

# LẬP TRÌNH MẠNG CĂN BẢN

Biên soạn: ThS. Đỗ Thị Hương Lan



TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN - ĐHQG-HCM  
**KHOA MẠNG MÁY TÍNH & TRUYỀN THÔNG**  
FACULTY OF COMPUTER NETWORK AND COMMUNICATIONS

Tầng 8 - Tòa nhà E, trường ĐH Công nghệ Thông tin, ĐHQG-HCM

# **Chương 2**

# **Luồng dữ liệu (Stream)**

# **và I/O**

# Nội dung chi tiết

---

- Giới thiệu về I/O, Streams
- Đọc & ghi dữ liệu với FileStream
- Đọc & ghi dữ liệu với StreamReader & StreamWriter
- Kết nối ứng dụng với cơ sở dữ liệu

# Nội dung chi tiết

---

- Giới thiệu về I/O, Streams
- Đọc & ghi dữ liệu với FileStream
- Đọc & ghi dữ liệu với StreamReader & StreamWriter
- Kết nối ứng dụng với cơ sở dữ liệu

# I/O – Input&Output

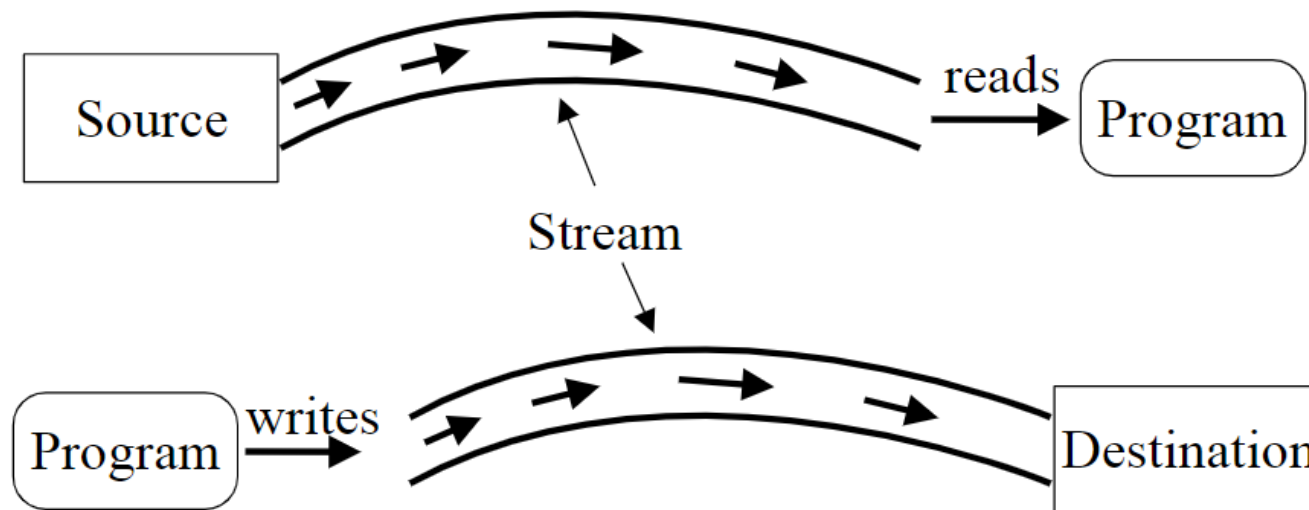
- **I/O** là vấn đề rất quan trọng đối với truyền thông trên mạng
- **Luồng** (Stream) là luồng của thông tin, chứa thông tin sẽ được chuyển qua, còn tập tin (**File**) thì để lưu trữ thông tin, dữ liệu. File và Stream I/O (Input/Output) đề cập đến việc truyền dữ liệu đến hoặc từ một phương tiện lưu trữ.
- Trong .NET Framework, namespace **System.IO** bao gồm các loại có khả năng đọc và ghi (đồng bộ và không đồng bộ) đối với các luồng dữ liệu và file. Những namespace này cũng chứa các loại namespace thực hiện việc nén và giải nén các file.

# Streams

- Kiến trúc dựa trên Stream đã được phát triển trong .NET
- Các thiết bị I/O bao gồm từ máy in, đĩa cứng cho đến card mạng
- Không phải các thiết bị đều có chức năng giống nhau → Stream cũng không hỗ trợ các phương thức giống nhau
- `canRead()`, `canSeek()`, `canWrite()` chỉ khả năng Stream ứng với thiết bị cụ thể

# Streams

- Dữ liệu được truyền theo **hai** hướng
  - **Đọc dữ liệu**: đọc dữ liệu từ bên ngoài vào chương trình.
  - **Ghi dữ liệu**: đưa dữ liệu từ chương trình ra bên ngoài.



# Streams trong C#

- Lớp Stream hỗ trợ **đọc** và **ghi** byte
- Tất cả các lớp đại diện cho các luồng đều kế thừa từ lớp Stream.
- Lớp Stream và các lớp dẫn xuất của nó cung cấp một cái nhìn chung về các nguồn dữ liệu và giúp lập trình viên không cần phải đi quá chi tiết về các đặc điểm của hệ điều hành và các thiết bị bên dưới.



# Streams trong C#

- Streams bao hàm ba thao tác cơ bản:
  - **Đọc**: đưa dữ liệu từ một luồng vào một cấu trúc dữ liệu, chẳng hạn như một mảng byte
  - **Ghi**: đưa dữ liệu vào một luồng từ một nguồn dữ liệu
  - **Tìm kiếm**: truy vấn và sửa đổi vị trí hiện tại trong một luồng

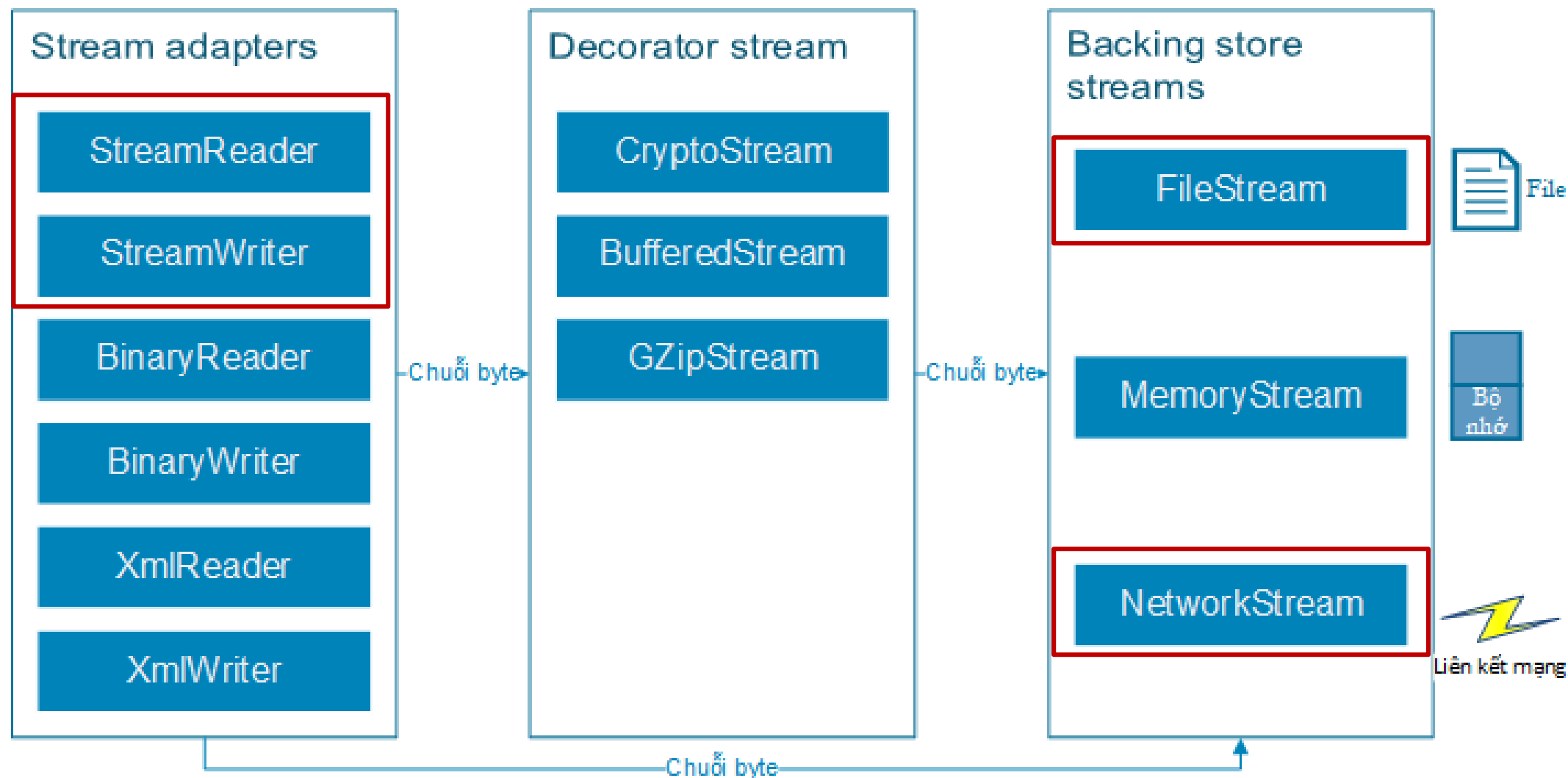
# Streams trong C#

- Hai stream quan trọng: **networkStream** và **fileStream**
- Hai cách dùng stream: **đồng bộ** và **bất đồng bộ**
- Khi dùng đồng bộ: luồng (thread) tương ứng sẽ tạm ngưng đến khi tác vụ hoàn thành hoặc lỗi
- Khi dùng bất đồng bộ: luồng (thread) tương ứng sẽ ngay tức thì quay về phương thức gọi nó và bất cứ lúc nào tác vụ hoàn thành sẽ có dấu hiệu chỉ thị, hoặc lỗi xảy ra

# Đồng bộ và bất đồng bộ

- Kiểu chương trình “treo” để chờ tác vụ hoàn thành không “thân thiện” cho lắm, do đó phương thức gọi đồng bộ phải dùng một luồng riêng
- Bằng cách dùng các luồng và phương thức gọi bất đồng bộ làm cho có cảm giác máy tính có thể làm được nhiều việc cùng lúc. Thực tế, hầu hết máy tính chỉ có 1 CPU, nên điều trên đạt được là do chuyển giữa các tác vụ trong khoảng một vài milliseconds

# Kiến trúc Stream trong .NET



# Nội dung chi tiết

---

- Giới thiệu về I/O, Streams
- **Đọc & ghi dữ liệu với FileStream**
- Đọc & ghi dữ liệu với StreamReader & StreamWriter
- Kết nối ứng dụng với cơ sở dữ liệu

# FileStream

- Cung cấp Stream cho File, hỗ trợ cả thao tác đọc và ghi đồng bộ và không đồng bộ.
- Để sử dụng: khai báo namespace **System.IO**

C#

 Copy

```
public class FileStream : System.IO.Stream
```

# FileStream

Phương thức hoặc thuộc tính	Mục đích
Constructor	Khởi tạo một thực thể mới của FileStream
Length	Độ dài của file, giá trị kiểu long
Position	Lấy ra hoặc thiết lập vị trí của con trỏ file, giá trị kiểu long
BeginRead()	Bắt đầu đọc bất đồng bộ
BeginWrite()	Bắt đầu ghi bất đồng bộ
Write()	Ghi một khối byte vào stream dùng dữ liệu trong bộ đệm
Read()	Đọc một khối byte từ stream và ghi vào trong bộ đệm
Lock()	Ngăn cản việc các tiến trình khác truy xuất vào tất cả hoặc một phần của file

# FileStream Constructor

`FileStream(String, FileMode, FileAccess, FileShare)`

Append
Create
CreateNew
Open
OpenOrCreate
Truncate

Read
ReadWrite
Write

Delete
Inheritable
None
Read
ReadWrite
Write

```
FileStream fs = new FileStream("demo.txt", FileMode.OpenOrCreate, FileAccess.ReadWrite, FileShare.Read)
```



# FileStream – đọc dữ liệu

```
// Cách đọc sử dụng FileStream
private void btnFileStream_Click(object sender, EventArgs e)
{
    // Hiển thị Hộp thoại OpenFileDialog cho phép chọn 1 file
    // Thuộc tính FileName của OpenFileDialog trả về đường dẫn của file
    ofd.ShowDialog();
    fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate);
    bytes = new byte[fs.Length];
    fs.Read(bytes, 0, (int)fs.Length);
    content = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
    richTextBox1.Text = content;
    fs.Close();
}
```

C#

```
public override int Read (byte[] array, int offset, int count);
```

# FileStream – đọc dữ liệu (Bất đồng bộ)

```
// Cách đọc sử dụng FileStream (đọc bất đồng bộ)
private async void btnFileReadAsync_Click(object sender, EventArgs e)
{
    ofd.ShowDialog();
    using (FileStream fs = File.Open(ofd.FileName, FileMode.Open))
    {
        bytes = new Byte[fs.Length];
        await fs.ReadAsync(bytes, 0, (int)fs.Length);
    }
    content = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
    richTextBox1.Text = content;
}
```

C#

```
public override System.Threading.Tasks.Task<int> ReadAsync (byte[] buffer, int offset, int
count, System.Threading.CancellationToken cancellationToken);
```

# Encoding data

- Trong ví dụ trước ta đã dùng **Encoding.UTF8.GetString()** để chuyển đổi một mảng byte thành string.
- Các dạng hợp lệ là Unicode (Encoding.Unicode), ASCII, UTF7
- UTF8 dùng 1 byte cho một ký tự, Unicode dùng 2 byte cho mỗi ký tự

# FileStream – ghi dữ liệu (đồng bộ)

```
1 using System;
2 using System.IO;
3
4 class FStream
5 {
6     static void Main()
7     {
8         const string fileName = "Test###.dat";
9         // Create random data to write to the file.
10        byte[] dataArray = new byte[100000];
11        new Random().NextBytes(dataArray);
12        using(FileStream fileStream = new FileStream(fileName, FileMode.Create))
13        {
14            // Write the data to the file, byte by byte.
15            for(int i = 0; i < dataArray.Length; i++)
16            {
17                fileStream.WriteByte(dataArray[i]);
18            }
19            // Set the stream position to the beginning of the file.
20            fileStream.Seek(0, SeekOrigin.Begin);
21            // Read and verify the data.
22            for(int i = 0; i < fileStream.Length; i++)
23            {
24                if(dataArray[i] != fileStream.ReadByte())
25                {
26                    Console.WriteLine("Error writing data.");
27                    return;
28                }
29            }
30            Console.WriteLine("The data was written to {0} " + "and verified.", fileStream.Name);
31        }
32    }
33 }
```

# FileStream – ghi dữ liệu (bất đồng bộ)

```
1 using System;
2 using System.Text;
3 using System.Threading.Tasks;
4 using System.Windows;
5 using System.Windows.Controls;
6 using System.IO;
7
8 namespace WpfApplication1
9 {
10     public partial class MainWindow : Window
11     {
12         public MainWindow()
13         {
14             InitializeComponent();
15         }
16         private async void Button_Click(object sender, RoutedEventArgs e)
17         {
18             UnicodeEncoding uniencoding = new UnicodeEncoding();
19             string filename = @"c:\Users\exampleuser\Documents\userinputlog.txt";
20             byte[] result = uniencoding.GetBytes(UserInput.Text);
21             using (FileStream SourceStream = File.Open(filename, FileMode.OpenOrCreate))
22             {
23                 SourceStream.Seek(0, SeekOrigin.End);
24                 await SourceStream.WriteAsync(result, 0, result.Length);
25             }
26         }
27     }
28 }
```

# Nội dung chi tiết

---

- Giới thiệu về I/O, Streams
- Đọc & ghi dữ liệu với FileStream
- Đọc & ghi dữ liệu với StreamReader & StreamWriter
- Kết nối ứng dụng với cơ sở dữ liệu

# StreamReader

- **StreamReader** kế thừa từ **TextReader (abstract class)**
- StreamReader và Stream
  - Stream được thiết kế cho đầu vào và đầu ra byte
  - StreamReader nhận input đầu vào là ký tự → Đọc file văn bản

C#

```
public class StreamReader : System.IO.TextReader
```

# StreamReader

Phương thức hoặc Thuộc tính	Mục đích
Constructor	Khởi tạo một thực thể mới của StreamReader
Peek	Trả về ký tự kế tiếp, hoặc giá trị -1 nếu đến cuối stream
Read	Đọc ký tự kế tiếp hoặc một tập các ký tự từ input stream
ReadBlock	Đọc các ký tự từ stream hiện hành và ghi dữ liệu vào bộ đệm, bắt đầu tại vị trí chỉ định
ReadLine	Đọc một dòng ký tự từ stream hiện hành và trả về dưới dạng string
ReadToEnd	Đọc từ vị trí hiện hành đến cuối stream.
ReadAsync	Đọc bất đồng bộ từ stream vào Bộ nhớ hoặc vào đệm



# StreamReader – đọc dữ liệu (đồng bộ)

```
// Cách đọc sử dụng StreamReader
private void btnStreamReader_Click(object sender, EventArgs e)
{
    ofd.ShowDialog();
    StreamReader str = new StreamReader(ofd.FileName);
    content = str.ReadToEnd();
    richTextBox1.Text = content;
    str.Close();
}
```

# StreamReader – đọc dữ liệu (bất đồng bộ)

```
// Cách đọc sử dụng StreamReader (Bất đồng bộ)
private async void btnStreamReadAsync_Click(object sender, EventArgs e)
{
    ofd.ShowDialog();
    using (StreamReader sr = new StreamReader(ofd.FileName))
    {
        Char[] bytes = new Char[(int)sr.BaseStream.Length];
        await sr.ReadAsync(bytes, 0, (int)sr.BaseStream.Length);
    }
    content = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
    richTextBox1.Text = content;
    str.Close();
}
```

# StreamWriter

- **StreamWriter** kế thừa từ **TextWriter (abstract class)**
- StreamWriter và Stream
  - Stream được thiết kế cho đầu vào và đầu ra byte
  - StreamWriter có đầu ra là ký tự → Ghi file văn bản

C#

```
public class StreamWriter : System.IO.TextWriter
```

# StreamWriter

<b>Close()</b>	Đóng đối tượng StreamWriter hiện tại và Underlying Stream
<b>Flush()</b>	Xóa tất cả buffer cho Writer hiện tại và làm cho bất kỳ dữ liệu được đệm nào để được ghi tới Underlying Stream
<b>Dispose()</b>	Giải phóng tất cả các tài nguyên được sử dụng bởi đối tượng StreamWriter (Kế thừa từ StreamWriter)

# StreamWriter

**Write(char value)**

Ghi một ký tự tới Stream

**Write(string value)**

Ghi một chuỗi (string) tới Stream

**WriteLine()**

Ghi một line terminator tới Text string hoặc stream

**WriteAsync(String)**

Ghi bất đồng bộ viết một chuỗi (string) vào stream

**WriteAsync(Char[])**

Ghi bất đồng bộ một mảng ký tự vào luồng văn bản không đồng bộ

**WriteAsync(Char[], Int32, Int32)**

Ghi bất đồng bộ một mảng ký tự vào luồng từ một vị trí index

# StreamWriter – ghi dữ liệu (đồng bộ)

```
using System;
using System.IO;

class Program
{
    static void Main(string[] args)
    {
        // Create a string array with the lines of text
        string[] lines = { "First line", "Second line", "Third line" };

        // Set a variable to the Documents path.
        string docPath =
            Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        // Write the string array to a new file named "WriteLines.txt".
        using (StreamWriter outputFile = new StreamWriter(Path.Combine(docPath, "WriteLines.txt")))
        {
            foreach (string line in lines)
                outputFile.WriteLine(line);
        }
    }
}
```

# StreamWriter – ghi dữ liệu (bất đồng bộ)

```
using System;
using System.IO;
using System.Threading.Tasks;

class Program
{
    static async Task Main()
    {
        // Set a variable to the Documents path.
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        // Write the specified text asynchronously to a new file named "WriteTextAsync.txt".
        using (StreamWriter outputFile = new StreamWriter(Path.Combine(docPath, "WriteTextAsync.txt")))
        {
            await outputFile.WriteLineAsync("This is a sentence.");
        }
    }
}

// The example creates a file named "WriteTextAsync.txt" with the following contents:
// This is a sentence.
```

# Nội dung chi tiết

---

- Giới thiệu về I/O, Streams
- Đọc & ghi dữ liệu với FileStream
- Đọc & ghi dữ liệu với StreamReader & StreamWriter
- Kết nối ứng dụng với cơ sở dữ liệu



# Tổng quan Lập trình kết nối CSDL

- Có 2 chủ đề tập trung:
  - **Chuỗi kết nối**: chỉ vị trí và kiểu của dữ liệu
  - **Các phát biểu SQL**: mô tả hoạt động đối với dữ liệu
- Sử dụng namespace phù hợp
  - **Microsoft Excel**: `using Excel = Microsoft.Office.Interop.Excel;`
  - **Microsoft Word**: `using Word = Microsoft.Office.Interop.Word;`
  - **Microsoft Access**: ví dụ - `System.Data.OleDb`
  - **SQL Server**: ví dụ - `System.Data.SqlClient`

# Chuỗi kết nối

Kiểu Database	Chuỗi kết nối
SQL Server	Data Source=<tên máy chủ>/<địa chỉ IP>; Initial Catalog=<tên database >; User ID=<user>; Password=<password>; Persist Security Info=True/False; Integrated security=True/False; <i>Trong đó: Integrated security=True thì sử dụng Windows Authentication</i> <i>Persist security info: Thiết lập mặc định cho persist security info là false. Thiết lập sang giá trị true sẽ cho phép các dữ liệu nhạy cảm bao gồm UserID và password có thể đọc được khi kết nối mở</i>

Connection	
Connection String	Data Source=ASUS;Initial Catalog=TestNetProg;User ID=test;Password=*****
Provider	.NET Framework Data Provider for SQL Server

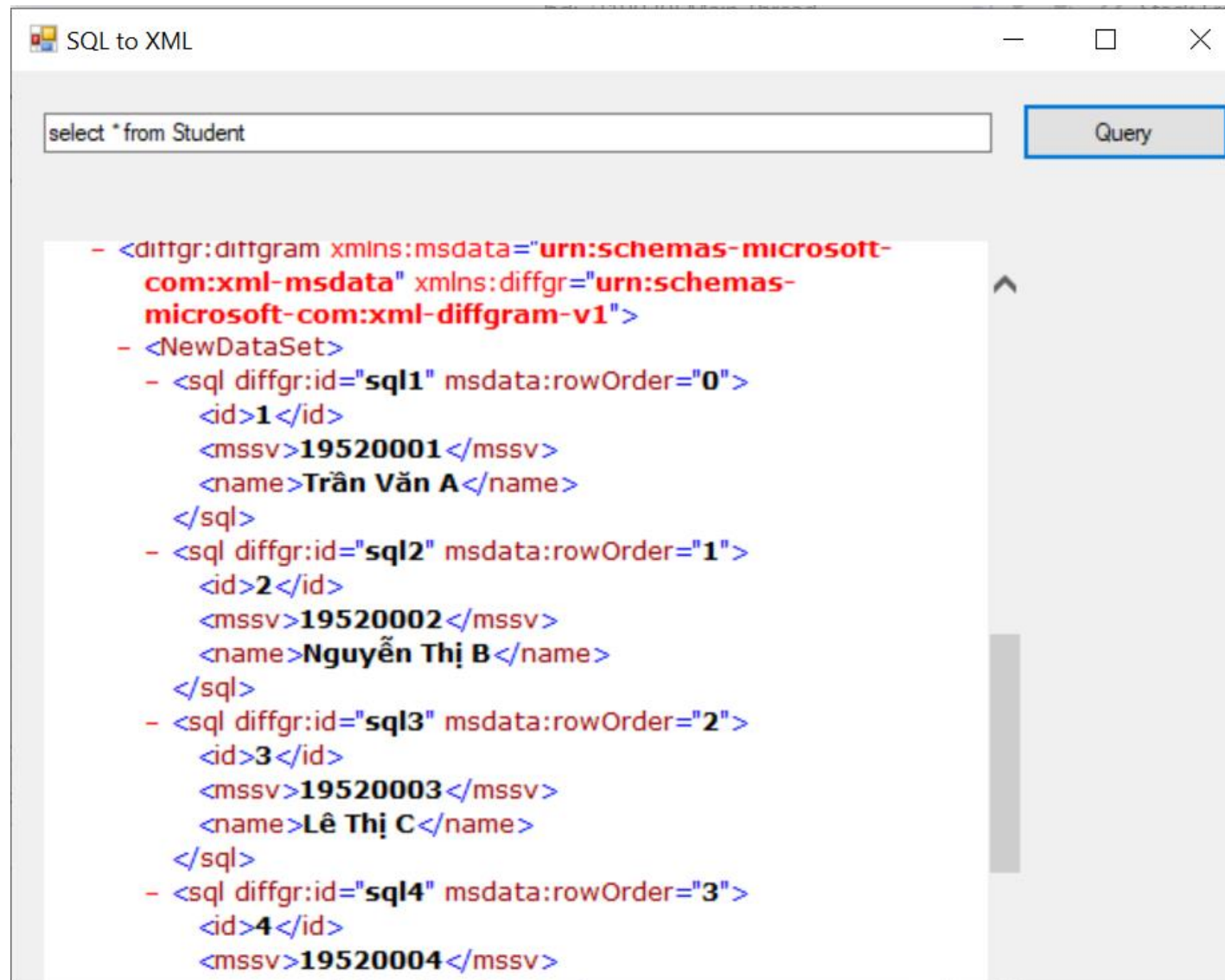
# Bốn thao tác chính

- Đọc dữ liệu ra: lệnh **Select**
- Thêm các dòng dữ liệu mới vào: lệnh **Insert**
- Xóa bỏ các dòng dữ liệu: lệnh **Delete**
- Cập nhật thông tin cho các dòng đã có: lệnh **Update**

# Ví dụ các thao tác

- Lệnh **Select** tổng quát có dạng:  
Select \* from table where column='điều kiện lọc'
- Lệnh **Update** tổng quát có dạng:  
Update table set column='dữ liệu mới' where column='điều kiện lọc'
- Lệnh **Delete** tổng quát có dạng:  
Delete from table where column='điều kiện lọc'
- Lệnh **Insert** tổng quát có dạng:  
Insert into table (column) values ('dữ liệu mới')

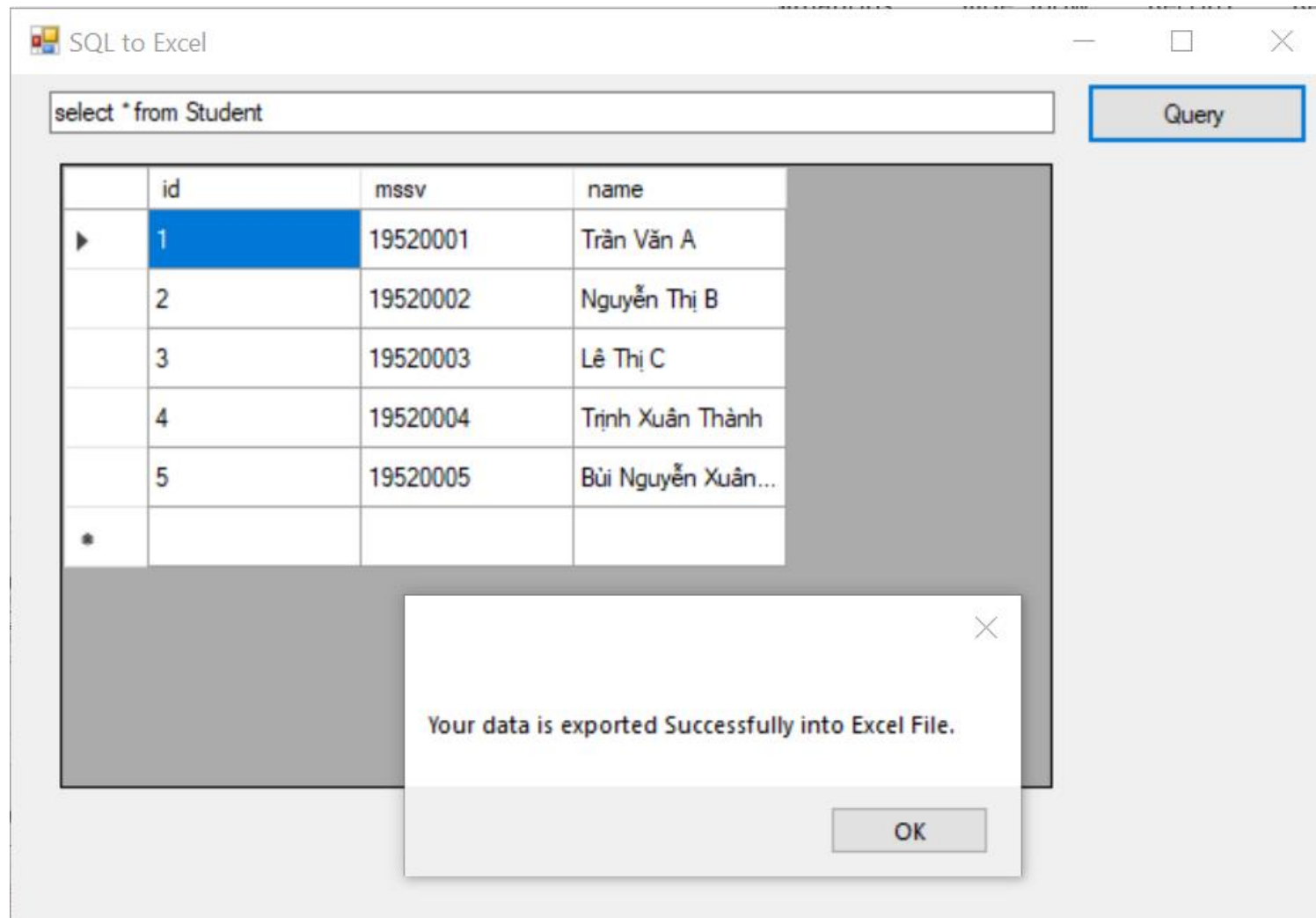
# Ví dụ: SQL to XML



The screenshot shows a window titled "SQL to XML" with a text input field containing the SQL query "select \* from Student" and a "Query" button. Below the input field, the resulting XML is displayed in a scrollable area. The XML is a DiffGram structure with four rows of student data.

```
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet>
- <sql diffgr:id="sql1" msdata:rowOrder="0">
  <id>1</id>
  <mssv>19520001</mssv>
  <name>Trần Văn A</name>
</sql>
- <sql diffgr:id="sql2" msdata:rowOrder="1">
  <id>2</id>
  <mssv>19520002</mssv>
  <name>Nguyễn Thị B</name>
</sql>
- <sql diffgr:id="sql3" msdata:rowOrder="2">
  <id>3</id>
  <mssv>19520003</mssv>
  <name>Lê Thị C</name>
</sql>
- <sql diffgr:id="sql4" msdata:rowOrder="3">
  <id>4</id>
  <mssv>19520004</mssv>
```

# Ví dụ: SQL to Excel



# Bài tập

---

- Cài đặt các chương trình đã minh họa trong bài giảng của chương bằng ngôn ngữ C#