

LẬP TRÌNH .NET (.NET PROGRAMMING)

Giảng viên:

Đặng Hoài Phương

Bộ môn:

Công nghệ phần mềm

Khoa:

Công nghệ Thông tin

Trường Đại học Bách Khoa

Đại học Đà Nẵng





MÔ HÌNH 3-LAYERS



Lambda

```
var x = ProductList.Select(m => new { m.ID, m.Name });
```

Query

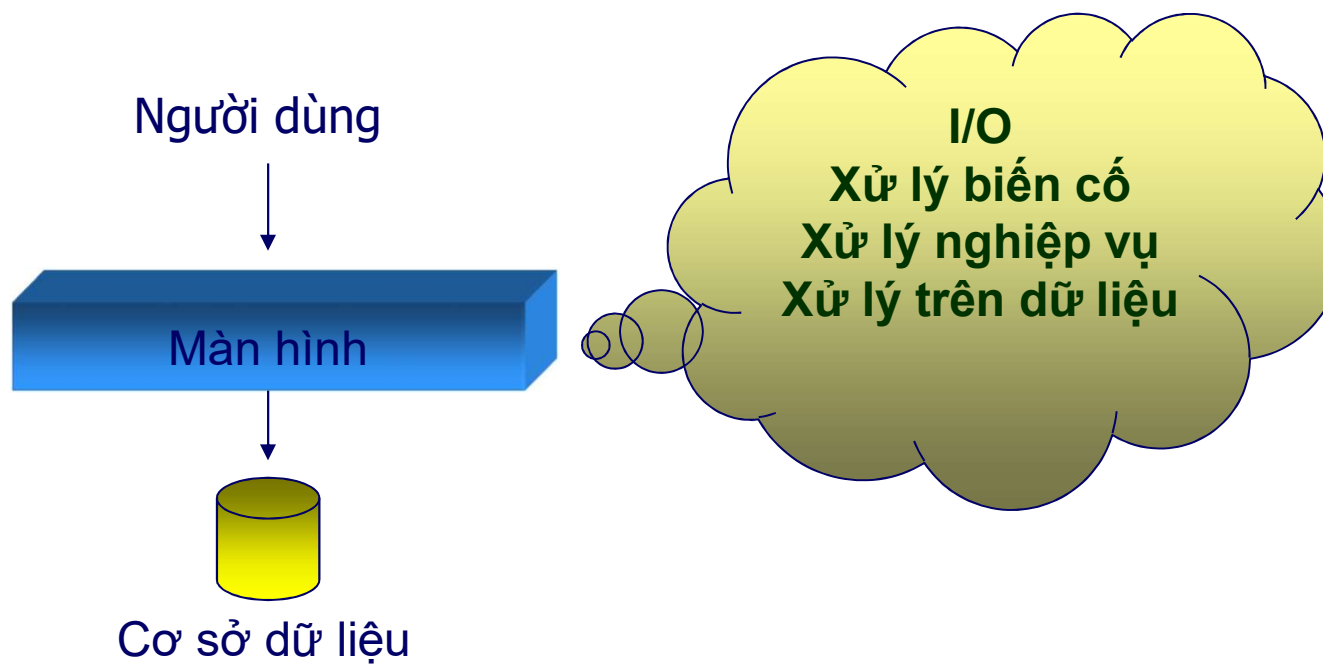
```
var x = from m in ProductList
        select new
        {
            m.ID,
            m.Name
        };
```



1.1 LỊCH SỬ .NET

TÌNH HÌNH TRƯỚC KHI MS .NET RA ĐỜI

<code>ProductList.Where(m=>m.ID <5)</code>	truy vấn trả về các Product có ID < 5
<code>ProductList.Where(m=>m.Name == "Product1")</code>	trả về Product có Name là Product1
<code>ProductList.skip(2)</code>	trả về danh sách product từ vị trí thứ 2 trở đi
<code>ProductList.take(3)</code>	Trả về 3 Product đầu tiên
<code>ProductList.skip(2) .take(3)</code>	Trả về 3 Product từ vị trí thứ 2
<code>ProductList.Max(m=>m.ID);</code>	Trả về ProductID có giá trị lớn nhất
<code>ProductList.Min(m=>m.ID);</code>	Trả về ProductID có giá trị nhỏ nhất
<code>ProductList.Sum(m=>m.Price);</code>	Trả về tổng giá tất cả các Product

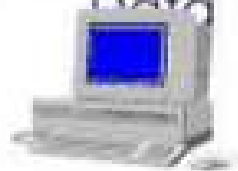


- Tối ưu hóa việc quản lý các thành phần của hệ thống → nhóm các thành phần có cùng chức năng lại với nhau.
- Trong phát triển phần mềm sử dụng kiến trúc đa tầng: Mô hình 3 lớp, mỗi lớp thực hiện 1 chức năng khác nhau.
 - Presentation;
 - Business Logic;
 - Data Access.
- layers phân chia ứng dụng thành các thành phần theo logic chức năng hoặc vai trò.

- 3-tiers là một kiến trúc kiểu client/server;
- Gồm 3 tầng (module) độc lập:
 - Giao diện người dùng (UI-user interface)
 - Các quy tắc xử lý (BR-business rule hay BL-business logic)
 - Lưu trữ dữ liệu.
- Mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế;
- 3-tiers liên quan đến cách phân chia vật lý các thành phần trên các máy tính khác nhau.

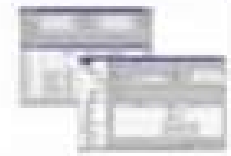
Physical view

Application +
Data



Logical view

GUI



Business logic



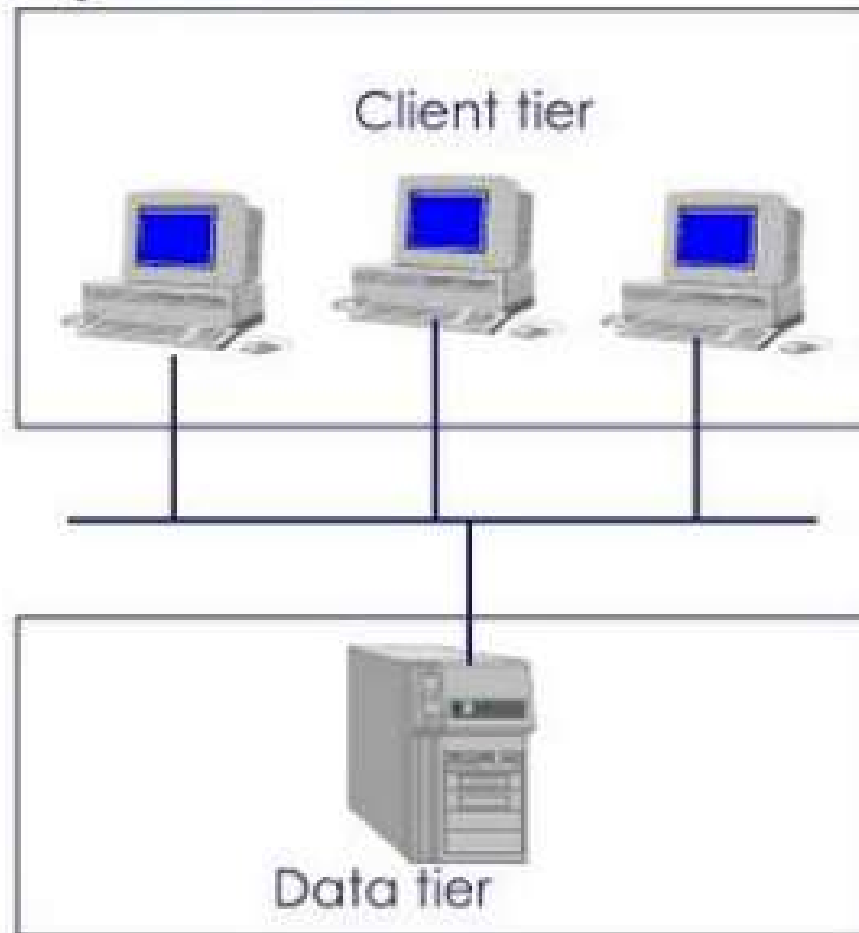
Data Access



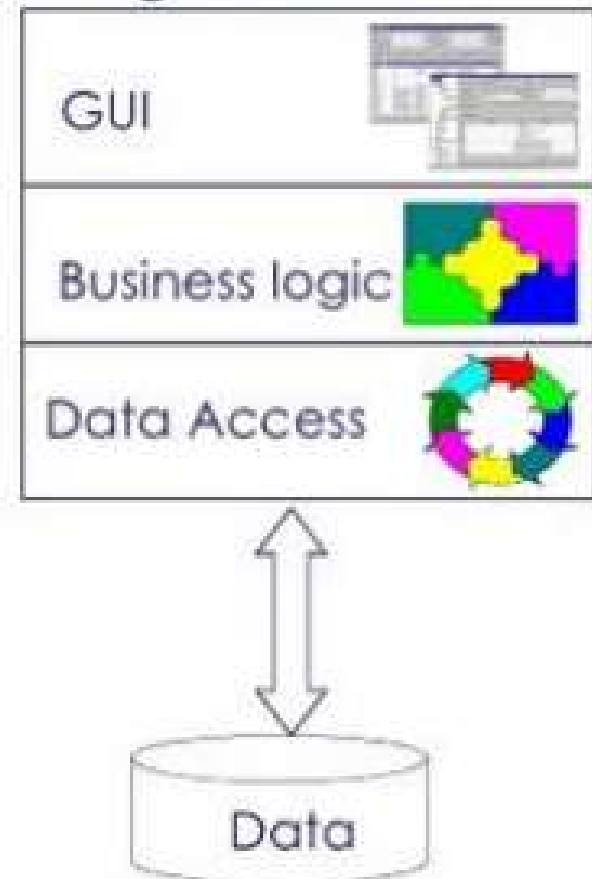
Data



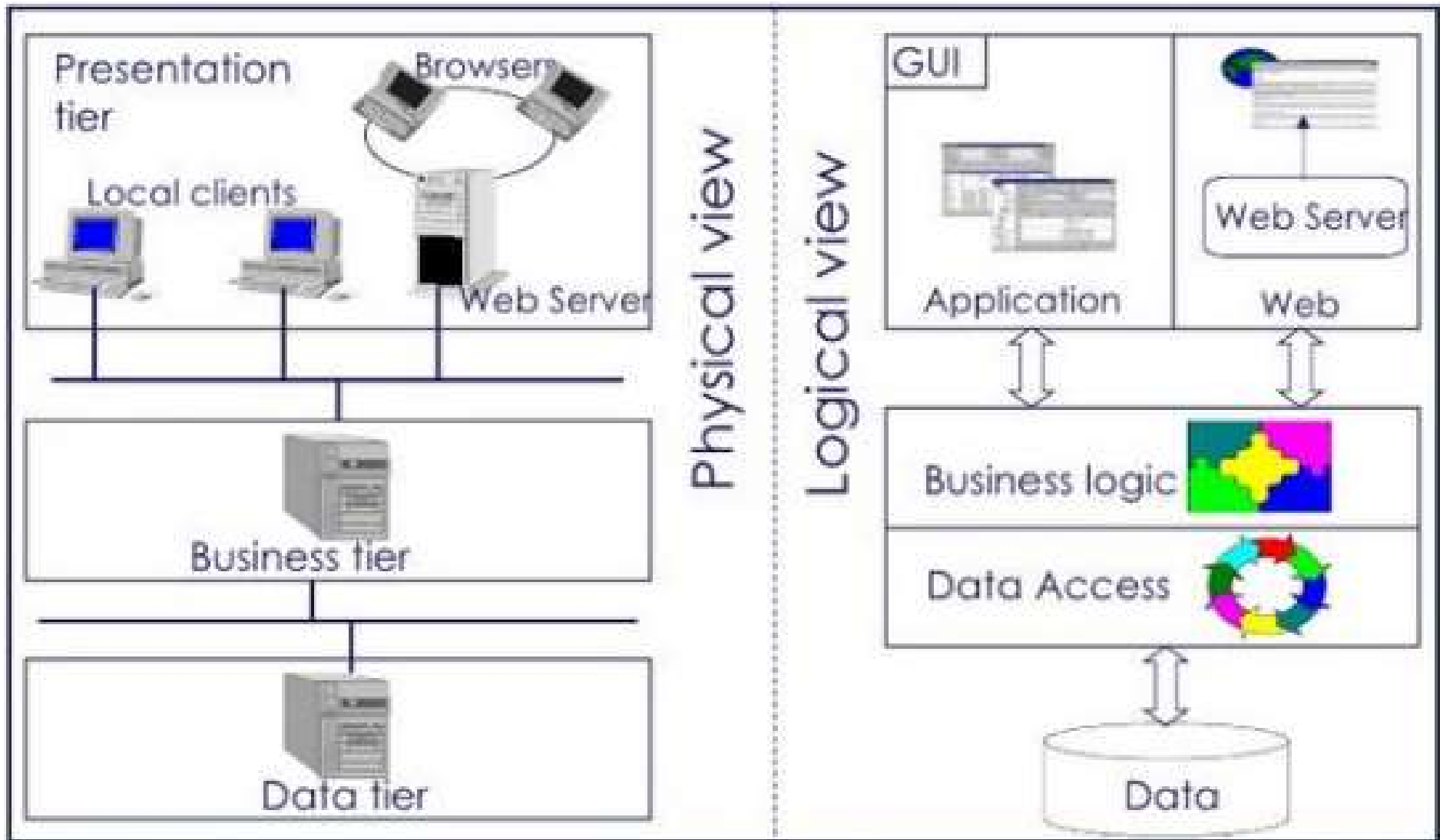
Physical view



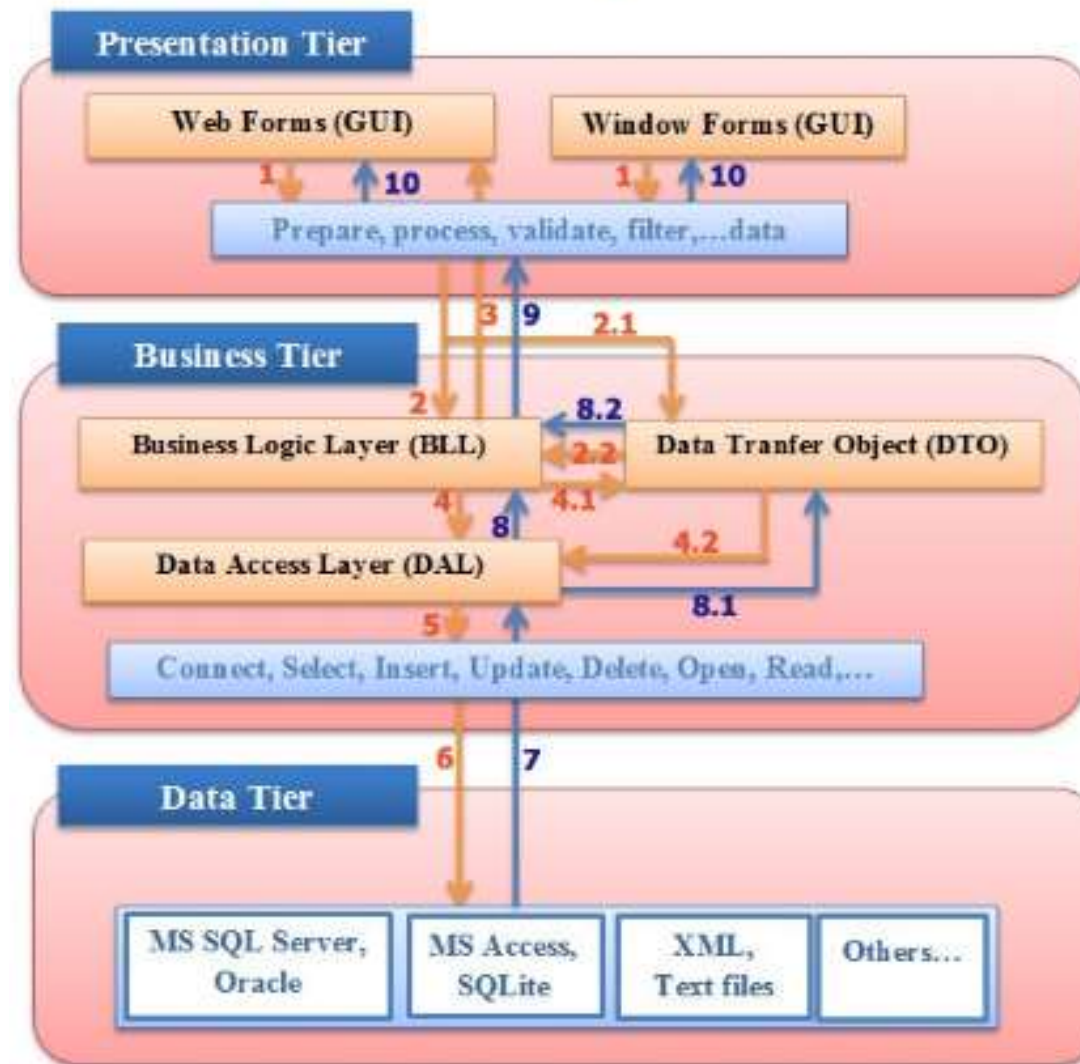
Logical view



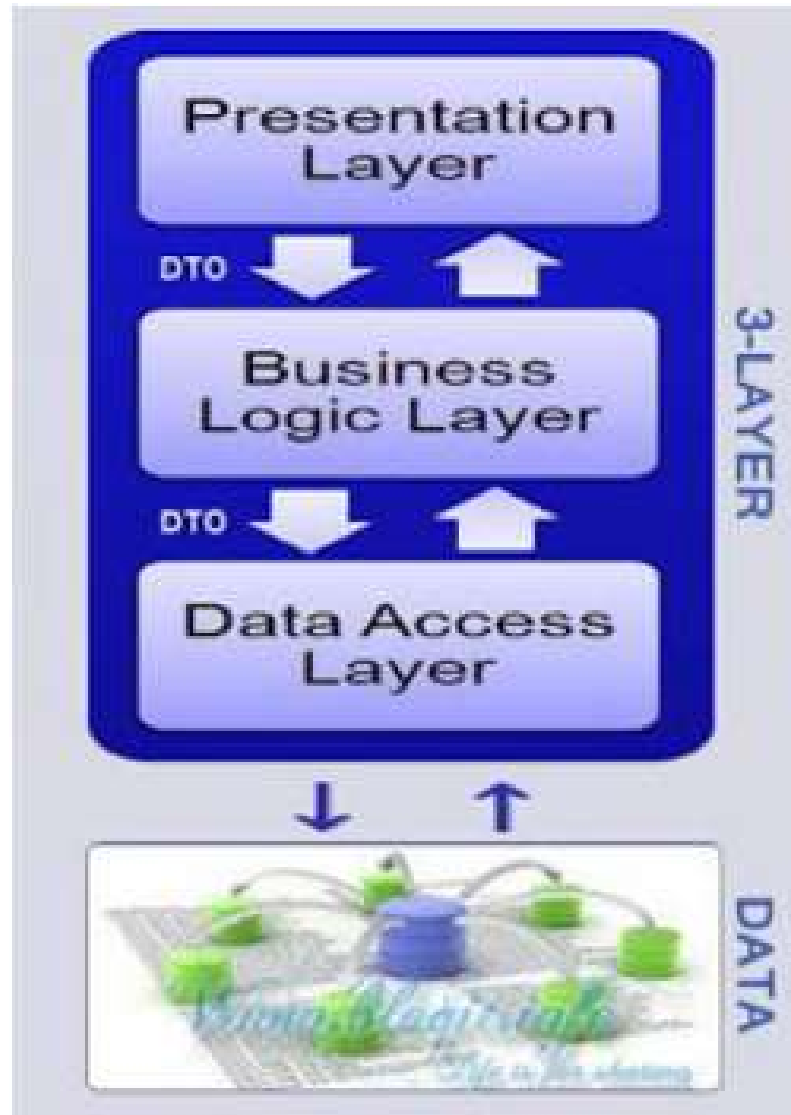
3-TIERS & 3-LAYERS



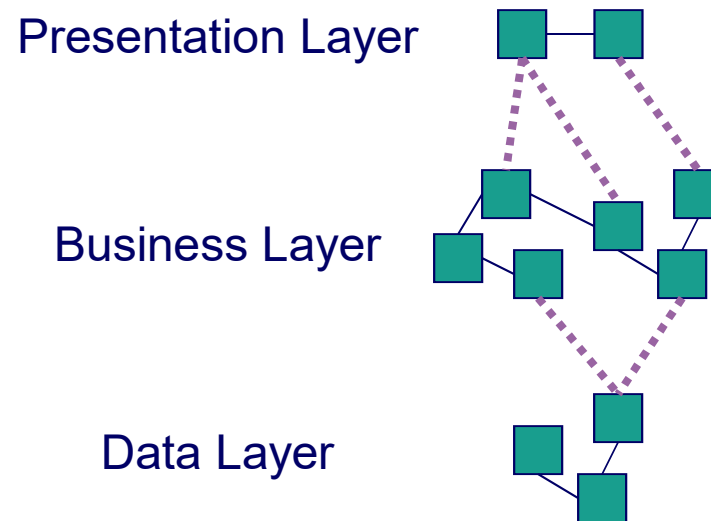
Three-Tiers & Three-Layers Architecture



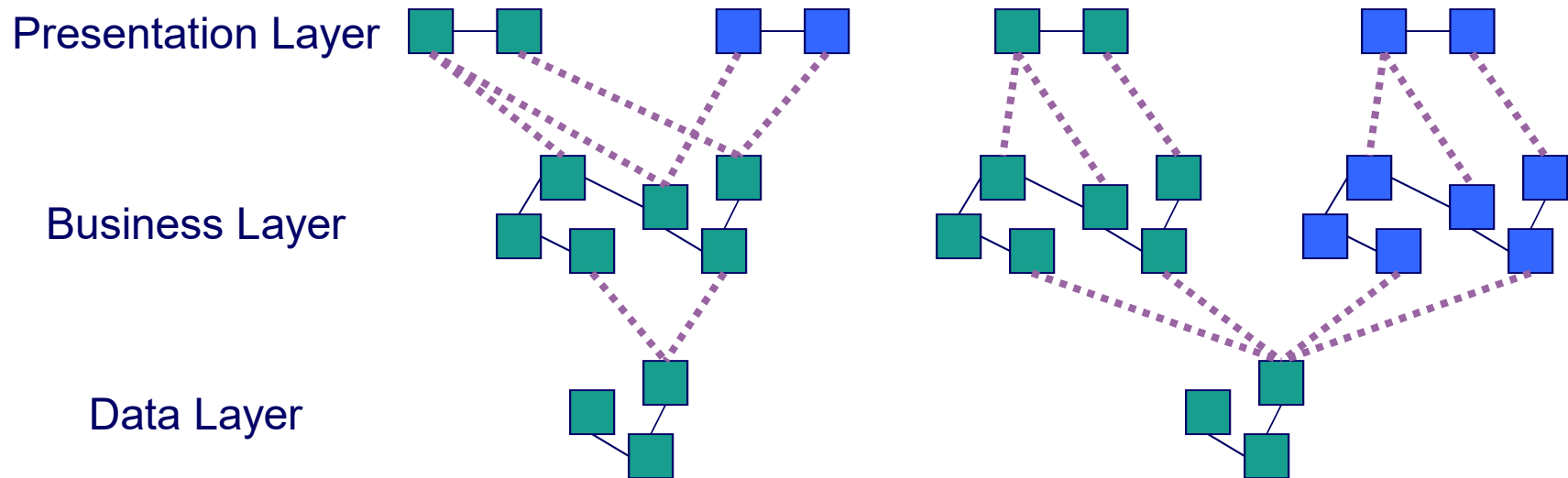
- GUI (Presentation) Layer: nhập liệu và trình bày dữ liệu, có thể bao gồm các bước kiểm tra dữ liệu trước khi gọi Business Logic Layer;
- Business Layer Logic: kiểm tra các yêu cầu nghiệp vụ trước khi cập nhật dữ liệu, quản lý các Transaction, quản lý các concurrent access.
- Data Access Layer: kết nối CSDL, thao tác với CSDL (thêm, sửa, xóa, tìm kiếm, ...).
- Data Transfer Object (DTO): là cầu nối giữa các Layer.
 - Ta cũng có thể dùng các parameter để truyền dữ liệu;
 - Nhiều parameters → dùng DTO.



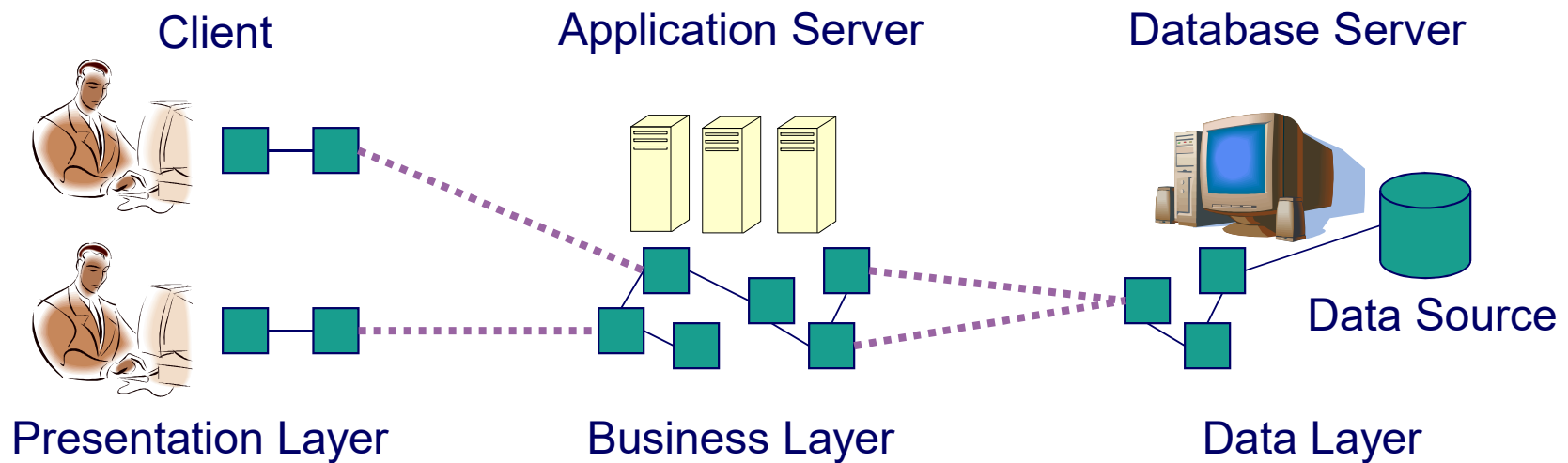
- Giảm sự gắn kết giữa các thực thể phần mềm (decoupling).



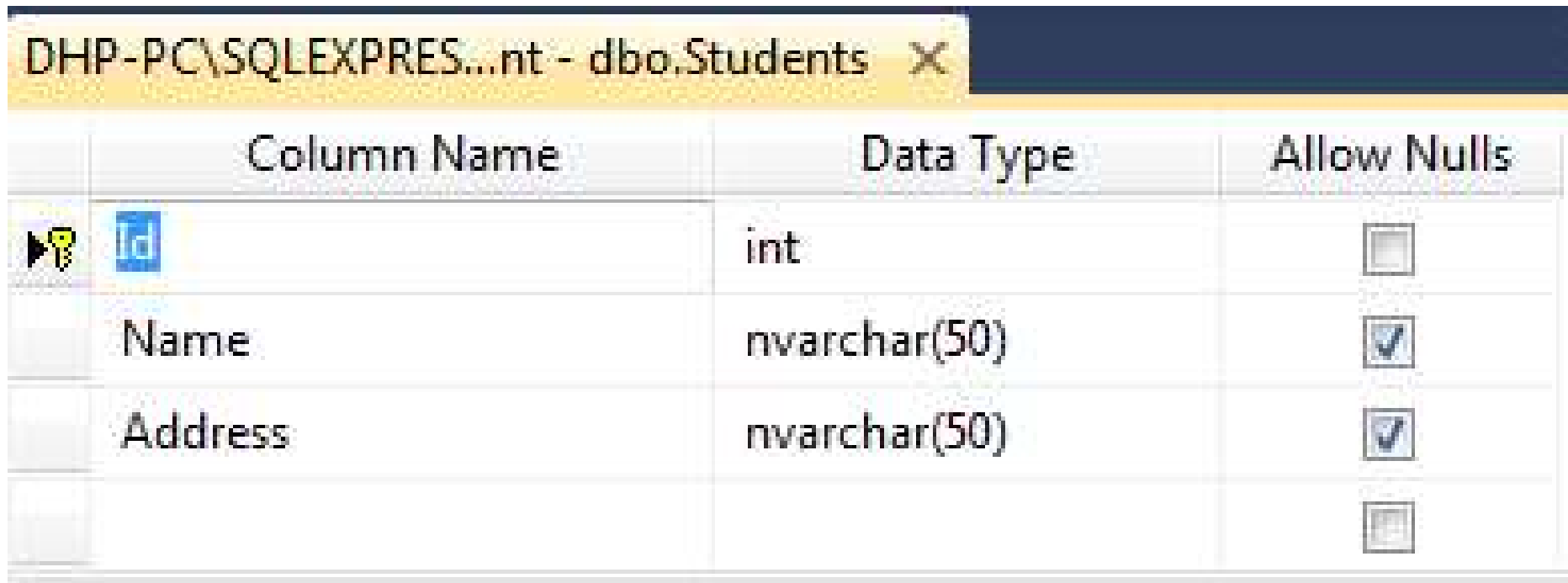
- Tái sử dụng.




- Chia sẻ trách nhiệm.



- Sử dụng mô hình 3-layers viết chương trình Quản lý sinh viên.
- Cơ sở dữ liệu: **DbStudent**
 - Trong đó Id là Khóa chính, tự động tăng.

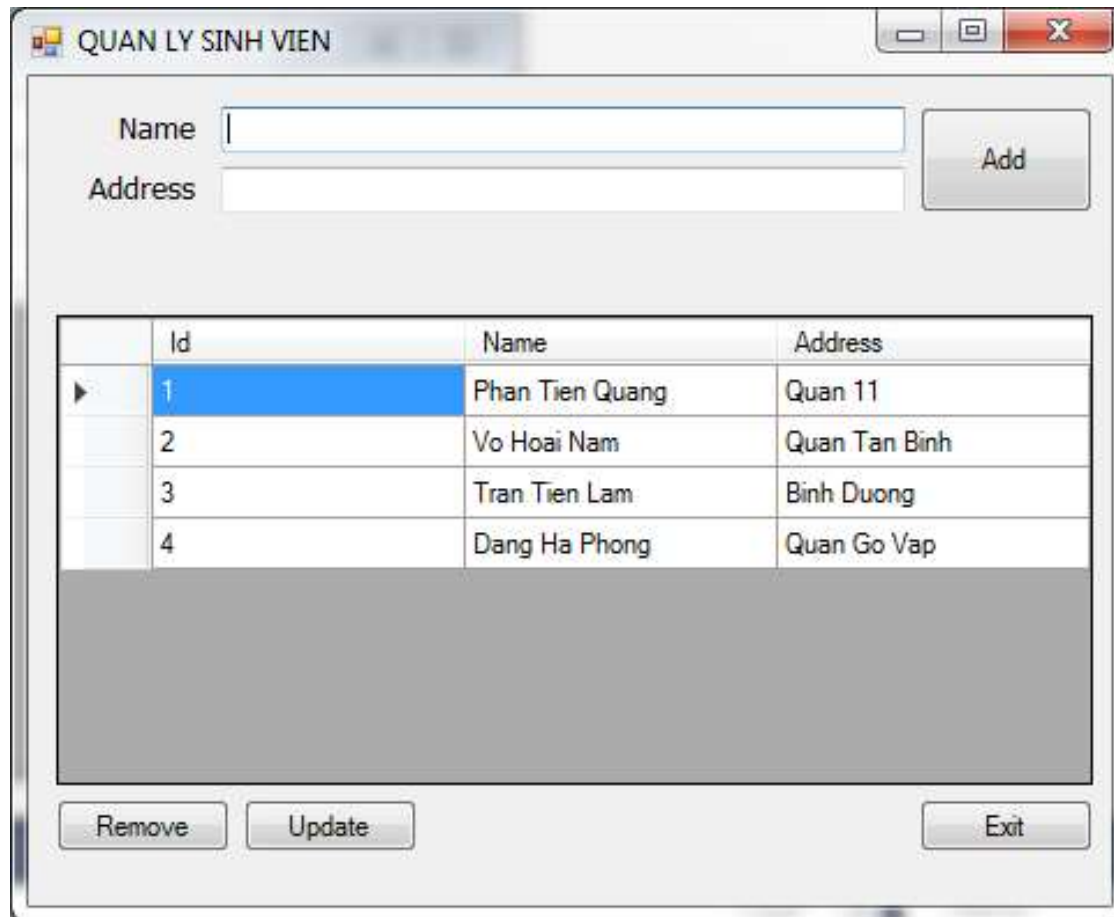


DHP-PC\SQLEXPRES...nt - dbo.Students X			
	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	nvarchar(50)	<input checked="" type="checkbox"/>
	Address	nvarchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

- Cơ sở dữ liệu: **DbStudent**

DHP-PC\SQLEXPRES...nt - dbo.Students X			
	Id	Name	Address
▶	1	Phan Tien Quang	Quan 11
	2	Vo Hoai Nam	Quan Tan Binh
	3	Tran Tien Lam	Binh Duong
	4	Dang Ha Phong	Quan Go Vap
✱	NULL	NULL	NULL

- Giao diện Winform:



The Winform application window, titled "QUAN LY SINH VIEN", features a standard Windows-style title bar with minimize, maximize, and close buttons. The main interface includes two text input fields labeled "Name" and "Address" at the top, followed by an "Add" button. Below these is a table with four columns: "Id", "Name", and "Address". The table contains four rows of student data, with the first row selected. At the bottom of the window are three buttons: "Remove", "Update", and "Exit".

	Id	Name	Address
▶	1	Phan Tien Quang	Quan 11
	2	Vo Hoai Nam	Quan Tan Binh
	3	Tran Tien Lam	Binh Duong
	4	Dang Ha Phong	Quan Go Vap

- Cấu trúc Project:

- BLL: **B**usiness **L**ogic **L**ayer

- **StudentBLL**: Gồm các nghiệp vụ xử lý chính cho lớp Student (Add, Remove, Edit, LoadData)

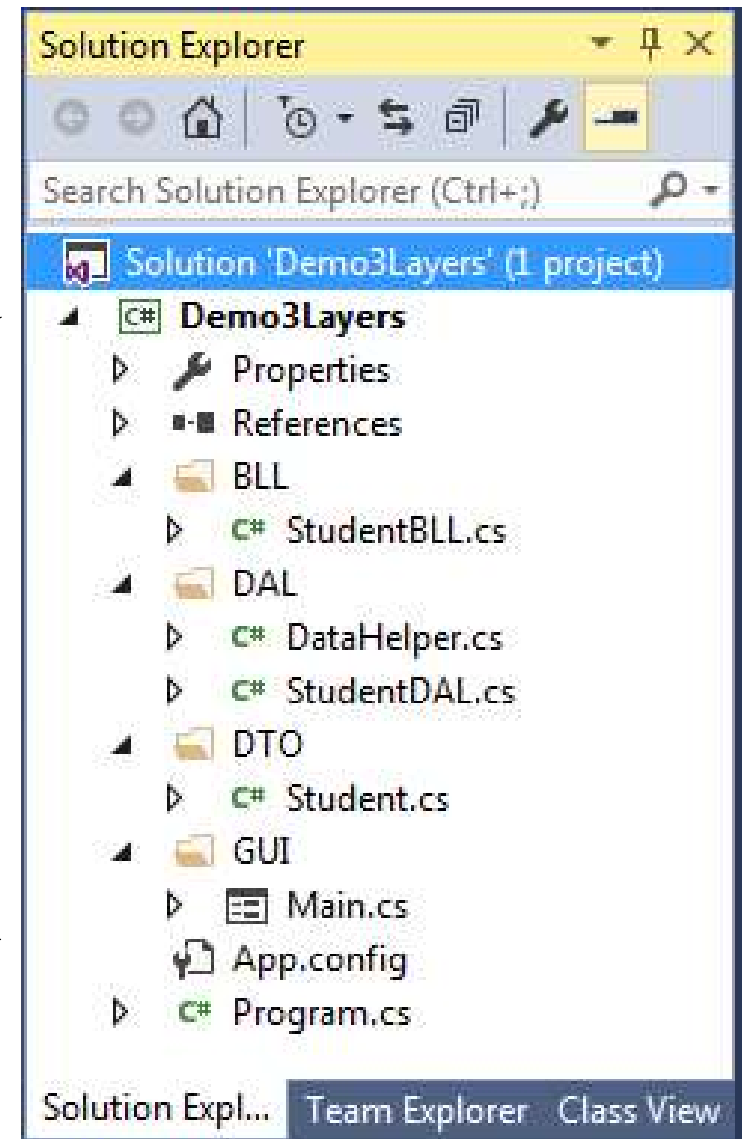
- DAL: **D**ata **A**ccess **L**ayer

- **DataHelper**
 - **StudentDAL**

- GUI

- DTO: **D**ata **T**ransfer **O**bject

- **Student**: đối tượng trao đổi dữ liệu giữa các layer.



- Class **StudentBLL**:

- Gồm các nghiệp vụ xử lý chính: Show, Add, Edit và Remove;
- Gọi các phương thức xử lý ở StudentDAL.

```
using Demo3Layers.DAL;
using Demo3Layers.DTO;

namespace Demo3Layers.BLL
{
    public class StudentBLL
    {
        StudentDAL dal = new StudentDAL();

        public Student[] GetList()
        {
            return null;
        }
    }
}
```

```
public bool Add(Student student)
{
    return false;
}

public bool Remove(string name)
{
    return false;
}

public bool Update(Student student)
{
    return false;
}
}
```

- Class **DataHelper**:

- Thực hiện các thao tác: Insert, Delete, Update.
- Truy vấn CSDL → trả về **datatable**;
- Kiểm tra kết nối CSDL, thông tin CSDL lấy từ file app.config;
- **Constructor**: Khởi tạo đối tượng kết nối;
- **DataTable ExecuteQuery(string query)**: truy vấn CSDL;
- **void ExecuteNonQuery(string query)**: các thao tác Insert, Delete, Update

```
public DataTable ExecuteQuery(string query)
{
    // todo
    return null;
}
```

```
public void ExecuteNonQuery(string query)
{
    // todo
}
```

- Class **StudentDAL**:
 - Xử lý trên table trong lớp DataHelper.
 - Hiển thị danh sách Student ra DataGridView:
 - **Student[] GetList()**: hàm lấy danh sách Student (khởi tạo 1 mảng các đối tượng Student) từ datatable của DataHelper;
 - **Student GetStudentFromDataRow(DataRow row)**: hàm khởi tạo 1 đối tượng Student từ 1 row trong datatable

GUI

use BLL

```
12 namespace Demo3Layers
13 {
14     public partial class Main : Form
15     {
16         StudentBLL bll = new StudentBLL();
17
18         public Main()
19         {
20             InitializeComponent();
21             this.Load += Main_Load;
22         }
23
24         void Main_Load(object sender, EventArgs e)
25         {
26             LoadData();
27         }
28
29         void LoadData()
30         {
31             dgvStudents.DataSource = bll.GetList();
32         }
33     }
34 }
```

use DAL

BLL

```
4 namespace Demo3Layers.BLL
5 {
6     public class StudentBLL
7     {
8         StudentDAL dal = new StudentDAL();
9
10         public Student[] GetList()
11         {
12             return dal.GetList();
13         }
14     }
15 }
```

DTO

DAL

Data Handle

```
4 namespace Demo3Layers.DAL
5 {
6     public class StudentDAL
7     {
8         DataHelper helper = new DataHelper();
9
10         private Student GetStudentFromDataRow(DataRow row)
11         {
12             // ...
13         }
14
15         public Student[] GetList()
16         {
17             // ...
18         }
19     }
20 }
```

- Lưu ý:
 - Nếu dùng câu lệnh query như trên ???
→ bị tấn công bằng **sql injection**
→ giải pháp: **add Parameters** hoặc **Stored Procedure**.
 - Sử dụng LINQ có tối ưu hơn không???
→ Có, vì nó auto generate DTO & các hàm tương tác CSDL (stored procedure) → sử dụng trong DAL.
 - Có thể sử dụng Project làm Layer hay không ??? → tạo liên kết giữa các Project trong mô hình 3-layers như thế nào ???