

# C# Programming

## Lập trình cơ sở dữ liệu với **ADO.NET**

By Hoàng Hữu Việt

Email: [viethh@vinhuni.edu.vn](mailto:viethh@vinhuni.edu.vn)

Viện Kỹ thuật và Công nghệ, Trường Đại học Vinh

Cao Thanh Sơn

Viện Nghiên cứu và Đào tạo trực tuyến, Trường Đại học Vinh

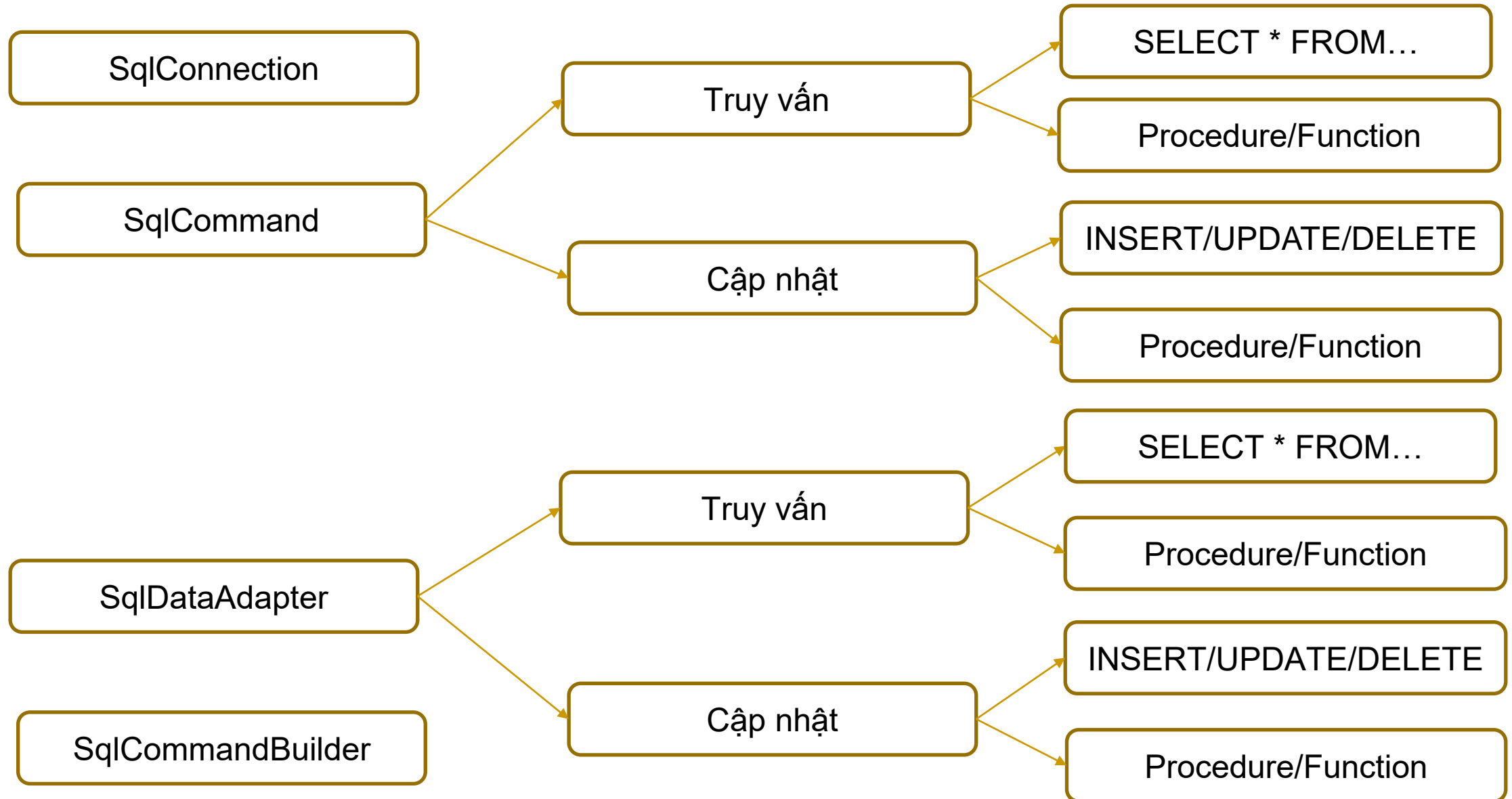
Vinh, 9/2020

# Mục đích, chuẩn đầu ra và nội dung

---

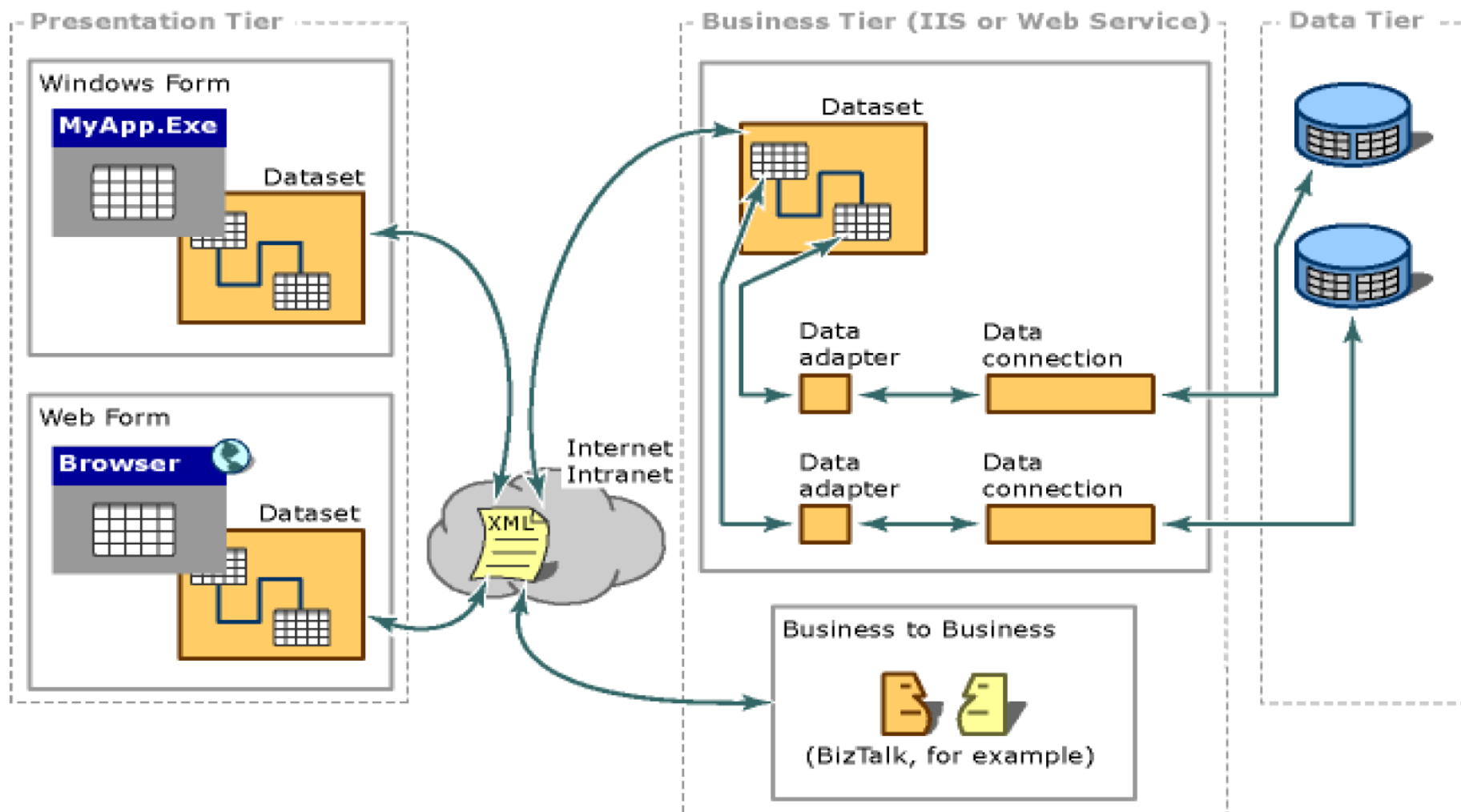
- Mục đích
  - Giới thiệu về lập trình cơ sở dữ liệu với ADO.NET (ActiveX Data Objects)
- Chuẩn đầu ra
  - Lập trình cơ sở dữ liệu sử dụng kỹ thuật ADO.NET
  - Kiểm thử và giải quyết các lỗi của ứng dụng
  - Phát triển kỹ năng lập trình, kỹ năng tìm kiếm và đọc hiểu tài liệu
- Nội dung
  - Giới thiệu về ADO.NET
  - Kết nối với cơ sở dữ liệu dùng lớp SqlConnection
  - Truy vấn và cập nhật dữ liệu dùng lớp SqlCommand
  - Truy vấn và cập nhật dữ liệu dùng lớp SqlDataAdapter
  - Cập nhật dữ liệu dùng lớp SqlCommandBuilder

# Mục đích, chuẩn đầu ra và nội dung



# Giới thiệu ADO.NET

## ■ Microsoft 3-Tier Architecture



# Giới thiệu ADO.NET

---

- Là bước phát triển của ADO (ActiveX Data Objects).
- Làm việc với các cơ sở dữ liệu MS Access, SQL Server, XML, Oracle.
- Gồm một tập các lớp dùng để truy nhập và thao tác với các cơ sở dữ liệu:
  - Kiểu kết nối (connected): gồm các lớp cung cấp dịch vụ quản lý (managed provider) để kết nối trực tiếp với cơ sở dữ liệu và đồng bộ dữ liệu của ứng dụng với cơ sở dữ liệu.
  - Kiểu ngắt kết nối (disconnected): gồm các lớp dữ liệu chung (generic data) được thiết kế để lưu trữ bản sao dữ liệu trong ứng dụng được lấy từ cơ sở dữ liệu.
  - Các lớp dữ liệu chung không phụ thuộc vào cơ sở dữ liệu.

# Giới thiệu ADO.NET

---

- Các lớp cung cấp dịch vụ quản lý:
  - Dùng để kết nối tới cơ sở dữ liệu, truy vấn và cập nhật dữ liệu vào cơ sở dữ liệu.
  - **SQL Server Managed Provider:**
    - Các lớp việc với cơ sở dữ liệu SQL Server.
    - Định nghĩa bằng tiền tố Sql, ví dụ SqlConnection.
  - **OLE DB Managed Provider:**
    - Các lớp dùng để làm việc với các cơ sở dữ liệu hỗ trợ OLEDB như Microsoft Access hoặc Oracle.
    - Định nghĩa bằng tiền tố OleDb, ví dụ OleDbConnection.
  - **ODBC Managed Provider:**
    - Các lớp dùng để làm việc với các cơ sở dữ liệu hỗ trợ ODBC.
    - Định nghĩa bằng tiền tố Odbc, ví dụ OdbcConnection.

# Giới thiệu ADO.NET

---

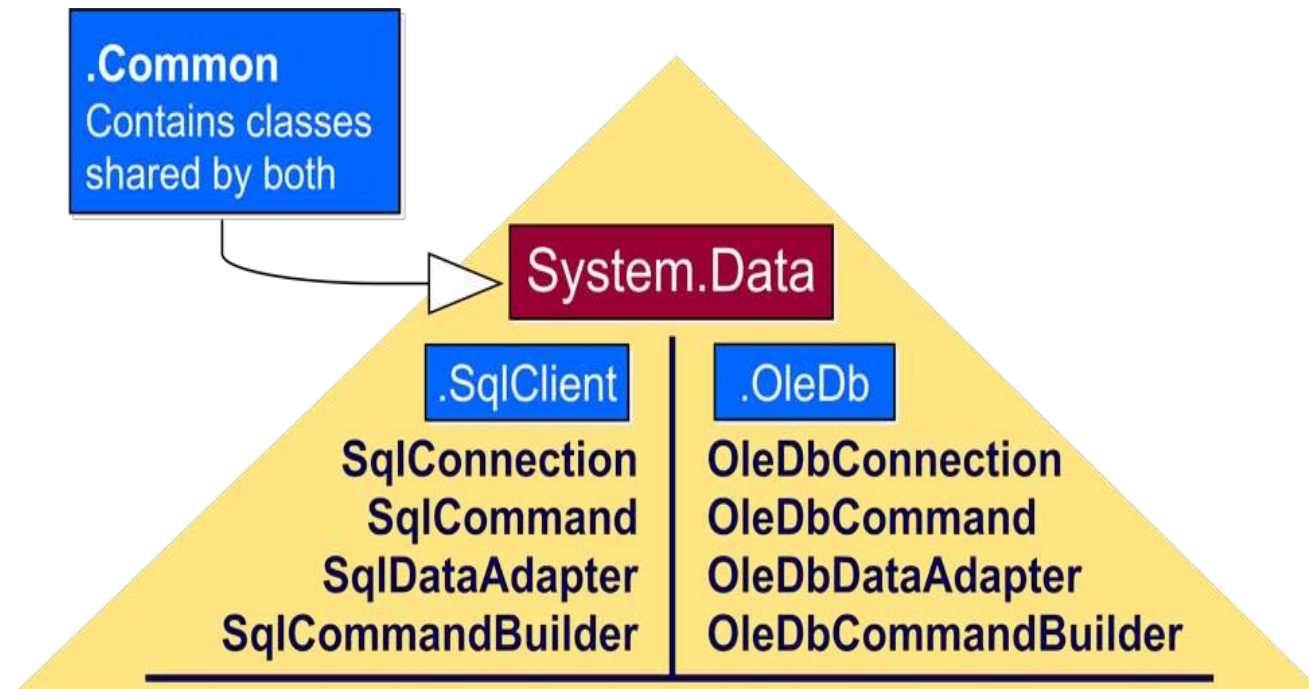
- Các lớp cung cấp dịch vụ quản lý:
  - **Connection**: gồm SqlConnection và OleDbConnection để kết nối tới sơ sở dữ liệu.
  - **Command**: gồm SqlCommand và OleDbCommand để thực hiện một lệnh SQL hoặc dùng để gọi một thủ tục.
  - **DataReader**: gồm SqlDataReader và OleDbDataReader để đọc dữ liệu từ cơ sở dữ liệu.
  - **Parameter**: gồm SqlParameter và OleDbParameter để thiết lập các tham số lớp Command.
  - **DataAdapter**: gồm SqlDataAdapter và OleDbDataAdapter để truy vấn và cập nhật dữ liệu.
  - **CommandBuilder**: gồm SqlCommandBuilder và OleDbCommandBuilder để cập nhật dữ liệu.

# Giới thiệu ADO.NET

---

## ■ Các không gian tên:

- ❑ `System.Data.SqlClient`: Định nghĩa các lớp làm việc với cơ sở dữ liệu Microsoft SQL Server.
- ❑ `System.Data.OleDb`: Định nghĩa các lớp làm việc với cơ sở dữ liệu OLEDB như Microsoft Access.





# Giới thiệu ADO.NET

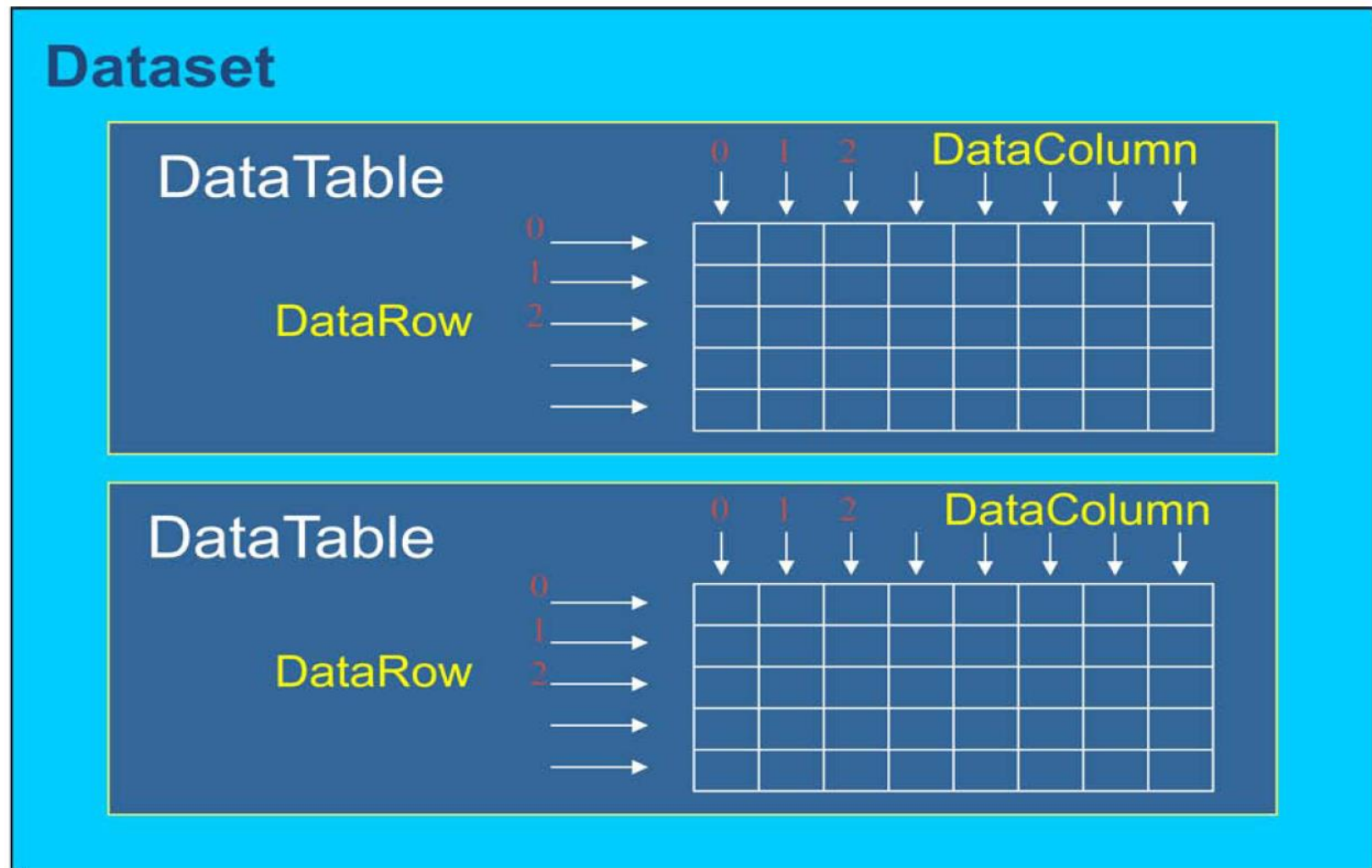
---

## ■ Các lớp dữ liệu chung:

- ❑ Các lớp dữ liệu chung (generic data) làm việc không phụ thuộc vào cơ sở dữ liệu.
- ❑ **DataSet**: Một đối tượng lớp DataSet dùng để lưu trữ một bản sao của một cơ sở dữ liệu trong ứng dụng.
- ❑ **DataTable**: Một đối tượng lớp DataTable được dùng để biểu diễn một bảng dữ liệu bao gồm các hàng và các cột
- ❑ **DataRow**: Một đối tượng lớp DataRow được dùng để biểu diễn một hàng của một bảng dữ liệu.
- ❑ **DataColumn**: Một đối tượng lớp DataColumn được dùng để biểu diễn một cột của một bảng dữ liệu.
- ❑ *Không gian tên cho các lớp dữ liệu chung: System.Data.*

# Giới thiệu ADO.NET

- Các lớp dữ liệu chung:



# Cở sở dữ liệu Books

---

## ■ Bảng Authors

- ❑ AuthorID - int
- ❑ FirstName - nvarchar(25)
- ❑ LastName - nvarchar(15)

## ■ Bảng Publishers

- ❑ PublisherID - int
- ❑ PublisherName - nvarchar(50)

## ■ Bảng AuthorISBN

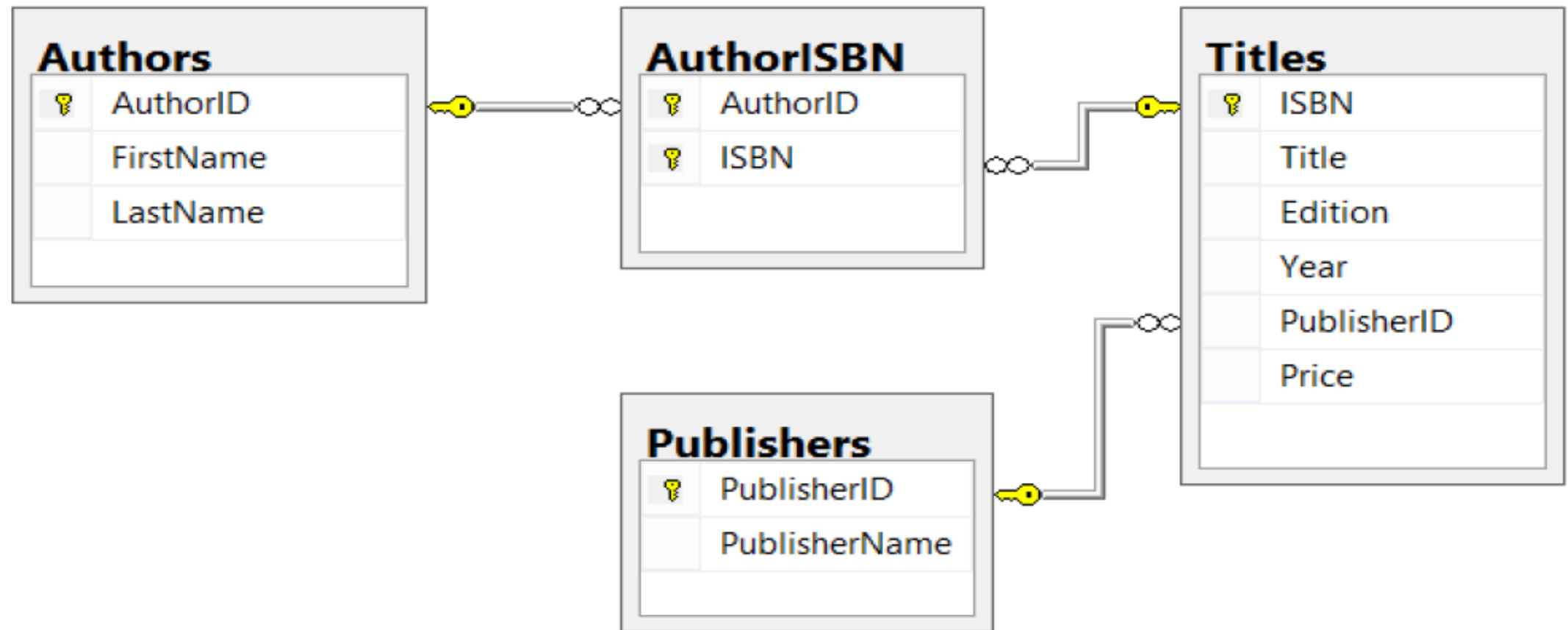
- ❑ AuthorID - int
- ❑ ISBN - nvarchar(10)

## ■ Bảng Titles

- ❑ ISBN - nvarchar(10)
- ❑ Title - nvarchar(50)
- ❑ Edition - int
- ❑ Year - int
- ❑ PublisherID - int
- ❑ Price - float

# Cơ sở dữ liệu Books

- Sơ đồ quan hệ các bảng dữ liệu



# Kết nối cơ sở dữ liệu

---

- Các bước kết nối cơ sở dữ liệu:
  - Định nghĩa chuỗi kết nối tới cơ sở dữ liệu SQL Server:
  - Tạo đối tượng kết nối lớp SqlConnection:
  - Tạo kết nối tới cơ sở dữ liệu:

```
string conStr = "server = <computer>;  
                database = <dataname>;  
                uid = <user>;  
                pwd = <password>;  
                connect timeout = <time>";
```

```
SqlConnection mySqlConnection = new SqlConnection(conStr);  
mySqlConnection.Open();
```

# Kết nối cơ sở dữ liệu

---

## ■ Ví dụ kết nối cơ sở dữ liệu Books

```
string conStr = "server = localhost;database = Books; uid = sa; pwd = sa;  
                connect timeout =150";  
SqlConnection mySqlConnection = new SqlConnection(conStr);  
mySqlConnection.Open();
```

## ■ Phát hiện các lỗi khi kết nối tới cơ sở dữ liệu:

```
try  
{  
    mySqlConnection = new SqlConnection(conStr);  
    mySqlConnection.Open();  
}  
catch (SQLException mySQLException)  
{  
    MessageBox.Show(mySQLException.Message, "Error");  
    return;  
}
```

# Lớp SqlCommand

---

- Đối tượng lớp SqlCommand dùng để truy vấn và cập nhật (nhập, sửa và xóa) dữ liệu trong cơ sở dữ liệu.
- Đối tượng lớp SqlCommand dùng để thực hiện một câu lệnh SQL thông qua một đối tượng lớp SqlConnection:
  - SELECT: Truy vấn dữ liệu.
  - INSERT: Nhập dữ liệu.
  - UPDATE: Sửa dữ liệu
  - DELETE: Xóa dữ liệu.
- Đối tượng lớp SqlCommand có thể dùng để gọi một thủ tục (stored procedure) của cơ sở dữ liệu SQL Server.

# Lớp SqlCommand

---

## ■ Tạo đối tượng SqlCommand:

- ❑ `SqlCommand mySqlCommand = new SqlCommand(string sSql, SqlConnection mySqlConnection);`

Trong đó:

- ❑ `sSql`: Định nghĩa lệnh SQL hoặc lệnh gọi thủ tục.
- ❑ `mySqlConnection`: Đối tượng đã kết nối tới cơ sở dữ liệu.

## ■ Các phương thức:

- ❑ `ExecuteNonQuery()`: Dùng để thực hiện một lệnh INSERT, UPDATE, DELETE hoặc một thủ tục không trả về kết quả.
- ❑ `ExecuteReader()`: Dùng để thực hiện lệnh SELECT hoặc các thủ tục với kết quả trả về là một đối tượng lớp `DataReader`.
- ❑ `ExecuteScalar()`: Dùng để thực hiện lệnh SELECT với kết quả trả về là một giá trị vô hướng như hàm `count()`, `sum()`.



# Lớp SqlCommand

---

## ■ Truy vấn dữ liệu dùng lớp SqlCommand:

### 1. Tạo đối tượng lớp SqlConnection và kết nối tới cơ sở dữ liệu:

```
SqlConnection mySqlConnection = new SqlConnection(conStr); mySqlConnection.Open();
```

### 2. Tạo đối tượng lớp SqlCommand:

```
SqlCommand mySqlCommand=new SqlCommand(sSql,mySqlConnection);
```

hoặc

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

```
mySqlCommand.CommandText = sSql;
```

### 3. Thực hiện phương thức ExecuteReader():

```
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
```

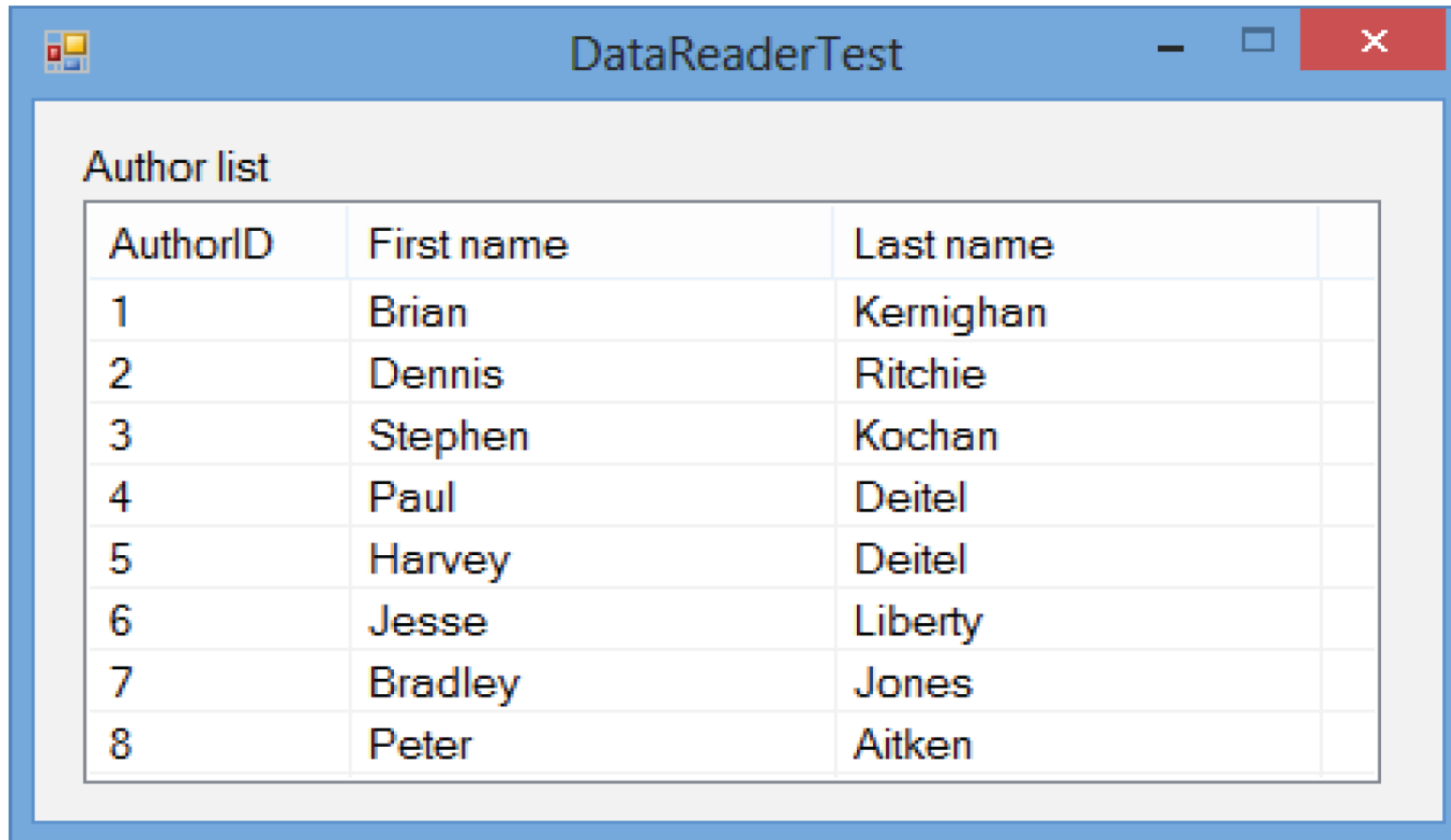
# Lớp SqlCommand

---

- Các thuộc tính thường dùng của SqlDataReader:
  - `FieldCount`: Số cột của dòng hiện thời.
  - `IsClosed`: Có/không đối tượng đã đóng.
- Các phương thức thường dùng của SqlDataReader:
  - `Read()`: Di chuyển con trỏ để đọc dòng tiếp theo.
  - `Close()`: Đóng đối tượng.
- Phương thức `Load()` của đối tượng DataTable:
  - `Load()`: Chuyển dữ liệu đối tượng SqlDataReader vào đối tượng DataTable.

# Lớp SqlCommand

- Ví dụ xây dựng form hiển thị dữ liệu bảng Authors lên đối tượng DataGridView:



The screenshot shows a Windows application window titled "DataReaderTest". Inside the window, there is a label "Author list" above a DataGridView. The DataGridView displays a table with 8 rows of author data. The columns are "AuthorID", "First name", and "Last name".

AuthorID	First name	Last name
1	Brian	Kernighan
2	Dennis	Ritchie
3	Stephen	Kochan
4	Paul	Deitel
5	Harvey	Deitel
6	Jesse	Liberty
7	Bradley	Jones
8	Peter	Aitken

# Lớp SqlCommand

---

- Tạo đối tượng lớp DataGridView, nháy chuột phải và chọn Add, mỗi cột khai báo:
  - **Name**: Tên cột dùng trong mã lệnh.
  - **Header text**: Tiêu đề hiển thị của cột.
  - **DataPropertyName**: Tên cột dữ liệu của DataTable.
- Các thuộc tính và sự kiện thường dùng:
  - **DataSource**: Tên DataTable cần hiển thị lên lưới.
  - **AutoGenerateColumns**: Có/không tự động lấy các cột.
  - **AllowUserToAddRows**: Có/không cho phép thêm dòng.
  - **AllowUserToDeleteRows**: Có/không cho phép xóa dòng.
  - **RowEnter()**: Sự kiện xảy ra khi con trỏ đưa vào một dòng.

# Lớp SqlCommand

---

## ■ Khai báo biến

```
private string conStr = "server =; database =; uid =; pwd =";  
private SqlConnection mySqlConnection;  
private SqlCommand mySqlCommand;
```

## ■ Sự kiện Form\_load()

```
mySqlConnection = new SqlConnection(conStr);  
mySqlConnection.Open();  
string sSql = "SELECT * FROM Authors ORDER by AuthorID";  
mySqlCommand = new SqlCommand(sSql, mySqlConnection);  
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();  
DataTable myDataTable = new DataTable();  
myDataTable.Load(mySqlDataReader);  
GridView1.DataSource = myDataTable;
```

# Lớp SqlCommand

---

- Cập nhật dữ liệu dùng lớp SqlCommand:

1. Tạo đối tượng lớp SqlConnection và kết nối tới cơ sở dữ liệu:

```
SqlConnection mySqlConnection = new SqlConnection(conStr); mySqlConnection.Open();
```

2. Tạo đối tượng lớp SqlCommand:

```
SqlCommand mySqlCommand=new SqlCommand(sSql,mySqlConnection);
```

hoặc

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
```

```
mySqlCommand.CommandText = sSql;
```

3. Thực hiện phương thức ExecuteNonQuery() :

```
mySqlCommand.ExecuteNonQuery();
```

# Lớp SqlCommand

- Ví dụ về truy vấn, nhập, sửa, xóa trong bảng Authors. Dữ liệu hiển thị lên DataGridView:

[illegible]

# Lớp SqlCommand

---

## ■ Sử dụng tham số trong lớp SqlCommand:

### 1. Tạo đối tượng lớp SqlCommand:

```
string sSql = "INSERT INTO Authors (FirstName, LastName)" +  
             " VALUES (@FirstName, @LastName)";  
SqlCommand mySqlCommand = new MySqlCommand(sSql,mySqlConnection);
```

### 2. Định nghĩa tham số cho đối tượng lớp SqlCommand:

```
mySqlCommand.Parameters.Add("@FirstName",SqlDbType.NVarChar,25);  
mySqlCommand.Parameters.Add("@LastName",SqlDbType.NVarChar,10);
```

### 3. Gán giá trị cho các tham số:

```
mySqlCommand.Parameters["@FirstName"].Value = "x";  
mySqlCommand.Parameters["@LastName"].Value = "y";
```

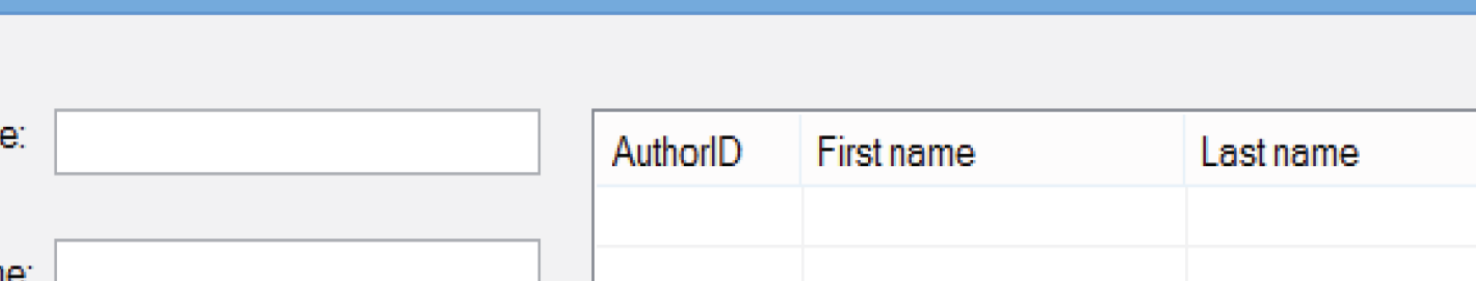
### 4. Thực hiện phương thức ExecuteNonQuery():

```
mySqlCommand.ExecuteNonQuery();
```



# Lớp SqlCommand

- Sử dụng tham số trong lớp SqlCommand:
  - Ví dụ về truy vấn, nhập, sửa, xóa trong bảng Authors. Dữ liệu hiển thị lên DataGridView:



The screenshot shows a Windows Forms application window titled "Add-Edit-Delete Authors". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area is light gray and contains the following elements:

- On the left side, there are two text boxes labeled "First name:" and "Last name:". Below these are six buttons arranged in two rows: "New", "Edit", "Delete" in the top row, and "Save", "Cancel", "Close" in the bottom row. The buttons have a light gray background and black text.
- On the right side, there is a table with three columns: "AuthorID", "First name", and "Last name". The table has a white background and a thin gray border. It contains 10 empty rows, including the header row.

# Lớp SqlCommand

---

- Thực hiện thủ tục của cơ sở dữ liệu:
  - ❑ Thủ tục là phương pháp thực hiện hiệu quả để cập nhật và sửa đổi dữ liệu của cơ sở dữ liệu.
  - ❑ `ExecuteNonQuery()`: thực hiện thủ tục không trả về kết quả.
  - ❑ `ExecuteReader()`: thực hiện thủ tục trả về kết quả.
- Thực hiện thủ tục không trả về kết quả:
  - ❑ Tạo một đối tượng lớp `SqlCommand`
  - ❑ Thiết lập thuộc tính `CommandText` để gọi thủ tục.
  - ❑ Định nghĩa các tham số cho thủ tục.
  - ❑ Gán giá trị cho các tham số truyền vào cho thủ tục.
  - ❑ Dùng phương thức `ExecuteNonQuery()`.
  - ❑ Đọc giá trị của các tham số ra của thủ tục nếu có.

# Lớp SqlCommand

---

## ■ Thủ tục thêm một tác giả:

```
CREATE PROCEDURE AddAuthor
    @FirstName nvarchar(25),
    @LastName nvarchar(10)
AS
    DECLARE @AuthorID int
BEGIN
    INSERT INTO Authors (FirstName, LastName)
    VALUES (@FirstName, @LastName)
    SET @AuthorID = SCOPE_IDENTITY()
RETURN @AuthorID
END
```

# Lớp SqlCommand

---

## ■ Thủ tục cập nhật một tác giả:

```
CREATE PROCEDURE UpdateAuthor
    @OldAuthorID int,
    @FirstName nvarchar(25),
    @LastName nvarchar(10)
AS
BEGIN
    UPDATE Authors
    SET FirstName = @FirstName, LastName = @LastName
    WHERE AuthorID = @OldAuthorID
END
```

# Lớp SqlCommand

---

```
CREATE PROCEDURE DeleteAuthor
    @OldAuthorID int
AS
BEGIN
    DELETE FROM Authors
    WHERE AuthorID = @OldAuthorID
END
```

## ■ Xóa tác giả có AuthorID = 10:

```
string sSql = "EXECUTE DeleteAuthor @OldAuthorID";
mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText = sSql;
mySqlCommand.Parameters.Add("@OldAuthorID", SqlDbType.Int, 0);
mySqlCommand.Parameters["@OldAuthorID"].Value = 10;
mySqlCommand.ExecuteNonQuery();
```

# Lớp SqlCommand

---

- Thực hiện thủ tục trả về kết quả: ExecuteReader()

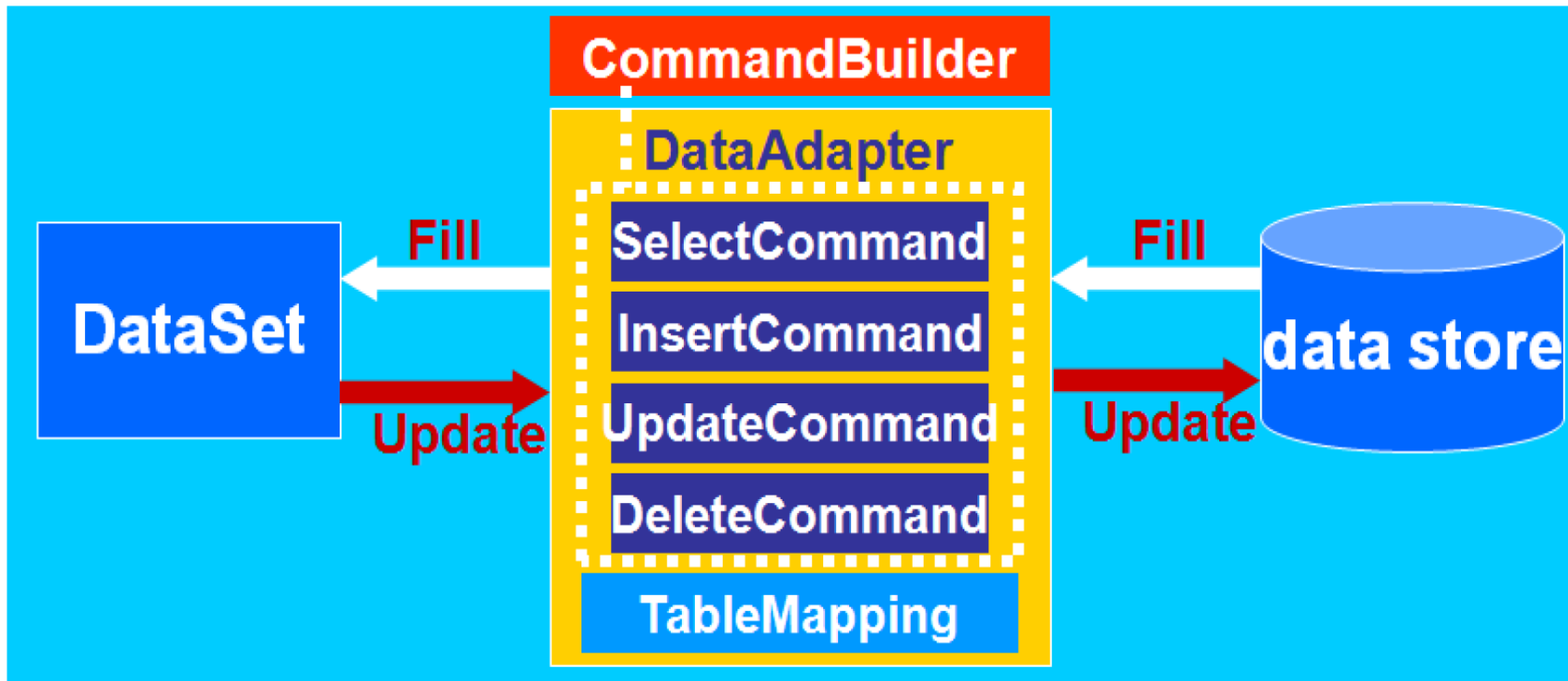
```
CREATE PROCEDURE DisplayAuthors  
  
AS  
  
    SELECT * FROM Authors  
  
GO
```

- Gọi thủ tục:

```
string sSql = "EXECUTE DisplayAuthors";  
mySqlCommand = mySqlConnection.CreateCommand();  
mySqlCommand.CommandText = sSql;  
SqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
```

# Lớp SqlDataAdapter

- Một đối tượng lớp SqlDataAdapter là một đường chuyển dữ liệu giữa ứng dụng và cơ sở dữ liệu:
  - Truy vấn dữ liệu: **Fill**
  - Cập nhật dữ liệu: **Update**



# Lớp SqlDataAdapter

---

- Tạo đối tượng lớp SqlDataAdapter:

  - `SqlDataAdapter(SqlCommand mySqlCommand);`

  - `SqlDataAdapter(string sSql, SqlConnection mySqlConnection);`

- Các thuộc tính:

  - ❑ `DeleteCommand`: thực hiện lệnh DELETE hoặc thủ tục.

  - ❑ `InsertCommand`: thực hiện lệnh INSERT hoặc thủ tục.

  - ❑ `UpdateCommand`: thực hiện lệnh UPDATE hoặc thủ tục.

  - ❑ `SelectCommand`: thực hiện lệnh SELECT hoặc thủ tục.

- Các phương thức:

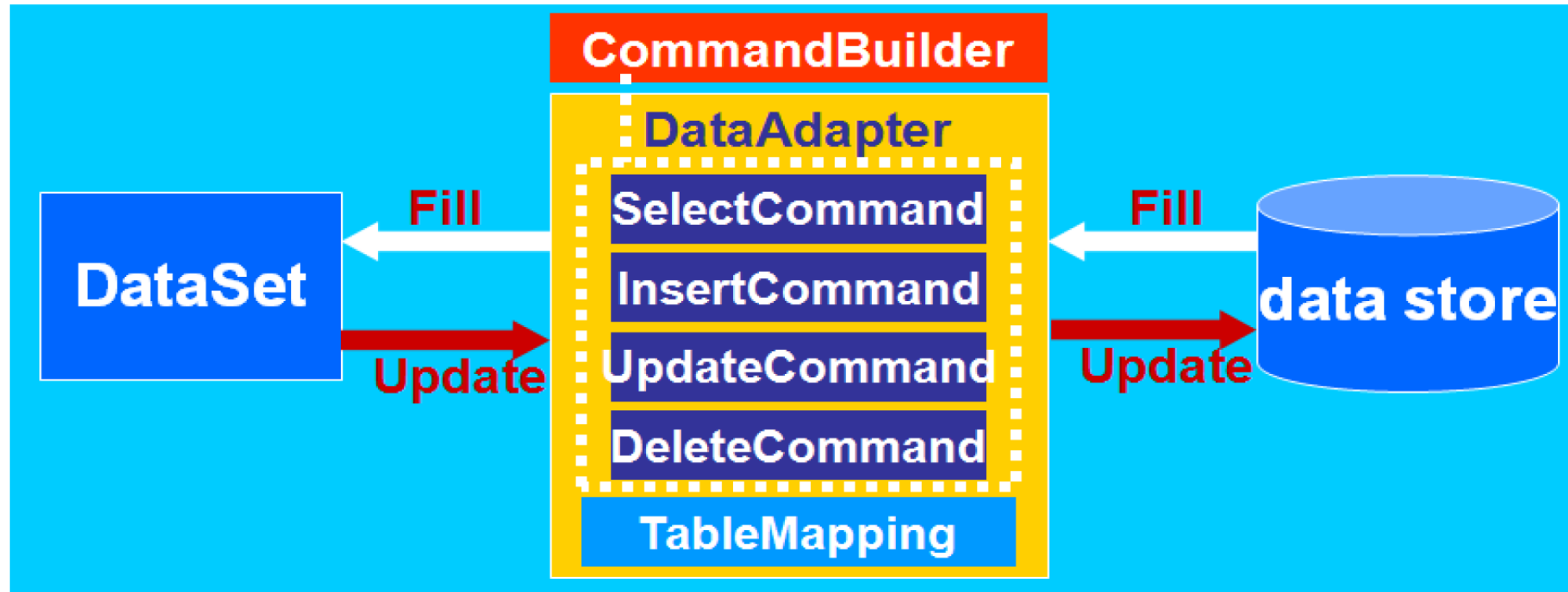
  - ❑ `Fill()`: truy vấn dữ liệu vào DataSet hoặc DataTable.

  - ❑ `Update()`: cập nhật dữ liệu từ DataSet hoặc DataTable.



# Lớp SqlCommandBuilder

- Sử dụng lớp SqlCommandBuilder để giảm mã lệnh viết cho các thuộc tính InsertCommand, UpdateCommand và DeleteCommand của SqlDataAdapter.
- Khi thay đổi dữ liệu trong DataSet/DataTable và gọi phương thức Update() thì dữ liệu trong DataSet/ DataTable sẽ đồng bộ với cơ sở dữ liệu.



# Lớp SqlCommandBuilder

---

- Các hạn chế của lớp SqlCommandBuilder:
  - ❑ Câu lệnh SELECT của đối tượng lớp SqlDataAdapter phải được truy vấn dữ liệu từ một bảng (trong trường hợp có cập nhật lại dữ liệu).
  - ❑ Bảng truy vấn trong cơ sở dữ liệu phải chứa một khóa chính.
  - ❑ Khóa chính của bảng phải có trong câu lệnh SELECT truy vấn dữ liệu.

# Lớp SqlCommandBuilder

---

## ■ Truy vấn dữ liệu vào đối tượng lớp DataTable:

- ❑ Tạo đối tượng lớp SqlConnection và kết nối cơ sở dữ liệu:

```
SqlConnection mySqlConnection = new SqlConnection(conStr);  
mySqlConnection.Open();
```

- ❑ Tạo đối tượng lớp SqlDataAdapter:

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter(sSql,mySqlConnection);
```

- ❑ Tạo đối tượng lớp SqlCommandBuilder:

```
SqlCommandBuilder mySqlCommandBuilder;  
mySqlCommandBuilder = new SqlCommandBuilder(mySqlDataAdapter);
```

- ❑ Tạo đối tượng lớp DataTable và truy vấn dữ liệu:

```
DataTable myDataTable = new DataTable();  
mySqlDataAdapter.Fill(myDataTable);
```

# Lớp SqlCommandBuilder

---

- Cập nhật dữ liệu vào đối tượng lớp DataTable:

- Nhập dữ liệu vào đối tượng lớp DataTable:

```
DataRow myDataRow = myDataTable.NewRow();
```

```
myDataRow["FirstName"] = "Peter";
```

```
myDataRow["LastName"] = "Aitken";
```

```
myDataTable.Rows.Add(myDataRow);
```

```
mySqlDataAdapter.Update(myDataTable);
```

- Xóa dữ liệu trong đối tượng lớp DataTable:

```
myDataTable.Rows[pos].Delete();
```

```
myDataAdapter.Update(myDataTable);
```

Chú ý: pos là dòng cần xóa.

# Lớp SqlCommandBuilder

- Sửa dữ liệu trong đối tượng lớp DataTable, pos là dòng cần sửa:  

```
DataRow editRow =myDataTable.Rows[pos];  
editRow["FirstName"] = "abc";  
editRow["LastName"] = "def";  
myDataAdapter.Update(myDataTable);
```
- Ví dụ xây dựng form sau:

[illegible]

# Lớp ListBox và ComboBox

---

- **ListBox**: Cho phép người dùng xem và chọn các dòng dữ liệu từ danh sách.
- **ComboBox**: Sự kết hợp của **TextBox** và **ListBox**.
- Các thuộc tính thường dùng:
  - **DataSource**: Nguồn dữ liệu, là một **DataTable**.
  - **DisplayMember**: Cột hiển thị trong **ListBox**.
  - **ValueMember**: Cột giá trị trả về khi chọn **ListBox**.
  - **SelectedIndex**: Dòng hiện thời được chọn.
  - **SelectedValue**: Giá trị được chọn trên **ListBox**.

# Xử lý lỗi ngoại lệ dùng lớp SQLException

---

## ■ Khối lệnh xử lý lỗi:

```
try
{
    <khối lệnh có thể xảy ra lỗi>
} catch (SQLException mySQLException)
{
    <xử lý lỗi sinh ra do cơ sở dữ liệu>
}
```

## ■ Các thuộc tính cơ bản của lớp SQLException:

- ❑ Number: Số hiệu lỗi.
- ❑ Message: Xâu chứa thông báo lỗi.
- ❑ StackTrace: Vị trí sinh ra lỗi gồm dòng lệnh và phương thức sinh ra lỗi.

# Xử lý lỗi ngoại lệ dùng lớp SQLException

---

- Một số mã lỗi của đối tượng lớp SQLException:
  - ❑ 53: Tên máy chủ cơ sở dữ liệu sai.
  - ❑ 4060: Không đúng tên cơ sở dữ liệu.
  - ❑ 18456: Không đúng tên và/hoặc mật khẩu truy nhập.
  - ❑ 547: Dữ liệu nhập/sửa vi phạm ràng buộc khóa ngoài.
  - ❑ 2601|2627: Trùng giá trị dữ liệu trường khóa.
  - ❑ 8152: Dữ liệu lưu trữ vào trường của bảng quá dài.



# Xây dựng lớp dịch vụ dữ liệu

---

- Mọi form không phụ thuộc vào cơ sở dữ liệu.
- Chỉ kết nối dữ liệu một lần khi chạy ứng dụng.
- Mỗi form sử dụng các đối tượng lớp.
  - Lấy dữ liệu vào DataTable
  - Cập nhật dữ liệu từ DataTable vào cơ sở dữ liệu.
  - Thực hiện các câu lệnh SQL thao tác với dữ liệu.

- Khai báo:

```
private static SqlConnection mySqlConnection;  
private SqlDataAdapter myDataAdapter;  
public DataServices()  
{  
}
```

# Xây dựng lớp dịch vụ dữ liệu

---

```
public bool OpenDB(string myComputer, string myDB, string uid, string psw)
{
    string conStr = “. . . connection string . . .”;
    try
    {
        mySqlConnection = new SqlConnection(conStr);
        mySqlConnection.Open();
    }
    catch (SqlException ex)
    {
        MessageBox.Show(ex.Message, "Error " + ex.Number.ToString());
        mySqlConnection = null;
        return false;
    }
    return true;
}
```

# Xây dựng lớp dịch vụ dữ liệu

---

```
public DataTable RunQuery(string QueryString)
{
    DataTable myDataTable = new DataTable();
    myDataAdapter = new SqlDataAdapter();
    try
    {
        myDataAdapter = new SqlDataAdapter(QueryString, mySqlConnection);
        SqlCommandBuilder mySqlCommandBuilder = new SqlCommandBuilder(myDataAdapter);
        myDataAdapter.Fill(myDataTable);
    } catch (SqlException ex)
    {
        MessageBox.Show(ex.Message, "Error " + ex.Number.ToString());
        return null;
    }
    return myDataTable;
}
```

# Xây dựng lớp dịch vụ dữ liệu

---

```
public void Update(DataTable myDataTable)
{
    try
    {
        myDataAdapter.Update(myDataTable);
    }catch (SqlException ex)
    {
        MessageBox.Show(ex.Message,"Error " + ex.Number.ToString());
    }
}
public void ExecuteNonQuery(string cmdString)
{
    SqlCommand mySqlCommand = new SqlCommand(cmdString,mySqlConnection);
    try
    {
        mySqlCommand.ExecuteNonQuery();
    }catch (SqlException ex)
    {
        MessageBox.Show(ex.Message,"Error " + ex.Number.ToString());
    }
}
```

# Kết thúc

---



**HỎI & TRẢ LỜI**