Kết nối cơ sở dữ liệu với ADO.NET

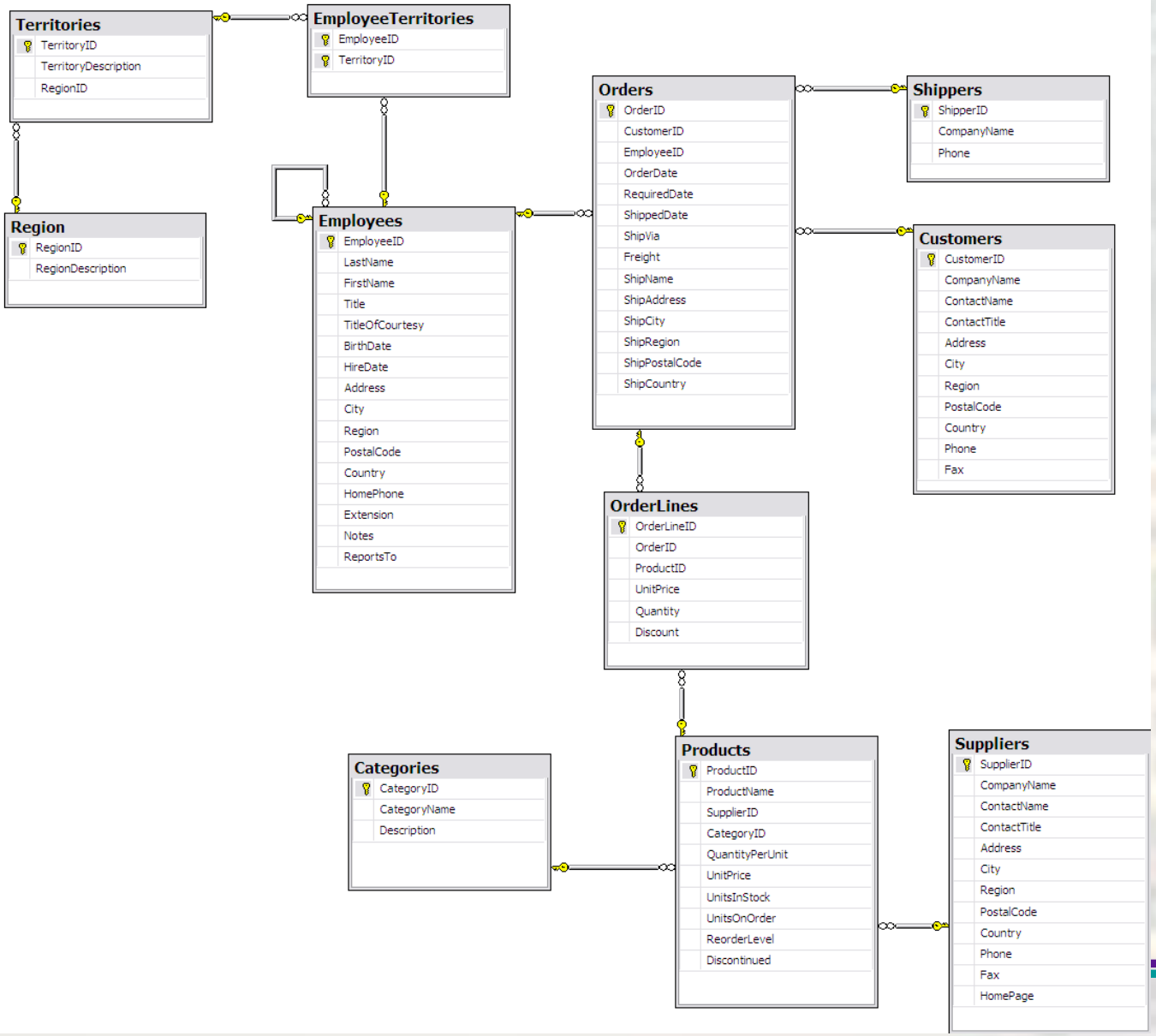
Ths VŨ HÀ TUẤN ANH
anh.vht@due.edu.vn

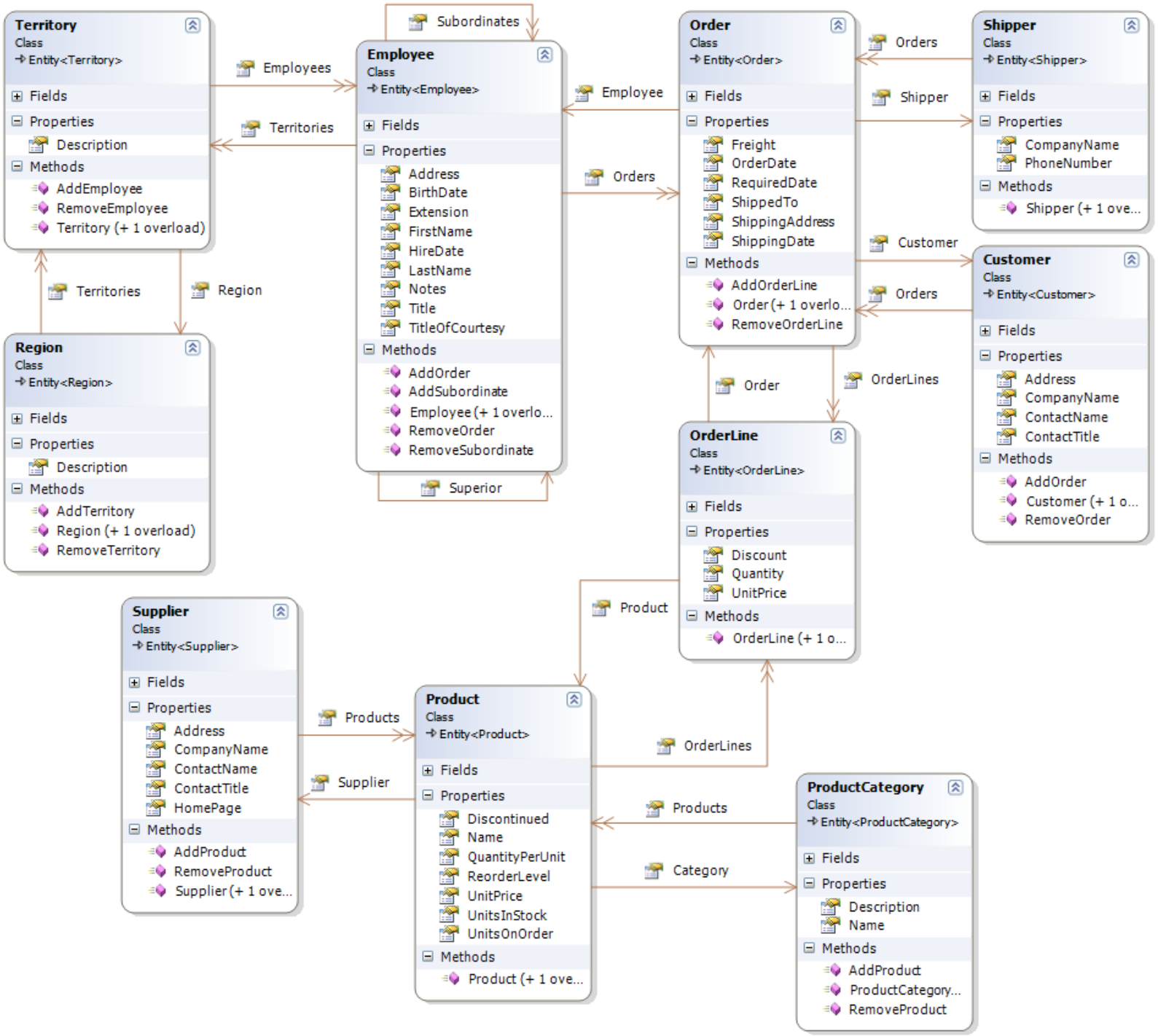
ĐÀ NẴNG 11/2010

NỘI DUNG

- **ADO.NET**
  - .Net Framework Data Providers
  - Connection
  - Command
  - DataReader
  - DataAdapter
  - DataSet
  - DataTable
  - DataColumn
  - DataRow

Cơ sở dữ liệu Northwind




Territory
Class
Entity<Territory>
Fields
Properties
Description
Methods
AddEmployee
RemoveEmployee
Territory (+ 1 overload)
Employees
Territories
Subordinates
Employee
Class
Entity<Employee>
Fields
Properties
Address
BirthDate
Extension
FirstName
HireDate
LastName
Notes
Title
TitleOfCourtesy
Methods
AddOrder
AddSubordinate
Employee (+ 1 overlo...
RemoveOrder
RemoveSubordinate
Superior
Territories
Region
Region
Class
Entity<Region>
Fields
Properties
Description
Methods
AddTerritory
Region (+ 1 overload)
RemoveTerritory
Employee
Orders
Order
Class
Entity<Order>
Fields
Properties
Freight
OrderDate
RequiredDate
ShippedTo
ShippingAddress
ShippingDate
Methods
AddOrderLine
Order (+ 1 overlo...
RemoveOrderLine
Orders
Shipper
Shipper
Class
Entity<Shipper>
Fields
Properties
CompanyName
PhoneNumber
Methods
Shipper (+ 1 ove...
Shipper
Customer
Orders
Customer
Class
Entity<Customer>
Fields
Properties
Address
CompanyName
ContactName
ContactTitle
Methods
AddOrder
Customer (+ 1 o...
RemoveOrder
Order
OrderLines
OrderLine
Class
Entity<OrderLine>
Fields
Properties
Discount
Quantity
UnitPrice
Methods
OrderLine (+ 1 o...
Product
Supplier
Class
Entity<Supplier>
Fields
Properties
Address
CompanyName
ContactName
ContactTitle
HomePage
Methods
AddProduct
RemoveProduct
Supplier (+ 1 ove...
Products
Supplier
Product
Class
Entity<Product>
Fields
Properties
Discontinued
Name
QuantityPerUnit
ReorderLevel
UnitPrice
UnitsInStock
UnitsOnOrder
Methods
Product (+ 1 ove...
OrderLines
Products
Category
ProductCategory
Class
Entity<ProductCategory>
Fields
Properties
Description
Name
Methods
AddProduct
ProductCategory...
RemoveProduct

Các mô hình truy xuất dữ liệu

- **DAO (Data Acess Objects)**
  - Cho phép kết nối các **CSDL nhỏ** trong hệ thống nội bộ.
  - Hữu ích kết nối bảng **Microsoft Acess** và các CSDL khác sử dụng ODBC driver
  - Là **mô hình truy xuất dữ liệu đầu tiên** theo định dạng Microsoft Jet Database Engine
- **RDO (Remote Data Objects)**
  - Cung cấp các đối tượng cho phép **kết nối tới máy chủ CSDL**, thực thi các truy vấn để làm việc với CSDL và cập nhật các thay đổi đến server
- **ADO (ActiveX Data Objects)**
  - Thực hiện **truy xuất dữ liệu đã kết nối**, việc **kết nối CSDL vẫn còn cho đến khi ứng dụng kết thúc**
  - Không cho phép chuyển dữ liệu thông qua firewall
- **ADO.NET**
  - Mô hình truy xuất dữ liệu **hiện đại nhất**
  - Thực hiện **truy xuất dữ liệu không kết nối**
  - **Cho phép chuyển dữ liệu qua firewall** bằng cách sử dụng định dạng XML

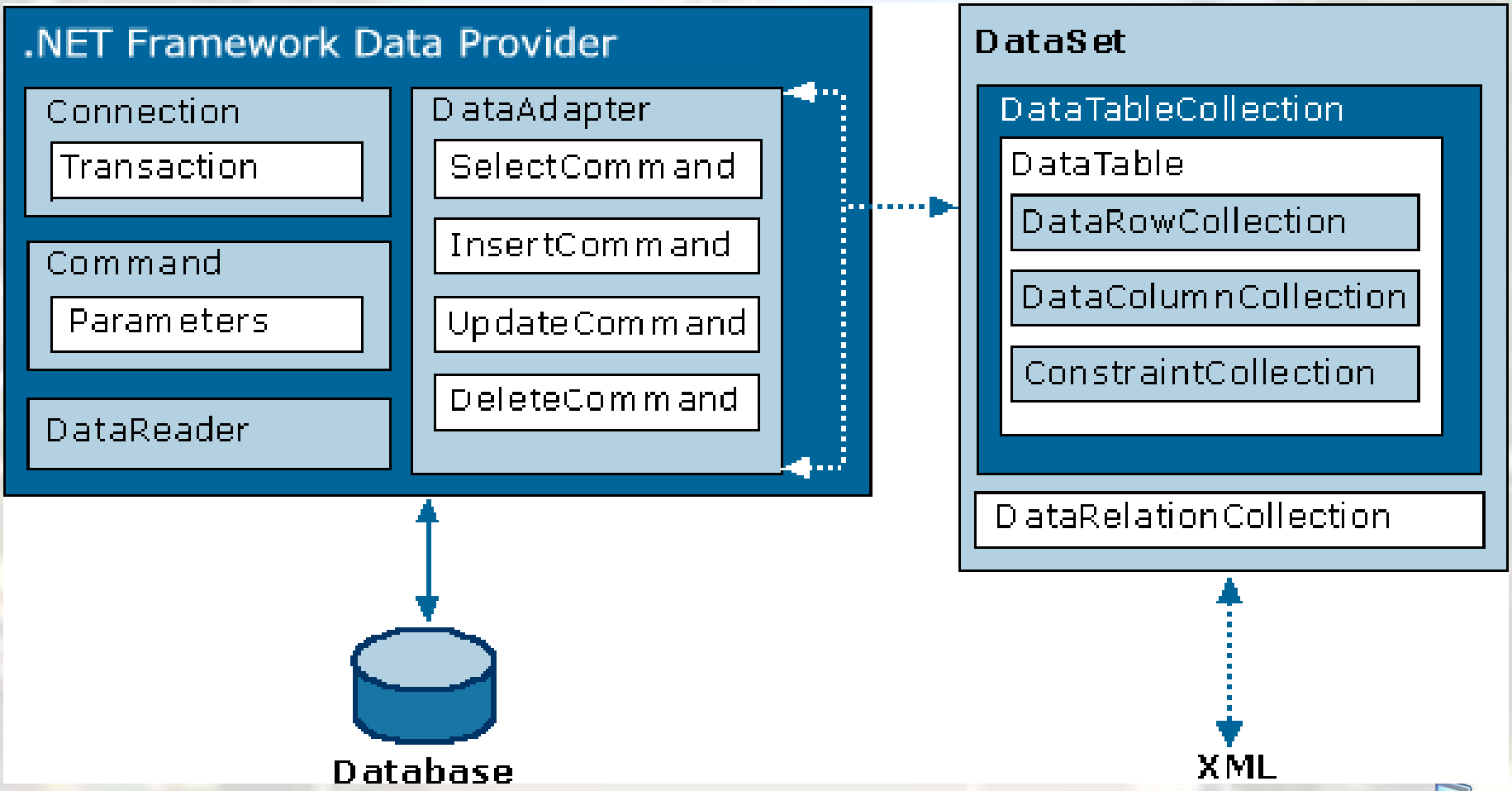
ADO.NET

- ADO.NET là kỹ thuật truy xuất cơ sở dữ liệu, đây là một phần của .NET Framework, và ta có thể sử dụng kỹ thuật này cho các ứng dụng .NET
- ADO.NET cho phép kết nối đến các hệ thống cơ sở dữ liệu như Microsoft SQL Server, Oracle, Microsoft Access, Sybase, ... sử dụng ODBC (Open Database Connectivity), OLE DB (Object Linking and Embedding Database).
- OLE DB và ODBC là các driver được sử dụng để kết nối đến nguồn dữ liệu

ADO.NET

- ADO.NET hỗ trợ kiến trúc dữ liệu không kết nối. Nghĩa là, chỉ thiết lập kết nối dữ liệu chỉ khi được yêu cầu.
- ADO.NET sử dụng XML (Extensible Markup Language) để tương tác với cơ sở dữ liệu. Tất cả dữ liệu trong csdl được chuyển thành dạng XML phù hợp với các phép toán quan hệ dữ liệu.

Kiến trúc ADO.NET



Data Provider

- **Data Provider được dùng để cung cấp và duy trì các kết nối CSDL. Data Provider gồm một tập các thành phần liên quan để thực hiện các phép toán trên CSDL và trả về kết quả.**
- **Data Provider trong .NET Framework**
  - .NET Framework Data Provider for **SQL Server**
  - .NET Framework Data Provider for **OLE DB**
  - .NET Framework Data Provider for **ODBC**
  - .NET Framework Data Provider for **Oracle**

Data Provider

- **Lập trình .NET Framework Data Provider cần không gian tên System.Data**
  - **System.Data**: là không gian tên gồm các lớp, interface và enumeration biểu diễn kiến trúc ADO.NET, các thành phần này được kết hợp với các Data Provider
  - **System.Data** còn có các không gian tên phục vụ cho từng loại Data Provider:
    - **System.Data.SqlClient**
    - **System.Data.OleDb**
    - **System.Data.Odbc**
    - **System.Data.OracleClient**

Data Provider

- **Các đối tượng trong Data Provider**
  - **Connection**: Thiết lập một kết nối tới một nguồn dữ liệu cụ thể
  - **Command**: Thực thi một lệnh với một nguồn dữ liệu. Cho phép sử dụng tham số Parameters và có thể thực thi trong một phạm vi với Transaction từ một Connection.
  - **DataReader**: Đọc một dòng dữ liệu hướng tới và chỉ đọc (forward-only, read-only) của dữ liệu từ một nguồn dữ liệu.
  - **DataAdapter**: Đưa dữ liệu vào một DataSet và giải quyết việc cập nhật dữ liệu vào nguồn dữ liệu.

Connection

- Connection cho phép ta tạo một kết nối giữa ứng dụng của ta và CSDL. Connection lưu thông tin được yêu cầu để thiết lập kết nối. Bằng các thiết lập kết nối, ta có thể truy xuất và thao tác dữ liệu trong CSDL.
- Kết nối với SQL Server

```
SqlConnection conn = new SqlConnection("Data Source=localhost;
            Initial Catalog=Bank; Integrated Security=SSPI;");
conn.Open();
Console.WriteLine("Server Version : " + conn.ServerVesion);
Console.WriteLine("Database : " + conn.Database);
conn.Close();
```

Connection

- **Kết nối với OLEDB**

```
OleDbConnection conn = new OleDbConnection(
   "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\\Bank.mdb");
conn.Open;
Console.WriteLine("Server Version : " + conn.ServerVesion);
Console.WriteLine("Database : " + conn.Database);
conn.Close();
```

Connection

- **SqlConnection và các thuộc tính, phương thức, biến cố:**
  - **ConnectionString**: chuỗi được khai báo để mở một kết nối đến CSDL
  - **State**: trạng thái hiện tại của kết nối
  - **Close**(): phương thức đóng kết nối CSDL
  - **CreateCommand**(): phương thức tạo và trả về đối tượng SqlCommand, đối tượng này được kết hợp với SqlConnection
  - **Open**(): phương thức mở một kết nối tới CSDL với thuộc tính được khai báo trong ConnectionString
  - **StateChange**: biến cố này xảy ra khi trạng thái của kết nối thay đổi

Command

- Command cho ta thực thi một lệnh kết với CSDL. Command lưu và thực thi truy vấn SQL thông qua kết nối dữ liệu.
- Command còn cho phép ta thực thi một lời gọi đến thủ tục lưu trữ (stored procedure).
- Command có thể sử dụng để thực thi phát biểu SQL như UPDATE, DELETE, INSERT, hoặc SELECT.
- Đối tượng Command được sử dụng theo từng data provider cụ thể
  - System.Data.SqlClient.SqlCommand
  - System.Data.OleDb.OleDbCommand
- SqlCommand chỉ ra các kiểu phép toán có thể thực hiện với CSDL SQL Server. Tương tự, OleDbCommand cho phép thực thi các phát biểu SQL với các nguồn dữ liệu sử dụng OLE DB provider.

Command

- **Các thuộc tính, phương thức của SqlCommand và OleDbCommand.**
  - **CommandText**: Thuộc tính đặt tả các câu lệnh để thực hiện với nguồn dữ liệu
  - **Connection**: Thuộc tính đặt tả kết nối SqlConnection hoặc OleDbConnection
  - **CommandTimeout**: Phương thức đặt tả thời gian chờ trước khi đi đến kết thúc một lệnh đã thực hiện và sinh ra một lỗi
  - **CommandType**: Phương thức đặt tả một giá trị chỉ ra cách mà CommandText được diễn dịch
  - **ExecuteNonQuery**: Phương thức thực thi phát biểu SQL đến nguồn dữ liệu thông qua kết nối Connection và nhận về **số mẫu tin bị tác động trong nguồn dữ liệu**
  - **ExecuteReader**: Phương thức gởi phát biểu trong CommandText đến kết nối Connection và xây dựng một **đối tượng đọc dữ liệu**
  - **ExecuteScalar**: Phương thức thực hiện câu truy vấn dữ liệu và nhận về **cột đầu tiên của hàng đầu tiê**n trong tập kết quả
  - **ExecuteXmlReader**: Phương thức gởi phát biểu trong CommandText đến với kết nối Connection và xây dựng một **đối tượng XmlReader**

Command

- **Ví dụ SqlCommand**

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "CREATE TABLE ProductDetails(Model varchar(50))";
cmd.ExecuteNonQuery();
```

ExecuteNonQuery

```
using System;
using System.Data.SqlClient;
public class ExecuteNonQueryExample
{
   public static void Main(string[] args)
   {
      string source = "server=(local)\\NetSDK;" +
                      "uid=QSUser;pwd=QSPassword;" +
                      "database=Northwind";
      string select = "UPDATE Customers " +
                      "SET ContactName = 'Bob' " +
                      "WHERE ContactName = 'Bill'";
      SqlConnection  conn = new SqlConnection(source);
      conn.Open();
      SqlCommand cmd = new SqlCommand(select, conn);
      int rowsReturned = cmd.ExecuteNonQuery();
      Console.WriteLine("{0} rows returned.", rowsReturned);
      conn.Close();
}  }
```

ExecuteScalar

```
using System;
using System.Data.SqlClient;
public class ExecuteScalarExample
{
   public static void Main(string[] args)
   {
      string source = "server=(local)\\NetSDK;" +
                      "uid=QSUser;pwd=QSPassword;" +
                      "database=Northwind";
      string select = "SELECT COUNT(*) FROM Customers";
      SqlConnection conn = new SqlConnection(source);
      conn.Open();
      SqlCommand cmd = new SqlCommand(select, conn);
      object o = cmd.ExecuteScalar();
      Console.WriteLine ( o ) ;
   }
}
```

DataReader

- DataReader đọc một dòng dữ liệu (stream) từ CSDL. Nó đọc một hàng tại một thời điểm và di chuyển đến một hàng kế tiếp.
- Ưu điểm của DataReader là tăng hiệu quả của ứng dụng khi nó CHỈ ĐỌC một hàng tại một thời điểm. Tuy nhiên, DataReader yêu cầu phải sử dụng kết nối được mở trong toàn bộ thời gian sống của nó.

DataReader

- **Ví dụ DataReader**

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Customer", conn);
conn.Open();
SqlDataReader reader = cmd.ExecuteReader();
while(reader.Read())
{
    Console.WriteLine("Name: " + reader[1]);
}
```

DataReader

- **Các thuộc tính của SqlDataReader**
  - **FieldCount**: Lấy tổng số cột trong hàng hiện hành
  - **HasRows**: Lấy một giá trị xác định rằng SqlDataReader gồm một hay nhiều hàng
  - **IsClosed**: Lấy một giá trị bool xác định rằng SqlDataReader là mở hay đóng
  - **Item**: Lấy giá trị cột trong một định dạng tự nhiên của nó
  - **RecordAffected**: Lấy tổng số hàng được thay đổi, được thêm vào hoặc được xóa bởi các phát biểu Transact-SQL

DataReader

- **Các phương thức của SqlDataReader**
  - **Close**: Đóng đối tượng SqlDataReader
  - **GetByte**: Lấy giá trị của cột cụ thể theo bye
  - **GetChar**: Lấy giá trị của cột cụ thể theo một ký tự đơn
  - **GetInt32**: Lấy giá trị của cột cụ thể theo một số nguyên 32 bit
  - **GetName**: Lấy tên của cột cụ thể
  - **GetSqlString**: Lấy giá trị của cột cụ thể theo một SqlString
  - **GetString**: Lấy giá trị của cột cụ thể theo định dạng chuỗi
  - **GetValue**: Lấy giá trị của cột cụ thể theo định dạng tự nhiên của nó
  - **Read**: Cho phép SqlDataReader đọc mẫu tin kế tiếp

ExecuteReader

```
using System;
using System.Data.SqlClient;
public class ExecuteReaderExample
{
   public static void Main(string[] args)
   {
      string source = "server=(local)\\NetSDK;" +
                      "uid=QSUser;pwd=QSPassword;" +
                      "database=Northwind";
      string select = "SELECT ContactName,CompanyName FROM Customers";
      SqlConnection conn = new SqlConnection(source);
      conn.Open();
      SqlCommand cmd = new SqlCommand(select, conn);
      SqlDataReader reader = cmd.ExecuteReader();
      while(reader.Read())
      {
         Console.WriteLine("Contact : {0,-20} Company : {1}" ,
                           reader[0] , reader[1]);
}  }  }
```

DataAdapter

- **DataAdapter:**
  - DataAdapter hoạt động như một **cầu nối giữa một DataSet và CSDL**, nó này cập nhật CSDL để phù hợp với dữ liệu trong DataSet.
  - DataAdapter sử dụng **phương thức Fill()** để dựa dữ liệu từ CSDL vào DataSet.
  - **Phương thức Update()** được sử dụng để insert, update, delete dữ liệu từ CSDL để phù hợp với DataSet.
  - DataAdapter tồn tại trong không gian tên **System.Data.Common**
- **Các DataAdapter khác nhau có thể được sử dụng:**
  - SqlDataAdapter
  - OleDbAdapter
  - OdbcDataAdapter
  - OracleDataAdapter

DataAdapter

- **SqlDataAdapter**
  - SqlDataAdapter giúp ta **lưu kết quả của truy vấn vào DataSet** và DataTable.
  - SqlDataAdapter giúp **liên kết giữa DataSet và CSDL SQL Server** cho việc truy xuất và lưu dữ liệu.
  - SqlDataAdapter được sử dụng kết hợp với **SqlConnection** và **SqlCommand** để kết nối đến CSDL SQL Server.
  - SqlDataAdapter tồn tại trong không gian tên **System.Data.SqlClient**

DataAdapter

- **Các thuộc tính và biến cố của SqlDataAdapter**
  - **DeteleteCommand**: Thuộc tính đặt tả phát biểu Transact-SQL hoặc stored procedure được sử dụng để xóa (**DELETE**) các record từ DataSet
  - **InsertCommand**: Thuộc tính đặt tả phát biểu Transact-SQL hoặc stored procedure được sử dụng để chèn (**INSERT**) các record từ DataSet
  - **SelectCommand**: Thuộc tính đặt tả phát biểu Transact-SQL hoặc stored procedure được sử dụng để chọn (**SELECT**) các record từ DataSet
  - **UpdateCommand**: Thuộc tính đặt tả phát biểu Transact-SQL hoặc stored procedure được sử dụng để cập nhật (**UPDATE**) các record từ DataSet
  - **RowUpdate**: Biến cố xảy ra khi **một hàng được cập nhật** trong SQL Server
  - **RowUpdating**: Biến cố xảy ra **trong khi một hàng đang được cập nhật** trong SQL Server

DataAdapter

- **Ví dụ SqlDataAdapter**

```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Student", conn);
DataSet ds = new DataSet();
da.Fill(ds);
da.Update(ds);
```

DataSet

- **DataSet là một biểu diễn các đối tượng CSDL trong bộ nhớ đệm. Các đối tượng CSDL gồm các bảng quan hệ và các ràng buộc.**
- **DataSet tương tác gián tiếp với nguồn dữ liệu. Nó chỉ có thể giữ dữ liệu được lấy từ nguồn dữ liệu. Ta có thể đưa dữ liệu vào DataSet sử dụng DataAdapter và có thể thay đổi dữ liệu được lấy.**
- **Các tính chất của DataSet:**
  - Làm việc với dữ liệu **ngắt kết nối**
  - Làm việc với dữ liệu phân cấp
  - Các thay đổi được lưu vào bộ đệm
  - Hỗ trợ tương tác với XML
  - Hiện thực các chức năng đồng nhất

DataSet

- **Các thuộc tính của DataSet**
  - **DataSetName**: Đặt tả hoặc lấy tên DataSet hiện hành
  - **GetType**: Lấy kiểu của thể hiện hiện hành
  - **Locale**: Đặt tả hoặc lấy thông tin được sử dụng để so sánh các chuỗi trong một bảng
  - **Relations**: Lấy tập hợp của quan hệ mà nó liên kết các bảng và cho phép định hướng từ bảng cha đến bảng con
  - **Tables**: Lấy tập hợp của quan hệ mà các bảng kết hợp và cho phép duyệt từ bảng cha đến bảng con

DataSet

- **Các phương thức của DataSet**
  - **AcceptChanges**: Thực hiện các thay đổi đã thực hiện cho DataSet
  - **Clear**: Xóa bỏ tất cả các hàng của tất cả các bảng trong DataSet
  - **GetXml**: Lấy dữ liệu được lưu trữ theo cách trình bày XML
  - **Merge**: Trộn một DataSet, DataTable cụ thể hay một dãy các DataRow vào DataSet hoặc DataTable
  - **ReadXml**: Đọc XML schema và dữ liệu trong DataSet
  - **RejectChanges**: Chối bỏ tất cả các thay đổi đã thực hiện trong DataSet, kể từ lúc nó được tạo hoặc kể từ lần cuối phương thức AcceptChanges() được gọi
  - **WriteXml**: Ghi dữ liệu XML hoặc XML schema vào DataSet

DataSet

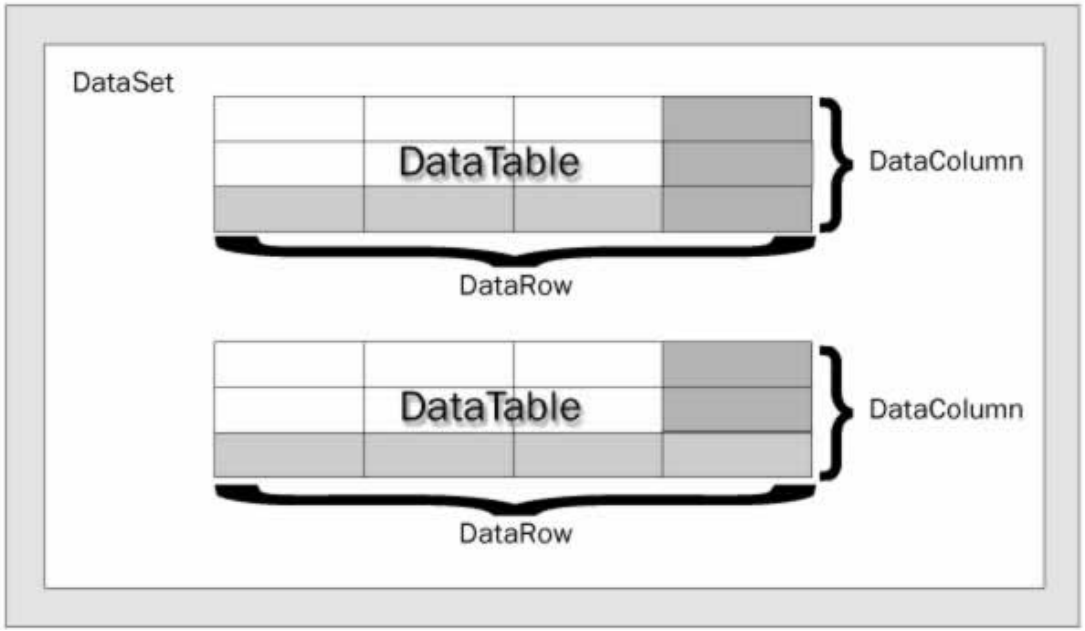
- **Ví dụ DataSet**

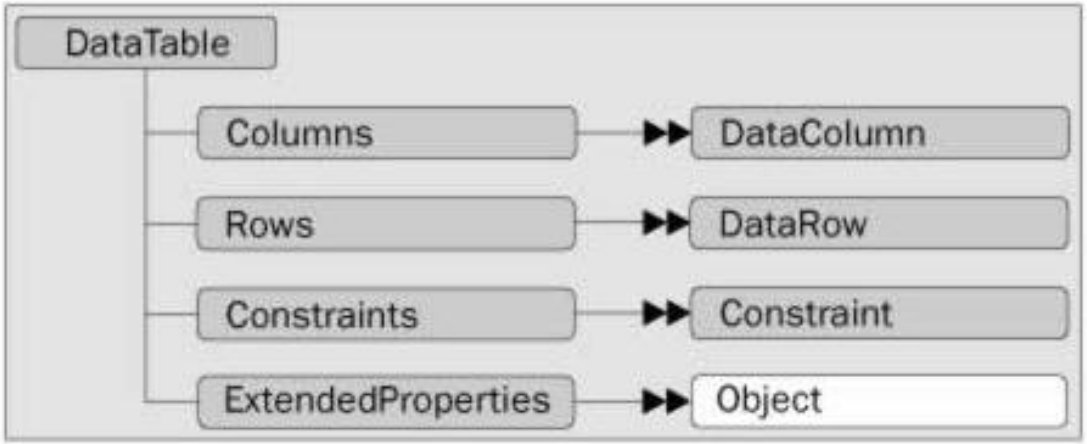
```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Students", conn);
DataSet ds = new DataSet();
ds.DataSetName = "Students";
int totalRecords = da.Fill(ds, "Students");
Console.WriteLine("Total Records: " + totalRecords);
ds.AcceptChanges();
Console.WriteLine("Total Tables: " + ds.Tables.Count);
ds.Clear();
```

DataTable

- **DataTable trình bày một bảng trong DataSet**
  - Được sử dụng để **lưu trữ dữ liệu lấy từ nguồn dữ liệu**
  - Gồm **một tập các hàng và cột** được tổ chức thành dạng bảng.
  - Được sử dụng rộng rãi trong **môi trường ngắt kết nối**
- **DataTable có thể được được hiện thực cho bất kỳ CSDL nào. Nó có thể lưu trữ kết quả của truy vấn SQL, mà có thể đã được truy cập sau đó hoặc được thao tác sử dụng DataRow và DataColumn**
- **DataTable gồm các đối tượng ràng buộc để bảo đảm dữ liệu toàn vẹn trong các bảng quan hệ**

DataTable


DataSet
DataTable
DataColumn
DataRow
DataTable
DataColumn
DataRow


DataTable
Columns
DataColumn
Rows
DataRow
Constraints
Constraint
ExtendedProperties
Object

DataTable

- **Các thuộc tính của DataTable**
  - **ChildRelations**: Lấy một tập các **QUAN HỆ CON** trong DataTable hiện hành
  - **Columns**: Lấy tập các **CỘT** của bảng
  - **DataSet**: Lấy DataSet mà bảng hiện hành đang thuộc về
  - **IsInitialized**: Lấy giá trị chỉ thị rằng có hay không DataTable được khởi tạo
  - **ParentRelations**: Lấy một tập các **QUAN HỆ CHA** trong DataTable hiện hành
  - **Rows**: Lấy tập các **HÀNG** của bảng
  - **TableName**: Đặt tả hoặc lấy tên của DataTable

DataTable

- **Các phương thức của DataTable**
  - **AcceptChanges**: Chấp nhận tất cả các thay đổi đến bảng, kể từ lần cuối phương thức AcceptChanged được gọi
  - **Clear**: Xóa bỏ tất cả dữ liệu trong bảng DataTable
  - **Load**: Lấp đầy DataTable với các mẫu tin từ nguồn dữ liệu sử dụng bằng cách dùng IDataReader. Dữ liệu đã được yêu cầu được kết nối với các hàng đang tồn tại.
  - **NewRow**: Tạo một hàng mới với schema của table
  - **ReadXml**: Đọc XML schema và dữ liệu vào DataTable
  - **Select**: Lấy một dãy các đối tượng DataRow
  - **WriteXML**: Ghi dữ liệu vào DataTable dưới định dạng XML

DataTable

- **Các biến cố của DataTable**
  - **ColumnChanged**: Xảy ra sau khi một giá trị trong DataColumn cụ thể bị thay đổi trong một DataRow
  - **RowChanged**: Xảy ra sau khi một DataRow bị thay đổi
  - **RowDeleted**: Xảy ra sau khi một hàng bị xóa
  - **TableNewRow**: Xảy ra sau khi một DataRow được thêm mới

DataTable

- **Ví dụ DataTabe**

```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Orders", conn);
DataTable table = new DataTable();
table.TableName = "Orders";
table.AcceptChanges();
da.Fill(table);
Console.WriteLine("OrderID: " + table.Rows[0]["OrderID"].ToString());
```

DataTableCollection

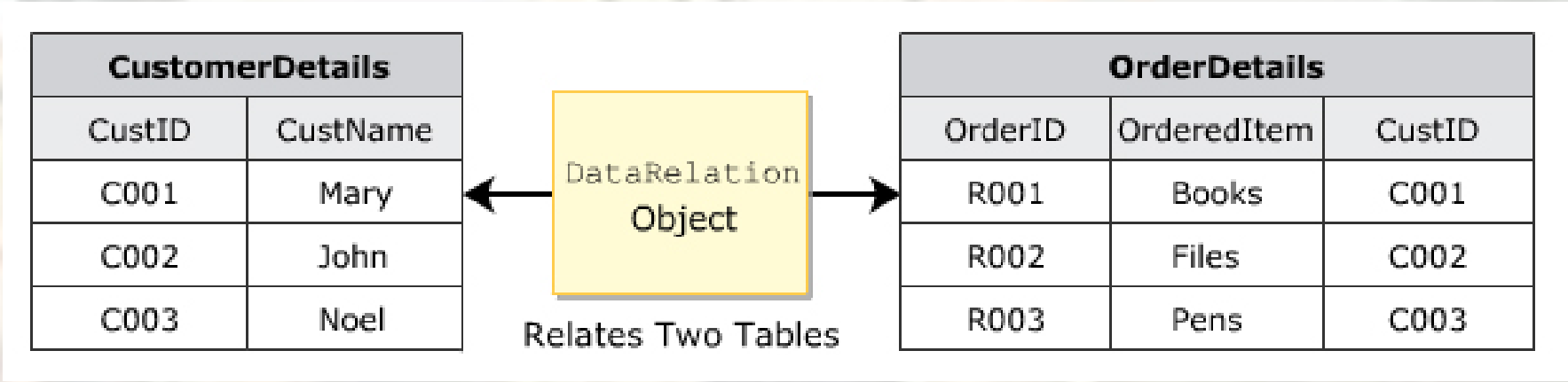
- **DataTableCollection là một tập các bảng trong một DataSet. Nó cho phép ta truy xuất tập hợp các đối tượng DataTable và điều hướng thông qua các bảng trong DataSet**
- **Các phương thức, thuộc tính, biến cố của DataTableCollection**
  - **Count**: thuộc tính lấy tổng số các thành phần trong một collection
  - **Item**: thuộc tính lấy một đối tượng DataTable trong collection
  - **Add**: phương thức thêm mới một đối tượng DataTable vào collection
  - **Clear**: phương thức xóa bỏ rất cả DataTable trong collection
  - **Remove**: phương thức gỡ bỏ một DataTable cụ thể ra khỏi collection
  - **CollectionChanged**: biến cố xảy ra sau khi thêm hay hủy các đối tượng trong collection

DataTableCollection

```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Orders", conn);
DataSet ds = new DataSet();
da.Fill(ds, "Orders");
da = new SqlDataAdapter("SELECT * FROM [Customers]", conn);
da.Fill(ds, "Customers");
DataTableCollection tableCollection = ds.Tables;
foreach (DataTable t in tableCollection)
{
   Console.WriteLine("Table name: " + t.TableName);
}
DataTable tableSuppliers = new DataTable("Suppliers");
da = new SqlDataAdapter("SELECT * FROM Suppliers", conn);
da.Fill(tableSuppliers);
tableCollection.Add(tableSuppliers);
foreach (DataTable t in tableCollection)
{
   Console.WriteLine("Table name: " + t.TableName);
}
```

DataRelation

- DataRelation trình bày mối quan hệ cha con giữa các bảng trong DataSet. DataRelation cho phép điều hướng giữa các DataTable trong cùng DataSet.
- DataRelation duy trì ràng buộc toàn vẹn bằng cách sử dụng ràng buộc khóa ngoại.



DataRelation

- **Ví dụ DataTabe**
  - Tình huống sử dụng bảng **Invoice**, **ProductDetails**, 2 bảng này quan hệ với nhau bằng khóa **InvoiceNo**

```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Invoice", conn);
DataSet ds = new DataSet();
da.Fill(ds, "Invoice");
da = new SqlDataAdapter("SELECT * FROM ProductDetails", conn);
da.Fill(ds, "ProductDetails");
DataRelation dr = new DataRelation("InvoiceDetails",
       ds.Tables["Invoice"].Colums["InvoiceNo"],
       ds.Tables["ProductDetails"].Columns["InvoiceNo"]);
ds.Relations.Add(dr);
```

DataColumn

- **DataColumn được sử dụng để tạo một schema cho một cột cụ thể. Nó là yếu tốt cơ bản của DataTable. Schema được sử dụng để định nghĩa kiểu dữ liệu và các ràng buộc. Ta có thể thêm các đối tượng DataColumn vào DataColumnCollection để tạo schema của DataTable.**
- **Thuộc tính của DataColumn**
  - **Caption**: Đặt tả hoặc lấy tiêu đề cột
  - **ColumnName**: Đặt tả hoặc lấy tên cột trong DataColumnCollection
  - **DataType**: Đặt tả hoặc lấy kiểu dữ liệu được lưu trong cột
  - **DefaultValue**: Đặt tả hoặc lấy giá trị mặc định của cột khi tên một hàng mới
  - **MaxLength**: Đặt tả hoặc lấy chiều dài tối đa của một cột văn bảng
  - **Table**: Lấy một DataTable mà cột này đang thuộc về
  - **Unique**: Đặt tả hoặc lấy giá trị mà nó chỉ ra rằng giá trị trong mỗi cột là duy nhất

DataColumn

- **Ví dụ DataColumn**

```
DataTable table = new DataTable("Products");
DataColumn column = new DataColumn();
column.DataType = Type.GetType("System.Decimal");
column.Caption = "Price";
column.ColumnName = "Price";
column.DefaultValue = 25;
table.Columns.Add(column);
```

DataRow

- **DataRow trình bày một hàng trong DataTable. Đối tượng của lớp là yếu tố cơ bản của DataTable. DataRow được sử dụng để chèn thêm, cập nhật và xóa bỏ các giá trị trong DataRable.**
- **Các thuộc tính và phương thức của DataRow**
  - **Item**: Thuộc tính đặt tả và lấy dữ liệu được lưu trữ trong một cột cụ thể
  - **RowError**: Thuộc tính đặt tả và lấy mô tả lỗi do người dùng định nghĩa cho một hàng
  - **RowState**: Thuộc tính lấy trạng thái của hàng hiện tại trong quan hệ với DataRowCollection. Lớp DataRowCollection trình bày một dãy các hàng trong DataTable. Các giá trị trong thuộc tính này được định nghĩa trong kiểu liệt kê DataRowState
  - **Table**: Thuộc tính lấy DataTable
  - **AcceptChanges()**: Phương thức chấp nhận thay đổi hàng hiện tại, kể từ lần cuối phương thức này được gọi
  - **Delete()**: Phương thức này xóa bỏ DataRow
  - **SetAdded()**: Phương thức này thao tác RowState của DataRow để được thêm vào

DataRow

- **Ví dụ DataRow**

```
DataTable table = new DataTable("Employees");
DataColumn column = new DataColumn();
column.DataType = Type.GetType("System.String");
column.Caption = "EmployeeID";
column.ColumnName = "EmployeeID";
table.Columns.Add(column);
DataRow row = table.NewRow();
row["EmployeeID"] = "E001";
row.RowError = "An error has taken place while dealing with a row";
table.Rows.Add(row);
```

DataRow & DataColumn

```
static void GetSchemaInfo(SqlConnection connection)
{
    using (connection)
    {
        SqlCommand command = new SqlCommand(
          "SELECT CategoryID, CategoryName FROM Categories;",
          connection);
        connection.Open();

        SqlDataReader reader = command.ExecuteReader();
        DataTable schemaTable = reader.GetSchemaTable();

        foreach (DataRow row in schemaTable.Rows)
        {
            foreach (DataColumn column in schemaTable.Columns)
            {
                Console.WriteLine(String.Format("{0} = {1}",
                   column.ColumnName, row[column]));
            }
}   }   }
```

TỔNG KẾT

- **.Net Framework Data Providers**
  - Connection
  - Command
  - DataReader
  - DataAdapter
- **DataSet**
  - DataTable
  - DataColumn
  - DataRow