



# **LẬP TRÌNH C# 3**

## **BÀI 3: LẬP TRÌNH CƠ SỞ DỮ LIỆU**

- TỔNG QUAN VỀ ADO.NET
- Các đối tượng của ADO.NET

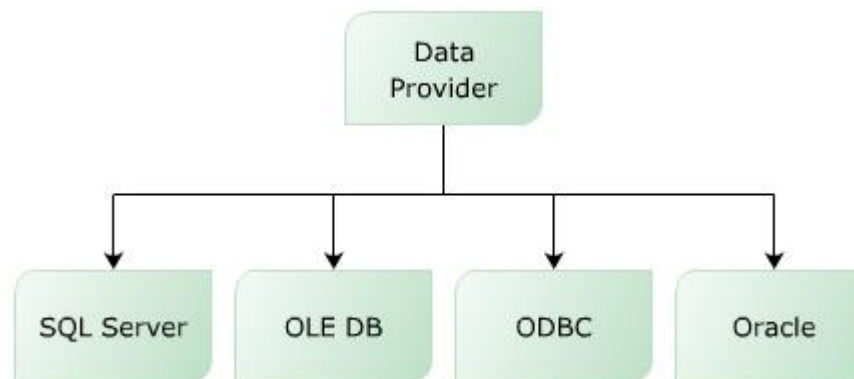


- ❑ Chúng ta thấy rằng hầu hết các ứng dụng hiện nay đều cần đến dữ liệu. Dữ liệu từ người dùng nhập vào, dữ liệu được lưu trữ trong các ứng dụng hoặc dữ liệu từ hệ thống này được truy xuất sử dụng bởi hệ thống khác. Tất cả các dữ liệu đều là nguồn thông tin mà ứng dụng cần xử lý với chức năng chính tìm kiếm, tính toán, thống kê và ra quyết định
- ❑ Dữ liệu không đơn giản chỉ là các file văn bản hay các file nhị phân với cấu trúc record do người dùng định nghĩa. Thay vào đó hầu hết các ứng dụng được tổ chức logic dựa trên cấu trúc cơ sở dữ liệu quan hệ và lưu trữ vật lý dữ liệu dựa vào các hệ quản lý cơ sở dữ liệu quan hệ như: Access, SQL, Oracle, DB2
- ❑ Những vấn đề mà người thiết kế và lập trình ứng dụng quan tâm khi làm việc với dữ liệu là đảm bảo tính toàn vẹn, nhất quán và bền vững của dữ liệu, có thể cung cấp khả năng truy xuất đồng thời dữ liệu của nhiều người dùng, bảo mật an toàn dữ liệu giữa các hệ thống lưu trữ khác nhau

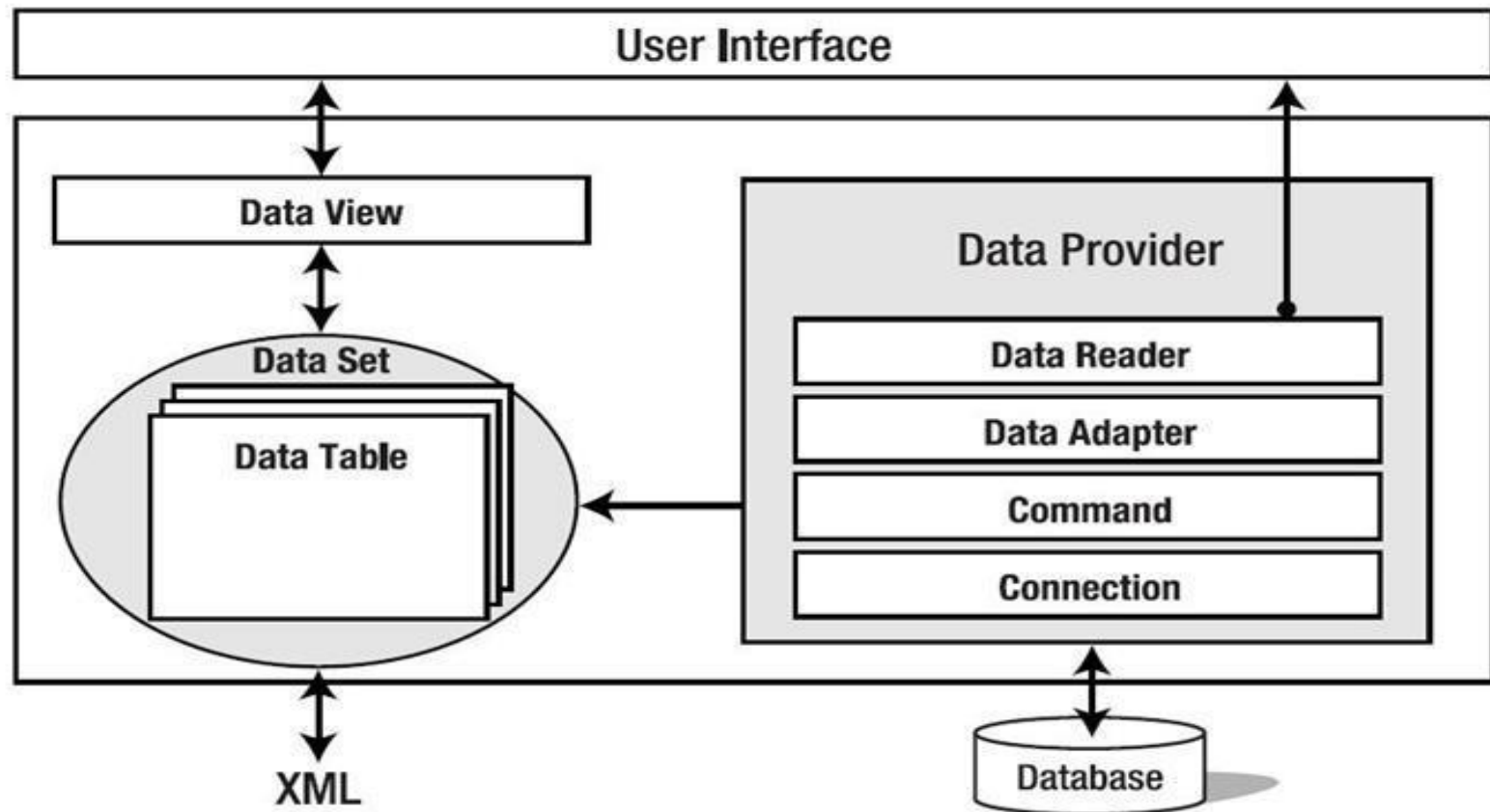
- ❑ ADO.NET là một bộ các thư viện hướng đối tượng (OOP) cho phép bạn tương tác với dữ liệu nguồn.
- ❑ Mục tiêu chính của ADO.NET là:
  - ❖ Cung cấp các lớp để thao tác CSDL trong cả hai môi trường là phi kết nối (Disconnected data) và kết nối (Connected data).
  - ❖ Tương tác với nhiều nguồn dữ liệu thông qua mô tả dữ liệu chung
  - ❖ Tối ưu truy cập nguồn dữ liệu (OleDb & SQL server).

❑ Các lớp của ADO.NET được đặt trong Namespace là System.Data. ADO.NET bao gồm 2 Provider (2 bộ thư viện) (thường dùng) để thao tác với các CSDL là:

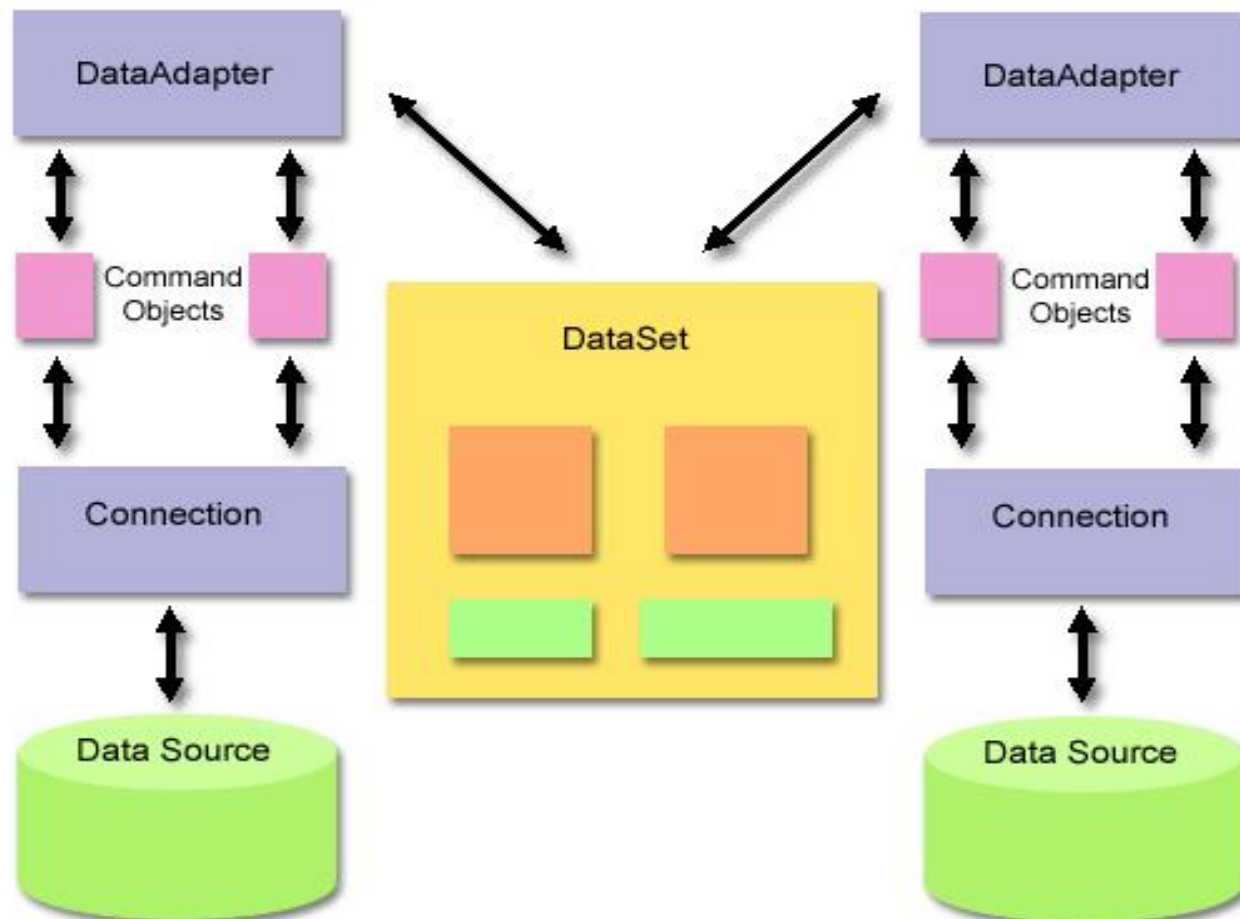
- ❖ OleDb Provider (nằm trong System.Data.OleDb) dùng để truy xuất đến bất kỳ CSDL nào có hỗ trợ OleDb
- ❖ SQL Provider (nằm trong System.Data.SqlClient) chuyên dùng để truy xuất đến CSDL SQL Server



## ❑ KIẾN TRÚC ADO.NET



## ❑ KIẾN TRÚC ADO.NET

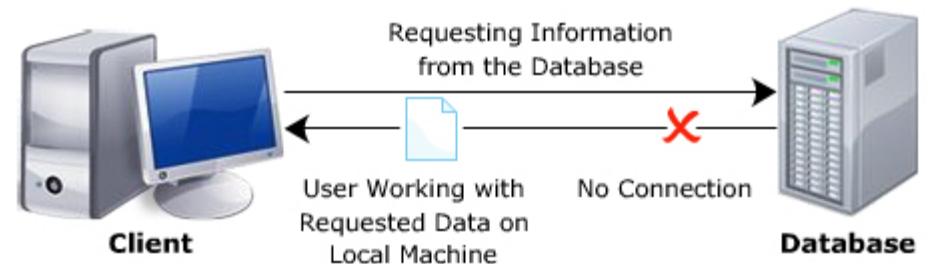


## ❑ Mô hình Connected và Disconnected



Multiple Data Causing Network Traffic

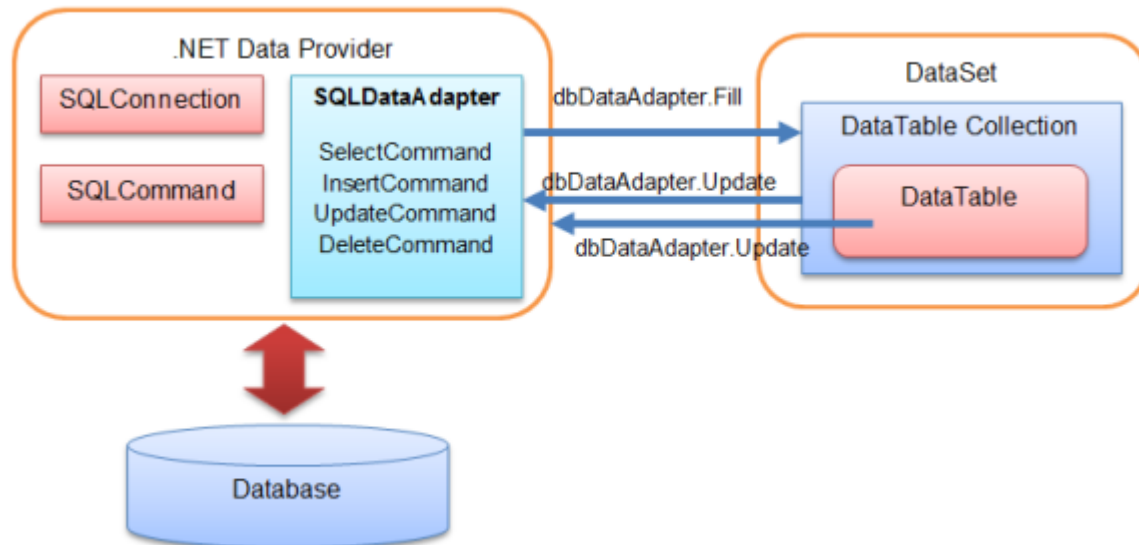
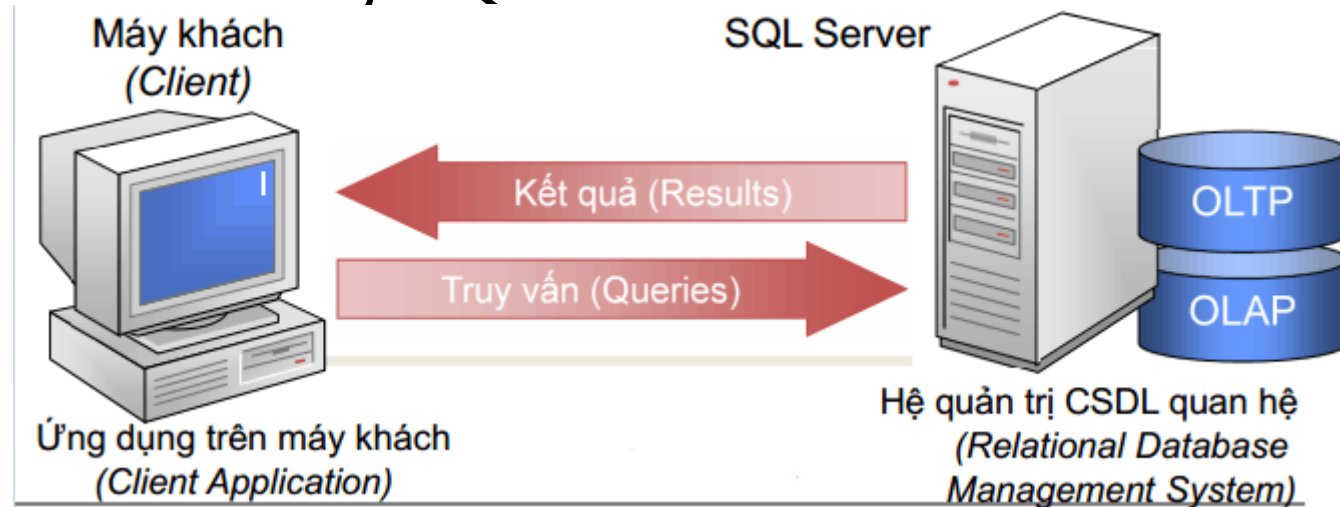
Connected Architecture



Disconnected Architecture



## ❑ Mô hình dùng SQL Provider



## ❑ Đối tượng SqlConnection

- ❖ Để tương tác với database, bạn phải có một kết nối tới nó
- ❖ Kết nối giúp xác định database server, database name, user name, password, và các tham số cần thiết để kết nối tới database
- ❖ Một đối tượng connection được dùng bởi đối tượng command vì thế chúng sẽ biết database nào để thực thi lệnh
- ❖ Đối tượng Connection có hai phương thức hay dùng là Open() để mở kết nối tới CSDL và Close() để đóng kết nối tới CSDL

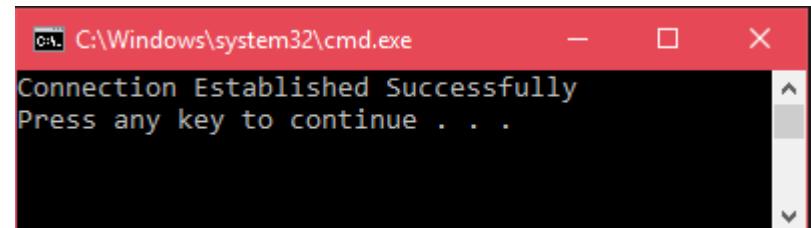
- ❑ Sql Connection String chứa các cặp key/value để xác định cách tạo một kết nối đến database. Chúng bao gồm vị trí, tên của database và chế độ bảo mật.

Connection String Parameter Name	Description
Data Source	Identifies the server. Could be local machine, machine domain name, or IP Address.
Initial Catalog	Database name.
Integrated Security	Set to <b>SSPI</b> to make connection with user's Windows login
User ID	Name of user configured in SQL Server.
Password	Password matching SQL Server User ID.

## ❑ Sử dụng SqlConnection

```
static void Main(string[] args)
{
    new Program().Connecting();
}

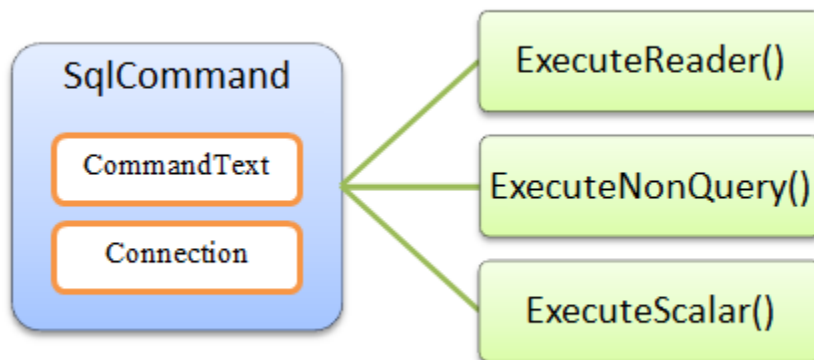
public void Connecting()
{
    using (
        // Creating Connection
        SqlConnection con = new SqlConnection("data source=.; database=student; integrated security=SSPI")
    )
    {
        con.Open();
        Console.WriteLine("Connection Established Successfully");
    }
}
```



A screenshot of a Windows Command Prompt window. The title bar is red and shows the path "C:\Windows\system32\cmd.exe". The window contains the text "Connection Established Successfully" followed by "Press any key to continue . . .".

## ❑ Đối tượng SqlCommand:

- ❖ cho phép bạn chọn kiểu tương tác mà bạn muốn thực hiện với database.
- ❖ có thể thực hiện các lệnh select, insert, modify, và delete các dòng trong một table của database
- ❖ Đối tượng này có thể được dùng để hỗ trợ mô hình quản lý dữ liệu ngắt kết nối (disconnected)



❑ Đối tượng Command có một số phương thức sau:

- ❖ ExecuteScalar(): Thực hiện câu lệnh mà kết quả trả về chỉ có 1 ô (Ví dụ câu lệnh Select Count(\*)...).
- ❖ ExecuteReader(): Thực hiện câu lệnh Select và trả về một DataReader
- ❖ ExecuteNonQuery(): Thực hiện câu lệnh OLEDB nhưng không trả về kết quả (Delete, Update, Insert ...).
- ❖ ExecuteXmlReader(): Tạo một bộ đọc từ file XML. Phương thức này không có trong OleDbCommand, chỉ có trong SqlCommand

## ❑ Ví dụ tạo database tên "ComputerShop"

```
static void Main(string[] args)
{
    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial Catalog=master;
    string query = "Create Database ComputerShop";
    SqlCommand cmd = new SqlCommand(query, con);
    try
    {
        con.Open();
        cmd.ExecuteNonQuery();
        Console.WriteLine("Database Created Successfully");
    }
    catch(SqlException e)
    {
        Console.WriteLine("Error Generated. Details: " + e.ToString());
    }
    finally
    {
        con.Close();
        Console.ReadKey();
    }
}
```

## ❑ Ví dụ đổi tên database tên "ComputerShop" thành MobileShop

```
static void Main(string[] args)
{
    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial Catalog=master;
    string query = "ALTER DATABASE ComputerShop MODIFY NAME = MobileShop";
    SqlCommand cmd = new SqlCommand(query, con);
    try
    {
        con.Open();
        cmd.ExecuteNonQuery();
        Console.WriteLine("Database Renamed Successfully");
    }
    catch(SqlException e)
    {
        Console.WriteLine("Error Generated. Details: " + e.ToString());
    }
    finally
    {
        con.Close();
        Console.ReadKey();
    }
}
```



- ❑ Querying Data – Truy vấn dữ liệu: dùng phương thức `ExecuteReader()` để trả về một đối tượng

`SqlDataReader`

```
// 1. Instantiate a new command with a query and connection
SqlCommand cmd = new SqlCommand("select CategoryName
from Categories", conn);
```

```
// 2. Call Execute reader to get query result
SqlDataReader rdr = cmd.ExecuteReader();
```

- ❑ Inserting Data – Chèn dữ liệu: dùng phương thức `ExecuteNonQuery()` của đối tượng `SqlCommand`

```
// prepare command string
string insertString = @"
insert into Categories
(CategoryName, Description)
values ('Miscellaneous', 'Whatever doesn't fit elsewhere');"
```

```
// 1. Instantiate a new command with a query and connection
SqlCommand cmd = new SqlCommand(insertString, conn);
```

```
// 2. Call ExecuteNonQuery to send command
cmd.ExecuteNonQuery();
```

## □ Updating Data – Cập nhật dữ liệu

```
// prepare command string
string updateString = @"
    update Categories
    set CategoryName = 'Other'
    where CategoryName = 'Miscellaneous'";

// 1. Instantiate a new command with command text only
SqlCommand cmd = new SqlCommand(updateString);

// 2. Set the Connection property
cmd.Connection = conn;

// 3. Call ExecuteNonQuery to send command
cmd.ExecuteNonQuery();
```

## □ Deleting Data – Xóa dữ liệu

```
// prepare command string
string deleteString = @"
    delete from Categories
    where CategoryName = 'Other'";

// 1. Instantiate a new command
SqlCommand cmd = new SqlCommand();

// 2. Set the CommandText property
cmd.CommandText = deleteString;

// 3. Set the Connection property
cmd.Connection = conn;

// 4. Call ExecuteNonQuery to send command
cmd.ExecuteNonQuery();
```



# DEMO

-Demo Truy vấn dữ liệu, thêm, xóa,  
Sửa dữ liệu bảng Employees trong  
CSDL northwind



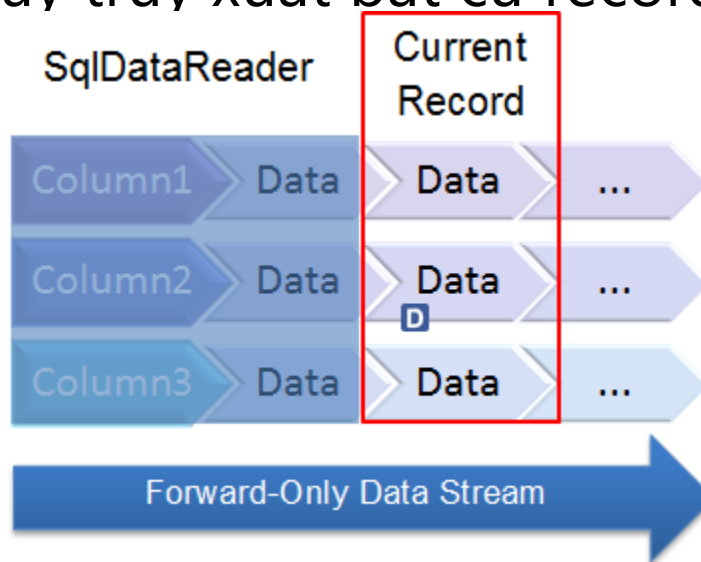


# **LẬP TRÌNH C# 3**

## **BÀI 3: LẬP TRÌNH CƠ SỞ DỮ LIỆU(P2)**

## ❑ Đọc dữ liệu với SqlDataReader:

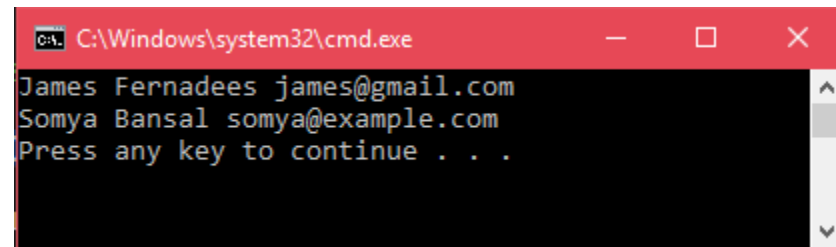
- ❖ Đối tượng DataReader được .NET Framework cung cấp nhằm phục vụ việc truy cập vào cơ sở dữ liệu nhanh và hiệu quả cụ thể là việc đọc dữ liệu từ Database
- ❖ DataReader chỉ xử lý 1 record tại một thời điểm và chỉ được truy xuất 1 chiều, không có các thao tác phức tạp như sắp xếp hay truy xuất bất cứ record ngẫu nhiên nào



## ❑ Đọc dữ liệu với SqlDataReader:

```
static void Main(string[] args)
{
    new Program().GetData();
}

public void GetData()
{
    SqlConnection con = null;
    try
    {
        // Creating Connection
        con = new SqlConnection("data source=.; database=student; integrated security=SSPI");
        // writing sql query
        SqlCommand cm = new SqlCommand("select * from student", con);
        // Opening Connection
        con.Open();
        // Executing the SQL query
        SqlDataReader sdr = cm.ExecuteReader();
        while (sdr.Read())
        {
            Console.WriteLine(sdr["name"] + " " + sdr["email"]);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("Oops, something went wrong." + e);
    }
    // Closing the connection
    finally
    {
        con.Close();
    }
}
```



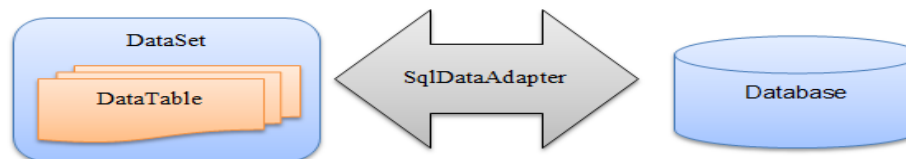
```
C:\Windows\system32\cmd.exe
James Fernadees james@gmail.com
Somya Bansal somya@example.com
Press any key to continue . . .
```

## ❑ Làm việc với Disconnected Data – DataSet và SqlDataAdapter

- ❖ Một 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.
- ❖ 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.

## ❑ Làm việc với Disconnected Data – DataSet và SqlDataAdapter

- ❖ Một 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.
- ❖ 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.





## ❑ Các thuộc tính Dataset

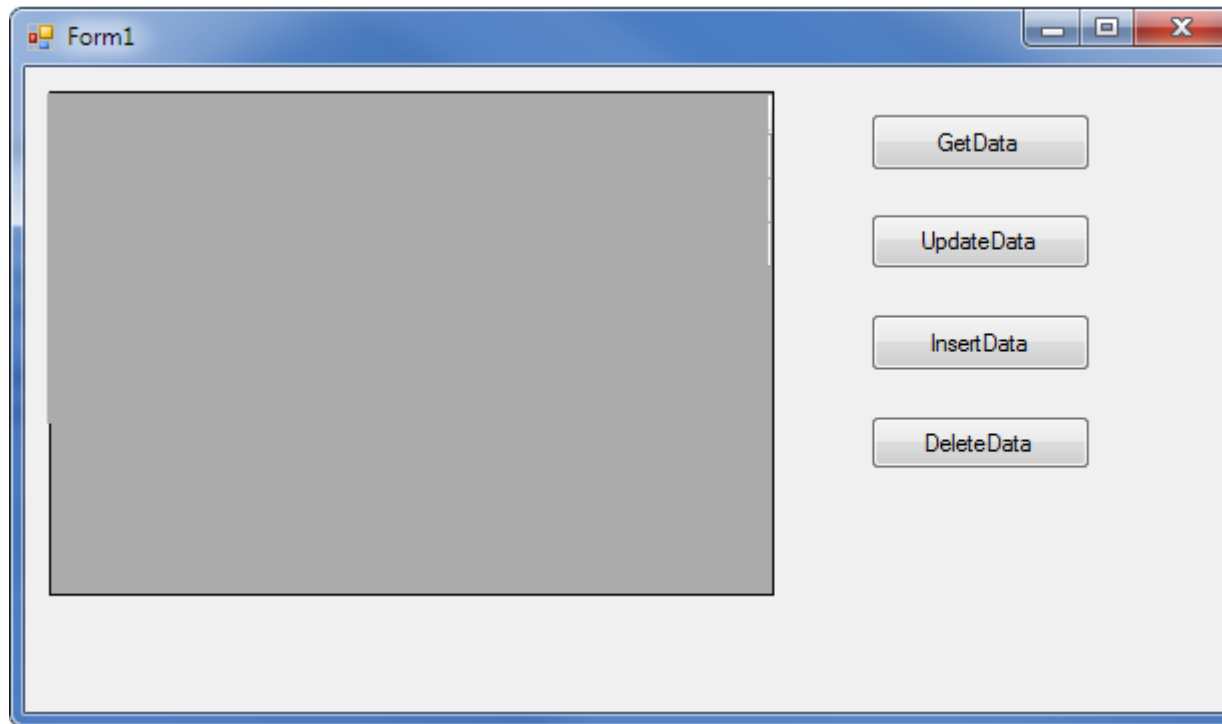
Properties	Description
<code>DeleteCommand</code>	It is used for Deleting Records from DataSource
<code>InsertCommand</code>	It is used for adding New Record to a DataSource
<code>SelectCommand</code>	It is used for Selecting Records from a DataSource
<code>UpdateCommand</code>	It is used for Updating Records in a DataSource.
<code>TableMapping</code>	It is used for mapping actual database tables and datasets.

## ❑ Các method của Dataset

Method	Description
<code>Fill</code>	This method Fills Records from DataAdapters to DataSets.
<code>Update</code>	This method update DataSource with DataSets.

## ❑ Ví dụ DATASET và GRID VIEW

### ❖ Thiết kế form



- ❑ Ví dụ DATASET và GRID VIEW
  - ❖ Đổ dữ liệu DataSet tới GridView

The screenshot shows a Windows Form titled 'Form1'. Inside the form, there is a GridView displaying a table of product data. The table has four columns: ID, Name, Price, and Date. The first two rows are populated with data, and the third row is a new, empty row indicated by an asterisk in the first column. To the right of the GridView, there are four buttons stacked vertically: 'GetData', 'UpdateData', 'InsertData', and 'DeleteData'.

	ID	Name	Price	Date
▶	1	LED Screen	\$120	27-01-2017
	2	USB Keyboard	\$20	25-05-2017
*				

## ❑ Ví dụ DATASET và GRID VIEW

### ❖ Adding New Row in DataTable

private

Form1

	ID	Name	Price	Date
▶	1	LED Screen	\$120	27-01-2017
	2	New Printer	\$20	25-05-2017
		4GB DDR3 RAM	\$50	26-05-2017
★				

GetData

UpdateData

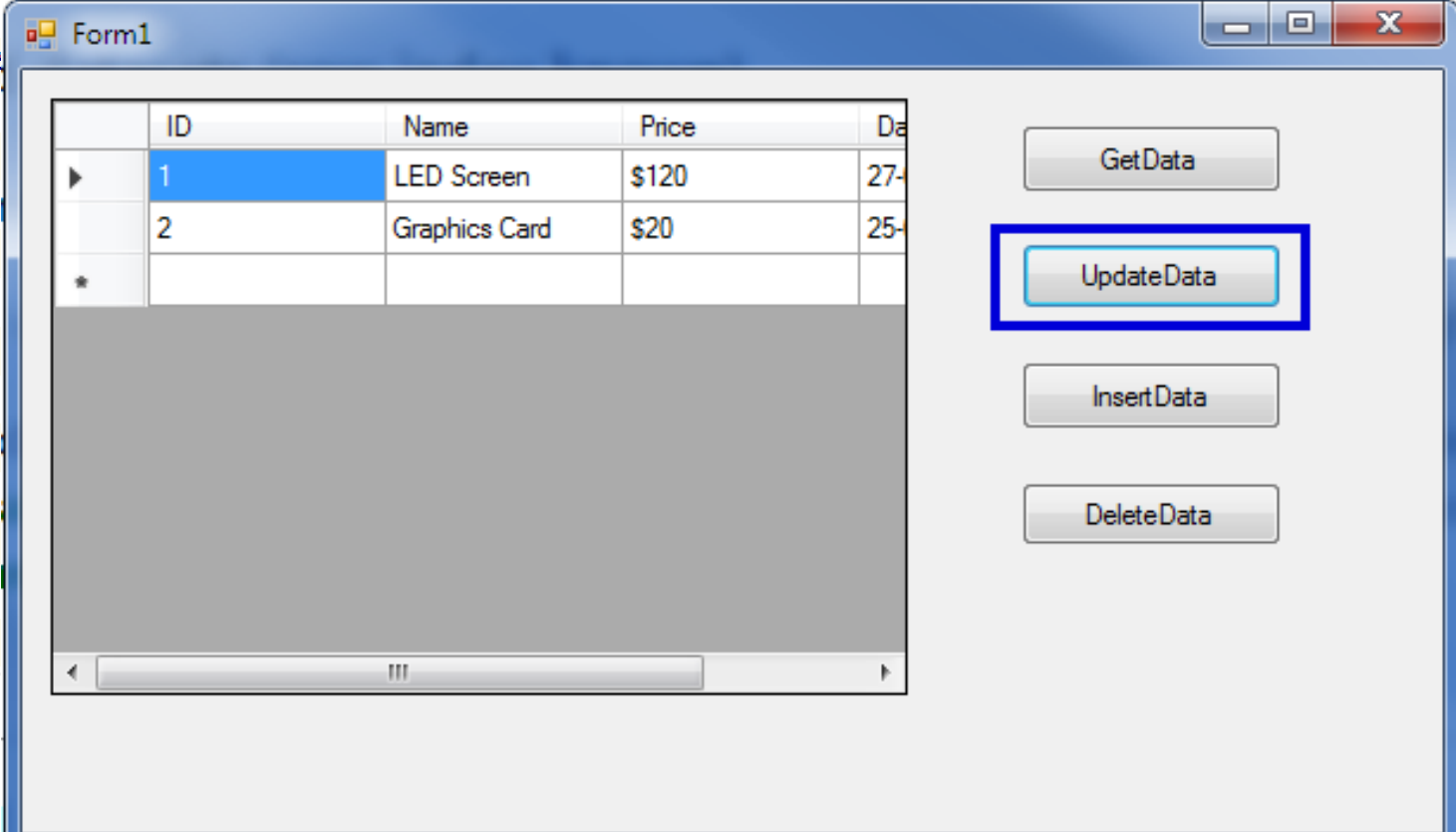
InsertData

DeleteData

## ❑ Ví dụ DATASET và GRID VIEW

### ❖ Edit or Update Row in DataSet

private



	ID	Name	Price	Date
▶	1	LED Screen	\$120	27-
	2	Graphics Card	\$20	25-
*				

op; Integrat

## ❑ Ví dụ DATASET và GRID VIEW

### ❖ Delete Row in DataSet

```
private void btnDelete_Click(object sender, EventArgs e)
{
    //Fill Dataset
    string ConString = @"Data Source=.\SQLEXPRESS;Initial Catalog=ComputerShop;Integrated Security=True;User=sa;Password=1234567890";
    string Query = "SELECT * FROM Items";
    SqlDataAdapter adapter = new SqlDataAdapter(Query, ConString);
    DataSet set = new DataSet();
    adapter.Fill(set, "Items");

    set.Tables["Items"].Rows[1].Delete();

    dataGridView1.DataSource = set.Tables["Items"];
}
```

## ❑ Ví dụ DATASET và GRID VIEW

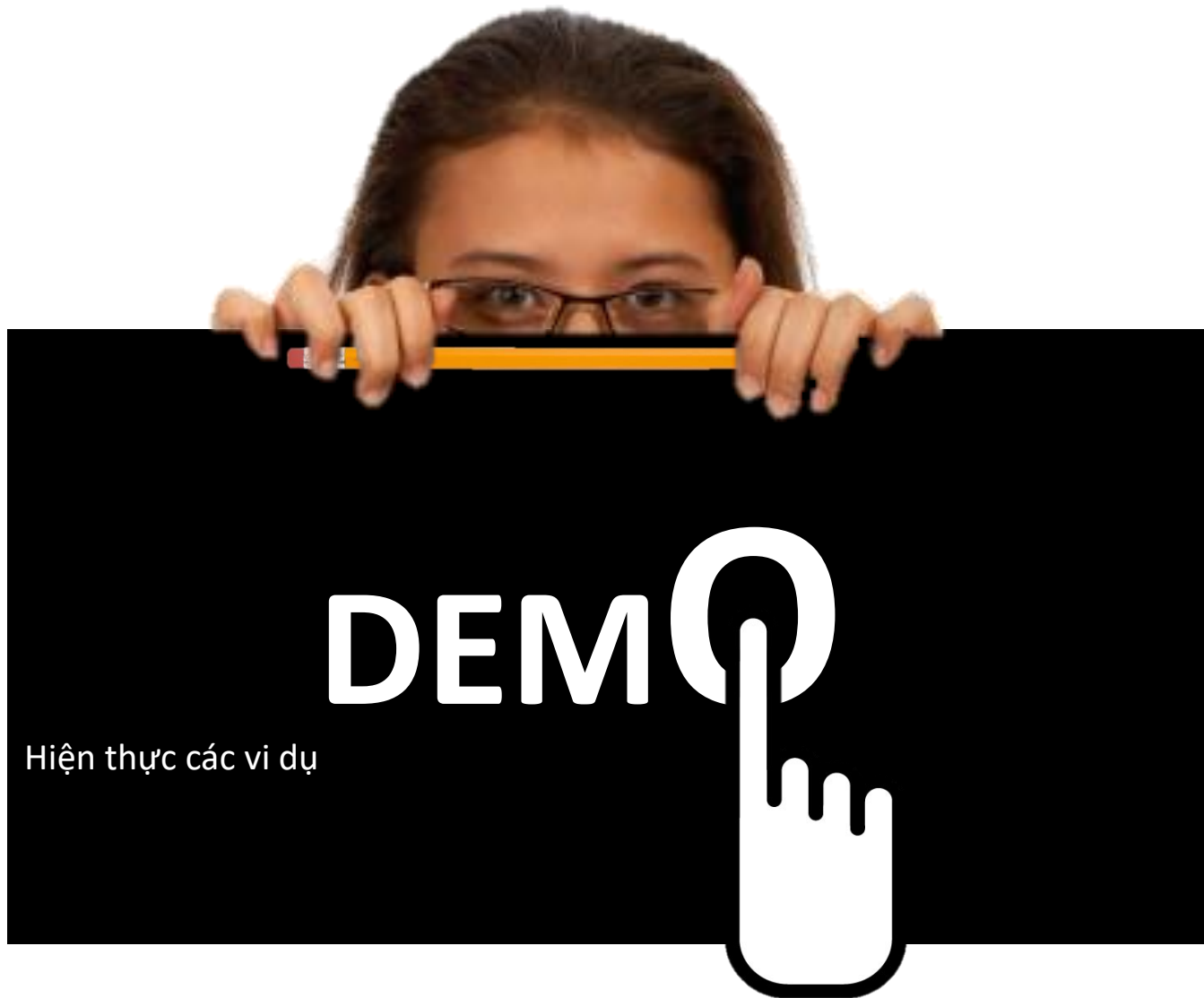
### ❖ SAVE DATASET CHANGES TO DATABASE

```
private void btnSave_Click(object sender, EventArgs e)
{
    //Fill Dataset
    string ConString = @"Data Source=.\SQLEXPRESS;Initial Catalog=ComputerShop;Integrated Security=True;User=root;Password=1234567890";
    string Query = "SELECT * FROM Items";
    SqlDataAdapter adapter = new SqlDataAdapter(Query, ConString);
    DataSet set = new DataSet();
    adapter.Fill(set, "Items");

    //Adding New Row to DataSet and Update
    DataRow row = set.Tables["Items"].NewRow();
    row["Name"] = "4GB DDR3 RAM";
    row["Price"] = "$50";
    row["Date"] = "26 May 2017";
    set.Tables["Items"].Rows.Add(row);

    //Updating Database Table
    SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
    adapter.Update(set.Tables["Items"]);

    MessageBox.Show("DataSet Saved to Database Successfully");
}
```





## Tổng kết bài học

◎ TỔNG QUAN VỀ ADO.NET

◎ Các đối tượng của ADO.NET





**KẾT THÚC**