



LẬP TRÌNH CƠ SỞ DỮ LIỆU

Chương 1: Tổng quan về ADO.NET



1.1. Giới thiệu ADO.NET



1.2. Kiến trúc ADO.NET



1.3. Kết nối MS SQL Server bằng SQL Server Data Provider



1.4. Kết nối Oracle bằng Oracle Data Provider

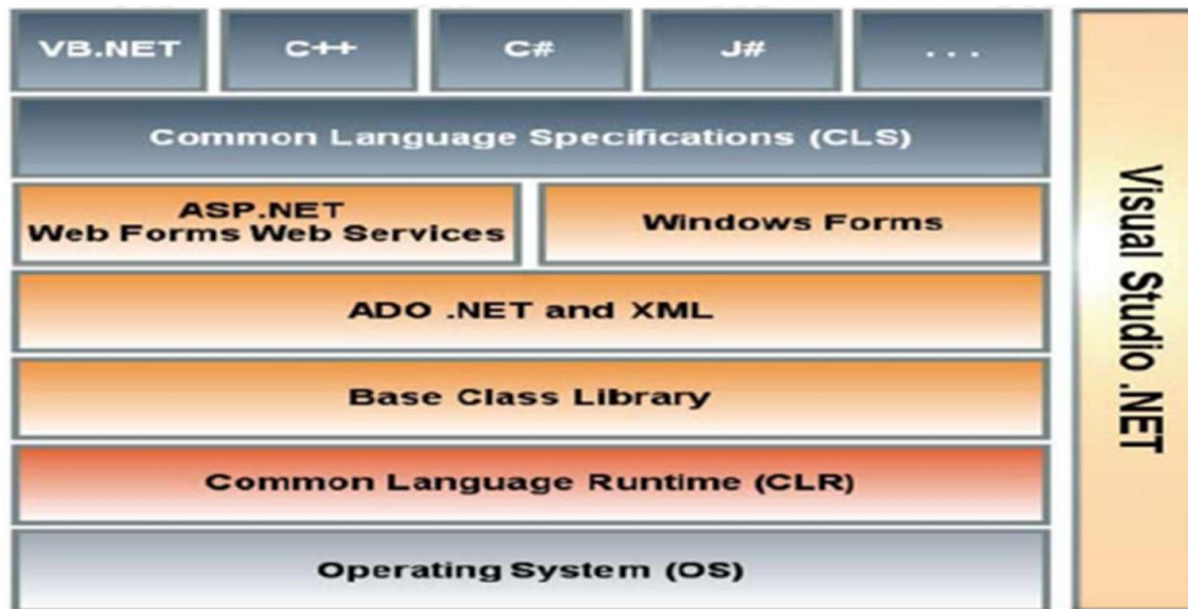


1.5. Kết nối MS Access bằng Access Data Provider

1.1 Giới thiệu ADO.NET



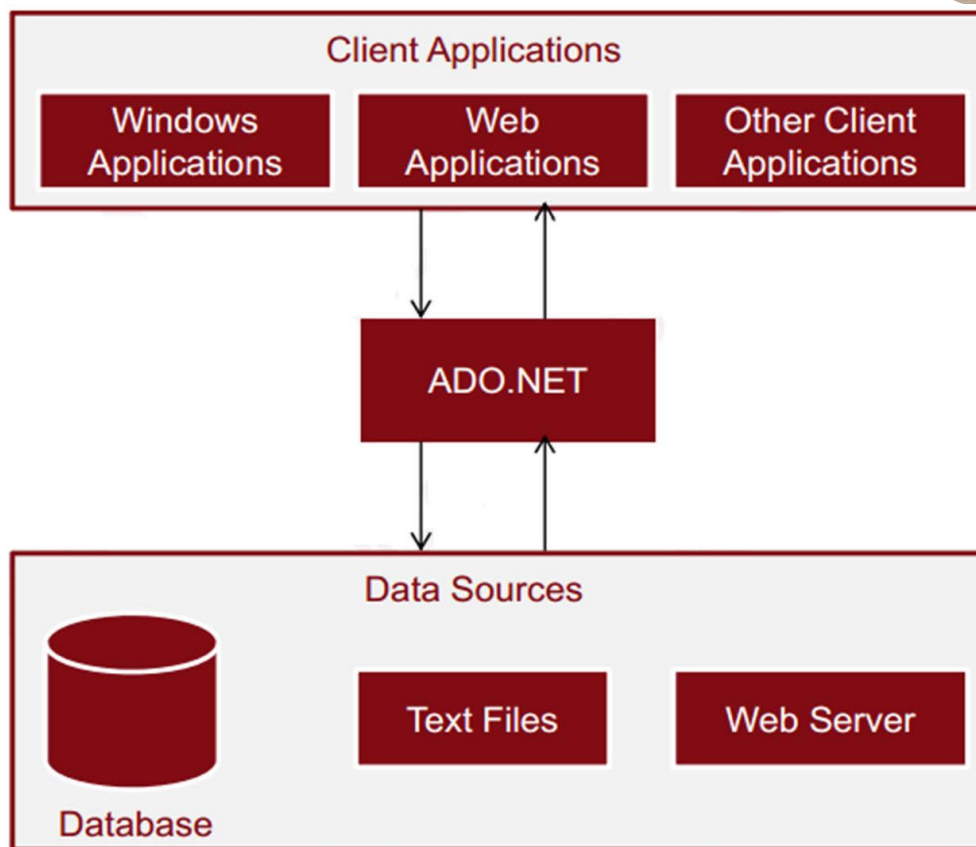
- ❖ ADO.NET (ActiveX Data Objects.NET) là một thành phần trong .NET FRAMEWORK đảm nhận vai trò thao tác với CSDL.
- ❖ Cung cấp **các lớp đối tượng** và **hàm thư viện** phục vụ kết nối và xử lý dữ liệu.



1.1 Giới thiệu ADO.NET



- ❖ ADO.NET là cầu nối giữa ứng dụng và CSDL.
- ❖ ADO.NET hỗ trợ việc kết nối và truy cập CSDL đối với nhiều hệ quản trị CSDL khác nhau như: SQL Server, Oracle, Access, ...

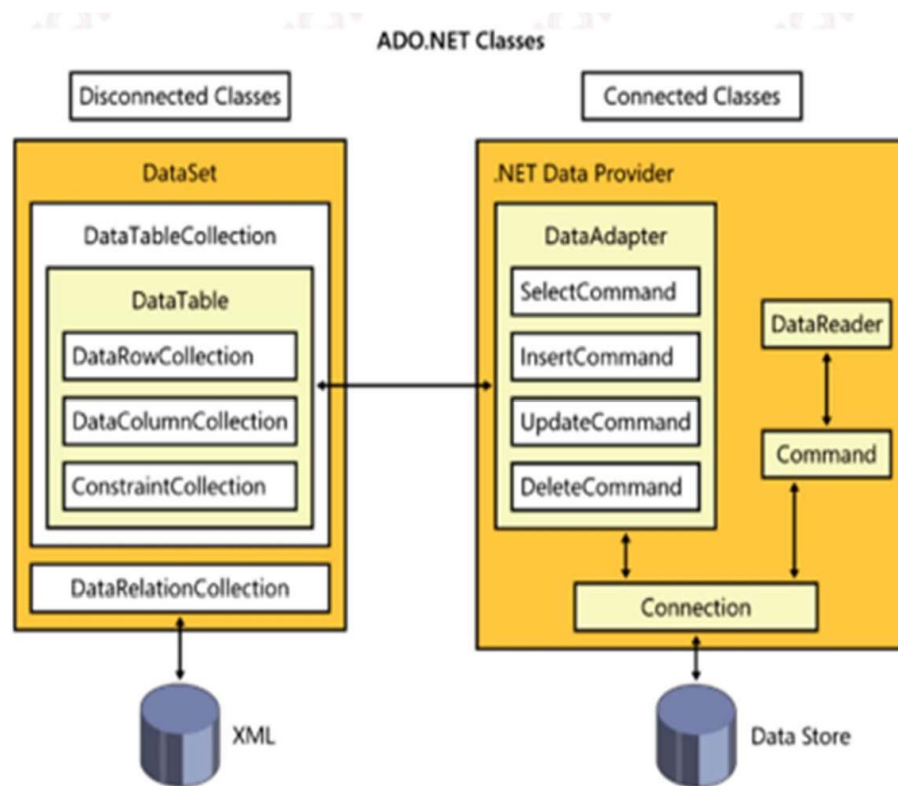


1.1 Giới thiệu ADO.NET



- ❖ ADO.NET hoạt động theo 2 kiến trúc (kết nối và ngắt kết nối).

- **Mô hình ngắt kết nối:** Cho phép lấy cả một cấu trúc dữ liệu phức tạp từ CSDL sau đó ngắt kết nối với CSDL và thực hiện xử lý dữ liệu.
- **Mô hình kết nối:** Luôn phải duy trì kết nối trong suốt quá trình xử lý dữ liệu.



1.1 Giới thiệu ADO.NET



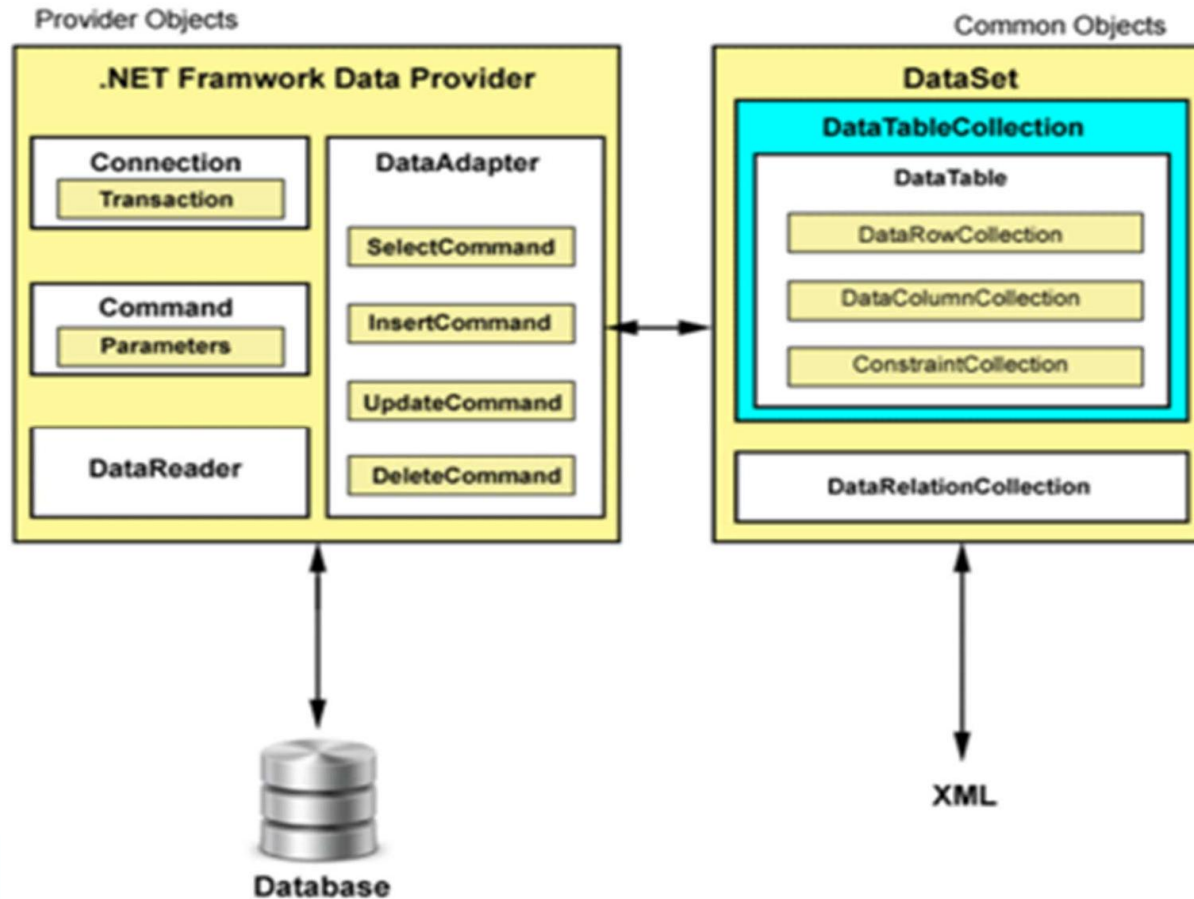
❖ Các đặc trưng của ADO.NET.

- Hỗ trợ lập trình:
 - Cung cấp các lớp thao tác với CSDL giúp lập trình nhanh hơn và giảm lỗi;
 - Cung cấp các công cụ để thao tác với CSDL ngay trên phần Disigner giúp thao tác với CSDL dễ dàng hơn.
- Khả năng mở rộng:
 - Sử dụng kiến trúc ngắt kết nối giúp giảm tải cho server, hỗ trợ nhiều người sử dụng truy cập CSDL đồng thời tốt hơn.
- Khả năng tích hợp:
 - ADO.NET có thể gửi dữ liệu cho bất cứ loại ứng dụng nào;
 - Hỗ trợ XML.

1.2 Kiến Trúc ADO.NET



- ❖ Kiến trúc ADO.NET gồm 2 thành phần chính:
ADO.NET Architecture



1.2 Kiến Trúc ADO.NET



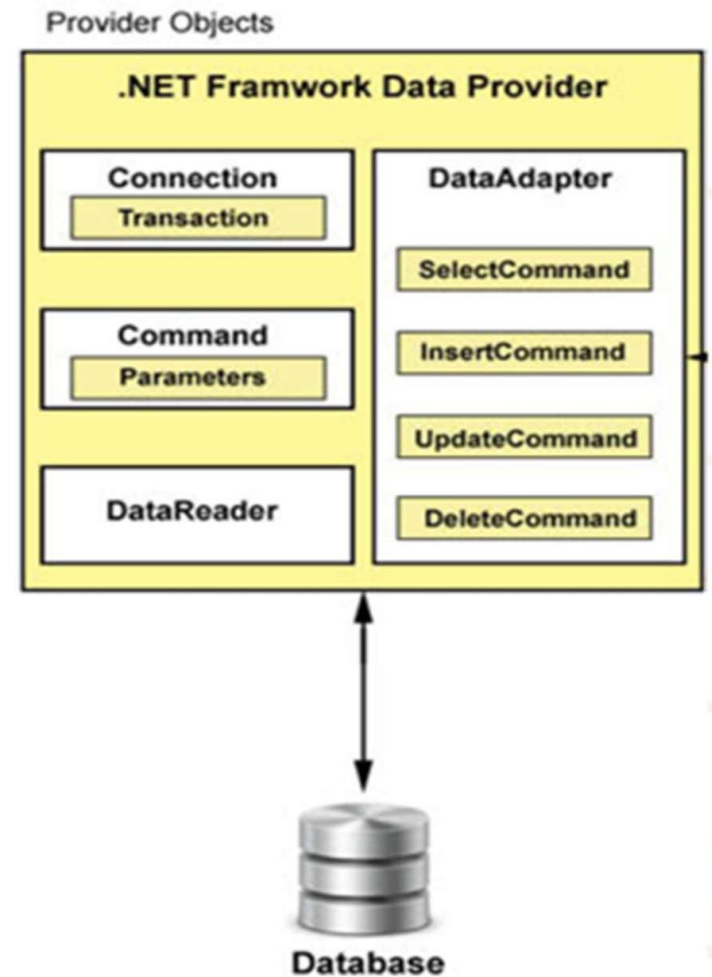
- ❖ **.NET Framework Data Provider:** cung cấp các thư viện để làm việc với các hệ quản trị cơ sở dữ liệu.
 - **System.Data.OleDb:** Làm việc với CSDL Access.
 - **System.Data.SqlClient:** Làm việc với CSDL SQL Server.
 - **System.Data.OracleClient:** Làm việc với CSDL Oracle.

1.2 Kiến Trúc ADO.NET



❖ **Mô hình kết nối:** Sử dụng khi kết nối CSDL và thực hiện các thao tác xử lý dữ liệu. Bao gồm các thành phần chính sau:

- **Connection:** quản lý việc mở, đóng CSDL;
- **Command:** lệnh truy vấn, tương tác dữ liệu khi đang lập kết nối;
- **DataReader:** đọc dữ liệu, chỉ xử lý một dòng dữ liệu tại một thời điểm;
- **DataAdapter:** Cầu nối giữa CSDL với DataSet.

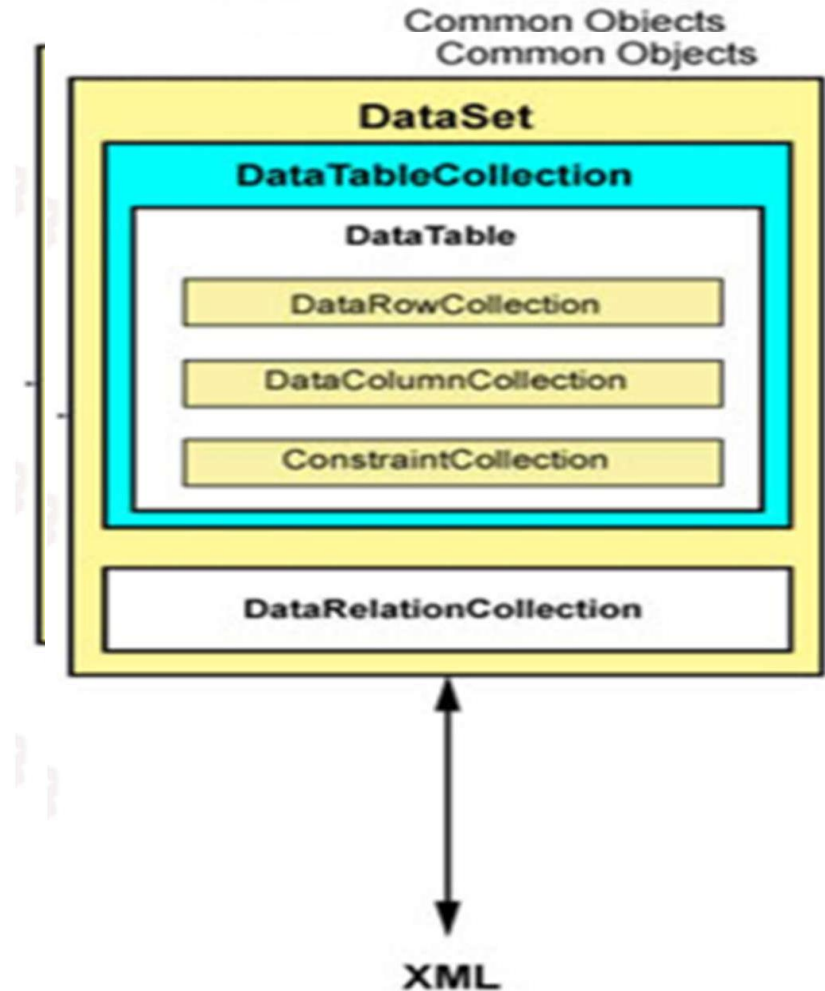


1.2 Kiến Trúc ADO.NET



❖ Mô hình ngắt kết nối DataSet:

- DataSet không quan tâm đến kiểu của CSDL;
- Lấy dữ liệu từ DataAdapter;
- DataSet xem như một CSDL trong bộ nhớ;
- DataSet gồm các thành phần con như: DataTable, DataRow, DataColumn, DataRelation, ...



1.3 Kết nối MS SQL Server bằng SQL Server Data Provider



❖ Các bước tạo kết nối tới CSDL SQL Server.

- Bước 1: Khai báo đối tượng SqlConnection
`SqlConnection conn = null;`
- Bước 2: Tạo chuỗi kết nối tới CSDL SQL Server
`string connectionString = "Data Source=Server; Initial Catalog=QuanLyBanHang; UID=sa; PWD=123";`
- Bước 3: Tạo đối tượng Connection
`conn = new SqlConnection(connectionString);`
- Bước 4: Mở kết nối qua đối tượng Connection
`conn.Open();`

1.3 Kết nối MS SQL Server bằng SQL Server Data Provider



❖ Demo

1.4 Kết nối Oracle bằng Oracle Data Provider



❖ Các bước tạo kết nối tới CSDL Oracle.

- Bước 1: Khai báo đối tượng OracleConnection

OracleConnection conn = null;

- Bước 2: Tạo chuỗi kết nối tới CSDL Access

```
string connectionString = "Data Source=Server; Initial  
Catalog=QuanLyBanHang; UID=sa; PassWord="";
```

- Bước 3: Tạo đối tượng Connection

```
conn = new OracleConnection(connectionString);
```

- Bước 4: Mở kết nối qua đối tượng Connection

```
conn.Open();
```

1.4 Kết nối Oracle bằng Oracle Data Provider



❖ Demo

1.5 Kết nối MS Access bằng OLE DB Data Provider



❖ Các bước tạo kết nối tới CSDL Access.

- Bước 1: Khai báo đối tượng OleDbConnection
`OleDbConnection conn = null;`
- Bước 2: Tạo chuỗi kết nối tới CSDL Access
`string connectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\\Database\\QuanLyBanHang.accdb; Persist Security Info=False;";`
- Bước 3: Tạo đối tượng Connection
`conn = new OleDbConnection(connectionString);`
- Bước 4: Mở kết nối qua đối tượng Connection
`conn.Open();`

1.5 Kết nối MS Access bằng OLE DB Data Provider



❖ Demo

1.5 Kết nối MS Access bằng OLE DB Data Provider



❖ Bài tập thiết kế giao diện nhập danh sách sinh viên như hình

NHẬP DANH SÁCH SINH VIÊN

Lớp: 

Họ tên:

Giới tính: 

Ngày sinh: 

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------

Chương 2: Đối tượng Connection



Nội dung



2.1 Giới thiệu đối tượng Connection

2.2. Khởi tạo đối tượng

2.3. Chuỗi kết nối cơ sở dữ liệu

2.4. Đóng và mở kết nối

2.5. Connection Pooling

2.1 Giới thiệu đối tượng Connection



❖ **Đối tượng Connection:**

- **Chức năng:** Tạo kết nối với CSDL.
- **Thuộc tính:** `ConnectionString`: Thuộc tính chuỗi kết nối với CSDL (`DataSource`);
- **Phương thức:**
 - **Open():** Phương thức mở kết nối với CSDL;
 - **Close():** Phương thức đóng kết nối.



2.1 Giới thiệu đối tượng Connection



❖ Quy trình kết nối đến CSDL

- Tạo đối tượng Connection và xác định chuỗi kết nối (Connection String)
- Tạo đối tượng Command và xác định câu lệnh SQL
- Mở đối tượng Connection (Open)
- Thực hiện câu lệnh SQL và xử lý kết quả (Execute)
- Đóng đối tượng Connection (Close)

2.2 Khởi tạo đối tượng



1. Đối tượng **SqlConnection**

- *Khai báo Namespace*
 - *Using System.Data.**Sql**Client*
 - *Khai báo biến đối tượng kết nối*
 - **SqlConnection** Conn;
 - *Khởi tạo biến đối tượng kết nối*
 - Conn = new **SqlConnection**();
 - *Hoặc vừa khai báo và khởi tạo biến đối tượng kết nối*
 - **SqlConnection** Conn = new **SqlConnection**();
- Hoặc*
- **SqlConnection** Conn = new **SqlConnection**(ConnStr);
ConnStr là biến lưu trữ chuỗi kết nối

2.2 Khởi tạo đối tượng



2. Đối tượng **OracleConnection**

- *Khai báo Namespace*
 - *Using System.Data. **OracleClient***
 - *Khai báo biến đối tượng kết nối*
 - **OracleConnection** Conn;
 - *Khởi tạo biến đối tượng kết nối*
 - Conn = new **OracleConnection**();
 - *Hoặc vừa khai báo và khởi tạo biến đối tượng kết nối*
 - **OracleConnection** Conn = new **OracleConnection**();
- Hoặc*
- **OracleConnection** Conn = new **OracleConnection**(ConnStr); ConnStr là biến lưu trữ chuỗi kết nối

2.2 Khởi tạo đối tượng



3. Đối tượng OleDbConnection

- *Khai báo Namespace*
 - *Using System.Data.OleDbClient*
 - *Khai báo biến đối tượng kết nối*
 - `OleDbConnection Conn;`
 - *Khởi tạo biến đối tượng kết nối*
 - `Conn = new OleDbConnection();`
 - *Hoặc vừa khai báo và khởi tạo biến đối tượng kết nối*
 - `OleDbConnection Conn = new OleDbConnection();`
- Hoặc*
- `OleDbConnection Conn = new OleDbConnection(ConnStr);` ConnStr là biến lưu trữ chuỗi kết nối

2.3 Chuỗi kết nối cơ sở dữ liệu



❖ Chuỗi kết nối cơ sở dữ liệu

- Các giá trị trên chuỗi kết nối

Giá trị	Mô tả
Data source/Server	Tên Server CSDL
Initial catalog/DataBase	Tên của CSDL
UserID	Tên của user đăng nhập vào CSDL
PWD	Mật khẩu đăng nhập vào CSDL

2.3 Chuỗi kết nối cơ sở dữ liệu



1. Chuỗi kết nối cơ sở dữ liệu MS SQL Server

@ "Server=.\SQLEXPRESS;Database=QuanLyBanHang; Integrated Security=SSPI;"

- Hoặc @ "Server=.\SQLEXPRESS;Database=QuanLyBanHang; Trusted_Connection=True;"

- Hoặc

"Data Source=.\SQLEXPRESS; Initial Catalog=QuanLyBanHang; Integrated Security=True;"

2.3 Chuỗi kết nối cơ sở dữ liệu



2. Chuỗi kết nối cơ sở dữ liệu Oracle

@ "Server=.\ Oracle;Database=QuanLy
BanHang; Integrated Security=SSPI;"

- Hoặc @ "Server=.\Oracle;Database=
QuanLyBanHang; Trusted_Connection=True;"

- Hoặc

"Data Source=.\Oracle; Initial Catalog=
QuanLyBanHang; Integrated Security=True;"

2.3 Chuỗi kết nối cơ sở dữ liệu



3. Chuỗi kết nối cơ sở dữ liệu Access

- Chuỗi kết nối CSDL Access 97, 2000, 2002, 2003.

Provider=Microsoft.Jet.OLEDB.4.0;

Data Source=C:\\Database\\QuanLyBanHang.mdb;

- Chuỗi kết nối CSDL Access 2007, 2010, 2013.

Provider= Microsoft.ACE.OLEDB.12.0;

Data Source=C:\\Database\\QuanLyBanHang.accdb;

2.4 Đóng và mở kết nối



- **Mở kết nối với CSDL bằng cách gọi phương thức Open() của lớp Connection**

`Conn.open();` hoặc `Conn.open(ConnStr);`

- **Đóng kết nối với CSDL bằng cách gọi phương thức Close() của lớp Connection**

`Conn.close();`

2.5 Connection Pooling



❖ **Pooling cho phép sử dụng lại các connection để mang lại hiệu quả trong việc kết nối đến CSDL**

- Mỗi Data Provider sẽ cài đặt connection pooling theo cách riêng
- Nếu kết nối thực hiện đã có trong pool thì sẽ được sử dụng lại
- Nếu kết nối thực hiện chưa có trong pool thì sẽ được tạo mới

❖ **Hoạt động Connection Pooling của Data Provider for SQL Server**

- Dựa trên thông tin của chuỗi kết nối, thuộc tính Pooling có giá trị là True hoặc False (mặc định là True)
- Nếu thuộc tính Pooling có giá trị là True thì có thể sử dụng lại kết nối từ pool.
- Nếu thuộc tính Pooling có giá trị là False thì kết nối sẽ được tạo mới

2.6 Bài tập ứng dụng



- ❖ **Demo bài tập ứng dụng (Viết code để tạo kết nối giao diện ứng dụng với cơ sở dữ liệu quản lý sinh viên)**

NHẬP DANH SÁCH SINH VIÊN

Lớp: 

Họ tên:

Giới tính: 

Ngày sinh: 

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------

Chương 3. Đối tượng Command





3.1 Giới thiệu đối tượng Command

3.2 Khởi tạo đối tượng Command

3.3 Thực hiện đối tượng Command

3.4 Truyền tham số bằng Parameters

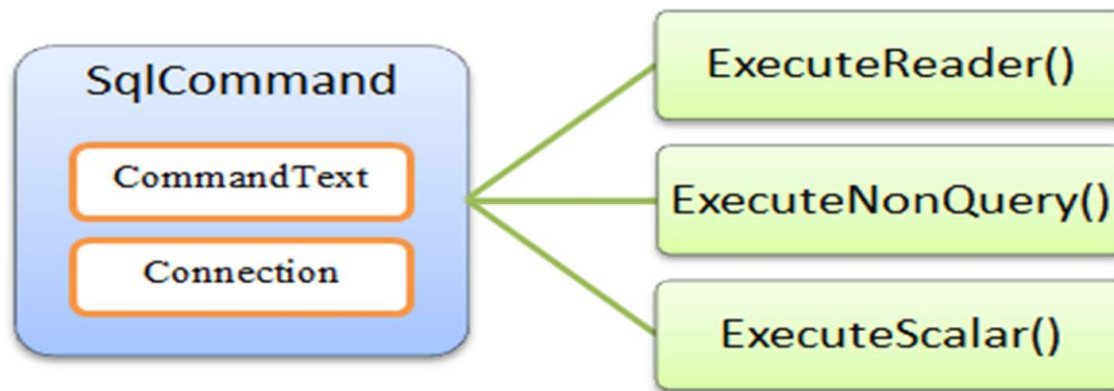
3.5 Bài tập ứng dụng

3.1 Giới thiệu đối tượng Command



❖ Đối tượng Command trên .NET cho phép bạn thi hành trực tiếp những câu lệnh SQL, chẳng hạn như INSERT, SELECT, UPDATE và DELETE lên dữ liệu nguồn để thêm, chọn, cập nhật và xóa dữ liệu trên CSDL. Bạn cũng có thể dùng đối tượng command để thi hành những stored procedure trong CSDL.

❖ Đối tượng Command cho phép bạn chọn kiểu tương tác mà bạn muốn thực hiện với database. Ví dụ, chúng ta có thể thực hiện các lệnh select, insert, modify, và delete các dòng trong một table của database.



3.1 Giới thiệu đối tượng Command



❖ **Đối tượng Command**

- Command là đối tượng dùng để thực hiện các truy vấn SQL và các lệnh quản lý dữ liệu trước khi gửi chúng đến đích. Command sử dụng các Parameter cho phép bạn gọi các thủ tục nội tại (stored procedures) hoặc các truy vấn..

❖ **Thuộc tính:**

- Connection: Đối tượng Connection, xác định kết nối thực hiện lệnh;
- CommandText: Câu lệnh sql cần thực hiện;
- CommandType: Loại câu lệnh (Text, TableDirect, StoredProc);
- CommandTimeout: Thiết lập/Lấy thời gian chờ thực hiện lệnh. Sau khi chờ 1 khoảng thời gian nếu vượt quá sẽ báo lỗi.
- Parameters: Các tham số truyền vào cho đối tượng command.
- Transaction: Thiết lập/lấy giao tác mà đối tượng Command thực thi.

3.1 Giới thiệu đối tượng Command



❖ Phương thức

- **ExecuteReader:** Thực thi câu lệnh CommandText của đối tượng Command và trả về kiểu DataReader.
- **ExecuteNonQuery:** Thực thi câu lệnh CommandText của đối tượng Command, đây là dạng câu lệnh cập nhật cơ sở dữ liệu (xoá /sửa) nên chỉ trả về số dòng bị ảnh hưởng mà không trả về dòng dữ liệu nào.
- **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..

3.1 Giới thiệu đối tượng Command



- **Các phương thức của lớp COMMAND**

Để gọi thi hành một Command bạn sử dụng các phương thức của lớp đối tượng Command như sau :

Các phương thức	Mô tả
ExecuteNonQuery	Cho thi hành một câu lệnh SQL đối với một Connection và trả về số mẫu tin bị ảnh hưởng. Chỉ dùng để thi hành câu lệnh không trả về mẫu tin nào cả, chẳng hạn lệnh DELETE.
ExecuteReader	Overloaded. Hàm này được dùng khi bạn muốn thi hành một câu lệnh có trả về các mẫu tin, chẳng hạn như lệnh SELECT. Các mẫu tin sẽ được trả về cho đối tượng SqlCommand. Bạn để ý rằng lớp DataReader là một lớp forward-only data nghĩa là chỉ dùng để đọc lại dữ liệu để cho hiển thị chứ không cập nhật được.

3.1 Giới thiệu đối tượng Command



Các phương thức	Mô tả
ExecuteScalar	Hàm này cho thi hành một câu truy vấn (query), và trả về cột đầu tiên của mẫu tin đầu tiên của result set được trả về. Nếu có nhiều mẫu tin hoặc nhiều cột thì hàm sẽ phớt lờ. Hàm này chạy nhanh hơn ExecuteReader và ít tốn overhead. Như vậy chỉ dùng hàm này khi bạn chỉ muốn một mẫu tin trả về hoặc khi bạn dùng một hàm nhóm như COUNT chẳng hạn.
ExecuteXmlReader	Hàm này cũng tương tự như ExecuteReader, nhưng trả lại các mẫu tin dưới dạng XML.

Lưu ý : bạn không được dùng ExecuteNonQuery để thi hành một câu lệnh SELECT vì ExecuteNonQuery không trả về dữ liệu

3.1 Giới thiệu đối tượng Command



❖ Các hàm khởi tạo đối tượng Command

- **New():** không có tham số nào.
- **New(cmdText as String):** Trong đó cmdText là câu lệnh truyền vào cho đối tượng Command.
- **New(cmdText as String, connection as SqlConnection):** Trong đó cmdText như trên, connection là đối tượng kết nối truyền vào cho đối tượng Command.
- **New(cmdText as String, connection as SqlConnection, transaction as SqlTransaction):** Trong đó cmdText, connection như trên, Transaction: là giao tác truyền cho đối tượng Command.

3.2 Khởi tạo đối tượng



1. Đối tượng SqlCommand

- ❖ Khai báo và khởi tạo đối tượng Command.
- ❖ SqlCommand cmd;
- ❖ String SQL_Cmd="Select * from SinhVien"; Cmd = new SqlCommand(SQL_Cmd, Conn); // Conn là biến đối tượng kết nối đã được mở
 - **Hoặc**
- ❖ SqlCommand cmd = new SqlCommand(SQL_Cmd, Conn);
- ❖ **Khai báo và khởi tạo đối tượng Command.**
 - **Hoặc**
- ❖ SqlCommand cmd; SqlCommand cmd = new();
- ❖ //Gán giá trị cho các thuộc tính của đối tượng Command
- ❖ cmd.Connection= Conn; cmd.CommandText=SQL_Cmd;

3.2 Khởi tạo đối tượng



- **Các thuộc tính của lớp Command**

Các thuộc tính	Mô tả
CommandText	Có thể là một câu lệnh SQL, tên một Store Procedure hoặc tên một bảng tùy vào trị của CommandType.
CommandTimeout	Đây là thời gian (tính theo giây) chờ thi hành Command này, mặc định là 30 giây. Nếu Command không được thi hành trong khoảng thời gian này thì một biệt lệ sẽ được tung ra.
CommandType	Thuộc tính này cho biết làm thế nào để phân biệt nội dung của CommandText là Text, StoredProcedure hoặc TableDirect...
Connection	Đây là connection nối kết cơ sở dữ liệu. Chẳng hạn SqlConnection

3.2 Khởi tạo đối tượng



• Các thuộc tính của lớp Command

Các thuộc tính	Mô tả
DesignTimeVisible	Thuộc tính này dùng để cho biết liệu đối tượng Command có nên thực hiện lên trên một ô control Windows Form Designer hay không. Trị mặc định là False
Parameters	Thuộc tính này cho tìm lại collection (SqlParameter Collection) các thông số của thuộc tính CommandText
Transaction	Thuộc tính này được dùng để tìm lại hoặc đặt giao dịch mà command đang thi hành. Bạn phải để ý là đối tượng transaction cũng được kết nối về cùng một đối tượng connection như đối tượng command.
UpdatedRowsource	Thuộc tính này lấy hoặc đặt, để kết quả thi hành câu lệnh Command này sẽ áp dụng đối với đối tượng DataRow khi nó được sử dụng bởi hàm Update của DbDataAdapter. Thuộc tính này sẽ mang một trong những giá trị của enumeration

3.2 Khởi tạo đối tượng



2. Đối tượng OracleCommand

- ❖ Khai báo và khởi tạo đối tượng OracleCommand.
- ❖ OracleCommand cmd;
- ❖ String SQL_Cmd="Select * from SinhVien"; Cmd = new OracleCommand (SQL_Cmd, Conn); // Conn là biến đối tượng kết nối đã được mở
 - **Hoặc**
- ❖ OracleCommand cmd = new OracleCommand (SQL_Cmd, Conn);
- ❖ **Khai báo và khởi tạo đối tượng Command.**
 - **Hoặc**
- ❖ OracleCommand cmd; OracleCommand cmd = new();
- ❖ //Gán giá trị cho các thuộc tính của đối tượng Command
- ❖ cmd.Connection= Conn; cmd.CommandText=SQL_Cmd;

3.2 Khởi tạo đối tượng



3. Đối tượng OleDbCommand

- ❖ Khai báo và khởi tạo đối tượng OleDbCommand.
- ❖ OleDbCommand cmd;
- ❖ String SQL_Cmd="Select * from SinhVien"; Cmd = new OleDbCommand(SQL_Cmd, Conn); // Conn là biến đối tượng kết nối đã được mở
 - **Hoặc**
- ❖ OleDbCommand cmd = new OleDbCommand(SQL_Cmd, Conn);
- ❖ **Khai báo và khởi tạo đối tượng Command.**
 - **Hoặc**
- ❖ OleDbCommand cmd; OleDbCommand cmd = new();
- ❖ //Gán giá trị cho các thuộc tính của đối tượng Command
- ❖ cmd.Connection= Conn; cmd.CommandText=SQL_Cmd;

3.3 Thực hiện đối tượng Command



- ❖ Bước 1: Thực hiện kết nối CSDL.
- ❖ Bước 2: Khởi tạo đối tượng Command
- ❖ Bước 3: Thực thi

3.3 Thực hiện đối tượng Command



Ví dụ: Sử dụng đối tượng Command để đếm số khách hàng.

- Tạo đối tượng SqlConnection và thực hiện mở kết nối tới CSDL QuanLyBanHang.
- Tạo đối tượng Command và sử dụng phương thức ExecuteScalar để đếm số bản ghi trong bảng KhachHang như sau:

```
SqlCommand command = new  
    SqlCommand("",conn); command.CommandText = "Select  
COUNT(*) From KhachHang";
```

```
int Sokhachhang =(int) command.ExecuteScalar();
```

```
MessageBox.Show("Total: “ + Sokhachhang);
```

3.3 Thực hiện đối tượng Command



❖ Querying Data – Truy vấn dữ liệu

- Khi dùng một lệnh SQL select, bạn lấy được một dữ liệu từ database để hiển thị. Để làm được điều này với SqlCommand, bạn cần dùng phương thức ExecuteReader để trả về một đối tượng SqlDataReader.

// 1. Khởi tạo mới một đối tượng Command với kết nối conn

```
SqlCommand cmd = new SqlCommand("select *  
from KhachHang", conn);
```

// 2. Gọi phương thức Execute reader

```
SqlDataReader rdr = cmd.ExecuteReader();
```

3.3 Thực hiện đối tượng Command



❖ Inserting Data – Chèn dữ liệu

- Để chèn dữ liệu vào database, dùng phương thức ExecuteNonQuery() của đối tượng SqlCommand. Đoạn code sau cho thấy cách chèn dữ liệu vào một bảng trong database:

```
// prepare command string
string insertString = @"
insert into MonHoc (MaMh, TenMh, LT, TH) values ('0036',
        'Lập Trình CSDL',30,30)";

// 1. Khởi tạo mới một đối tượng Command với kết nối conn
SqlCommand cmd = new SqlCommand(insertString, conn);

// 2. Gọi phương thức ExecuteNonQuery thực thi câu lệnh
cmd.ExecuteNonQuery();
```

3.3 Thực hiện đối tượng Command



❖ Updating Data – Cập nhật dữ liệu

- Phương thức ExecuteNonQuery cũng được dùng để cập nhật dữ liệu, như đoạn mã sau:

```
// prepare command string
string updateString = @"
update MonHoc set MaMh= '0036'
where Tenmh= 'Lập Trình Cơ Sở Dữ Liệu';

// 1. Tạo mới đối tượng command với command text
SqlCommand cmd = new SqlCommand(updateString);

// 2. Thiết lập kết nối cho thuộc tính Connection
cmd.Connection = conn;

// 3. Gọi phương thức ExecuteNonQuery thực hiện
cmd.ExecuteNonQuery();
```

3.3 Thực hiện đối tượng Command



❖ Deleting Data – Xóa dữ liệu

- Chúng ta cũng có thể xóa dữ liệu bằng phương thức `ExecuteNonQuery()`. Ví dụ sau cho thấy cách xóa một dòng từ database với phương thức `ExecuteNonQuery()`:

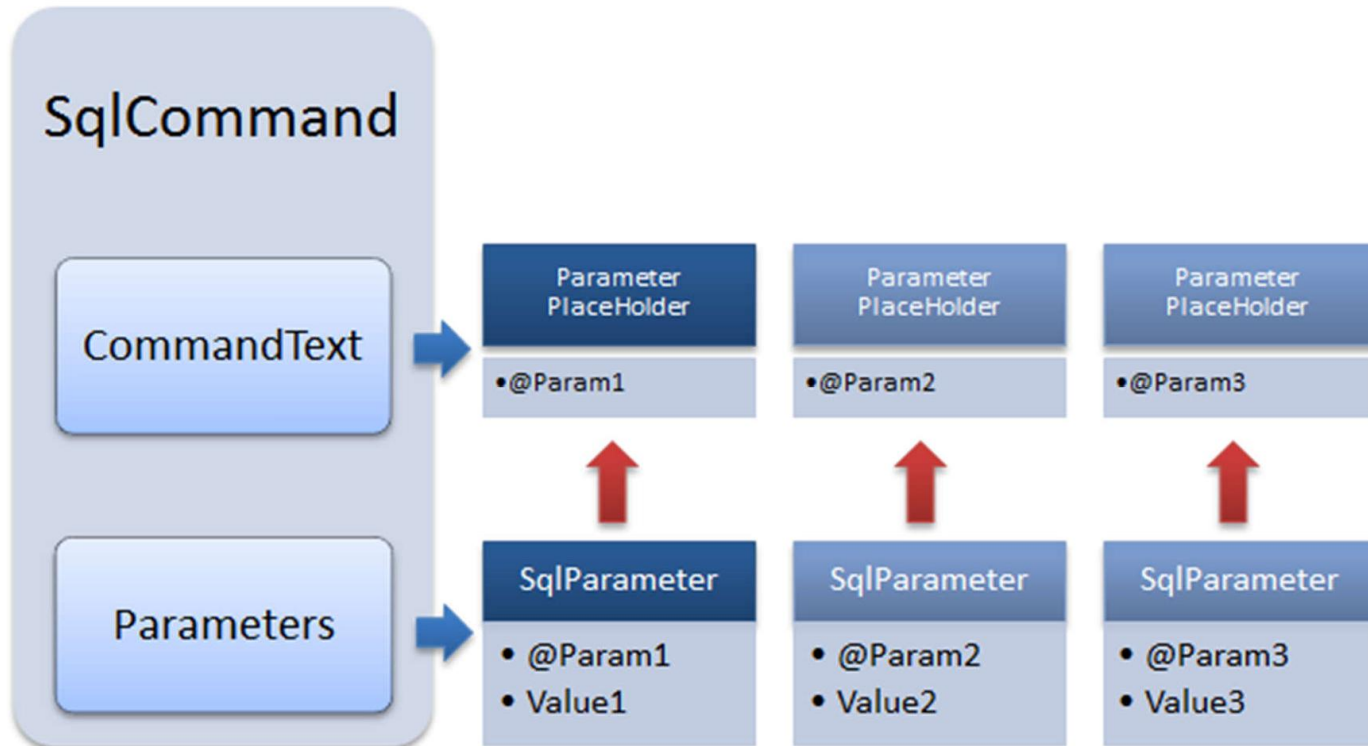
```
// prepare command string
string deleteString = @"delete from Monhoc
where Mamh= '0036'";
// 1. Tạo mới đối tượng command SqlCommand cmd = new
    SqlCommand();
// 2. Gán giá trị cho thuộc tính CommandText
    cmd.CommandText = deleteString;
// 3. Thiết lập kết nối cho thuộc tính Connection
    cmd.Connection = conn;
// 4. Gọi phương thức ExecuteNonQuery thực hiện
    cmd.ExecuteNonQuery();
```

3.4 Truyền tham số bằng Parameters



- ❖ Khi bạn làm việc với dữ liệu, bạn sẽ thường xuyên cần lọc kết quả dựa trên một vài điều kiện. Thông thường điều này được thực hiện bằng cách lấy dữ liệu nhập từ người dùng và tạo ra câu truy vấn SQL từ đó.
- ❖ Trong trường hợp này chúng ta sử dụng đối tượng Parameters.

3.4 Truyền tham số bằng Parameters



(Mô hình kết hợp giữa SqlParameter và SqlCommand)

3.4 Truyền tham số bằng Parameters



❖ ***Dùng câu truy vấn với parameter bao gồm ba bước sau:***

- Tạo một SqlCommand từ một câu lệnh có parameter.
- Khai báo một đối tượng SqlParameter, gán giá trị thích hợp cho nó.
- Gán đối tượng SqlParameter vào property Parameters của đối tượng SqlCommand.

3.4 Truyền tham số bằng Parameters



❖ Tạo một đối tượng SqlCommand sử dụng Parameter

- Bước đầu tiên là tạo một câu lệnh chứa các parameter placeholder (tên của parameter). Tên này sẽ được thay thế bởi giá trị thực sự của parameter khi SqlCommand thực thi. Cú pháp đúng của một parameter là dùng kí hiệu tiền tố '@' trong tên của parameter như sau:

// 1. declare command object with parameter

```
SqlCommand cmd = new SqlCommand("select * from Customers  
where city = @City", conn);
```

Trong constructor của SqlCommand trên, tham số đầu tiên chứa một khai báo parameter, @City. Ví dụ này dùng một parameter, nhưng bạn có thể có nhiều parameter tùy theo số lượng bạn cần để tạo câu truy vấn. Mỗi parameter sẽ so khớp với một đối tượng SqlParameter được gán vào đối tượng SqlCommand.

3.4 Truyền tham số bằng Parameters



❖ Khai báo một đối tượng SqlParameter

- Mỗi parameter trong câu lệnh SQL phải được định nghĩa. Đây là mục đích của kiểu SqlParameter. Mã nguồn của bạn phải định nghĩa một đối tượng SqlParameter cho mỗi parameter trong câu lệnh SQL của đối tượng SqlCommand. Đoạn mã sau định nghĩa một SqlParameter cho parameter @City trong phần trước:

```
// 2. define parameters used in command object SqlParameter param  
    = new SqlParameter(); param.ParameterName = "@City";  
    param.Value = inputCity;
```

Lưu ý: Property ParameterName của đối tượng SqlParameter phải được viết đúng với parameter được dùng trong câu lệnh SQL của SqlCommand. Bạn cũng phải xác định giá trị cho các parameter. Khi đối tượng SqlCommand được thực thi, parameter sẽ được thay thế bằng giá trị của nó.

3.4 Truyền tham số bằng Parameters



❖ Kết hợp đối tượng SqlParameter với đối tượng SqlCommand

- Với mỗi parameter được định nghĩa trong câu lệnh SQL của đối tượng SqlCommand phải được định nghĩa một SqlParameter. Bạn cũng phải để đối tượng SqlCommand biết về SqlParameter bằng cách gán đối tượng SqlParameter cho property Parameters của đối tượng SqlCommand. Dòng mã sau cho thấy cách làm điều này:

```
// 3. add new parameter to command object  
cmd.Parameters.Add(param);
```

Đối tượng SqlParameter là tham số trong phương thức Add() của property Parameters của đối tượng SqlCommand trên. Bạn phải thêm một SqlParameter duy nhất cho mỗi parameter đã định nghĩa trong câu lệnh SQL của đối tượng SqlCommand.

3.5 Bài tập ứng dụng



❑ **Nhắc lại kiến thức về đối tượng Command**

- ❖ **Đối tượng Command trên .NET FRAMEWORK cho phép thi hành trực tiếp những câu lệnh SQL, chẳng hạn như INSERT, SELECT, UPDATE và DELETE lên dữ liệu nguồn để thêm, chọn, cập nhật và xóa dữ liệu trên CSDL nguồn. Cụ thể là cho phép tương tác chọn, thêm, xóa, cập nhật trên giao diện người dùng với CSDL.**

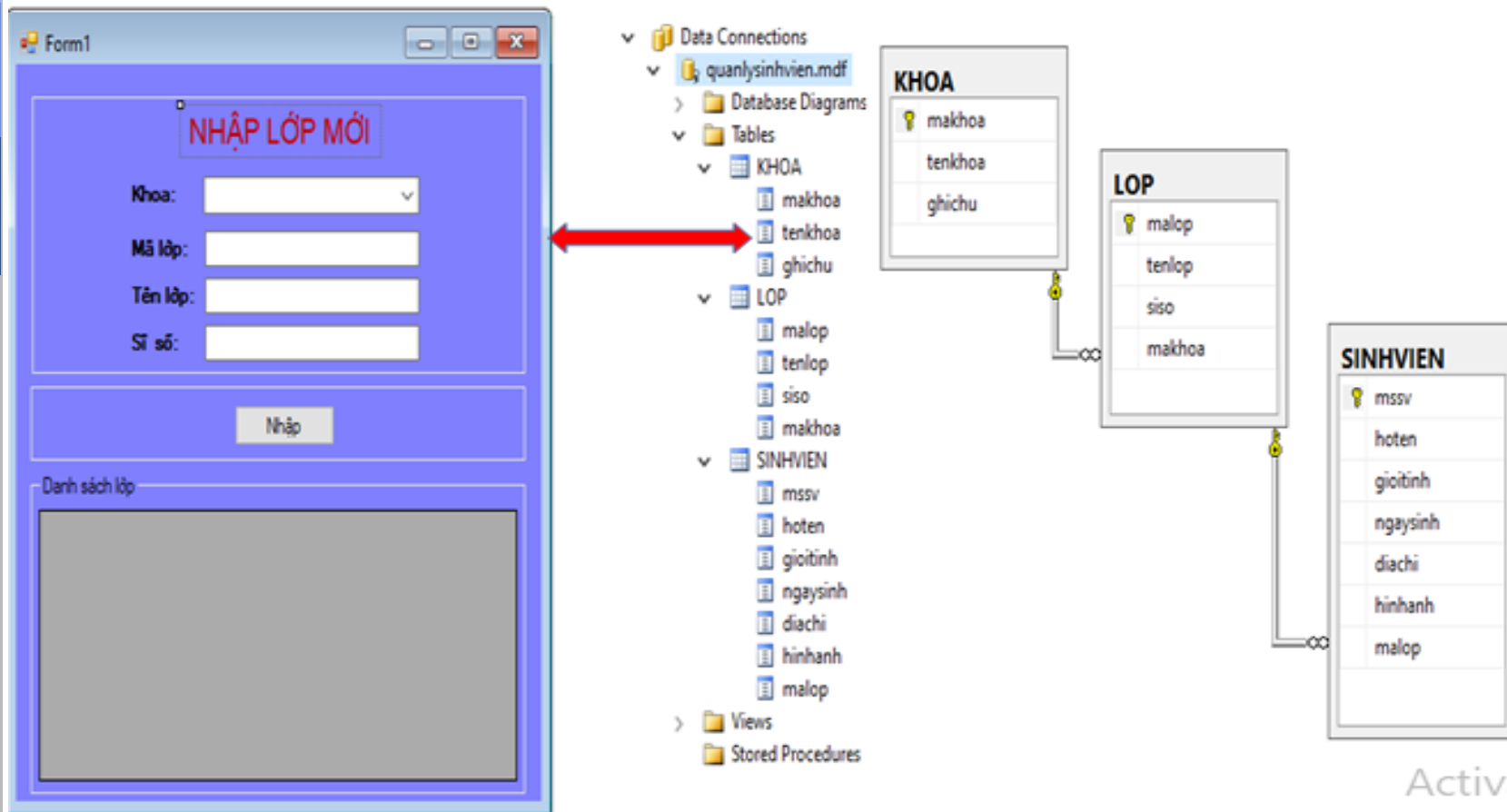
❑ **Demo ứng dụng đối tượng Command**

- ❖ **Bài tập: Xây dựng phần mềm quản lý sinh viên và thực hiện các chức năng sau:**
 - Truy vấn dữ liệu
 - Thêm
 - Sửa
 - Xóa
 - Truyền tham số

Truy vấn dữ liệu với đối tượng Command



- ❖ Sinh viên xây dựng giao diện phần mềm quản lý sinh viên và thiết kế cơ sở dữ liệu như hình



Truy vấn dữ liệu với đối tượng Command



❖ Nhập dữ liệu cho bảng Khoa và Lớp như hình:

KHOA: Query(viet...NLYSINHVIEN.MDF) X				LOP: Query(vietk...ANLYSINHVIEN.MDF)		Form1.cs*	
	makhoa	tenkhoa	qhichu				
	CNTT	Khoa Công nghệ thông tin	NULL				
	ĐT	Khoa Điện tử	NULL				
	ĐL	Khoa Điện Lạnh	NULL				
	CK	Khoa Cơ Khí	NULL				
	KT	Khoa Kinh tế	NULL				
	NN	Khoa Ngoại ngữ	NULL				
	DL	Khoa Du Lịch	NULL				
▶*	NULL	NULL	NULL				
LOP: Query(vietk...ANLYSINHVIEN.MDF) X							
	malop	tenlop	siso	makhoa			
	17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT			
	17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT			
▶*	NULL	NULL	NULL	NULL			

Truy vấn dữ liệu với đối tượng Command



❖ Tạo store procedure phục vụ cho việc truy vấn dữ liệu

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Server Explorer' pane shows the database structure for 'quanlysinhvien.mdf'. The 'Tables' folder is expanded, showing tables under 'KHOA' (makhoa, tenkhoa, ghichu) and 'LOP' (malop, tenlop, siso, makhoa). The 'SINHVIEN' folder is also expanded, showing tables (mssv, hoten, gioitinh, ngaysinh, diachi, hinhanh, malop). The 'Stored Procedures' folder is visible at the bottom. On the right, the 'Form1.Designer.cs' file is open, showing the definition of a stored procedure named 'sp_Select'.

```
CREATE PROCEDURE sp_Select @tenbang nvarchar(50)
AS
BEGIN
    declare @sql nvarchar(4000) set @sql=''
    @sql = @sql+'select * from '+@tenbang
    exec(@sql)
END
RETURN
```

Truy vấn dữ liệu với đối tượng Command



- ❖ Sinh viên tiến hành viết code truy vấn dữ liệu load lên **ComboBox (Khoa)** - ID: **cboKhoa** và **DataGridView (Danh sách lớp)** - ID: **dgrLop** trên giao diện màn hình theo các bước sau:

Form1

NHẬP LỚP MỚI

Khoa: **Khoa Cơ khí**

Mã lớp: Khoa Công nghệ thông tin

Tên lớp: Khoa Điện tử

ST số: Khoa Kinh tế

Khoa Ngoại ngữ

Khoa Xây dựng

Nhập

Danh sách lớp

malop	tenlop	siso	makhoa
17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT

KHOA: Query(viet...NLYSINHVIEN.MDF)

makhoa	tenkhoa	qhichu
CNTT	Khoa Công nghệ thông tin	NULL
ĐT	Khoa Điện tử	NULL
ĐL	Khoa Điện Lạnh	NULL
CK	Khoa Cơ Khí	NULL
KT	Khoa Kinh tế	NULL
NN	Khoa Ngoại ngữ	NULL
DL	Khoa Du Lịch	NULL
NULL	NULL	NULL

LOP: Query(vietk...ANLYSINHVIEN.MDF)

malop	tenlop	siso	makhoa
17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT
NULL	NULL	NULL	NULL

Truy vấn dữ liệu với đối tượng Command



❖ Bước 1: Khai báo các đối tượng và khởi tạo chuỗi kết nối

```
SqlConnection cn;  
SqlCommand cmd;  
SqlDataAdapter da;  
string connectionString = "Data Source=.\SQLEXPRESS;  
AttachDbFilename= E:\\LTCSDL\\T_Khanh_18C1\\  
QuanLySinhVien\\QuanLySinhVien\\App_Data\\quanlysinhvien.mdf;  
Integrated Security=True;User Instance=True";
```

```
public string ConnectionString  
{  
    get { return connectionString; }  
    set { connectionString = value; }  
}
```

Truy vấn dữ liệu với đối tượng Command



❖ Bước 2: Xây dựng hàm mở kết nối và đóng kết nối với CSDL

```
public void Connect()
{
    try
    {
        cn = new SqlConnection(ConnectionString);
        if (cn.State == ConnectionState.Closed || cn.State ==
            ConnectionState.Broken)
        {
            cn.Open();
        }
    }
    catch (SqlException sqllex)
    {
        throw sqllex;
    }
}

public void Disconnect()
{
    cn.Close();
}
```

Truy vấn dữ liệu với đối tượng Command



❖ Bước 3: Viết code lấy thông tin bảng KHOA load lên cboKhoa trên Form giao diện

```
public DataTable LayThongTinBang(string tenBang)
{
    cmd = new SqlCommand("sp_Select", cn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@tenbang", SqlDbType.NVarChar).Value = tenBang;
    da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);

    return dt;
}

public void LoadKhoa()
{
    Connect();
    cboKhoa.DisplayMember = "tenkhoa";
    cboKhoa.ValueMember = "makhoa";
    cboKhoa.DataSource = LayThongTinBang("KHOA");
    Disconnect();
}

private void Form1_Load(object sender, EventArgs e)
{
    LoadKhoa();
}
```


- ❖ **Bước 4: Nhấn F5 biên dịch chương trình kết quả như sau:**

Form1

NHẬP LỚP MỚI

Khoa:

Mã lớp:

Tên lớp:

ST số:

Nhập

Danh sách lớp

Truy vấn dữ liệu với đối tượng Command



- ❖ **Bước 5: Viết code lấy thông tin bảng LOP load lên DataGridView trên Form giao diện**



```
private void LoadDanhSachLop()
{
    dgrDSLop.DataSource = LayThongTinBang("LOP").DefaultView;
}
private void Form1_Load(object sender, EventArgs e)
{
    LoadKhoa();
    LoadDanhSachLop();
}
```

Truy vấn dữ liệu với đối tượng Command



- ❖ **Bước 6: Nhấn F5 biên dịch chương trình kết quả như sau:**

Form1

NHẬP LỚP MỚI

Khoa:

Mã lớp:

Tên lớp:

ST số:

Danh sách lớp

	malop	tenlop	siso	makhoe
	17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
	17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT

Thêm dữ liệu với đối tượng Command



- ❖ Sinh viên tiến hành viết code thêm dữ liệu LOP vào CSDL khi người dùng nhập đầy đủ thông tin và nhấn vào nút **Nhập** trên giao diện phần mềm theo các bước sau:

Form1

NHẬP LỚP MỚI

Khoa: Khoa Công nghệ thông tin

Mã lớp: 18C1-LTM1

Tên lớp: Lập trình máy tính 1 - Khóa 2018

ST số: 65

Nhập

Danh sách lớp

	malop	tenlop	siso	makhoa
	17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
	17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT
▶	18C1-LTM1	Lập trình máy tính 1 - Khóa 2018	65	CNTT
*				

LOP: Query(vietk...ANLYSINHVIEN.MDF) X Form1.cs Form1.Designer.cs Form1.cs

	malop	tenlop	siso	makhoa
▶	17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
	17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT
	18C1-LTM1	Lập trình máy tính 1 - Khóa 2018	65	CNTT
*		NULL	NULL	NULL

Thêm dữ liệu với đối tượng Command



❖ Bước 1: Tạo store procedure phục vụ cho việc thêm dữ liệu LOP vào cơ sở dữ liệu

Server Explorer

- Data Connections
 - quanlysinhvien.mdf
 - Database Diagrams
 - Tables
 - KHOA
 - LOP
 - malop
 - tenlop
 - siso
 - makhoa
 - SINHVIEN
 - Views
 - Stored Procedures
 - sp_InsertLop
 - sp_Select

dbo.sp_InsertLop:....ANLVSINHVIEN.MDF)* X Form1.cs Form1.Designer.cs Form1.cs [D

```
CREATE PROCEDURE sp_InsertLop @malop nchar(10),@tenlop nvarchar(50),
                             @siso int, @makhoa nchar(10)
AS
BEGIN
    insert into LOP values(@malop ,@tenlop ,@siso , @makhoa )
END
RETURN
```

Thêm dữ liệu với đối tượng Command



❖ **Bước 2: Viết code lấy dữ liệu từ Form khi người dùng nhập thông tin trên giao diện phần mềm**

```
string malop, tenlop, makhoa;  
int siso;  
public void LayDuLieuTuForm() {  
    malop = txtMaLop.Text;  
    tenlop = txtTenLop.Text;  
    siso = int.Parse(txtSiSo.Text);  
    makhoa = cboKhoa.SelectedValue.ToString();  
}
```

Thêm dữ liệu với đối tượng Command



❖ Bước 3: Viết code tạo phương thức thêm thông tin lớp học vào CSDL

```
public void ThemLop()
{
    Connect();
    cmd = new SqlCommand("sp_InsertLop", cn);
    cmd.CommandType =
        CommandType.StoredProcedure;
    cmd.Parameters.Add("@malop",
        SqlDbType.NChar).Value = malop;
    cmd.Parameters.Add("@tenlop",
        SqlDbType.NVarChar).Value = tenlop;
    cmd.Parameters.Add("@siso",
        SqlDbType.Int).Value = siso;
    cmd.Parameters.Add("@makhoa",
        SqlDbType.NChar).Value = makhoa;
    cmd.ExecuteNonQuery();
    Disconnect();
}
```

Thêm dữ liệu với đối tượng Command



❖ Bước 4: Viết code cho nút **Nhập** thêm thông tin lớp học vào CSDL

```
private void btnNhap_Click(object sender, EventArgs e)
{
    LayDuLieuTuForm();
    ThemLop();
    LoadDanhSachLop();
}
```

Thêm dữ liệu với đối tượng Command



- ❖ **Bước 5: Nhấn F5 biên dịch chương trình kết quả chạy như sau:**

Form1

NHẬP LỚP MỚI

Khoa: Khoa Công nghệ thông tin

Mã lớp: 18C1-LTM1

Tên lớp: Lập trình máy tính 1 - Khóa 2018

Sĩ số: 65

Nhập

Danh sách lớp

	malop	tenlop	siso	makhhoa
	17C1-LTM1	Lập trình máy tính 1 - Khóa 2017	60	CNTT
	17C1-LTM2	Lập trình máy tính 2 - Khóa 2017	75	CNTT
▶	18C1-LTM1	Lập trình máy tính 1 - Khóa 2018	65	CNTT
*				

< >

Chương 4. Đối tượng DataReader





4.1

Giới thiệu đối tượng Datareader

4.2

Nhận kết quả trả về từ Command

4.3

Xử lý kết quả trả về

4.4

Xử lý kết quả trả về từ nhiều câu truy vấn

4.5

Bài tập ứng dụng



4.1 Giới thiệu đối tượng Datareader

- ❖ DataReader là đối tượng phù hợp để đọc dữ liệu một cách hiệu quả nhất. Như tên gọi, chúng ta không thể dùng nó để ghi dữ liệu.
- ❖ Chúng ta có thể đọc dữ liệu từ đối tượng DataReader theo hướng forward-only trong một thứ tự nhất định. Mỗi lần đọc một vài dữ liệu, chúng ta phải lưu nó nếu cần thiết bởi vì bạn không thể quay trở lại và đọc nó một lần nữa.
- ❖ Kiểu thiết kế forward-only của DataReader để giúp nó hoạt động nhanh. Nó không thể di chuyển trực tiếp đến các dòng dữ liệu ở vị trí bất kỳ và không thể ghi vào dữ liệu nguồn. Do đó, nếu bạn chỉ yêu cầu đọc một nhóm dữ liệu một lần và cần phương pháp nhanh nhất, DataReader là lựa chọn tốt nhất.

4.1 Giới thiệu đối tượng Datareader



❖ Cách sử dụng

- DataReader được sử dụng để chứa kết quả trả về từ đối tượng Command với việc thực thi hàm ExecuteReader.

❖ Một số thuộc tính và phương thức thông dụng của DataReader:

- FieldCount: trả về số trường có trong record hiện hành.
- IsClosed: trả về giá trị boolean xác định đối tượng DataReader có bị đóng hay không.

4.1 Giới thiệu đối tượng Datareader



❖ **Phương thức:**

- Close(): Đóng.
- GetBoolean(), GetByte(), GetChar(), GetDateTime(), GetDecimal(): lấy các giá trị tại cột đang xét tùy vào kiểu.
- GetValue(), GetValues(): lấy về giá trị hoặc tập giá trị ở dạng “nguyên thủy” (kiểu dữ liệu gốc của Database).
- NextResult(): Nhóm Kết quả tiếp theo.
- Read(): đọc Record tiếp theo.

4.2 Nhận kết quả trả về từ Command



- ❖ Có một chút khác biệt để lấy được một thể hiện của SqlDataReader so với các đối tượng ADO.NET khác. Chúng ta phải gọi phương thức *ExecuteReader()* của một đối tượng SqlCommand, như:

SqlDataReader rdr = cmd.ExecuteReader();

Phương thức ExecuteReader() của đối tượng SqlCommand, cmd, trả về một thể hiện của SqlDataReader.

Lưu ý: Không dùng toán tử **new** để tạo một SqlDataReader

4.3 Xử lý kết quả trả về từ Command



❖ **Đọc dữ liệu**

- SqlDataReader trả về dữ liệu qua một luồng liên tục. Để đọc dữ liệu, chúng ta phải lấy dữ liệu từ một bảng từng dòng một (row-by-row).
- Mỗi lần một dòng được đọc, dòng trước đó sẽ không còn hiệu lực.
- Để đọc lại dòng đó, bạn cần phải tạo một thể hiện mới của SqlDataReader và đọc xuyên qua luồng dữ liệu một lần nữa.

4.3 Xử lý kết quả trả về từ Command



❖ Đọc dữ liệu

- Phương pháp điển hình để đọc từ luồng dữ liệu trả về bởi SqlDataReader là lặp qua mỗi dòng với một vòng lặp while. Đoạn code sau cho thấy cách làm điều này:

```
while (rdr.Read())
{ // get the results of each column  string Mamh =
  (string)rdr["MaMh"];  string TenMh=
  (string)rdr["TenMh"];  string LT = (string)rdr["LT"];
  string TH = (string)rdr["TH"];
  // print out the results  Console.Write("{0,-25}", Mamh);
  Console.Write("{0,-20}", TenMh);
  Console.Write("{0,-25}", LT);
  Console.Write("{0,-25}", TH);  Console.WriteLine();
}
```



4.4 Xử lý kết quả trả về từ nhiều truy vấn

❖ Demo (Sử dụng DataReader hiển thị danh sách sinh viên lên lưới).

NHẬP DANH SÁCH SINH VIÊN

Lớp: 

Họ tên: 

Giới tính: 

Ngày sinh: 

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------

4.5 Bài tập ứng dụng



- ❖ **Demo xây dựng phần mềm quản lý sinh viên như hình.**



Chương 5. Đối tượng DataAdapter và DataSet





5.1 Khái niệm đối tượng DataAdapter

5.2 Khởi tạo đối tượng DataAdapter

5.3 Đối tượng DataSet

5.4 Sử dụng DataSet

5.5 Cập nhật những thay đổi vào CSDL

5.1 Khái niệm đối tượng DataAdapter



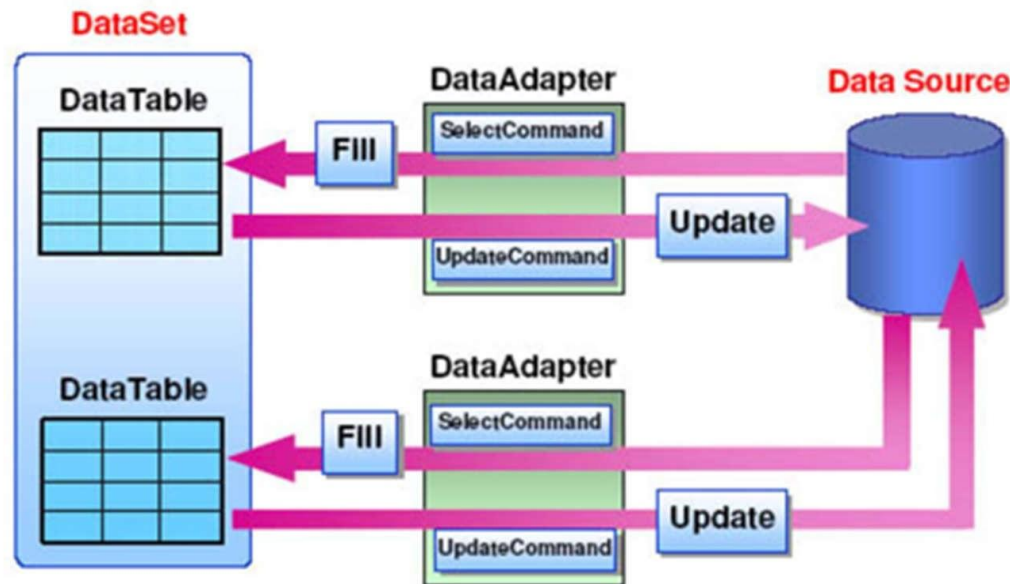
- ❖ DataAdapter là đối tượng làm trung gian lấy dữ liệu về cho DataSet, để DataSet thực hiện xử lý ngắt kết nối. Do vậy, mặc dù DataAdapter được liệt kê là đối tượng hướng kết nối nhưng thực chất nó phục vụ cho việc ngắt kết nối.
- ❖ **Thuộc tính**
 - SelectCommand, InsertCommand, UpdateCommand, DeleteCommand: các câu lệnh select , insert, update ,delete dữ liệu.
- ❖ **Phương thức:**
 - Fill: phương thức thực hiện kết quả trong câu lệnh select, rồi trả về cho DataSet xử lý.
 - Update: Gọi lệnh cập nhật các thay đổi của dữ liệu lên dữ liệu nguồn.

5.1 Khái niệm đối tượng 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... Và sẽ đổ toàn bộ dữ liệu vào DataSet được lưu trên bộ nhớ, sau đó sẽ sử dụng dữ liệu trên DataSet.



5.2 Khởi tạo đối tượng DataAdapter



❖ Khai báo sử dụng thư viện:

- Sử dụng đối tượng với dữ liệu Access: using System.Data.OleDb;
- Sử dụng đối tượng với dữ liệu SQL:

using System.Data.SqlClient;

❖ Khai báo khởi tạo đối tượng:

- Cú pháp 1:

```
OleDbDataAdapter <TênBiến> = new OleDbDataAdapter ();
```

- Hoặc

```
SqlDataAdapter <TênBiến> = new SqlDataAdapter ();
```

5.2 Khởi tạo đối tượng DataAdapter



❖ Ví dụ:

```
SqlCommand cmd = new SqlCommand();  
    cmd.CommandType = CommandType.Text;  
    cmd.CommandText = strSQL; cmd.Connection  
    = conn;  
SqlDataAdapter da = new SqlDataAdapter();  
da.SelectCommand = cmd;
```

5.2 Khởi tạo đối tượng DataAdapter



❖ Khai báo khởi tạo đối tượng:

- Cú pháp 2:

```
SqlDataAdapter <TênBiến> = new SqlDataAdapter(CâuLệnhSQL,  
BiếnKếtNối);
```

- Ví dụ:

```
string strSQL = "Select * From SinhVien"; SqlDataAdapter da = new  
SqlDataAdapter (strSQL,conn);
```

5.2 Khởi tạo đối tượng DataAdapter



Ví dụ: Chương trình đọc dữ liệu từ bảng KháchHang qua DataAdapter và đổ vào DataSet.

❖ **Bước 1:** Viết thủ tục OpenConnection

❖ **Bước 2:** Tạo DataAdapter và đọc dữ liệu từ CSDL sau đó đổ vào DataSet.

//Lấy dữ liệu qua mệnh đề SQL

```
string strSQL = "Select * From KháchHang";
```

//Tạo DataAdapter và lấy dữ liệu từ bảng KháchHang SqlDataAdapter

```
adapter = new SqlDataAdapter(strSQL, conn); OpenConnection();
```

```
DataSet ds = new DataSet();
```

//Đổ dữ liệu từ DataAdapter và DataSet

```
adapter.Fill(ds); conn.Close();
```

5.3 Đối tượng DataSet, DataTable, DataColumn, DataRow



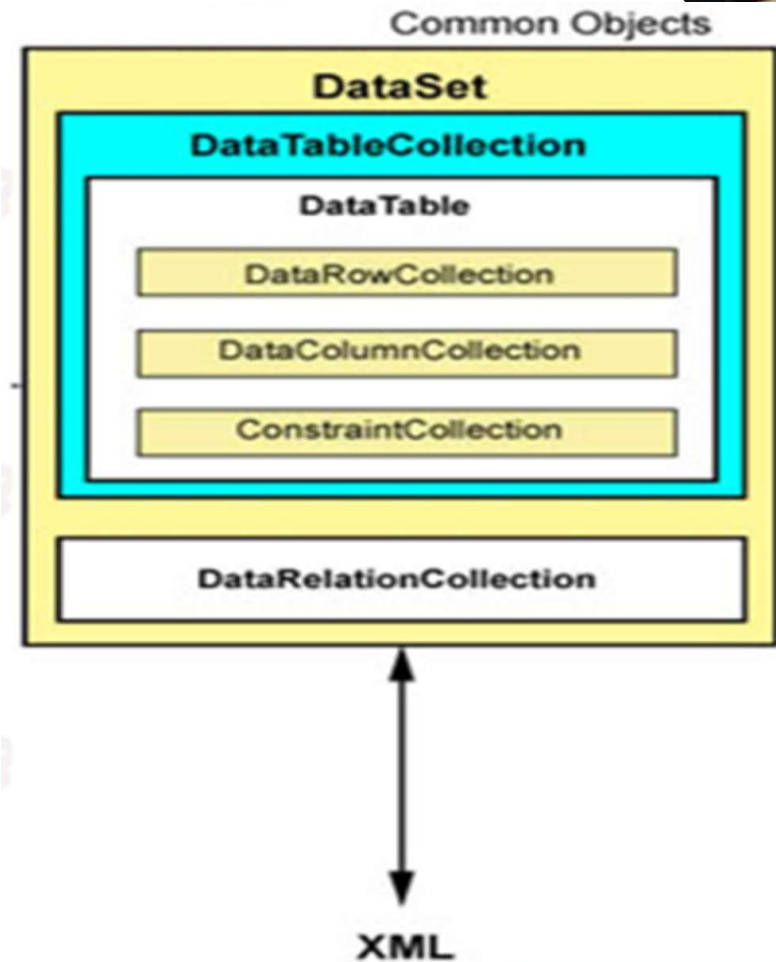
- ❖ DataSet là một đối tượng chứa dữ liệu trong bộ nhớ và có thể gồm nhiều bảng. DataSet chỉ chứa dữ liệu chứ không tương tác với nguồn dữ liệu. Thay vào đó, SqlDataAdapter sẽ được dùng để quản lý các kết nối với nguồn dữ liệu và cho chúng ta chế độ làm việc disconnected.
- ❖ SqlDataAdapter mở một kết nối chỉ khi cần thiết và đóng nó ngay sau khi tác vụ được hoàn thành. Ví dụ, SqlDataAdapter thực hiện các tác vụ sau, khi đổ dữ liệu vào DataSet:
 - Mở kết nối
 - Đổ dữ liệu vào DataSet (Fill)
 - Đóng kết nối

5.3 Đối tượng DataSet, DataTable, DataColumn, DataRow



❖ Mô hình ngắt kết nối DataSet:

- DataSet không quan tâm đến kiểu của CSDL;
- Lấy dữ liệu từ DataAdapter;
- DataSet xem như một CSDL trong bộ nhớ;
- DataSet gồm các thành phần con như: DataTable, DataRow, DataColumn, DataRelation, ...



5.3 Đối tượng DataSet, DataTable, DataColumn, DataRow



❖ Đối tượng DataSet

- Là phần cơ sở dữ liệu được lưu trữ trong bộ nhớ;
- Cơ chế ngắt kết nối;
- Thao tác với CSDL qua cầu nối là DataAdapter;
- Hỗ trợ đầy đủ đặc tính XML;
- Thao tác được với tất cả các mô hình lưu trữ dữ liệu: Flat file database, Relational database, Hierarchical.

❖ DataSet và DataReader đều sử dụng để lưu trữ dữ liệu, sự khác nhau giữa DataSet và DataReader:

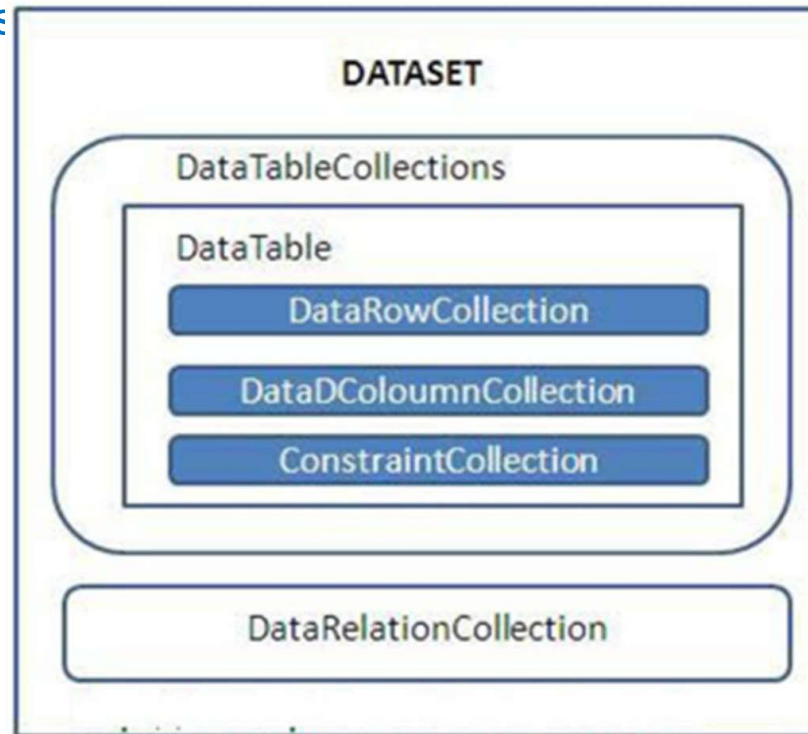
DataSet	DataReader
Có thể đọc/ghi dữ liệu	Chỉ đọc dữ liệu
Bao gồm nhiều bảng từ nhiều CSDL	Dựa trên một câu lệnh SQL từ một CSDL
Ngắt kết nối	Kết nối
Di chuyển qua các bản ghi theo hai chiều	Chỉ di chuyển qua các bản ghi theo một chiều (chiều tiến)
Truy cập chậm	Truy cập nhanh

5.3 Đối tượng DataSet, DataTable, DataColumn, DataRow



❖ Các thành phần chính của DataSet:

- Tables: Tập các bảng;
- Relations



5.3 Đối tượng DataSet, DataTable, DataColumn, DataRow



DataTable: Thể hiện một bảng trong CSDL.

❖ Thuộc tính và phương thức:

- TableName: Tên bảng;
- Columns: Danh sách các cột (trường) trong bảng;
- Rows: danh sách các hàng (bản ghi) trong bảng;
- PrimaryKey: Danh sách các cột là khóa chính của bảng;
- NewRow(): Phương thức tạo một hàng mới.

DataColumn: Đại diện cho một cột trong bảng

- ColumnName: Tên cột;
- DataType: Kiểu dữ liệu của cột;

DataRow: Đại diện cho một dòng (bản ghi)

- RowState: Trạng thái Added, Modified, Deleted...
- [i]: Truy xuất đến hàng có thứ tự i;
- Delete(): Phương thức xóa bản ghi.

5.4 Sử dụng DataSet để nhận giá trị trả về từ DataAdapter



❖ Đưa danh sách khách hàng lên DataGridView.

- **Ví dụ:** Đọc dữ liệu từ bảng Kháchhang trong CSDL QuanLyBanHang và đưa vào DataGridView.

- **Bước 1:** Kéo thả trên giao diện đối tượng DataGridView và đặt tên là gvwKhachHang

- **Bước 2:** Khai báo đối tượng SqlConnection, SqlDataAdapter, DataSet
`SqlConnection conn; SqlDataAdapter da; DataSet ds;`

- **Bước 3:** Viết thủ tục OpenConnection

```
private void OpenConnection()
{
    string connectionString = "Data Source=Server; Initial
        Catalog= QuanLyBanHang; UID=sa; PassWord="";
    conn = new SqlConnection(connectionString); conn.Open();
}
```

5.4 Sử dụng DataSet



- **Bước 4:** Đọc dữ liệu vào DataSet hiển thị trên DataGridView.

```
string strSQL = "Select * From KhachHang";  
//Gọi OpenConnection để mở kết nối với CSDL  
OpenConnection();  
//Đọc dữ liệu vào DataAdapter  
SqlDataAdapter adapter = new SqlDataAdapter(strSQL, conn);  
//Đổ dữ liệu vào DataSet  
DataSet dsKhachHang = new DataSet();  
    adapter.Fill(dsKhachHang);  
//Hiển thị dữ liệu từ DataSet ra DataGridView  
    gvwKhachHang.DataSource = dsKhachHang.Tables[0];  
    conn.Close();
```

5.4 Sử dụng DataSet



- **Bước 5:**

//Hiển thị dữ liệu từ DataSet ra DataGridView

```
gwwKhachHang.DataSource = dsKhachHang;  
gwwKhachHang.DataMember = "KhachHang";  
conn.Close();
```

5.5 Cập nhật những thay đổi vào CSDL



❖ Cập nhật thay đổi vào CSDL

- Sau khi thay đổi được thực hiện trên dữ liệu, bạn sẽ cần ghi lại vào database. Dòng mã sau cho thấy cách dùng phương thức Update của SqlDataAdapter để cập nhật các thay đổi vào database.

```
dsKhachHang.Update(dsKhachHang, "KhachHang");
```

Phương thức Update() trên được gọi trên thể hiện của SqlDataAdapter có tham số đầu tiên là chính đối tượng gọi phương thức. Tham số thứ hai của phương thức Update() chỉ ra bảng nào trong DataSet sẽ được cập nhật. Bảng chứa một danh sách các dòng dữ liệu đã bị thay đổi và các property Insert, Update, Delete của SqlDataAdapter chứa các lệnh SQL dùng để thực hiện thay đổi database.

5.6 Bài tập ứng dụng



Khi chúng ta thao tác trên cơ sở dữ liệu như insert, delete, update, select là chúng ta đã tác động / cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu phải không?

NHẬP DANH SÁCH SINH VIÊN

Lớp:

Họ tên:

Giới tính:

Ngày sinh:

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
1	Tran Viet Khanh	☐	17/02/1979	365/16 Nguyen Thi Kieu, P.TTH, Q. 12		Web01B1

Database

DESKTOP-MTKN23D...n - dco.SINHVIEN

id	hoten	gioitinh	ngaysinh	diachi	hinhanh	malcp
1	Tran Viet Khanh	False	1979-02-17	365/16 Nguyen ...	<Binary data>	Web01B1
*	NULL	NULL	NULL	NULL	NULL	NULL

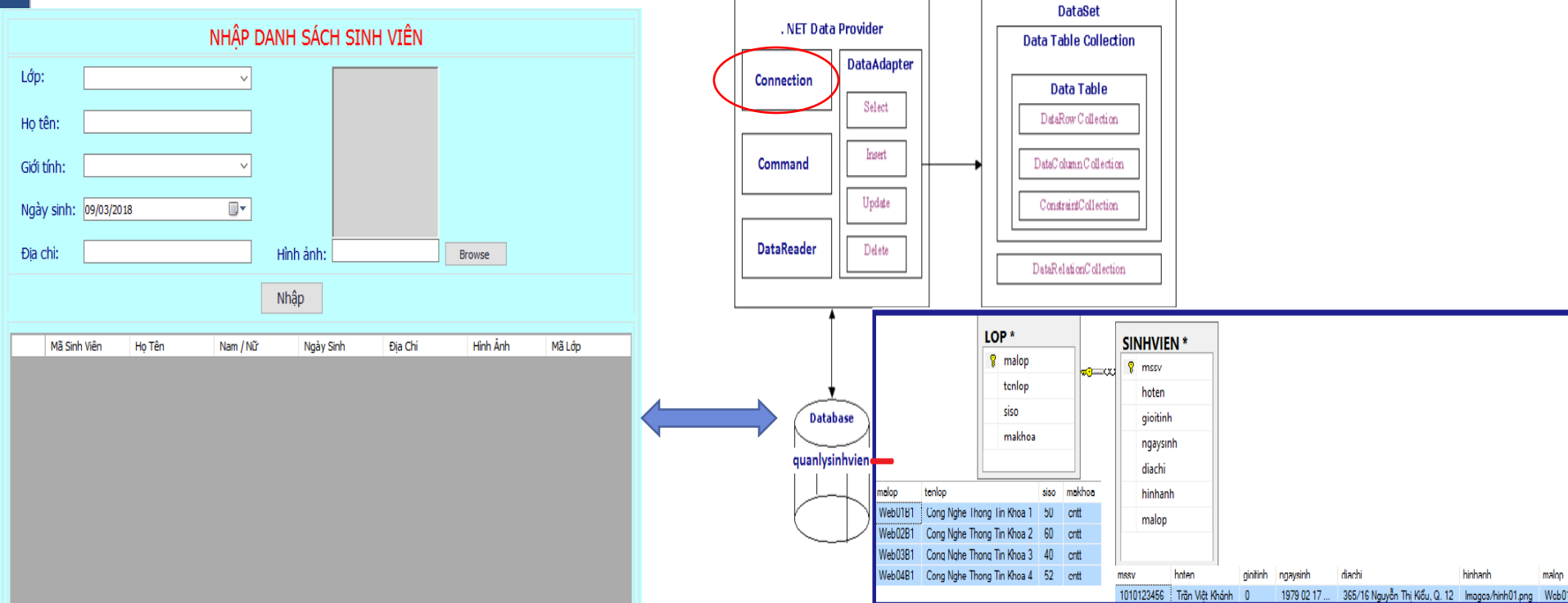
Hình 1 – Cập nhật thông tin dữ liệu vào cơ sở dữ liệu

5.6 Bài tập ứng dụng



❖ Đối tượng Connection ?

Vai trò của Connection trong .NET DataProvider là tạo kết nối giữa ứng dụng và cơ sở dữ liệu nguồn, Connection trong .NET DataProvider chỉ đóng vai trò tạo kênh thông tin giữa ứng dụng và CSDL, không thực hiện các lệnh truy xuất, cập nhật những thay đổi của dữ liệu vào trong CSDL.



Hình 2 – Tạo cầu nối giữa giao diện phần mềm với CSDL thông qua Connection

5.6 Bài tập ứng dụng



❖ Khởi tạo đối tượng Connection

- Chúng ta cần khai báo rõ đối tượng Connection để sử dụng với cơ sở dữ liệu SQLServer là SqlConnection. Đối tượng này thuộc thư viện System.Data.SqlClient.
- SqlConnection được khai báo và sử dụng như sau:

```
SqlConnection cn = new SqlConnection(string connectionString);
```

Ví dụ :

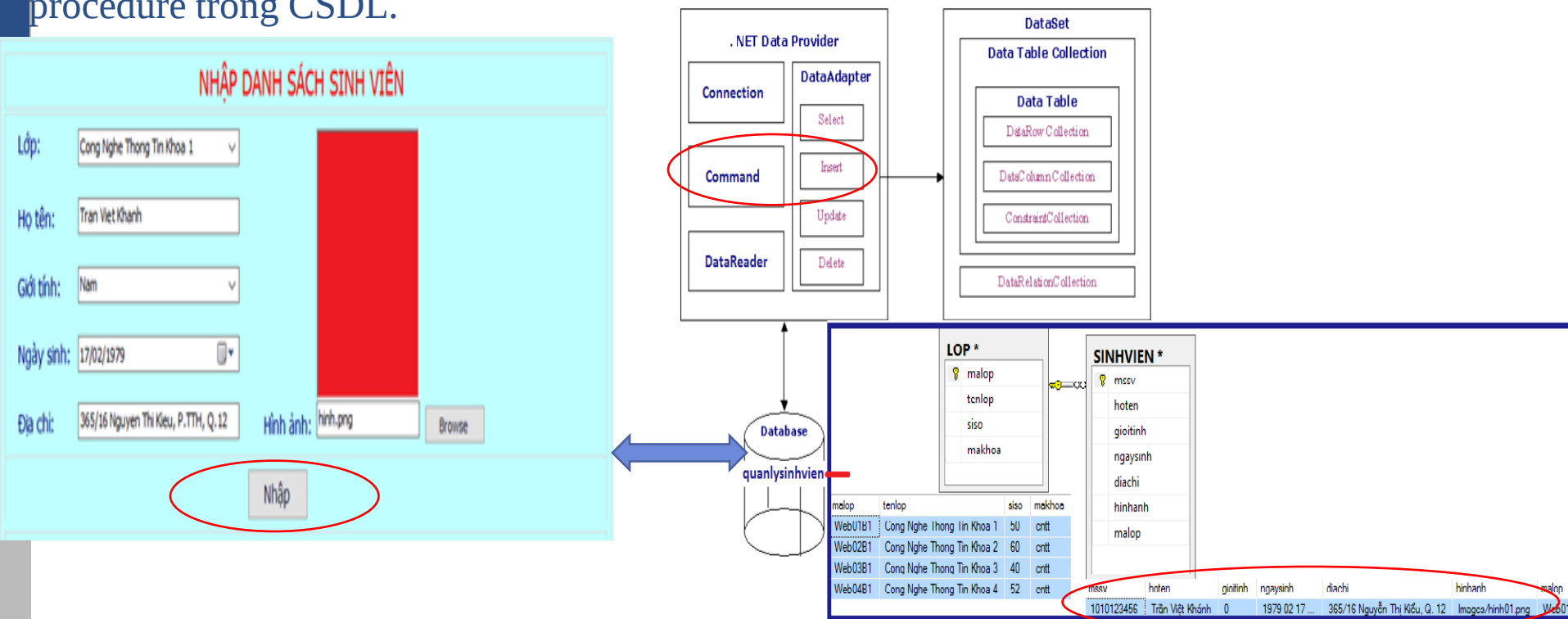
```
string ConnectionString = "Data Source=.;Initial Catalog=quanlysinhvien;Integrated Security=True";  
SqlConnection cn=new SqlConnection(ConnectionString);  
cn.Open();
```

5.6 Bài tập ứng dụng



❖ Đối tượng Command ?

■ Đối tượng Command trong .NET Framework cho phép gọi thi hành trực tiếp những câu lệnh SQL, chẳng hạn như INSERT, SELECT, UPDATE và DELETE của đối tượng DataAdapter lên cơ sở dữ liệu nguồn để thêm, chọn, cập nhật và xóa dữ liệu trên CSDL và thi hành những stored procedure trong CSDL.



Hình 3 - Sử dụng đối tượng Command (lệnh Insert) để thêm thông tin vào CSDL.

Giảng viên: Trần Việt Khánh

5.6 Bài tập ứng dụng



❖ Khai báo và sử dụng đối tượng Command ?

■ Chúng ta cần khai báo để sử dụng đối tượng Command với cơ sở dữ liệu SQLServer là SqlCommand. Đối tượng này thuộc thư viện System.Data.SqlClient.

- SqlCommand được khai báo và sử dụng như sau:
**SqlCommand cmd = new SqlCommand(string storeProcedure,
SqlConnection connection);**

Ví dụ: Thêm thông tin sinh viên vào cơ sở dữ liệu

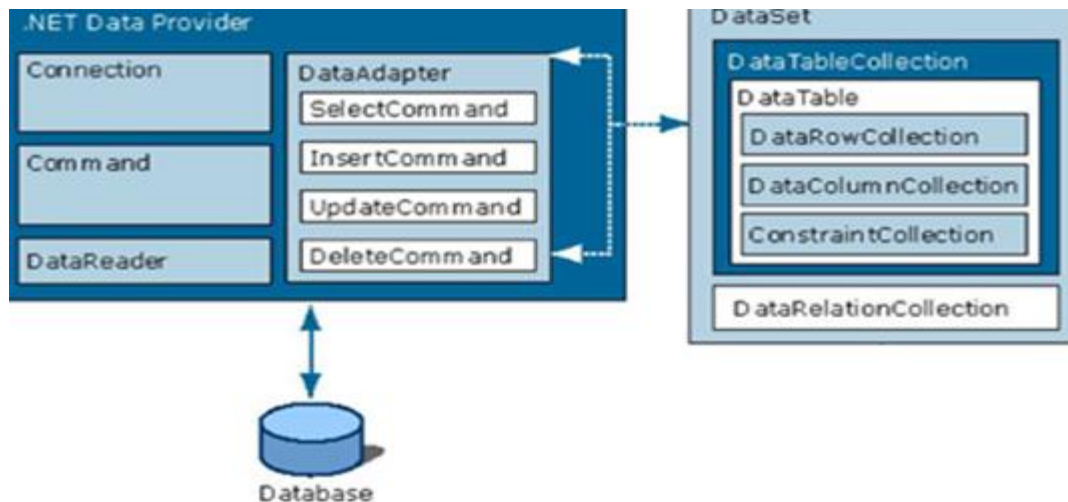
```
cmd = new SqlCommand("sp_ThemSinhVien", cn);  
cmd.CommandType = CommandType.StoredProcedure;  
cmd.Parameters.Add("@hoten", SqlDbType.NVarChar).Value = hoten;  
cmd.Parameters.Add("@gioitinh", SqlDbType.Bit).Value = gioitinh;  
cmd.Parameters.Add("@ngaysinh", SqlDbType.DateTime).Value = ngaysinh;  
cmd.Parameters.Add("@diachi", SqlDbType.NVarChar).Value = diachi;  
cmd.Parameters.Add("@hinhanh", SqlDbType.Image).Value = image;  
cmd.Parameters.Add("@malop", SqlDbType.NChar).Value = malop;  
cmd.ExecuteNonQuery();
```

5.6 Bài tập ứng dụng



❖ Đối tượng DataAdapter ?

■ Để lấy thông tin của các table(bảng) từ cơ sở dữ liệu nguồn về cho DataSet, DataTable chúng ta sử dụng một đối tượng trung gian gọi là DataAdapter. Chúng ta có thể thao tác di chuyển, thêm, sửa hủy trên các bảng của DataSet và cập nhật các thay đổi đó vào cơ sở dữ liệu nguồn thông qua các phương thức SelectCommand, InsertCommand, UpdateCommand, DeleteCommand mà đối tượng DataAdapter này cung cấp.



Hình 4 - Sử dụng DataAdapter lấy dữ liệu từ CSDL về cho DataSet, DataTable

5.6 Bài tập ứng dụng



❖ Khởi tạo đối tượng DataAdapter

- Chúng ta cần khai báo rõ DataAdapter sử dụng theo Data Provider với SQLServer là SqlDataAdapter. Lớp này thuộc thư viện System.Data.SqlClient.

SqlDataAdapter có các phương thức như sau:

- Dạng 1 : không có đối số.

```
public SqlDataAdapter();
```

Ví dụ:

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

- Dạng 2: có một đối số là sqlCommand được khai báo như một thuộc tính.

```
public SqlDataAdapter(SqlCommand selectCommand);
```

Ví dụ :

```
string strSQL="select makh, tenkh from khachhang" //nên sử dụng StoreProcedure
```

```
SqlCommand cmd=new SqlCommand(strSQL,myconnect);
```

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

5.6 Bài tập ứng dụng



❖ Khởi tạo đối tượng **DataAdapter**

- Dạng 3: có hai đối số là một `selectCommand` và một đối tượng `SqlConnection`.

```
public SqlDataAdapter(string selectCommandText, SqlConnection cn);
```

Ví dụ :

```
string strSQL="select makh, tenkh from khachhang";//nên sử dụng StoreProcedure  
SqlDataAdapter da=new SqlDataAdapter(strSQL,cn);
```

- Dạng 4: có hai đối số là một `selectCommand` và một chuỗi `connectionstring`.

```
public SqlDataAdapter(string selectCommandText, string connectionString);
```

Ví dụ :

```
string strconn="Intergrated Security=SSPI; Initial Catalog=QLBH;  
                Data Source=(local)";  
string strSQL="select makh, tenkh from khachhang";  
SqlDataAdapter da=new SqlDataAdapter(strSQL,strconn);
```

5.6 Bài tập ứng dụng



❖ Khái niệm đối tượng DataSet, DataTable, DataColumn, DataRow

- **DataSet:** là một mô hình CSDL quan hệ thu nhỏ nhằm đáp ứng các yêu cầu cần thiết của ứng dụng. DataSet chứa các bảng (DataTable), các quan hệ (DataRelation) và các ràng buộc (Constraint).

DataSet thuộc thư viện sau : **System.Data.DataSet**

Khai báo để sử dụng:

```
DataSet ds =new DataSet ();
```

Hoặc

```
DataSet ds =new DataSet (string dataSetName)
```

- **DataTable:** dữ liệu của một bảng trong cơ sở dữ liệu được lấy về và đưa vào DataTable. DataTable thuộc thư viện sau : **System.Data.DataTable**

Khai báo để sử dụng:

```
DataTable dt=new DataTable();
```

Hoặc

```
DataTable dt=new DataTable(<tên bảng>);
```

5.6 Bài tập ứng dụng



❖ Khái niệm đối tượng DataSet, DataTable, DataColumn, DataRow

- **DataColumn:** là tập hợp các cột trong cấu trúc của một bảng.

Id	hoten	gioitinh	ngaysinh	diachi	hinhanh	malop
4334	Nguyen Thi Hoa	False	17/02/1979	long an	\images\hinhanh1....	cntt11
4545	Van Xuan	False	25/03/1982	phan van tri	\images\hinhanh2....	cntt12
123456	Tran Viet Khanh	True	16/02/1986	hcm	\images\hinhanh3....	cntt10
123457	Nguyen Thai Qui	False	23/10/1990	dong thap	\images\hinhanh4....	cntt10

- **DataRow:** là tập hợp các dòng dữ liệu của một bảng

Id	hoten	gioitinh	ngaysinh	diachi	hinhanh	malop
4334	Nguyen Thi Hoa	False	17/02/1979	long an	\images\hinhanh1....	cntt11
4545	Van Xuan	False	25/03/1982	phan van tri	\images\hinhanh2....	cntt12
123456	Tran Viet Khanh	True	16/02/1986	hcm	\images\hinhanh3....	cntt10
123457	Nguyen Thai Qui	False	23/10/1990	dong thap	\images\hinhanh4....	cntt10

5.6 Bài tập ứng dụng



❖ Đưa dữ liệu vào DataSet

- **Bước 1:** Lấy dữ liệu từ nguồn cơ sở dữ liệu

```
SqlCommand cmd = new SqlCommand("sp_LayDSSinhVienTheoLop", cn);  
cmd.CommandType = CommandType.StoredProcedure;  
cmd.Parameters.Add("@malop", SqlDbType.NVarChar).Value = malop;  
SqlDataAdapter da = new SqlDataAdapter(cmd);
```

Sử dụng
StoreProcedure

- **Bước 2:** Đổ dữ liệu vào DataSet

```
DataSet ds = new DataSet();  
da.Fill(ds);
```



5.6 Bài tập ứng dụng



❖ Đưa dữ liệu vào DataTable

- **Bước 1:** Lấy dữ liệu từ nguồn cơ sở dữ liệu

```
SqlCommand cmd = new SqlCommand("sp_LayDanhSachLop",cn);  
cmd.CommandType = CommandType.StoredProcedure;  
SqlDataAdapter da = new SqlDataAdapter(cmd);
```

Sử dụng
StoreProcedure

- **Bước 2:** Đổ dữ liệu vào DataTable

```
DataTable dt = new DataTable();  
da.Fill(dt);
```



Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



Đầu tiên: sinh viên thiết kế giao diện như hình dưới đây:

Form1.cs [Design]* X Form1.cs*

Form1

Home Help

New Open Close Save Save As Find File

B I U Format

Skins Exit

Mail

Inbox Outbox Drafts Trash Organizer

Calendar Tasks

NHẬP DANH SÁCH SINH VIÊN

Lớp:

Họ tên:

Giới tính:

Ngày sinh:

Địa chỉ:

Hình ảnh: Browse

Nhập

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



❖ **Tiến hành viết code theo các bước sau:**

Bước 1: Viết code tạo kết nối giao diện phần mềm với cơ sở dữ liệu.

// Khai báo các đối tượng Connection, Command, DataAdapter để sử dụng

```
SqlConnection cn;  
SqlDataAdapter da;  
SqlCommand cmd;
```

```
string ConnectionString = "Data Source=.;Initial Catalog=quanlysinhvien;Integrated Security=True";
```

```
public void Connect()  
{
```

```
    try  
    {
```

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

```
    }
```

```
    catch (Exception sqllex)
```

```
    {
```

```
        throw sqllex;
```

```
    }
```

```
}  
public void Disconnect()  
{
```

```
    cn.Close();
```

```
}
```



Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



❖ **Kết thúc bước 1 ta được kết quả như hình**

NHẬP DANH SÁCH SINH VIÊN

Lớp:

Họ tên:

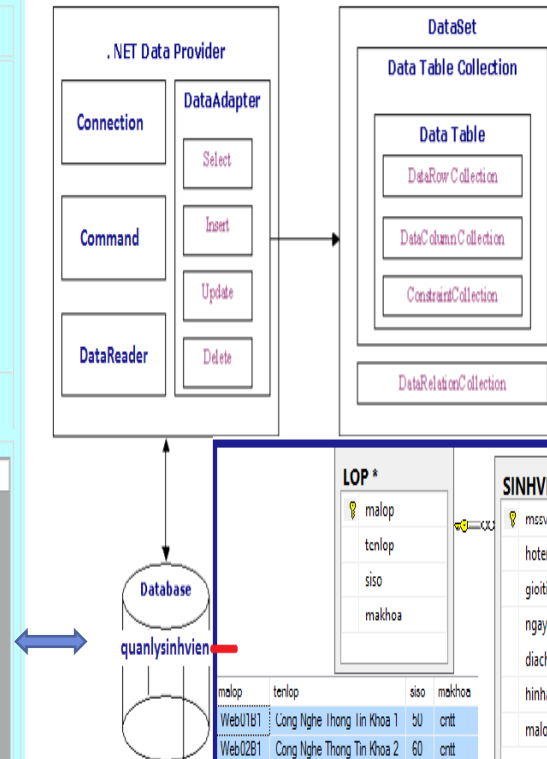
Giới tính:

Ngày sinh: 09/03/2018

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------



LOP *				SINHVIEN *						
malop	tenlop	siso	makhoe	mssv	hoten	gioitinh	ngaysinh	diachi	hinhanh	malop
Web0161	Cong Nghe Thong Tin Khoa 1	50	critt							
Web0261	Cong Nghe Thong Tin Khoa 2	60	critt							
Web0361	Cong Nghe Thong Tin Khoa 3	40	critt							
Web0461	Cong Nghe Thong Tin Khoa 4	52	critt							

1010123456	Trần Việt Khánh	0	1979 02 17 ...	365/16	Nguyễn Thị Kiều, Q. 12	Images/hinh01.png	Web01
------------	-----------------	---	----------------	--------	------------------------	-------------------	-------

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



Bước 2: Viết code lấy thông tin bảng lớp từ CSDL load vào Control lớp trên giao diện phần mềm.

```
public Form1()
{
    LoadLop();
}

private void LoadLop()
{
    Connect();
    cmd = new SqlCommand("sp_LayThongTinLop", cn);
    cmd.CommandType = CommandType.StoredProcedure;
    da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    cboLop.DisplayMember = "tenlop";
    cboLop.ValueMember = "malop";
    cboLop.DataSource = dt;
    Disconnect();
}
```



Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.

❖ Kết thúc bước 2 ta được kết quả như hình

NHẬP DANH SÁCH SINH VIÊN

Lớp:

Họ tên:

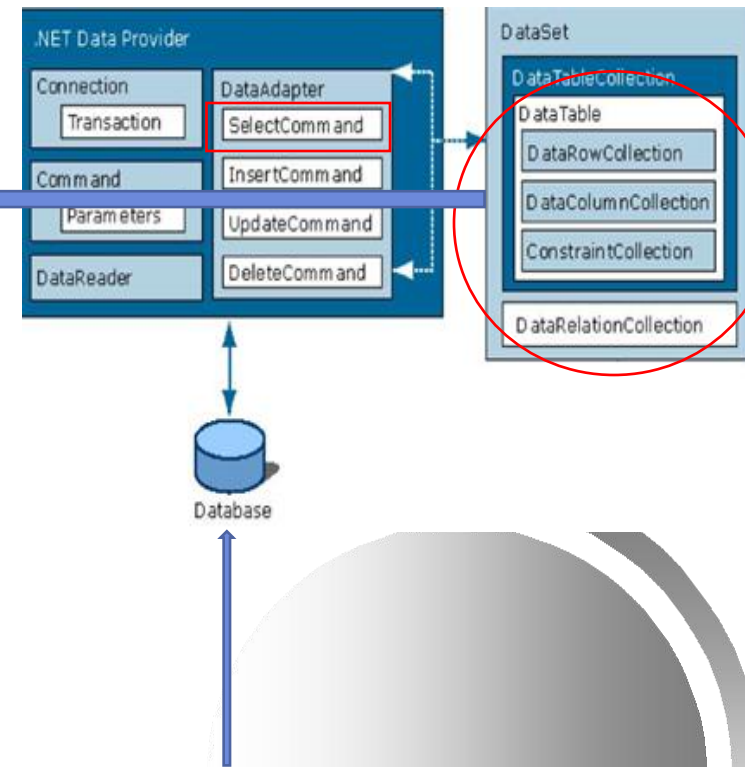
Giới tính:

Ngày sinh:

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
--------------	--------	----------	-----------	---------	----------	--------



DESKTOP-MTKN23D....nhvien - dbo.LOP

	malop	tenlop	siso	makhhoa
	Web01B1	Cong Nghe Thong Tin Khoa 1	50	cntt
	Web02B1	Cong Nghe Thong Tin Khoa 2	60	cntt
	Web03B1	Cong Nghe Thong Tin Khoa 3	40	cntt
	Web04B1	Cong Nghe Thong Tin Khoa 4	52	cntt

Giảng viên: Trần Việt Khánh

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



Bước 3: Viết code lấy thông tin dữ liệu trên Form và lưu hình ảnh vào thư mục Images.

```
private void LayDuLieuTuForm()
{
    hoten = txtHoTen.Text;
    gioitinh = cboGioiTinh.SelectedIndex;
    ngaysinh = DateTime.Parse(dtNgaySinh.Text);
    diachi = txtDiaChi.Text;
    malop = cboLop.SelectedValue.ToString() ;
}
private void LuuHinh()
{
    //saveFile.FileName = txtHinhAnh.Text;
    string duongDanHinh = Application.StartupPath + @"\Images\";
    if (openFile.FileName != "")
    {
        switch (openFile.FilterIndex)
        {
            case 1: // nếu là jpg
                picHinhAnh.Image.Save(System.IO.Directory.GetCurrentDirectory()
                    + @"\Images\" + txtHinhAnh.Text,
```

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



```
        System.Drawing.Imaging.ImageFormat.Jpeg);  
        break;  
    case 2:  
        picHinhAnh.Image.Save(System.IO.Directory.GetCurrentDirectory()  
+@"\Images\" + txtHinhAnh.Text, System.Drawing.Imaging.ImageFormat.Bmp);  
        break;  
    case 3:  
        picHinhAnh.Image.Save(System.IO.Directory.GetCurrentDirectory() +  
@"\Images\" + txtHinhAnh.Text, System.Drawing.Imaging.ImageFormat.Gif);  
        break;  
    case 4:  
        picHinhAnh.Image.Save(System.IO.Directory.GetCurrentDirectory() +  
@"\Images\" + txtHinhAnh.Text, System.Drawing.Imaging.ImageFormat.Png);  
        break;  
    }  
}  
}
```

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



```
private void btnBrowse_Click(object sender, EventArgs e)
{
    openFile.InitialDirectory = @"C:";
    openFile.FileName = "";
    openFile.Filter = "All files (*.*)|*.*";
    //openFile.FilterIndex = 1;
    openFile.RestoreDirectory = true;
    openFile.ShowDialog();
    if (openFile.FileName != "")
    {
        txtHinhAnh.Text =
            openFile.SafeFileName.Substring(openFile.SafeFileName.LastIndexOf(@"\") + 1);
        picHinhAnh.Image = Image.FromFile(openFile.FileName);

        //Read jpg into file stream, and from there into Byte array.
        FileStream fsFile = new FileStream(openFile.FileName, FileMode.Open,
            FileAccess.Read);
        image = new Byte[fsFile.Length];
        fsFile.Read(image, 0, image.Length);
        fsFile.Close();
    }
}
```

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



❖ **Kết thúc bước 3 thư mục Images của ứng dụng phần mềm có các hình ảnh được lưu như hình bên dưới:**



Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.



Bước 4: Viết code cho nút Nhập để cập nhật thông tin dữ liệu vào trong cơ sở dữ liệu.

```
private void btnNhap_Click(object sender, EventArgs e)
{
    LayDuLieuTuForm();
    LuuHinh();
    Connect();
    cmd = new SqlCommand("sp_ThemSinhVien", cn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@hoten", SqlDbType.NVarChar).Value = hoten;
    cmd.Parameters.Add("@gioitinh", SqlDbType.Bit).Value = gioitinh;
    cmd.Parameters.Add("@ngaysinh", SqlDbType.DateTime).Value = ngaysinh;
    cmd.Parameters.Add("@diachi", SqlDbType.NVarChar).Value = diachi;
    cmd.Parameters.Add("@hinhanh", SqlDbType.Image).Value = image;
    cmd.Parameters.Add("@malop", SqlDbType.NChar).Value = malop;

    cmd.ExecuteNonQuery();
    Disconnect();
    LoadSinhVienLenLuoi(malop);
}
```

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.

❖ Kết thúc bước 4 ta được kết quả như hình:

NHẬP DANH SÁCH SINH VIÊN

Lớp:

Họ tên:

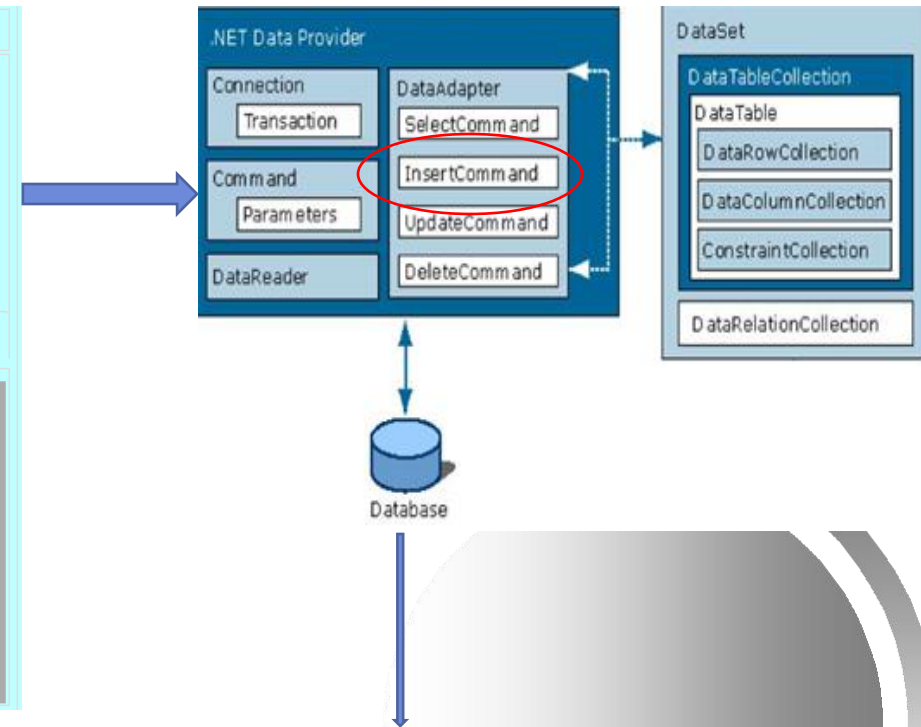
Giới tính:

Ngày sinh:

Địa chỉ:

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp



DESKTOP-MTKN23D....n - dbo.SINHVIEN

	id	hoten	gioitinh	ngaysinh	diachi	hinhanh	malop
▶	1	Tran Viet Khanh	False	1979-02-17	365/16 Nguyen ...	<Binary data>	Web01B1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.

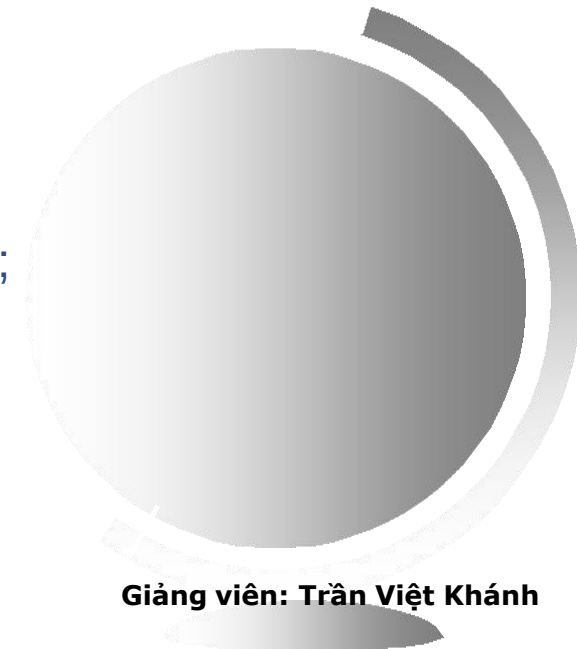


Bước 5: Viết code để lấy thông tin sinh viên từ CSDL load lên DataGridView trên giao diện phần mềm.

```
private void LoadSinhVienLenLuoi(string strMaLop)
{
    Connect();
    cmd = new SqlCommand("sp_LayThongTinSinhVienTheLop",cn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@malop", SqlDbType.NChar).Value = strMaLop;
    da = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    da.Fill(ds);

    dgridSinhVien.DataSource = ds.Tables[0].DefaultView;

    Disconnect();
}
```



Hướng dẫn các bước lập trình cập nhật những thay đổi của dữ liệu vào trong cơ sở dữ liệu.

❖ Kết thúc bước 5 ta được kết quả như hình:

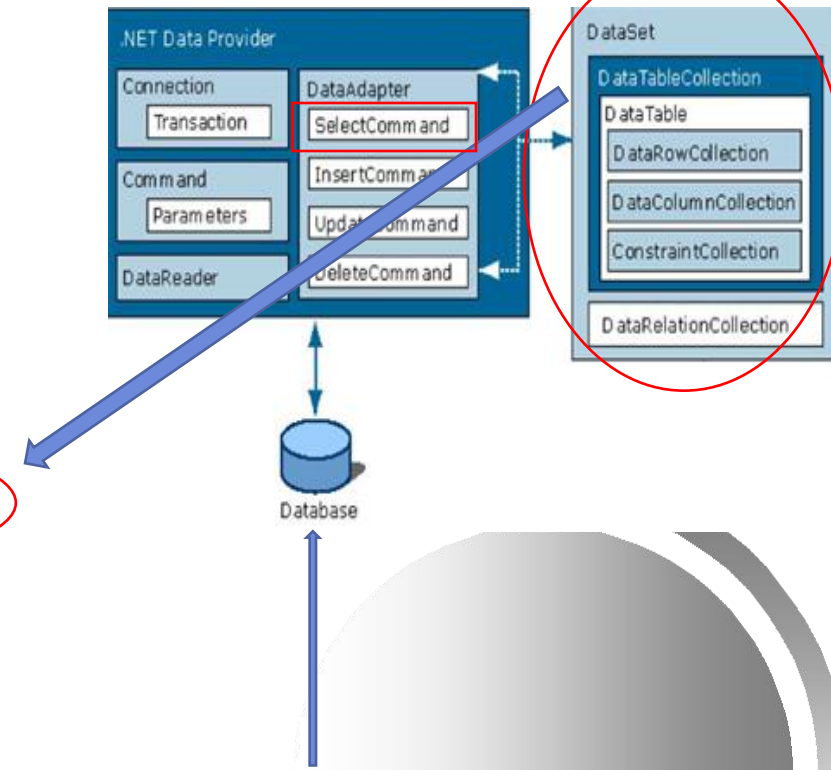


NHẬP DANH SÁCH SINH VIÊN

Lớp: Cong Nghe Thong Tin Khoa 1
Họ tên: Tran Viet Khanh
Giới tính: Nam
Ngày sinh: 17/02/1979
Địa chỉ: 365/16 Nguyen Thi Kieu, P.TTH, Q. 12

Hình ảnh:

Mã Sinh Viên	Họ Tên	Nam / Nữ	Ngày Sinh	Địa Chỉ	Hình Ảnh	Mã Lớp
1	Tran Viet Khanh	☐	17/02/1979	365/16 Nguyen Thi Kieu, P.TTH, Q. 12		Web01B1



DESKTOP-MTKN23D....n - dbo.SINHVIEN

	id	hoten	gioitinh	ngaysinh	diachi	hinhanh	malop
▶	1	Tran Viet Khanh	False	1979-02-17	365/16 Nguyen ...	<Binary data>	Web01B1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Bài tập về nhà



Sinh viên chọn thực hiện xây dựng ứng dụng phần mềm một trong ba bài tập sau:

- Quản lý Học sinh
- Quản lý Khách sạn
- Quản lý thư viện
- Chú ý: Phần mô tả của từng bài tập đã được nêu trong danh sách đề tài.

Chương 6. CRYSTAL REPORT VÀ LINKQ



Nội dung



-  **6.1. Giới thiệu Crystal Report**
-  **6.2. Giới thiệu LINQ**
-  **6.3. Kiến trúc LINQ**
-  **6.4. Sử dụng LINQ với Object**
-  **6.5. Sử dụng LINQ với ADO.NET**

6.1 Giới thiệu Crystal Report



- ❖ Báo cáo là phần hiển thị của dữ liệu theo một khuôn dạng nhất định.

Ví dụ

Dữ liệu dạng bảng

MSSV	TenSV	NamSinh	MaLop
D64K301001	An	1990	K301
D64K301002	Bình	1991	K301
D64K301003	Dũng	1989	K301

Báo biểu

MSSV	HoTen	NamSinh	MaLop
D64K301001	An	1990	K301
D64K301002	Bình	1991	K301
D64K301003	Dũng	1989	K301

Danh sách có 03 sinh viên

6.1 Giới thiệu Crystal Report

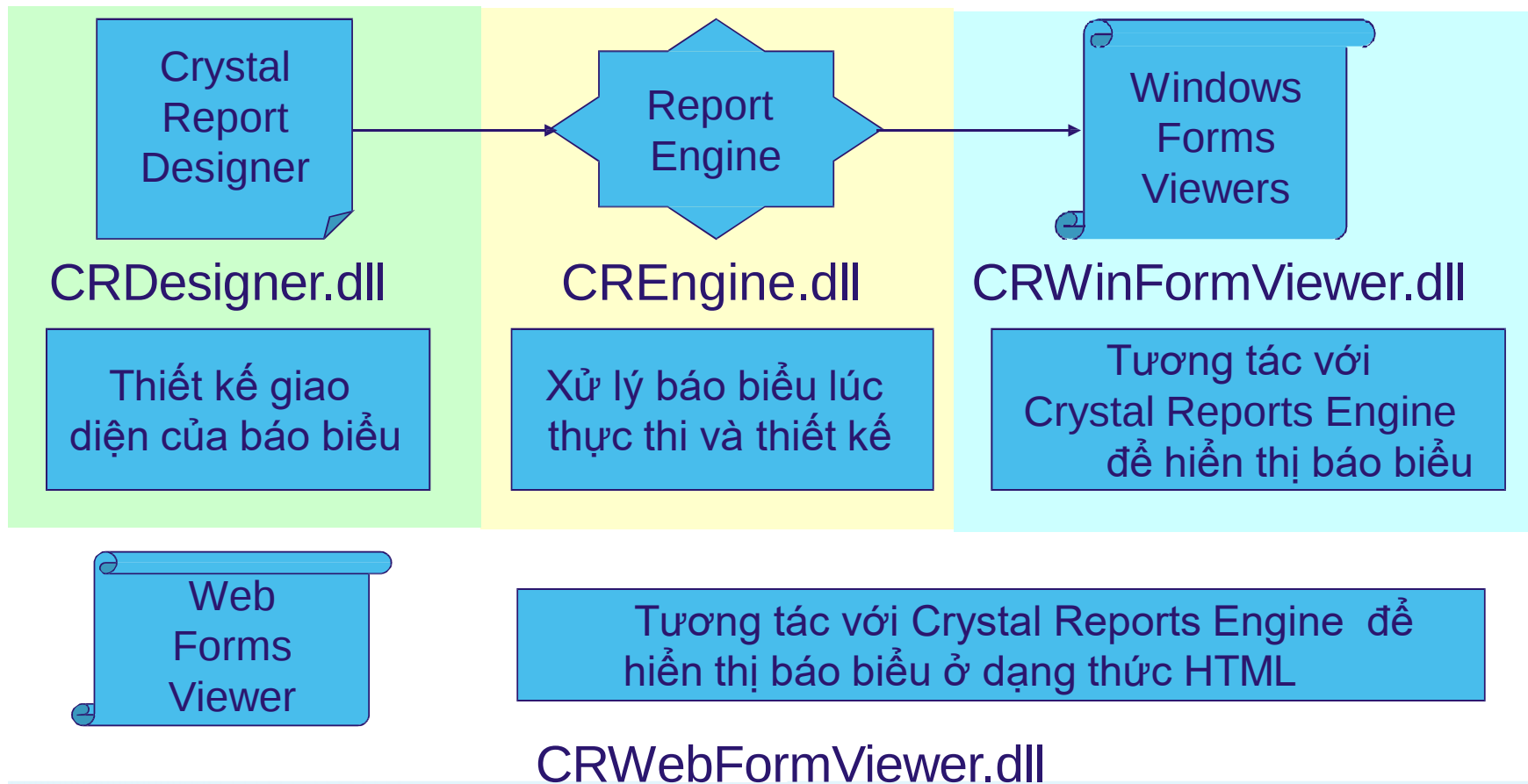


Crystal Reports là công cụ để tạo báo biểu trong Visual Studio.NET. Dễ dàng tạo các báo biểu phức tạp.

6.1 Giới thiệu Crystal Report



Kiến trúc Crystal Reports





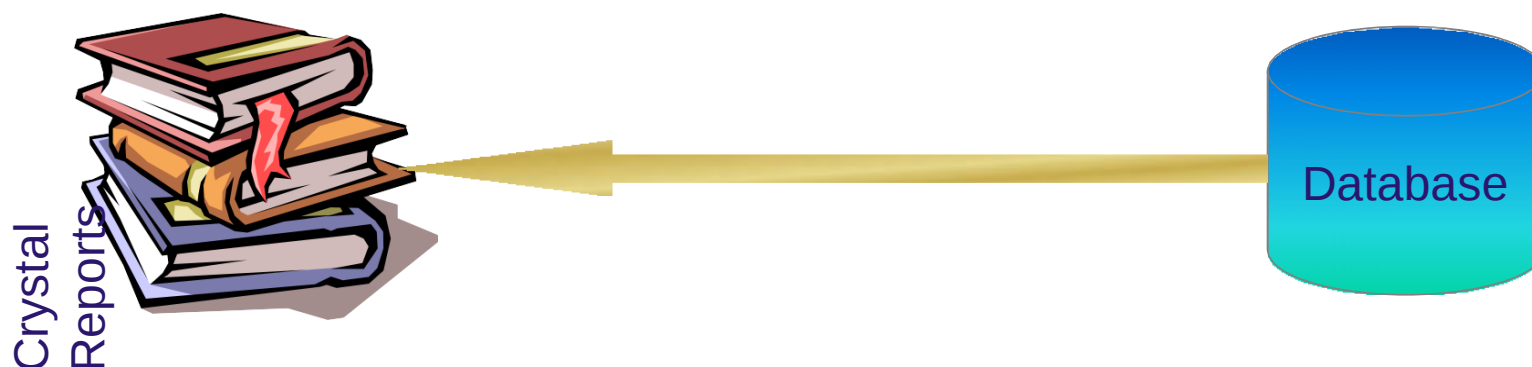
Truy cập dữ liệu trong Crystal Reports

- **Crystal Report sử dụng các trình điều khiển dữ liệu để nối kết với cơ sở dữ liệu.**
- **2 mô hình: Push & Pull.**

6.1 Giới thiệu Crystal Report



Mô hình Pull

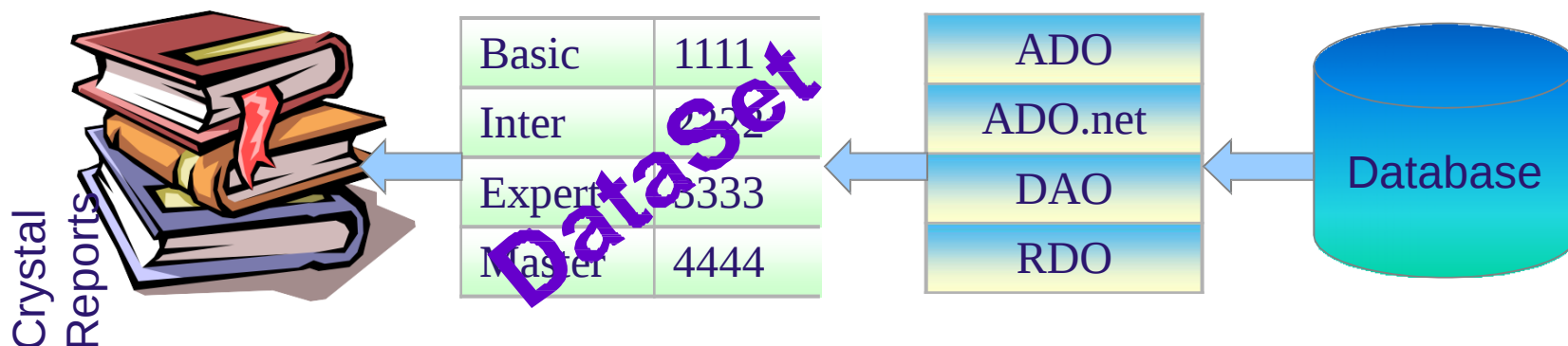


- ❖ **Crystal Report thực hiện việc nối kết đến cơ sở dữ liệu và thực thi các câu truy vấn SQL.**

6.1 Giới thiệu Crystal Report



Push Model



- **Viết mã lệnh tạo DataSet và gọi đối tượng đến báo biểu.**
- **Cho phép chia sẻ đối tượng nối kết dữ liệu với ứng dụng**

6.1 Giới thiệu Crystal Report

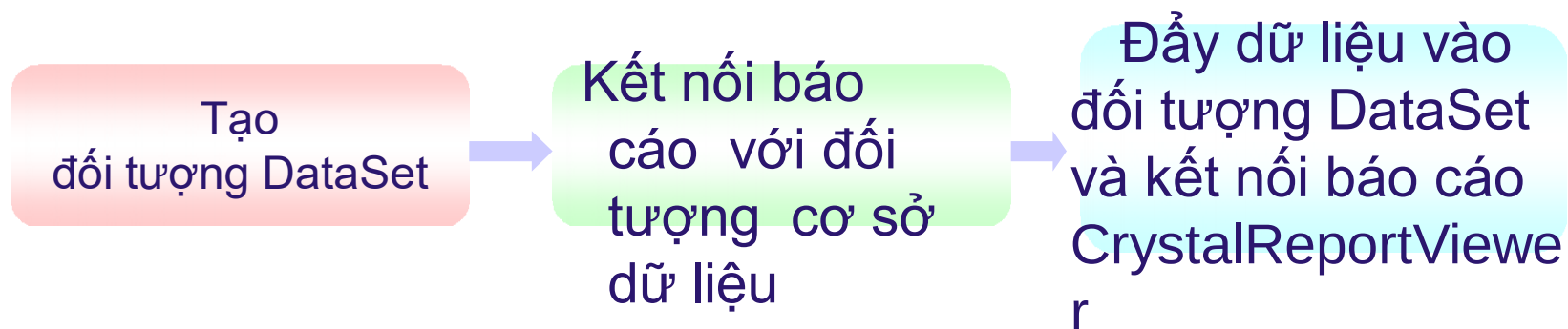


Các nguồn dữ liệu được hỗ trợ trong Crystal Reports

- Cơ sở dữ liệu với trình cung cấp ODBC
 - Cơ sở dữ liệu với trình cung cấp OLEDB
 - ADO.net datasets
 - ADO recordsets
 - DAO recordsets
 - RDO recordsets
 - Cơ sở dữ liệu Microsoft Access
 - Bảng tính Microsoft Excel



Các bước tạo báo cáo bằng Crystal Reports



6.1 Giới thiệu Crystal Report



Tạo đối tượng DataSet

1. Trong Solution Explorer panel, nhấp phải trên tên project, chỉ vào Add rồi nhấp Add New Item.
2. Trong vùng Categories của hộp thoại Add New Item chọn Data.
3. Trong vùng Templates chọn DataSet
4. Nhập tên DataSet vào hộp Name rồi nhấp Add
5. Nhấp Server Explore để mở Server Explore panel ta nhấp Connect to DataBase, chọn Server name và database name, Nhấp OK
6. Chọn các table, view, store procedure muốn đưa vào dataset rồi kéo vào cửa sổ thiết kế dataset

6.1 Giới thiệu Crystal Report



Đẩy dữ liệu vào đối tượng DataSet và kết nối báo cáo CrystalReportViewer

1. Add thêm 1 form mới vào project
2. Đặt lên form một CrystalReportViewer và đặt thuộc tính DisplayGroupTree là False
3. Trong sự kiện rptViewer_Load của ta viết đoạn code sau đây

6.2 Giới thiệu LINQ

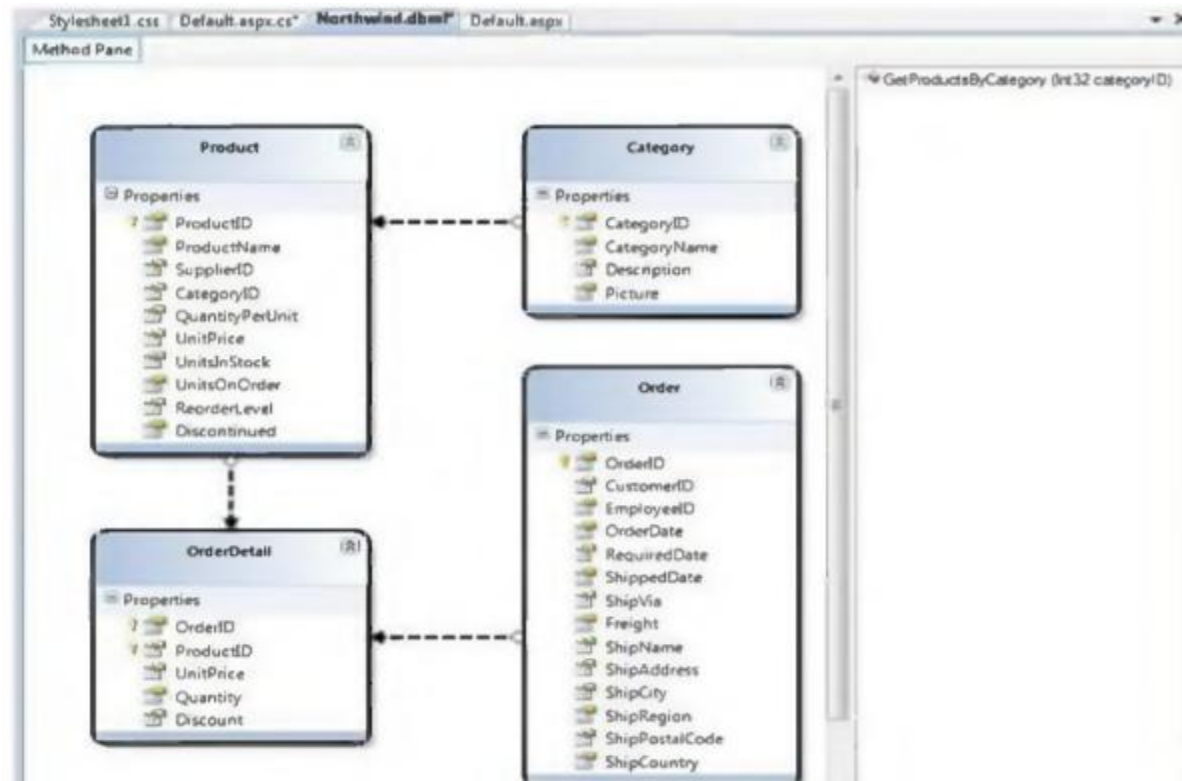


- ❖ **LINQ: là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework bản "Orcas" (nay là .NET 3.5), nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu (CSDL) dùng LINQ, cũng như cập nhật/thêm/xóa dữ liệu từ đó.**
- ❖ **LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP). Nó cũng cung cấp một cách dễ dàng để thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu của bạn.**

6.3 Kiến trúc LINKQ



- ❖ Visual Studio “Orcas” đã tích hợp thêm một trình thiết kế LINQ như một công cụ dễ dàng cho việc mô hình hóa một cách trực quan các CSDL dùng LINQ to SQL.
- ❖ Bằng cách dùng trình thiết kế LINQ to SQL, có thể dễ dàng tạo một mô hình cho CSDL mẫu “Northwind” giống như dưới đây:



6.3 Kiến trúc LINKQ

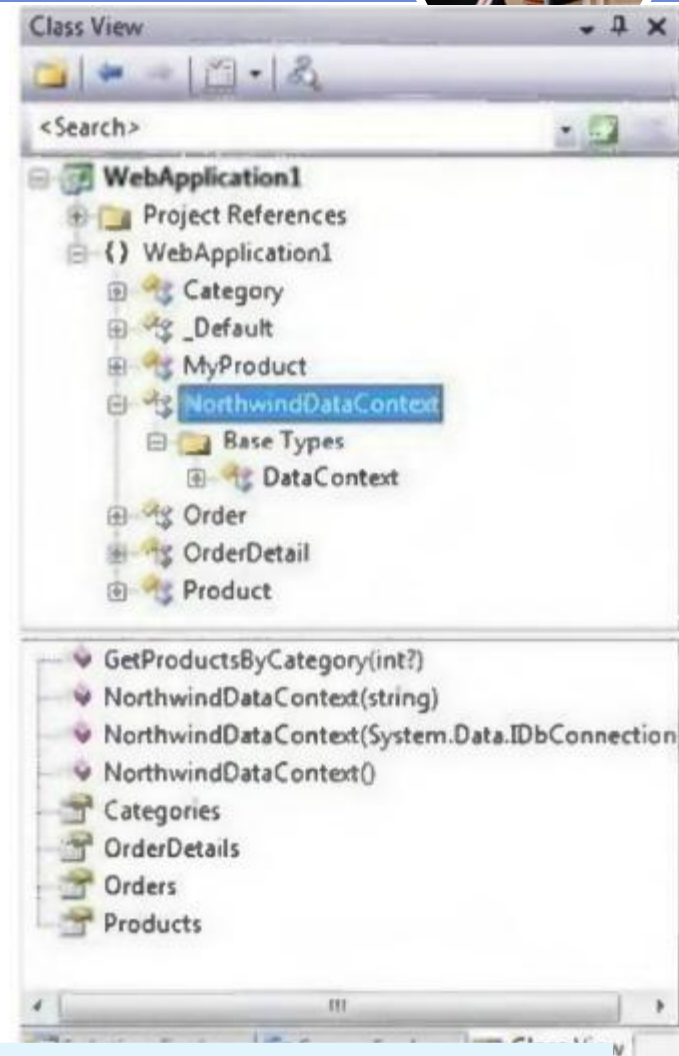


- ❖ Mô hình LINQ to SQL ở trên định nghĩa bốn lớp thực thể: **Product**, **Category**, **Order** và **OrderDetail**. Các thuộc tính của mỗi lớp ánh xạ vào các cột của bảng tương ứng trong CSDL. Mỗi instance của một lớp biểu diễn một dòng trong bảng dữ liệu.
- ❖ Các mũi tên giữa bốn lớp thực thể trên biểu diễn quan hệ giữa các thực thể khác nhau, chúng được tạo ra dựa trên các mối quan hệ primary-key/foreign-key trong CSDL. Hướng của mũi tên chỉ ra mối quan hệ là một - một hay một - nhiều. Các thuộc tính tương ứng sẽ được thêm vào các lớp thực thể trong các trường hợp này. Lấy ví dụ, lớp **Category** ở trên có một mối quan hệ một nhiều với lớp **Product**, điều này có nghĩa nó sẽ có một thuộc tính "Categories" là một tập hợp các đối tượng **Product** trong **Category** này. Lớp **Product** cũng sẽ có một thuộc tính "Category" chỉ đến đối tượng "Category" chứa **Product** này bên trong.
- ❖ Bảng các phương thức bên tay phải bên trong trình thiết kế LINQ to SQL ở trên chứa một danh sách các SP để tương tác với mô hình dữ liệu của chúng ta. Trong ví dụ trên tôi đã thêm một thủ tục có tên "GetProductsByCategory". Nó nhận vào một categoryID và trả về một chuỗi các **Product**. Chúng ta sẽ xem bằng cách nào có thể gọi được thủ tục này trong một đoạn code dưới đây.

6.3 Kiến trúc LINKQ



- ❖ **Tìm hiểu lớp DataContext**
- ❖ **Khi bấm nút “Save” bên trong màn hình thiết kế LINQ to SQL, Visual Studio sẽ lưu các lớp .NET biểu diễn các thực thể và quan hệ bên trong CSDL mà chúng ta vừa mô hình hóa. Cứ mỗi một file LINQ to SQL chúng ta thêm vào solution, một lớp DataContext sẽ được tạo ra, nó sẽ được dùng khi cần truy vấn hay cập nhật lại các thay đổi. Lớp DataContext được tạo sẽ có các thuộc tính để biểu diễn mỗi bảng được mô hình hóa từ CSDL, cũng như các phương thức cho mỗi SP mà chúng ta đã thêm vào.**
- ❖ **Lấy ví dụ, hình bên là lớp NorthwindDataContext được sinh ra dựa trên mô hình chúng ta tạo ra:**



6.3 Kiến trúc LINKQ



❖ Lấy các Product từ CSDL.

- Đoạn lệnh dưới đây dùng cú pháp LINQ để lấy về một tập Enumerable các đối tượng Product. Các sản phẩm được lấy ra phải thuộc phân loại "Beverages":

```
NorthwindDataContext db = new NorthwindDataContext();  
var products = from p in db.Products  
                where p.category.categoryName ==  
                "Beverages"  
                select p;
```

❖ Cập nhật lại giá tiền và lưu lại CSDL

```
NorthwindDataContext db = new NorthwindDataContext();  
Product product = db.Products.single (p =>  
p.productName == "Toy 1");  
product.UnitPrice = 99; product.UnitsInStock =  
5;  
db.SubmitChanges();
```

6.3 Kiến trúc LINKQ



- ❖ **Chèn thêm một phân loại mới và hai sản phẩm vào CSDL**
Đoạn mã dưới đây biểu diễn cách tạo một phân loại mới, và tạo hai sản phẩm mới và đưa chúng vào trong phân loại đã tạo. Cả ba sau đó sẽ được đưa vào cơ sở dữ liệu.

Chú ý rằng không cần phải tự quản lý các mối quan hệ primary key/foreign key, thêm các đối tượng Product vào tập hợp Products của đối tượng category, và rồi thêm đối tượng category vào tập hợp Categories của DataContext, LINQ to SQL sẽ biết cách thiết lập các giá trị primary key/foreign key một cách thích hợp.

```
NorthwindDataContext db = new NorthwindDataContext();  
// Create new category and Products  
category category = new categoryQ ; category.categoryName = "Scott'S Toys";  
product product1 = new ProductO; product1.ProductName = "Toy 1";  
Product product2 = new ProductO; product2.ProductName = "Toy 2" ;  
// Associate Products with Category  
category.products.Add(product1); category.Products.Add(product2);  
// Add category to database and save changes  
db.categories.Add(category);  
db.SubmitChanges();
```

6.3 Kiến trúc LINKQ



❖ Xóa tất cả các sản phẩm Toy khỏi CSDL:

```
NorthwindDataContext db = new NorthwindDataContext();  
var toyproducts = from p in db.Products where p.  
ProductName.contains("Toy") se le ct p;  
db.Products. RemoveAll(toyproducts);  
db. submitchangesQ;
```

❖ Gọi một thủ tục: Đoạn mã dưới đây biểu diễn cách lấy thực thể Product mà không dùng cú pháp của LINQ, mà gọi đến thủ tục "GetProductsByCategory". Nhớ rằng một khi đã lấy về kết quả, phải cập nhật/xóa gọi hàm db.SubmitChanges() để cập nhật các thay đổi trở lại vào CSDL

```
NorthwindDataContext db = new NorthwindDataContext();  
var products = db.GetProductsByCategory(5);  
foreach (Product product in products)  
{  
    product.  
}
```



145

6.3 Kiến trúc LINKQ



❖ Lấy các sản phẩm và phân trang

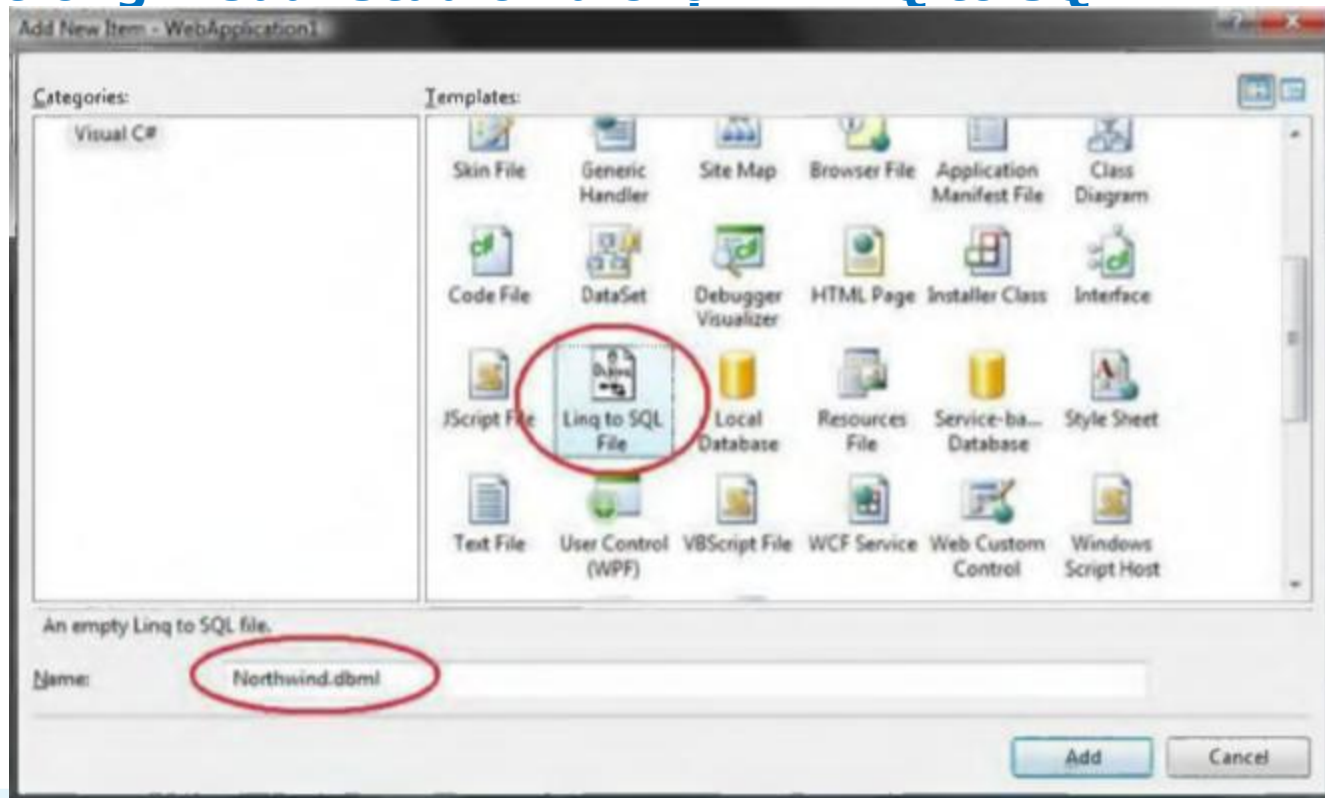
Đoạn mã dưới đây biểu diễn cách phân trang trên server như một phần của câu truy vấn LINQ. Bằng cách dùng các toán tử Skip() và Take(), chúng ta sẽ chỉ trả về 10 dòng từ CSDL - bắt đầu từ dòng 200.

```
NorthwindDataContext db = new NorthwindDataContext();  
var products = (from p in db.Products  
                 where p.category.categoryName.StartsWith("C")  
                 select p).Skip(200).Take(10);
```

6.4 Sử dụng LINKQ với Object



- ❖ Có thể thêm một mô hình dữ liệu LINQ to SQL và một dự án ASP.NET, Class Library hay Windows bằng cách dùng tùy chọn “Add New Item” bên trong Visual Studio và chọn “LINQ to SQL”:



6.4 Sử dụng LINKQ với Object



- ❖ Dùng các Stored Procedure: LINQ to SQL cho phép mô hình hóa các thủ tục lưu trữ như là các phương thức trong lớp DataContext. Ví dụ, định nghĩa một thủ tục đơn giản có tên SPROC để lấy về các thông tin sản phẩm dựa trên một Category ID:

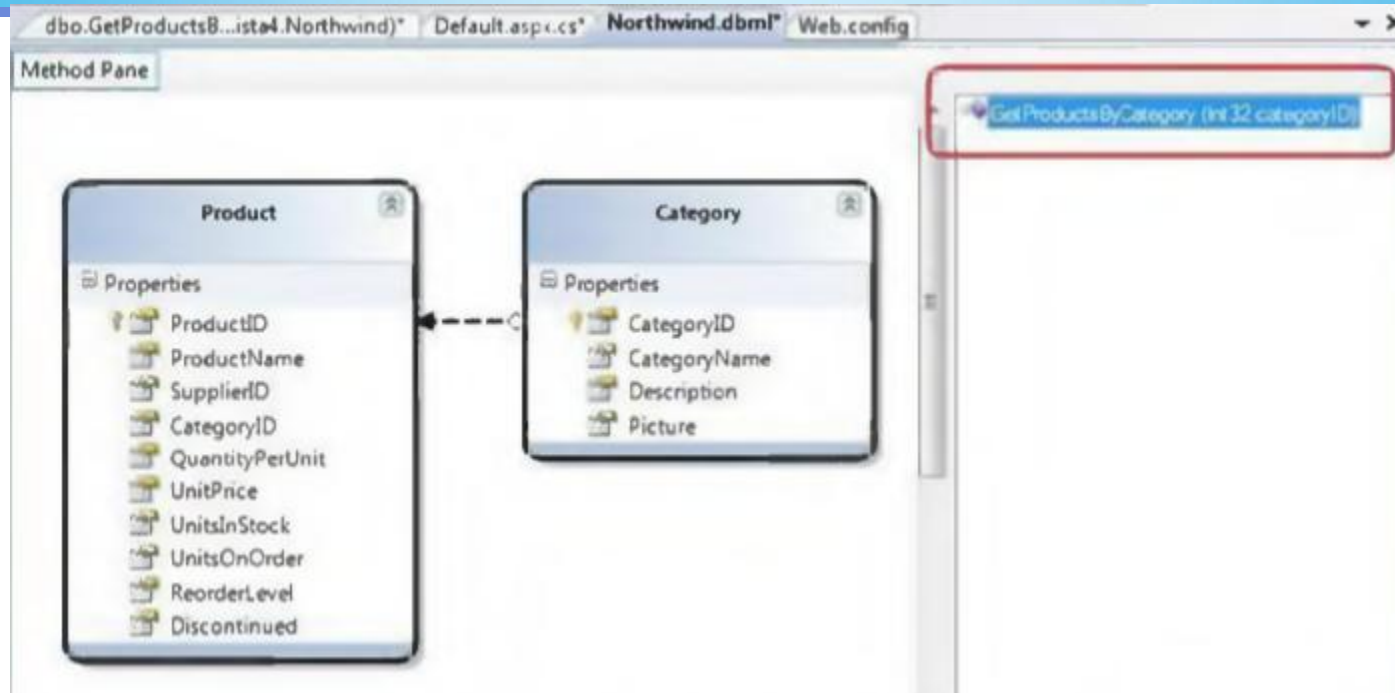
```
ALTER PROCEDURE dbo. G etProductsBycategory  
( @ categoryID int )
```

```
AS
```

```
SELECT * from Products where CategoryID=@ categoryID
```

Có thể dùng Server Explorer trong VS để kéo/thả thủ tục SPROC lên trên cửa sổ soạn thảo LINQ to SQL để có thể thêm một phương thức cho phép gọi SPROC. Nếu tôi thả SPROC lên trên thực thể "Product", LINQ to SQL designer sẽ khai báo SPROC để trả về một tập kết quả có kiểu **IEnumerable<Product>**:

6.4 Sử dụng LINKQ với Object



- ❖ Sau đó có thể dùng cú pháp LINQ to SQL hay gọi thẳng phương thức ở trên để lấy về các thực thể từ CSDL:

6.4 Sử dụng LINKQ với Object

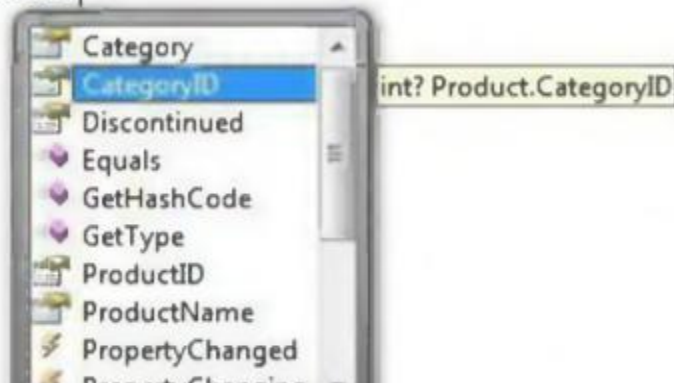


```
NorthwindDataContext db = new NorthwindDataContext();

// Retrieve products based on an adhoc query
IEnumerable<Product> products = from p in db.Products
                                where p.CategoryID == 1
                                select p;

// Retrieve products instead using a SPROC method
products = db.GetProductsByCategory(1);

// Iterate over results
foreach (Product product in products)
{
    product.|
```



6.5 Sử dụng LINKQ với ADO.NET



Đẩy dữ liệu vào đối tượng DataSet và kết nối báo cáo CrystalReportViewer

- 1. Add thêm 1 form mới vào project**
- 2. Đặt lên form một CrystalReportViewer và đặt thuộc tính DisplayGroupTree là False**
- 3. Trong sự kiện rptViewer_Load của ta viết đoạn code sau đây**

Chương 7. Mô hình MVC





7.1 Giới thiệu mô hình MVC

7.2 Kiến trúc mô hình MVC

7.3 Lớp Models

7.4 Lớp Controls

7.5 Lớp Views

7.1 Giới thiệu mô hình MVC



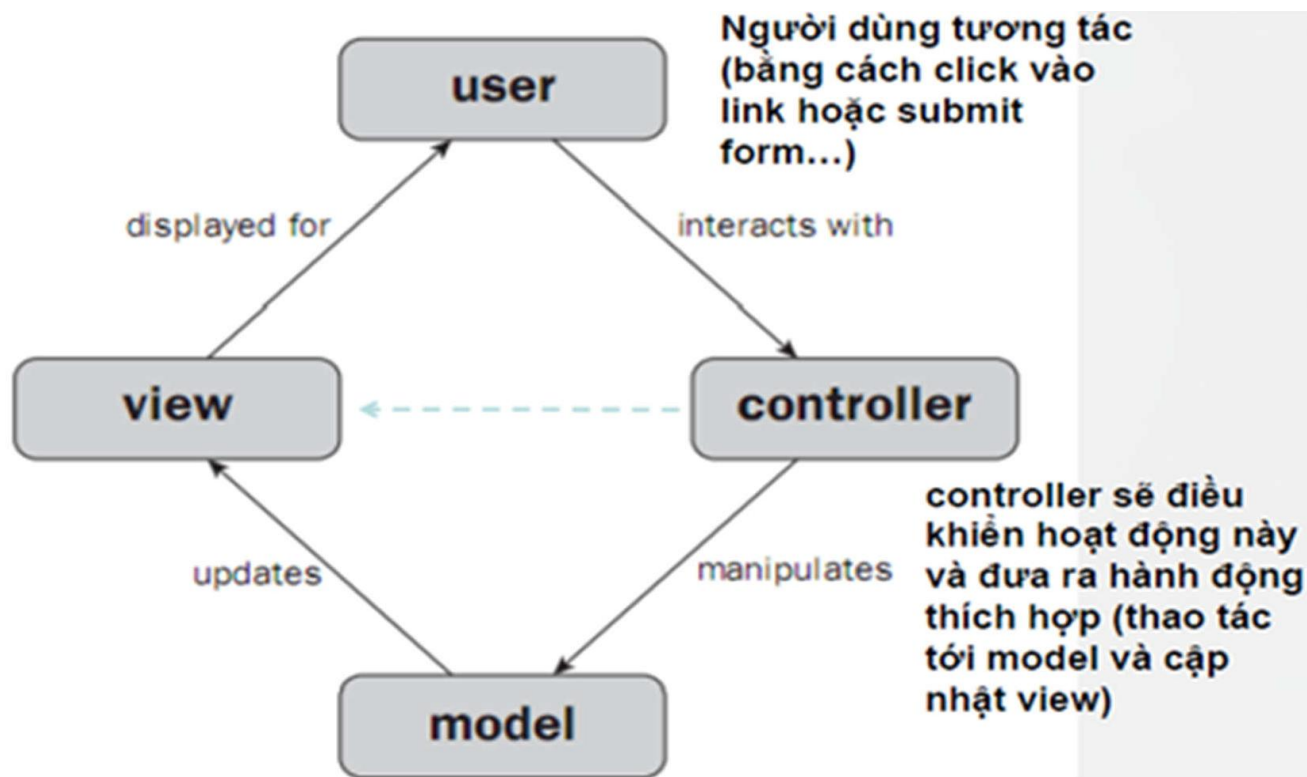
MVC (Model – View - Controller): cho phép developer có thể chia nhỏ code của mình ra thành 3 phần:

❖ **Model**: : Được giao nhiệm vụ cung cấp dữ liệu cho cơ sở dữ liệu và lưu dữ liệu vào các kho chứa dữ liệu. Là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (sqlserver, mysql,...), bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu...

❖ **View**: là nơi chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh... nó đảm nhận nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.

❖ **Controller**: là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ khác nhau giúp lấy đúng dữ liệu thông tin cần thiết do lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng thông qua lớp View.

7.2 Kiến trúc mô hình MVC



Mô hình
MVC

7.2 Kiến trúc mô hình MVC



Giải thích ý nghĩa của mô hình:

❖ **Controller:** Có thể gửi yêu cầu đến View liên kết của nó để thay đổi hiển thị trên View, nó cũng có thể gửi yêu cầu đến model để cập nhật trạng thái của model.

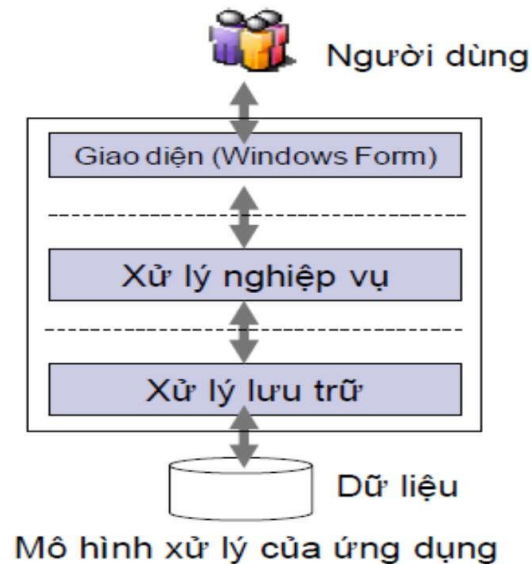
❖ **Model:** Thông báo đến các View và Controller có liên quan khi có thay đổi trạng thái. Thông báo này cho phép các View tạo ra các hiển thị được cập nhật và cho phép các Controller thay đổi các command.

❖ **View:** thông qua Controller yêu cầu Model gửi các thông tin mà nó cần để tạo ra các hiển thị trên View.

7.3 Lớp Models



- Để kết nối và truy xuất với cơ sở dữ liệu SQL Server. Nếu chúng ta viết code trên cùng một Form thì sẽ rất khó khăn trong việc chỉnh sửa code nếu code có cả hàng ngàn dòng.
- Vì thế, trong phần này chúng ta sẽ học cách thức xây dựng lớp lưu trữ dữ liệu (models). Lớp lưu trữ dữ liệu đảm nhận trách nhiệm thực hiện các thao tác kết nối, truy xuất và cập nhật dữ liệu.



7.3 Lớp Models



```
using System.Data;
using System.Data.SqlClient; public class
Database
{
#region "Khai báo biến thành viên "
#endregion
#region "Danh sách các thuộc tính "
#endregion
#region "Nhóm hàm đóng mở kết nối"
#endregion
#region "Nhóm hàm truy xuất dữ liệu"
#endregion
}
```

7.3 Lớp Models



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

• Với mỗi lần kết nối ta sử dụng cùng một connectionString chung. Khai báo một biến static và một property để lưu giữ connectionString xuyên suốt trong chương trình.

```
protected static string _connectionString;    public static string
```

```
ConnectionString  
{  
    get{  
        return _connectionString;  
    }  
    set{  
        _connectionString = value;  
    }  
}
```

7.3 Lớp Models



- ❖ Đồng thời ta cần khai báo các biến để thực hiện thao tác trên cơ sở dữ liệu bao gồm:

```
protected SqlConnection connection; protected SqlDataAdapter  
adapter; protected SqlCommand;
```

Ta viết hàm kết nối CSDL:

```
public void Connect()  
{  
    connection = new SqlConnection(_connectionString);  
}
```

Và hàm ngắt kết nối CSDL:

```
public void Disconnect()  
{  
    Connection.Close();  
}
```

7.3 Lớp Models



- ❖ Để thực hiện truy vấn dữ liệu với các câu truy vấn dữ liệu có sẵn ta tạo hàm truy vấn executeQuery để trả về 1 DataReader

```
public IDataReader      executeQuery(string strStore)
{
    command = new SqlCommand(strStore, connection);
    command.CommandType = CommandType.StoredProcedure;
    return command.ExecuteReader();
}
public void executeNonQuery(string strStore)
{
    command = new SqlCommand(strStore, connection);
    command.CommandType = CommandType.StoredProcedure;
    command.ExecuteNonQuery();
}
public object executeScalar(string strStore)
{
    command = new SqlCommand(strStore, connection);
    command.CommandType = CommandType.StoredProcedure;
    return command.ExecuteScalar();
}
```

7.3 Lớp Models



- ❖ Tạo lớp HocSinhData (add class HocSinhData.cs) để thực hiện các thao tác cập nhật cơ sở dữ liệu với dữ liệu học sinh tương ứng.
 - Lớp HocSinhData sẽ chịu trách nhiệm thực hiện cập nhật CSDL thông qua DataProvider đã có.
 - Mỗi đối tượng HocSinhData sẽ giữ một DataProvider để thực hiện truy xuất CSDL.

```
using System.Data;  
using System.Data.SqlClient; namespace QLHocSinh  
{  
class DataProvider  
{  
protected static string _connectionString; protected SqlConnection  
connection; protected SqlDataAdapter adapter; protected  
SqlCommand;
```

7.3 Lớp Models



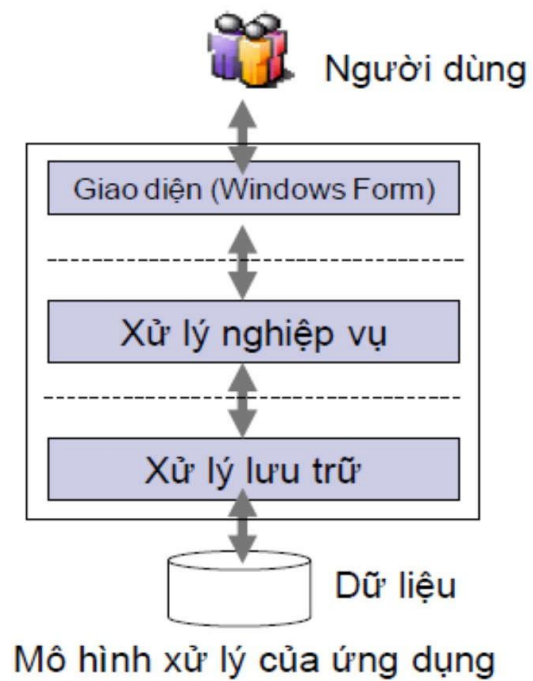
```
public static string ConnectionString
{
    get{return _connectionString;} set{_connectionString = value;}
}
public void Connect(){
    connection = new SqlConnection(_connectionString);
}
public void Disconnect(){ connection.Close(); }
public IDataReader executeQuery(string strStore){ command = new SqlCommand(strStore,
connection);
command.CommandType = CommandType.StoredProcedure; return
command.ExecuteReader();
}
public void executeNonQuery(string strStore) {
command = new SqlCommand(strStore,connection); command.CommandType =
CommandType.StoredProcedure; command.ExecuteNonQuery();
}
public object executeScalar(string strStore){
command = new SqlCommand(strStore,connection); command.CommandType =
CommandType.StoredProcedure; return command.ExecuteScalar();
}
}
}
```

7.4 Lớp Controls



- Để hiển thị dữ liệu trên Windows Form, chúng ta đã viết khá nhiều dòng lệnh trên đó chẳng hạn như : tạo chuỗi kết nối, thực hiện đọc bảng dữ liệu, viết xử lý liên kết dữ liệu ... Việc để lẫn lộn những đoạn code về truy cập dữ liệu và xử lý trên giao diện gây không ít khó khăn trong việc xây dựng, phát triển và bảo trì ứng dụng.
- Vì thế, trong phần này chúng ta sẽ học cách thức xây dựng lớp xử lý (controls). Lớp xử lý đảm nhận trách nhiệm thực hiện các thao tác truy xuất và cập nhật dữ liệu, hiển thị dữ liệu trên Form và tiếp nhận thông tin người dùng.
- Việc phân chia công việc cụ thể cho từng đối tượng không những giúp cho chúng ta xây dựng và phát triển ứng dụng một cách hiệu quả mà còn dễ dàng trong quá trình bảo trì, phù hợp với xu hướng phát triển phần mềm sử dụng các ngôn ngữ lập trình hiện đại.

7.4 Lớp Controls



7.4 Lớp Controls



1. Xây dựng lớp xử lý (Controls) lưu trữ :

1. Thiết kế tổng quan :

Để dễ dàng theo dõi cấu trúc chi tiết của lớp xử lý lưu trữ (XL_BANG), chúng ta bắt đầu tìm hiểu thiết kế tổng quan của nó.

Lớp xử lý dữ liệu (XL_BANG) thực hiện các chức năng :

- Truy xuất dữ liệu từ cơ sở dữ liệu.
- Thực hiện các câu lệnh Sql.

7.4 Lớp Controls



```
using System.Data;
using System.Data.SqlClient; public class
XL_BANG
{
#region "Khai báo biến thành viên "
#endregion
#region "Danh sách các thuộc tính "
#endregion
#region "Nhóm hàm cung cấp thông tin "
#endregion
#region "Nhóm hàm xử lý tính toán "
#endregion
#region "Xử lý sự kiện " #endregion
}
```

7.4 Lớp Controls



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

/// <summary>
/// Summary description for XL_BANG
/// </summary>
public class XL_BANG : DataTable
{
+ "Khai báo biến thành viên "
+ "Danh sách các thuộc tính "
+ "Nhóm hàm cung cấp thông tin "
+ "Nhóm hàm xử lý tính toán "
+ "Xử lý sự kiện "
}
```

Phân vùng với region

7.4 Lớp Controls



- Nhóm từ khóa **#region** và **#endregion** tạo ra các phân vùng, giúp chúng ta dễ dàng quản lý các đoạn lệnh trong quá trình xây dựng lớp.
- Lớp **XL_BANG** được kế thừa từ lớp **DataTable**, đồng nghĩa với việc nó sẽ thừa hưởng tất cả những thuộc tính, phương thức,... từ lớp **DataTable**.
- Trong lớp xử lý trên, chúng ta có thực hiện các thao tác truy xuất và cập nhật dữ liệu, do đó chúng ta cần sử dụng bộ thư viện của **ADO.Net**. Bộ thư viện được sử dụng trong lớp xử lý này là **System.Data.SqlClient**.

7.4 Lớp Controls



2. Cấu trúc chi tiết lớp XL_BANG :

1. Khai báo biến thành viên :

```
#region "Khai báo biến thành viên "  
    //Đối tượng truy cập dữ liệu  
private SqlDataAdapter mBo_doc_ghi;  
//Biến chuỗi chứa nội dung truy cập dữ liệu private string  
mChuoi_Sql;  
//Biến chứa tên bảng muốn truy vấn private string  
mTen_bang;  
//Biến kết nối dùng chung đến nguồn dữ liệu private  
SqlConnection mKet_noi;  
//Biến chứa thông tin vị trí nguồn dữ liệu.  
//Giá trị này phải được gán trước khi sử dụng lớp. public  
string Chuoi_CSDL;  
#endregion
```

7.4 Lớp Controls



Các từ khóa khai báo :

Giá trị	Mô tả
public	Cho biết phương thức được gọi ở mọi nơi
protected	Cho biết phương thức chỉ được gọi trong phạm vi của Class khai báo và các lớp con (Subclass)
private	Cho biết phương thức chỉ được gọi trong phạm vi của Class

7.4 Lớp Controls



Danh sách các thuộc tính :

Ứng với mỗi biến thành viên cần giao tiếp ra bên ngoài, chúng ta cung cấp thuộc tính tương ứng để làm việc.

```
#region "Danh sách các thuộc tính " public string  
Chuoi_Sql  
{  
get { return mChuoi_Sql; } set { mChuoi_Sql = value; }  
}
```

7.4 Lớp Controls



```
public string Ten_bang
{
    get { return mTen_bang; } set {
    mTen_bang = value;}
}
public SqlConnection Ket_noi
{
    get { return mKet_noi; } set {
    mKet_noi = value; }
}
```

7.4 Lớp Controls



Nhóm hàm cung cấp thông tin

```
#region "Nhóm hàm cung cấp thông tin "  
//Thực hiện lấy cấu trúc và dữ liệu vào DataTable  
//sau đó phát sinh bộ lệnh cập nhật dữ liệu public void  
Doc_bang()  
{  
if (mChuoai_Sql == "")  
mChuoai_Sql = "Select * From " + mTen_bang; if (mKet_noi  
== null)  
{  
mKet_noi = new SqlConnection();  
mKet_noi.ConnectionString = Chuoi_lien_ket + "Data  
Source=" + Chuoi_CSDL;  
}
```

7.4 Lớp Controls



```
try
{
    mBo_doc_ghi = new SqlDataAdapter(mChuoai_Sql,
    mKet_noi);
    mBo_doc_ghi.Fill(this);
    mBo_doc_ghi.FillSchema(this, SchemaType.Mapped);
    mBo_doc_ghi.SelectCommand.CommandText = "Select *
    from mTen_bang";
    SqlCommandBuilder Bo_phat_sinh = new
    SqlCommandBuilder(mBo_doc_ghi);
}
catch
{
}
}
#endregion
```

7.4 Lớp Controls



Nhóm hàm xử lý tính toán

```
#region "Nhóm hàm xử lý tính toán "  
//Hàm cập nhật các thay đổi trên DataTable vào CSDL  
public Boolean Ghi()  
{  
    Boolean ketqua=true; try  
    {  
        mBo_doc_ghi.Update(this); this.AcceptChanges();  
    }  
    catch  
    {  
        this.RejectChanges(); ketqua=false;  
    }  
    return ketqua;  
}
```

7.4 Lớp Controls



```
//Hàm thực hiện một câu lệnh truy vấn (hành động), nếu thành công trả  
//về số mẫu tin được cập nhật, ngược lại hàm trả về -1 public  
Int32 Thuc_hien_lenh(string pLenh)  
{  
    try  
    {  
        SqlCommand Cau_lenh=new  
        SqlCommand(pLenh,mKet_noi); mKet_noi.Open();  
        int ketqua=Cau_lenh.ExecuteNonQuery(); mKet_noi.Close();  
        return ketqua;  
    }  
    catch {return -1;}  
}
```

7.4 Lớp Controls



```
//Hàm thực hiện nội dung lệnh tính toán thống kê, nếu thành  
//công trả về kết quả thống kê, ngược lại trả về Nothing  
public Object Thuc_hien_lenh_tinh_toan(string pLenh)  
{  
    try  
    {  
        SqlCommand Cau_lenh=new SqlCommand(pLenh,mKet_noi);  
        mKet_noi.Open();  
        Object ketqua=Cau_lenh.ExecuteScalar(); mKet_noi.Close();  
        return ketqua;  
    }  
    catch  
    {  
        return null;  
    }  
}  
#endregion
```

7.4 Lớp Controls



❖ Tạo lớp `HocSinhInfo` chứa các thông tin của một học sinh với ràng buộc nghiệp vụ tương ứng.

Lớp `HocSinhInfo` chứa các thông tin lưu trữ của một đối tượng học sinh. Lớp này chỉ gồm các biến và thuộc tính (hoạt động tương tự như một struct). Đây chính là lớp truyền tải dữ liệu giữa tầng giao diện và tầng xử lý tính toán.

```
using System; namespace QLHocSinh
{
    class HocSinhInfo
    {
        private string _maHS; private string _tenHS; private bool
            _gioitinh; private DateTime _ngaysinh; private string _diachi;
        Byte[] image;
        private string _maLop;
```

7.4 Lớp Controls



```
public string MaHS
{
    get{
        return _maHS;
    }
    set
    {
        if ( value == null)
            throw new Exception("Ma HS khong duoc rong");
        _maHS = value;
    }
}

public string TenHS
{
    get{
        return _tenHS;
    }
    set
    {
        if ( value == null)
            throw new Exception("Ten HS khong duoc rong");
        _tenHS = value;
    }
}
```

7.4 Lớp Controls



```
public bool GioiTinh
{
    get{
        return _gioitinh;
    }
    set
    {
        _gioitinh = value;
    }
}

public DateTime NgaySinh
{
    get{
        return _ngaysinh;
    }
    set
    {
        _ngaysinh = value;
    }
}
```

7.4 Lớp Controls



```
public string DiaChi
{
    get{
        return _diachi;
    }
    set
    {
        if ( value == null)
            throw new Exception("Dia chi HS khong duoc
            rong");
        _diachi = value;
    }
}
```

7.4 Lớp Controls



```
public Byte[] HinhAnh
{
    get { return image;
    }
    set { image = value; }
}
public string MaLop
{
    get{
        return _maLop;
    }
    set
    {
        _maLop = value;
    }
}
```

7.4 Lớp Controls



Tạo lớp HocSinhCtrl để thực hiện các công việc nghiệp vụ:

Trong mỗi đối tượng HocSinhCtrl giữ các đối tượng HocSinhInfo và HocSinhData.

Thông qua đối tượng HocSinhInfo truyền nhận dữ liệu với HocSinhData để thực hiện tương tác với tầng cơ sở dữ liệu.

```
using System; namespace QLHocSinh
```

```
{
```

```
class HocSinhCtrl
```

```
{
```

```
private HocSinhInfo info; private string _tenHS; private bool _gioitinh; private  
DateTime _ngaysinh; private string _diachi;
```

```
Byte[] image;
```

```
private string _maLop;
```

7.4 Lớp Controls



```
public string MaHS
{
    get{
        return _maHS;
    }
    set
    {
        if ( value == null)
            throw new Exception("Ma HS khong duoc rong");
        _maHS = value;
    }
}

public string TenHS
{
    get{
        return _tenHS;
    }
    set
    {
        if ( value == null)
            throw new Exception("Ten HS khong duoc
rong");
        _tenHS = value;
    }
}
```

7.4 Lớp Controls



```
public bool GioiTinh
{
    get{
        return _gioitinh;
    }
    set
    {
        _gioitinh = value;
    }
}

public DateTime NgaySinh
{
    get{
        return _ngaysinh;
    }
    set
    {
        _ngaysinh = value;
    }
}
```

7.4 Lớp Controls



```
public string DiaChi
{
    get{
        return _diachi;
    }
    set
    {
        if ( value == null)
            throw new Exception("Dia chi HS khong duoc
            rong");
        _diachi = value;
    }
}

public Byte[] HinhAnh
{
    get { return image;
    }
    set { image = value; }
}
}
```

7.4 Lớp Controls



```
public string MaLop
{
    get{
        return _maLop;
    }
    set
    {
        _maLop = value;
    }
}
```

7.4 Lớp Controls



Tạo lớp **HocSinhCtl** để thực hiện các công việc nghiệp vụ:

Trong mỗi đối tượng **HocSinhCtl** giữ các đối tượng **HocSinhInfo** và **HocSinhData**. Thông qua đối tượng **HocSinhInfo** truyền nhận dữ liệu vào **HocSinhData** để thực hiện tương tác với tầng cơ sở dữ liệu.

```
using System;
namespace QLHocSinh
{
    class HocSinhCtl
    {
        private HocSinhInfo info = new HocSinhInfo();
        private HocSinhData data = new HocSinhData();

        public HocSinhInfo HocSinh
        {
            get
            {
                return info;
            }
            set
            {
                info = value;
            }
        }

        public void insert()
        {
            data.insert(info.MaHS, info.TenHS, info.Ngaysinh,
                info.Diachi, info.DTB, info.MaLop);
        }

        public void delete()
        {
            data.delete(info.MaHS);
        }

        public void update()
        {
            data.update(info.MaHS, info.TenHS, info.Ngaysinh,
                info.Diachi, info.DTB, info.MaLop);
        }
    }
}
```

7.5 Lớp Views



Xây dựng Form giao diện nhập thông tin học sinh như hình cho lớp



5. Bài tập:

- Hoàn chỉnh tầng giao diện của ứng dụng. (gợi ý: trong tầng giao diện chỉ sử dụng các lớp **HocSinhCtl**, **HocSinhInfo**)
- Viết cấu trúc lớp **LopInfo** để binding danh sách lớp vào combobox.
- Viết các lớp để xử lý thêm, xóa, sửa Lop
- ...

Tài liệu tham khảo



- [1] Vidya Vrat Agarwal & James Huddleston, “Beginning C Sharp 2008 Databases From Novice to Professional”, Apress, 2008
- [2] Phạm Hữu Khang, “C# 2005 - Tập 4, Quyển 1: Lập Trình Cơ Sở Dữ Liệu”, 2006.
- [3] Bipin Joshi, “Beginning XML with C# 2008: From Novice to Professional”, Apress, 2008.
- [4] Fabio Claudio Ferracchiati, “LINQ for Visual C# 2008”, Apress, 2008.
- [5] <http://www.glib.hcmus.edu.vn> (Thư viện Đại học Khoa Học Tự Nhiên TP. HCM)