursor) để xử lý những hàng được truy xuất từ cơ sở dữ liệu mỗi hàng một lần. Một con trỏ cho phép bạn bước qua những hàng được trả về bởi một phát biểu SELECT cụ thể.

Bạn theo những bước này khi sử dụng một con trỏ:

1. Khai báo những biến để cất giữ những giá trị cột từ phát biểu SELECT.
2. Khai báo con trỏ (cursor), chỉ rõ phát biểu SELECT của bạn.
3. Mở con trỏ (cursor) các bạn.
4. Nạp những hàng từ con trỏ của bạn về.
5. Đóng con trỏ của bạn.

Bạn sẽ học những chi tiết của những bước này trong những mục sau đây.

Bước 1: Khai báo những biến để lưu trữ giá trị cột từ phát biểu SELECT

Những biến này phải thích hợp với những kiểu cột của những hàng được truy xuất. Chẳng hạn, bạn sẽ muốn sử dụng một biến int để lưu trữ giá trị từ một cột int , vân vân.

Ví dụ sau đây khai báo ba biến để lưu trữ những cột ProductID, ProductName, và UnitPrice từ bảng Products:

```
DECLARE @MyProductID int
DECLARE @MyProductName nvarchar(40)
DECLARE @MyUnitPrice money
```

Bước 2: Khai báo Con trỏ (Cursor)

Một khai báo con trỏ gồm có một tên mà bạn gán tới con trỏ và phát biểu SELECT mà bạn muốn thực hiện. Phát biểu SELECT này không thật sự được chạy cho đến khi bạn mở con trỏ (cursor). Bạn khai báo con trỏ của bạn sử dụng phát biểu DECLARE .

Ví dụ sau đây khai báo một con trỏ có tên ProductCursor với một phát biểu SELECT mà truy xuất những cột ProductID, ProductName, và UnitPrice cho 10 sản phẩm đầu tiên từ bảng Products:

```
DECLARE ProductCursor CURSOR FOR
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE ProductID <= 10
```

Bước 3: Mở con trỏ (Cursor)

Bây là lúc mở con trỏ của bạn, nó chạy phát biểu SELECT được định nghĩa trước đó trong phát biểu DECLARE . Bạn mở một con trỏ sử dụng phát biểu OPEN. Ví dụ sau đây mở ProductCursor, và do đó cũng chạy phát biểu SELECT để truy xuất những hàng từ bảng Products:

```
OPEN ProductCursor
```

Bước 4: Tải về những hàng từ Con trỏ

Bây giờ bạn phải đọc mỗi hàng từ con trỏ của bạn. Để làm điều này, bạn sự sử dụng phát biểu FETCH . Con trỏ của bạn có thể chứa nhiều hàng, và do đó một vòng lặp "WHILE" được đòi hỏi để đọc lần lượt mỗi hàng. Để xác định khi vòng lặp kết thúc, bạn có thể sử dụng hàm @@FETCH_STATUS. Hàm này trả lại một trong số những giá trị có thể xảy ra được trình bày trong Bảng 4.1.

Bảng 4.1: những giá trị trả về từ hàm @@FETCH_STATUS

Giá trị	Mô tả
0	Phát biểu FETCH trả về một hàng một cách thành công .
-1	Phát biểu FETCH bị hỏng hay hàng yêu cầu bên ngoài tập hợp kết quả trả về.
-2	Hàng được tải về không hiện hữu.

Ví dụ sau đây trình bày một vòng lặp đọc từng hàng đến từ ProductCursor:

```
FETCH NEXT FROM ProductCursor
INTO @MyProductID, @MyProductname, @MyUnitPrice
PRINT '@MyProductID = ' + CONVERT(nvarchar, @MyProductID)
PRINT '@MyProductName = ' + CONVERT(nvarchar, @MyProductName)
PRINT '@MyUnitPrice = ' + CONVERT(nvarchar, @MyUnitPrice)
WHILE @@FETCH_STATUS = 0
```

```
BEGIN
  FETCH NEXT FROM ProductCursor
  INTO @MyProductID, @MyProductname, @MyUnitPrice
  PRINT '@MyProductID = ' + CONVERT(nvarchar, @MyProductID)
  PRINT '@MyProductName = ' + CONVERT(nvarchar, @MyProductName)
  PRINT '@MyUnitPrice = ' + CONVERT(nvarchar, @MyUnitPrice)
END
```

Bạn chú ý : điều kiện @@FETCH_STATUS = 0 được sử dụng trong vòng lặp "WHILE" để kiểm tra phát biểu FETCH có trả về một hàng cách thành công hay không. Khi điều kiện này không còn đúng nữa (false), vòng lặp kết thúc.

Mẹo nhỏ:

Bạn có thể lấy số lượng của hàng được lưu trữ trong một con trỏ sử dụng hàm @@CURSOR_ROWS. Bạn sẽ học nhiều hơn về những hàm sau trong mục "Sử dụng những hàm ".

Bước 5: Đóng con trỏ

Đóng con trỏ của bạn sử dụng phát biểu CLOSE. Những ví dụ sau đây kết thúc ProductCursor:

```
CLOSE ProductCursor
```

Bạn cũng cần phải loại bỏ sự tham chiếu tới con trỏ của bạn sử dụng phát biểu DEALLOCATE . Việc này giải phóng những tài nguyên hệ thống được dùng bởi con trỏ của bạn. Ví dụ sau đây loại bỏ sự tham chiếu tới ProductCursor sử dụng phát biểu DEALLOCATE (thu hồi phân bổ).

```
DEALLOCATE ProductCursor
```

Mục sau đây cho thấy một script ví dụ đầy đủ mà bạn có thể chạy sử dụng Query Analyzer. Script này chứa tất cả năm bước về sử dụng một con trỏ.

Ví dụ đầy đủ: ProductCursor.sql

Danh sách 4.1 trình bày script ProductCursor.sql . Bạn có thể tải file này vào trong Query Analyzer và chạy thử nó

Danh sách 4.1: Sử dụng những con trỏ

```
/*
  ProductCursor.sql uses a cursor to display
  the ProductID, ProductName, and UnitPrice columns
  from the Products table
*/

USE Northwind

-- step 1: declare the variables
DECLARE @MyProductID int
DECLARE @MyProductName nvarchar(40)
DECLARE @MyUnitPrice money

-- step 2: declare the cursor
DECLARE ProductCursor CURSOR FOR
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE ProductID <= 10

-- step 3: open the cursor
OPEN ProductCursor
```

```
-- step 4: fetch the rows from the cursor
FETCH NEXT FROM ProductCursor
INTO @MyProductID, @MyProductname, @MyUnitPrice
PRINT '@MyProductID = ' + CONVERT(nvarchar, @MyProductID)
PRINT '@MyProductName = ' + CONVERT(nvarchar, @MyProductName)
PRINT '@MyUnitPrice = ' + CONVERT(nvarchar, @MyUnitPrice)
WHILE @@FETCH_STATUS = 0
BEGIN
  FETCH NEXT FROM ProductCursor
  INTO @MyProductID, @MyProductName, @MyUnitPrice
  PRINT '@MyProductID = ' + CONVERT(nvarchar, @MyProductID)
  PRINT '@MyProductName = ' + CONVERT(nvarchar, @MyProductName)
  PRINT '@MyUnitPrice = ' + CONVERT(nvarchar, @MyUnitPrice)
END

-- step 5: close the cursor
CLOSE ProductCursor
DEALLOCATE ProductCursor
```

Đầu ra cho hai hàng đầu tiên được đọc bởi con trỏ như sau:

```
@MyProductID = 1
@MyProductName = Chai
@MyUnitPrice = 18.00
@MyProductID = 2
@MyProductName = Chang
@MyUnitPrice = 19.00
```

Sử dụng những hàm

SQL Server cung cấp một số hàm bạn có thể sử dụng để lấy những giá trị từ cơ sở dữ liệu. Chẳng hạn, bạn có thể sử dụng hàm count() để lấy số lượng của hàng có trong một bảng. Những hàm được tách ra vào trong những phạm trù được trình bày trong Bảng 4.2.

Bảng 4.2: những hàm

Phạm trù Chức năng	Mô tả
Aggregate	Trả về thông tin dựa vào một hoặc nhiều hàng trong một bảng.
Mathematical	Thực hiện những sự tính toán.
String	Thực hiện những thao tác chuỗi.
Date and time	Làm việc với ngày tháng và những giờ
System	Trả về thông tin trên SQLServer.
Configuration	Trả về thông tin trên cấu hình của Server.
Cursor	Trả về thông tin trên những con trỏ.
Metadata	Trả về thông tin trên cơ sở dữ liệu và nhiều tiết mục cơ sở dữ liệu, như những bảng.
Security	Trả lại thông tin trên những người sử dụng cơ sở dữ liệu và những vai trò.
System statistical	Trả về thông tin thống kê trên SQL Server.

Text and image	Thực hiện những thao tác văn bản và ảnh.

Bạn sẽ học về năm hàm đầu tiên trong những mục sau đây. Những hàm khác bên ngoài phạm vi của sách này, vì chúng thuộc về sự quan tâm chính yếu của những người quản trị cơ sở dữ liệu. Bạn có thể học về những chức năng đó trong những tài liệu sách trực tuyến SQL Server.

Sử dụng những chức năng Tổng thể

Trước đó, Bạn đã thấy cách sử dụng hàm tổng thể COUNT() để lấy số lượng hàng. COUNT() và một số hàm tổng thể khác bạn có thể sử dụng với SQL Srver được liệt kê trong Bảng 4.3. Biểu thức bạn có thể chuyển cho những hàm tổng thể điển hình là một cột đơn, nhưng nó cũng có thể là một trường được tính toán. ALL có nghĩa là hàm được ứng dụng vào tất cả những giá trị cột, trong khi DISTINCT chỉ có nghĩa là hàm chỉ ứng dụng tới những giá trị duy nhất. ALL là mặc định.

Bảng 4.3: những hàm tổng thể

HÀM	**MÔ TẢ**
AVG([ALL | DISTINCT] *expression*)	Trả lại giá trị trung bình cộng của những giá trị trong một nhóm.
COUNT([ALL | DISTINCT] *expression*] | *)	Trả lại số lượng hàng trong một nhóm. COUNT() trả lại một giá trị kiểu dữ liệu int.
COUNT_BIG([ALL | DISTINCT] *expression*} | *)	Trả lại số lượng giá trị trong một nhóm. COUNT_BIG() trả về một giá trị kiểu dữ liệu bigint
MAX([ALL | DISTINCT] *expression*)	Trả lại giá trị lớn nhất.
MIN([ALL | DISTINCT] *expression*)	Trả lại giá trị nhỏ nhất.
SUM([ALL | DISTINCT] *expression*)	Trả lại tổng của bất kỳ giá trị không null nào. SUM() chỉ có thể được sử dụng với những biểu thức số.
STDEV(*expression*)	Trả lại độ lệch tiêu chuẩn cho tất cả những giá trị.
STDEVP(*expression*)	Trả lại độ lệch tiêu chuẩn cho tập hợp của tất cả những giá trị.
VAR(*expression*)	Trả lại sự chênh lệch cho tất cả những giá trị.
VARP(*expression*)	Trả lại sự chênh lệch cho tập hợp của tất cả những giá trị.

Chúng ta hãy xem xét những ví dụ sử dụng một số những hàm tổng thể.

Bạn sử dụng hàm AVG() để có trị bình quân. Chẳng hạn, phát biểu sau đây lấy trị trung bình của cột UnitPrice của bảng những sản phẩm sử dụng hàm AVG() :

```
SELECT AVG(UnitPrice)
FROM Products;
```

Ví dụ này trả về 28,8663. Vì ALL là mặc định được dùng với những hàm, ví dụ này sử dụng mọi hàng trong bảng những sản phẩm khi thực hiện sự tính toán. Nếu bạn muốn chỉ sử dụng những giá trị duy nhất trong sự tính toán, thì bạn sử dụng tùy chọn DISTINCT, như ví dụ sau đây

```
SELECT AVG(DISTINCT UnitPrice)
FROM Products;
```

Ví dụ này trả lại 31,4162, hơi cao hơn so với kết quả trước đây bởi vì chỉ những giá trị duy nhất (không trùng lặp) trong cột được sử dụng trong lần này.

Ngoài việc gởi một cột tới một hàm, bạn có thể cũng gởi qua một trường được tính toán. Chẳng hạn, phát biểu sau đây gởi qua trường được tính toán UnitPrice* 1.20 tới hàm AVG() :

```
SELECT AVG(UnitPrice * 1.20)
FROM Products;
```

Ví dụ này trả lại 34,639,636; trị trung bình sau khi những giá trị UnitPrice được tăng 20 phần trăm.

Bạn có thể giới hạn những hàng được chuyển cho một hàm sử dụng một mệnh đề WHERE . Chẳng hạn, phát biểu SELECT sau đây tính toán trị trung bình của UnitPrice cho những hàng với một CategoryID là 1

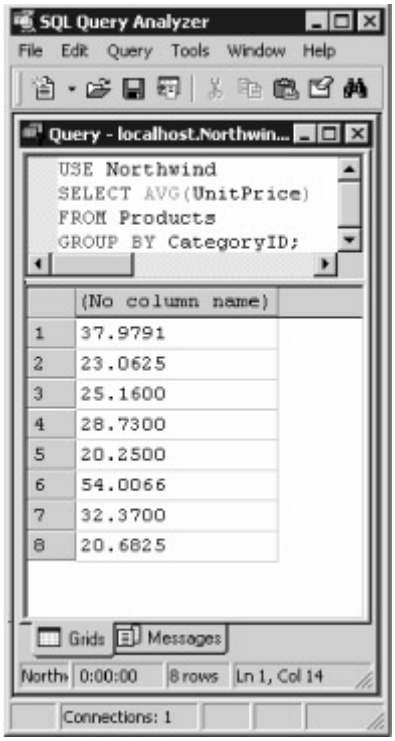
```
SELECT AVG(UnitPrice)
FROM Products
WHERE CategoryID = 1;
```

Ví dụ này trả về 37.9791.

Bạn có thể kết hợp một hàm với một mệnh đề GROUP BY để thực hiện một tính toán trên mỗi nhóm của những hàng. Chẳng hạn, phát biểu SELECT sau đây tính toán trị trung bình của cột UnitPrice cho mỗi khối của những hàng được nhóm lại bởi CategoryID:

```
SELECT AVG(UnitPrice)
FROM Products
GROUP BY CategoryID;
```

Hình 4.2 cho thấy những kết quả của phát biểu SELECT này.



Hình 4.2: Sử dụng hàm AVG() để tính toán trị bình quân của cột UnitPrice

Bạn cũng có thể cung cấp một mệnh đề HAVING để hạn chế những nhóm được dùng trong một phát biểu SELECT. Ví dụ, phát biểu sau đây thêm một mệnh đề HAVING vào ví dụ trước để trích ra những nhóm có trị trung bình lớn hơn 50:

```
SELECT AVG(UnitPrice)
FROM Products
GROUP BY CategoryID
HAVING AVG(UnitPrice) > 50;
```

Ví dụ này trả lại 54.0066.

Chúng ta hãy xem xét tại một số những hàm tổng thể khác. Bạn lấy tổng số lượng hàng sử dụng hàm COUNT() . Chẳng hạn, phát biểu sau đây lấy tổng số lượng hàng trong bảng những sản phẩm sử dụng hàm COUNT():

```
SELECT COUNT(*)
FROM Products;
```

Ví dụ này trả lại 77.

Bạn sử dụng những hàm MAX() và MIN() để lấy những giá trị cực đại và cực tiểu. Chẳng hạn, phát biểu sau đây sử dụng những hàm này để lấy giá trị cực đại và cực tiểu của cột UnitPrice:

```
SELECT MAX(UnitPrice), MIN(UnitPrice)
FROM Products;
```

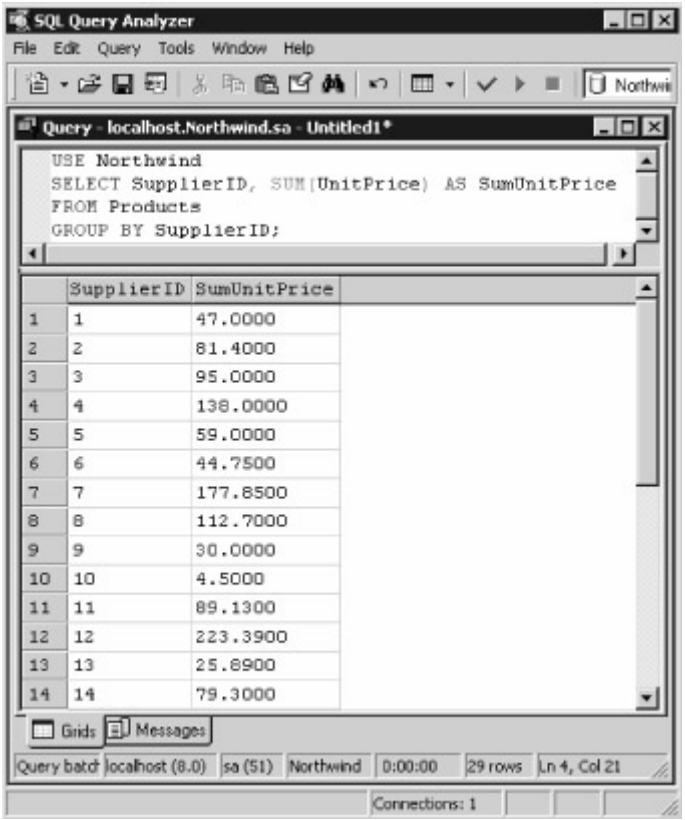
Ví dụ này trả lại 263.5000 và 2.5000 cho những giá trị cực đại và cực tiểu tương ứng.

Bạn sử dụng hàm SUM() để lấy tổng số của bất kỳ giá trị không null nào. Chẳng hạn, phát biểu sau đây lấy tổng của những giá trị cột UnitPrice cho mỗi nhóm của những hàng sử dụng hàm SUM().

```
SELECT SupplierID, SUM(UnitPrice) AS SumUnitPrice
FROM Products
GROUP BY SupplierID;
```

Mệnh đề GROUP BY của ví dụ này trả về một hàng cho mỗi nhóm của những hàng có những giá trị cột SupplierID giống nhau. Hàm SUM() tính tổng những giá trị cột UnitPrice cho tất cả những hàng bên trong mỗi nhóm và trả về một giá trị đơn. Chẳng hạn, SUM() trả về 47.0000 cho nhóm có SupplierID là 1. Đây là tổng của những giá trị cột UnitPrice cho tất cả những hàng có SupplierID là 1. Tương tự, SUM() trả lại 81.4000 nơi SupplierID là 2, vân vân. Mệnh đề AS trong ví dụ này gán tên cho những kết quả được trả về bởi hàm SUM() như một cột mới có tên SumUnitPrice.

Hình 4.3 cho thấy những kết quả của phát biểu SELECT này.



Hình 4.3: sử dụng gàm SUM() để tính toán tổng số của cột UnitPrice

Sử dụng những hàm Toán học

Những hàm toán học cho phép bạn thực hiện những thao tác số, như việc lấy giá trị tuyệt đối của một số.

Bảng 4.4 liệt kê những hàm toán học sẵn có trong SQL Server. Biểu thức bạn có thể chuyển cho những hàm toán học điển hình là một cột đơn hay giá trị, nhưng nó cũng có thể là một trường được tính toán.

Bảng 4.4: những hàm Toán học

HÀM	MÔ TẢ
ABS(*expression*)	Trả lại giá trị tuyệt đối của biểu thức. nó luôn luôn là một số dương.
ACOS(*expression*)	Trả lại arccosine của biểu thức.
ASIN(*expression*)	Trả lại arcsine của biểu thức.
ATAN(*expression*)	Trả lại arctangent của biểu thức.
ATN2(*expression1*, *expression2*)	Trả lại arctangent của góc giữa biểu thức1 và biểu thức2.
CEILING(*expression*)	Trả lại số nguyên nhỏ nhất lớn hơn hay bằng biểu thức.
COS(*expression*)	Trả lại cosin của biểu thức.
COT(*expression*)	Trả lại cô tang của biểu thức.
DEGREES(*expression*)	Chuyển đổi góc được tính theo radian tới một góc tính theo độ.
EXP(*expression*)	Trả lại giá trị số mũ của biểu thức.
FLOOR(*expression*)	Trả lại số nguyên lớn nhất ít hơn hay bằng biểu thức.
LOG(*expression*)	Trả lại lôga tự nhiên của biểu thức.
LOG10(*expression*)	Trả lại lôga cơ số 10 của biểu thức.
PI()	Trả lại hằng số toán học Pi.
POWER(*expression*, *y*)	Trả lại giá trị của biểu thức được nâng lên lũy thừa Y.
RADIANS(*expression*)	Chuyển đổi góc được cung cấp tính bằng độ tới một góc tính theo radian.
RAND([*expression*])	Trả lại một số dấu chấm động ngẫu nhiên giữa 0 và 1. Biểu thức là một giá trị khởi đầu mà bạn có thể dùng để phát sinh số ngẫu nhiên.
ROUND(*expression*, *length* [, *function*])	Trả lại giá trị của biểu thức được làm tròn hay cắt bỏ phần thập phân được chỉ định bởi length (độ dài). Tùy chọn function được sử dụng để chỉ định kiểu thao tác để thực hiện: 0 (mặc định) làm tròn số, và những giá trị khác là cát bỏ số.
SIGN(*expression*)	Trả lại 1, 0, hay-1 phụ thuộc vào dấu của biểu thức. Trả lại 1 cho một số dương ,0 cho chữ số không, hay-1 cho một số âm.
SIN(*expression*)	Trả lại sin của biểu thức.
SQUARE(*expression*)	Trả lại trị bình phương của biểu thức.
SQRT(*expression*)	Trả lại căn bậc hai của biểu thức.
TAN(*expression*)	Trả lai tang của biểu thức.

Chúng ta hãy xem xét những ví dụ sử dụng một số những hàm toán học. Bạn sử dụng những hàm ABS() để lấy giá trị tuyệt đối. Ví dụ sau đây trả lại 10 và 15:

```
SELECT ABS(-10), ABS(15);
```

Bạn sử dụng những hàm ACOS(), ASIN(), và ATAN() để lấy arccosine, arcsine, và Arctangent của một số. Ví dụ sau đây trả lại 0.0, 1.5707963267948966, và 0.78539816339744828:

```
SELECT ACOS(1), ASIN(1), ATAN(1);
```

Bạn sử dụng hàm CEILING() để lấy số nguyên nhỏ nhất lớn hơn hay bằng với giá trị được chuyển cho hàm. Ví dụ sau đây trả lại 2 và-1

```
SELECT CEILING(1.4), CEILING(-1.4);
```

Bạn sử dụng hàm FLOOR() để lấy số nguyên lớn nhất nhỏ hơn hay bằng với giá trị chuyển cho hàm. Ví dụ sau đây trả lại 1 và-2

```
SELECT FLOOR(1.4), FLOOR(-1.4);
```

Bạn sử dụng PI() để lấy hằng số Pi toán học. Ví dụ sau đây trả lại 3.1415926535897931

```
SELECT PI();
```

Bạn sử dụng hàm POWER() để lấy giá trị của một số nâng lên với một lũy thừa chỉ định. Ví dụ sau đây trả lại 8

```
SELECT POWER(2, 3);    /*    ( 2^3 = 8 )   */
```

Bạn sử dụng hàm ROUND() để lấy giá trị của một số được làm tròn hay cắt bỏ tới một chiều dài được chỉ định. Ví dụ sau đây trả lại 1.23500, là từ số 1.23456 được làm tròn tới ba chữ số thập phân

```
SELECT ROUND(1.23456, 3);
```

Ví dụ kế tiếp gởi qua một số khác không như tham số thứ ba đến hàm ROUND(), nó chỉ định con số sẽ được cắt, thay vì làm tròn như được làm trong ví dụ trước đây:

```
SELECT ROUND(1.23456, 3, 1);
```

Ví dụ này trả lại 1.23400, là từ số 1.23456 được cắt bỏ sau ba chữ số thập phân.

Bạn sử dụng hàm SQUARE() để lấy bình phương của một số. Ví dụ sau đây trả lại 16.0

```
SELECT SQUARE(4);
```

Bạn sử dụng hàm SQRT() để lấy căn bậc hai của một số . Ví dụ sau đây trả lại 4.0

```
SELECT SQRT(16);
```

Sử dụng những hàm chuỗi

Những hàm chuỗi cho phép bạn thao tác những chuỗi. Chẳng hạn, bạn có thể thay thế những ký tự chỉ định trong một chuỗi. Bảng 4.5 liệt kê những hàm chuỗi sẵn có trong SQL Server.

Bảng 4.5: những hàm chuỗi

HÀM	MÔ TẢ
ASCII(*charExpression*)	Trả lại mã ASCII cho ký tự đầu tiên bên trái của biểu thức chuỗi char (charExpression).
CHAR(*intExpression*)	Trả lại ký tự tương ứng cho mã ASCII được chỉ rõ bởi intExpression.
CHARINDEX (*charExpression1*, *charExpression2* [, *start*])	Trả lại vị trí của những ký tự được chỉ rõ bởi charExpression1 trong charExpression2, bắt đầu tại vị trí tùy chọn được chỉ rõ bởi start.
DIFFERENCE (*charExpression1*, *charExpression2*)	Trả lại sự khác nhau giữa những giá trị SOUNDEX của hai biểu thức ký tự . Bạn sử dụng mã SOUNDEX để đánh giá sự giống nhau về ngữ âm của hai chuỗi. Giá trị trả về giữa 0 và 4; 4 chỉ báo hai biểu thức có ngữ âm đồng nhất.

LEFT(*charExpression*, *intExpression*)	Trả lại những ký tự mút trái được chỉ rõ bởi intExprssion (số ký tự mút trái cần lấy) từ charExpression (chuỗi ký tự cần lọc).
LEN(*charExpression*)	Trả lại số lượng ký tự trong charExpression (chuỗi ký tự).
LOWER(*charExpression*)	Chuyển đổi những ký tự trong charExpression thành những ký tự thường và trả về những ký tự này.
LTRIM(*charExpression*)	Loại bỏ bất kỳ khoảng trống nào từ mút trái của chuỗi charExpression và trả về những ký tự còn lại.
NCHAR(*intExpression*)	Trả lại ký tự Unicode với mã được chỉ định bởi biểu thức số intExpression.
PATINDEX('%pattern%', *charExpression*)	Trả lại vị trí khởi đầu của lần xuất hiện đầu tiên của mẫu (pattern) có trong chuỗi charExpression. Nếu mẫu không được tìm thấy thì những chữ số không được trả về.
REPLACE (*charExpression1*, *charExpression2*, *charExpression3*)	Thay thế tất cả các xuất hiện của chuỗi con charExpression2 có trong chuỗi mẹ charExpression1 với chuỗi thay thế charExpression3.
QUOTENAME ('*charString*' [, '*quoteChar*'])	Trả lại một chuỗi Unicode với những dấu phân cách được chỉ rõ bởi quoteChar được thêm vào để tạo cho charString một định danh giới hạn hợp lệ.
REPLICATE (*charExpression*, *intExpression*)	Lặp lại chuỗi charExpression với số lần đươc chỉ định bởi intExpression.
REVERSE(*charExpression*)	Đảo ngược trật tự của những ký tự trong chuỗi charExpression và trả về những ký tự đó.
RIGHT(*charExpression*, *intExpression*)	Trả lại những ký tự mút bên phải của chuỗi charExpression với số ký tự trích ra được chỉ rõ bởi intExprssion .
RTRIM(*charExpression*)	Loại bỏ bất kỳ khoảng trống nào từ cuối cùng bên phải của chuỗi charExpression và trả về những ký tự còn lại.
SOUNDEX(*charExpression*)	Trả lại mã SOUNDEX bốn ký tự. Bạn sử dụng mã này để đánh giá sự giống nhau về ngữ âm của hai chuỗi.
SPACE(*intExpression*)	Trả lại một chuỗi của những khoảng cách được lặp lại với tổng số lần được chỉ rõ bởi intExpression.
STR(*floatExpression* [, *length* [, *decimal*]])	Chuyển đổi con số được chỉ rõ bởi floatExpression thành những ký tự ; length chỉ rõ số lượng ký tự mà bạn muốn nhìn thấy (bao gồm những chữ số và những khỏang cách , dấu dương hay dấu trừ và dấu phẩy ở số thập phân thêm vào); số thập phân chỉ rõ số lượng chữ số ở bên phải của dấu phẩy ở số thập phân. Số được làm tròn nếu cần thiết.
STUFF (*charExpression1*, *start*, *length*, *charExpression2*)	Xóa những ký tự từ chuỗi charExpression1, bắt đầu tại vị trí được chỉ định bởi start với một tổng số ký tự cần xóa length , và rồi chèn những ký tự chỉ rõ bởi charExpression2.
SUBSTRING(*expression*, *start*, *length*)	Trả lại bộ phận biểu thức của một ký tự , nhị phân,văn bản, hay ảnh.

UNICODE('nCharExpression')	Trả lại giá trị Unicode cho ký tự đầu tiên của biểu thức nCharExpression -nchar hay nvarchar .
UPPER(*charExpression*)	Chuyển đổi những ký tự trong chuỗi charExpression thành chữ hoa và trả lại những ký tự đó.

Chúng ta hãy xem xét những ví dụ mà sử dụng một số những hàm chuỗi.

Bạn sử dụng hàm ASCII() để lấy mã ASCII cho ký tự mút trái của biểu thức ký tự được cung cấp. Ví dụ sau đây trả lại 65 và 97

```
SELECT ASCII('A'), ASCII('a');
```

Bạn sử dụng hàm CHAR() để lấy ký tự tương ứng tới mã ASCII của biểu thức số nguyên được cung cấp. Ví dụ sau đây trở lại A và a :

```
SELECT CHAR(65), CHAR(97);
```

Bạn sử dụng hàm CHARINDEX() để lấy vị trí của những ký tự . Ví dụ sau đây trả lại 16, nó là vị trí khởi đầu của từ " ten" :

```
SELECT CHARINDEX('ten', 'Four-score and ten years');
```

Chú thích : ký tự đầu tiên của chuỗi có vị trí 1, 16 làvị trí của ký tự "t" tính từ ký tự đầu tiên của chuỗi bao gồm cả dấu cách.

Bạn sử dụng hàm DIFFERENCE() để thu được sự khác nhau giữa những giá trị SOUNDEX của hai biểu thức ký tự . Ví dụ sau đây trả lại 4, cho biết Browne và 'Browne' có ngữ âm đồng nhất

```
SELECT DIFFERENCE('Brown', 'Browne');
```

Bạn sử dụng hàm LEFT() để thu được những ký tự mút trái của một biểu thức ký tự. Ví dụ sau trả lại "Four-score", là 10 ký tự mút trái của chuỗi "Four-score and ten years":

```
SELECT LEFT('Four-score and ten years', 10);
```

Bạn sử dụng hàm RIGHT() để thu được những ký tự mút bên phải của một biểu thức ký tự . Ví dụ sau đây trả lại "year", là năm ký tự mút bên phải của chuỗi "Four-score and ten years":

```
SELECT RIGHT('Four-score and ten years', 5);
```

Bạn sử dụng hàm LEN() để thu được con số cho biết tổng số ký tự (kể cả những kí tự đặc biệt và dấu cách) có trong một biểu thức ký tự . Ví dụ sau đây trả lại 24:

```
SELECT LEN('Four-score and ten years');
```

Bạn sử dụng hàm LOWER() để thu được phiên bản chữ thường của một biểu thức ký tự . Ví dụ sau đây trả lại "four-score and ten years":

```
SELECT LOWER('FOUR-SCORE AND TEN YEARS');
```

Bạn sử dụng hàm UPER() để thu được phiên bản chữ hoa của một biểu thức ký tự . Ví dụ sau đây trả lại "FOUR-SCORE AND TEN YEARS":

```
SELECT UPPER('four-score and ten years');
```

Bạn sử dụng hàm LTRIM() và RTRIM() để loại bỏ bất kỳ khoảng trống nào về bên trái và bên phải của một biểu thức ký tự . Ví dụ sau đây trả lại "FOUR-SCORE" và "AND TEN YEARS" (khoảng trống đã bị loại bỏ):

```
SELECT LTRIM(' FOUR-SCORE'), RTRIM('AND TEN YEARS ');
```

Bạn sử dụng hàm STR() để chuyển đổi một giá trị số tới một chuỗi có chứa những chữ số. Tham số đầu tiên là số để chuyển đổi, tham số thứ hai là tổng số lượng ký tự bạn muốn có trong chuỗi của bạn, và tham số thứ ba là số lượng chữ số sau dấu phẩy ở số thập phân. Ví dụ sau đây trả lại 123.46

```
SELECT STR(123.456, 6, 2);
```

Số 123.456 được chuyển đổi tới một chuỗi gồm sáu ký tự , với hai chữ số sau dấu phẩy ở số thập phân, và được làm tròn.

Bạn sử dụng hàm STUFF() để thay thế những ký tự . Tham số đầu tiên là chuỗi bạn muốn thay thế những ký tự trong đó, tham số thứ hai là vị trí ký tự bắt đầu được thay thế, tham số thứ ba là tổng số lượng ký tự được thay tính từ ký tự bắt đầu thay , và tham số thứ tư là tập hợp những ký tự để thay vào. Ví dụ sau đây trả lại "Five-score and ten":

```
SELECT STUFF('Four-score and ten', 1, 4, 'Five');
```

Bạn sử dụng hàm SUBSTRING() để lấy một phần của một chuỗi. Tham số đầu tiên là chuỗi, tham số thứ hai là vị trí bắt đầu lấy ra, và tham số thứ ba là tổng số lượng ký tự cần lấy tính từ vị trí bắt đầu . Ví dụ sau đây trả lại Four:

```
SELECT SUBSTRING('Four-score and ten', 1, 4);
```

Bạn sử dụng hàm UNICODE() để thu được giá trị Unicode cho ký tự đầu tiên. Ví dụ sau đây trả lại 65 và 97

```
SELECT UNICODE('A'), UNICODE('a');
```

Sử dụng những hàm ngày tháng và thời gian

Những hàm ngày tháng và thời gian cho phép bạn thao tác với ngày tháng và thời gian. Chẳng hạn, bạn có thể thêm một số ngày vào một ngày tháng đã cho. Bảng 4.6 liệt kê ngày tháng và những chức năng thời gian sẵn có trong SQL Server.

Bảng 4.6: những hàm ngày tháng và thời gian

HÀM	MÔ TẢ
DATEADD(*interval*, *number*, *date*)	Trả lại một ngày tháng (datetime) là kết quả của việc thêm số lượng (number) chỉ định của đơn vị khoảng thời gian (interval) vào ngày tháng (date). Những khoảng thời gian hợp lệ bao gồm : năm, quý, tháng ,dayofyear, ngày, tuần, giờ, phút, giây, và mili-giây (year, quarter, month, dayofyear, day, week, hour, minute, second, and millisecond).
DATEDIFF (*interval*, *startDate*, *endDate*)	Trả lại sự chênh lệch giữa startDate và endDate, với sự chênh lệch được tính toán theo những đơn vị khoảng thời gian (interval) như (năm, quý vân vân).
DATENAME(*interval*, *date*)	Trả lại một chuỗi ký tự đại diện cho tên của khoảng thời gian (interval) của ngày tháng (date).
DATEPART(*interval*, *date*)	Trả lại một số nguyên đại diện cho khoảng thời gian (interval) của ngày tháng (date).
DAY(*date*)	Trả lại một số nguyên mà đại diện cho ngày của ngày tháng (date).
GETDATE()	Trả lại một ngày tháng (datetime) chứa ngày tháng hệ thống hiện thời.
GETUTCDATE()	Trả lại một ngày tháng (datetime) chứa ngày tháng hệ thống hiện thời theo thời gian UTC (Tọa độ giờ quốc tế hay giờ Greenwich). Giờ UTC được bắt nguồn từ giờ địa phương hiện thời và sự thiết đặt hệ thống thời gian khu vực.
MONTH(*date*)	Trả lại một số nguyên đại diện cho tháng của ngày tháng (date).
YEAR(*date*)	Trả lại một số nguyên đại diện cho năm của ngày tháng (date) .

Chúng ta hãy xem xét những ví dụ sử dụng một số hàm ngày tháng và thời gian.

Bạn sử dụng hàm DATEADD() để thêm một số khoảng thời gian (intervals) vào một ngày tháng. Ví dụ sau đây thêm hai ngày tới ngày tháng 12/ 20/ 2003 và trả về 2003-12-22 00: 00: 00.000

```
SELECT DATEADD(day, 2, '12/20/2003');
```

Bạn sử dụng hàm DATEDIFF() để thu được sự chênh lệch giữa hai ngày tháng. Ví dụ sau đây thu được sự chênh lệch (tính theo ngày) giữa ngày 12/ 20/ 2003 và ngày 12/ 22/ 2003 và trả về 2 ngày

```
SELECT DATEDIFF(day, '12/20/2003', '12/22/2003');
```

Bạn sử dụng hàm DATENAME() để thu được một chuỗi ký tự đại diện cho phần tử của một ngày tháng. Ví dụ sau đây lấy tên tháng của 12/ 20/ 2003 và December (tháng mười hai) được trả về.

```
SELECT DATENAME(month, '12/20/2003');
```

Bạn sử dụng phương thức DATEPART() để thu được một số nguyên đại diện cho một phần tử của một ngày tháng . Ví dụ sau đây lấy số tháng của 12/ 20/ 2003 và trả lại 12

```
SELECT DATEPART(month, '12/20/2003');
```

Bạn sử dụng hàm DAY() để thu được một số nguyên đại diện cho một phần tử của một ngày tháng . Ví dụ sau đây lấy số ngày của 12/ 20/ 2003 và trả lại 20

```
SELECT DAY('12/20/2003');
```

Bạn sử dụng hàm MONTH() để thu được một số nguyên đại diện cho một phần tử của một ngày tháng . Ví dụ sau đây lấy số tháng của 12/ 20/ 2003 và trả lại 12

```
SELECT MONTH('12/20/2003');
```

Bạn sử dụng hàm YEAR() để thu được một số nguyên đại diện cho một phần tử của một ngày tháng . Ví dụ sau đây lấy số năm của 12/ 20/ 2003 và trả lại 2003

```
SELECT YEAR('12/20/2003');
```

Bạn sử dụng hàm GETDATE() để thu được ngày tháng hệ thống hiện thời. Ví dụ sau đây trả lại 2002-07-16 12: 59: 50.823 (ngày hôm nay)

```
SELECT GETDATE();
```

Bạn sử dụng hàm GETUTCDATE() để thu được ngày tháng hệ thống hiện thời với thời gian UTC (giờ quốc tế). Ví dụ sau đây trả lại 2002-07-16 20: 02: 18.123 (giờ quốc tế hôm nay)

```
SELECT GETUTCDATE();
```

Sử dụng những hàm Hệ thống

Những hàm hệ thống cho phép bạn thao tác và lấy những thông tin về những giá trị, những đối tượng và những thiết định trong SQL Server. Chẳng hạn, bạn có thể chuyển đổi một giá trị trong một kiểu tới kiểu khác.

Bảng 4.7 liệt kê một số hàm hệ thống sẵn có trong SQL Server.

Bảng 4.7: những hàm Hệ thống

TÊN HÀM	MÔ TẢ
CONVERT(*dataType expression* [, *style* [(*length*)],])	Chuyển đổi giá trị trong biểu thức (expression) tới kiểu chỉ rõ bởi dataType. Nếu bạn đang chuyển đổi một kiểu nchar, nvarchar, char, varchar, nhị phân, hay kiểu varbinary , bạn cũng có thể chỉ rõ một tùy chọn length , nó chỉ định chiều dài của giá trị mới. Bạn có thể sử dụng tùy chọn style (kiểu). ▪ Chuyển đổi dữ liệu datetime hay smalldatetime tới dữ liệu ký tự; style là định dạng cho ngày tháng và thời gian. ▪ Chuyển đổi dữ liệu float, real, money, or smallmoney tới dữ liệu ký tự ; style là định dạng chuỗi cho con số. Bạn có thể tìm xem những chi tiết cho tùy chọn style trong những tài liệu sách trực tuyến Người phục vụ SQL.
COALESCE(*expression1* [, ... *expressionN*])	Trả về biểu thức không null đầu tiên trong danh sách của những biểu thức.

DATALENGTH(*expression*).	Trả lại số bytes được dùng để trình bày biểu thức.
@@ERROR	Trả lại số đặc trưng của lỗi cho phát biểu T- SQL cuối cùng mà được thực hiện.
@@IDENTITY	Trả lại giá trị nhận dạng được chèn vào sau cùng .
ISDATE(*expression*)	Trả lại 1 khi biểu thức là một ngày tháng hợp lệ, nếu không số 0 được trở lại.
ISNULL(*expression*, *replacementValue*)	Nếu biểu thức có giá tri null, thì replacementValue được trả lại, cách khác biểu thức được trả lại.
ISNUMERIC(*expression*).	Trả lại 1 khi biểu thức là một số hợp lệ, cách khác 0 được trở lại.
NEWID()	Trả lại một giá trị duy nhất của kiểu uniqueidentifier.
NULLIF(*expression1*, *expression2*)	Trả lại một giá trị null nếu expression1 bằng expression2.
@@ROWCOUNT	Trả lại số lượng hàng được ảnh hưởng bởi câu lệnh T- SQL được thực hiện sau cùng.
@@TRANCOUNT	Trả lại số lượng giao dịch hoạt động cho kết nối hiện thời (currentconnection) tới cơ sở dữ liệu.

Chúng ta hãy xem xét những ví dụ sử dụng một số những hàm hệ thống.

Bạn sử dụng hàm CONVERT() để chuyển đổi một giá trị từ kiểu này sang kiểu khác. Ví dụ sau đây chuyển đổi số 123.456 tới một nvarchar ,và 123.456 được trả lại

```
SELECT CONVERT(nvarchar, 123.456);
```

Bạn sử dụng hàm COALESCE() để thu được biểu thức không null đầu tiên trong một danh sách. Ví dụ sau đây trả lại 123.456

```
SELECT COALESCE(null, null, 123.456, null);
```

Bạn sử dụng hàm DATALENGTH() để thu được số bytes được sử dụng để trình bày một biểu thức. Ví dụ sau đây trình bày số bytes dùng để trình bày giá trị cất giữ trong cột CompanyName của bảng những khách hàng cho hàng có CustomerID bằng ALFKI:

```
SELECT DATALENGTH(CompanyName), CompanyName
FROM Customers
WHERE CustomerID = 'ALFKI';
```

Ví dụ này trả lại 38 và Alfreds Futterkiste, chứa 19 ký tự. mỗi ký tự được chứa trong 2 bai, và chuỗi 19-ký tự giữ tới 38 bai (2* 19).

Bạn sử dụng hàm ISDATE() để xác định phải chăng một biểu thức là một ngày tháng hợp lệ. ISDATE() trả về 1 khi biểu thức là một ngày tháng hợp lệ, cách khác nó trả lại 0. Ví dụ sau đây trả lại 1 và 0

```
SELECT ISDATE('12/20/2004'), ISDATE(1234);
```

Bạn sử dụng hàm ISNUMERIC() để xác định phải chăng một biểu thức là một số hợp lệ. Ví dụ sau đây trả lại 1 và 0

```
SELECT ISNUMERIC(1234), ISNUMERIC('abc');
```

Bạn sử dụng hàm ISNULL() để thay thế một giá trị null với giá trị khác. Ví dụ sau đây trả lại 10 và 20

```
SELECT ISNULL(null, 10), ISNULL(20, 10);
```

Tạo ra những hàm do người dùng định nghĩa

Bạn có thể tạo ra những hàm người dùng định nghĩa của riêng mình trong SQL Server. Chẳng hạn, bạn có thể muốn tạo ra hàm của riêng mình để tính toán giá chiết khấu từ một giá gốc và hệ số nhân với giá này . Bạn tạo ra một hàm sử dụng phát biểu CREATE FUNCTION. Có ba kiểu hàm người dùng định nghĩa:

Scalar functions: (những hàm vô hướng) trả lại một giá trị đơn. Giá trị trả về có thể thuộc bất kỳ kiểu dữ liệu nào ngoại trừ text, ntext, image (ảnh), cursor (con trỏ), table, timestamp, và những kiểu dữ liệu do người dùng định nghĩa.

Inline table-valued functions: (những hàm định trị bảng nội tuyến) trả về một đối tượng kiểu bảng . Bạn có thể hiểu một bảng như một bảng cơ sở dữ liệu bình thường, ngoại trừ nó được cất giữ trong bộ nhớ. Một "hàm định trị bảng nội tuyến" có thể trả về những kết quả được truy xuất bởi chỉ một phát biểu SELECT đơn.

Multistatement table-valued functions (những hàm định trị bảng nhiều phát biểu): trả về một đối tượng kiểu bảng. Không giống một Inline table-valued function, một Multistatement table-valued function có thể chứa nhiều phát biểu T- SQL.

Bạn sẽ xem những ví dụ của ba kiểu hàm này trong những mục sau đây.

Sử dụng những hàm vô hướng

Những hàm vô hướng trả lại một giá trị đơn. Danh sách 4.2 trình bày script DiscountPrice.sql nó tạo ra hàm DiscountPrice(), hàm này trả về giá chiết khấu = *giá gốc* của một món được nhân với một *hệ số chiết khấu*. Những giá trị này được gởi qua như những tham số tới hàm DiscountPrice() . Bạn có thể tải file này vào Query Analyzer và chạy nó.

Danh sách 4.2: DISCOUNTPRICE.SQL

```
/*

  DiscountPrice.sql creates a scalar function to
  return the new price of an item given the original
  price and a discount factor
*/

CREATE FUNCTION DiscountPrice(@OriginalPrice money, @Discount float)
RETURNS money
AS
BEGIN
  RETURN @OriginalPrice * @Discount
END
```

Những tham số gởi tới hàm được đặt trong cặp dấu móc sau tên của hàm trong phát biểu CREATE FUNCTION .

Cảnh báo:

Bảo đảm là bạn đã chọn cơ sở dữ liệu Northwind từ hộp danh sách xổ xuống trên thanh công cụ Query Analyzer trước khi chạy script. Bằng cách này, hàm sẽ được tạo ra trong cơ sở dữ liệu Northwind.

Bạn cũng có thể tạo ra những hàm sử dụng Enterprise Manager. Bạn làm điều này bởi kích chuột phải trên node "User Defined Functions" trong thư mục Databases . Rồi bạn có thể cắt và dán nội dung của DiscountPrice.sql vào trong hộp thoại Enterprise Manager properties, như được trình bày trong Hình 4.4.

Hình 4.4: Sử dụng Enterprise Manager để định nghĩa một hàm

Bạn có thể xem và sửa đổi một hàm bởi nhấn đúp tên hàm trong Enterprise Manager. Bạn cũng có thể xóa một hàm sử dụng Enterprise Manager. Object Browser (Bộ duyệt Đối tượng) của Query Analyzer (trình phân tích truy vấn) còn cho phép bạn xem, sửa đổi, và xóa những hàm nữa.

Mẹo nhỏ:

Bạn cũng có thể xóa một hàm sử dụng phát biểu DROP FUNCTION , và Bạn có thể sửa đổi một hàm sử dụng phát biểu ALTER FUNCTION.

Một khi bạn đã tạo ra hàm , bạn có thể gọi nó. Khi gọi một hàm vô hướng , bạn sử dụng cú pháp sau đây:

owner.functionName

Với owner là người sử dụng cơ sở dữ liệu -người sở hữu hàm , và functionName là tên của hàm.

Cho là bạn đã tạo ra hàm DiscountPrice() sử dụng người sử dụng dbo, thì bạn gọi hàm này sử dụng cú pháp : dbo.DiscountPrice(). Ví dụ sau đây trả về 3.0000, tức là 10* 0.3 :

```
SELECT dbo.DiscountPrice(10, 0.3);
```

Như với bất kỳ hàm nào khác, bạn có thể gởi một cột tới DiscountPrice(). Ví dụ sau đây trả lại 5.4000 và 18.0000; 5.4000 = 18.0000* 0.3

```
SELECT dbo.DiscountPrice(UnitPrice, 0.3), UnitPrice
FROM Products
WHERE ProductID = 1;
```

Tất nhiên bạn cũng có thể gởi những biến như những tham số tới một hàm . như trước đây, ví dụ này trả lại 5.4000 và 18.0000

```
DECLARE @MyDiscountFactor float
SET @MyDiscountFactor = 0.3
SELECT dbo.DiscountPrice(UnitPrice, @MyDiscountFactor), UnitPrice
FROM Products
WHERE ProductID = 1;
```

Sử dụng những hàm định trị bảng nội tuyến

Một hàm định trị bảng nội tuyến (inline table-valued function) trả về một đối tượng kiểu bảng, được cư trú - sử dụng một phát biểu SELECT đơn. Không giống một hàm vô hướng (scalar function), một hàm định trị bảng nội

tuyến không chứa đựng một thân của những phát biểu được đặt bên trong những phát biểu BEGIN và END . Thay vào đó, chỉ một phát biểu SELECT đơn được đặt bên trong hàm.

Chẳng hạn, Danh sách 4.3 trình bày script "ProductsToBeReordered.sql" nó tạo ra hàm ProductsToBeReordered(). hàm này trả về một bảng chứa những hàng từ bảng những sản phẩm với một giá trị cột UnitsInStock nhỏ hơn hay bằng với tham số mức "reorder" gởi đến cho hàm.

Danh sách 4.3: PRODUCTSTOBEREORDERED.SQL

```
/*
  ProductsToBeReordered.sql creates an inline table-valued function to
  return the rows from the Products table whose UnitsInStock column
  is less than or equal to the reorder level passed as a parameter
  to the function
*/

CREATE FUNCTION ProductsToBeReordered(@ReorderLevel int)
RETURNS table
AS
RETURN
(
  SELECT *
  FROM Products
  WHERE UnitsInStock <= @ReorderLevel
)
```

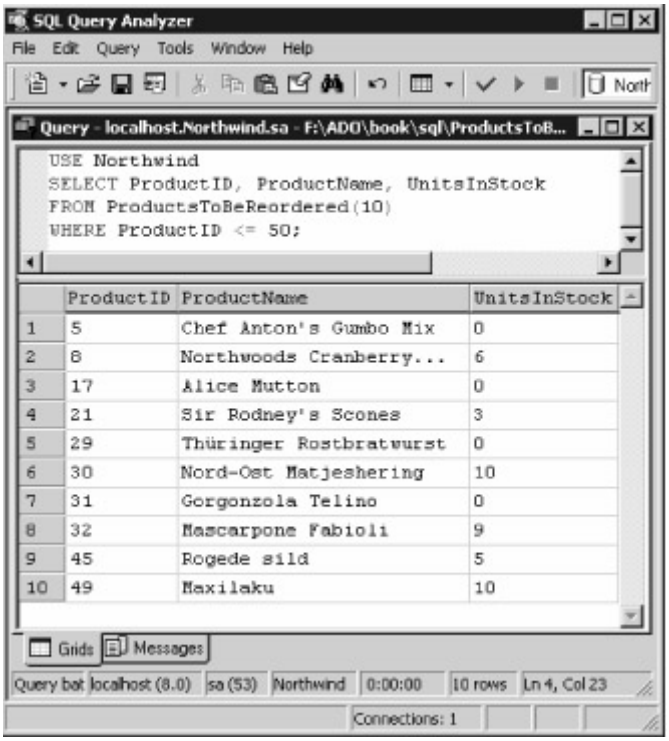
Không giống một hàm vô hướng, bạn không phải thêm owner (tên tài khoản người dùng) khi sự gọi một hàm "inline table-valued function". Bạn sử dụng một phát biểu SELECT để đọc bảng được trả về bởi hàm như mọi bảng khác. Chẳng hạn, phát biểu SELECT sau đây trình bày tất cả những hàng và những cột được trả về bởi gọi hàm ProductsToBeReordered(10):

```
SELECT *
FROM ProductsToBeReordered(10);
```

Tất nhiên bạn cũng có thể trình bày chỉ những cột và những hàng được lựa chọn từ bảng được trả về bởi một hàm "inline table-valued function". Chẳng hạn:

```
SELECT ProductID, ProductName, UnitsInStock
FROM ProductsToBeReordered(10)
WHERE ProductID <= 50;
```

Hình 4.5 cho thấy những kết quả của phát biểu SELECT này.



Hình 4.5: sử dụng một hàm định trị bảng nội tuyến

Sử dụng những hàm định trị bảng đa phát biểu

Những hàm định trị bảng đa phát biểu (Multistatement table-valued functions) trả lại một đối tượng kiểu bảng. Không giống một hàm định trị bảng nội tuyến (inline table-valued function), một hàm định trị bảng đa phát biểu có thể chứa nhiều câu lệnh T- SQL, và cho phép bạn xây dựng những hàm phức tạp.

Chẳng hạn, Danh sách 4.4 trình bày script "ProductsToBeReordered2.sql" tạo ra hàm ProductsToBeReordered2() . Hàm này trả về một bảng chứa những cột ProductID, ProductName, và UnitsInStock từ bảng những sản phẩm (Products) với một giá trị cột UnitsInStock nhỏ hơn hay bằng với tham số mức "reorder". Ngoài ra, một cột mới tên Reorder được thêm vào bảng, chứa từ YES hay NO, tùy thuộc vào sản phẩm có được đặt mua bổ xung (reorder) hay không.

Danh sách 4.4: PRODUCTSTOBEREORDERED2. SQL

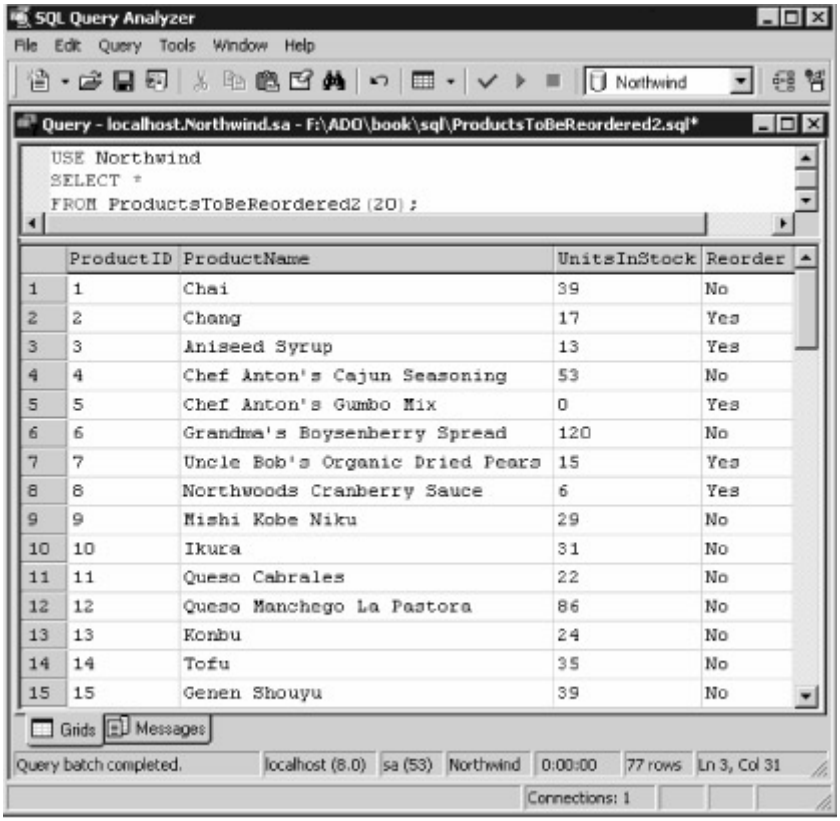
```
/*
    ProductsToBeReordered2 .sql tạo ra một hàm định trị bảng nội tuyến mà trả về những hàng
    từ bảng những sản phẩm có cột UnitsInStock nhỏ hơn hay bằng với mức phải đặt mua bổ
    xung được gởi qua như một tham số tới hàm

*/

CREATE FUNCTION ProductsToBeReordered2(@ReorderLevel int)
RETURNS @MyProducts table
(
  ProductID int,
  ProductName nvarchar(40),
  UnitsInStock smallint,
  Reorder nvarchar(3)
)
AS
BEGIN

  -- truy xuất những hàng từ bảng Products và
  -- và chèn chúng vào bảng MyProducts,
```

```
    -- thiết đặt cột Reorder tới 'No'
    INSERT INTO @MyProducts
      SELECT ProductID, ProductName, UnitsInStock, 'No'
      FROM Products;

    -- cập nhật bảng MyProducts, thiết đặt cột
    -- Reorder tới 'Yes' khi cột UnitsInStock
    -- nhỏ hơn hay bằng @ReorderLevel
    UPDATE @MyProducts
    SET Reorder = 'Yes'
    WHERE UnitsInStock <= @ReorderLevel

    RETURN

  END
```

như với một hàm định trị bảng nội tuyến, bạn không phải thêm owner (tên tài khoản người dùng) khi gọi một hàm định trị bảng nội tuyến. Bạn sử dụng một phát biểu SELECT để đọc bảng được trả về bởi hàm như mọi bảng cơ sở dữ liệu bình thường khác. Chẳng hạn, phát biểu SELECT sau đây trình bày tất cả những hàng và những cột được trả về bởi gọi hàm ProductsToBeReordered2(20):

```
SELECT *
FROM ProductsToBeReordered2(20);
```

Hình 4.6 cho thấy những kết quả của phát biểu SELECT này.



Hình 4.6: sử dụng một hàm multistatement table-valued function

Trong mục kế tiếp, bạn sẽ học sử dụng những thủ tục lưu trữ như thế nào.

Giới thiệuvề những thủ tục lưu trữ

SQL Server cho phép bạn lưu trữ những thủ tục trong một cơ sở dữ liệu. Những thủ tục lưu trữ khác với những hàm do người dùng định nghĩa trong đó những thủ tục này có thể trả về một mảng rộng lớn của những kiểu dữ liệu.

Bạn điển hình sẽ tạo ra một thủ tục lưu trữ khi bạn cần thực hiện một tác vụ mà sử dụng cách mạnh mẽ cơ sở dữ liệu, hay bạn muốn tập trung mã trong cơ sở dữ liệu mà bất kỳ người sử dụng nào cũng có thể gọi thay vì mỗi người sử dụng phải viết chương trình riêng của mình để thực hiện nhiệm vụ giống như vậy. Một ví dụ về sử dụng cơ sở dữ liệu mạnh là một ứng dụng ngân hàng mà bạn cần cập nhật những tài khoản vào cuối mỗi ngày. Một ví dụ về khi nào bạn cần sử dụng mã tập trung là khi bạn muốn hạn chế những người sử dụng truy nhập tới những bảng cơ sở dữ liệu: bạn có thể đã muốn những người sử dụng có khả năng thêm một hàng vào một bảng chỉ thông qua một thủ tục lưu trữ để không sảy ra những lỗi thực thi.

Trong mục này, bạn sẽ học cách tạo ra một thủ tục lưu trữ trong cơ sở dữ liệu Northwind và nó được chạy như thế nào sử dụng công cụ Query Analyzer .

Tạo ra một Thủ tục lưu trữ

Thủ tục bạn sẽ thấy trong mục này có tên AddProduct(). Thủ tục này thêm một hàng vào bảng những sản phẩm, gán những giá trị cột cho hàng mới với những giá trị được gởi qua như những tham số tới thủ tục.

Cột ProductID cho hàng mới được gán một giá trị một cách tự động bởi cơ sở dữ liệu thông qua việc sử dụng một identity được thiết lập khi bảng lúc thoạt tiên được tạo . Giá trị identity này có thể được đọc sử dụng hàm @@IDENTITY sau khi hàng mới được thêm vào bảng. Thủ tục lưu trữ AddProduct() bạn sẽ thấy ở đây trả về giá trị identity này cho phát biểu gọi hàm.

Bạn tạo ra một thủ tục sử dụng phát biểu CREATE PROCEDURE , và Danh sách 4.5 cho thấy script "AddProduct.sql" tạo ra thủ tục lưu trữ AddProduct() .

Danh sách 4.5: ADDPRODUCT.SQL

```
/*
  AddProduct.sql tạo ra một 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.
*/

CREATE PROCEDURE AddProduct
  @MyProductName nvarchar(40),
  @MySupplierID int,
  @MyCategoryID int,
  @MyQuantityPerUnit nvarchar(20),
  @MyUnitPrice money,
  @MyUnitsInStock smallint,
  @MyUnitsOnOrder smallint,
  @MyReorderLevel smallint,
  @MyDiscontinued bit
AS
DECLARE @ProductID 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 @@IDENTITY function to get the last inserted
-- identity value, which in this case is the ProductID of
-- the new row in the Products table
SET @ProductID = @@IDENTITY

-- return the ProductID
RETURN @ProductID
```

Bạn cũng có thể tạo ra những thủ tục sử dụng Enterprise Manager (bộ quản lý doanh nghiệp). Bạn làm điều này bởi kích chuột phải, nút trên node "Stored Procedures" (thủ tục lưu trữ) trong thư mục Databases (những cơ sở dữ liệu) và chọn New Stored Procedure (Thủ tục lưu trữ mới). Rồi bạn có thể cắt và dán nội dung của AddProduct.sql vào trong hộp thoại Enterprise Manager properties (những thuộc tính bộ quản lý doanh nghiệp), như được trình bày trong hình 4.7. Bạn chú ý - Tôi có thêm một số chú thích vào doạn đầu của file này cho biết những gì thủ tục thực hiện.

Hình 4.7: Sử dụng Bộ quản lý doanh nghiệp (Enterprise Manager) để định nghĩa một thủ tục

Bạn có thể xem và sửa đổi một thủ tục bởi nhấn đúp tên thủ tục trong Enterprise Manager. Bạn cũng có thể xóa một thủ tục sử dụng Enterprise Manager . Object Browser (Bộ duyệt đối tượng) của Query Analyzer (bộ phân tích truy vấn) còn cho phép bạn xem, sửa đổi, và xóa những thủ tục nữa.

> **Mẹo nhỏ:** Bạn cũng có thể xóa một thủ tục sử dụng lệnh DROP PROCEDURE , và bạn có thể sửa đổi một Thủ tục sử dụng lệnh ALTER PROCEDURE .

Trong mục kế tiếp, bạn sẽ thấy cách chạy một thủ tục lưu trữ như thế nào.

Chạy một Thủ tục lưu trữ

Bạn chạy một thủ tục sử dụng phát biểu EXECUTE . Chẳng hạn, những phát biểu sau đây chạy thủ tục AddProduct() :

```
DECLARE @MyProductID int
EXECUTE @MyProductID = AddProduct 'Widget', 1, 1, '1 Per box', 5.99, 10, 5, 5, 1
PRINT @MyProductID
```

Với sự thiết đặt ban đầu của những hàng trong bảng những sản phẩm, giá trị khóa chính kế tiếp được phát sinh bởi SQL Server cho ProductID là 78, đó là giá trị được trình bày bởi ví dụ trước nếu bạn chạy nó.

Tất nhiên bạn cũng có thể gởi những biến như những tham số tới một thủ tục. Ví dụ sau đây trình bày 79- là giá trị khóa chính của cột ProductID tiếp theo:

```
DECLARE @MyProductID int
DECLARE @MyProductName nvarchar(40)
DECLARE @MySupplierID int
DECLARE @MyCategoryID int
DECLARE @MyQuantityPerUnit nvarchar(20)
DECLARE @MyUnitPrice money
DECLARE @MyUnitsInStock smallint
DECLARE @MyUnitsOnOrder smallint
DECLARE @MyReorderLevel smallint
DECLARE @MyDiscontinued bit

SET @MyProductName = 'Wheel'
SET @MySupplierID = 2
SET @MyCategoryID = 1
SET @MyQuantityPerUnit = '4 per box'
SET @MyUnitPrice = 99.99
SET @MyUnitsInStock = 10
SET @MyUnitsOnOrder = 5
SET @MyReorderLevel = 5
SET @MyDiscontinued = 0

EXECUTE @MyProductID = AddProduct @MyProductName,
  @MySupplierID, @MyCategoryID, @MyQuantityPerUnit,
  @MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder,
  @MyReorderLevel, @MyDiscontinued

PRINT @MyProductID
```

Giới thiệu về những triggers

Một trigger cơ sở dữ liệu (a database trigger) là một kiểu đặc biệt của thủ tục lưu trữ, nó được chạy tự động bởi cơ sở dữ liệu- hay theo những thuật ngữ trigơ, được khai hỏa- sau khi một phát biểu chèn , cập nhật hay xóa chỉ định chạy tiếp cận một bảng cơ sở dữ liệu được chỉ rõ. Những trigơ rất hữu ích để thực hiện những thứ như kiểm định những sự thay đổi được thực hiện tới những giá trị cột trong một bảng.

Một trigơ cũng có thể khởi chạy thay cho một INSERT, UPDATE, hay DELETE. Chẳng hạn, thay vì thực hiện một INSERT để thêm một hàng vào bảng những sản phẩm, một trigơ đã có thể ném ra một lỗi nếu một sản phẩm với cùng một ProductID đã tồn tại trong bảng.

Như đã đề cập, những trigơ rất hữu ích cho sự kiểm định những sự thay đổi đã làm tới những giá trị cột . Trong mục này, bạn sẽ xem xét một ví dụ một trigơ mà sẽ kiểm định những thay đổi được thực hiện tới bảng những sản phẩm.

Đồng thời, khi một phát biểu cập nhật sửa đổi cột UnitPrice của một hàng trong bảng những sản phẩm, một hàng sẽ được thêm vào bảng ProductAudit. Cuối cùng, Khi một phát biểu xóa loại bỏ một hàng từ bảng những sản phẩm, một hàng sẽ được thêm vào bảng ProductAudit.

Trước khi bạn xem những trigơ, bạn sẽ cần tạo ra bảng ProductAudit. Danh sách 4.6 trình bày một script "ProductAudit.sql" mà tạo ra bảng ProductAudit.

Danh sách 4.6: PRODUCTAUDIT.SQL

```
/*
  ProductAudit.sql creates a table that is used to
  store the results of triggers that audit modifications
  to the Products table
*/

USE Northwind

CREATE TABLE ProductAudit (
  ID int IDENTITY(1, 1) PRIMARY KEY,
  Action nvarchar(100) NOT NULL,
  PerformedBy nvarchar(15) NOT NULL DEFAULT User,
  TookPlace datetime NOT NULL DEFAULT GetDate()
)
```

Mệnh đề IDENTITY tạo ra một mã khóa cho cột khóa chính ID của bảng ProductAudit. Một mã khóa tự động phát sinh những giá trị cho một cột. Mã khóa cho cột ID bắt đầu với giá trị 1, nó được tăng thêm 1 sau mỗi lần INSERT. Cột Action lưu trữ một chuỗi ghi hoạt động đã thực hiện, chẳng hạn,' Sản phẩm bổ sung với ProductID là 80'. Cột PerformedBy lưu giữ tên của người dùng đã thực hiện hành động; Đây là mặc định với User (Người dùng), nó trả về Người dùng hiện thời. Cột TookPlace lưu giữ ngày tháng và thời gian khi hành động xảy ra; đây này mặc định sử dụng hàm GetDate() , nó trả về ngày tháng và thời gian hiện thời.

Trong những mục sau đây, bạn sẽ học cách tạo và sử dụng những trigơ sau đây như thế nào:

InsertProductTrigger khởi chạy sau khi một phát biểu INSERT được thực hiện trên bảng những sản phẩm.

UpdateUnitPriceProductTrigger khởi chạy sau khi một phát biểu UPDATE được thực hiện trên bảng những sản phẩm.

DeleteProductTrigger khởi chạy sau khi một phát biểu DELETE được thực hiện trên bảng những sản phẩm.

Trước hết, chúng ta hãy khảo sát InsertProductTrigger.

Tạo ra InsertProductTrigger

Bạn tạo ra một trigơ sử dụng phát biểu CREATE TRIGGER . Danh sách 4.7 trình bày một script "InsertProductTrigger.sql" tạo ra trigơ InsertProductTrigger, nó kiểm định sự thêm những hàng mới tới bảng những sản phẩm.

Danh sách 4.7: INSERTPRODUCTTRIGGER.SQL

```
/*
  InsertProductTrigger.sql creates a trigger that fires
  after an INSERT statement is performed on the
  Products table
*/

CREATE TRIGGER InsertProductTrigger
ON Products
AFTER INSERT
AS

  -- don't return the number of rows affected
  SET NOCOUNT ON

  -- declare an int variable to store the new
```

```
-- ProductID
DECLARE @NewProductID int

-- get the ProductID of the new row that
-- was added to the Products table
SELECT @NewProductID = ProductID
FROM inserted

-- add a row to the ProductAudit table
INSERT INTO ProductAudit (
  Action
) VALUES (
  'Product added with ProductID of ' +
    CONVERT(nvarchar, @NewProductID)
)
```

Có vài thứ bạn cần phải chú ý về phát biểu CREATE TRIGGER này.

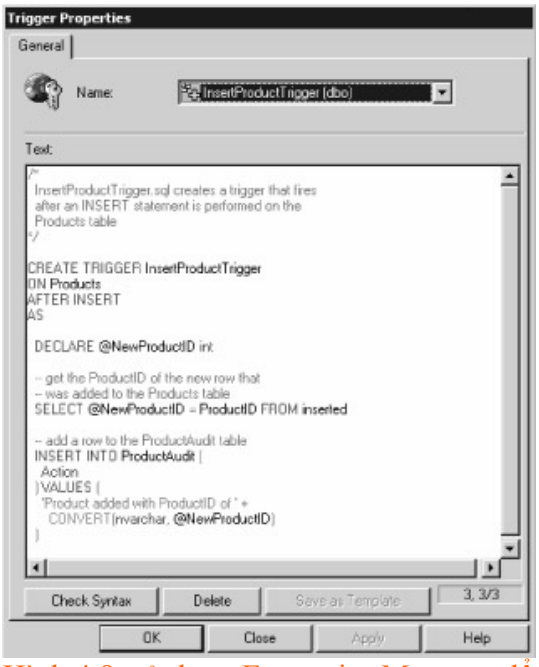
Mệnh đề AFTER INSERT chỉ rõ trigơ sẽ khởi chạy sau khi một phát biểu INSERT được thực hiện.

SET NOCOUNT ON ngăn ngừa trigơ trả về số lượng hàng bị ảnh hưởng. Điều này cải thiện sự thực thi của trigơ.

Bạn có thể truy xuất những giá trị cột cho phát biểu INSERT mà gây cho trigơ khởi chạy bởi thực hiện một SELECT tiếp cận bảng inserted đặc biệt. Chẳng hạn, bạn có thể truy xuất tất cả những cột của một hàng mới bổ sung sử dụng SELECT * FROM inserted . Mã trigơ truy xuất cột ProductID của hàng mới từ bảng inserted .

Phát biểu INSERT thêm một hàng vào bảng ProductAudit cung cấp một giá trị chỉ cho cột Action . Đây là vì những giá trị cột ID, PerformedBy, Và TookPlace được gán tự động bởi SQL Server.

Bạn cũng có thể tạo ra, soạn thảo, và xóa những trigơ -sử dụng Enterprise Manager. Bạn làm điều này bởi kích node Tables trong thư mục Databases, rồi kick chuột phải trên bảng Bạn muốn sửa đổi, và rồi chọn *All Tasks* ➢ *Manage Triggers*, . Hình 4.8 cho thấy InsertProductTrigger trong Enterprise Manager. Bạn chú ý -Tôi có thêm một số chú thích vào bắt đầu của mã cho biết những gì mà trigơ làm.



Hình 4.8: sử dụng Enterprise Manager để xem một trigơ

Bộ duyệt đối tượng (Object Browser) của bộ phân tích truy vấn (Query Analyzer) cũng cho phép bạn xem, sửa đổi, và xóa những trigơ.

Mẹo nhỏ: Bạn có thể xóa một trigơ sử dụng phát biểu DROP TRIGGER, và Bạn có thể sửa đổi một Trigơ sử dụng sự phát biểu ALTER TRIGGER.

Thử InsertProductTrigger

Để thử InsertProductTrigger, Bạn chỉ cần thêm một hàng vào bảng những sản phẩm sử dụng một phát biểu INSERT. Chẳng hạn:

```
INSERT INTO Products (
  ProductName, SupplierID, UnitPrice
) VALUES (
  'Widget', 1, 10
)
```

Bạn có thể kiểm tra InsertProductTrigger khởi chạy bởi việc truy xuất những hàng từ bảng ProductAudit sử dụng phát biểu SELECT sau :

```
SELECT *
FROM ProductAudit
```

Hàng được thêm vào bảng ProductAudit bởi InsertProductTrigger như một kết quả của việc thực hiện phát biểu INSERT trước được đưa vào Bảng 4.8.

Bảng 4.8: Hàng thêm vào bảng ProductAudit bởi InsertProductTrigger

ID	ACTION hành động	PERFORMEDBY thực hiện bởi	TOOKPLACE
1	Sản phẩm được thêm vào với ProductID là 80	dbo	2002-07-18 13:55:12.620

Tạo ra và kiểm tra UpdateUnitPriceProductTrigger

Trigơ UpdateUnitPriceProductTrigger khởi chạy sau khi một phát biểu UPDATE được thực hiện trên cột UnitPrice của bảng những sản phẩm. Nếu sự giảm đơn giá của một sản phẩm lớn hơn 25 phần trăm, thì một hàng được thêm vào bảng ProductAudit để kiểm định sự thay đổi. Danh sách 4.8 trình bày một script UpdateUnitPriceProductTrigger.sql.

Danh sách 4.8: UPDATEUNITPRICEPRODUCTTRIGGER.SQL

```
/*
  UpdateUnitPriceProductTrigger.sql creates a trigger
  that fires after an UPDATE statement is performed on the
  the UnitPrice column of the Products table.
  If the reduction of the unit price of a product is
  greater than 25% then a row is added to the ProductAudit table
  to audit the change.
*/

CREATE TRIGGER UpdateUnitPriceProductTrigger
ON Products
AFTER UPDATE
AS

  -- don't return the number of rows affected
```

```
 SET NOCOUNT ON

 -- only run the code if the UnitPrice column
 -- was modified
IF UPDATE(UnitPrice)
BEGIN

 -- declare an int variable to store the
 -- ProductID
 DECLARE @MyProductID int

 -- declare two money variables to store the
 -- old unit price and the new unit price
 DECLARE @OldUnitPrice money
 DECLARE @NewUnitPrice money

 -- declare a float variable to store the price
 -- reduction percentage
 DECLARE @PriceReductionPercentage float

 -- get the ProductID of the row that
 -- was modified from the inserted table
 SELECT @MyProductID = ProductID
 FROM inserted

 -- get the old unit price from the deleted table
 SELECT @OldUnitPrice = UnitPrice
 FROM deleted
 WHERE ProductID = @MyProductID

 -- get the new unit price from the inserted table
 SELECT @NewUnitPrice = UnitPrice
 FROM inserted

 -- calculate the price reduction percentage
 SET @PriceReductionPercentage =
  ((@OldUnitPrice -@NewUnitPrice) / @OldUnitPrice) * 100

 -- if the price reduction percentage is greater than 25%
 -- then audit the change by adding a row to the PriceAudit table
 IF (@PriceReductionPercentage > 25)
 BEGIN

  -- add a row to the ProductAudit table
  INSERT INTO ProductAudit (
    Action
  ) VALUES (
   'UnitPrice of ProductID #' +
     CONVERT(nvarchar, @MyProductID) +
   ' was reduced by ' +
     CONVERT(nvarchar, @PriceReductionPercentage) +
     '%'
   )

  END
 END
```

Có một đôi điều bạn cần phải chú ý về phát biểu CREATE TRIGGER này.

- Mệnh đề AFTER UPDATE chỉ rõ trigơ sẽ khởi chạy sau khi một phát biểu UPDATE được thực hiện.
- Bạn có thể truy xuất những giá trị cột cũ trước khi sự Cập nhật được áp dụng từ bảng đã bị xóa, và bạn có thể truy xuất những giá trị cột mới sau khi sự Cập nhật được áp dụng từ bảng được chèn vào.

Để thử UpdateUnitPriceProductTrigger, Bạn chỉ cần giảm bớt giá trị của cột UnitPrice cho một hàng trong bảng những sản phẩm sử dụng một phát biểu UPDATE. Chẳng hạn, phát biểu UPDATE sau đây nhân UnitPrice với 0.70 cho hàng với ProductID là 80 (điều này giảm bớt UnitPrice của hàng này 30 phần trăm)

```
UPDATE Products
SET UnitPrice = UnitPrice * 0.70
WHERE ProductID = 80
```

Hàng thêm vào bảng ProductAudit như một kết quả của việc thực hiện phát biểu UPDATE này được trình bày trong Bảng 4.9. Hàng này được thêm vào bởi UpdateUnitPriceProductTrigger.

Bảng 4.9: Hàng thêm vào bảng ProductAudit bởi UpdateUnitPriceProductTrigger

ID	ACTION (hành động)	PERFORMEDBY (thực hiện bởi)	TOOKPLACE (thời gian thực hiện)
2	Đơn giá của ProductID #80 được giảm 30%	dbo	2002-07-18 17:26:37.590

Tạo ra và kiểm tra DeleteProductTrigger

Trigơ DeleteProductTrigger khởi chạy sau khi một phát biểu DELETE được thực hiện trên bảng những sản phẩm. Trigơ này thêm một hàng vào bảng ProductAudit để kiểm định sự thay đổi. Danh sách 4.9 trình bày một script DeleteProductTrigger.sql.

Danh sách 4.9: DELETEPRODUCTTRIGGER.SQL

```
/*
  DeleteProductTrigger.sql creates a trigger that fires
  after a DELETE statement is performed on the
  Products table
*/
CREATE TRIGGER DeleteProductTrigger
ON Products
AFTER DELETE
AS

  -- don't return the number of rows affected
  SET NOCOUNT ON

  -- declare an int variable to store the
  -- ProductID
  DECLARE @NewProductID int

  -- get the ProductID of the row that
  -- was removed from the Products table
  SELECT @NewProductID = ProductID
  FROM deleted

  -- add a row to the ProductAudit table
  INSERT INTO ProductAudit (
```

```
Action
) VALUES (
 'Product #' +
  CONVERT(nvarchar, @NewProductID) +
  ' was removed'
)
```

Để thử DeleteProductTrigger, Bạn chỉ cần loại bỏ một hàng từ bảng những sản phẩm sử dụng một phát biểu DELETE . Ví dụ, Phát biểu DELETE sau loại bỏ hàng với ProductID là 80

```
DELETE FROM Products
WHERE ProductID = 80
```

Hàng được thêm vào bảng ProductAudit như một kết quả của việc thực hiện phát biểu DELETE này như trong Bảng 4.10. Hàng này thêm vào bởi DeleteProductTrigger.

Bảng 4.10: Hàng thêm vào bảng ProductAudit bởi DeleteProductTrigger

ID	**ACTION (hành động)**	**PERFORMEDBY (thực hiện bởi)**	**TOOKPLACE (ngày giờ thực hiện)**
3	Sản phẩm #80 đã bị xóa	dbo	2002-07-18 17:35:53.510

Tóm lược

Trong chương này, bạn đã học về lập trình với Transact- SQL. T- SQL cho phép bạn viết những chương trình chứa những câu lệnh SQL, cùng với những cấu trúc lập trình tiêu chuẩn như những biến, lôgic có điều kiện, những vòng lặp, những thủ tục, và những hàm.

SQL Server cung cấp một số hàm bạn có thể sử dụng để lấy những giá trị từ cơ sở dữ liệu. Chẳng hạn, bạn có thể sử dụng hàm COUNT() để lấy số lượng hàng trong một bảng. Bạn đã thấy cách sử dụng những hàm sau : tổng thể, toán học, chuỗi, ngày tháng và thời gian, hệ thống .

Bạn có thể tạo ra những hàm do người dùng định nghĩa của riêng mình trong SQL Server. Chẳng hạn, bạn có thể đã muốn tạo ra hàm của riêng mình để tính toán giá chiết khấu được dựa trên giá gốc và hệ số để nhân với giá gốc này .

SQL Server cho phép bạn lưu giữ những thủ tục trong một cơ sở dữ liệu. Những thủ tục lưu trữ khác với những hàm do người dùng định nghĩa trong đó những thủ tục có thể trả lại một mảng rộng lớn những kiểu dữ liệu. Bạn sẽ điển hình tạo ra một thủ tục lưu trữ khi cần thực hiện một tác vụ mà sử dụng cách mạnh mẽ cơ sở dữ liệu, hay Bạn muốn tập trung mã trong cơ sở dữ liệu mà bất kỳ người dùng nào cũng có thể gọi thay vì mỗi người dùng phải viết chương trình của riêng của mình để thực hiện những tác vụ như nhau.

Trong chương kế tiếp, bạn sẽ học về các lớp ADO.NET

@@@

Chương 5: Tổng quan về những lớp ADO.NET

Tổng quan

ADO.NET cho phép bạn tương tác trực tiếp với một cơ sở dữ liệu sử dụng những đối tượng của những lớp managed provider (nhà cung cấp có quản lý). Những đối tượng này cho phép bạn kế nối tới cơ sở dữ liệu và thực thi những câu lệnh SQL trong lúc kết nối trực tiếp tới cơ sở dữ liệu. Chương trình ví dụ bạn thấy trong Chương 1 chỉ ra cách để kết nối trực tiếp tới một cơ sở dữ liệu và đọc những hàng từ một bảng trong một hướng duy nhất đi tới.

ADO.NET cũng cho phép bạn làm việc trong một tình trạng không kết nối. Khi thực hiện điều này, bạn lưu trữ thông tin từ một cơ sở dữ liệu một cách cục bộ trong bộ nhớ của máy tính trên đó chương trình của bạn đang chạy. Bạn cất giữ thông tin này sử dụng những đối tượng của những lớp Dataset. Một khi bạn có thông tin này trong bộ nhớ, bạn có thể đọc và thao tác với thông tin này. Chẳng hạn, bạn có hiển thị những cột cho những hàng, thêm những hàng mới, sửa đổi những hàng, và xóa những hàng. Một cách định kỳ, bạn sẽ kết nối lại tới cơ sở dữ liệu để đồng bộ hóa (hay cập nhật) những thay đổi mà bạn đã làm cách cục bộ với cơ sở dữ liệu.

Khiểu ngắt kết nối này cho phép bạn viết những ứng dụng chạy trên Internet, cũng như cho những thiết bị không được luôn kết nối tới PDAs cơ sở dữ liệu- ví dụ như Palm và Pocket PC.

Chương này cung cấp những mô tả về những lớp ADO.NET , cũng như một chương trình C# đầy đủ - nó kết nối tới một cơ sở dữ liệu, lưu giữ những hàng vào vùng nhớ cục bộ, rồi ngắt kết nối khỏi cơ sở dữ liệu, và đọc nội dung của những hàng cục bộ này trong tình trạng ngắt kết nối với cơ sở dữ liệu. Khả năng lưu giữ một bản sao cục bộ của những hàng truy xuất được từ cơ sở dữ liệu là một trong số những sức mạnh chính của ADO.NET. Chương trình ví dụ minh họa những ý tưởng cơ bản của việc sử dụng ADO.NET theo kiểu ngắt kết nối để đọc những hàng từ cơ sở dữ liệu và lư trữ chúng một cách cục bộ trong bộ nhớ. Trong những chương sau , bạn sẽ thấy cách để sửa đổi dữ liệu cục bộ và sau đó đồng bộ hóa những sự thay đổi đó với cơ sở dữ liệu như thế nào.

Chương này đặt nền tảng cho Phần II, "Lập trình cơ sở dữ liệu cơ bản với ADO.NET, " Bạn sẽ thấy những chi tiết về những lớp khác nhau của ADO.NET trong phần II này.

Đặc trưng trong chương này:

- Nhà cung cấp được quản lý (Managed Provider) và những lớp Tập dữ liệu chung (Generic Data Set)
- Thực hiện một phát biểu SELECT SQL và cất giữ những hàng cục bộ.

Nhà cung cấp được quản lý và những lớp tập dữ liệu chung

Để cung cấp sự truy cập cơ sở dữ liệu cả có kết nối lẫn không kết nối, ADO.NET định nghĩa hai tập hợp của lớp: Managed Provider (nhà cung cấp được quản lý) và generic data (dữ liệu chung).

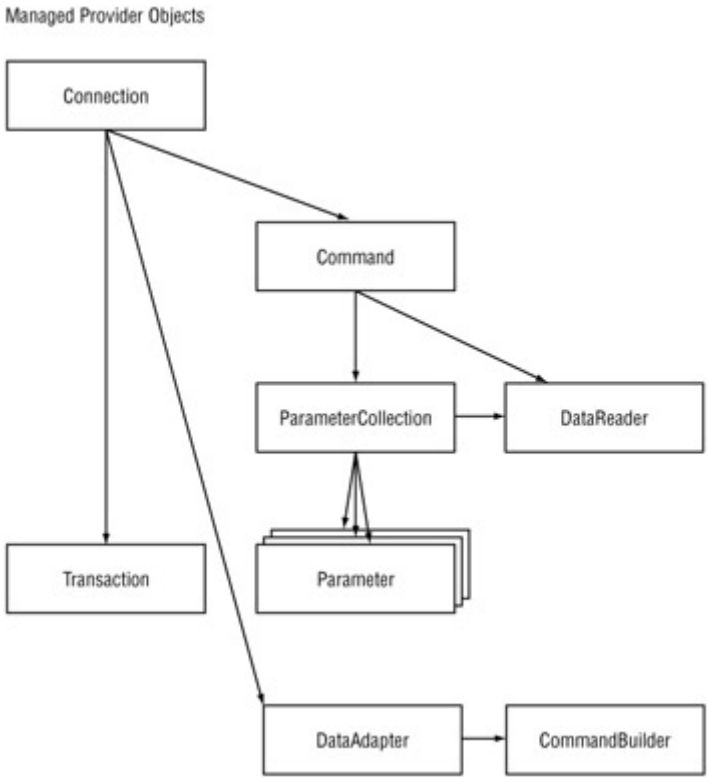
Bạn sử dụng những đối tượng của những lớp managed provider để trực tiếp kết nối tới một cơ sở dữ liệu và để đồng bộ hóa dữ liệu cục bộ được cất giữ của bạn với cơ sở dữ liệu. Bạn có thể sử dụng những lớp managed provider để đọc những hàng từ cơ sở dữ liệu trong một hướng duy nhất đi tới . Bạn sử dụng một tập hợp khác nhau của những lớp managed provider tùy thuộc vào cơ sở dữ liệu bạn đang sử dụng.

Bạn sử dụng những đối tượng của những lớp generic data (dữ liệu chung) để cất giữ một bản sao cục bộ của thông tin được truy xuất từ cơ sở dữ liệu. Bản sao này được lưu trữ trong bộ nhớ của máy tính nơi chương trình C# đang chạy. Lớp dữ liệu chung chính là lớp System.Data.DataSet . Những lớp dữ liệu chung, như tên của gợi ý của chúng, không có gì đặc biệt so với bất kỳ cơ sở dữ liệu nào, và bạn luôn luôn sử dụng cùng những lớp này bất kể cơ sở dữ liệu nào bạn đang sử dụng. Những lớp dữ liệu chung đại diện cho thông tin được truy xuất từ cơ sở dữ liệu như XML.

Những lớp Managed Provider (Nhà cung cấp được quản lý)

Những đối tượng managed provider (nhà cung cấp được quản lý) cho phép bạn trực tiếp truy cập một cơ sở dữ liệu, và bạn sẽ được giới thiệu về những lớp cho phép bạn tạo ra những đối tượng này trong mục này. Bạn sử dụng những đối tượng managed provider để kế nối tới cơ sở dữ liệu , đọc và viết thông tin xuôi ngược tới cơ sở dữ liệu.

Hình 5.1 minh họa một số những đối tượng managed provider (nhà cung cấp được quản lý) và chúng liên quan lẫn nhau như thế nào.



Hình 5.1: Một số những đối tượng nhà cung cấp được quản lý

Hiện thời có ba tập hợp của những lớp managed provider , và mỗi tập hợp được thiết kế để làm việc với những tiêu chuẩn cơ sở dữ liệu khác nhau:

Những lớp **SQL Server Managed Provider** (bộ cung cấp có quản lý SQL Server) : Bạn sử dụng những lớp SQL Server managed provider để kết nối tới một cơ sở dữ liệu SQL Server.

Những lớp **OLE DB Managed Provider** : Bạn sử dụng những lớp DB OLE (Object Linking and Embedding for Databases) managed provider để kết nối với bất kỳ cơ sở dữ liệu nào hỗ trợ OLE DB, như Access hoặc Oracle.

Những lớp ODBC Managed Provider : Bạn sử dụng những lớp ODBC (Open Database Connectivity) managed provider để nối tới bất kỳ cơ sở dữ liệu nào hỗ trợ ODBC. Tất cả những cơ sở dữ liệu chính hỗ trợ ODBC, nhưng ODBC điển hình chậm hơn hai tập hợp lớp trước khi làm việc với. NET. Bạn cần phải sử dụng những lớp ODBC managed provider chỉ khi không còn bất kỳ lớp OLE DB managed provider thay thế nào khác.

Ba tập hợp lớp này đều thực thi cùng chức năng cơ bản như nhau.

> **Ghi chú:**
>
> Bất cứ khi nào mà bạn thấy "Sql" tại khởi đầu của một tên lớp managed provider, bạn biết lớp này được sử dụng với một cơ sở dữ liệu SQL Server. Chẳng hạn, SqlConnection cho phép bạn kết nối tới một cơ sở dữ liệu SQL Server. Tương tự, "OleDb" dành cho những cơ sở dữ liệu hỗ trợ OLE DB. Chẳng hạn, OleDbConnection cho phép bạn kết nối tới một cơ sở dữ liệu sử dụng OLE DB. Cuối cùng, "Odbc" dành cho những cơ sở dữ liệu hỗ trợ ODBC. Chẳng hạn, OdbcConnection cho phép bạn kết nối tới một cơ sở dữ liệu sử dụng ODBC. Tôi tham chiếu tới tất cả những lớp này như những lớp kết nối (Connection).

Bạn sẽ thấy một số lớp managed provider khác nhau trong những mục sau đây.

Những lớp kết nối (Connection)

Có ba lớp kết nối SqlConnection, OleDbConnection, và OdbcConnection. Bạn sử dụng một đối tượng của lớp SqlConnection để kết nối tới một cơ sở dữ liệu Máy chủ phục vụ SQL. Bạn sử dụng một đối tượng của lớp OleDbConnection để kết nối tới bất kỳ cơ sở dữ liệu nào hỗ trợ OLE DB, như uy nhTrập hay Lời tiên tri.Access hay Oracle bạn sử dụng một đối tượng của lớp OdbcConnection để kết nối tới bất kỳ cơ sở dữ liệu nào hỗ trợ ODBC. Cuối cùng, tất cả truyền thông với một cơ sở dữ liệu được thực hiện thông qua một đối tượng kết nối.

Những lớp Command

Có ba lớp Command : SqlCommand, OleDbCommand, và OdbcCommand. Bạn sử dụng một đối tượng lệnh (Command) để chạy một câu lệnh SQL, như một phát biểu SELECT, INSERT, UPDATE, hay DELETE . Bạn cũng có thể sử dụng một đối tượng Command để gọi một thủ tục lưu trữ hay truy xuất những hàng từ một bảng chỉ định. Bạn chạy lệnh được cất giữ trong một đối tượng Command sử dụng một đối tượng Connection.

Những lớp Tham số

Có ba lớp Tham số: SqlParameter, OleDbParameter, và OdbcParameter. Bạn sử dụng một đối tượng Parameter để gởi một tham số tới một đối tượng Command. Bạn có thể sử dụng một Parameter để gởi một giá trị tới một câu lệnh SQL hay một phương thức gọi thủ tục lưu trữ. Bạn có thể lưu giữ nhiều đối tượng tham số (Parameter) trong một đối tượng Command thông qua một đối tượng ParameterCollection (tập hợp tham số).

Những lớp ParameterCollection (tập hợp tham số)

Có ba lớp ParameterCollection: SqlParameterCollection, OleDbParameterCollection, Và OdbcParameterCollection. Bạn sử dụng một đối tượng ParameterCollection để cất giữ nhiều đối tượng tham số cho một đối tượng Command.

Những lớp DataReader

Có ba lớp DataReader: SqlDataReader, OleDbDataReader, và OdbcDataReader. Bạn sử dụng một đối tượng DataReader để đọc những hàng được truy xuất từ cơ sở dữ liệu sử dụng một đối tượng Command.

Những đối tượng DataReader chỉ có thể được dùng để đọc những hàng theo một hướng đi tới. Những đối tượng DataReader đóng vai một giải pháp cho một đối tượng Dataset. Bạn không thể sử dụng một DataReader để sửa đổi những hàng trong cơ sở dữ liệu.

> **Mẹo nhỏ:**
>
> Đọc những hàng sử dụng một đối tượng DataReader điển hình nhanh chóng hơn đọc từ một Dataset.

Những lớp DataAdapter

Có ba lớp DataAdapter: SqlDataAdapter, OleDbDataAdapter, và OdbcDataAdapter. Bạn sử dụng một đối tượng DataAdapter để dời chuyển những hàng giữa một đối tượng Dataset và một cơ sở dữ liệu. Bạn sử dụng một đối tượng DataAdapter để đồng bộ hóa (cập nhật) những hàng được lưu giữ cục bộ của bạn tới cơ sở dữ liệu. Sự đồng bộ hóa này được thực hiện thông qua một đối tượng Connection. 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 một đối tượng Connection.

Những lớp CommandBuilder

Có ba lớp CommandBuilder: SqlCommandBuilder, OleDbCommandBuilder, và OdbcCommandBuilder. Bạn sử dụng một đối tượng CommandBuilder để tự động phát sinh những lệnh (Commands) INSERT, UPDATE, and DELETE bảng đơn, nó đồng bộ hóa bất kỳ sự thay đổi nào bạn thực hiện với một đối tượng Dataset tới cơ sở dữ liệu. Sự đồng bộ hóa này được thực hiện thông qua một đối tượng DataAdapter.

Những lớp Giao dịch (Transaction)

Có ba lớp Giao dịch: SqlTransaction, OleDbTransaction, và OdbcTransaction. Bạn sử dụng một đối tượng Transaction để đại diện cho một Giao dịch cơ sở dữ liệu. Một giao dịch cơ sở dữ liệu là một nhóm những sự phát biểu mà sửa đổi những hàng trong cơ sở dữ liệu. Những phát biểu này được coi như một đơn vị tác vụ lôgíc . Chẳng hạn, trong trường hợp của một giao dịch công việc ngân hàng, bạn có thể đã muốn rút tiền từ một

tài khoản và chuyển tiền vào trong tài khoản khác. Rồi bạn giao phó cả hai sự thay đổi này như một đơn vị, hay nếu sảy ra vấn đề, hồi nguyên cả hai sự thay đổi này.

Không gian tên cho những lớp Nhà cung cấp được quản lý (Managed Provider)

Những lớp managed provider cho SQL Server (SqlConnection vân vân) được khai báo trong không gian tên System .Data.SqlClient . Những lớp cho những cơ sở dữ liệu tương thích OLE DB (SqlDbConnection vân vân) được khai báo trong không gian tên System.Data.OleDb. Những lớp cho những cơ sở dữ liệu tương thích ODBC (OdbcConnection vân vân) được khai báo trong khong gian tên System.Data.Odbc .

> **Ghi chú:**
>
> Vào thời gian viết mã, bạn phải tải những lớp ODBC managed provider từ trang web của Microsoft tại http: // msdn.microsoft.com /Download. việc tải này là riêng biệt từ NET SDK.. Tìm kiếm " ODBC.NET Data Provider" trong mục lục MSDN.

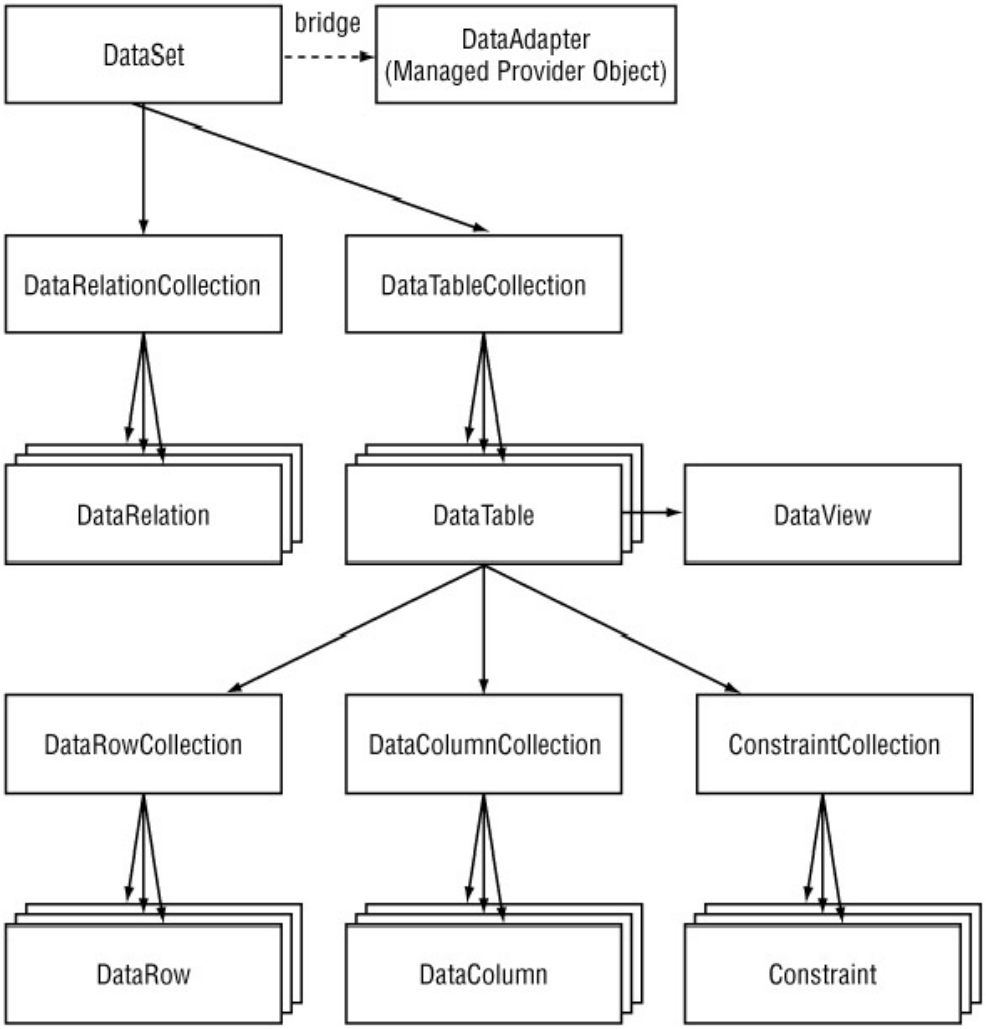
Trong mục sau đây, bạn sẽ học về những lớp dữ liệu chung.

Những lớp Dữ liệu Chung

Như bạn đã học trong mục trước đây, bạn có thể sử dụng những đối tượng managed data provider (bộ cung cấp dữ liệu được quản lý) để kết nối tới cơ sở dữ liệu thông qua một đối tượng Connection, phát hành một câu lệnh SQL thông qua một đối tượng Command, và đọc những hàng được truy xuất sử dụng một đối tượng DataReader; tuy nhiên, bạn có thể đọc những hàng chỉ trong một hướng duy nhất đi tới và bạn phải được kết nối tới cơ sở dữ liệu.
Những đối tượng dữ liệu chung (generic data) cho phép bạn lưu giữ một bản sao cục bộ của thông tin được cất giữ trong cơ sở dữ liệu. Điều này cho phép bạn làm việc với thông tin trong khi ngắt kết nối lới cơ sở dữ liệu. Bạn có thể đọc những hàng trong bất kỳ trật tự nào, và bạn có thể tìm kiếm, phân loại, và lọc những hàng đó trong một cách linh hoạt. Bạn có thể thậm chí sửa đổi những hàng này và sau đó đồng bộ hóa những sự thay đổi tới cơ sở dữ liệu

Hình 5.2 minh họa một số những đối tượng tập dữ liệu chung và chúng liên hệ lẫn nhau như thế nào. Cầu nối giữa bộ cung cấp được quản lý (managed provider) và những đối tượng tập dữ liệu chung (generic data set objects) là DataAdapter, bạn thường dùng nó để đồng bộ hóa những sự thay đổi giữa Dataset của bạn và cơ sở dữ liệu.

Những đối tượng tập dữ liệu chung



Hình 5.2: Một số những đối tượng tập dữ liệu chung

Những mục sau đây phác thảo một số lớp dữ liệu chung.

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 mà bản sao cục bộ trong Datset của bạn và sau đó sau đó đồng bộ hóa những thay đổi đó với cơ sở dữ liệu thông qua một đối tượng DataAdapter bộ cung cấp được quản lý. Một đối tượng 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 việc thi hành những sự ràng buộc khóa chính và khóa ngoại.

Bạn cũng có thể sử dụng một đối tượng Dataset để đại diện cho dữ liệu XML. Trong thực tế, tất cả thông tin được cất giữ trong một Dataset được đại diện cho sử dụng XML, Bao gồm thông tin truy xuất từ cơ sở dữ liệu.

Lớp DataTable

Bạn sử dụng một đối tượng của lớp DataTable để đại diện cho một bảng. Bạn có thể cất giữ nhiều đối tượng DataTable trong một Dataset thông qua một đối tượng DataTableCollection. Một đối tượng Dataset có một thuộc tính tên Tables, mà bạn thường sử dụng nó để truy cập DataTableCollection chứa những đối tượng DataTable được cất giữ trong Dataset này.

Lớp DataRow

Bạn sử dụng một đối tượng của lớp DataRow để đại diện cho một hàng. Bạn có thể cất giữ nhiều đối tượng DataRow trong một DataTable thông qua một đối tượng DataRowCollection. Một đối tượng DataTable có một thuộc tính tên Rows , bạn thường sử dụng nó để truy cập DataRowCollection chứa những đối tượng DataRow được cất giữ trong DataTable này.

Lớp DataColumn

Bạn sử dụng một đối tượng của lớp DataColumn để đại diện cho một cột. Bạn có thể lưu giữ nhiều đối tượng DataColumn trong một DataTable thông qua một đối tượng DataColumnCollection. Một đối tượng DataTable có một thuộc tính tên Columns, bạn thường sử dụng nó để truy cập DataColumnCollection chứa những đối tượng DataColumn được cất giữ trong DataTable này.

Lớp Ràng buộc (Constraint Class)

Bạn sử dụng một đối tượng của lớp Constraint (Ràng buộc) để đại diện cho một sự ràng buộc cơ sở dữ liệu ,nó sẽ kết buộc trên một hoặc nhiều đối tượng DataColumn của một DataTable. Bạn có thể cất giữ nhiều đối tượng Constraint trong một DataTable thông qua một đối tượng ConstraintCollection. Một đối tượng DataTable có một thuộc tính tên Constraints, bạn thường sử dụng nó để truy cập ConstraintCollection chứa những đối tượng Constraints được cất giữ trong DataTable này.

Lớp DataView

Bạn sử dụng một đối tượng của lớp DataView để xem chỉ những hàng chỉ định trong một đối tượng DataTable sử dụng một bộ lọc (filter), nó chỉ rõ tiêu chuẩn để hạn chế những hàng.

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 có thể sử dụng một đối tượng DataRelation để mô hình hóa mối quan hệ cha con giữa hai bảng cơ sở dữ liệu. Bạn có thể lưu giữ nhiều đối tượng DataRelation trong một Dataset thông qua một đối tượng DataRelationCollection. Một đối tượng Dataset có một thuộc tính tên Relations, bạn thường sử dụng nó để truy cập DataRelationCollection chứa những đối tượng DataRelation cất giữ trong Dataset này.

Lớp UniqueConstraint

Bạn sử dụng một đối tượng của lớp UniqueConstraint (ràng buộc khóa chính) để đại diện cho một sự ràng buộc cơ sở dữ liệu, nó bắt buộc giá trị này - được cất giữ trong một đối tượng DataColumn phải là duy nhất. Lớp UniqueConstraint được bắt nguồn từ lớp Constraint . Bạn có thể lưu giữ nhiều đối tượng UniqueConstraint ở một DataTable thông qua một đối tượng ConstraintCollection.

Lớp ForeignKeyConstraint (ràng buộc khóa ngoại)

Bạn sử dụng một đối tượng của lớp ForeignKeyConstraint để chỉ rõ hành động được thực hiện khi những giá trị cột trong bảng cha được cập nhật hay đã bị xóa.

Lớp ForeignKeyConstraint được bắt nguồn từ lớp Constraint. Bạn có thể có cả hai - những hàng con bị xóa (hoạt động liên tầng), và gán những cột con tới null, hay gán những cột con tới một giá trị mặc định. Bạn có thể cất giữ nhiều đối tượng ForeignKeyConstraint trong một DataTable thông qua một đối tượng ConstraintCollection.

Không gian tên cho những lớp Dữ liệu chung

Những lớp Dataset, DataTable, DataRow, DataColumn, DataRelation, Constraint (Sự Ràng buộc), và

DataView đều được khai báo trong Hệ thống. Dữ liệu namespace. Namespace này chứa đựng những lớp khác mà bạn có thể sử dụng trong những chương trình (của) các bạn. Bạn có thể nhìn nhiều tập hợp (của) những lớp khai báo trong không gian tên (namespace) System.Data sử dụng tài liệu .NET.. Chương 1 giải thích cách bạn truy cập tài liệu này như thế nào.

Trong mục kế tiếp bạn sẽ thấy một ví dụ đơn giản minh họa cách phát hành một phát biểu SQL để truy xuất những hàng từ bảng những khách hàng, và rồi cất giữ những hàng được trả về trong một đối tượng Dataset. Chương trình này cho bạn một sự hiểu biết cơ bản về việc làm sao sử dụng một số lớp bộ cung cấp được quản lý và những lớp dữ liệu chung đả phác thảo trước đó. trong phần II, Bạn sẽ thấy những chi tiết của những lớp khác nhau sử dụng trong ví dụ này.

Thực hiện một phát biểu SQL SEL:ECT và lưu giữ những hàng cục bộ.

Trong ví dụ đặc trưng trong mục này, bạn sẽ xem xét cách kết nối tới cơ sở dữ liệu Northwind máy chủ phục vụ SQL và thực hiện một phát biểu SELECT SQL như thế nào để truy xuất những cột CustomerID, CompanyName, ContactName, và Address cho 10 hàng đầu tiên từ bảng những khách hàng như thế nào. Những hàng này được lưu giữ trong một đối tượng Dataset.

Ghi nhớ: Vì tôi sẽ sử dụng một cơ sở dữ liệu máy chủ phục vụ SQL, Tôi sẽ sử dụng những lớp SQL Server managed provider trong ví dụ.

Phác thảo Thủ tục

Bạn có thể sử dụng những bước sau đây để truy xuất những hàng vào trong một đối tượng Dataset :

1. Công thức hóa một chuỗi chứa những chi tiết của kết nối cơ sở dữ liệu.

2. Tạo ra một đối tượng SqlConnection để kết nối tới cơ sở dữ liệu, gởi connection string (chuỗi kết nối) tới bộ khởi dựng.

3. Công thức hóa một chuỗi chứa một phát biểu SELECT để truy xuất những cột cho những hàng từ bảng những khách hàng.

4. Tạo ra một đối tượng SqlCommand để giữ phát biểu SELECT.

5. Gán thuộc tính CommandText của SqlCommand tới chuỗi SELECT.

6. Tạo ra một đối tượng SqlDataAdapter.

7. Gán thuộc tính SelectCommand của đối tượng SqlAdapter tới đối tượng SqlCommand.

8. Tạo ra một đối tượng Dataset để lưu giữ những kết quả của phát biểu SELECT.

9. Mở kết nối cơ sở dữ liệu sử dụng phương thức OPEN() của đối tượng SqlConnection.

10. Phương thức Fill() của đối tượng SqlDataAdapter truy xuất những hàng từ bảng, rồi lưu giữ những hàng cục bộ trong một DataTable của đối tượng Dataset.

11. Đóng kết nối cơ sở dữ liệu sử dụng phương thức Close() của đối tượng SqlConnection được tạo ra trong bước 1.

12. Lấy đối tượng DataTable từ đối tượng Dataset.

13. Hiển thị những cột cho mỗi hàng trong DataTable, sử dụng một đối tượng DataRow để truy cập mỗi hàng trong DataTable.

Trong những mục sau đây bạn sẽ học về những chi tiết của những bước này và xem mã ví dụ.

Bước 1: Công thức hóa một chuỗi chứa những chi tiết của kết nối Cơ sở dữ liệu

Khi kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL, chuỗi kết nối của bạn phải chỉ rõ những yếu tố sau:

- Tên của máy tính trên đó SQL Server đang chạy. Bạn gán tên này trong trong phần Server của chuỗi kết nối. Nếu SQL Server đang chạy trên máy tính cục bộ của bạn, bạn có thể sử dụng localhost làm tên Server. Ví dụ: Server= Localhost.
- Tên của cơ sở dữ liệu. Bạn gán tên này trong phần database của chuỗi kết nối. Ví dụ: database = Northwind.
- Tên người dùng để kết nối tới cơ sở dữ liệu . Bạn đặt tên này trong phần uid của chuỗi. Ví dụ: uid= sa.
- Mật khẩu của người dùng cơ sở dữ liệu. Bạn đặt mật mã này trong phần pwd của chuỗi kết nối. Ví dụ: pwd= sa.

Ghi nhớ: Điển hình, DBA (database administrator: người quản trị cơ sở dữ liệu) của tổ chức của bạn sẽ cung cấp cho bạn những giá trị thích hợp cho chuỗi kết nối. DBA chịu trách nhiệm về điều hành cơ sở dữ liệu.

Ví dụ sau đây tạo ra một chuỗi đặt tên connectionString và gán nó tới một chuỗi thích hợp để kết nối tới cơ sở dữ liệu Northwind đang chạy trên máy tính cục bộ của bạn, sử dụng người dùng sa (với mật khẩu : sa) để kết nối tới cơ sở dữ liệu này:

```
string connectionString =
  "server=localhost;database=Northwind;uid=sa;pwd=sa";
```

Chuỗi kết nối của bạn sẽ khác tùy thuộc cách bạn kết nối tới cơ sở dữ liệu Northwind của bạn như thế nào.

Bước 2: Tạo một đối tượng SqlConnection để kết nối tới Cơ sở dữ liệu

Tạo ra một đối tượng SqlConnection để kết nối tới cơ sở dữ liệu, gởi chuỗi kết nối được tạo ra trong bước trước tới bộ khởi dựng (constructor). Bạn sử dụng một đối tượng của lớp SqlConnection để kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL.

Ví dụ sau đây tạo ra một đối tượng SqlConnection có tên mySqlConnection, chuyển ConnectionString (được tạo ra trong bước 1) tới bộ khởi dựng:

```
SqlConnection mySqlConnection =
  new SqlConnection(connectionString);
```

Bước 3: Công thức hóa một chuỗi chứa phát biểu SELECT

Công thức hóa một chuỗi chứa phát biểu SELECT để truy xuất những cột CustomerID, CompanyName, ContactName, và Address cho 10 hàng đầu tiên từ bảng những khách hàng. Chẳng hạn:

```
string selectString =
  "SELECT TOP 10 CustomerID, CompanyName, ContactName, Address "+
  "FROM Customers "+
  "ORDER BY CustomerID";
```

Ghi nhớ: Bạn sử dụng từ khóa TOP trong sự kết hợp với một mệnh đề ORDER BY để truy xuất N hàng đầu tiên từ một phát biểu SELECT. Bạn có thể học nhiều hơn về từ khóa TOP trong Chương 3.

Bước 4: Tạo ra một đối tượng SqlCommand để giữ phát biểu SELECT

Bạn có thể gọi phương thức CreateCommand() của mySqlConnection để tạo ra một đối tượng SqlCommand mới cho kết nối này. phương thức CreateCommand() trả về một đối tượng SqlCommand mới cho đối tượng SqlConnection.

Trong ví dụ sau đây, một đối tượng SqlCommand mới có tên mySqlCommand được gán tới đối tượng SqlCommand được trả về bởi việc gọi phương thức CreateCommand() của mySqlConnection:

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

Bước 5: gán thuộc tính CommandText của đối tượng SqlCommand tới chuỗi SELECT

Gán thuộc tính CommandText của đối tượng SqlCommand của bạn tới chuỗi SELECT được tạo ra trong bước 4. Thuộc tính CommandText chứa câu lệnh SQL bạn muốn thực hiện. Trong ví dụ sau đây, thuộc tính CommandText của mySqlCommand được gán tới selectString:

```
mySqlCommand.CommandText = selectString;
```

Bước 6: Tạo ra một đối tượng SqlDataAdapter

Bạn sử dụng một đối tượng SqlDataAdapter để dời chuyển thông tin giữa đối tượng Dataset của bạn và cơ sở dữ liệu. Bạn sẽ thấy cách tạo ra một đối tượng Dataset trong bước 8. Ví dụ sau đây tạo ra một đối tượng SqlDataAdapter có tên mySqlDataAdapter:

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

Bước 7: Gán thuộc tính SelectCommand của đối tượng SqlAdapter tới đối tượng SqlCommand

Thuộc tính SelectCommand chứa phát biểu SELECT bạn muốn chạy. Trong ví dụ sau đây, thuộc tính SelectCommand của mySqlDataAdapter được gán tới mySqlCommand:

```
mySqlDataAdapter.SelectCommand = mySqlCommand;
```

Bước 8: Tạo ra một đối tượng Dataset để lưu giữ những kết quả của phát biểu SELECT

Bạn sử dụng một đối tượng Dataset để lưu giữ một bản sao cục bộ của thông tin được truy xuất từ cơ sở dữ liệu. Ví dụ sau đây tạo ra một đối tượng Dataset có tên myDataSet:

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

Bước 9: Mở kết nối cơ sở dữ liệu sử dụng phương thức Open() của đối tượng SQLConnection

Ví dụ sau đây gọi phương thức Open() cho mySqlConnection:

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

Một khi bạn mở kết nối cơ sở dữ liệu, bạn có thể truy cập cơ sở dữ liệu.

Bước 10: Gọi phương thức Fill() của đối tượng SqlDataAdapter để truy xuất những hàng từ bảng

Gọi phương thức Fill() của đối tượng SqlDataAdapter của bạn để truy xuất những hàng từ cơ sở dữ liệu, lưu giữ những hàng này cục bộ trong một DataTable của đối tượng Dataset của bạn.

Phương thức Fill() là quá tải, và phiên bản bạn sẽ thấy trong ví dụ chấp nhận hai tham số:

- Một đối tượng Dataset
- Một chuỗi chứa tên của đối tượng DataTable được tạo ra trong Dataset .

Phương thức Fill() tạo ra một DataTable trong Dataset với tên chỉ định và chạy phát biểu SELECT. DataTable được tạo ra trong Dataset của bạn rồi được cư trú với những hàng được truy xuất bởi phát biểu SELECT.

Ví dụ sau đây gọi phương thức Fill() của mySqlDataAdapter, chuyển MyDataSet và " Customers " tới phương thức Fill():

```
mySqlDataAdapter.Fill(myDataSet, "Customers");
```

Phương thức Fill() tạo ra một đối tượng DataTable ở myDataSet tên Customers trong myDataset và cư trú nó với những hàng được truy xuất bởi phát biểu SELECT. Bạn có thể truy cập những hàng này, Ngay cả khi ngắt kết nối với cơ sở dữ liệu.

Bước 11: Đóng kết nối với Cơ sở dữ liệu

Đóng kết nối cơ sở dữ liệu sử dụng phương thức Close() của đối tượng SqlConnection được tạo ra trong bước đầu tiên. Chẳng hạn:

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

Ghi nhớ: tất nhiên, bạn không cần phải đóng ngay lập tức kết nối cơ sở dữ liệu trước khi đọc những hàng được lưu giữ cục bộ từ Dataset của bạn. Tôi đóng kết nối tại điểm này trong ví dụ để chỉ cho biết là quả thực bạn có thể đọc những hàng được lưu giữ cục bộ - thậm chí khi đã ngắt ra khỏi cơ sở dữ liệu.

Bước 12: Lấy đối tượng DataTable từ đối tượng Dataset

Lấy đối tượng DataTable được tạo ra trong bước 10 từ đối tượng Dataset.

Bạn lấy một DataTable từ Dataset của bạn sử dụng thuộc tính Tables, nó trả lại một đối tượng DataTableCollection . Để lấy DataTable riêng lẻ từ Dataset của bạn, bạn gởi tên của DataTable của bạn trong cặp dấu móc ("Customers ", chẳng hạn) tới thuộc tính Tables. Thuộc tính Tables sẽ trả về DataTable mà bạn yêu cầu, bạn có thể cất giữ nó trong một đối tượng DataTable mới mà bạn khai báo. Trong ví dụ sau đây, myDataSet.Tables ["Customers "] trả về Customers DataTable được tạo ra trong myDataSet trong bước 10, và lưu giữ DataTable được trả về trong myDataTable:

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

Ghi nhớ: Bạn cũng có thể chỉ rõ DataTable bạn muốn có bởi gởi một giá trị số tới những thuộc tính Tables . Chẳng hạn, myDataSet.Table [0] cũng trả về Customers DataTable.

Bước 13: Trình bày những cột cho mỗi hàng trong DataTable

Trình bày những cột cho mỗi hàng trong DataTable, sử dụng một đối tượng DataRow để truy cập mỗi hàng trong DataTable. Lớp DataTable định nghĩa một thuộc tính có tên Rows nó trả về một đối tượng DataRowCollection chứa những đối tượng DataRow cất giữ trong DataTable này. Bạn có thể sử dụng những thuộc tính Rows trong một vòng lặp foreach để lặp lại qua những đối tượng DataRow. Chẳng hạn:

```
foreach (DataRow myDataRow in myDataTable.Rows)
{
  // ... access the myDataRow object
}
```

Mỗi đối tượng DataRow cất giữ những đối tượng DataColumn chứa những giá trị được truy xuất từ những cột của bảng trong cơ sở dữ liệu. Bạn có thể truy cập những giá trị cột này bởi việc gởi tên của cột trong cặp dấu móc đôi tới đối tượng DataRow. Chẳng hạn, myDataRow[" CustomerID "] trả về giá trị của cột CustomerID.

Trong ví dụ sau đây, một vòng lặp foreach lặp lại qua những đối tượng DataRow trong myDataTable, và những giá trị cột được trình bày cho mỗi hàng.

```
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"]);
}
```

Như bạn có thể thấy, tên của mỗi cột được gởi đi trong những dấu móc tới mỗi đối tượng DataRow, mà rồi trả về giá trị cột.

> **Ghi nhớ:** Bạn cũng có thể chỉ rõ cột bạn muốn có bởi việc gởi đi một giá trị số trong cặp dấu móc đôi. Chẳng hạn, myDataRow[0] cũng trả về giá trị cột CustomerID.

Đặt mọi thứ cùng nhau

Danh sách 5.1 cho thấy một chương trình đầy đủ sử dụng những bước này. Chương trình này có tên SelectIntoDataSet.cs và được định vị trong thư mục ch05.

Danh sách 5.1: SELECTINTODATASET.CS

```
/*
 SelectIntoDataSet.cs illustrates how to perform a SELECT statement
 and store the returned rows in a DataSet object
*/

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

class SelectIntoDataSet
{
 public static void Main()
 {
  // step 1: formulate a string containing the details of the
  // database connection
  string connectionString =
   "server=localhost;database=Northwind;uid=sa;pwd=sa";

  // step 2: create a SqlConnection object to connect to the
  // database, passing the connection string to the constructor
  SqlConnection mySqlConnection =
   new SqlConnection(connectionString);

  // step 3: formulate a SELECT statement to retrieve the
  // CustomerID, CompanyName, ContactName, and Address
  // columns for the first ten rows from the Customers table
  string selectString =
   "SELECT TOP 10 CustomerID, CompanyName, ContactName, Address "+
```

```
            "FROM Customers " +
            "ORDER BY CustomerID";

        // step 4: create a SqlCommand object to hold the SELECT statement
        SqlCommand mySqlCommand = mySqlConnection.CreateCommand();

        // step 5: set the CommandText property of the SqlCommand object to
        // the SELECT string
        mySqlCommand.CommandText = selectString;

        // step 6: create a SqlDataAdapter object
        SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();

        // step 7: set the SelectCommand property of the SqlAdapter object
        // to the SqlCommand object
        mySqlDataAdapter.SelectCommand = mySqlCommand;
        // step 8: create a DataSet object to store the results of
        // the SELECT statement
        DataSet myDataSet = new DataSet();

        // step 9: open the database connection using the
        // Open() method of the SqlConnection object
        mySqlConnection.Open();

        // step 10: 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 Customers table");
        mySqlDataAdapter.Fill(myDataSet, "Customers");

        // step 11: close the database connection using the Close() method
        // of the SqlConnection object created in Step 1
        mySqlConnection.Close();

        // step 12: get the DataTable object from the DataSet object
        DataTable myDataTable = myDataSet.Tables["Customers"];

        // step 13: display the columns 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("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
CustomerID = ALFKI
CompanyName = Alfreds Futterkiste
ContactName = Maria Anders
```

Address = Obere Str. 57
CustomerID = ANATR
CompanyName = Ana Trujillo Emparedados y helados
ContactName = Ana Trujillo
Address = Avda. de la Constitución 2222
CustomerID = ANTON
CompanyName = Antonio Moreno Taquería
ContactName = Antonio Moreno
Address = Mataderos 2312
CustomerID = AROUT
CompanyName = Around the Horn
ContactName = Thomas Hardy
Address = 120 Hanover Sq.
CustomerID = BERGS
CompanyName = Berglunds snabbköp
ContactName = Christina Berglund
Address = Berguvsvägen 8
CustomerID = BLAUS
CompanyName = Blauer See Delikatessen
ContactName = Hanna Moos
Address = Forsterstr. 57
CustomerID = BLONP
CompanyName = Blondesddsl père et fils
ContactName = Frédérique Citeaux
Address = 24, place Kléber
CustomerID = BOLID
CompanyName = Bólido Comidas preparadas
ContactName = Martín Sommer
Address = C/ Araquil, 67
CustomerID = BONAP
CompanyName = Bon app'
ContactName = Laurence Lebihan
Address = 12, rue des Bouchers
CustomerID = BOTTM
CompanyName = Bottom-Dollar Markets
ContactName = Elizabeth Lincoln
Address = 23 Tsawassen Blvd.

Tóm lược

Trong chương này, bạn đã có một tổng quan về những lớp ADO.NET , và bạn đã khảo sát một chương trình đầy đủ kết nối tới một cơ sở dữ liệu, cất giữ những hàng cục bộ, ngắt kết nối cơ sở dữ liệu, và đọc nội dung của những hàng cục bộ này trong khi ngắt kết nối với cơ sở dữ liệu.

ADO.NET cho phép bạn tương tác trực tiếp với một cơ sở dữ liệu sử dụng những đối tượng của những lớp managed provider (nhà cung cấp được quản lý). Những đối tượng này cho phép bạn kết nối tới cơ sở dữ liệu và thực hiện những câu lệnh SQL trong khi kết nối trực tiếp tới cơ sở dữ liệu. Bạn sử dụng những tập hợp khác nhau của những lớp managed provider , phụ thuộc vào cơ sở dữ liệu Bạn sử dụng.

ADO.NET cũng cho phép bạn làm việc trong một trạng thái không kết nối. Khi làm điều này, bạn lưu trữ thông tin từ một cơ sở dữ liệu vào trong bộ nhớ của máy tính trên đó chương trình của bạn đang chạy. Bạn lưu giữ thông tin này sử dụng những đối tượng của những lớp Dataset.

Một số lớp "bộ cung cấp có quản lý SQL Server" bao gồm : SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter, và SqlTransaction. Bạn sử dụng một đối tượng của lớp SqlConnection để kết nối tới một cơ sở dữ liệu Máy chủ phục vụ SQL. Bạn sử dụng một đối tượng của lớp SqlCommand để đại diện cho một câu lệnh

SQL hay sự gọi thủ tục lưu trữ mà bạn sẽ thực thi. Bạn sử dụng một đối tượng của lớp SqlDataReader để đọc những hàng truy xuất được từ một cơ sở dữ liệu Máy chủ phục vụ SQL. Bạn sử dụng một đối tượng của lớp SqlDataAdapter để dời chuyển những hàng giữa một đối tượng Dataset và một cơ sở dữ liệu Máy chủ phục vụ SQL.

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 một cơ sở dữ liệu. Bạn cũng có thể sử dụng một đối tượng Dataset để đại diện cho dữ liệu XML. Vài đối tượng bạn có thể cất giữ trong một Dataset bao gồm những đối tượng : DataTable, DataRow, DataColumn, DataRelation, và DataView .

Trong Chương 6, bạn sẽ học sử dụng Visual Studio .NET như thế nào để tạo ra những chương trình Windows.

Chương 6: Giới thiệu những ứng dụng Windows và ADO.NET

Tổng quan

Trong những chương trước đây, bạn chạy những chương trình sử dụng công cụ Command Prompt (dấu nhắc lệnh) Windows. Trong chương này, bạn sẽ được giới thiệu tới những ứng dụng Windows. Windows cung cấp những phần tử giao diện đồ họa, như những thực đơn, những hộp văn bản, và những nút radiô, để bạn có thể xây dựng một giao diện trực quan rất dễ sử dụng. Bạn có thể tạo ra những ứng dụng Windows sử dụng ADO.NET, và bạn sẽ học cách làm điều đó - sử dụng Visual Studio .NET (VS .NET) như thế nào, trong chương này.

Những ứng dụng Windows là rất đơn giản để học và sử dụng bởi vì mọi người đã trở nên quen thuộc cách tương tác với những máy trong một giao diện trực quan. Ở khắp mọi nơi - Microsoft Word và Excel thực sự là hai ví dụ về sự thành công của những ứng dụng Windows như nthế nào ,có thể là bởi vì chúng kết hợp cả hai sức mạnh và sự dễ dàng sử dụng .

Đặc trưng trong chương này:

- Phát triển một ứng dụng Windows đơn giản
- Sử dụng những điều khiển Windows
- Truy cập một cơ sở dữ liệu với một điều khiển DataGrid
- Tạo ra một Windows Form với Data Form Wizard

Phát triển một ứng dụng Windows Đơn giản

Trong mục này bạn sẽ thấy cách tạo ra một ứng dụng Windows đơn giản sử dụng VS .NET. Ứng dụng này sẽ gồm có một form đơn chứa một nhãn và một nút. Khi bạn kích nút, văn bản trong nhãn sẽ thay đổi tới một lời trích dẫn từ vở kịch của Shakespeare, Macbeth. Bạn cũng sẽ thấy cách biên dịch và chạy ứng dụng ví dụ như thế nào.

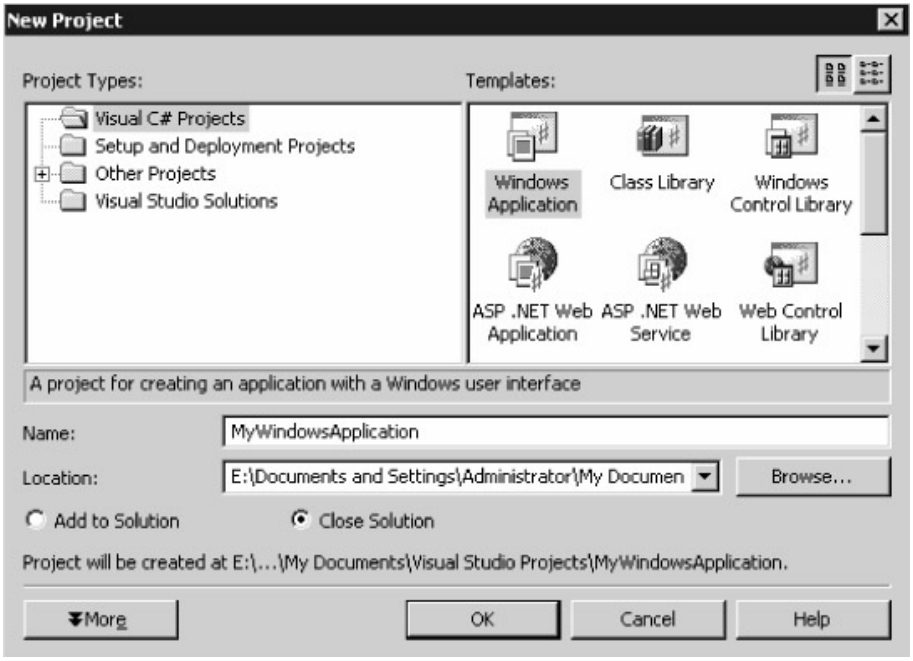
Tạo ra ứng dụng Windows

Khởi chạy VS .NET bởi chọn Start ➢Programs ➢Microsoft Visual Studio .NET . Để tạo ra một ứng dụng Windows mới, kích nút New Project trên trang Start page, hay chọn File ➢New ➢Project.

Mẹo nhỏ: Bạn cũng có thể tạo ra một dự án mới bằng cách nhấn Ctrl+ Shift+ N trên bàn phím của bạn.

Bạn sẽ thấy hộp thoại New Project, mà bạn sử dụng để lựa chọn kiểu Dự án muốn tạo ra. Bởi vì bạn sẽ tạo ra một ứng dụng Windows C#, do đó bạn chọn thư mục Visual C# Projects từ danh sách Project Types, và chọn

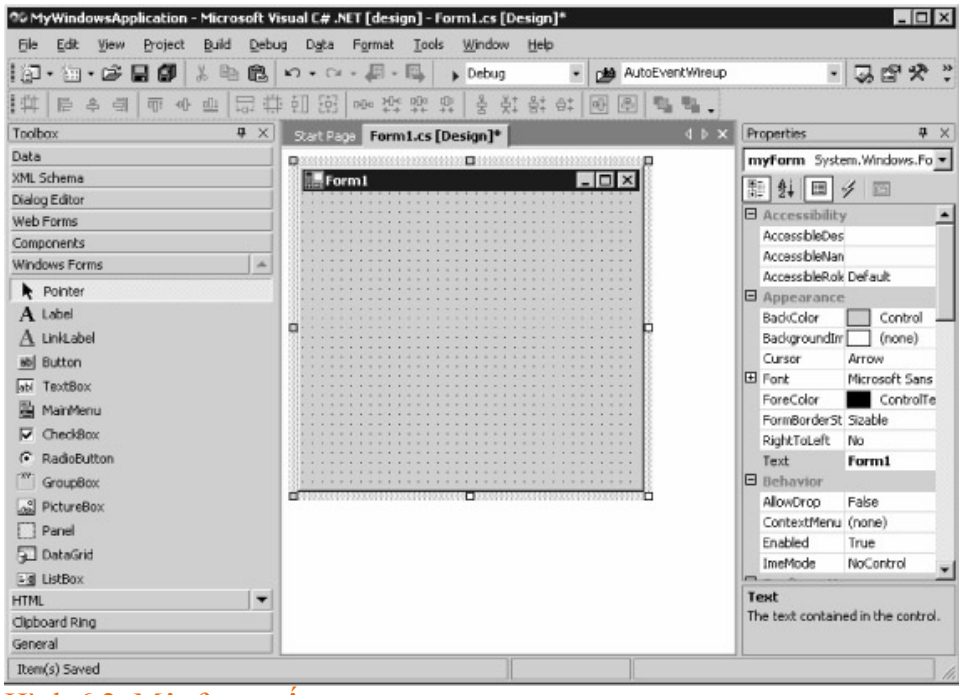
Windows Application từ Templates area (vùng khuôn mẫu) của hộp thoại New Project (Dự án mới). VS .NET sẽ gán một tên mặc định cho dự án của bạn; tên mặc định này là WindowsApplication1, hay cái gì đó tương tự. Bạn có thể chỉ định tên bạn muốn cho dự án của bạn bởi thay đổi văn bản trong trường Name; như trình bày trong hình 6.1



Hình 6.1: Tạo ra một ứng dụng Windows C# trong Visual Studio .NET

Ghi nhớ: Trường Location (vị trí) chỉ rõ thư mục nơi những files thuộc dự án mới của bạn được cất giữ. VS .NET sẽ gán một thư mục mặc định, nhưng bạn có thể thay đổi điều này bằng cách nhập vào thư mục riêng của mình. Thư mục mặc định này là Documents and Settings trên ổ cứng của bạn.

Kích nút Ok để tiếp tục. VS .NET sẽ tạo ra một thư mục con mới có tên MyWindowsApplication trong thư mục được chỉ rõ trong trường Location (vị trí). Một khi VS .NET tạo ra thư mục, cùng với một số file ban đầu cho dự án của bạn, VS .NET sẽ hiển thị một form trống, như trình bày trong Hình 6.2. Bạn có thể hiểu form này như tấm vải bạt mà trên đó bạn có thể sắp đặt những điều khiển Windows tiêu chuẩn, như những nhãn, những hộp văn bản, và những nút nhấn. Bạn sẽ thêm những điều khiển vào Form của bạn không lâu sau đây.



Hình 6.2: Một form trống

Làm việc với Toolbox (hộp công cụ)

Bạn thêm những điều khiển vào form của bạn bởi chọn điều khiển từ Toolbox và kéo vào form của bạn. Bạn cũng có thể kích và kéo , hay nhấn đúp vào điều khiển để thả một một cái mới của kiểu này lên trên form. Như bạn có thể nhìn thấy trong Hình 6.2 được trình bày trước đó, Toolbox ở bên trái của form trống này.

> **Ghi chú**: Nếu bạn không nhìn thấy Toolbox (hộp công cụ), Bạn có thể hiển thị nó bởi chọn View ➢ Toolbox, hay nhấn Ctrl+ Alt+ X trên bàn phím.

Bạn có thể thấy những phần tử sẵn có trong Toolbox được phân loại vào trong những nhóm với những tên như Data (Dữ liệu) và XML Schema (Mô hình XML). Toolbox sẽ hiện ra chỉ những phạm trù liên quan tới kiểu ứng dụng bạn đang phát triển. Danh sách sau đây mô tả nội dung của một số những phạm trù này:

Data: phạm trù Dữ liệu chứa những lớp cho phép bạn truy cấp và cất giữ thông tin từ một cơ sở dữ liệu. Phạm trù Dữ liệu bao gồm những lớp sau đây: SqlConnection, SqlCommand, Dataset, và DataView, và những thứ khác liên quan đến dữ liệu.

XML Schema: phạm trù mô hình XML chứa những lớp phép bạn truy cập dữ liệu XML.

Dialog Editor: phạm trù "Bộ biên tập hộp thoại" chứa những điều khiển mà bạn có thể đặt trên những hộp thoại Windows.

Web Forms: những phạm trù Web Forms chứa những điều khiển dành cho việc hình thành những trang web. Bạn có thể thiết kế những form web sử dụng VS .NET và triển khai chúng tới Người phục vụ thông tin Internet (IIS) của Microsoft. Những form web này có thể sẽ được chạy thông qua Internet.

Components: phạm trù những thành phần chứa những lớp như: FileSystemWatcher, nó cho phép bạn theo dõi những sự thay đổi trong một hệ thống tập tin máy tính. Những lớp khác bao gồm EventLog, DirectoryEntry, DirectorySearcher, MessageQueue, PerformanceCounter, Process, ServiceController, và Timer. Chúng cho phép bạn thực hiện nhiều thao tác hệ thống.

Windows Forms: phạm trù của những form Windows chứa những điều khiển mà bạn có thể thêm vào một form Windows . chúng bao gồm những nhãn (label), những nút nhấn (button), và những hộp văn bản (text boxe) và một số điều khiển khác. Bạn sẽ sử dụng một số những điều khiển trong chương này.

HTML: phạm trù HTML chứa những điều khiển mà bạn có thể thêm vào một form web. Chúng bao gồm những nhãn , những nút nhấn, những bảng, và những hình ảnh, và những điều khiển khác nữa trên form web.

Trong mục kế tiếp, bạn sẽ học về những cửa sổ thuộc tính.

Làm việc với những cửa sổ thuộc tính

Những cửa sổ thuộc tính (Properties window) chứa những khía cạnh của một điều khiển mà bạn có thể thiết đặt. Chẳng hạn, bạn có thể gán màu nền của form sử dụng thuộc tính BackColor. Một số thuộc tính khác của điều khiển trên form bao gồm ForeColor (màu nền) và BackgroundImage (một ảnh nền). Những kiểu điều khiển khác nhau có những kiểu thuộc tính khác nhau.

Như bạn có thể nhìn thấy từ Hình 6.2 được trình bày trước đó, những cửa sổ thuộc tính ở bên phải của form trống.

> **Ghi chú:**
>
> Nếu bạn không nhìn thấy cửa sổ những thuộc tính, bạn có thể hiển thị nó bởi chọn View □ Properties Window, hay bởi nhấn F4 trên bàn phím .

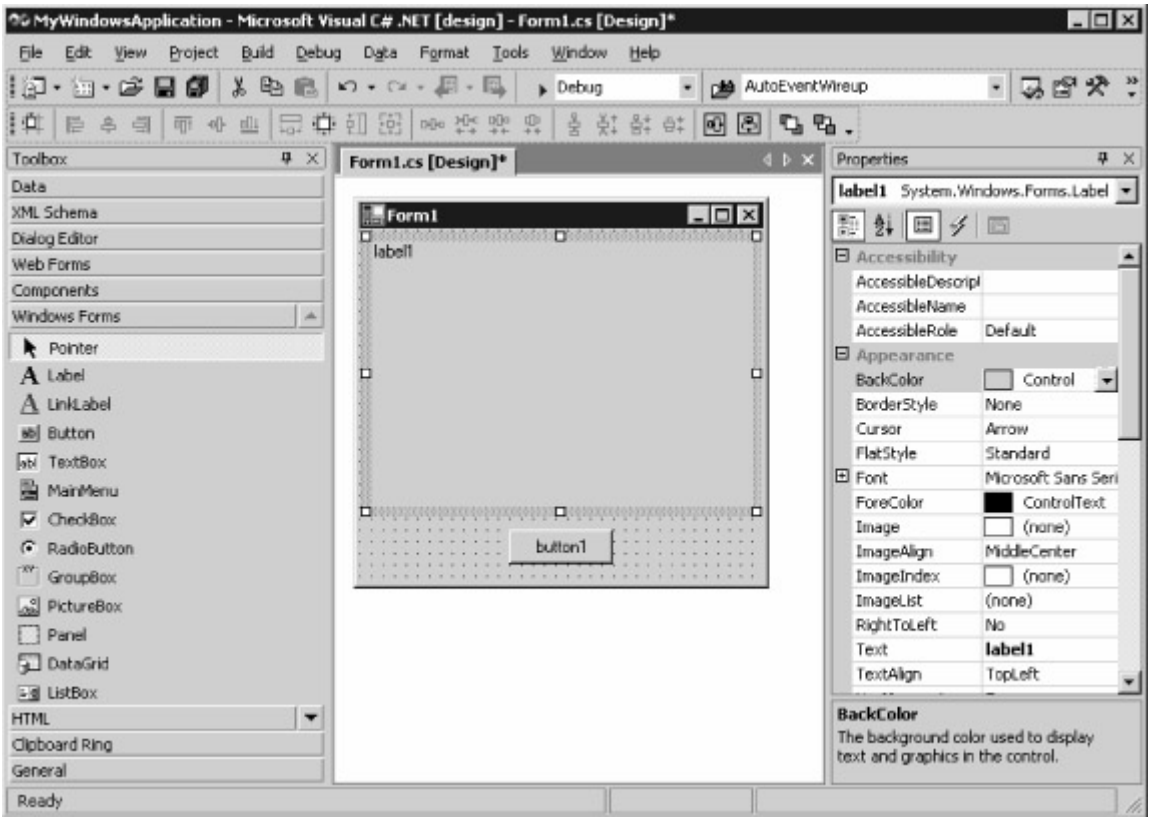
Bạn thiết đặt thuộc tính bởi kích vùng bên phải của tên thuộc tính. hãy bắt đầu và kích vùng bên phải của thuộc tính BackColor để xem vài màu sắc mà bạn có thể kich chọn để thiết đặt thuộc tính này.

Trong mục kế tiếp, bạn sẽ học cách thêm một điều khiển nhãn và nút nhấn vào form của bạn như thế nào. Bạn cũng sẽ đặt một cặp thuộc tính cho những điều khiển này.

Thêm một Nhãn và một điều khiển Nút

Để thêm một nhãn và một điều khiển nút vào form của bạn chọn điều khiển thích hợp từ Toolbox và kéo nó đến form của bạn. Chẳng hạn, để thêm một nhãn vào form bạn, bạn chọn điều khiển nhãn (label) từ Toolbox. Một khi bạn kéo một nhãn tới form , bạn có thể thay đổi kích thước nó bằng cách sử dụng con chuột hay gán thuộc tính Size (kích thước) trong cửa sổ những thuộc tính (Properties window). Bạn cũng có thể kích vào nhãn trong Toolbox và kéo nó lên form.

Làm cho nhãn của bạn đủ lớn sao cho nó trải ra theo chiều dài của form. Tiếp theo, thêm một điều khiển nút ở bên dưới nhãn của bạn, như trong Hình 6.3.



Hình 6.3: Form với một nhãn và một điều khiển nút

Tiếp theo, bạn sẽ thay đổi một số thuộc tính cho nhãn và nút của bạn. Bạn làm điều này bởi sử dụng cửa sổ những thuộc tính. Đặt thuộc tính *Name* của nhãn là *myLabel*. Gán những thuộc tính **Name** và *Text* cho nút của bạn là *myButton* và *Press me!*, tương ứng. Đồng thời, đặt thuộc tính *Text* của form của bạn là *My Form*.

Ghi chú:

Bạn sử dụng thuộc tính Name khi tham chiếu đến một điều khiển Windows trong mã C#.

Tiếp theo, bạn sẽ thêm một dòng mã tới phương thức myButton_Click() . Phương thức này được thực thi khi nút myButton được kích trong form đang chạy của bạn. Câu lệnh bạn sẽ thêm vào myButton_Click() để gán thuộc tính Text của myLabel tới một chuỗi. Chuỗi này sẽ chứa đựng một dòng từ vở kịch của Shakespeare, Macbeth. Để thêm mã, nhấn đúp nút myButton và nhập vào mã sau đây trong vùng mã của phương thức myButton_Click() :

```
myLabel.Text =
  "Is this a dagger which I see before me,\n" +
  "The handle toward my hand? Come, let me clutch thee.\n" +
  "I have thee not, and yet I see thee still.\n" +
  "Art thou not, fatal vision, sensible\n" +
  "To feeling as to sight? or art thou but\n" +
  "A dagger of the mind, a false creation,\n" +
  "Proceeding from the heat-oppressed brain?";
```

Ghi chú:

Nếu bạn là một người hâm mộ Shakespeare, bạn sẽ đoán nhận hàng này từ cảnh trước khi Macbeth giết chết Vua Duncan.

Bây giờ bạn đã hoàn tất form của bạn. Và hãy xây dựng dự án của bạn bởi Build ➢Build Solution, hay bởi nhấn Ctrl+ Shift+ B trên bàn phím.

Để chạy form , bạn chọn Debug ➢Start without Debugging , hay nhấn Ctrl+ F5 trên bàn phím .

Mẹo nhỏ:

Bạn có thể sử dụng một phím tắt khi xây dựng và chạy form của bạn: nếu Bạn đơn giản chạy form của bạn mà không xây dựng nó trước (Build Solution), VS .NET sẽ kiểm tra xem phải chăng bạn đã có thực hiện bất kỳ sự thay đổi nào tới form của bạn từ lần sau cùng chạy nó . Nếu bạn đã thực hiện một sự thay đổi, thì VS .NET sẽ trước tiên xây dựng lại dự án của bạn rồi sau đó mới chạy nó.

Hình 6.4 trình bày form được chạy sau khi nút Press me được kích.

Hình 6.4: Form đang chạy

Bây giờ bạn đã được tạo ra và chạy form, chúng ta hãy xem xét mã được phát sinh bởi VS .NET cho chúng. Mã C# cho form của bạn được chứa trong file Form1.cs . Bạn sẽ khảo sát mã này trong mục kế tiếp.

Khảo sát Mã bên dưới Form

File Form1.cs chứa mã cho form của bạn. Mã này thường được tham chiếu đến như mã bên dưới form của bạn, bởi vì bạn có thể hiểu nó như một thứ bên dưới sự thiết kế trực quan cho form. Bạn có thể xem mã form của bạn bởi chọn View Code, hay bởi nhấn khóa F7 trên bàn phím . Danh sách 6.1 cho thấy nội dung của file Form1.cs.

Danh sách 6.1: Form1.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace MyWindowsApplication
{
 /// <summary>
 /// Summary description for Form1.
 /// </summary>
 public class Form1 : System.Windows.Forms.Form
 {
```

```
private System.Windows.Forms.Label myLabel;
private System.Windows.Forms.Button myButton;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
 //
 // Required for Windows Form Designer support
 //
 InitializeComponent();

 //
 // TODO: Add any constructor code after InitializeComponent call
 //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose(bool disposing)
{
 if(disposing)
 {
  if (components != null)
  {
   components.Dispose();
  }
 }
 base.Dispose(disposing);
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
 this.myLabel = new System.Windows.Forms.Label();
 this.myButton = new System.Windows.Forms.Button();
 this.SuspendLayout();
 //
 // myLabel
 //
 this.myLabel.Location = new System.Drawing.Point(8, 8);
 this.myLabel.Name = "myLabel";
 this.myLabel.Size = new System.Drawing.Size(288, 184);
 this.myLabel.TabIndex = 0;
 this.myLabel.Text = "label1";
 //
 // myButton
 //
 this.myButton.Location = new System.Drawing.Point(120, 200);
```

```
            this.myButton.Name = "myButton";
            this.myButton.Size = new System.Drawing.Size(72, 24);
            this.myButton.TabIndex = 1;
            this.myButton.Text = "Press Me!";
            this.myButton.Click += new System.EventHandler(this.myButton_Click);
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(304, 237);
            this.Controls.AddRange(new System.Windows.Forms.Control[] {
              this.myButton,
              this.myLabel});
            this.Name = "Form1";
            this.Text = "My Form";
            this.ResumeLayout(false);

        }
        #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
          Application.Run(new Form1());
        }

        private void myButton_Click(object sender, System.EventArgs e)
        {
          myLabel.Text =
            "Is this a dagger which I see before me,\n" +
            "The handle toward my hand? Come, let me clutch thee.\n" +
            "I have thee not, and yet I see thee still.\n" +
            "Art thou not, fatal vision, sensible\n" +
            "To feeling as to sight? or art thou but\n" +
            "A dagger of the mind, a false creation,\n" +
            "Proceeding from the heat-oppressed brain?";

        }
      }
    }
```

Như bạn có thể thấy, lớp Form1 được bắt nguồn từ lớp System.Windows.Forms.Form . Lớp Form đại diện cho một form Windows.

Ghi chú:

Không gian tên System.Windows.Forms chứa nhiều lớp khác nhau để tạo ra những ứng dụng Windows. Hầu hết những lớp trong namespace này được bắt nguồn từ lớp System.Windows.Forms.Control; lớp này cung cấp chức năng cơ bản cho những điều khiển bạn có thể đặt trên một form.

Lớp Form1 khai báo hai đối tượng riêng (Private) có tên myLabel và myButton, chúng là những điều khiển nhãn và nút bạn thêm vào form của bạn trước đó. Vì đối tượng myLabel và myButton là riêng (Private), Đây có nghĩa là chúng chỉ có thể tiếp cận trong lớp Form1.

Access modifiers (những từ bổ nghĩa truy cập) cho phép bạn chỉ rõ mức độ mà với mức đó một thành viên của lớp có thể truy cập từ bên ngoài lớp. Bạn cũng có thể sử dụng một Access modifiers để chỉ rõ mức độ mà với mức đó chính lớp này có thể được truy cập.

Bảng 6.1 cho thấy rằng những từ bổ nghĩa truy cập trong trật tự giảm dần tính khả truy cập: Public có tính khả truy cập nhất, và Private có tính khả truy cập kém nhất.

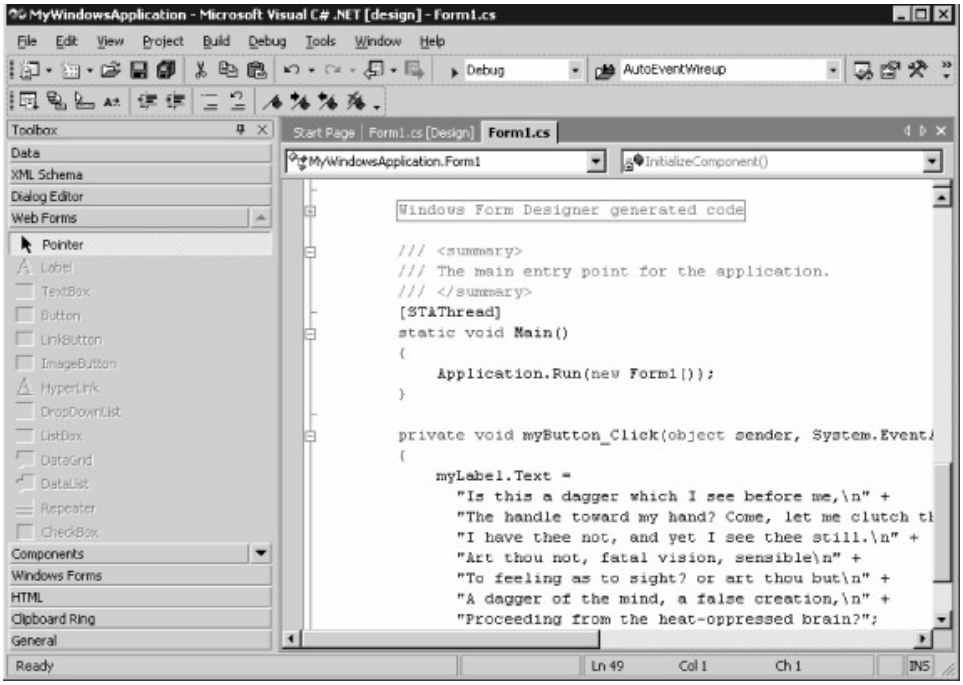
Bảng 6.1: những từ bổ nghĩa truy cập

ACCESS MODIFIER (từ bổ nghĩa truy cập)	**ACCESSIBILITY (khả năng truy cập)**
public	Thành viên có thể truy cập lớp này không hạn chế.
protected internal	Chỉ những thành viên bên trong lớp, thành viên một lớp dẫn xuất hay lớp trong cùng chương trình (hay assembly) có thể truy cập lớp này.
internal	Chỉ những thành viên bên trong lớp hay thành viên của lớp trong cùng chương trình (hay assembly) có thể truy cập.
protected	Chỉ những thành viên bên trong lớp hay thành viên trong những lớp được dẫn xuất có thể truy cập .
private	Chỉ thành viên bên trong lớp có thể tiếp cận. Đây là mặc định.

Bộ khởi dựng của lớp Form1 gọi phương thức InitializeComponent() . Phương thức này thêm myLabel và myButton vào form và gán những thuộc tính cho những đối tượng này. Những thuộc tính này bao gồm sự định vị (Vị trí trên form) Name, Size, TabIndex (thứ tự điều khiển được truy cập khi nhấn phím Tab), và Text. Chẳng hạn, mã sau đây gán những thuộc tính cho myLabel:

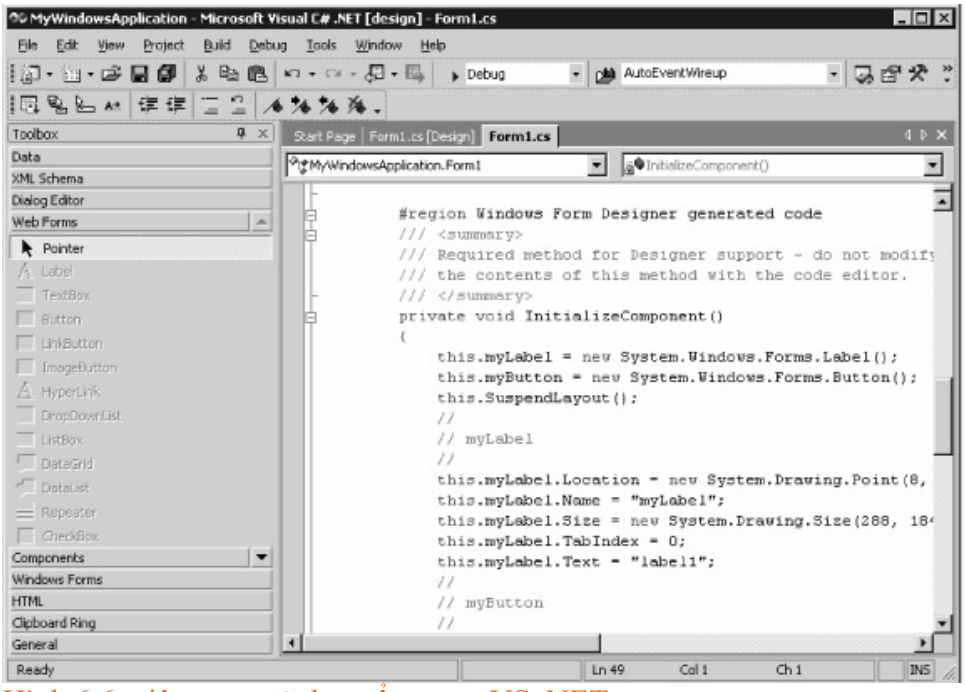
```
this.myLabel.Location = new System.Drawing.Point(8, 8);
this.myLabel.Name = "myLabel";
this.myLabel.Size = new System.Drawing.Size(288, 184);
this.myLabel.TabIndex = 0;
this.myLabel.Text = "label1";
```

Bạn chú ý rằng phương thức InitializeComponent() được bao trong những từ chỉ thị bộ tiền xử lý #region và #endregion . Những chỉ thị này bao lấy một vùng mã và có thể ẩn trong cửa sổ biên tập mã của VS .NET , chỉ để lại văn bản #region lập tức hiển thị . Hình 6.5 cho thấy mã được ẩn xuất hiện như thế nào trong VS .NET.



Hình 6.5: Ẩn mã trong VS .NET sử dụng từ chỉ thị #region

Để xem mã đang ẩn, bạn chỉ cần kích biểu tượng dấu cộng ở bên trái của mã. Hình 6.6 cho thấy mã bên trong những từ chỉ thị #region và #endregion .



Hình 6.6: việc xem mã được ẩn trong VS .NET

Phương thức Main() chạy form bởi gọi phương thức Application.Run() . Lớp Application là Static (tĩnh) và cung cấp một số phương thức mà bạn có thể sử dụng trong chương trình Windows của bạn. Vì lớp này là Static, bạn không cần phải tạo ra một thể hiện (instance) của lớp này, và những thành viên của nó luôn luôn sẵn sàng bên trong form của bạn. Khi phương thức Run() được gọi , form của bạn đã sẵn sàng đợi những sự kiện từ con chuột và bàn phím. Một ví dụ về một sự kiện kích của nút trong form của bạn.

Phương thức myButton_Click() là phương thức mà bạn biên soạn trước nó gán thuộc tính Text của myLabel tới một chuỗi chứa lời trích dẫn từ vở kịch Macbeth. Khi myButton được kích, phương thức myButton_Click() được gọi và văn bản trong myLabel được thay đổi; bạn đã thấy điều này khi bạn chạy form của bạn trước đó.

Trong mục kế tiếp, bạn sẽ học về VS .NET Solution Explorer (bộ duyệt giải pháp VS .NET).

Làm việc với Bộ thăm dò Giải pháp

Bạn có thể sử dụng VS .NET Solution Explorer để xem những phần tử trong dự án của bạn, như namespace cho dự án của bạn. Tất nhiên, một dự án có thể chứa hơn một namespace. Để xem Solution Explorer, bạn chọn View ➢ Solution Explorer.

Mẹo nhỏ: Bạn cũng có thể xem Bộ thăm dò giải pháp bởi nhấn Ctrl+ Alt+ L trên bàn phím.

Bạn có thể sử dụng Solution Explorer (Bộ thăm dò giải pháp) để xem những phần tử sau đây trong một namespace của dự án :

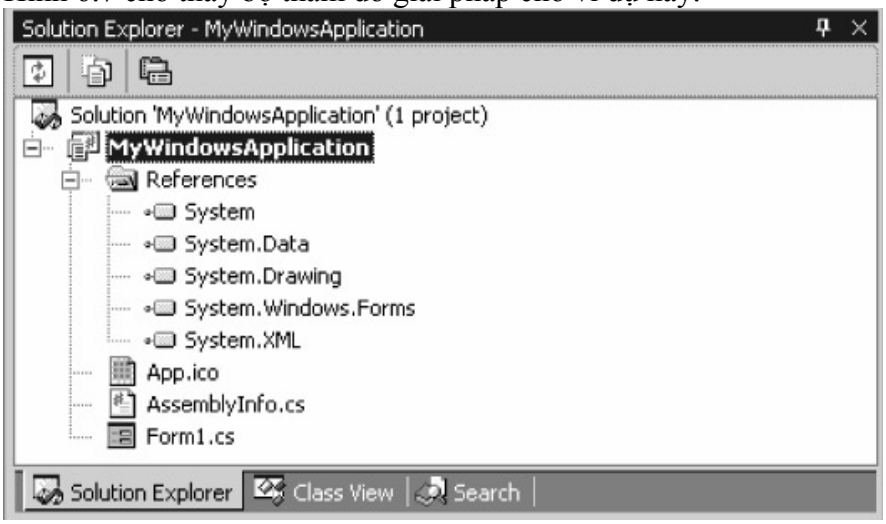
References: Những tham chiếu bao gồm những không gian tên (namespaces) khác và những lớp mà tới đó mã form của bạn viện đến. Bạn có thể sử dụng phát biểu *using* để tham chiếu *Namespaces* và những lớp khác.

Icon File: một file biểu tượng có phần mở rộng như . Ico. Bạn sử dụng một file biểu tượng để gán hình ảnh được hiển thị trong Windows Explorer cho ứng dụng của bạn.

Assembly File : một file assembly chứa siêu dữ liệu cho assembly của ứng dụng của bạn. Một assembly là tập hợp của mã cho ứng dụng của bạn.

Code Files: Một file mã là một tập tin nguồn chương trình, như Mã cho một Form. Bạn đã thấy một ví dụ về điều này trong mục trước đó " khảo sát Mã đằng sau form ".

Hình 6.7 cho thấy bộ thăm dò giải pháp cho ví dụ này.



Hình 6.7: Bộ thăm dò Giải pháp

Như bạn có thể thấy trong Hình 6.7, bạn có thể mở rộng hay gom lại những phần tử trình bày trong Solution Explorer (Bộ thăm dò giải pháp) bởi kích biểu tượng dấu cộng hay trừ, tương ứng. Bạn cũng có thể trình bày những thuộc tính cho một phần tử trong Bộ thăm dò giải pháp : Khi bạn có cửa sổ những thuộc tính được trình bày, chọn một phần tử trong Bộ thăm dò giải pháp cũng sẽ hiển thị những thuộc tính cho phần tử này. Chẳng hạn, trong Hình 6.7, những thuộc tính cho dự án MyWindowsApplication được trình bày; bạn có thể nhìn thấy file dự án là MyWindowsApplication.csproj.

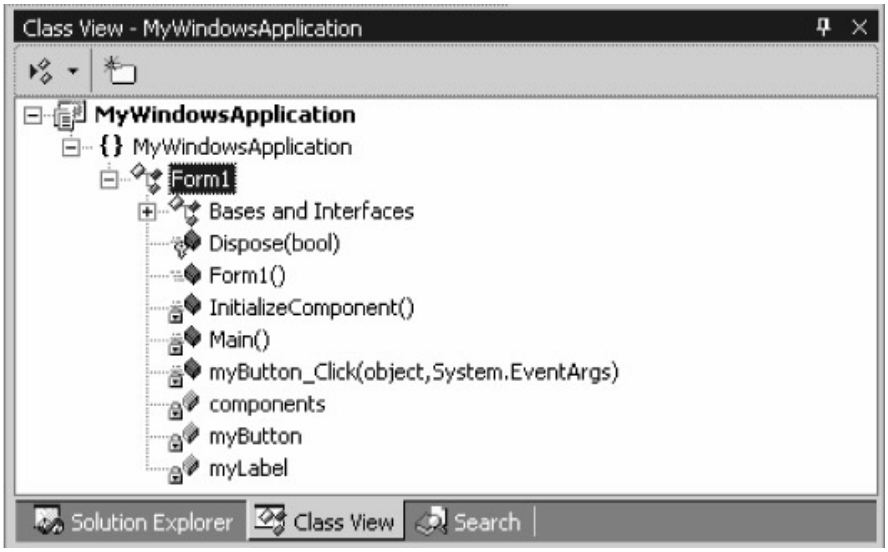
Trong mục kế tiếp, bạn sẽ học về VS .NET Class View.

Làm việc với Class View

Bạn sử dụng VS .NET Class View để khảo sát những lớp, những phương thức và những đối tượng trong dự án của bạn. Để xem Class View, bạn chọn View ➢Class View.

Mẹo nhỏ: Bạn cũng có thể xem Class View bởi nhấn Ctrl+ Shift+ C trên bàn phím .

Hình 6.8 trình bày Class View cho ví dụ.



Hình 6.8: Class View

Như bạn có thể thấy trong Hình 6.8, bạn có thể xem những lớp, những phương pháp và những đối tượng cho ví dụ. Bạn cũng có thể xem những thuộc tính cho một phần tử được chọn trong cửa sổ những thuộc tính. Ví dụ, Hình 6.8 cũng cho thấy những thuộc tính của lớp Form1.

Tiếp theo, bạn sẽ được giới thiệu tới những loại khác của những điều khiển trong Windows.

Sử dụng những điều khiển Windows

Bảng 6.2 liệt kê những điều khiển trên form Windows mà bạn có thể nhặt từ mục Windows Forms của Toolbox (hộp công cụ). Bạn có thể đặt bất kỳ điều khiển nào trong số chúng lên form của bạn.

Bảng 6.2: những điều khiển Windows Form thông dụng

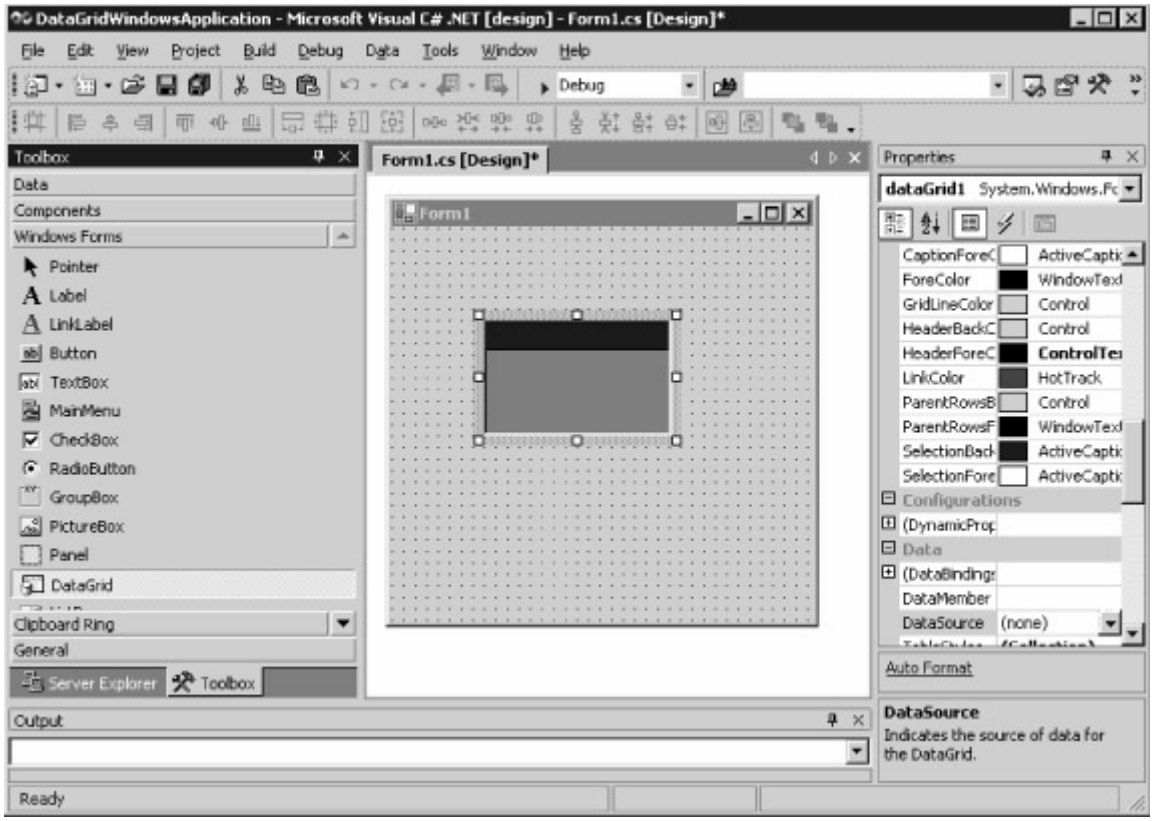
CONTROL	**DESCRIPTION**
Label	Hiển thị văn bản. Bạn gán văn bản mà bạn muốn trình bày cho thuộc tính Text.
LinkLabel	Tương tự như một nhãn, ngoại trừ nó trình bày một mối siêu liên kết (hyperlinks). Bạn gán mối siêu liên kết -hyperlink mà Bạn muốn trình bày sử dụng thuộc tính Text. Bạn gán đường dẫn qua sự kiện LinkClicked.
Button	Một nút có thể nhấn. Thuộc tính Text xác định văn bản được hiển thị trên nút.
TextBox	Một hộp chứa văn bản mà người sử dụng form của bạn có thể soạn thảo khi chạy chương trình. Thuộc tính Text chứa văn bản sẽ trình bày trong TextBox.
MainMenu	Một thực đơn bạn có thể thêm vào một form.
CheckBox	Một hộp kiểm chứa một giá trị Boole true/false, nó được gán là true khi người sử dụng kích đặt dấu kiểm trong hộp.Thuộc tính Checked chỉ định giá trị Boole cho hộp kiểm.
RadioButton	Một nút radiô chứa một giá trị đại số Boole true/false, nó d8ược gán tới true bởi người sử dụng nếu họ kích nút. Thuộc tính Checked chỉ định giá trị Boole cho hộp.
GroupBox	Một nhóm hộp cho phép bạn nhóm những điều khiển liên quan lại thành một nhóm. Chẳng hạn, bạn có thể nhóm những nút radiô liên quan với nhau. Quan trọng nhất, nó cho phép bạn đối sử những điều khiển này như một nhóm.
PictureBox	Một hộp ảnh (picture box) trình bày một hình ảnh mà bạn gán cho thuộc Image của hộp ảnh.
Panel	Một khung chứa cho những điều khiển khác như những nút radiô hay những hộp nhóm (group boxes).
DataGrid	Một khung lưới chứa dữ liệu được truy xuất từ một nguồn dữ liệu, như một cơ sở dữ liệu chẳng hạn. Bạn gán nguồn dữ liệu sử dụng thuộc tính DataSource của DataGrid.
ListBox	Một danh sách của những tùy chọn. Bạn gán danh sách của những tùy chọn sử dụng phương thức Add() của thuộc tính tập hợp Items .
CheckedListBox	Tương tự như một hộp danh sách ngoại trừ một nút kiểm được đặt ở bên trái của mỗi tiết mục trong danh sách. Nút kiểm cho phép người sử dụng lựa chọn những tiết mục thông qua một hộp kiểm , khác với sự chọn đồng thời nhiều tiết mục (multiselecting) với phím Shift hoặc Ctrl.
ComboBox	Kết hợp một trường soạn thảo với một hộp danh sách.

Trong mục kế tiếp, bạn sẽ học cách sử dụng một điều khiển DataGrid để truy nhập những hàng trong một bảng cơ sở dữ liệu như thế nào.

Sử dụng một điều khiển DataGrid để truy nhập một Cơ sở dữ liệu

Trong mục này, bạn sẽ học sử dụng một điều khiển DataGrid để truy cập những hàng trong một bảng cơ sở dữ liệu như thế nào. Theo những bước sau để tạo ra một DataGrid sử dụng VS .NET:

1. Đầu tiên, chọn File □ New Project. Trong hộp thoại New Project , chọn Windows Application (ứng dụng Windows), và nhập DataGridWindowsApplication vào trường Name.
2. Kích OK để tiếp tục. Dự án mới của bạn sẽ chứa một form trống.
3. Thêm một điều khiển DataGrid vào form bởi lựa chọn View □ Toolbox, chọn một DataGrid, và kéo nó lên form của bạn. Hình 6.9 cho thấy một form với một DataGrid. Chỉnh sửa DataGrid của bạn lớn gần bằng form của bạn bởi kéo những góc của DataGrid ra tới sát các cạnh của form.



Hình 6.9: Form với một DataGrid

Tiếp theo, bạn sẽ thêm một đối tượng SqlConnection và một đối tượng SqlDataAdapter vào form của bạn.

Ghi chú:

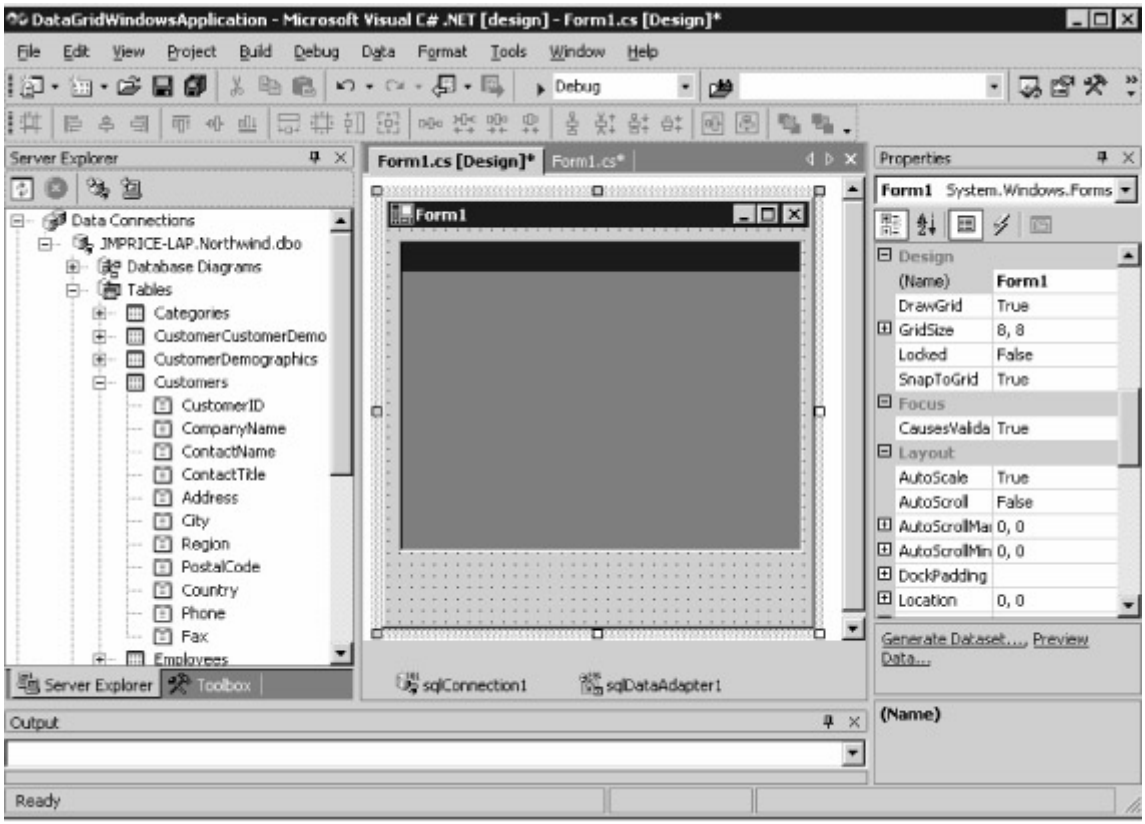
Bạn sử dụng một đối tượng SqlConnection để kết nối tới một cơ sở dữ liệu máy chủ phục vụ SQL, và một đối tượng SqlDataAdapter để dời chuyển những hàng giữa Máy chủ phục vụ SQL và một đối tượng Dataset. Bạn sẽ học những chi tiết về cách kéo những hàng từ cơ sở dữ liệu vào trong một Dataset như thế nào trong Chương 10, và cách để đẩy những thay đổi đã thực hiện trong một Dataset tới cơ sở dữ liệu trong Chương 11.

Bạn có thể kéo một bảng từ một cơ sở dữ liệu Máy chủ phục vụ SQL lên trên form của bạn và có được những đối tượng SqlConnection và SqlDataAdapter được tạo ra đồng thời trong một bước. Bạn sử dụng Server Explorer cho việc này. Với những cơ sở dữ liệu mà không hiển thị trong Server Explorer , những sự lựa chọn của bạn bị hạn chế. Bạn có thể sử dụng những điều khiển trong mục Data của Toolbox để kéo mỗi phần tử tới form của bạn, và rồi gán những thuộc tính cho mỗi đối tượng Dữ liệu với Properties window (cửa sổ những thuộc tính).

Ghi chú: Để mở Server Explorer, chọn View > Server Explorer, hay nhấn Cntl+ Alt+ S.

Để thêm một đối tượng SqlConnection và SqlDataAdapter vào form , bạn thực hiện những bước sau đây:

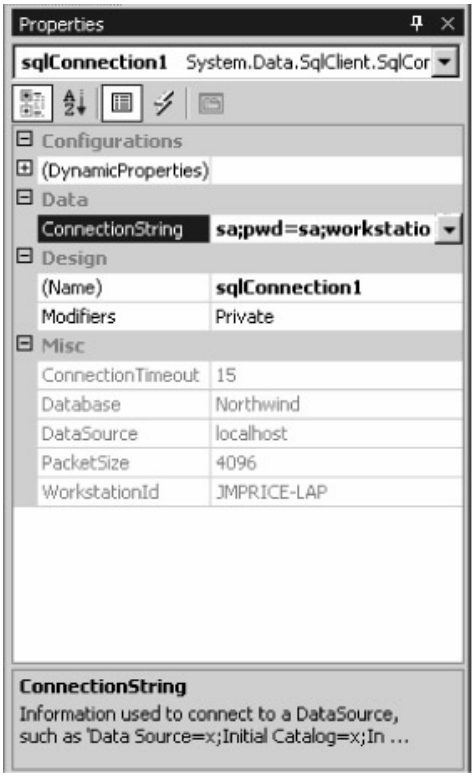
1. Mở Server Explorer.
2. Mở kết nối tới cơ sở dữ liệu Northwind máy chủ phục vụ SQL của bạn (hay tạo ra một kết nối mới nếu cần bởi nhấn chuột phải trên node Data Connections và chọn Add Connection, và nhập vào username sa và password cho cơ sở dữ liệu Northwind của bạn; bạn có thể lấy mật khẩu từ người quản trị cơ sở dữ liệu .
3. mở sâu vào tới bảng Customers trong cơ sở dữ liệu Northwind và kéo tới form của bạn. Điều này tạo ra một đối tượng SqlConnection có tên sqlConnection1 và một đối tượng SqlDataAdapter có tên sqlDataAdapter1, như trong hình 6.10.



Hình 6.10: Form với những đối tượng SqlConnection và SqlDataAdapter

4. Kích đối tượng sqlConnection1 của bạn để trình bày những thuộc tính cho đối tượng này trong cửa sổ những thuộc tính.

5. Để cho phép sqlConnection1 truy cập cơ sở dữ liệu, bạn cần đặt password (mật khẩu) kết nối. Để làm điều này, thêm một chuỗi con đang chứa đựng pwd vào thuộc tính ConnectionString của sqlConnection1. tiến hành thêm pwd= sa (Bạn có lẽ cần có mật khẩu cho người sử dụng "sa" từ người quản trị cơ sở dữ liệu của bạn) tới thuộc tính ConnectionString, như trong hình 6.11.

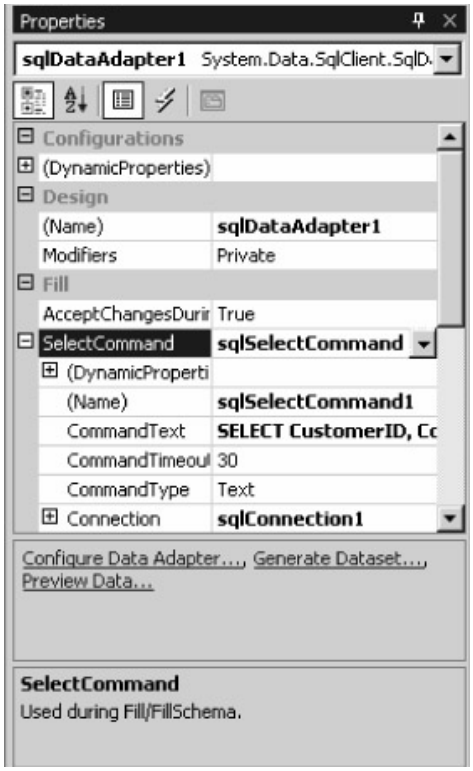


Hình 6.11: đặt thuộc tính ConnectionString cho đối tượng sqlConnection1

Tiếp theo, bạn sẽ sửa đổi phát biểu SELECT SQL được dùng để truy xuất những hàng từ bảng những khách hàng

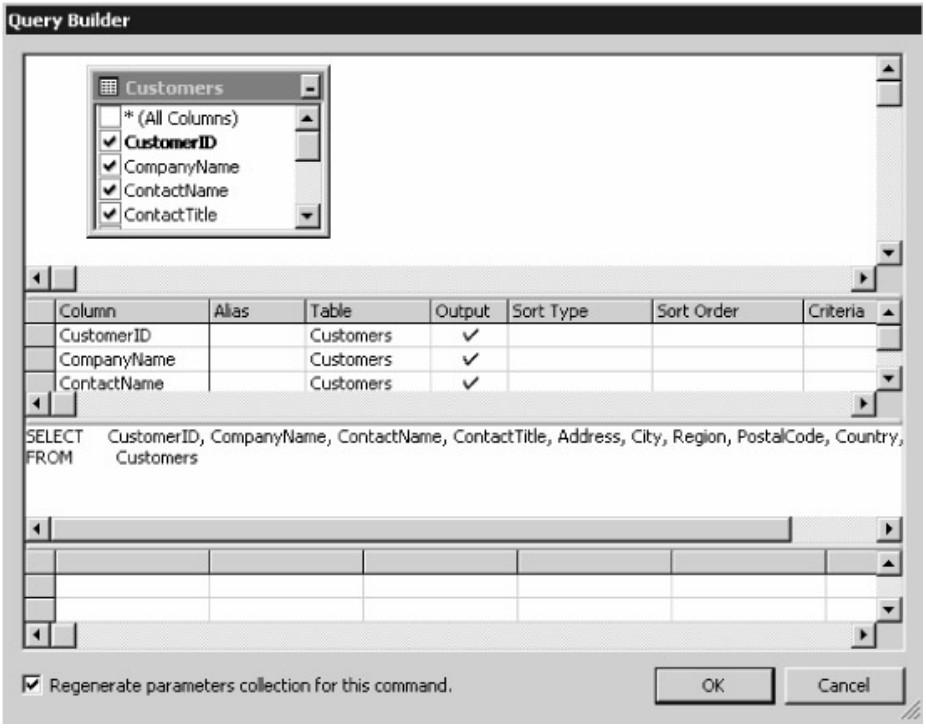
1. Kích đối tượng sqlDataAdapter1 của bạn để trình bày những thuộc tính đối tượng này.

2. Kích biểu tượng dấu + ở bên trái thuộc tính SelectCommand để trình bày những thuộc tính động (dynamic properties); một trong số những thuộc tính động là thuộc tính CommandText, nó chứa phát biểu SELECT (xem Hình 6.12).



Hình 6.12: thuộc tính SelectCommand cho đối tượng sqlDataAdapter1

3. Kích CommandText, Và sau đó Kích nút với ellipsis để trình bày Người xây dựng Câu hỏi, như được đưa vào Hình 6.13.



Hình 6.13: Bộ xây dựng truy vấn

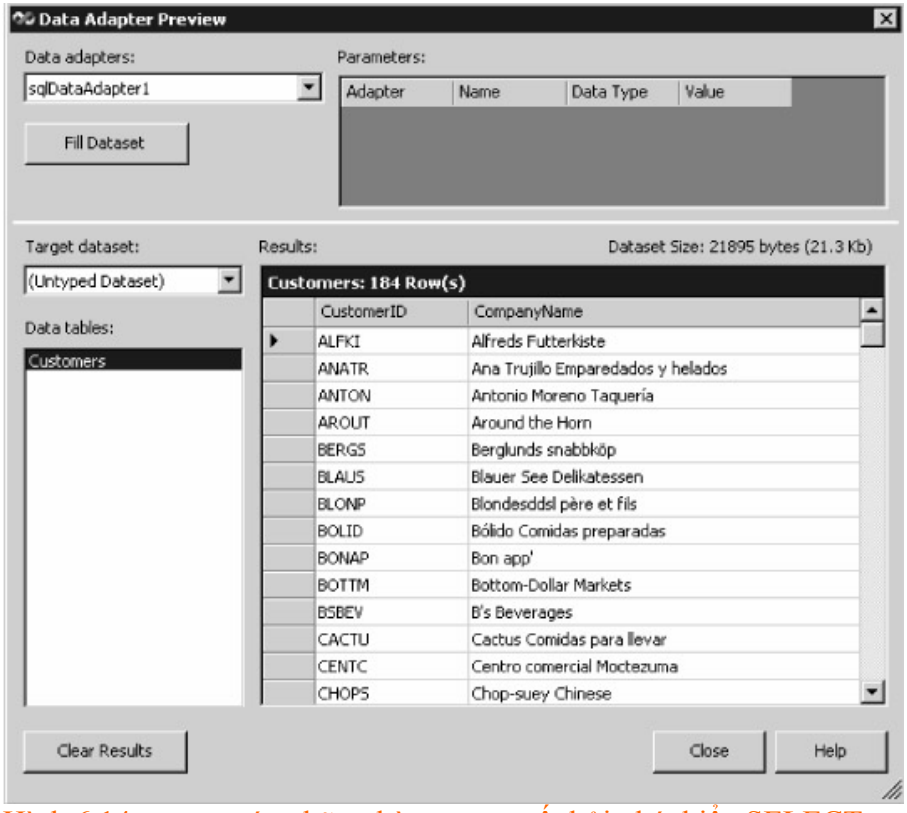
4. Bạn sử dụng Bộ xây dựng truy vấn để định nghĩa những câu lệnh SQL. Bạn có thể nhập câu lệnh SQL, hay bạn cũng có thể xây dựng nó một cách trực quan. Chắc chắn rằng tất cả những cột đều được chọn từ bảng những khách hàng sử dụng những khách hàng cái hộp tại đỉnh bỏ đi (của) Người xây dựng Câu hỏi.

5. Kích OK để tiếp tục.

Để kiểm tra những hàng trả lại bởi phát biểu SELECT này, thực hiện những bước sau đây:

1. Kích liên kế Preview Data link gần đáy cửa sổ những thuộc tính. Việc này trình bày hộp thoại Xem trước Bộ tiếp hợp Dữ liệu.

2. Trong Dữ liệu Bộ tiếp hợp Xem trước hộp thoại, kích nút Tập dữ liệu Khối đắp để chạy sự phát biểu Được lựa chọn, như được đưa vào Hình 6.14.



Hình 6.14: xem trước những hàng truy xuất bởi phát biểu SELECT

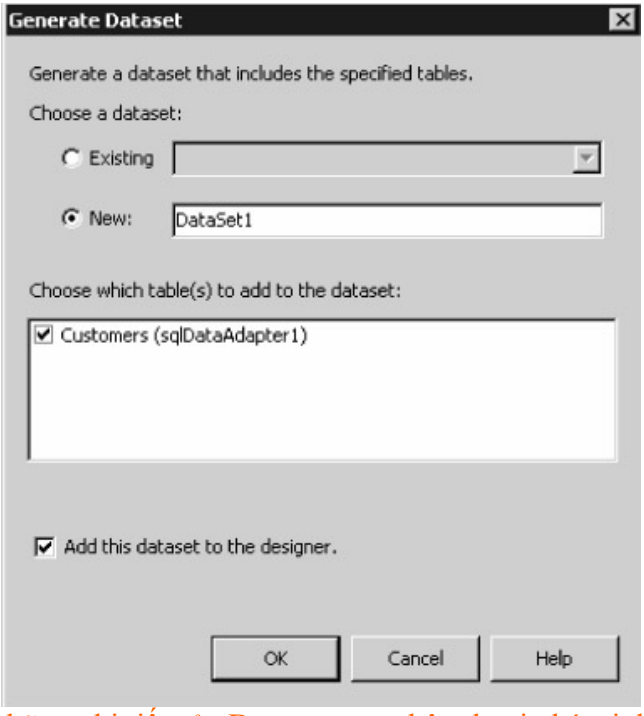
3. Kích nút Close để đóng hộp thoại Data Adapter Preview (xem trước Bộ tiếp hợp dữ liệu).

Tiếp theo, bạn cần tạo ra một đối tượng Dataset. Bạn sử dụng một đối tượng Dataset để lưu trữ một bản sao cục bộ về thông tin được cất giữ trong cơ sở dữ liệu. Một đối tượng Dataset có thể đại diện cho những cấu trúc cơ sở dữ liệu như những bảng, những hàng, và những cột v..v.. Trong ví dụ này, bạn sẽ sử dụng một đối tượng Dataset để dự trữ những hàng từ bảng những khách hàng .

1. Kích một vùng trên form của bạn bên ngoài DataGrid.

2. Kích mối liên kết Generate Dataset (phát sinh Dataset) gần đáy cửa sổ những thuộc tính. Việc này làm hiển thị hộp thoại Generate Dataset .

3. Chọn nút New radio (rađiô mới) và chắc chắn rằng trường bên phải của nút rađiô này chứa DataSet1, như trong hình 6.15.



Hình 6.15: Nhập vào những chi tiết của Dataset trong hộp thoại phát sinh tập dữ liệu Generate Dataset .

4. Kích nút Ok để tiếp tục. Điều này thêm một đối tượng Dataset mới có tên dataSet11 vào form của bạn.

Tiếp theo, bạn sẽ cần gán thuộc tính DataSource của DataGrid tới đối tượng dataset của bạn. Điều này đặt nguồn của dữ liệu cho DataGrid, cho phép những hàng từ Dataset sẽ được trình bày trong DataGrid của bạn. Để thiết đặt thuộc tính DataSource, bạn thực hiện những bước sau đây:

1. Kích đối tượng DataGrid của bạn và gán thuộc tính DataSource tới dataSet11. Customers.

2. Bây giờ, bạn sẽ thêm một nút nhấn, nó sẽ điền đầy sqlDataAdapter1 với những hàng truy xuất bởi phát biểu SELECT của bạn. chọn Button từ Toolbox và kéo nó lên trên form của bạn đến vị trí ngay bên dưới DataGrid của bạn.

3. Gán thuộc tính Text cho nút nhấn của bạn là Run SELECT trong cửa sổ những thuộc tính.

Để cư trú sqlDataAdapter1 với những hàng được truy xuất bởi phát biểu SELECT, bạn cần gọi phương thức Fill() cho đối tượng này. Phương thức này sẽ được gọi khi nút được kích. Để thêm mã cần thiết, thực hiện những bước sau đây:

1. Nhấn đúp nút nhấn bạn đã thêm trước đó. Điều này mở cửa sổ biên tập mã và định vị trí con trỏ trong phương thức button1_Click() .

2. Nhập mã sau đây vào phương thức này:

```
dataSet11.Clear();
sqlDataAdapter1.Fill(dataSet11, "Customers");
```

Ghi chú: Bạn đã cũng có thể gọi phương thức Fill() trong sự kiện Form1_Load. Sự kiện này xuất hiện khi Form thoạt tiên được tải.

Tiếp theo, thêm nút nhấn khác mà sẽ cho phép bạn lưu bất kỳ sự thay đổi nào bạn làm cho những hàng trong DataGrid:

1. Tiếp tục và thêm nút nhấn khác và gán thuộc tính Text của nút này tới Update (cập nhật).

3. Nhấn đúp nút nhấn này và thêm phát biểu sau đây vào phương thức button2_Click() :

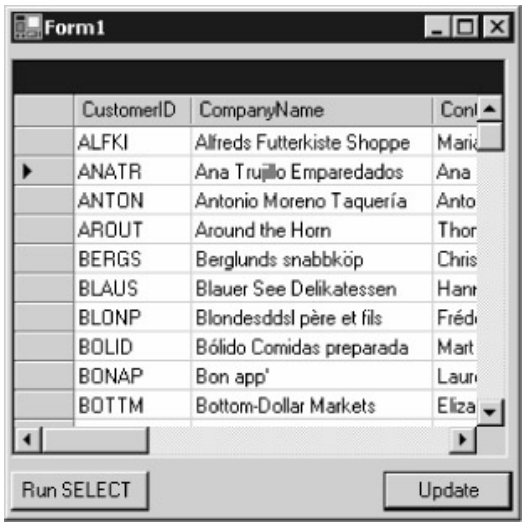
```
sqlDataAdapter1.Update(dataSet11, " Customers ");
```

Phát biểu này cập nhật một hàng với những giá trị cột mới mà bạn nhập vào trong DataGrid của bạn.
Bây giờ bạn đã hoàn tất Form của bạn. Xây dựng dự án bởi chọn Build ➢Build Solution.
Cuối cùng, bạn đã sẵn sàng để chạy Form của bạn! Thực hiện những bước sau đây:

1. Chọn Debug ➢Start without Debugging để khởi chạy Form của bạn.

2. Kích nút "Run SELECT" trên Form của bạn để chạy phát biểu SELECT của bạn. Điều này truy xuất những hàng từ bảng những khách hàng và trình bày chúng trong DataGrid trên form của bạn.

3. Sửa đổi cột CompanyName của hàng đầu tiên thành "Alfreds Futterkiste Shoppe " và kích nút Update; việc này giao phó sự thay đổi bạn thực hiện tới hàng trong bảng những khách hàng (xem Hình 6.16).



Hình 6.16: Form đang chạy

4. Thiết đặt lại CompanyName cho hàng đầu tiên trở lại nguyên bản trước đó bởi loại bỏ "Shoppe" từ phần cuối và kích Update lại lần nữa.

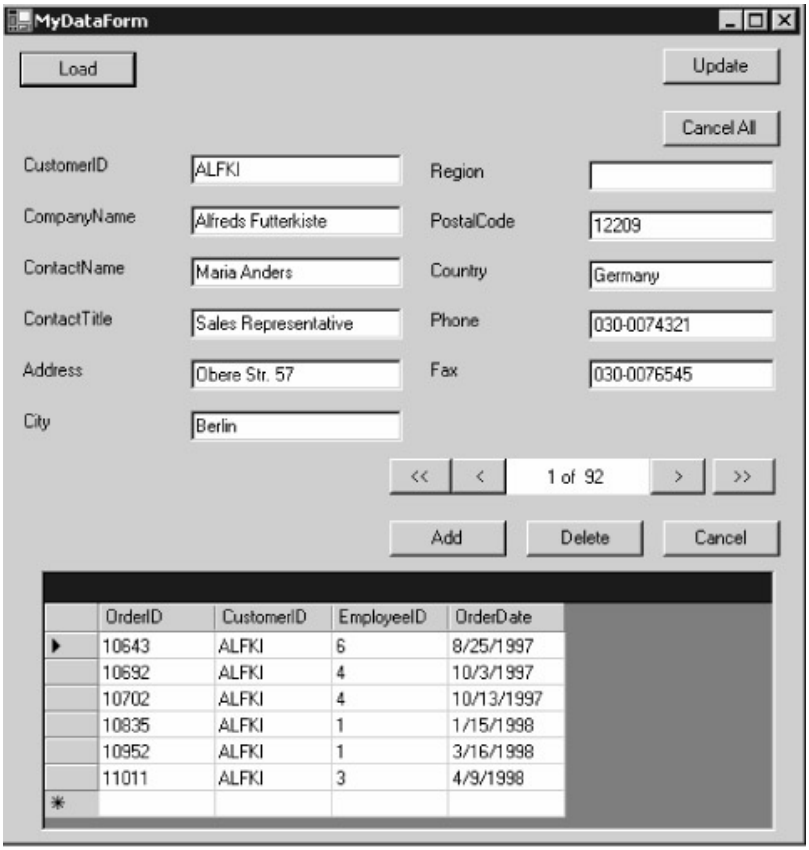
Trong mục kế tiếp, bạn học cách sự sử dụng "VS .NET Data Form Wizard" để tạo ra một ứng dụng Windows tiên tiến hơn để truy cập cơ sở dữ liệu Northwind máy chủ phục vụ SQL.

Sử dụng Data Form Wizard để tạo ra một form Windows

Trong mục này, bạn sẽ sử dụng VS .NET Data Form Wizard để tạo ra một ứng dụng Windows truy cập cả hai bảng Customers và Orders . Bảng Orders chứa những hàng đại diện cho những đơn đặt được đặt bởi những khách hàng.

Những hàng trong bảng Orders liên quan đến những hàng trong bảng Customers thông qua một khóa ngoại : Bảng Orders chứa một cột có tên CustomerID là một khóa ngoại liên kết tới cột CustomerID của bảng Customers (CustomerID là khóa chính cho bảng Customers). Sự sử dụng của khóa ngoại là định nghĩa một mối quan hệ cha con giữa những bảng Customers và Orders .

Form mà bạn sẽ tạo ra trình bày một hàng từ bảng những khách hàng, cùng với bất kỳ hàng nào có liên quan từ bảng Orders . Để cho bạn một ý tưởng rõ ràng về mục đích cuối cùng của nó, Hình 6.17 trình bày đầy đủ hình ảnh form đang chạy. Chú ý phần đỉnh của form trình bày chi tiết cho hàng từ bảng Customers có CustomerID là ALFKI; phần đáy của form chứa một điều khiển DataGrid trình bày những hàng từ bảng Orders cho khách hàng này. Khi bạn di chuyển tới hàng kế tiếp trong bảng những khách hàng, những hàng từ bảng Orders cho khách hàng kế tiếp được tự động thay đổi theo trong DataGrid.



Hình 6.17: Form đang chạy

Thực hiện những bước này để bắt đầu xây dựng Form:

1. Lựa chọn File ➢New Project.

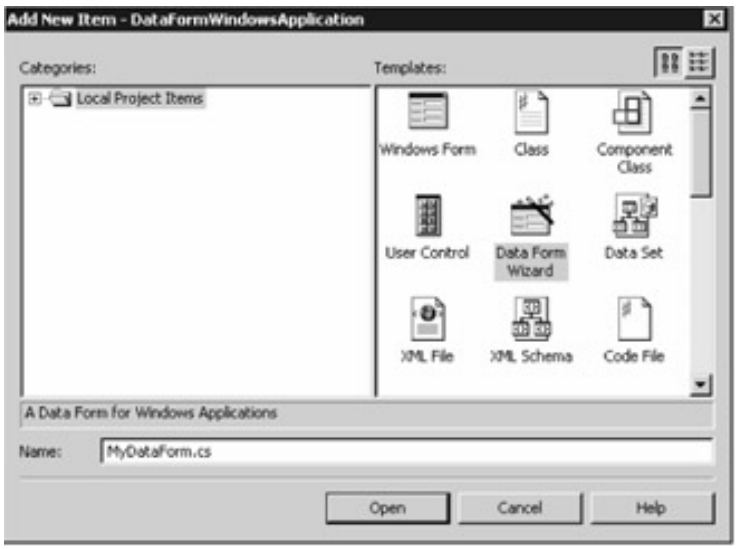
2. Trong hộp thoại New Project , chọn Empty Project, và nhập DataFormWindowsApplication trong trường Name. Vì bạn sẽ được thêm một Form mới vào ứng dụng mới của bạn không lâu nữa, không có yêu cầu nào để VS .NET phát sinh form trống như thường lệ cho bạn; lý do là bạn đang tạo ra một dự án trống .

3. Kích OK để tiếp tục. VS .NET sẽ tạo ra một dự án trống mới cho bạn.

Tiếp theo, bạn sẽ sử dụng Data Form Wizard để tạo ra một form truy cập những bảng Customers và Orders trong cơ sở dữ liệu Northwind.

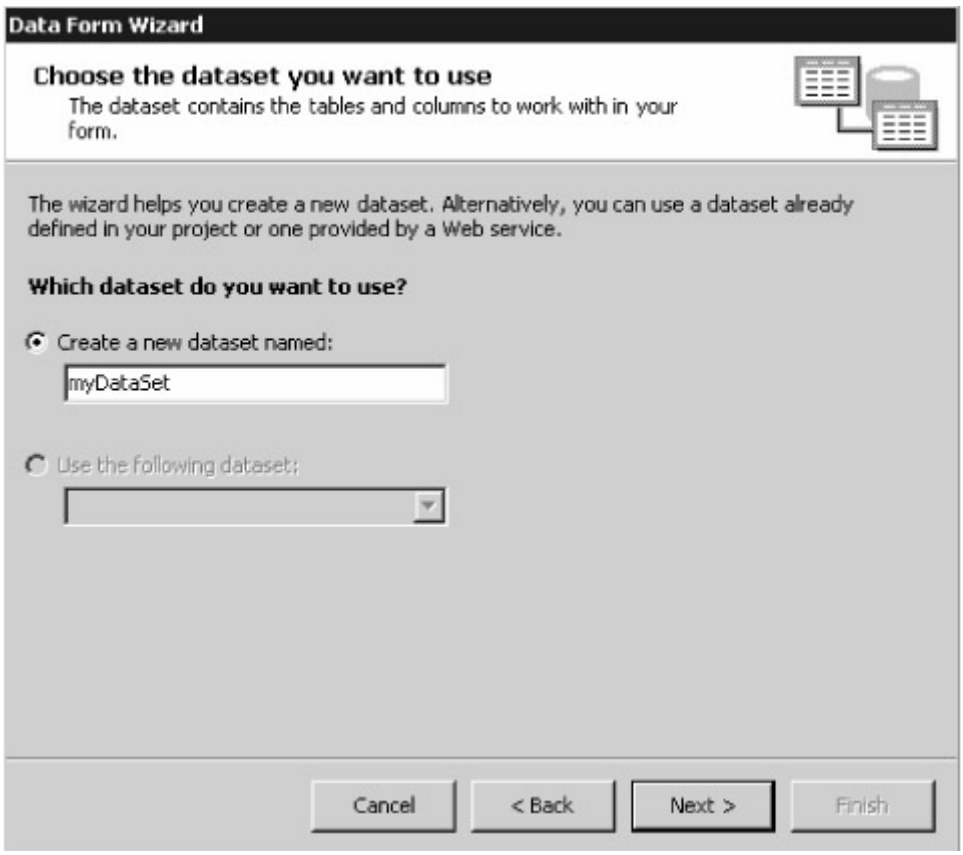
1. Chọn Project ➢Add New Item.

2. Chọn Data Form Wizard từ mục Templates (những khuôn mẫu) ở bên phải, nhập tên của Form là MyDataForm.cs, và kích Open (xem Hình 6.18). Bạn sẽ thấy trang welcome cho Data Form Wizard.



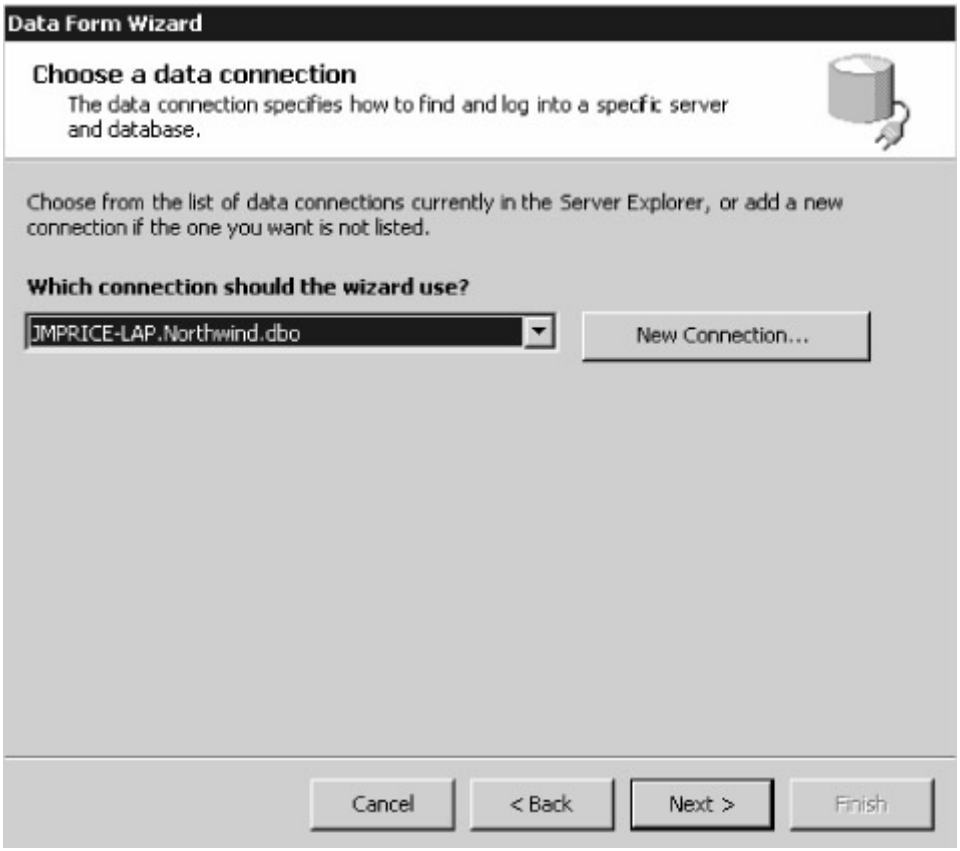
Hình 6.18: Thêm một from dữ liệu sử dụng Data Form Wizard

3. Kích nút Next để tiếp tục.
4. Bây giờ bạn nhập đối tượng Dataset bạn muốn sử dụng trong form của bạn. Bạn có thể nhặt một Dataset đang tồn tại, hoặc bạn có thể tạo ra một Dataset mới. Vì đây là một dự án mới, bạn sẽ tạo ra một Dataset mới. Nhập myDataSet như một tên cho Dataset của bạn, như trình bày trong Hình 6.19.



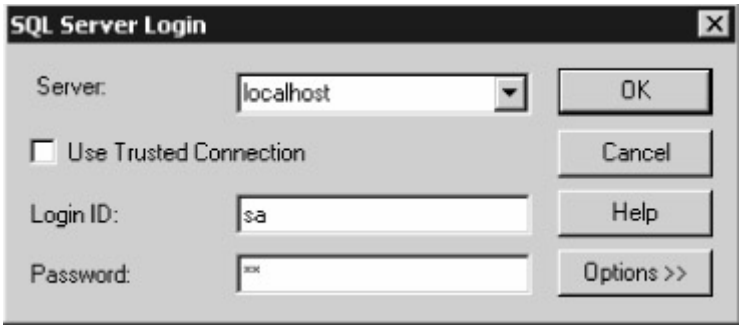
Hình 6.19: Nhập tên của dataset mới

5. Kích nút Next để tiếp tục.
6. Bây giờ bạn phải chọn một data connection (kết nối dữ liệu) để truy cập cơ sở dữ liệu. Bạn có thể chọn một kết nối đã có, hay bạn có thể tạo ra một kết nối mới. Chọn kết nối của bạn, như trình bày trong Hình 6.20-tất nhiên, tên kết nối của bạn có thể sẽ khác với trong sách này.



Hình 6.20: Chọn kết nối dữ liệu

7. Kích nút Next để tiếp tục.
8. Bây giờ bạn đăng nhập vào cơ sở dữ liệu bởi ghi rõ mật khẩu cho người sử dụng cơ sở dữ liệu. Bạn sử dụng người sử dụng "sa" khi tạo ra kết nối cơ sở dữ liệu trước đó, và do đó bạn cần nhập mật khẩu cho người sử dụng này, như trình bày trong Hình 6.21.



Hình 6.21: đăng nhập tới cơ sở dữ liệu Northwind máy chủ phục vụ SQL

9. Kích nút Ok để tiếp tục.

Bây giờ bạn chọn những bảng cơ sở dữ liệu hay những views bạn muốn sử dụng trong form của bạn. Vùng bên trái ở đáy của hộp thoại trình bày những bảng và những views bạn có thể truy cập sử dụng form của bạn. Vùng bên phải ở đáy hộp thoại trình bày những bảng và những views mà bạn đã thêm vào . Bạn thêm một bảng hay view vào form của bạn bởi chọn nó từ vùng bên trái và kích nút mũi tên trỏ về bên phải.

Mẹo nhỏ:

Bạn cũng có thể nhấn đúp trên bảng hay view để thêm chúng vào form của bạn.

Khi bạn làm điều này, bảng hay view di chuyển tới bên phải, cho biết bạn đã chọn chúng để sử dụng trong form của bạn. Nếu bạn quyết định không muốn sử dụng một bảng hay view, bạn hủy chọn chúng sử dụng nút mũi tên chỉ trái. Bạn cũng có thể nhấn đúp bảng hay view để bỏ chọn chúng. Chọn những bảng Customers và Orders , như trình bày trong Hình 6.22. Kích nút Next để tiếp tục.

Hình 6.22: chọn những bảng Customers và Orders để sử dụng trong form

Bởi vì bạn đã chọn hai bảng- Customers và Orders- bước tiếp theo bạn sẽ định nghĩa một mối quan hệ giữa những bảng đó. Mối quan hệ này được sử dụng trong form của bạn để đồng bộ hóa sự dẫn hướng giữa những hàng trong bảng Customers với những hàng trong bảng Orders : khi bạn di chuyển tới một hàng kế tiếp trong bảng Customers, những hàng từ bảng Orders sẽ được trình bày trong form của bạn. thực hiện những việc sau đây trong hộp thoại (xem Hình 6.23):

1. Nhập myRelationship vào trường Name.
2. Chọn Customers như bảng cha .
3. Chọn Orders như bảng con.
4. Chọn CustomerID như khóa cho mỗi bảng.

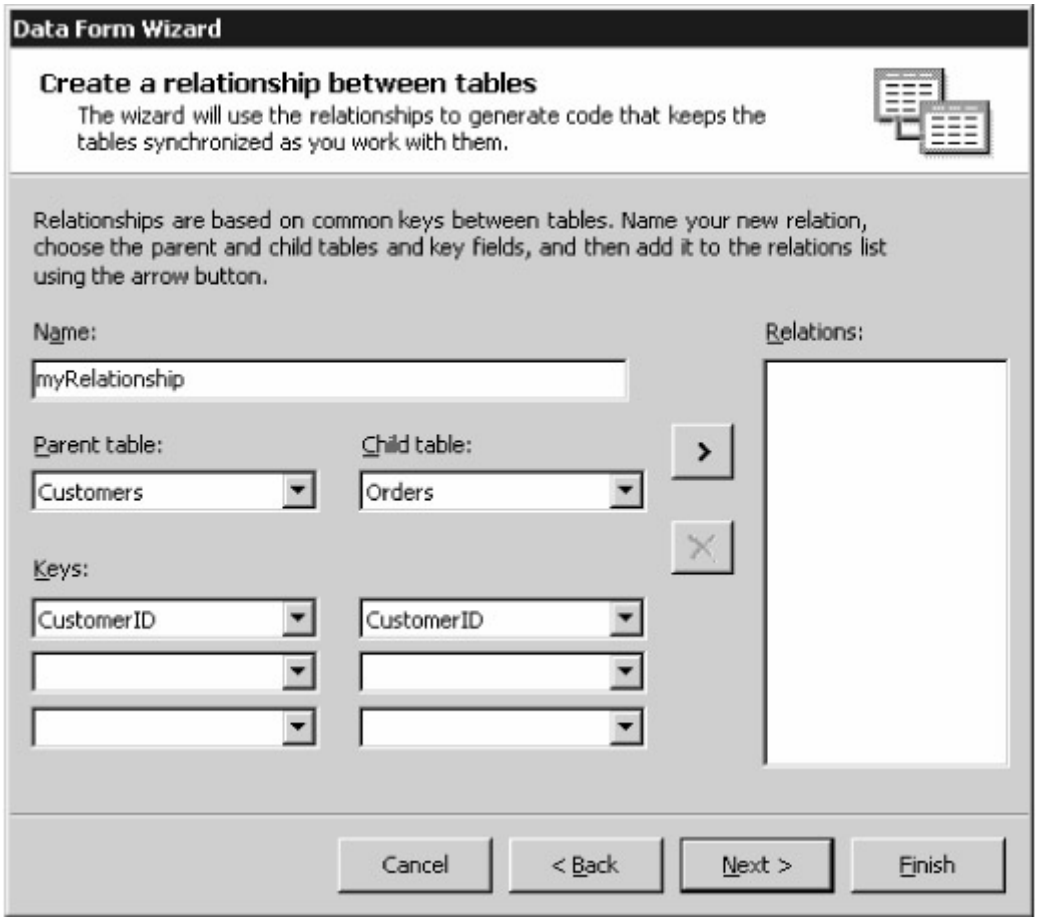
Cảnh báo:

Để thêm mối quan hệ vào form của bạn, kích nút mũi tên chỉ phải. Nếu bạn không làm điều này, mối quan hệ của bạn sẽ không được thêm vào form.

5. Kích nút Next để tiếp tục.
6. Chọn những cột từ những bảng bạn muốn hiển thị trong form của bạn. Vì bạn đã thêm những bảng Customers và Orders vào form của bạn, nên bạn sẽ chọn những cột để trình bày từ hai bảng này. Theo mặc định, tất cả những cột từ những bảng đã chọn sẽ được hiển thị. Bạn sẽ không trình bày tất cả những cột từ hai bảng này. Bỏ chọn cột City cho bảng Customers. (Sau này, bạn sẽ thấy cách thêm cột này vào form của bạn như thế nào bằng tay.)
7. Bỏ chọn những cột sau đây cho bảng Orders :

RequiredDate	ShipAddress
ShippedDate	ShipCity
ShipVia	ShipRegion
Freight	ShipPostalCode

ShipName ShipCountry

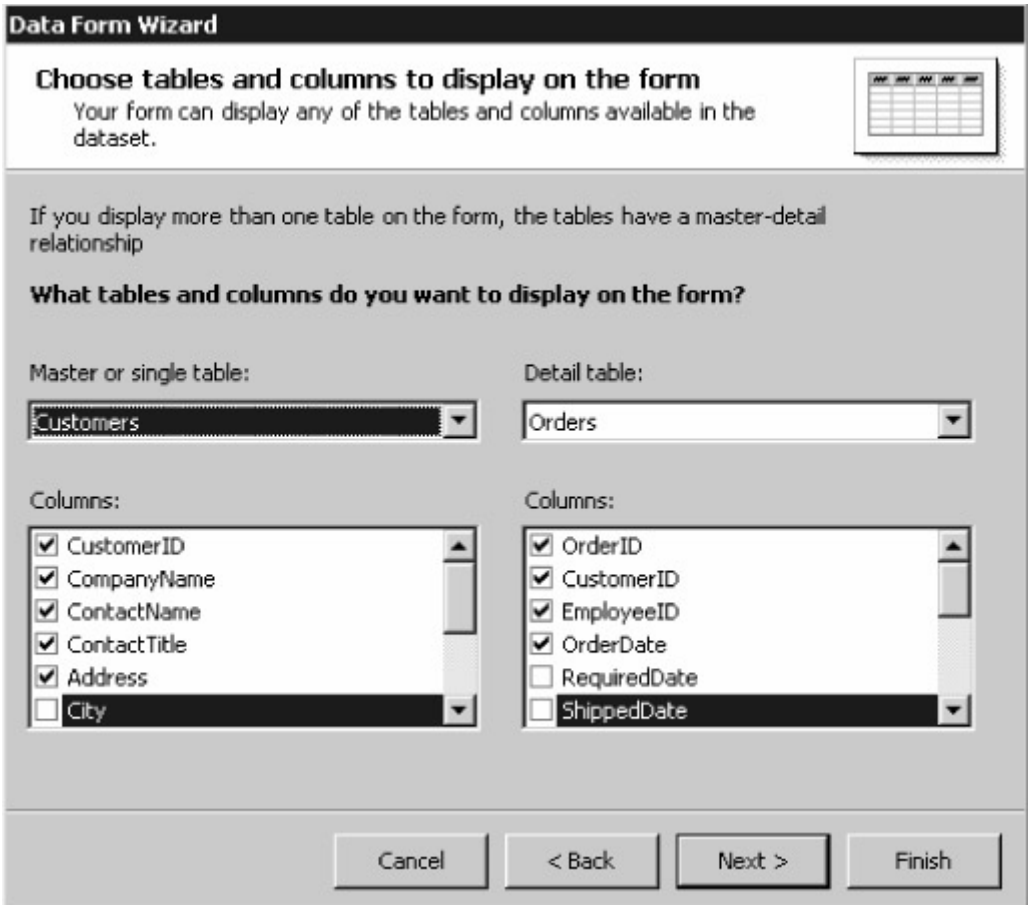


Hình 6.23: tạo ra một mối quan hệ giữa hai bảng

Ghi chú:

ghi nhớ: Bạn đang bỏ chọn những cột này, vì vậy bạn hủy chọn những cột cho bảng Orders.

Hình 6.24 trình bày hộp thoại đầy đủ với những cột được chọn để hiển thị từ mỗi bảng.



Hình 6.24: Chọn những cột để hiển thị từ mỗi bảng

8. Kích nút Next để tiếp tục.
9. Chọn kiểu trình bày cho những hàng (cũng được biết đến như những bản ghi (records)) trong bảng cha sẽ được trình bày trong form của bạn. Bạn có thể hiển thị những hàng trong một khung lưới, hay bạn có thể trình bày mỗi cột sử dụng một điều khiển riêng biệt. Bạn sẽ sử dụng một điều khiển riêng biệt cho những cột, vì vậy chọn "Single Record" (bản ghi đơn) trong nút radio "individual controls" . Những hộp kiểm khác trong hộp thoại cho phép bạn chọn những điều khiển bạn muốn thêm vào form. Những điều khiển này ảnh hưởng đến những hàng trong bảng chủ, và bạn có thể thêm những điều khiển sau đây vào form của bạn:

Ghi chú:

Trong ví dụ này, bảng cha là bảng Customers , và bảng con là bảng Orders . Những hàng cho bảng con được trình bày trong một điều khiển DataGrid.

Cancel All : nút Cancel All cho phép bạn huỷ bất kỳ sự thay đổi nào bạn đã làm tới hàng hiện thời.

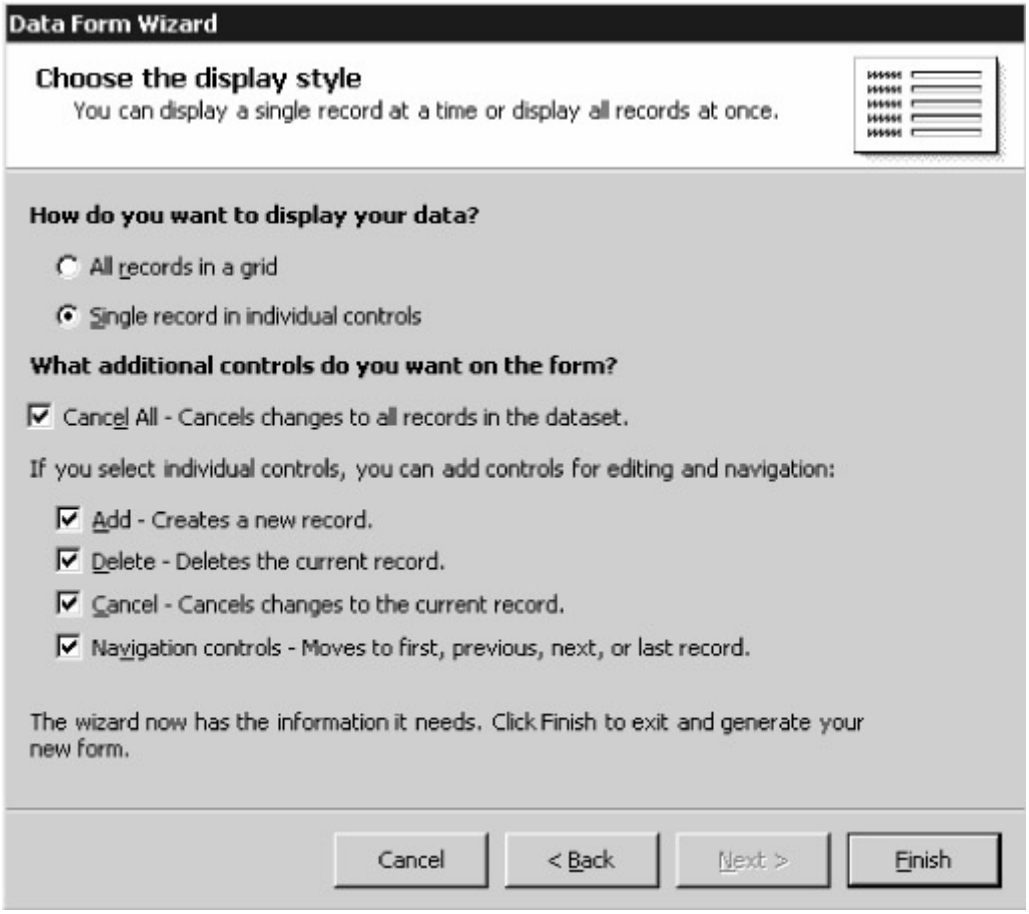
Add: nút Add cho phép bạn thêm một hàng mới.

Delete : nút Delete cho phép bạn xóa hàng hiện thời.

Cancel : nút Cancel cho phép bạn hủy bỏ một sự thay đổi được làm tới hàng hiện thời.

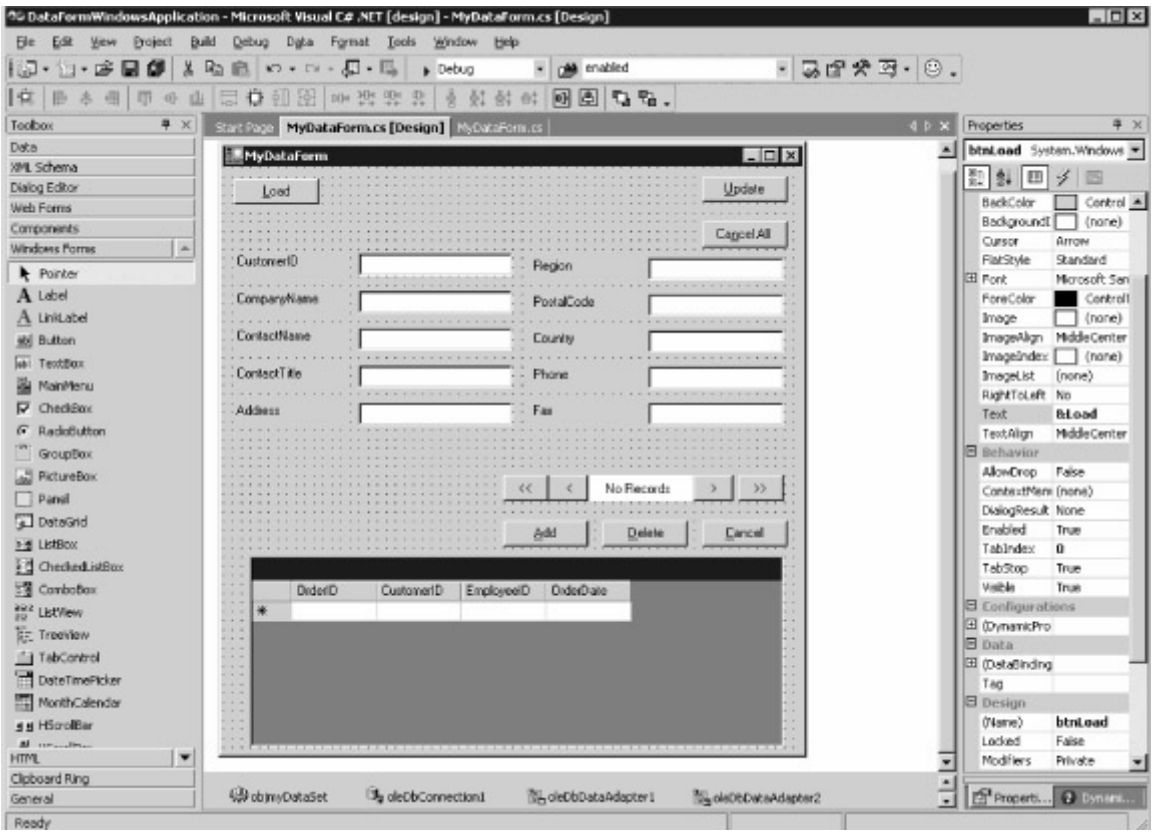
Navigation Controls : những điều khiển dẫn hướng gồm có bốn nút nó cho phép bạn di chuyển tới hàng đầu tiên, hàng kế trước, hàng kế tiếp, và hàng cuối cùng. Một chỉ báo cũng được trình bày để cho thấy hàng hiện thời.

Hình 6.25 cho thấy hộp thoại đầy đủ.



Hình 6.25: chọn kiểu trình bày

10. Bây giờ Bạn có hoàn thành tất cả những bước trong Data Form Wizard. Kích nút Finish để tạo form của bạn. VS .NET bây giờ sẽ trình bày form mới, như trong Hình 6.26.



Hình 6.26: form đầy đủ

Những đối tượng managed provider (bộ cung cấp được quản lý) trong form của bạn sử dụng những lớp OLE DB được chứa trong namespace System.Data.OleDb - Mặc dù một cơ sở dữ liệu SQL Server được sử dụng. Những đối tượng này làm việc với bất kỳ cơ sở dữ liệu tương hợp DB OLE nào. Mã sẽ hiệu quả hơn nếu những lớp "bộ cung cấp được quản lý" trong không gian tên System.Data.SqlClient được sử dụng thay vào đó; những