

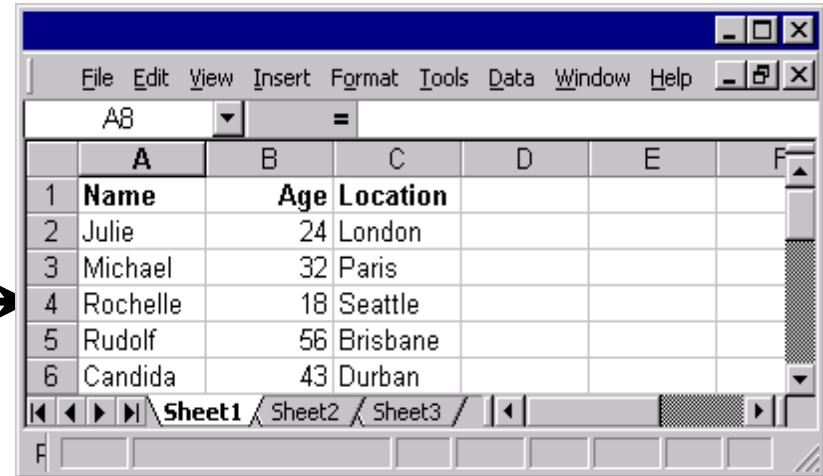
# ADO.NET

GV: Đặng Hữu Nghị

# Lập trình cơ sở dữ liệu

Data

Stored  
into



	A	B	C	D	E	F
1	<b>Name</b>	<b>Age</b>	<b>Location</b>			
2	Julie	24	London			
3	Michael	32	Paris			
4	Rochelle	18	Seattle			
5	Rudolf	56	Brisbane			
6	Candida	43	Durban			

Database



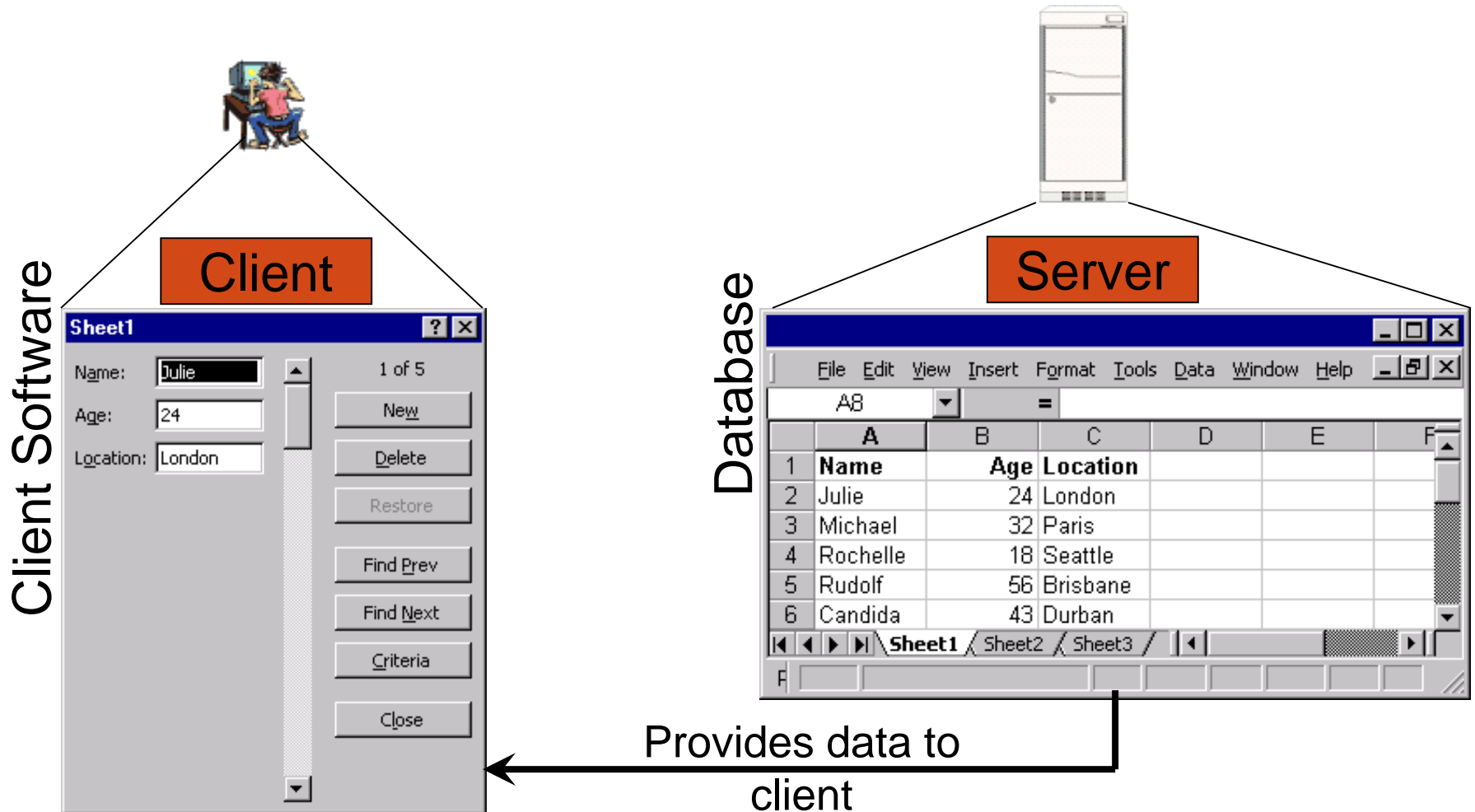
Client

ADO.net

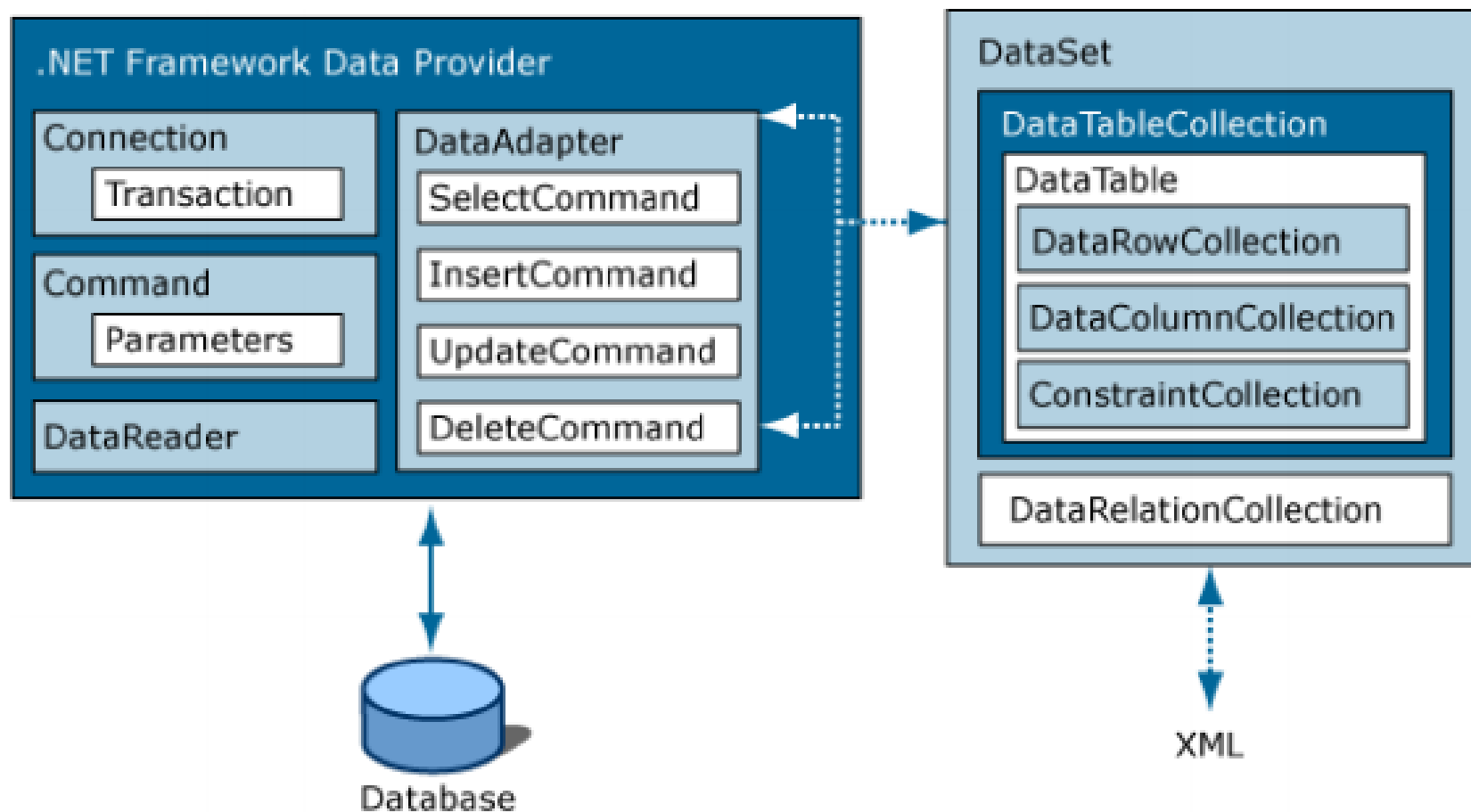
Data access technology

# Lập trình cơ sở dữ liệu

## Client-Server Application



## 5.1 Kiến trúc tổng quan của ADO.NET



Kiến trúc của ADO.NET bao gồm hai thành phần chính: *Thành phần truy cập nguồn dữ liệu* và *thành phần lưu trữ xử lý dữ liệu*

## 5.1 Kiến trúc tổng quan của ADO.NET

- Trong ADO.NET, có 4 đối tượng chính với các chức năng cơ bản như sau:
  - Connection: giúp thực hiện kết nối đến các CSDL
  - Command: giúp truy cập đến CSDL và thực hiện các phát biểu SQL hay thủ tục lưu trữ sẵn (stored procedure) của CSDL
  - DataReader: dùng để đọc nhanh nguồn dữ liệu, chỉ được duyệt tuần tự theo chiều tiến của các record
  - DataAdapter: dùng để chuyển dữ liệu truy vấn được cho các đối tượng lưu trữ và xử lý (DataSet, DataTable). DataAdapter chủ yếu thực hiện các thao tác như SELECT, INSERT, UPDATE, DELETE

## 5.1 Kiến trúc tổng quan của ADO.NET

- .NET Framework Data Provider Cung cấp giao diện lập trình chung để làm việc với các nguồn dữ liệu
- Mỗi *nhà cung cấp* sẽ đưa ra một loại data provider riêng

Interface	SQL Server Provider	Oracle Provider	OLEDB Provider	ODBC Provider
<b>IDbConnection</b>	SqlConnection	OracleConnection	OleDbConnection	OdbcConnection
<b>IDbDataAdapter</b>	SqlDataAdapter	OracleDataAdapter	OleDbDataAdapter	OdbcDataAdapter
<b>IDbCommand</b>	SqlCommand	OracleCommand	OleDbCommand	OdbcCommand
<b>IDbDataReader</b>	SqlDataReader	OracleDataReader	OleDbDataReader	OdbcDataReader

- Để sử dụng data provider nào, chúng ta phải tiến hành khai báo using namespace tương ứng. Chẳng hạn, using System.Data.SqlClient;

## 5.1 Kiến trúc tổng quan của ADO.NET

- *Thành phần thứ hai* trong kiến trúc ADO.NET là DataSet.
- DataSet được xem như là container dùng để lưu trữ đối tượng liên quan đến dữ liệu như DataTable, DataRelation, DataView.
- DataSet như là một CSDL thu nhỏ tại máy client, có thể chứa các đối tượng table, view, constraint, relationship giữa các table, ...

## 5.2 Tổng quan về các mô hình xử lý dữ liệu trong ADO.NET

- **5.2.1 Mô hình Kết nối**
- **5.2.2 Mô hình Ngắt Kết nối**

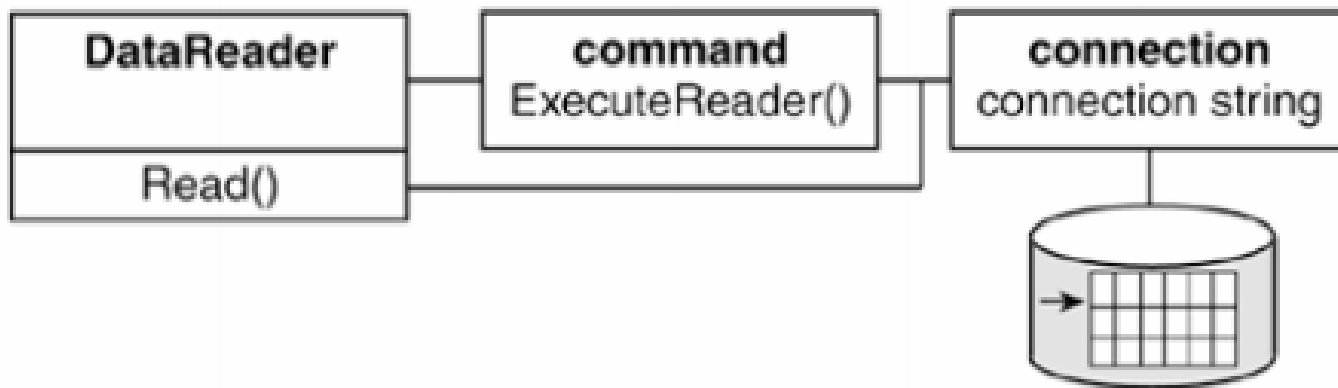


## 5.2.1. Mô hình Kết nối

- Mỗi user có một kết nối cố định tới data source
- Ưu điểm
  - Môi trường được bảo vệ tốt
  - Kiểm soát được sự đồng bộ
  - Dữ liệu luôn được mới
- Nhược
  - Phải có một kết nối mạng cố định
  - Scalability

## 5.2.1. Mô hình Kết nối

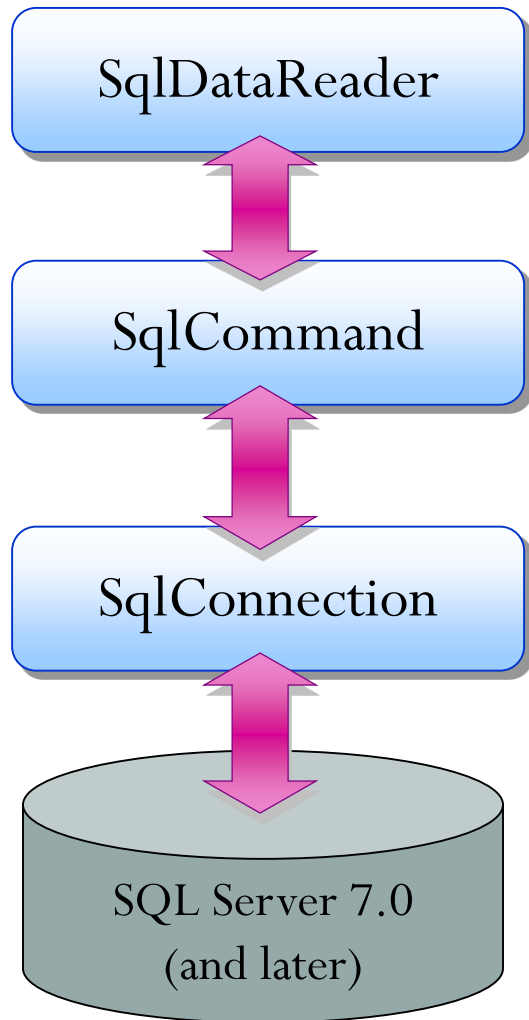
- Trong mô hình kết nối của ADO.NET, có một connection hoạt động được duy trì giữa đối tượng *DataReader* của ứng dụng và một *data source*



## 5.2.1. Mô hình Kết nối

- Các bước điển hình để làm việc với đối tượng DataReader là như sau:
  1. Tạo đối tượng Connection bằng cách truyền một chuỗi Connection string cho hàm khởi dựng của nó.
  2. Khởi tạo một biến chuỗi và gán cho câu lệnh SQL dựa theo dữ liệu muốn nạp về.
  3. Khởi tạo một đối tượng Command từ với nội dung câu lệnh SQL đã xác định ở trên.
  4. Tạo đối tượng DataReader bằng cách thực thi phương thức Command.ExecuteReader(). Đối tượng này sau đó sẽ được dùng để đọc kết quả của câu truy vấn mỗi dòng một lần.

## 5.2.1. Mô hình Kết nối



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

```
...
```

```
// (1) Tao Connection
```

```
SqlConnection cn = new SqlConnection(chuoiKetNoi);  
cn.Open();
```

```
// (2) Chuoi SQL thuc hien lay danh sach ten cac sinh vien xep tang dan  
theo NgaySinh
```

```
string sql = "SELECT HoTen FROM SinhVien ORDER BY NgaySinh";
```

```
// (3) Tao doi tuong Command
```

```
SqlCommand cmd = new SqlCommand(sql, conn);
```

```
// (4) Tao doi tuong DataReader
```

```
DbDataReader rdr;
```

```
rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
```

```
while (rdr.Read())
```

```
    listBox1.Items.Add(rdr["HoTen"]); // Fill ListBox  
rdr.Close(); // Dong datareader sau khi da su dung xong
```

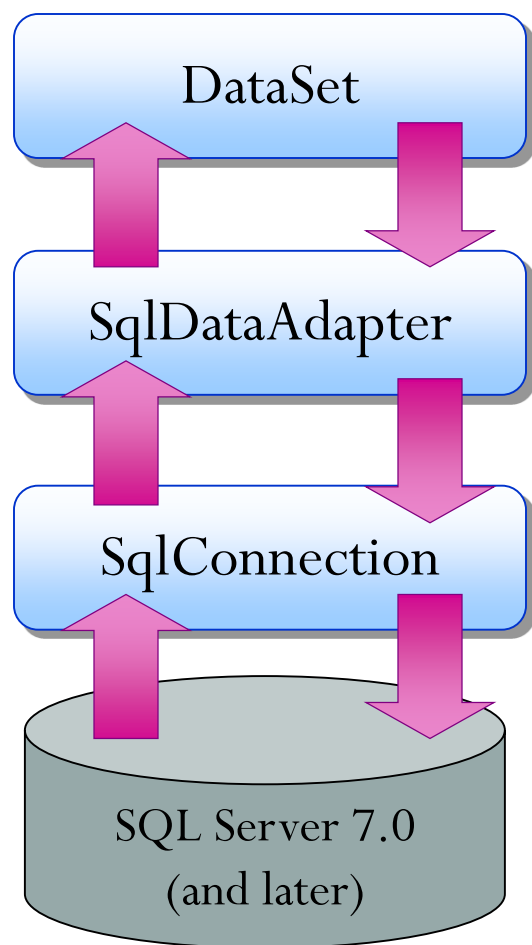
## 5.2.1. Mô hình Kết nối

- Triết lý của mô hình Ngắt kết nối đó là:
  - Dữ liệu được nạp – sử dụng một lệnh SQL – từ nguồn dữ liệu bên ngoài vào bộ nhớ đệm tại máy client;
  - tập kết quả được xử lý tại máy cục bộ;
  - mọi cập nhật sau đó sẽ được truyền từ dữ liệu trong bộ nhớ ngược trở lại nguồn dữ liệu.
- Mô hình được gọi là “ngắt kết nối” bởi vì đối tượng kết nối chỉ được mở đủ lâu để đọc dữ liệu từ nguồn dữ liệu và tiến hành các thao tác cập nhật

## 5.2.2 Mô hình Ngắt Kết nối

- Một tập con của dữ liệu trung tâm được sao chép và bổ sung độc lập, sau đó sẽ được merge lại vào dữ liệu trung tâm.
- Ưu điểm
  - Có thể làm việc bất cứ lúc nào, cũng như có thể kết nối bất kỳ vào Data Source
  - Cho phép user khác có thể kết nối
  - Nâng cao hiệu suất thực hiện của ứng dụng
- Khuyết
  - Dữ liệu không được cập nhật một cách nhanh nhất
  - Sự tranh chấp có thể xuất hiện và phải giải quyết

## 5.2.2 Mô hình Ngắt Kết nối



- Các thành phần chính của mô hình ngắt kết nối đó là DataAdapter và DataSet.
- DataAdapter làm nhiệm vụ như là cầu nối giữa nguồn dữ liệu và DataSet, nạp dữ liệu vào các bảng của DataSet và đẩy các thay đổi ngược trở lại nguồn

## 5.2.2 Mô hình Ngắt Kết nối

- Trong số các phương thức và thuộc tính của DataAdapter thì Fill() và Update() là hai phương thức quan trọng nhất.
- Fill() chuyển một query đến cơ sở dữ liệu và lưu tập kết quả trả về trong một DataTable nào đó;
- phương thức Update() thực hiện một thao tác thêm, xóa, cập nhật dựa trên những thay đổi của đối tượng DataSet.
- Các lệnh cập nhật thực sự được chứa trong các thuộc tính của DataAdapter.





## 5.2.2 Mô hình Ngắt Kết nối

```
string sql = "SELECT MaSinhVien, HoTen, NgaySinh FROM SinhVien";  
string connStr = "Data Source=MYSERVER;Initial Catalog=qlsinhvien; User  
Id=k28;Password=k28;";  
// (1) Tao doi tuong data adapter  
SqlDataAdapter da = new SqlDataAdapter(sql, connStr);  
// (2) Tao doi tuong dataset  
DataSet ds = new DataSet();  
// (3) Tao mot Table co ten "SinhVien" trong dataset va nap du lieu cho no  
da.Fill(ds, "SinhVien");  
// (4) Hien thi danh sach ten sinh vien ra list box  
DataTable dt = ds.Tables["SinhVien"];  
for (int i=0; i< dt.Rows.Count;i++)  
{ DataRow row = dt.Rows[i];  
listBox1.Items.Add(row["HoTen"]); }
```



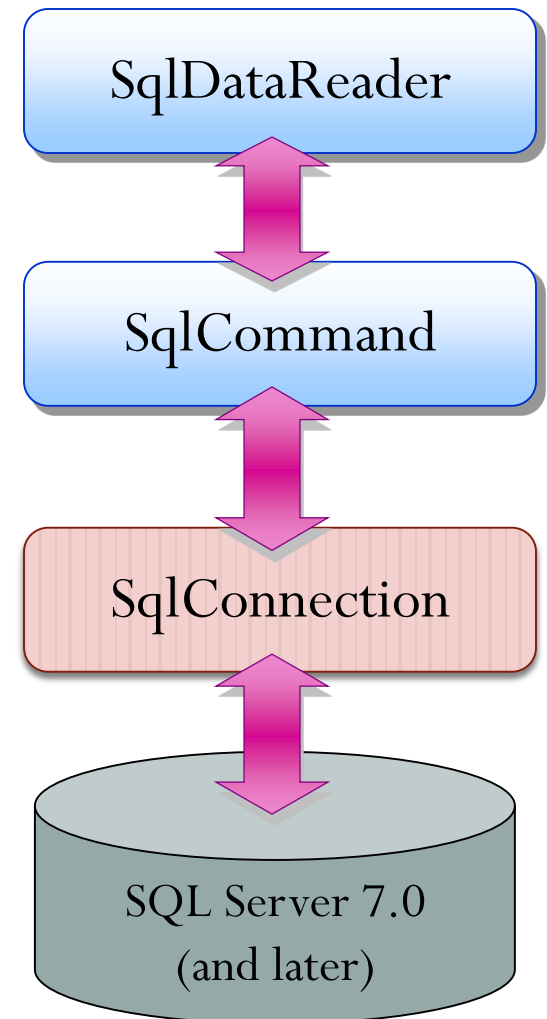
## 5.3 Làm việc với mô hình Kết nối trong ADO.NET

### 5.3.1. Lớp Connection

### 5.3.2. Đối tượng Command

### 5.3.3. Đối tượng DataReader

### 5.3.4. Bài thực hành



## 5.3.1 Lớp Connection

- Đối tượng Connection là đối tượng chịu trách nhiệm quản lý kết nối tới nguồn dữ liệu (DataSource).
- Có 2 dạng Connection tương ứng với 2 kiểu dữ liệu SQL Server và OLE DB đó là : SqlConnection và OleDbConnection.

## 5.3.1.1 Connection string

- Đối tượng Connection của ADO.NET chỉ nhận một tham số đầu vào là chuỗi kết nối (connection string).
- Chuỗi kết nối chứa các thông tin cơ bản để thực hiện kết nối đến một SQL Server, các thông tin được chứa theo cặp key=value;
  - ví dụ chuỗi *"key 1=value1;key 2=value2"*, ...
  - dưới đây tham khảo một số key

KEY	Mô tả
Connect Timeout	Số giây cố gắng kết nối trước khi phát sinh lỗi (mặc định 15s)
Timeout	
Connection Timeout	
Data Source	Key này dùng để gán địa chỉ mạng (tên máy hoặc IP hoặc domain) của SQL Server, hoặc là tên của hiện hành đang chạy của SQL Server.
Server	
Address	
Addr	
Initial Catalog	Tên của Database
Database	
Intergrated security	Integrated Security sẽ bảo mật khi bạn làm việc trên một máy đơn. Tuy nhiên, bạn sẽ thường xuyên cần phải định rõ mức bảo mật dựa trên SQL Server User ID với quyền hạn được xác định cho ứng dụng bạn sử dụng
Password	Password để kết nối
PWD	
User ID	Tài khoản (account) dùng để đăng nhập
UID	

## 5.3.1.1 Connection string

- Ví dụ:

```
string connectionString = "server = HN-SQLEXPRESS; Database= QLDH; User  
ID=sa;password=123456“
```

```
string connectionString = "Server=192.168.1.10;Database=exampledb;User  
Id=testuser;Password=testpass;";
```

```
string connectionString = "server =(local); database = QLDH; integrated security = true;";
```

```
string connectionString = "server =.; database = QLDH; integrated security = true;";
```

```
string connectionString = "Data Source=(local);Initial Catalog=Northwind;Integrated Security=SSPI");
```

### 5.3.1.2. Một số phương thức, thuộc tính SqlConnection

State	<p>Kiểu ConnectionState trạng thái kết nối:</p> <ul style="list-style-type: none"> <li>- ConnectionState.Closed kết nối đã đóng</li> <li>- ConnectionState.Connecting đang kết nối</li> <li>- ConnectionState.Executing đang thi hành lệnh nào đó</li> <li>- ConnectionState.Fetching đang nhận dữ liệu về</li> <li>- ConnectionState.Open kết nối đang mở</li> </ul> <p>Để kiểm tra cần thực hiện phép toán bitwise bằng phương thức FlagsAttribute, ví dụ:</p> <pre>if ((connection.State.HasFlag(ConnectionState.Open)) &amp;&amp; (connection.State.HasFlag(ConnectionState.Fetching))) { Console.WriteLine("Kết nối mở và đang nhận dữ liệu"); }</pre>
Database	Trả về tên Database - sau khi kết nối mở
StatisticsEnabled	Mặc định là false, nếu thiết lập bằng true thì nó cho phép thu thập thông tin về kết nối. Để lấy thông tin thu thập được dùng phương thức RetrieveStatistics()
Open()	Mở kết nối, sử dụng OpenAsync() nếu dùng kỹ thuật async
Close()	Đóng kết nối
CreateCommand()	Tạo đối tượng SqlCommand để thực hiện các lệnh SQL
RetrieveStatistics()	Lấy thông tin thống kê (trả về IDictionary)
StateChange	Event - phát sinh khi thay đổi trạng thái kết nối, muốn bắt sự kiện gán nó bằng delegate dạng (object sender, StateChangeEventArgs e) => { /.. }



## 5.3.1.2. Một số phương thức, thuộc tính SqlConnection

- Ví dụ: Kết nối với CSDL thông qua SqlConnection
- + Để sử dụng thư viện provider của SQL bạn cần khai báo

*using System.Data.SqlClient;*

- + Xây dựng chuỗi kết nối

*string chuoi = "server =.; database = QLDHang;  
integrated security = true;"*

- + Khai báo đối tượng kết nối và Cấp phát đối tượng

*SqlConnection conn = new SqlConnection(chuoi);*

## 5.3.1.2. Một số phương thức, thuộc tính SqlConnection

// Mở kết nối

conn.Open

// ..... Các lệnh truy xuất dữ liệu

// Đóng kết nối

Conn.Close

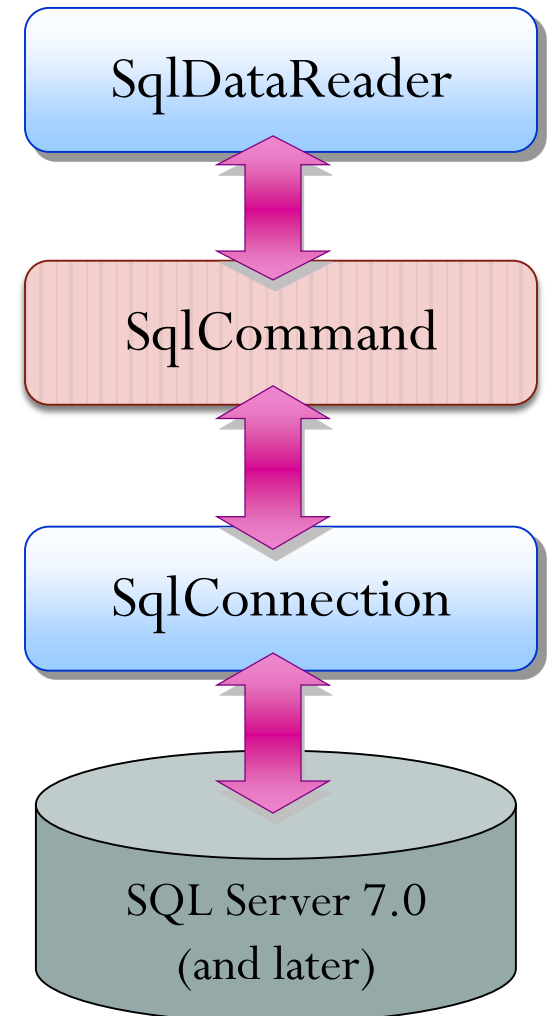
## 5.3 Làm việc với mô hình Kết nối trong ADO.NET

### 5.3.1. Lớp Connection

### 5.3.2. Đối tượng Command

### 5.3.3. Đối tượng DataReader

### 5.3.4. Bài thực hành



## 5.3.2.1 Tạo đối tượng Command

- Sau khi một đối tượng connection được tạo ra, bước tiếp là tạo ra một đối tượng Command để gửi một query (select) hay một action command (thêm, xóa, sửa) đến data source
- Ví dụ:

```
SqlConnection conn = new SqlConnection(connstr);  
conn.open();  
string sql =  
"INSERT INTO SinhVien (MaSinhVien, HoTen) VALUES (@pMaSinhVien, @pHoTen)";  
SqlCommand cmd = new SqlCommand();  
cmd.Connection = conn;  
cmd.CommandText = sql;  
cmd.Parameters.AddWithValue("@pMaSinhVien", 12);  
cmd.Parameters.AddWithValue("@pHoTen", "tnv spider");
```

## 5.3.2.2. Các thuộc tính của *SqlCommand*

- **CommandType**: Thiết lập hoặc lấy kiểu lệnh tương tác, lệnh tương tác này có thể là stored procedure hoặc là câu lệnh truy vấn.
- **CommandText**: Thiết lập hoặc lấy lệnh thao tác với dữ liệu. Lệnh này có thể là tên của stored procedure đã có sẵn trong cơ sở dữ liệu hoặc là câu lệnh truy vấn tùy thuộc vào thuộc tính CommandType.
- **CommandTimeout**: Thiết lập hoặc lấy thời gian chờ thực hiện lệnh. Sau khoảng thời gian này nếu tương tác cơ sở dữ liệu vẫn chưa xong thì chương trình sẽ báo lỗi.
- **Parameters**: Các tham số truyền vào cho đối tượng command. Thuộc tính này được sử dụng hiệu quả khi CommandType là stored procedure.
- **Connection**: Thiết lập hoặc lấy kết nối đang được đối tượng SqlCommand sử dụng.

### 5.3.2.3 Thực thi một Command

- Lệnh SQL được gán trong thuộc tính CommandText của đối tượng Command sẽ được thực thi bằng một trong các phương thức được chỉ ra dưới đây:
- ExecuteReader:
  - Thực thi câu lệnh CommandText của đối tượng SqlCommand và trả về kiểu SqlDataReader.
  - Phương thức này thường được sử dụng khi nội dung câu lệnh tương tác cơ sở dữ liệu là lệnh *select*.

```
cmd.CommandText = "SELECT * FROM SinhVien" + "WHERE YEAR(NgaySinh)  
> 1981";  
SqlDataReader rdr= cmd.ExecuteReader();
```

## 5.3.2.3 *Thực thi một Command*

- ExecuteNonQuery:
  - Thực thi câu lệnh CommandText của đối tượng SqlCommand
  - đây là dạng câu lệnh cập nhật cơ sở dữ liệu (thêm hoặc xoá hoặc sửa) nên chỉ trả về số dòng bị ảnh hưởng mà không trả về dòng dữ liệu nào.

```
cmd.CommandText = "DELETE SinhVien WHERE MaSinhVien=12";  
int soLuong = cmd.ExecuteNonQuery();
```

## 5.3.2.3 *Thực thi một Command*

- ExecuteScalar:
  - Thực thi câu truy vấn của đối tượng Command và chỉ trả về cột đầu tiên của dòng đầu tiên của kết quả. Các kết quả còn lại bị bỏ qua.

```
cmd.CommandText="SELECT COUNT(MaSinhVien) FROM SinhVien";  
int soSinhVien = (int)cmd.ExecuteScalar();
```



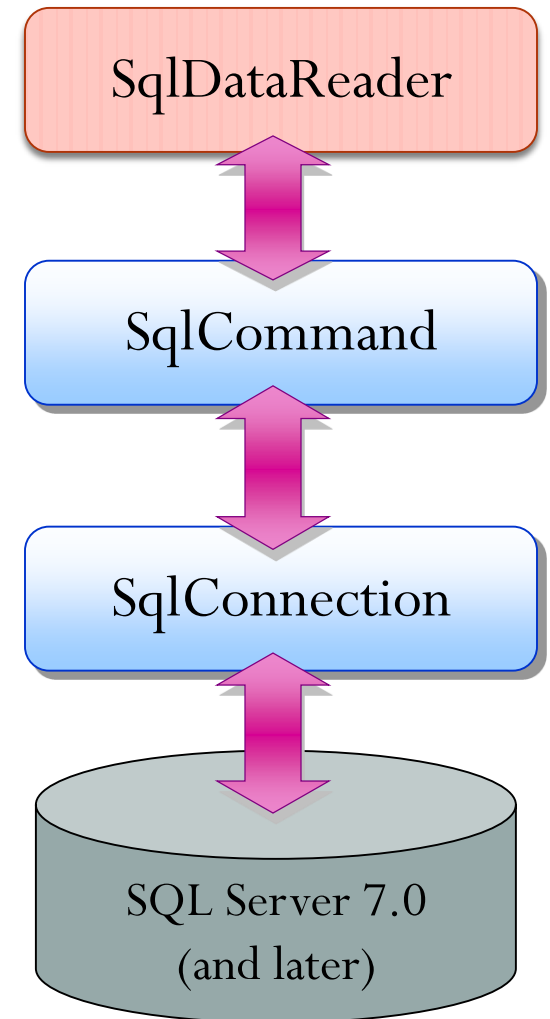
## 5.3 Làm việc với mô hình Kết nối trong ADO.NET

**5.3.1. Lớp Connection**

**5.3.2. Đối tượng Command**

**5.3.3. Đối tượng DataReader**

**5.3.4. Bài thực hành**



## 5.3.3. Đối tượng DataReader

- Một DataReader cho phép lấy các dòng và cột dữ liệu của dữ liệu trả về khi thực thi một query.
- Dữ liệu sau khi được truy vấn từ cơ sở dữ liệu sẽ được lưu trữ trong SqlDataReader dưới dạng bảng gồm nhiều dòng và nhiều cột giống như trong cơ sở dữ liệu.
- Dữ liệu trong SqlDataReader được truy xuất một cách tuần tự và một chiều.
- SqlDataReader không cung cấp cơ chế sắp xếp cũng như cơ chế truy xuất ngẫu nhiên

## *Các thuộc tính của SqlDataReader*

- **FieldCount**: thuộc tính này trả về số trường trong *record* hiện hành.
- **IsClosed**: thuộc tính này trả về trạng thái của SqlDataReader là đóng hay mở.
- **HasRows**: thuộc tính này chỉ định SqlDataReader có *record* dữ liệu nào hay không.

## *Các phương thức của SqlDataReader*

- **Close:** phương thức này thực hiện việc đóng SqlDataReader và giải phóng tài nguyên.
- **GetBoolean, GetByte, GetChar, GetDateTime, GetDecimal:** lấy các giá trị tại cột đang xét tùy vào kiểu dữ liệu.
- **GetValue, GetValues:** lấy về giá trị hoặc tập giá trị ở dạng “nguyên thủy” ( kiểu dữ liệu gốc của Database).
- **Read:** Đọc *record* tiếp theo của DataReader

## 5.3.3. Đối tượng DataReader

- Ví dụ 1:

```
string q1 = "SELECT * FROM SinhVien WHERE YEAR(NgaySinh) < 1981";
string q2 = "SELECT * FROM SinhVien WHERE YEAR(NgaySinh) > 1990";
cmd.CommandText = q1 + ";" + q2; // hai query được ngăn cách nhau bởi dấu ;
DbDataReader rdr = cmd.ExecuteReader();
bool readNext = true;
while (readNext)
{
    while (rdr.Read())
        MessageBox.Show(rdr.GetString(1));
    readNext = rdr.NextResult();
    // kiểm tra xem còn tập dữ liệu nào nữa không
}
rdr.Close();
conn.Close();
```

## 5.3.3. Đối tượng DataReader

- Ví dụ 2:

```
cmd.CommandText = "SELECT MaSinhVien, Hoten, GioiTinh, NgaySinh FROM  
SinhVien " + "WHERE YEAR(NgaySinh) = 1981";  
dr = cmd.ExecuteReader();  
dr.Read(); // Các cách để lấy dữ liệu kiểu string ở cột thứ 2 (HoTen)  
string stHoTen;  
stHoTen = dr.GetString(1);  
stHoTen = (string) dr.GetSqlString(1); // SqlClient provider  
stHoTen = (string)dr.GetValue(1);  
stHoTen = (string)dr["HoTen"];  
stHoTen = (string)dr[1];  
// Lấy dữ liệu kiểu DateTime ở cột thứ 4 (NgaySinh) có kiểm tra giá trị NULL  
if (!dr.IsDBNull(3))  
    DateTime dtNgaySinh = dr.GetDateTime(3);
```

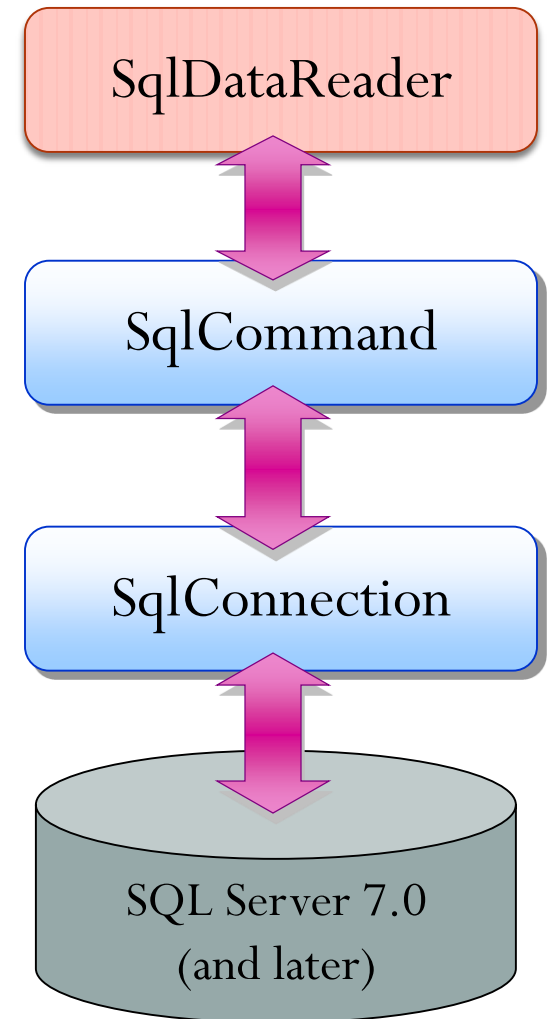
## 5.3 Làm việc với mô hình Kết nối trong ADO.NET

**5.3.1. Lớp Connection**

**5.3.2. Đối tượng Command**

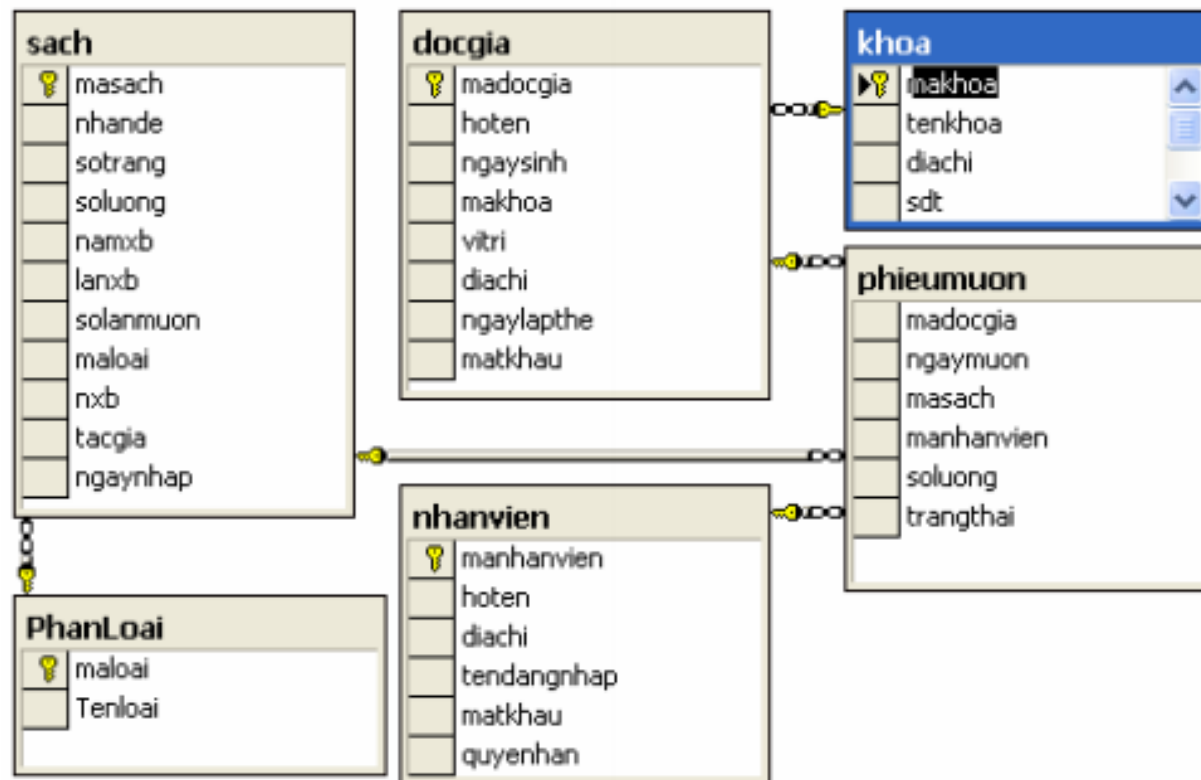
**5.3.3. Đối tượng DataReader**

**5.3.4. Bài thực hành**



## 5.3.4. Bài thực hành

- Tạo CSDL quanlythuvien trong SQL Server có quan hệ như sau:





## 5.3.4. Bài thực hành

- *Bài thực hành về đối tượng Connection, Command và DataReader*

Tạo mới một tài khoản

Manv	Họ tên	Địa chỉ	Tên đăng nhập	Quyền hạn

ListView1

Thông tin cơ bản

Mã nhân viên

Họ tên

Địa chỉ

Tên đăng nhập

Quyền hạn

< > >|

Tạo mới

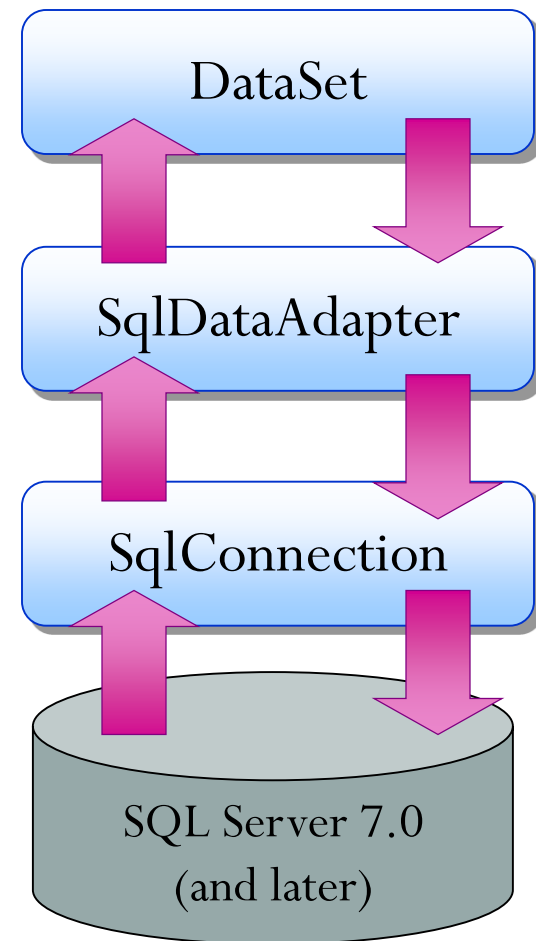
Tìm kiếm

Xoá bỏ

Thoát

## 5.4 Làm việc với mô hình Ngắt kết nối

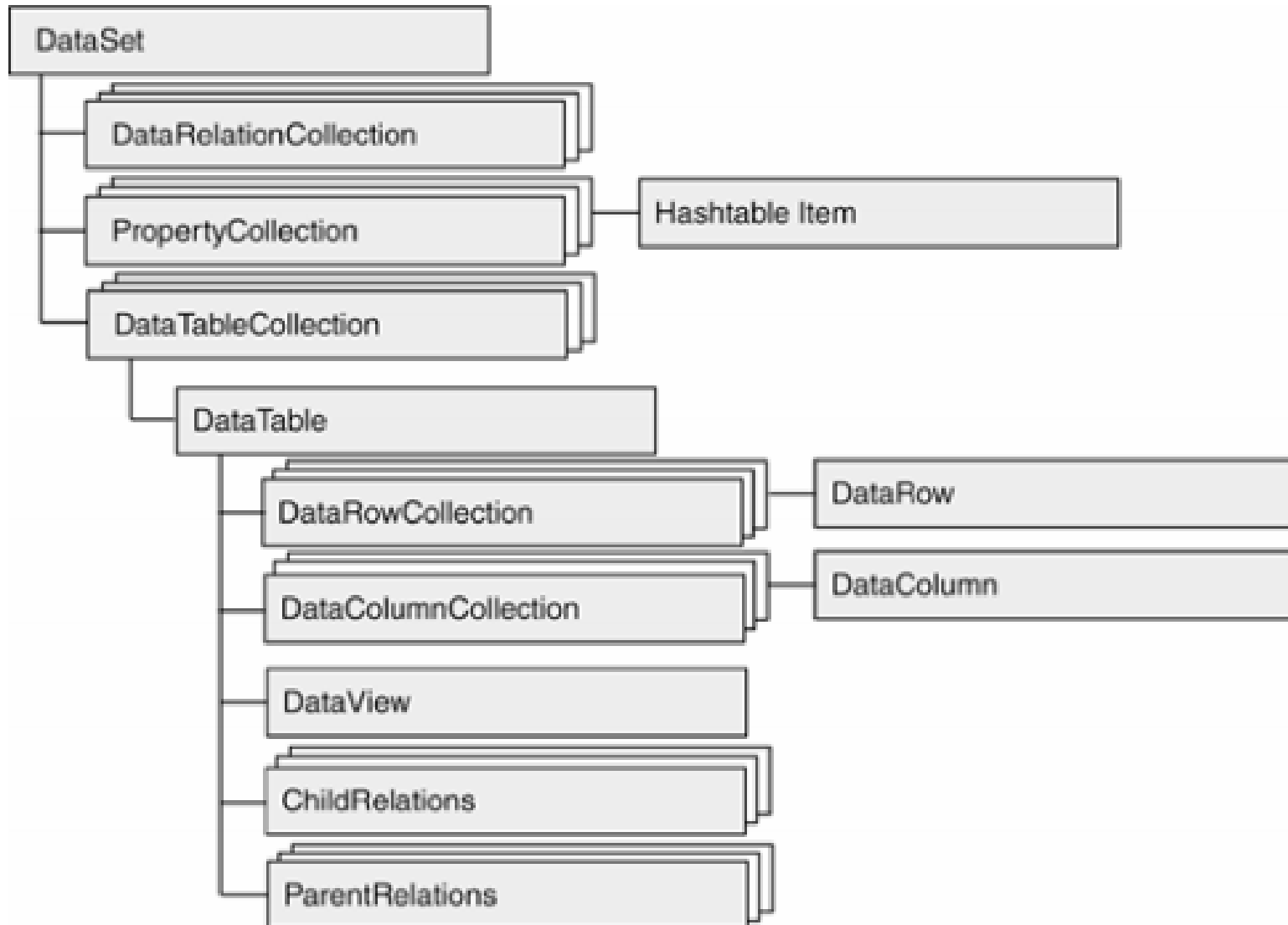
- Mô hình Ngắt Kết nối sử dụng đối tượng DataSet như là một vùng nhớ đệm.
- Một đối tượng DataAdapter làm nhiệm vụ trung gian giữa DataSet và data source để nạp dữ liệu vào vùng nhớ đệm.
- Sau khi DataAdapter hoàn thành nhiệm vụ nạp dữ liệu nó ngắt kết nối khỏi nguồn dữ liệu.



## 5.4.1. Lớp DataSet

- Là container dùng để lưu trữ đối tượng liên quan đến dữ liệu như DataTable, DataRelation, DataView. Thành phần này còn được gọi là lớp không kết nối (disconnected layer).
- DataSet như là một CSDL thu nhỏ tại máy client, có thể chứa các đối tượng table, view, constraint, relationship giữa các table, ...
- Tất cả dữ liệu từ nguồn dữ liệu thực sẽ được nạp vào DataSet dưới dạng các DataTable

# DataSet



## 5.4.1.1. DataTable

- Thuộc tính `DataSet.Tables` chứa các đối tượng `DataTable`.
- Mỗi đối tượng trong tập hợp này có thể được truy xuất bằng chỉ số hoặc bằng tên
- Các thuộc tính quan trọng nhất của lớp `DataTable` là `Columns` và `Rows` định nghĩa cấu trúc và nội dung bảng dữ liệu

## 5.4.1.2. DataColumn

- Thuộc tính `DataTable.Columns` chứa một tập các đối tượng `DataColumn` biểu diễn các trường dữ liệu trong `DataTable`.

## 5.4.1.2. DataColumn

Thuộc tính	Mô tả
ColumnName	Tên column
DataType	Kiểu của dữ liệu chứa trong column này Ví dụ: <code>col1.DataType = System.Type.GetType("System.String")</code>
MaxLength	Độ dài tối đa của một text column. -1 nếu không xác định độ dài tối đa
ReadOnly	Cho biết giá trị của column có được chỉnh sửa hay không
AllowDBNull	Giá trị Boolean cho biết column này có được chứa giá trị NULL hay không
Unique	Giá trị Boolean cho biết column này có được chứa các giá trị trùng nhau hay không
Expression	Biểu thức định nghĩa cách tính giá trị của một column Ví dụ: <code>colTax.Expression = "colSales * .085";</code>
Caption	Tiêu đề hiển thị trong thành phần điều khiển giao diện đồ họa
DataTable	Tên của đối tượng DataTable chứa column này

## 5.4.1.2. DataColumn

```
DataTable tb = new DataTable("DonHang");
DataColumn dCol = new DataColumn("MaSo", Type.GetType("System.Int16"));
dCol.Unique = true; // Dữ liệu của các dòng ở column này không được trùng nhau
dCol.AllowDBNull = false;
tb.Columns.Add(dCol);
dCol = new DataColumn("DonGia", Type.GetType("System.Decimal"));
tb.Columns.Add(dCol);
dCol = new DataColumn("SoLuong", Type.GetType("System.Int16"));
tb.Columns.Add(dCol);
dCol = new DataColumn("ThanhTien", Type.GetType("System.Decimal"));
dCol.Expression = "SoLuong*DonGia";
tb.Columns.Add(dCol);
// Liệt kê danh sách các Column trong DataTable
foreach (DataColumn dc in tb.Columns)
{
    Console.WriteLine(dc.ColumnName);
    Console.WriteLine(dc.DataType.ToString());
}
```



### 5.4.1.3. DataRow

- Dữ liệu được đưa vào table bằng cách tạo mới một đối tượng DataRow, gán giá trị cho các column của nó, sau đó bổ sung đối tượng DataRow này vào tập hợp Rows gồm các DataRow của table.

## 5.4.1.3. DataRows

```
DataRow row;  
row = tb.NewRow(); // Tạo mới DataRow  
row["DonGia"] = 22.95;  
row["SoLuong"] = 2;  
row["MaSo"] = 12001;  
tb.Rows.Add(row); // Bổ sung row vào tập Rows  
Console.WriteLine(tb.Rows[0]["ThanhTien"].ToString()); // 45.90
```

```
tb.Rows.Add(row); // Added  
tb.AcceptChanges(); // ...Commit changes  
Console.WriteLine(row.RowState); // Unchanged  
tb.Rows[0].Delete(); // Deleted  
// Undo deletion  
tb.RejectChanges(); // ...Roll back
```

## 5.4.1.4. DataView

- Đóng vai trò như tầng hiển thị dữ liệu lưu trữ trong DataTable. Nó cho phép người sử dụng sắp xếp, lọc và tìm kiếm dữ liệu.

## 5.4.1.4. DataView

//Giả sử đã có 1 dataset có tên là ds chứa dữ liệu của bảng DonHang

```
DataView dv = new DataView(ds.Tables["DonHang"];
```

// Lọc ra tất cả các hàng có giá từ 10 đến 100

```
dv.RowFilter = "soluong>=10 and soluong<=100";
```

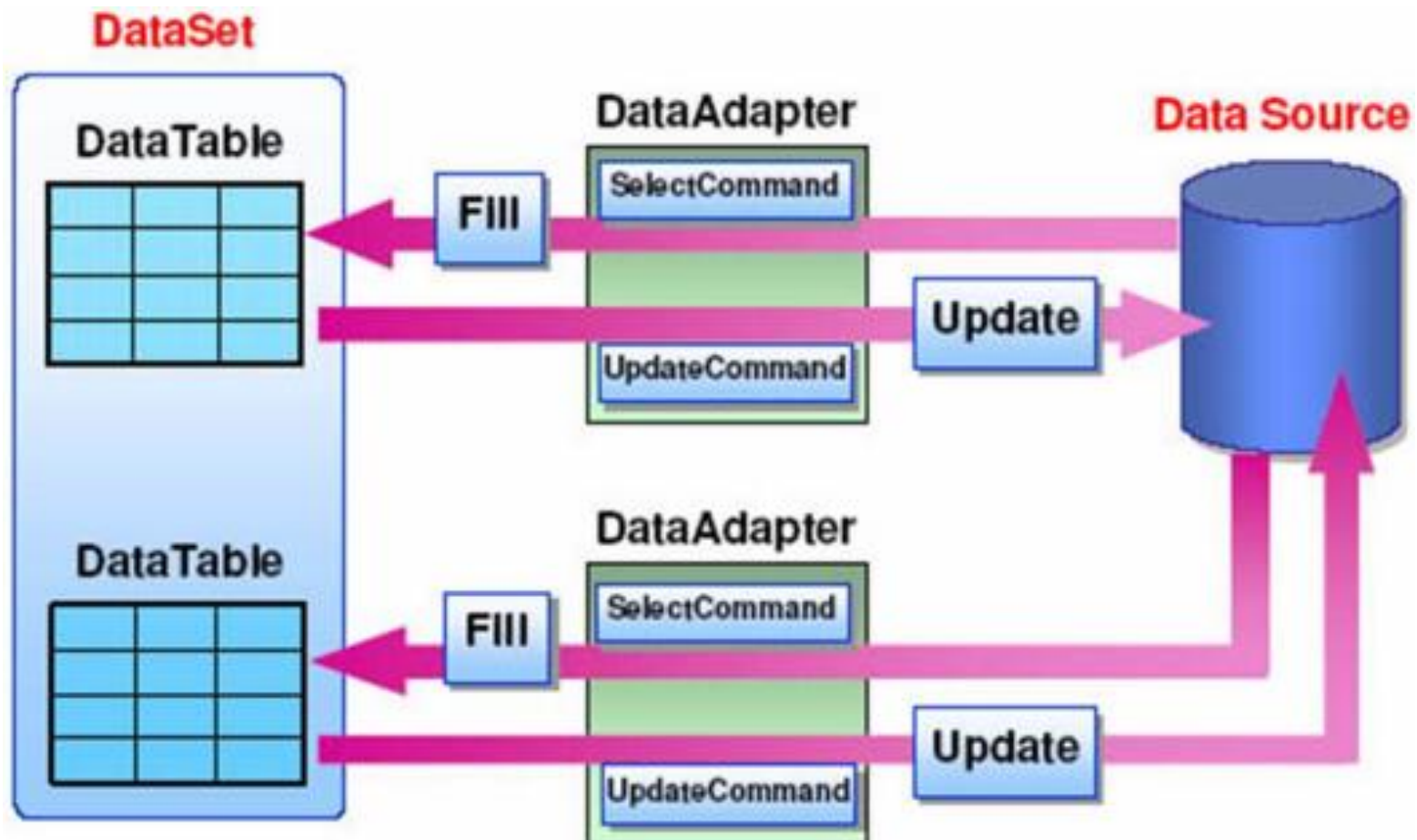
//Sắp xếp tăng dần theo số lượng nếu số lượng bằng nhau thì sắp xếp giảm dần thêm đơn giá

```
dv.Sort = "soluong, dongia DESC";
```

## 5.4.2 Nạp dữ liệu vào DataSet bằng DataAdapter

- Đối tượng DataAdapter là cầu nối giữa CSDL và DataSet
- Thường sử dụng DataAdapter để truy cập vào các loại CSDL khác nhau như SQL Server, Oracle...
- Toàn bộ dữ liệu sẽ được đổ vào DataSet được lưu trên bộ nhớ, sau đó sẽ sử dụng dữ liệu trên DataSet.

## 5.4.2 Nạp dữ liệu vào DataSet bằng DataAdapter



## 5.4.2. Nạp dữ liệu vào DataSet bằng DataAdapter

- Thuộc tính và phương thức của DataAdapter:
  - Fill (DataSet): Sử dụng SelectCommand để lấy dữ liệu từ CSDL và đổ vào DataSet;
  - Update(DataSet): InsertCommand, UpdateCommand, DeleteCommand, cập nhật dữ liệu trong DataSet vào cơ sở dữ liệu qua DataAdapter.
- Chú ý: Các thao tác với DataSet là thao tác với dữ liệu trên bộ nhớ máy tính, chưa tác động đến dữ liệu trong CSDL, để thực hiện các thao tác cập nhật, xóa,... dữ liệu trên CSDL ta phải sử dụng qua đối tượng DataAdapter



## 5.4.2. Nạp dữ liệu vào DataSet bằng DataAdapter

(1) Tạo từ Connection string và câu truy vấn SELECT:

```
String sql = "SELECT * FROM nhanvien";
```

```
qlDataAdapter da = new SqlDataAdapter(sql, connStr);
```

(2) Tạo từ đối tượng Connection và câu truy vấn SELECT:

```
qlConnection conn = new SqlConnection(connStr);
```

```
qlDataAdapter da = new SqlDataAdapter(sql, conn);
```

(3) Gán đối tượng Command cho thuộc tính

```
SelectCommand SqlDataAdapter da = new SqlDataAdapter();
```

```
SqlConnection conn = new SqlConnection(connStr);
```

```
da.SelectCommand = new SqlCommand(sql, conn);
```



## 5.4.2 Nạp dữ liệu vào DataSet bằng DataAdapter

```
string chuoi = "server =.; database = QLDHang; integrated security = true;";  
SqlConnection conn = new SqlConnection(chuoi);  
conn.Open();  
SqlDataAdapter da = new SqlDataAdapter("select * from KhachHang", conn);  
DataSet ds = new DataSet();  
da.Fill(ds);  
dataGridView1.DataSource = ds.Tables[0];
```

## 5.4.2 Nạp dữ liệu vào DataSet bằng DataAdapter

```
String sql = "SELECT * FROM nhanvien";  
SqlDataAdapter da = new SqlDataAdapter(sql, connStr);  
  
DataSet ds = new DataSet();  
// Tạo ra một DataTable, nạp dữ liệu vào DataTable, và đưa DataTable vào DataSet  
  
int nRecs = da.Fill(ds); // trả về số lượng record được nạp vào DataTable  
  
// Nếu muốn đặt tên cho DataTable trong DataSet thay vì lấy tên mặc định  
// thì sử dụng code như thế này  
  
int nRecs = da.Fill(ds, "nhanvien ")
```

## 5.4.3. Bài thực hành



## 5.5 Sử dụng Data Binding

- Khái niệm Data Binding
  - Simple Data Binding
  - Complex Data Binding
  - Data Binding đến Property
  - Data Binding đến DataGridView
  - Data Binding đến ComboBox
  - Data Binding đến ListBox
- Đồng bộ dữ liệu giữa các control và datasource
- Master – Detail
- Binding Source
- Binding Navigator

## 5.5.1. Khái niệm Data Binding

- Data binding là một giải pháp cho vấn đề liên kết giữa tập dữ liệu với các controls
- Phân loại: Có 2 loại Data Binding
  - Simple Data Binding
  - Complex Data Binding
- Data Binding giúp hiển thị dữ liệu trong Data Source lên control

## 5.5.1.1. Simple Data Binding

- Kết nối 1 property của control (loại property chỉ lưu 1 giá trị tại 1 thời điểm) với 1 cột (hay property) của data source
- Ví dụ:
  - `string tenControl.Text`
  - `object tenControl.Tag`
  - `Image picBox.Image`
  - `bool checkBox.Checked`
  - `bool radioButton.Checked`
  - `int trackBar.Value`
  - ...

## 5.5.1.1. Simple Data Binding

- Cách 1:

```
DataBinding binding;  
binding = new DataBinding("propertyName", dataSource, "dataMember", true);  
tenControl.DataBindings.Add(binding);
```

- Cách 2:

```
tenControl.DataBindings.Add("propertyName", dataSource, "dataMember", true);
```

## 5.5.1.2. Complex Data Binding

- Kết nối 1 control có khả năng hiển thị nhiều giá trị tại 1 thời điểm với 1 cột hay tất cả các cột trong data source
- Ví dụ:
  - DataGridView
  - ComboBox
  - ListBox
  - ...



## 5.5.1.2. Complex Data Binding

- Cách 1:

```
tenControl.DataSource = tenBang;
```

- Cách 2:

```
tenControl.DataSource = tenDataSet;  
tenControl.DataMember = tenBang;
```

## 5.5.1.3. Data Binding đến Property

- Data Binding đến thuộc tính Text

```
tenTextBox.DataBindings.Add("Text", tenBang, "tenCot");  
tenLabel.DataBindings.Add("Text", tenBang, "tenCot");  
tenButton.DataBindings.Add("Text", tenBang, "tenCot");  
tenCheckBox.DataBindings.Add("Text", tenBang, "tenCot");  
...
```

```
tenTextBox.DataBindings.Add("Text", tenDS, "tenBang.tenCot");  
tenLabel.DataBindings.Add("Text", tenDS, "tenBang.tenCot");  
tenButton.DataBindings.Add("Text", tenDS, "tenBang.tenCot");  
tenCheckBox.DataBindings.Add("Text", tenDS, "tenBang.tenCot");  
.....
```

### 5.5.1.3. Data Binding đến Property

- Data Binding đến thuộc tính Tag

```
tenControl.DataBindings.Add("Tag", tenBang, "tenCot");
```

```
tenControl.DataBindings.Add("Tag", tenDS, "tenBang.tenCot");
```

## 5.5.1.3. Data Binding đến Property

- Data Binding đến thuộc tính Checked

```
tenCheckBox.DataBindings.Add("Checked", tenBang, "tenCot");  
tenRadio.DataBindings.Add("Checked", tenBang, "tenCot");  
...
```

```
tenCheckBox.DataBindings.Add("Checked", tenDS, "tenBang.tenCot");  
tenRadio.DataBindings.Add("Checked", tenDS, "tenBang.tenCot");  
...
```

## 5.5.1.3. Data Binding đến Property

- Data Binding đến thuộc tính Value

```
tenDateTimePicker.DataBindings.Add("Value", tenBang, "tenCot");  
tenProcessBar.DataBindings.Add("Value", tenBang, "tenCot");  
tenTrackBar.DataBindings.Add("Value", tenBang, "tenCot");  
tenNumericUpDown.DataBindings.Add("Value", tenBang, "tenCot");  
tenVScrollBar.DataBindings.Add("Value", tenBang, "tenCot");  
tenHScrollBar.DataBindings.Add("Value", tenBang, "tenCot");  
...
```

## 5.5.1.3. Data Binding đến Property

- Data Binding đến thuộc tính Value

```
tenDateTimePicker.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
tenProcessBar.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
tenTrackBar.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
tenNumericUpDown.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
tenVScrollBar.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
tenHScrollBar.DataBindings.Add("Value", tenDS, "tenBang.tenCot");  
...
```

## 5.5.1.4. Data Binding đến DataGridView

- Cách 1:

```
tenGrid.DataSource = tenBang;
```

- Cách 2:

```
tenGrid.DataSource = tenDataSet;  
tenGrid.DataMember = "TenBang";
```

## 5.5.1.5. Data Binding đến ComboBox

- Cách 1:

```
tenComboBox.DataSource = tenBang;  
tenComboBox.DisplayMember = "tenCot1";  
tenComboBox.ValueMember = "tenCot2";
```

- Cách 2:

```
tenComboBox.DataSource = tenDataSet;  
tenComboBox.DisplayMember = "tenBang.tenCot1";  
tenComboBox.ValueMember = "tenBang..tenCot2";
```



## 5.5.1.6. Data Binding đến ListBox

- Cách 1:

```
tenListBox.DataSource = tenBang;  
tenListBox.DisplayMember = "tenCot1";  
tenListBox.ValueMember = "tenCot2";
```

- Cách 2:

```
tenListBox.DataSource = tenDataSet;  
tenListBox.DisplayMember = "tenBang.tenCot1";  
tenListBox.ValueMember = "tenBang.tenCot2";
```

## 5.5.2. Đồng bộ dữ liệu giữa các control và datasource

- Khái niệm
- Cơ chế đồng bộ của .NET 1.x
- Các thao tác của đối tượng CurrencyManager

## 5.5.2.1. Khái niệm

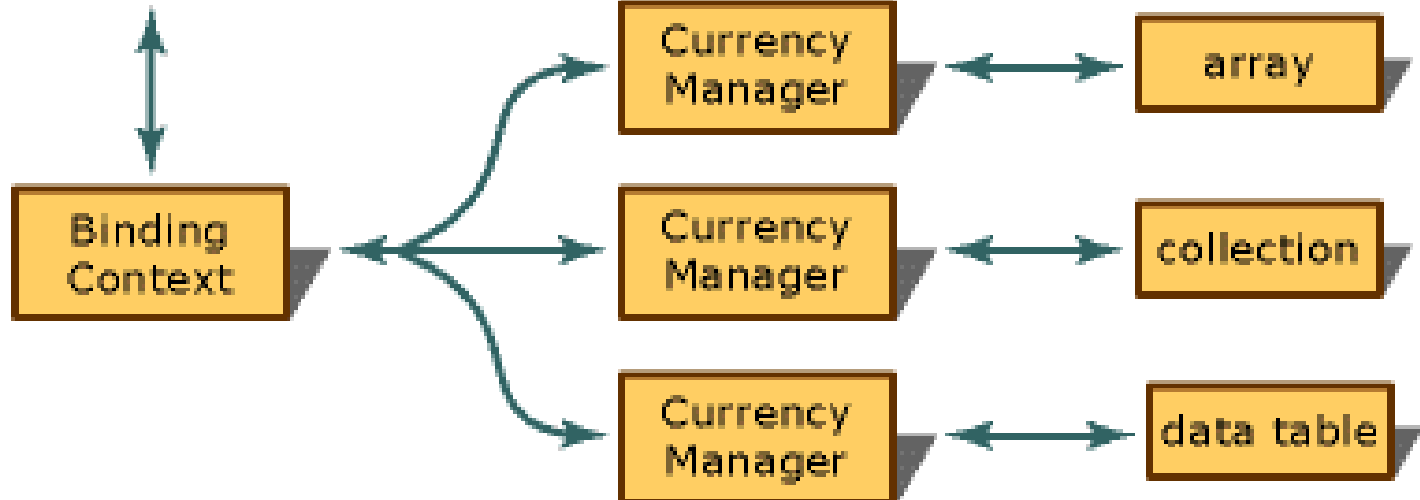
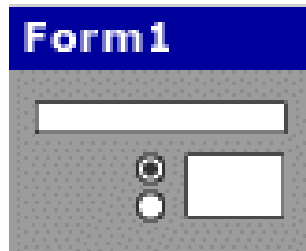
- Đồng bộ dữ liệu là khả năng cập nhật tức thì dữ liệu giữa các control hay giữa các control và Data source
- Ví dụ:
  - Khi nhấn button “Next”, chúng ta thấy các control đều cập nhật dữ liệu của record kế tiếp
  - Khi chọn record bằng listbox, chúng ta thấy các thông tin liên quan tự động xuất hiện trên các control khác của cùng form

## 5.5.2.2. Cơ chế đồng bộ của .NET 1.x

- Khả năng đồng bộ trong .NET 1.x được hiện thực trong Windows Form thông qua property BindingContext và lớp CurrencyManager
- Khi chúng ta kết nối các control với một data source, chương trình tự động tạo ra một đối tượng thuộc lớp CurrencyManager để theo dõi vị trí của record trong datasource
- Mỗi data source tương ứng có 1 đối tượng CurrencyManager
- Windows Form lưu tất cả đối tượng CurrencyManager vào trong mảngBindingContext

## 5.5.2.2. Cơ chế đồng bộ của .NET 1.x

**Windows Form**



## 5.5.2.3. Các thao tác của đối tượng CurrencyManager

- Cách 1:

```
CurrencyManager cm;  
cm=(CurrencyManager)this.BindingContext[tenDataSet,"Bảng"];
```

- Cách 2:

```
CurrencyManager cm;  
cm = (CurrencyManager)this.BindingContext[tenBang];
```

## 5.5.2.3. Các thao tác của đối tượng CurrencyManager

- Lớp CurrencyManager
  - Property:
    - int Position
    - int Count
  - Method:
    - void AddNew()
    - void EndCurrentEdit()
    - void CancelCurrentEdit()
    - void RemoveAt(int index)

## 5.5.2.3. Các thao tác của đối tượng CurrencyManager

- Next
  - `cm.Position++;`
- Previous
  - `cm.Position--;`
- First
  - `cm.Position = 0;`
- Last
  - `vt = cm.Count - 1;`
  - `cm.Position = vt;`



## 5.5.2.3. Các thao tác của đối tượng CurrencyManager

- Những thao tác cơ bản

```
CurrencyManager cm;  
cm = (CurrencyManager) this.BindingContext[dataTable];  
if (cm.Position < cm.Count - 1)  
{  
    cm.Position++;  
}
```

```
CurrencyManager cm;  
cm = (CurrencyManager) this.BindingContext[dataTable];  
if (cm.Position > 0)  
{  
    cm.Position--;  
}
```

## 5.5.2.3. Các thao tác của đối tượng CurrencyManager

- Insert
  - `void cm.AddNew();`
- Update
  - `cm.EndCurrentEdit();`
  - `da.Update(dataTable);`
- Delete
  - `index = cm.Position`
  - `cm.RemoveAt(index);`
- Cancel
  - `cm.CancelCurrentEdit();`

## 5.5.3. Master – Detail

- Master – detail: dạng quan hệ 1-n trong cơ sở dữ liệu

De tai - sinh vien

Đề tài

	madt	tendt
▶	dt01	quản lý bán hàng...
	dt02	nhận dạng chữ vi...
*		

Sinh viên

	masv	hoten	namsinh
▶	sv01	trần văn tuần	1985
	sv02	lê thanh thanh	1986
	sv04	lê minh hằng	1986
*			

De tai - sinh vien

Đề tài

	madt	tendt
	dt01	quản lý bán hàng...
▶	dt02	nhận dạng chữ vi...
*		

Sinh viên

	masv	hoten	namsinh
▶	sv03	nguyễn xuân minh	1988
*			

## 5.5.3. Master – Detail

- Bước 1: fill dữ liệu
  - Fill 2 bảng vào dataset (bao gồm khóa chính và khóa ngoại)
- Bước 2: tạo quan hệ
  - DataColumn colMaster =  
ds.Tables["tenMaster"].Columns["cột PK"];
  - DataColumn colDetail =  
ds.Tables["tenDetail"].Columns["cột FK"];
  - DataRelation relation = new DataRelation("tenquanhe",  
colMaster, colDetail);
  - ds.Relations.Add(relation);

## 5.5.3. Master – Detail

- Bước 3: (cách 1)
  - `dataGridViewMaster.DataSource = ds;`
  - `dataGridViewMaster.DataMember = "tenMaster";`
  - `dataGridViewDetail.DataSource = ds;`
  - `dataGridViewDetail.DataMember = "tenMaster.tenquanhe";`

## 5.5.3. Master – Detail

- Bước 3: (cách 2)
  - `dataGridViewMaster.DataSource = ds.Tables["tenMaster"];`
  - `dataGridViewDetail.DataSource = ds.Tables["tenMaster"];`
  - `dataGridViewDetail.DataMember = "tenquanhe";`

## 5.5.4. Binding Source

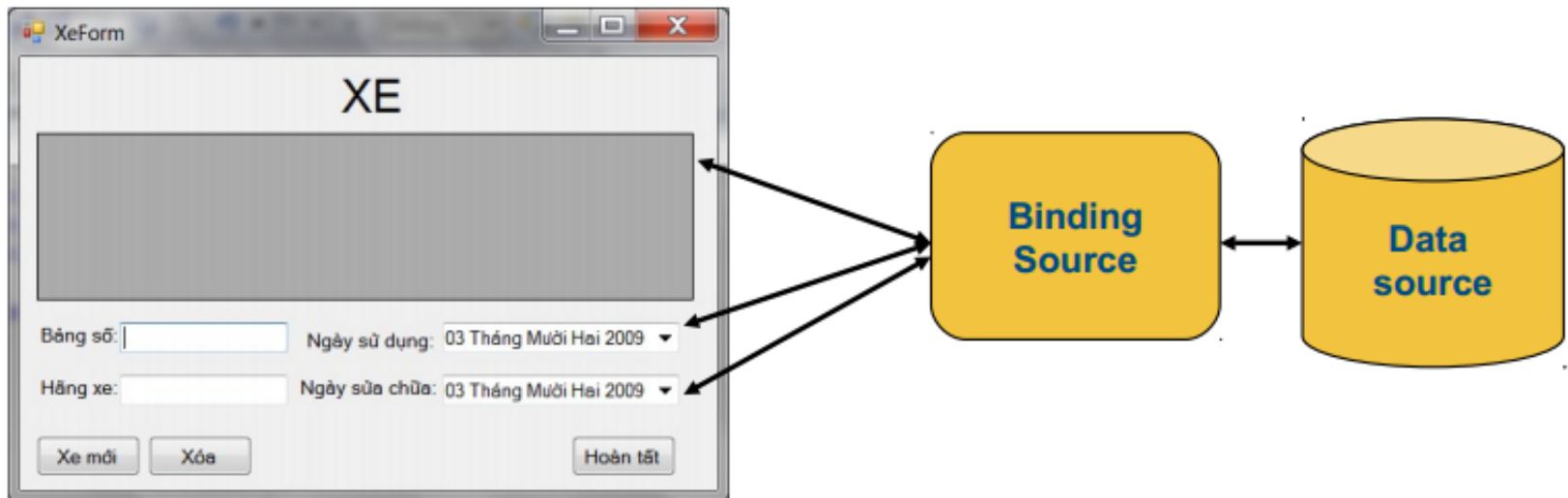
- Khái niệm
- Tạo đối tượng Binding Source
- Kết nối control với Binding Source
- Các thao tác của đối tượng BindingSource

## 5.5.4.1. Khái niệm

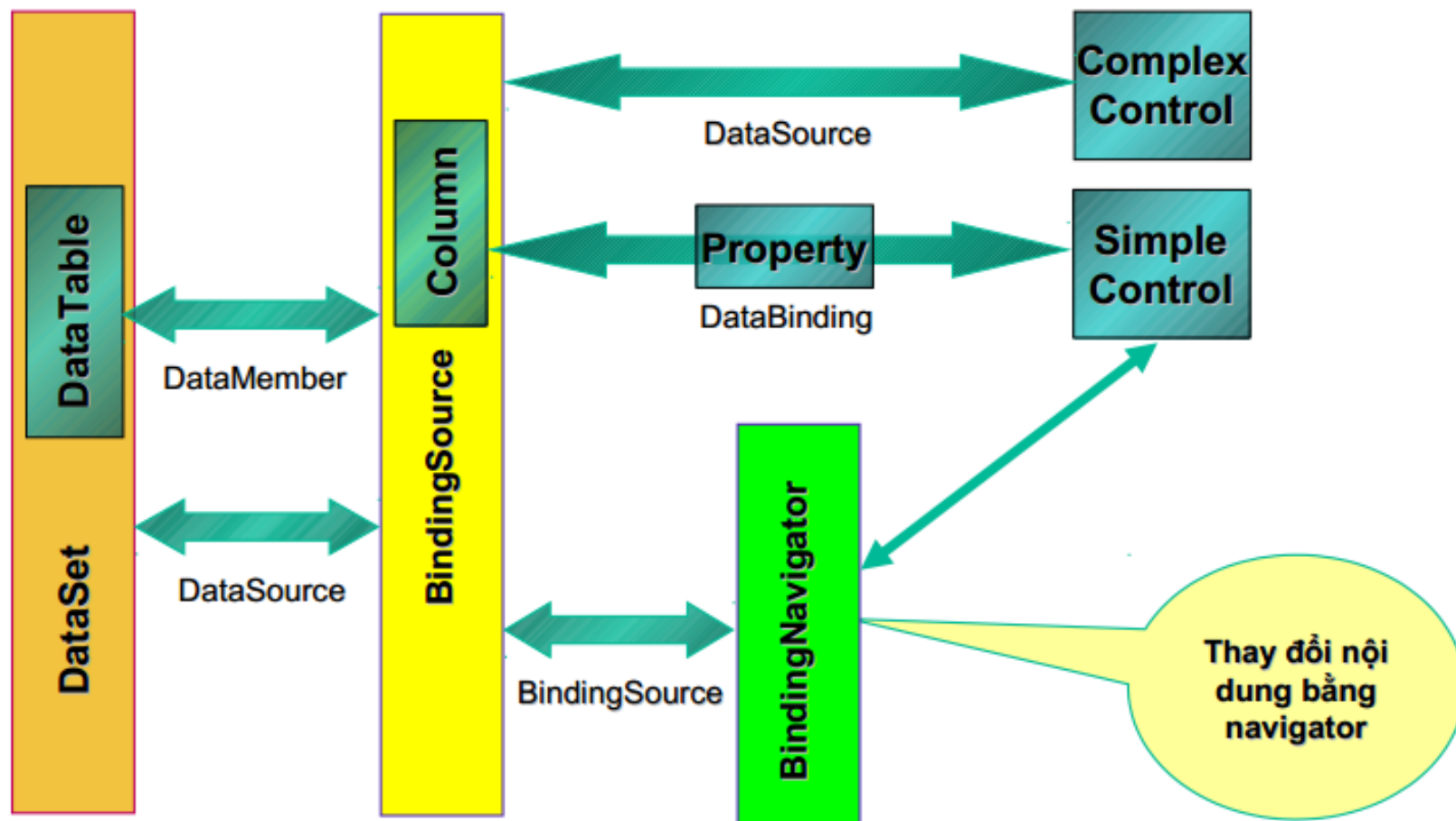
- Trong .NET 1.X, chỉ cho phép kết nối trực tiếp giữa Control đến Data Source
- Trong .NET 2.0, cho phép kết nối giữa Control và Data Source thông qua đối tượng trung gian Binding Source
- Ưu điểm khi sử dụng Binding Source
  - Dễ dàng thay đổi Data Source của các Control
  - Cung cấp nhiều sự kiện và các phương thức



## 5.5.4.1. Khái niệm



## 5.5.4.1. Khái niệm



## 5.5.4.2. Tạo đối tượng Binding Source

- Cách 1:

```
BindingSource bs;  
bs = new BindingSource();  
...  
bs.DataSource = dataSource;  
bs.DataMember = "dataMember";
```

- Cách 2:

```
BindingSource bs;  
bs = new BindingSource(dataSource, "dataMember");
```

## 5.5.4.3. Kết nối control với Binding Source

- Simple Data Binding:

```
tenControl.DataBindings.Add("propertyName",  
bs, "dataMember", true);
```

- Complex Data Binding:

```
tenControl.DataSource = bs;  
//tenControl.DataMember = "tenBang";
```

## 5.5.4.3. Các thao tác của đối tượng BindingSource

- Next
  - `bs.MoveNext();`
- Previous
  - `bs.MovePrevious();`
- First
  - `bs.MoveFirst();`
- Last
  - `bs.MoveLast();`

## 5.5.4.3. Các thao tác của đối tượng BindingSource

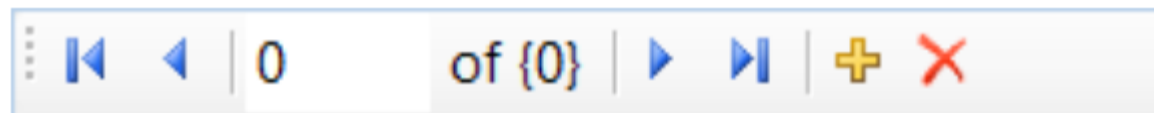
- Insert
  - object bs.AddNew();
- Update
  - bs.EndEdit();
  - da.Update(dataTable);
- Delete
  - void bs.RemoveAt(index);
  - void bs.RemoveCurrent();
- Cancel
  - bs.CancelEdit();

## 5.5.4.3. Các thao tác của đối tượng BindingSource

- Property
  - `int bs.Position`
  - `int bs.Count`
  - `CurrencyManager bs.CurrencyManager`
  - `string bs.Filter`
  - `string bs.Sort`
- Method
  - `void bs.ResetBinding(bool metaDataChanged);`
  - `void bs.ResetCurrentItem();`
  - `void bs.ResetItem(int index);`

## 5.5.5. Binding Navigator

- Binding Navigator là 1 phiên bản mới của ToolStrip trong .NET 2.0 bao bọc các chức năng: MoveNext, MovePrevious, MoveFirst, MoveLast, AddNew, RemoveCurrent, ...





## 5.5.5. Binding Navigator

- Các bước tạo Binding Navigator
  - Bước 1: Kéo Binding Navigator vào Form
  - Bước 2: Thiết lập BindddingSource

```
tenNavigator. BindingSource=tenBindingSource;
```