

# Apresentação

---

Curso:

Programação .NET I

Aula 08:

Objetivos:

Instalação e configuração do SQL Server LocalDB;

Acesso a banco de dados;

Visualização de dados no GridView;

Projeto Mini-Venda;

# Acesso a banco de dados (ADO.NET)

A partir deste momento, já sentimos necessidade de gravar dados de forma persistente. Para isso, utilizaremos o conjunto de classes ADO.NET.

Com o advento do framework .NET, a Microsoft resolveu atualizar o modelo de acesso a banco de dados, com isso, o ActiveX Data Objects (ADO) foi atualizado para o ADO.NET.

# Acesso a banco de dados (ADO.NET)

Uma das principais diferenças do ADO.NET está na extinção da classe RecordSet. Agora utiliza-se as classes TableAdapter e DataSet.

A TableAdapter e a DataSet suportam o acesso a dados e podem fazer isso de forma desconectadas, permitindo uma maior escalabilidade, pois não é mais necessário ficar conectado ao banco o tempo todo.

# Acesso a banco de dados (ADO.NET)

O modelo ADO.NET é projetado para ser de fácil utilização, possuindo diversos assistentes e recursos para gerar automaticamente os códigos de acesso ao banco.

# LocalDB (SQL Server for Dev.)

---

Para facilitar a vida do desenvolvedor, a Microsoft criou uma distribuição local do SQL Server.

O LocalDB pode ser considerado uma versão do “SQL Server Express otimizada para desenvolvedores”.

# LocalDB (SQL Server for Dev.)

---

O LocalDB é totalmente compatível com o SQL Server (Normal/Express) e tudo o que for desenvolvido/testado, segue a premissa de não precisar de nenhuma modificação para funcionar nas versões mais robustas (SQL Server/Server Express).

# LocalDB (SQL Server for Dev.)

---

## Apontamentos do LocalDB:

- Usa o mesmo sqlservr.exe que o servidor normal;
- Não cria serviços de banco. Os processos são iniciados e parados de acordo com a necessidade;
- Funciona com a propriedade “AttachDbFileName”;
- A instalação do LocalDB requer 140MB de espaço;
- Não utiliza o TCP/IP para troca de dados;

# LocalDB (SQL Server for Dev.)

---

Faça o teste para verificar se o LocalDB está em funcionamento. Abra o CMD do Windows.

Digite:

```
C:\> SqlLocalDb info
```

Resultado:

```
v11.0
```



# LocalDB (SQL Server for Dev.)

---

Caso não esteja apareça o resultado esperado, devemos instalar o LocalDB. O instalador está no diretório da VM, na pasta “Programas”.

A instalação é simples, feita através de um assistente.

# LocalDB (SQL Server for Dev.)

---

## Comandos básicos no LocalDB: **Criar Instância**

Digite:

```
C:\> SqlLocalDb create "NomeInstancia"
```

Resultado:

```
LocalDB instance "NomeInstancia" created  
with version 11.0.
```

# LocalDB (SQL Server for Dev.)

---

## Comandos básicos no LocalDB: **Iniciar Instância**

Digite:

```
C:\> SqlLocalDb start "NomeInstancia"
```

Resultado:

```
LocalDB instance "NomeInstancia" started.
```

# LocalDB (SQL Server for Dev.)

---

## Comandos básicos no LocalDB: **Criar e Iniciar Instância**

Digite:

```
C:\> SqlLocalDb create "NomeInstancia" -s
```

Resultado:

```
LocalDB instance "NomeInstancia" created with  
version 11.0.
```

```
LocalDB instance "NomeInstancia" started.
```

# LocalDB (SQL Server for Dev.)

---

## Comandos básicos no LocalDB: **Informações da Instância**

Digite:

```
C:\> SqlLocalDb info "NomeInstancia"
```



# LocalDB (SQL Server for Dev.)

---

## Comandos básicos no LocalDB: Informações da Instância

### Resultado:

<i>Name:</i>	<i>NomeInstancia</i>
<i>Version:</i>	<i>11.0.2100.60</i>
<i>Shared name:</i>	
<i>Owner:</i>	<i>nomeComputador\NomeUsuario</i>
<i>Auto-create:</i>	<i>No</i>
<i>State:</i>	<i>Running</i>
<i>Last start time:</i>	<i>08/09/2015 10:15:56</i>
<i>Instance pipe name:</i>	<i>np:\\.\pipe\LOCALDB#FB88DC0E\tsql\query</i>

# LocalDB (SQL Server for Dev.)

---

Comandos básicos no LocalDB:

## Deletar Instância (se ainda estiver rodando)

Digite:

```
C:\> SqlLocalDb delete "NomeInstancia"
```

## Resultado:

*Delete of LocalDB instance "NomeInstancia" failed because of the following error:*

*Requested operation on LocalDB instance cannot be performed because specified instance is currently in use. Stop the instance and try again.*

# LocalDB (SQL Server for Dev.)

---

Comandos básicos no LocalDB:

## Deletar Instância

Digite:

```
C:\> SqlLocalDb stop "NomeInstancia"
```

```
C:\> SqlLocalDb delete "NomeInstancia"
```

Resultado:

```
LocalDB instance "NomeInstancia" deleted.
```



# Criar Instancia no LocalDB

---

## Exercício 2:

Crie uma instância no LocalDB com o nome “AulaDotNet”. Inicie a instância criada e verifique as informações da instância.

## Lembrete:

SqlLocalDB create “NomeInstancia”

SqlLocalDB start “NomeInstancia”

SqlLocalDB create “NomeInstancia” –S

SqlLocalDB stop “NomeInstancia”

SqlLocalDB delete “NomeInstancia”

# SQL Server Management Studio

---

O SQL Server Management Studio é uma ferramenta que auxilia muito no gerenciamento de bancos criados no SQL Server.

É uma ferramenta livre e será utilizada para instalar o banco que usaremos nos nossos exemplos a seguir.

O arquivo de instalação está na pasta da rede com o nome: “SQLManagementStudio\_x64\_ENU.exe” ou “SQLManagementStudio\_x86\_ENU.exe”.



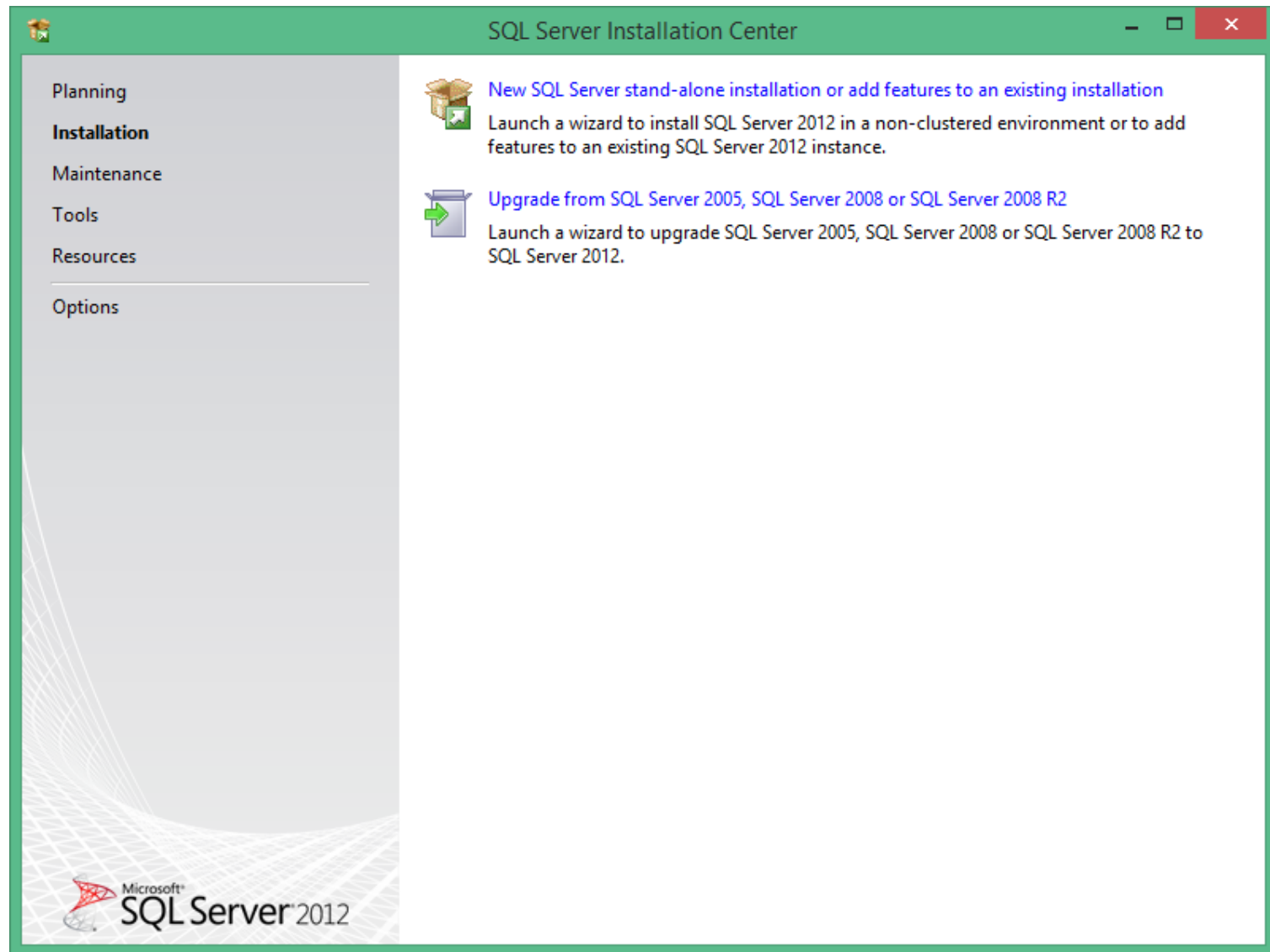
# SQL Server Management Studio

---

Escolher:

“New SQL Server standalone installation or add features to a existing installation”

# SQL Server Management Studio

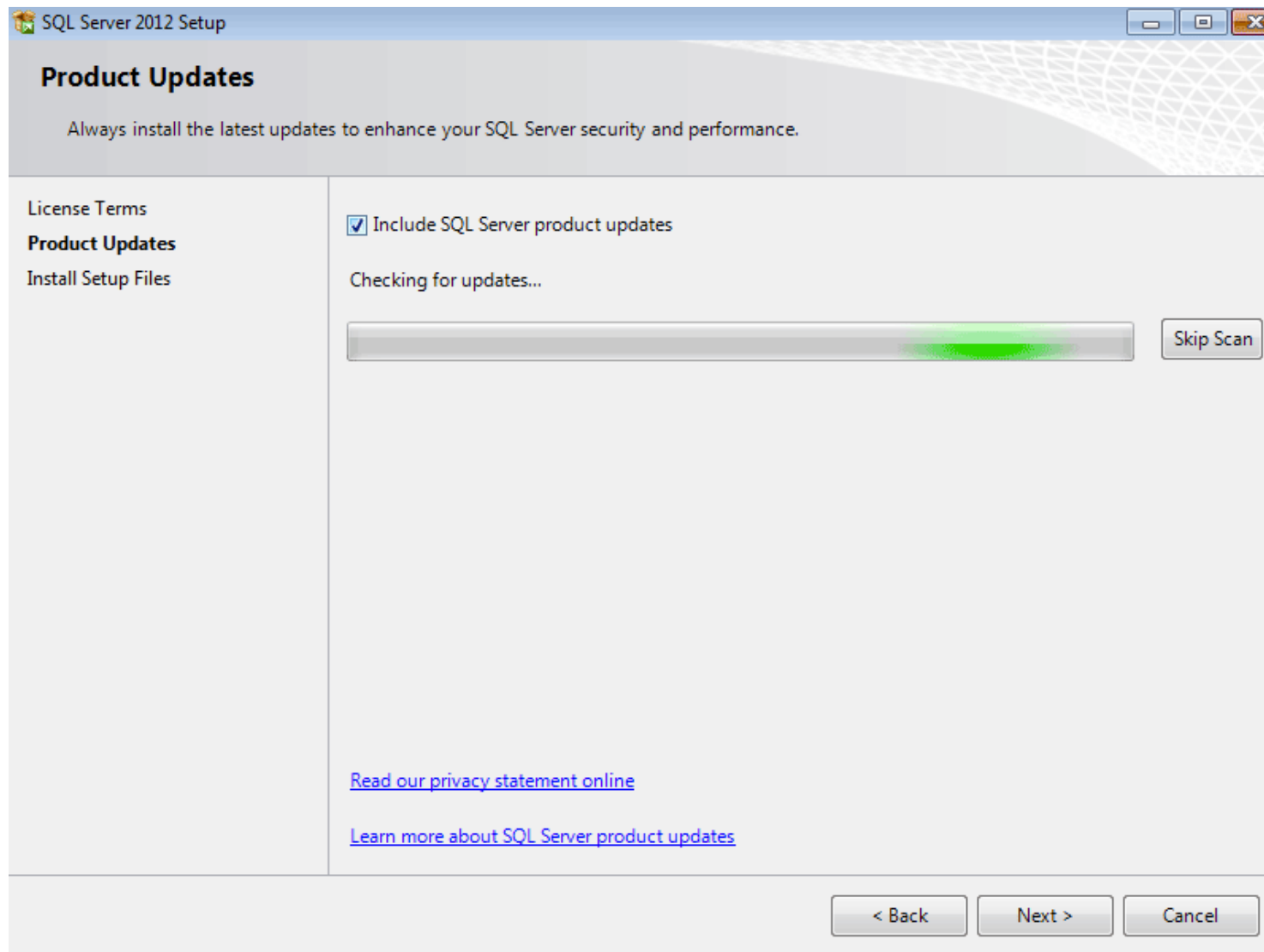


# SQL Server Management Studio

---

Aguardar o assistente procurar atualizações do SQL Server...

# SQL Server Management Studio

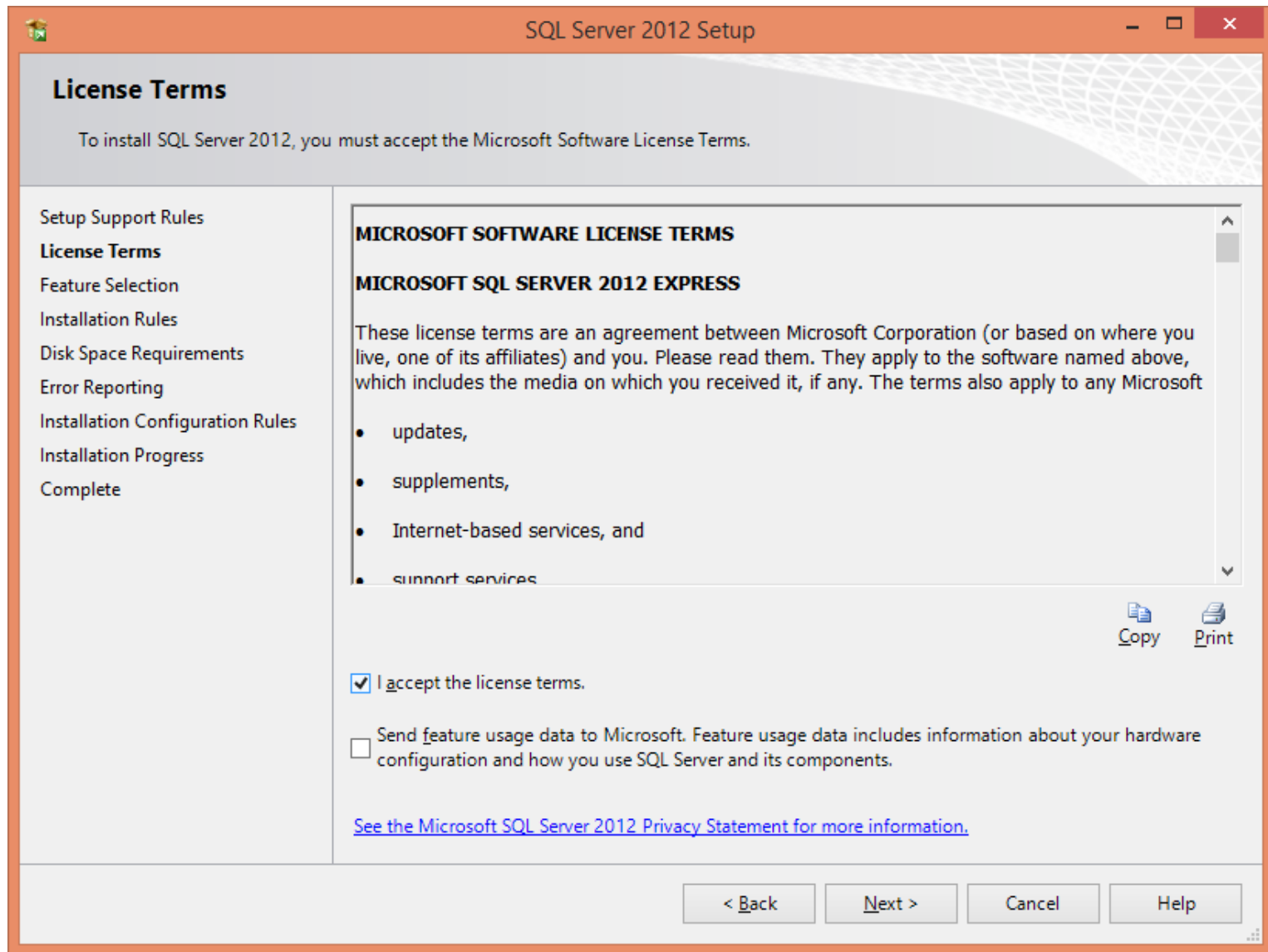


# SQL Server Management Studio

---

Aceitar os termos de uso.

# SQL Server Management Studio





# SQL Server Management Studio

---

Marcar:

“Management Tools – Basic”;

“SQL Client Connectivity”;

“LocalDB”

# SQL Server Management Studio

The screenshot shows the 'SQL Server 2012 Setup' window with the 'Feature Selection' tab active. The window title bar includes standard Windows window controls. On the left, a navigation pane lists various setup steps, with 'Feature Selection' highlighted. The main area is titled 'Select the Express features to install.' and is divided into three sections. The 'Features' section on the left contains a tree view with 'Instance Features' selected and expanded, showing 'Shared Features' (with checkboxes for 'Management Tools - Basic', 'SQL Client Connectivity SDK', and 'LocalDB') and 'Redistributable Features'. Below this are 'Select All' and 'Unselect All' buttons. The 'Feature description:' section on the right provides details for the selected 'Instance Features', stating they are isolated from other SQL Server instances. Below this, the 'Prerequisites for selected features:' section lists items already installed (Microsoft .NET Framework 4.0, Windows PowerShell 2.0, Microsoft .NET Framework 3.5) and items to be installed from media (Microsoft Visual Studio 2010 Shell). At the bottom, there are text boxes for 'Shared feature directory' and 'Shared feature directory (x86)', both pointing to 'C:\Program Files\Microsoft SQL Server\' and 'C:\Program Files (x86)\Microsoft SQL Server\' respectively. The bottom of the window features '< Back', 'Next >', 'Cancel', and 'Help' buttons.

SQL Server 2012 Setup

## Feature Selection

Select the Express features to install.

Setup Support Rules  
License Terms  
**Feature Selection**  
Installation Rules  
Disk Space Requirements  
Error Reporting  
Installation Configuration Rules  
Installation Progress  
Complete

Features:

**Instance Features**

Shared Features

- ☒ Management Tools - Basic
- ☒ SQL Client Connectivity SDK
- ☒ LocalDB

Redistributable Features

Feature description:

The configuration and operation of each instance feature of a SQL Server instance is isolated from other SQL Server instances. SQL Server instances can operate side-by-side on the same computer.

Prerequisites for selected features:

Already installed:

- Microsoft .NET Framework 4.0
- Windows PowerShell 2.0
- Microsoft .NET Framework 3.5

To be installed from media:

- Microsoft Visual Studio 2010 Shell

Select All Unselect All

Shared feature directory: C:\Program Files\Microsoft SQL Server\

Shared feature directory (x86): C:\Program Files (x86)\Microsoft SQL Server\

< Back Next > Cancel Help

# SQL Server Management Studio

---

Clicar no botão Next até terminar a instalação.

# SqlCommand (SQL Server LocalDB)

Entrando no modo “SqlCommand”, é possível criar Bancos de Dados, Tabelas e executar diversas tarefas por comandos T-SQL. Como por exemplo: Verificar a versão e informações do servidor de Banco.

# SqlCommand (SQL Server LocalDB)

Comandos básicos no SqlCommand:

Digite:

```
C:\> sqlcmd -S (Localdb)\NomeInstancia -E
```

Resultado:

```
1>
```

# SqlCommand (SQL Server LocalDB)

Comandos básicos no SQL Command:

Digite:

```
1> SELECT @@VERSION;
```

```
2> GO
```

# SqlCommand (SQL Server LocalDB)

Comandos básicos no SQL Command:

Resultado:

-----  
-----  
-----  
-----

*Microsoft SQL Server 2012 - 11.0.2100.60 (X64)*

*Feb 10 2012 19:39:15*

*Copyright (c) Microsoft Corporation*

*Express Edition (64-bit) on Windows NT 6.2 <X64> (Build 9200: )*

*(1 rows affected)*

# SQL Server Management Studio

---

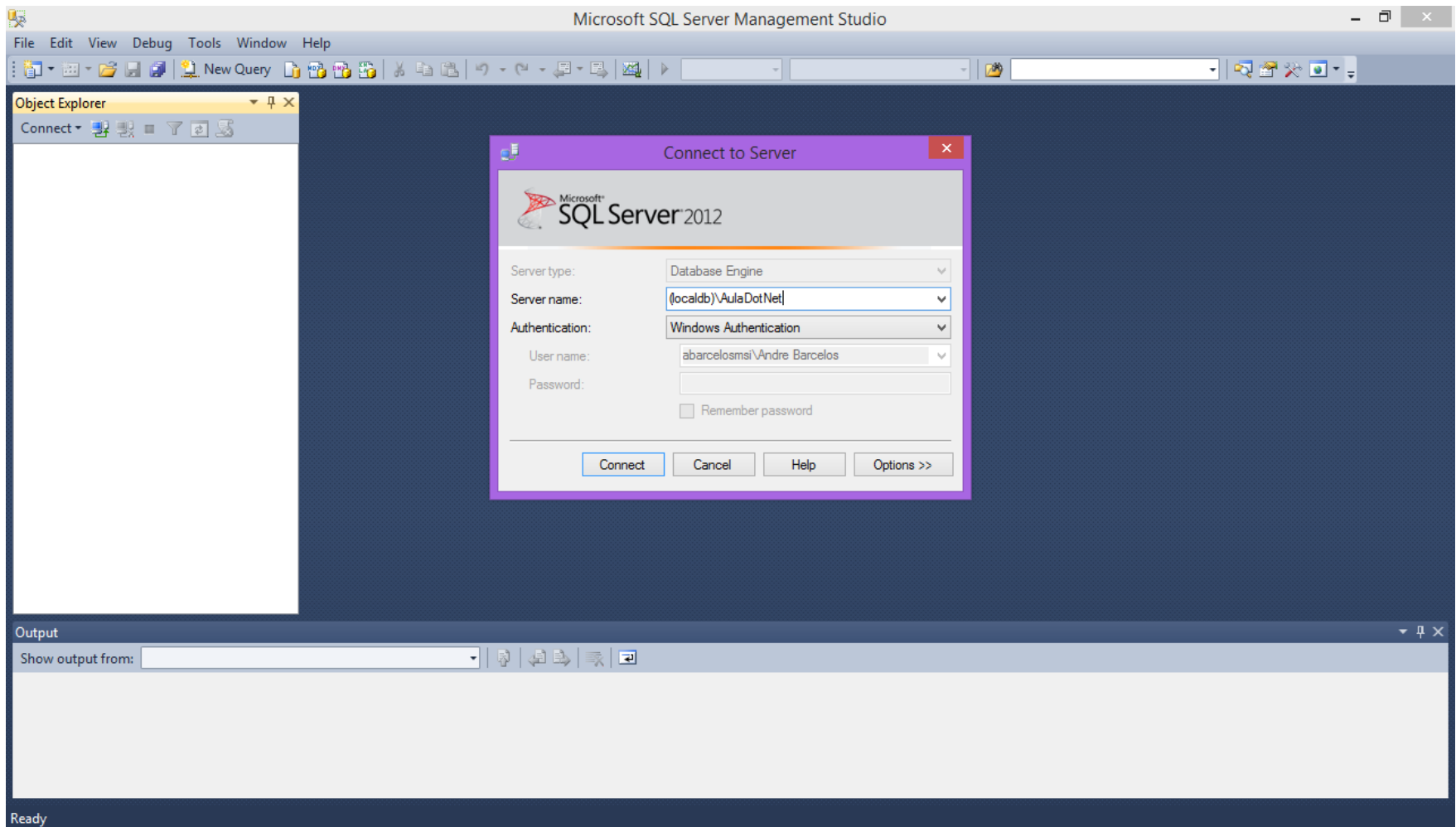
Após instalado o SQL Server Management Studio, vamos conectar à instância que foi criada. Assim será possível criar o banco de dados para executarmos os exemplos a seguir.

Coloque no campo **Server name:** (localdb)\NomeInstancia

Server name: (localdb)\AulaDotNet  
Authentication: Windows Authentication



# SQL Server Management Studio



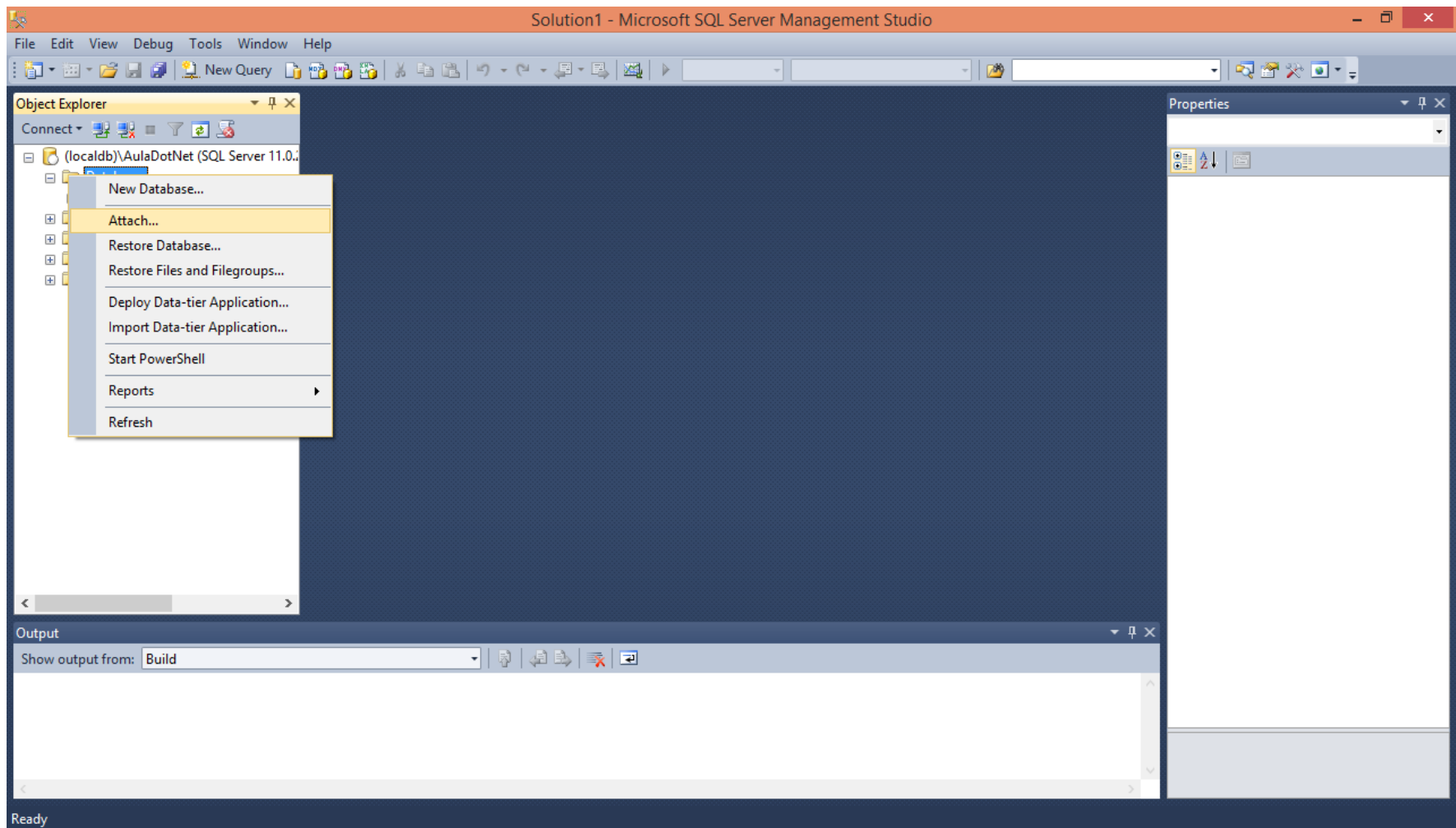
# SQL Server Management Studio

---

Após conectado à instância do LocalDb, vamos criar o Banco Northwind Traders (mostrado anteriormente).

O Arquivo de modelo do banco está no diretório da VM, na rede. O nome do arquivo é: “northwnd.mdf”.

# SQL Server Management Studio

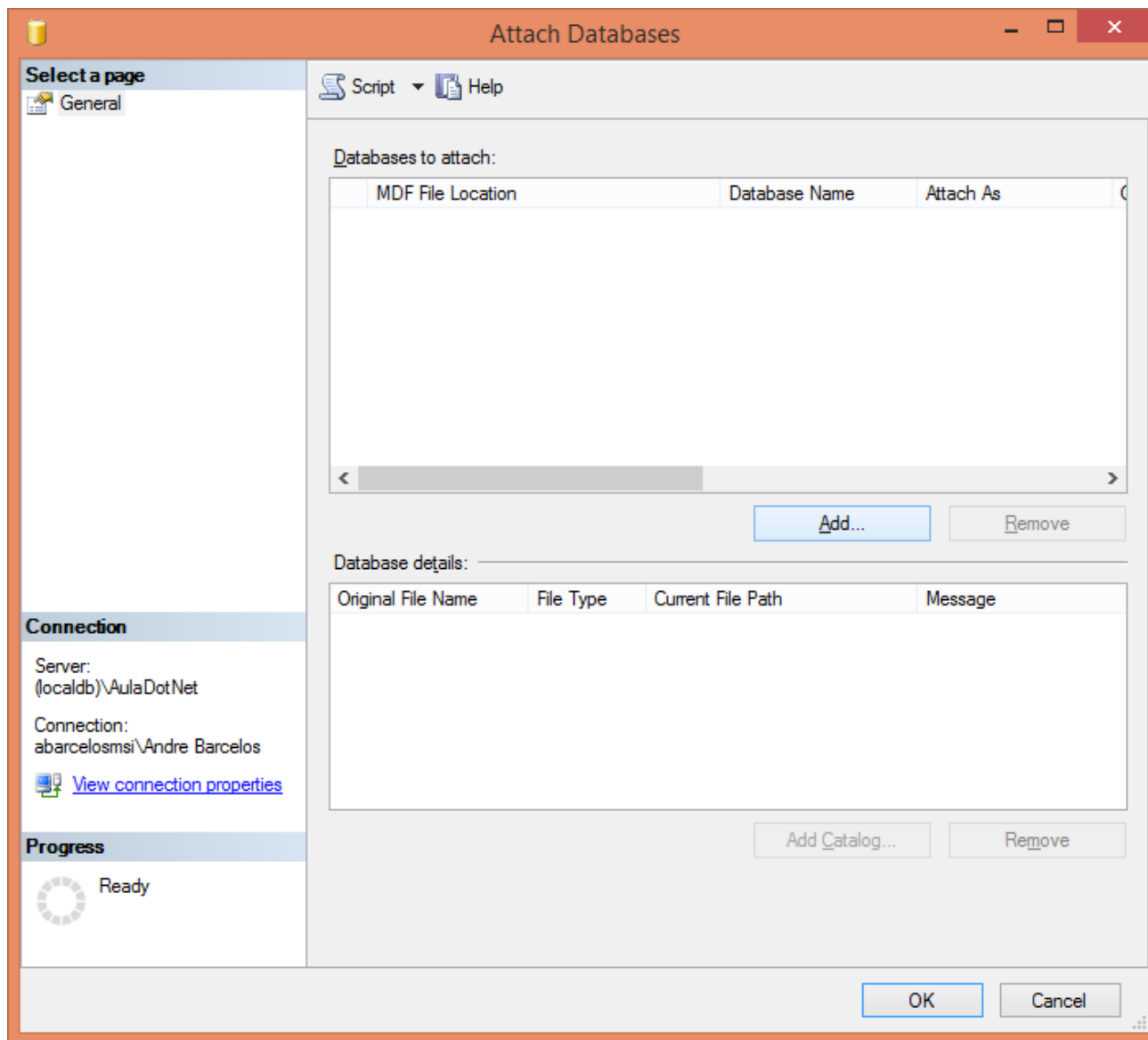


# SQL Server Management Studio

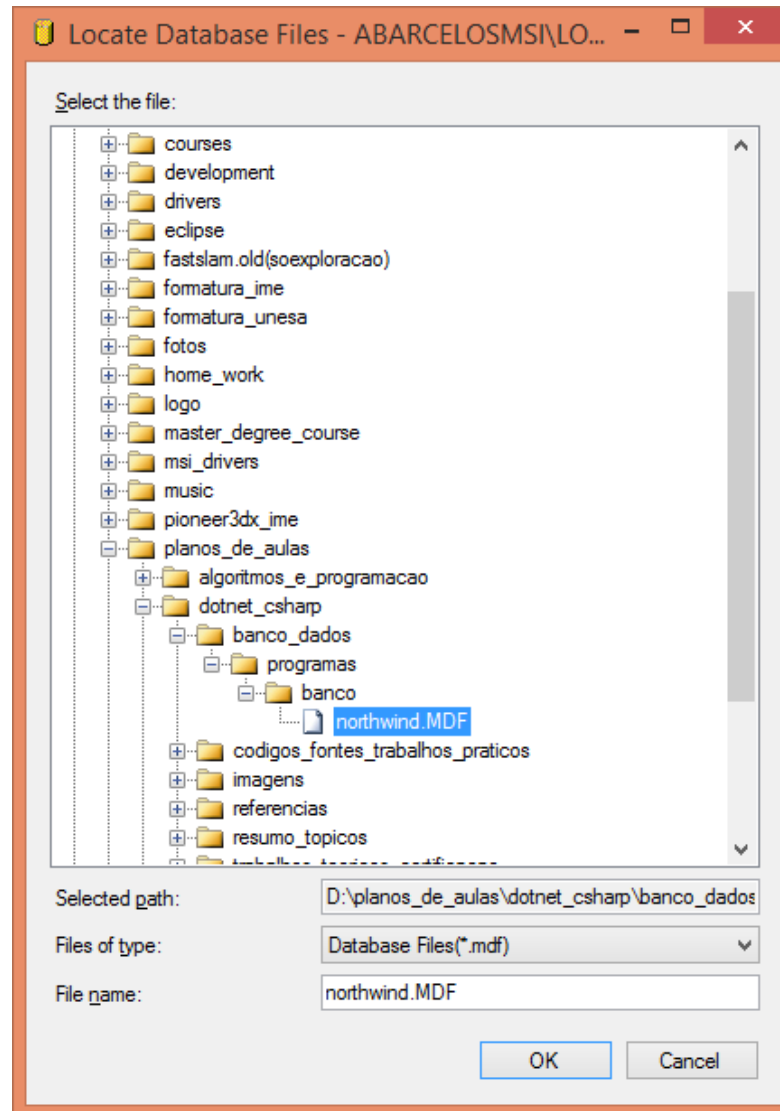
---

A janela de “Attach Databases” se abrirá. Clique no botão “Add”, selecione o arquivo “NORTWIND.MDF”.

# SQL Server Management Studio



# SQL Server Management Studio



# SQL Server Management Studio

---

O banco de dados, assim como os seus detalhes serão mostrados na janela “Attach Databases”.

Aqui haverá a necessidade de excluirmos dos detalhes do banco, o arquivo de “log”, pois este modelo aponta para um arquivo inexistente (oriundo do computador onde foi criado o modelo). O LocalDB irá criar um novo arquivo de log.

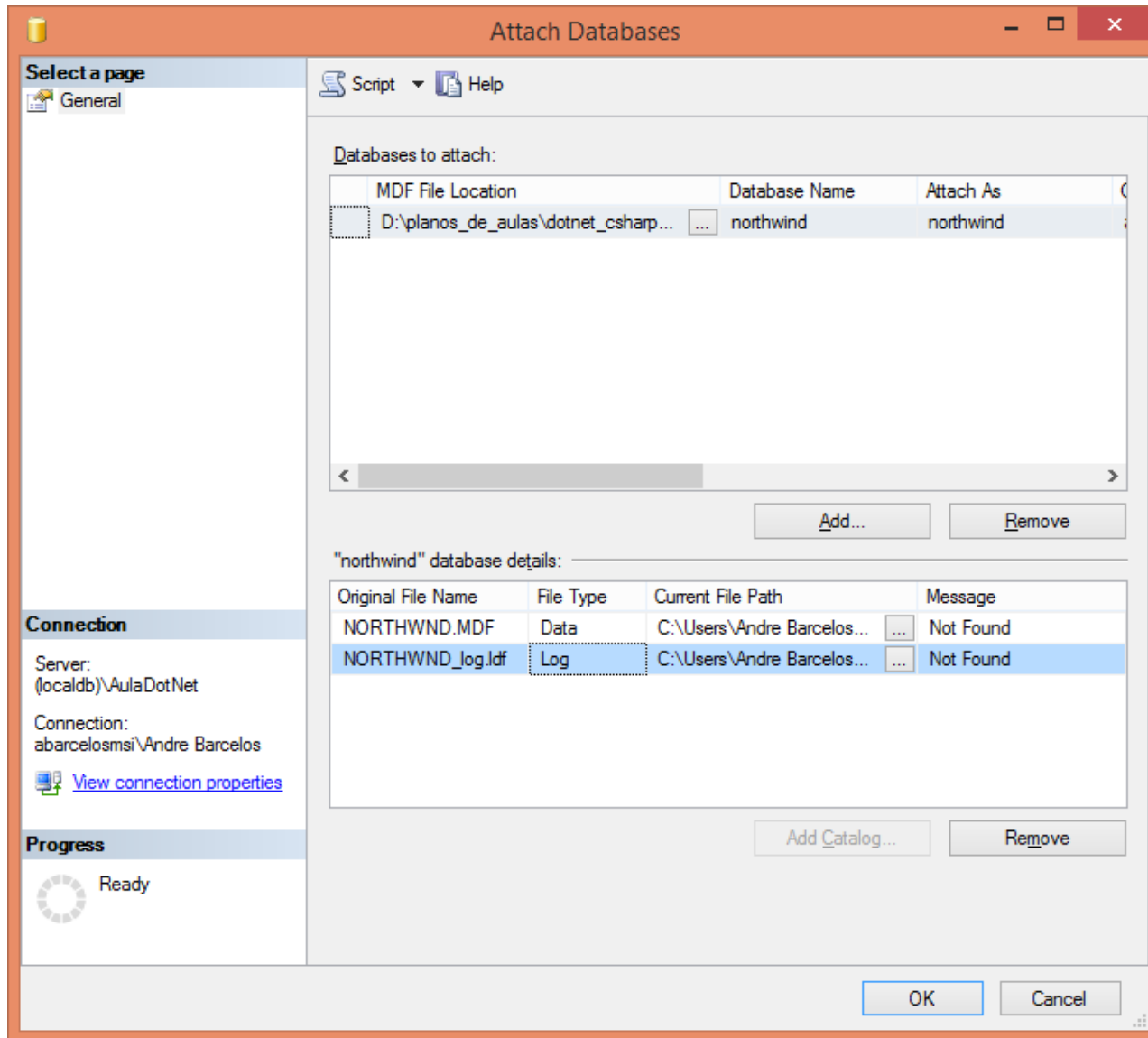
# SQL Server Management Studio

---

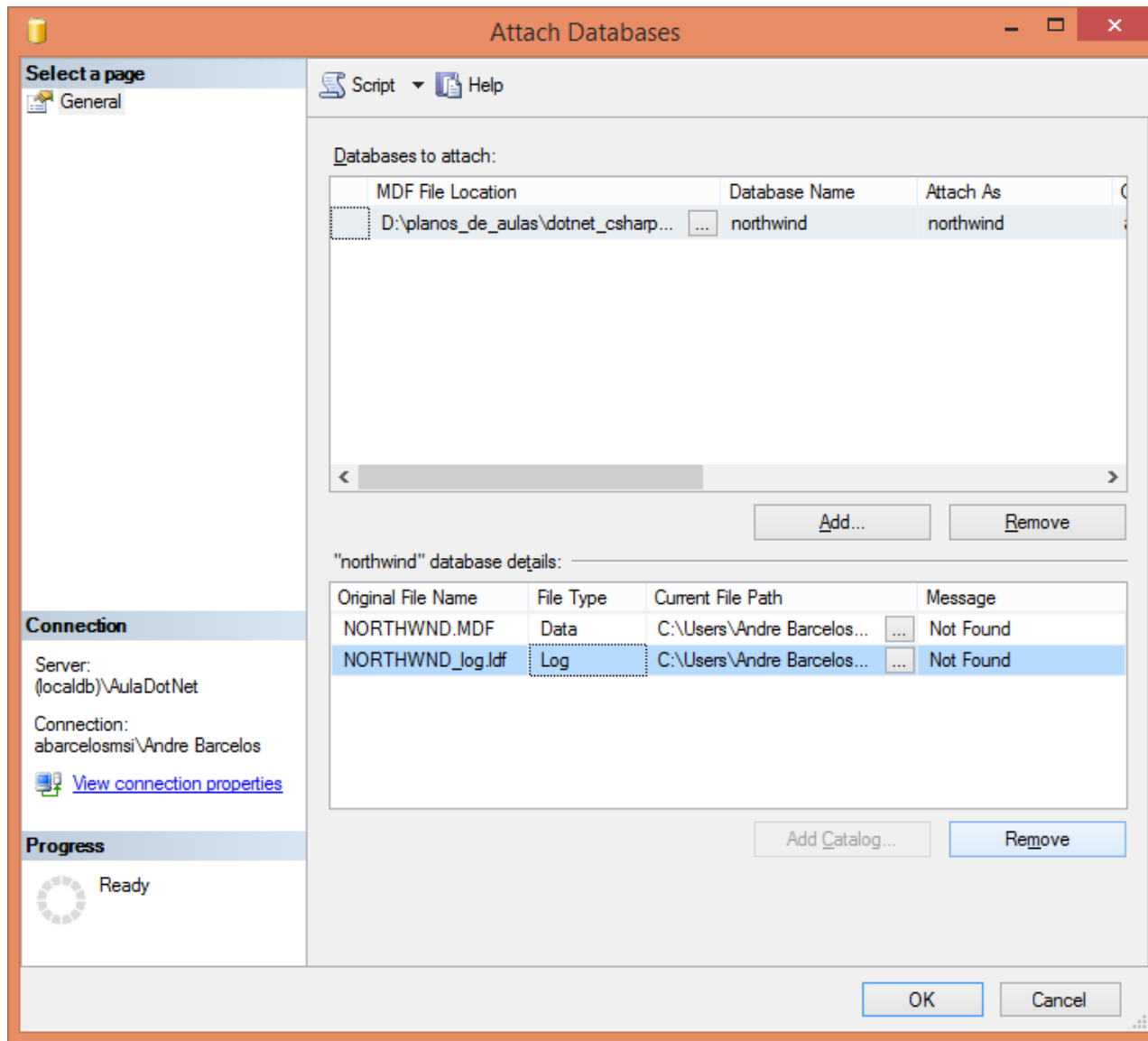
Nos detalhes também é possível escolher se os dados serão importados junto com a criação. Neste caso, como queremos utilizar para exemplo, iremos deixa-lo. Assim teremos dados para serem manipulados nos próximos exercícios.



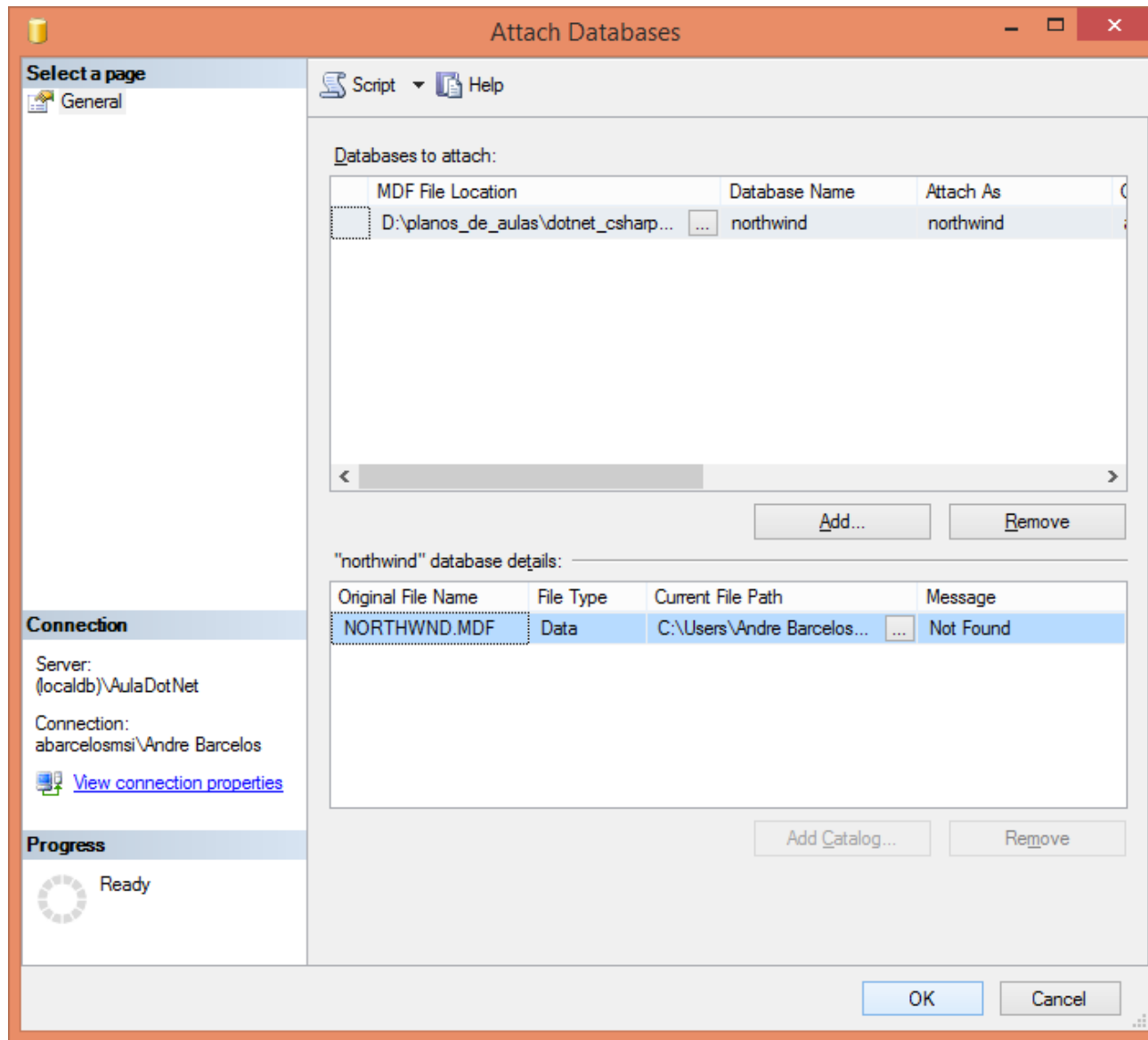
# SQL Server Management Studio



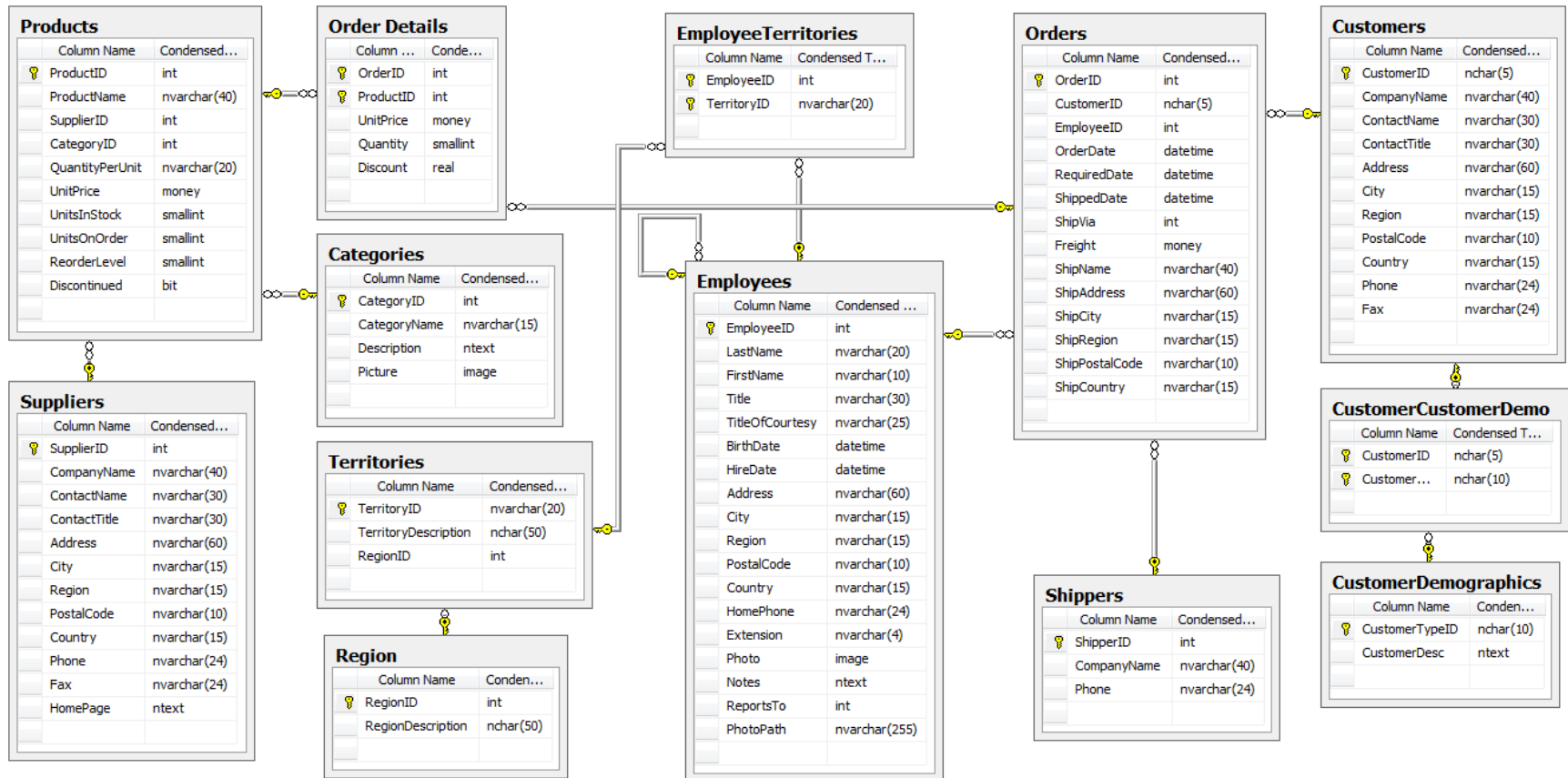
# SQL Server Management Studio



# SQL Server Management Studio



# Banco Northwind Traders (Exemplo)



# Banco Northwind Traders (Exemplo)

Basicamente iremos utilizar duas tabelas:

- Products
- Suppliers

O relacionamento entre estas tabelas é de “muitos-para-um”. Cada produto é fornecido por um único fornecedor, mas cada fornecedor pode fornecer muitos produtos.

# Visualizar dados no formulário

---

Vamos utilizar um assistente do Visual Studio 2012 e exibir os dados de uma tabela num controle DataGridView, utilizando um formulário do Windows Forms Application.

# Tipo-estrutura comum (struct)

---

## Exercício 3:

Adicione à Solution uma nova Application para exibir os dados da tabela “Products” do banco “Northwind traders”.

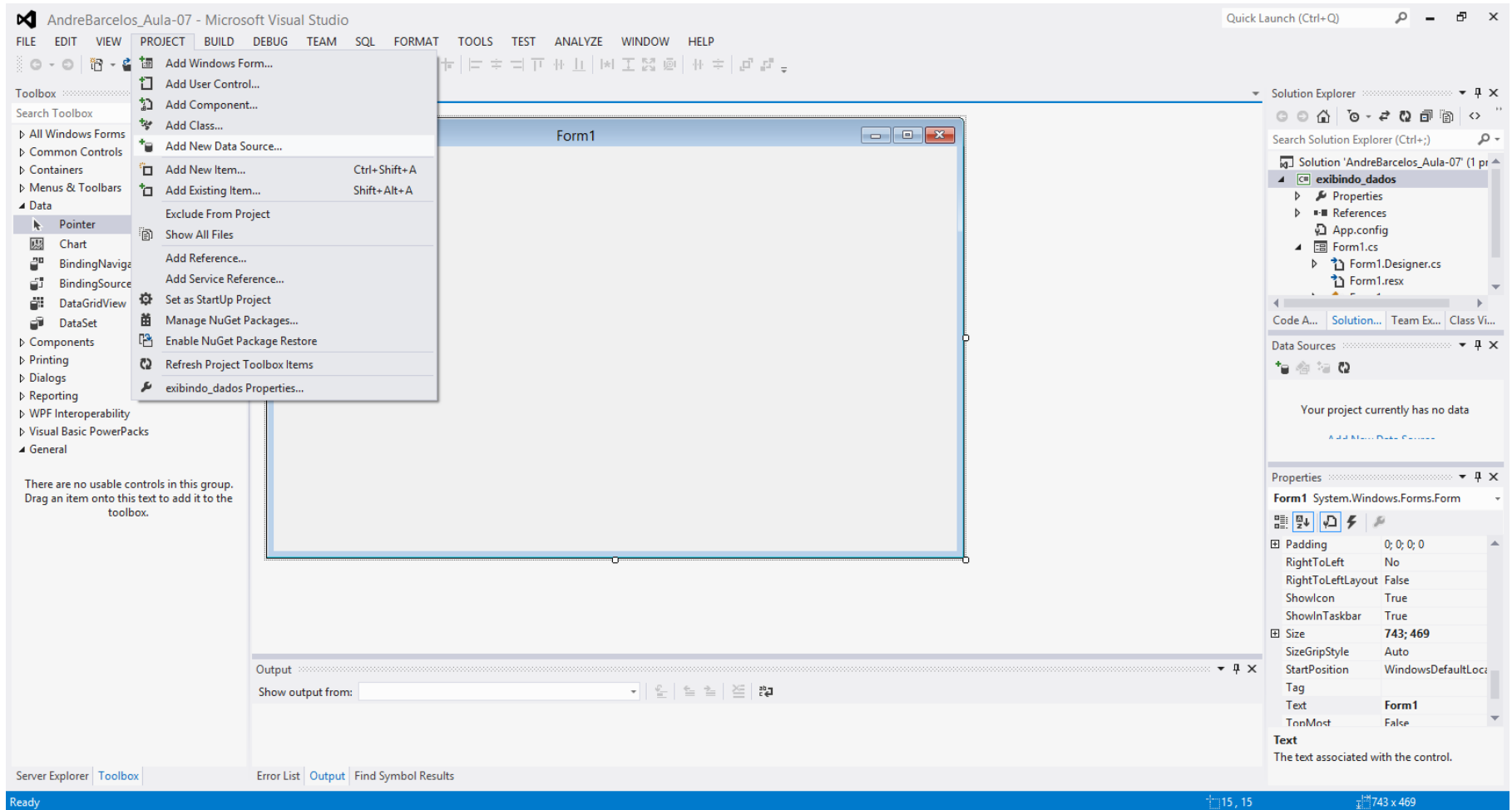
Template: C#; Windows; Windows Form

Name: “soma\_numero\_complexo”

Solution: “Add to solution”

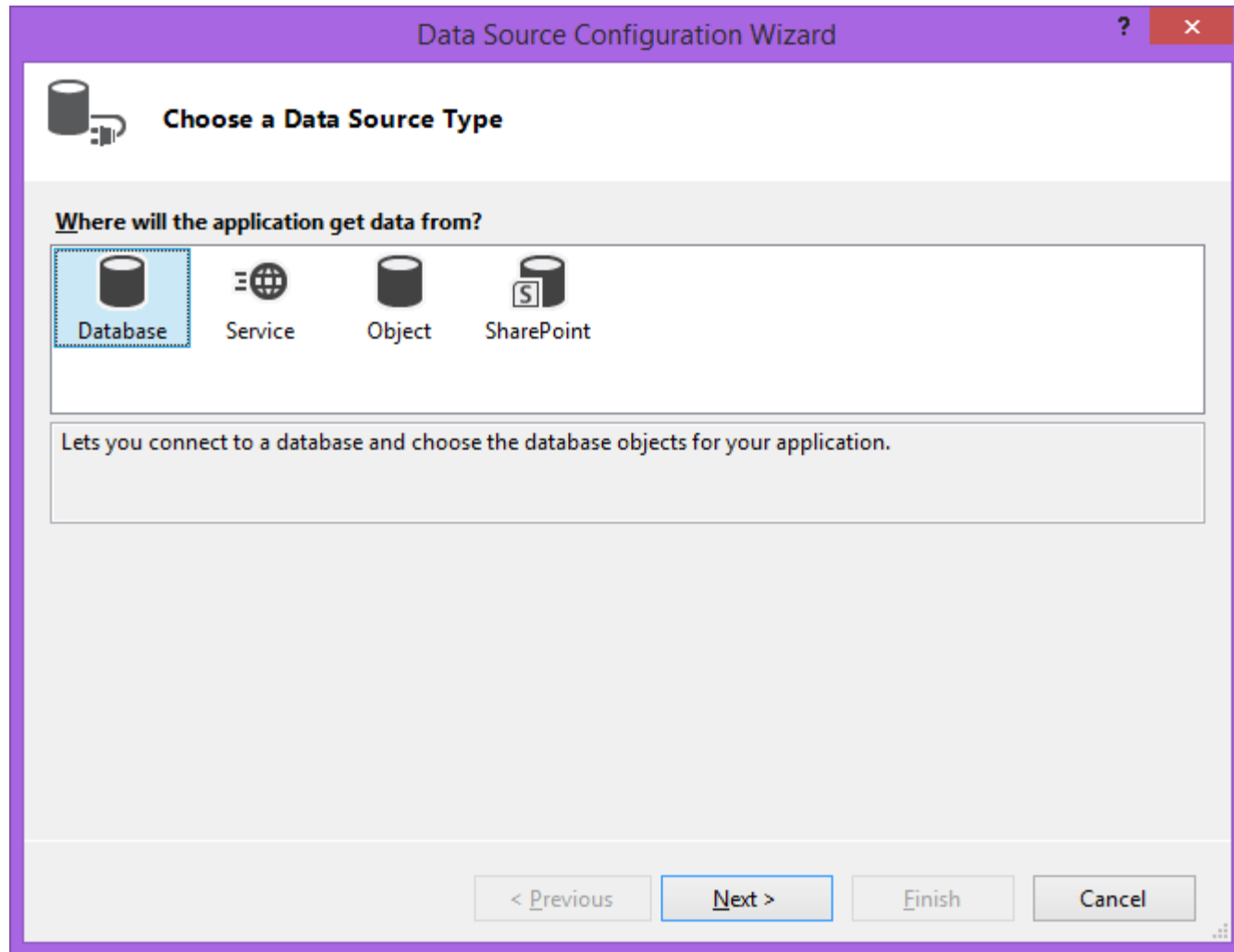


# Visualizar dados no formulário

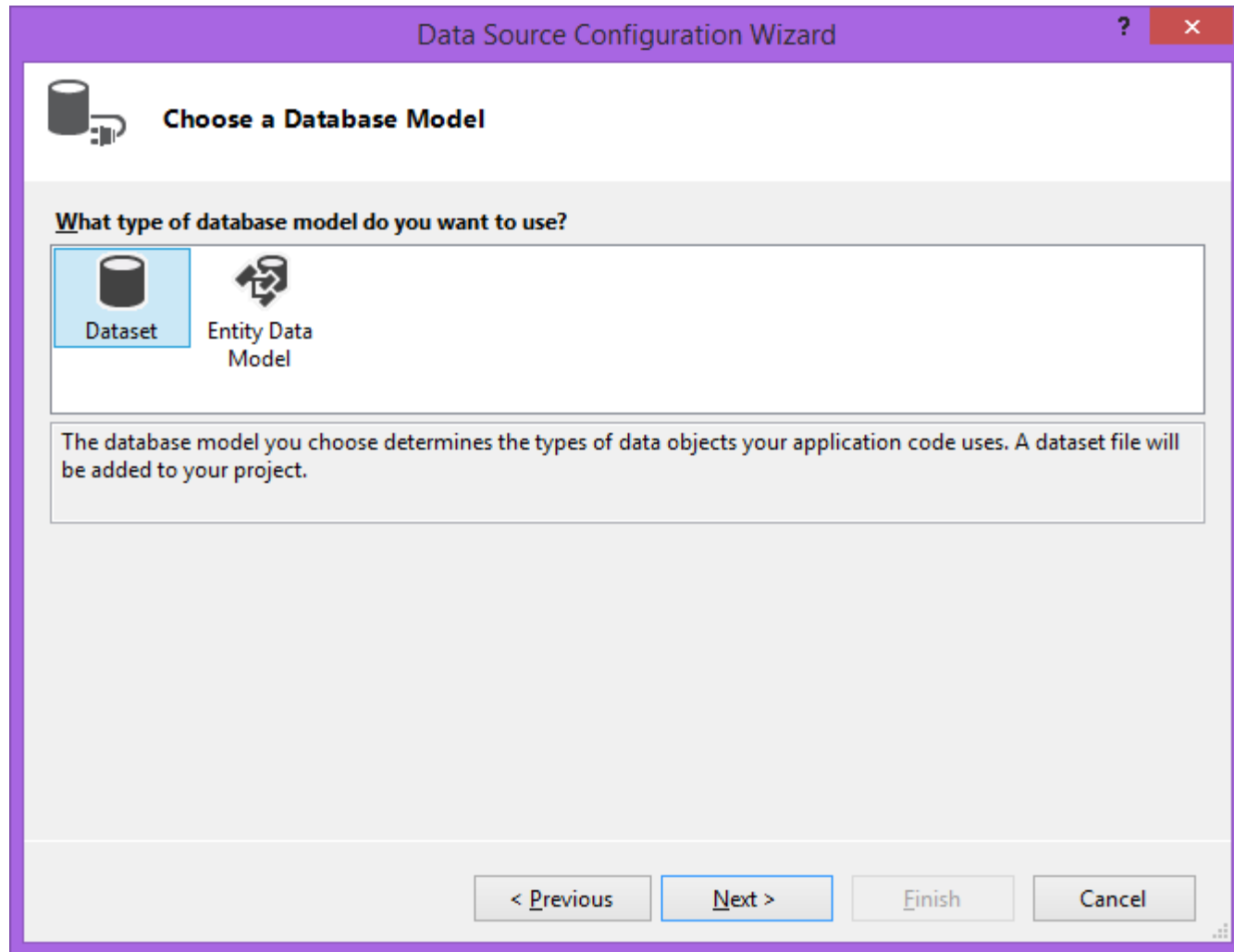




# Visualizar dados no formulário



# Visualizar dados no formulário



# Visualizar dados no formulário

Data Source Configuration Wizard

**Choose Your Data Connection**

Which data connection should your application use to connect to the database?

New Connection...

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set this information in my application code.

☐ Yes, include sensitive data in the connection string.

+ Connection string that you will save in the application (expand to see details)

< Previous   Next >   Finish   Cancel

# Visualizar dados no formulário

**Add Connection** ? x

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:  
Microsoft SQL Server (SqlClient) Change...

Server name:  
(localdb)\AulaDotNet Refresh

Log on to the server

☒ Use Windows Authentication  
☐ Use SQL Server Authentication

User name:   
Password:   
☐ Save my password

Connect to a database

☒ Select or enter a database name:  
northwind ▼

☐ Attach a database file:  
 Browse...  
Logical name:

Advanced...

Test Connection OK Cancel

# Visualizar dados no formulário

The screenshot displays the Microsoft Visual Studio IDE with the 'Form1.cs [Design]\*' window active. The 'Add Connection' dialog box is open, showing the 'Data source' as 'Microsoft SQL Server (SqlClient)' and the 'Server name' as '(localdb)\v11.0'. The 'Log on to the server' section has 'Use Windows Authentication' selected. A 'Test connection succeeded' message box is overlaid on the dialog. The 'Properties' window on the right shows the 'Form1' control with various properties like 'Padding', 'Size', and 'Text'.

Microsoft Visual Studio

AndreBarcelos\_Aula-07 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ANALYZE WINDOW HELP

Quick Launch (Ctrl+Q)

Form1.cs [Design]\*

Search Toolbox

- All Windows Forms
- Common Controls
- Containers
- Menus & Toolbars
- Data
  - Pointer
  - Chart
  - BindingNavigator
  - BindingSource
  - DataGridView
  - DataSet
- Components
- Printing
- Dialogs
- Reporting
- WPF Interoperability
- Visual Basic PowerPacks
- General

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Which data connection do you want to use?

This connection database. How sensitive data in it is?

- No, exclude sensitive data
- Yes, include sensitive data

Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient) Change...

Server name: (localdb)\v11.0 Refresh

Log on to the server

- Use Windows Authentication
- Use SQL Server Authentication

User name: Password:

Connect to a data source

- Select or enter a data source name: northwind
- Attach a database file: Browse...

Logical name:

Advanced...

Test Connection OK Cancel

Microsoft Visual Studio

Test connection succeeded.

OK

Connection...

to connect to the data source to include this data source in the code.

Cancel

Code A... Solution... Team Ex... Class Vi...

Data Sources

Your project currently has no data

Add New Data Source...

Properties

Form1 System.Windows.Forms.Form

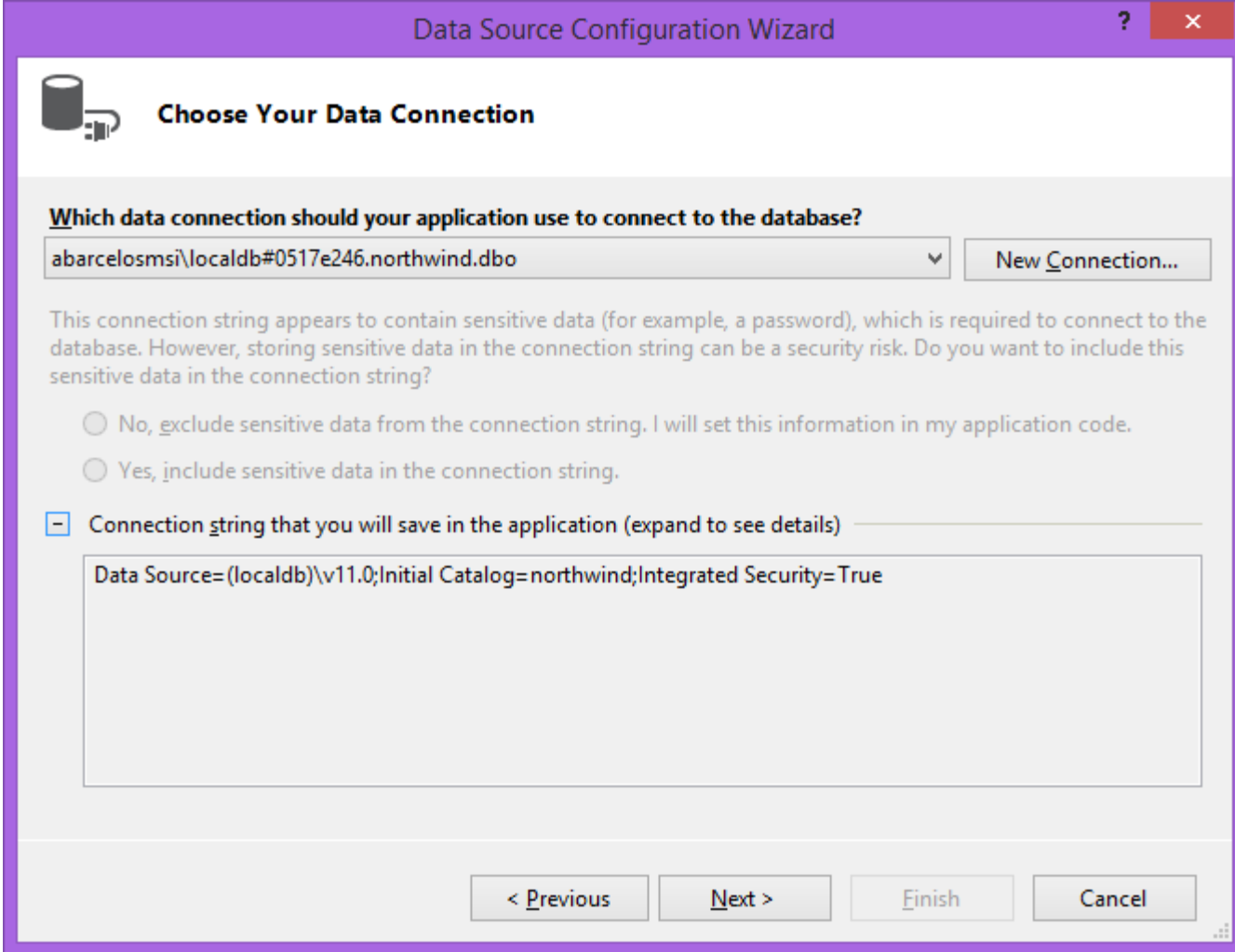
- Padding: 0; 0; 0; 0
- RightToLeft: No
- RightToLeftLayout: False
- ShowIcon: True
- ShowInTaskbar: True
- Size: 743; 469
- SizeGripStyle: Auto
- StartPosition: WindowsDefaultLocation
- Tag: Form1
- Text: The text associated with the control.

Server Explorer Toolbox Error List Output Find Symbol Results

Ready

15, 15 743 x 469

# Visualizar dados no formulário



The screenshot shows the 'Data Source Configuration Wizard' window. The title bar is purple with a question mark and a close button. The main area has a light gray background. At the top left, there is a database icon and the text 'Choose Your Data Connection'. Below this, a question asks 'Which data connection should your application use to connect to the database?'. A dropdown menu shows 'abarclosmsi\localdb#0517e246.northwind.dbo'. To the right of the dropdown is a button labeled 'New Connection...'. Below the dropdown, a paragraph explains that the connection string contains sensitive data and asks if it should be included. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set this information in my application code.' (selected) and 'Yes, include sensitive data in the connection string.'. Below the radio buttons is a checkbox labeled 'Connection string that you will save in the application (expand to see details)'. The checkbox is checked, and a text box below it shows the connection string: 'Data Source=(localdb)\v11.0;Initial Catalog=northwind;Integrated Security=True'. At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Data Source Configuration Wizard

**Choose Your Data Connection**

Which data connection should your application use to connect to the database?

abarclosmsi\localdb#0517e246.northwind.dbo

New Connection...

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set this information in my application code.

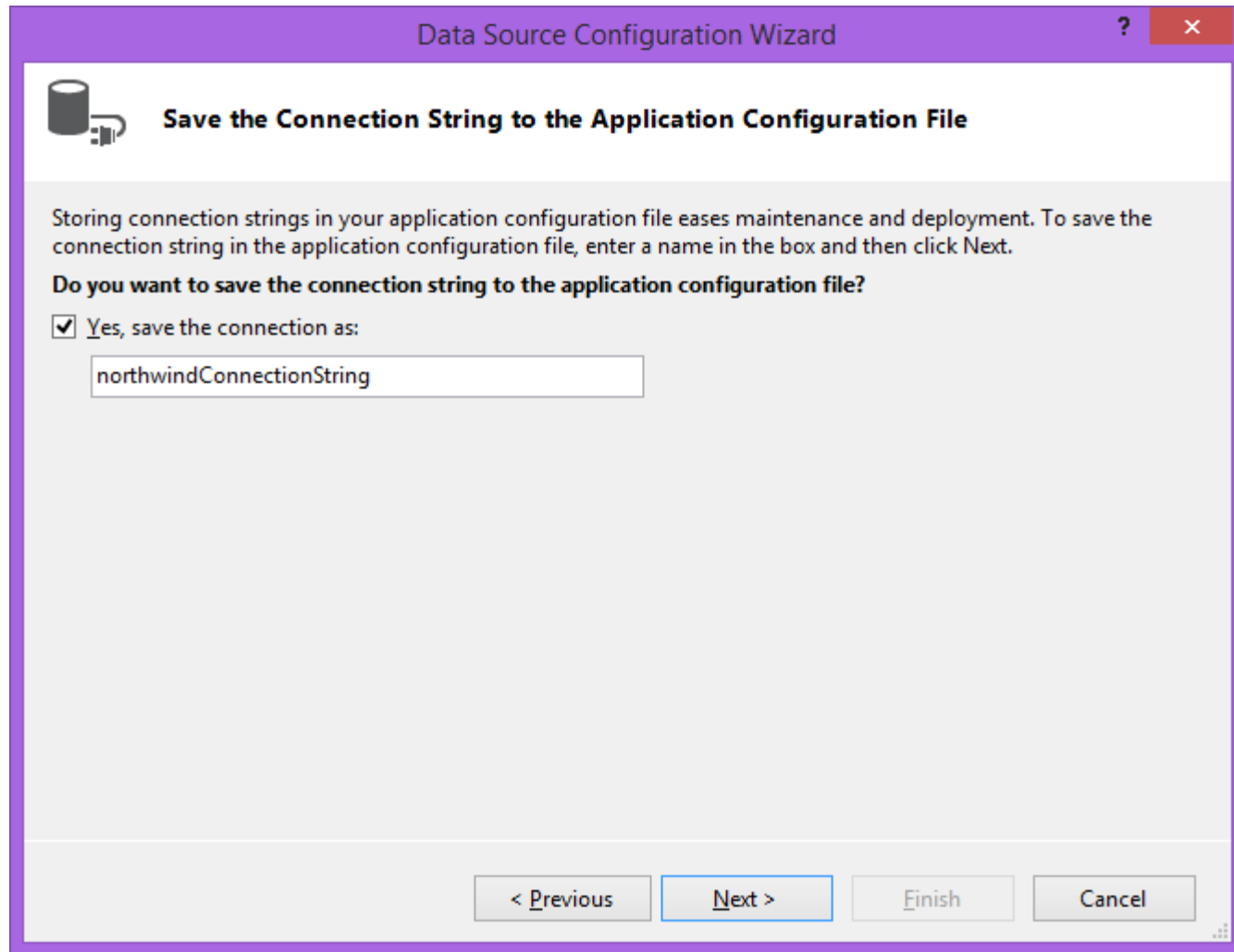
☐ Yes, include sensitive data in the connection string.

☒ Connection string that you will save in the application (expand to see details)

Data Source=(localdb)\v11.0;Initial Catalog=northwind;Integrated Security=True

< Previous   Next >   Finish   Cancel

# Visualizar dados no formulário



Data Source Configuration Wizard

**Save the Connection String to the Application Configuration File**

Storing connection strings in your application configuration file eases maintenance and deployment. To save the connection string in the application configuration file, enter a name in the box and then click Next.

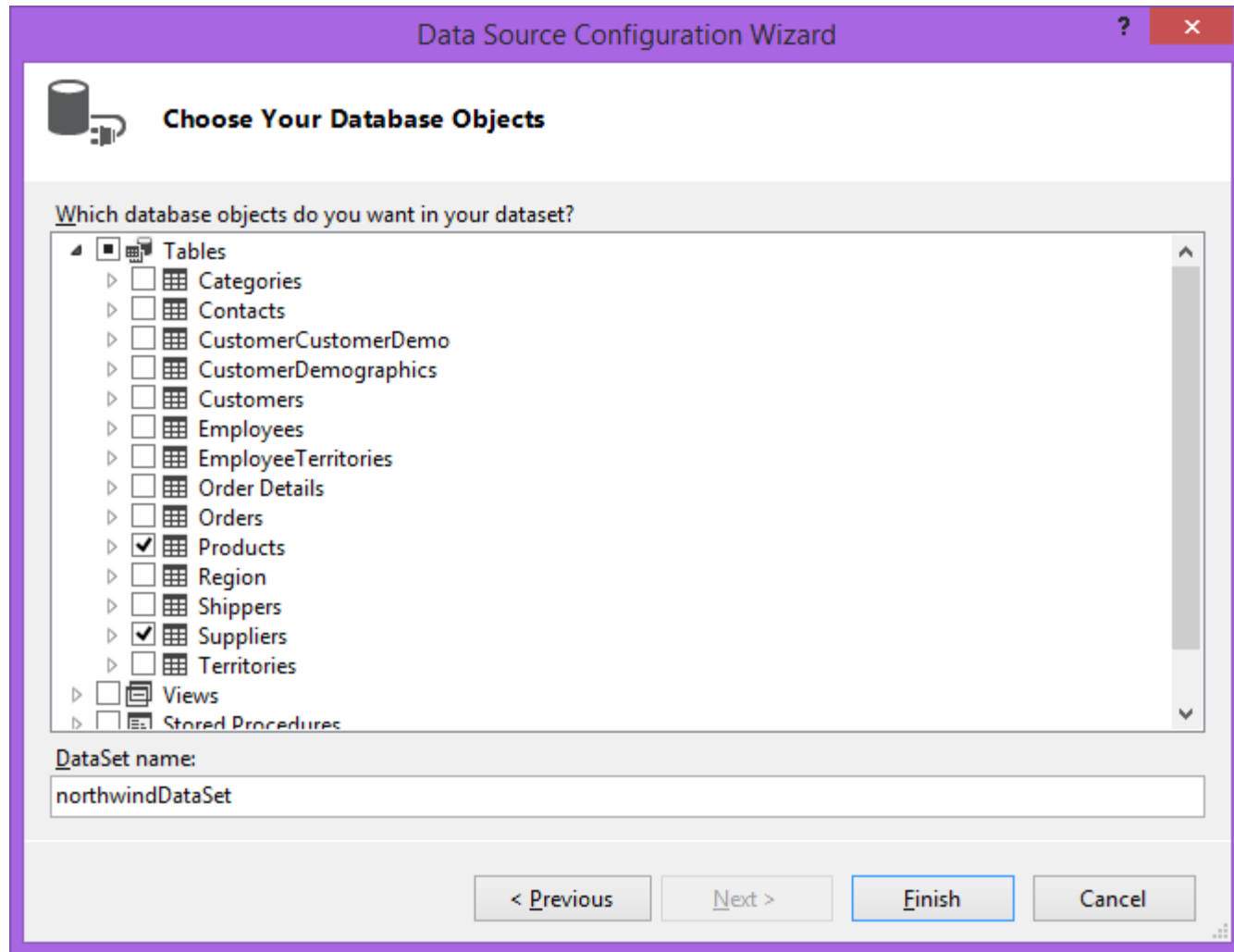
Do you want to save the connection string to the application configuration file?

☒ Yes, save the connection as:

northwindConnectionString

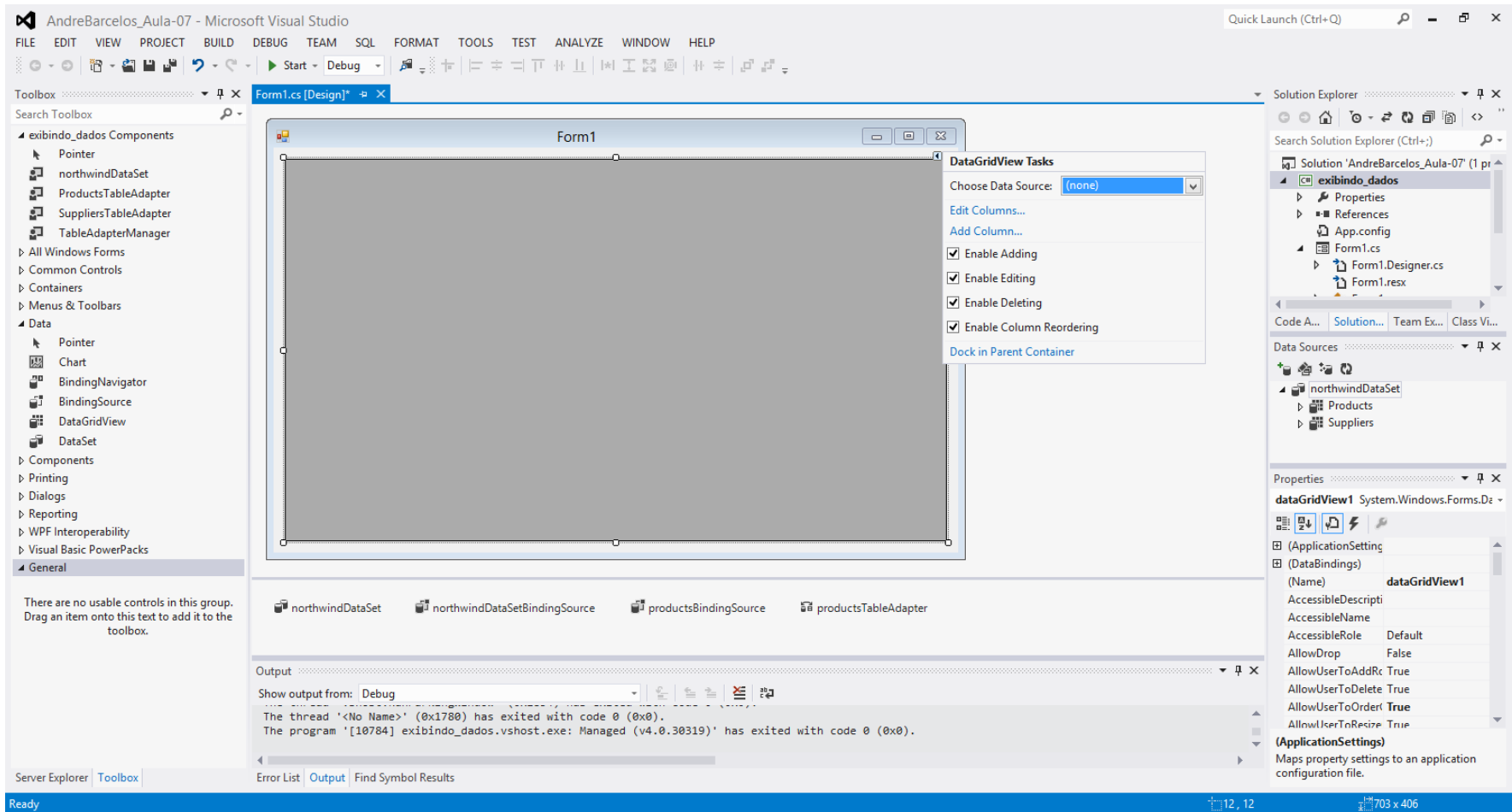
< Previous   Next >   Finish   Cancel

# Visualizar dados no formulário





# Visualizar dados no formulário



# Visualizar dados no formulário

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Toolbox:** Shows various controls under the 'Data' category, including 'DataGridView'.
- Form1 [Design]:** The main design view showing a 'DataGridView' control on a form.
- DataGridView Tasks:** A context menu is open, showing 'Choose Data Source: (none)'. The 'northwindDataSetBindingSource' is selected, and the 'Products' table is chosen from the list.
- Solution Explorer:** Shows the project structure for 'AndreBarcelos\_Aula-07', including 'Form1.Designer.cs' and 'Form1.resx'.
- Properties Window:** Shows the properties for 'dataGridView1', including 'DataSource' and 'DataMember'.
- Output Window:** Displays the output of the application, showing that the thread has exited with code 0 (0x0).

The 'DataGridView' control is currently empty, indicating that the data binding process is in progress or about to be completed.

# Acesso a banco de dados (ADO.NET)

Para acessar um banco de dados, são necessárias, basicamente as seguintes classes da ADO.NET:

- SqlConnection; (Conexão com o banco)
- SqlCommand; (Comando para o banco)
- SqlDataReader; (Manipulação dos dados)

É necessário utilizar a diretiva “`using System.Data.SqlClient;`” (Para o SQLServer) e ter acesso à essas classes.

# Acesso a banco de dados (ADO.NET)

De maneira auxiliar (para facilitar o trabalho em alguns casos) temos as classes:

- **SqlDataAdapter;** (Retorna dados e preenche um DataTable)
- **DataTable;** (Guarda a estrutura completa de uma tabela)

# Acesso a banco de dados (ADO.NET)

Os passos básicos para se acessar um banco são:

1. Abrir uma conexão;
2. Executar um comando;
3. Manipular os dados/Verificar modificações;
4. Fechar a conexão.

# Acesso a banco de dados (ADO.NET)

## 1. Abrir uma conexão;

Para uma conexão ser feita, um endereço do banco (e a instância de gerenciamento que o contém) precisa ser definida. Para isso, utiliza-se uma String de Conexão (ConnectionString).

O Objeto de conexão pode ser instanciado com um argumento de ConnectionString.



# Acesso a banco de dados (ADO.NET)

## 1. Abrir uma conexão; (Instanciando objeto)

```
SqlConnection sqlConn = new SqlConnection("StringDeConexão")  
  
sqlConn.Open();
```



# Acesso a banco de dados (ADO.NET)

1. A String de Conexão possui, basicamente os seguintes dados:

- Caminho da instância do banco;
  - `Data Source=CAMINHOTOINSTANCIA;`
- Nome do banco:
  - `Initial Catalog=NOMEBANCO;`
- Segurança:
  - `Integrated Security=True;`
  - `User ID=LOGIN; Password=SENHA;`



# Acesso a banco de dados (ADO.NET)

1. A String de Conexão possui, basicamente os seguintes dados:

- Caminho da instância do banco;
  - `Data Source=CAMINHODINSTANCIA;`
- Nome do banco:
  - `Initial Catalog=NOMEBANCO;`
- Segurança:
  - `Integrated Security=True;`

PS: O usuário logado no Windows pode, na maioria dos casos, não ter permissões de acesso ao banco!

# Acesso a banco de dados (ADO.NET)

## 2. Executar um comando;

Para executar um comando no banco de dados, é necessário usar um objeto da classe SqlCommand.

# Acesso a banco de dados (ADO.NET)

## 2. Executar um comando; (Instanciando objeto)

```
SqlCommand sqlCommand = new SqlCommand();  
sqlCommand.CommandText = "COMANDO SQL";
```

```
//para caso que retorna registros para serem manipulados  
(SELECT)  
sqlCommand.ExecuteReader();
```

```
//para o caso que não retorna registros (UPDATE, DELETE.  
CREATE)  
sqlCommand.ExecuteNonQuery();
```

# Acesso a banco de dados (ADO.NET)

## 3. Manipular registros; (ExecuteReader());)

```
SqlCommand sqlCommand = new SqlCommand();  
sqlCommand.CommandText = "COMANDO SQL";
```

```
//para caso que retorna registros para serem manipulados  
(SELECT)  
sqlCommand.ExecuteReader();
```

# Acesso a banco de dados (ADO.NET)

## 3. Manipular registros; (ExecuteReader());)

Para manipular os registros, é necessário usar a classe `DataReader`. Com a `DataReader`, é possível ter acesso aos valores de cada registro retornado.

# Acesso a banco de dados (ADO.NET)

## 3. Manipular registros; (Instanciando objeto)

```
SqlDataReader dataReader;
```

```
//passando a referência para ser manipulada pelo DataReader  
dataReader = sqlCommand.ExecuteReader();
```



# Acesso a banco de dados (ADO.NET)

## 3. Manipular registros; (Instanciando objeto)

```
SqlDataReader dataReader;
```

```
//passando a referência para ser manipulada pelo DataReader  
dataReader = sqlCommand.ExecuteReader();
```

```
dataReader.Read()) //retorna true sempre que há registro  
próximo e itera automaticamente (pode ser usado no while)
```

```
dataReader["ProductName"].ToString() //acesso ao valor do  
campo ProductID
```

# Acesso a banco de dados (ADO.NET)

---

## 3. Verificar modificações; (Instanciando objeto)

```
SqlCommand sqlCommand = new SqlCommand();  
sqlCommand.CommandText = "COMANDO SQL";  
  
//retorna o numero de registros afetados  
sqlCommand.ExecuteNonQuery();  
  
//Exemplo de teste  
if (sqlCommand.ExecuteNonQuery() > 0)  
{  
    Console.WriteLine("Estoque de Produto:{0} Atualizado com Sucesso!", item.Nome);  
}
```



# Acesso a banco de dados (ADO.NET)

## 3. Fechar a conexão;

```
SqlConnection sqlConn =  
    new SqlConnection("Data  
Source=NOMEINSTANCIA;Initial Catalog=NOMEBANCO;Integrated  
Security=True")  
sqlConn.Open();  
  
...  
  
sqlConn.Close();
```



# Projeto Mini-Venda

---

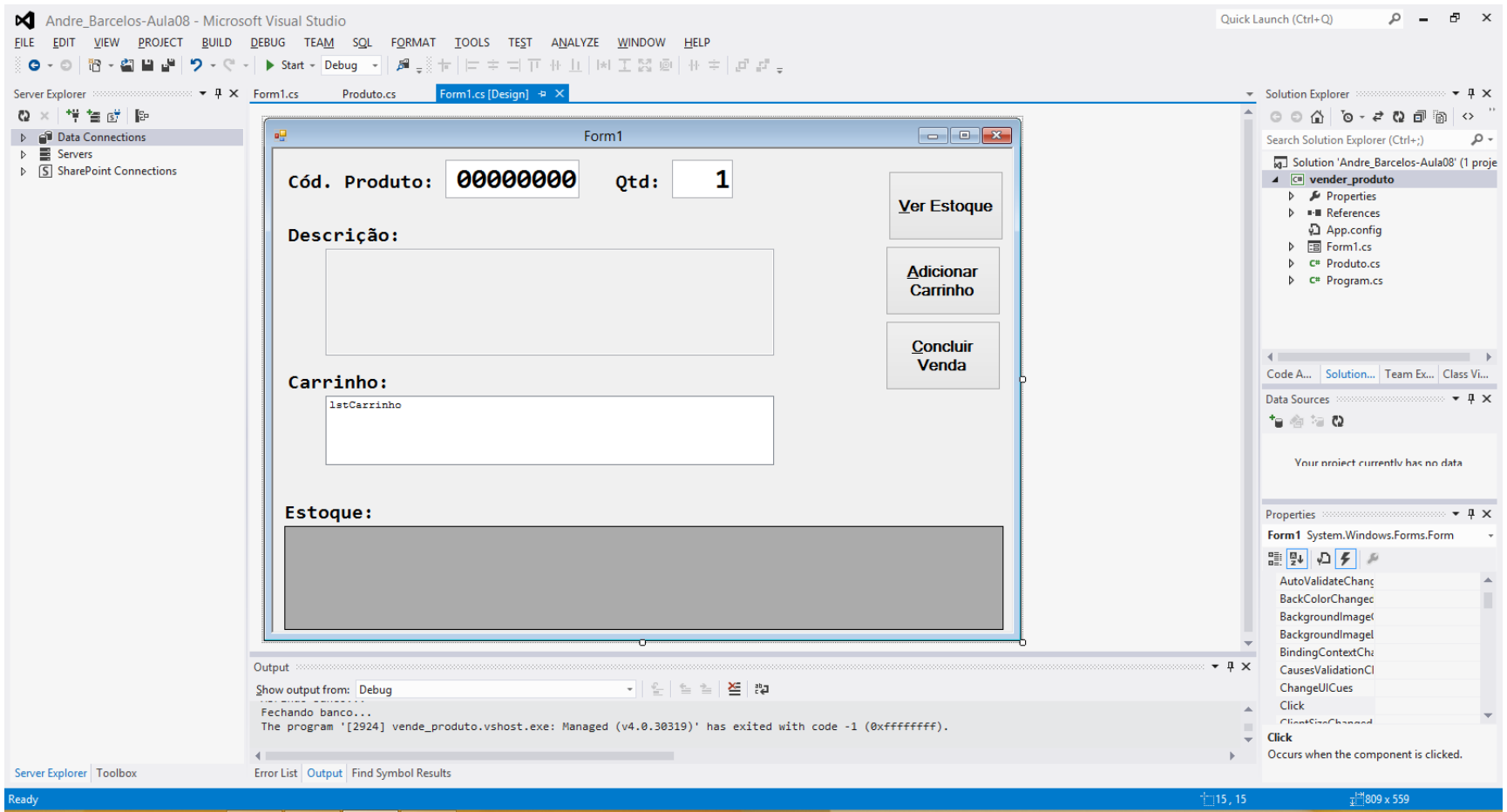
Este projeto consiste em consultar uma tabela e atualizar dados de registros, simulando uma compra comercial.

Baixe o arquivo “Aluno-Aula08.zip”;

Abra o projeto no VS2012;

Renomeie a Solution para “Nome\_Sobrenome-Aula08”.

# Projeto Mini-Venda



# Projeto Mini-Venda

---

Vamos criar uma classe Produto, para representar o registro da tabela Products.

As instâncias desse objeto irão ser alocadas numa coleção (lista) de produtos. Dessa maneira, é possível gerar uma lista de itens daquela compra (pedido).

# Projeto Mini-Venda

---

```
public class Produto
{
    private int id;
    private string nome;
    private int idFornecedor;
    private decimal precoUnitario;
    private double quantidade;

    public Produto(int id, string nome, int idFornecedor, decimal precoUnitario, double quantidade)
    {
        this.Id = id;
        this.Nome = nome;
        this.IdFornecedor = idFornecedor;
        this.PrecoUnitario = precoUnitario;
        this.Quantidade = quantidade;
    }
}
```

# Projeto Mini-Venda

---

```
public class Produto
{
    public double Quantidade
    {
        get { return quantidade; }
        set { quantidade = value; }
    }

    public decimal PrecoUnitario
    {
        get { return precoUnitario; }
        set { precoUnitario = value; }
    }

    public int IdFornecedor
    {
        get { return idFornecedor; }
        set { idFornecedor = value; }
    }

    public string Nome
    {
        get { return nome; }
        set { nome = value; }
    }

    public int Id
    {
        get { return id; }
        set { id = value; }
    }
}
```

# Projeto Mini-Venda

---

```
public partial class Form1 : Form
{
    //instanciando um objeto de produtos
    Produto prod;
    //instanciando uma coleção de produtos para criar o carrinho com os itens de venda
    List<Produto> lstProdutos = new List<Produto>();

    ...
}
```

# Projeto Mini-Venda

---

Será necessário fazer uma consulta ao banco cada vez que o operador digitar um código na caixa de texto “txtCodigo”. Para isso, vamos realizar essa lógica no evento “KeyUp” do TextBox.



# Projeto Mini-Venda

---

```
using (SqlConnection sqlConn =
    new SqlConnection("Data Source=(localdb)\\v11.0;Initial Catalog=northwind;Integrated
Security=True"))
{
    using (SqlCommand sqlCommand = new SqlCommand())
    {
        sqlCommand.Parameters.AddWithValue("PIId", txtCodigo.Text);
        sqlCommand.CommandText = "SELECT ProductID, ProductName," +
            "SupplierID, UnitPrice, UnitsInStock FROM Products WHERE ProductID = @PIId";
        sqlCommand.Connection = sqlConn;
        sqlCommand.Open();
        SqlDataReader dataReader;
        dataReader = sqlCommand.ExecuteReader(); //para caso que retorna registros

        if (dataReader.Read()) //encontrou registro
        {
            prod = new Produto(Int32.Parse(dataReader["ProductID"].ToString()),
                                dataReader[1].ToString(),
                                Int32.Parse(dataReader[2].ToString()),
                                Decimal.Parse(dataReader[3].ToString()),
                                Double.Parse(txtQuantidade.Text));

            //colocando os dados no formulario
            txtDescricao.Text = prod.Nome;
        }
        else
        {
            txtDescricao.Text = "";
            prod = null;
        }
        sqlConn.Close();
    }
}
```

# Projeto Mini-Venda

---

Para popular o DataGridView de forma programada (sem usar os assistentes), devemos consultar o banco e pegar todos os registros da tabela Products, instanciar um objeto DataTable (que pode guardar as informações estruturais da tabela, assim como os dados) e coloca-lo como fonte de dados (DataSource) para o DataGridView.

Esta lógica deverá ser colocada no evento de click do btnVerEstoque.

# Projeto Mini-Venda

---

```
private void btnVerEstoque_Click(object sender, EventArgs e)
{
    SqlConnection sqlConn = new SqlConnection("Data Source=(localdb)\\v11.0;Initial
Catalog=northwind;Integrated Security=True");
    SqlCommand sqlCommand = new SqlCommand();

    try
    {
        Console.WriteLine("Abrindo banco...");

        SqlDataAdapter dAdapter = new SqlDataAdapter("SELECT ProductID, ProductName, " +
"SupplierID, UnitPrice, UnitsInStock FROM Products", sqlConn);

        DataTable table = new DataTable();
        dAdapter.Fill(table);
        gridEstoque.DataSource = table;
    }
    catch (Exception ex)
    {
        Console.WriteLine("Acesso ao banco falhou!\n" + ex.ToString());
    }
    finally
    {
        sqlConn.Close();
        Console.WriteLine("Fechando banco...");
    }
}
```

# Projeto Mini-Venda

---

Um carrinho de compras deverá ser gerado, para que se tenha uma lista de itens que estão sendo comprados naquele pedido.

O objeto `IstProdutos` será o responsável por guardar cada produto da compra. Para isso, haverá um método `AdicionarCarrinho()` que populará o `IstProdutos` e escreverá no controle gráfico `IstCarrinho` que está no form.

# Projeto Mini-Venda

---

```
public void AdicionarCarrinho()
{
    if (prod != null)
    {
        prod.Quantidade = Double.Parse(txtQuantidade.Text);
        lstProdutos.Add(prod); //adiciona a coleção responsável pelos itens de compra

        //adiciona ao controle gráfico do formulário
        lstCarrinho.Items.Add(String.Format("{0,-40} {1,-12} {2,3}", prod.Nome,
                                             prod.PrecoUnitario, prod.Quantidade));
    }
}
```

# Projeto Mini-Venda

---

Após colocar a quantidade, se a tecla “Enter” for pressionada e solta, deve-se adicionar o item ao carrinho de compras.

Para isso, é necessário testar a tecla pressionada no evento KeyUp do TextBox txtQuantidade e, caso seja o Enter, deve-se executar o seguinte trecho de código.



# Projeto Mini-Venda

---

```
private void txtQuantidade_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter) //se for a tecla Enter, adicionar ao carrinho
    {
        AdicionarCarrinho();
    }
}
```

# Projeto Mini-Venda

---

A compra deverá ser fechada, para que a quantidade em estoque de cada produto seja atualizada.

A lógica é atualizar (UPDATE) cada produto da tabela Products, iterando a coleção de produtos, pegar o ID e diminuir a quantidade comprada. A codificação deve ser feita no evento de Click do Button btnFechVenda.



# Projeto Mini-Venda

---

```
private void btnFechVenda_Click(object sender, EventArgs e)
{
    using (SqlConnection sqlConn =
        new SqlConnection("Data Source=(localdb)\\v11.0;Initial Catalog=northwind;Integrated Security=True"))
    {
        using (SqlCommand sqlCommand = new SqlCommand())
        {
            sqlConn.Open();
            //atualizando o estoque dos produtos na tabela
            foreach (var item in lstProdutos)
            {
                sqlCommand.CommandText = "UPDATE Products SET UnitsInStock = UnitsInStock - '" + item.Quantidade +
                    "' WHERE ProductID = '" + item.Id + "'";
                sqlCommand.Connection = sqlConn;

                if (sqlCommand.ExecuteNonQuery() > 0)
                {
                    Console.WriteLine("Estoque de Produto:{0} Atualizado com Sucesso!", item.Nome);
                }
                else
                {
                    Console.WriteLine("Erro ao atualizar estoque Produto:{0}", item.Nome);
                }
            }
            lstProdutos.Clear();
            lstCarrinho.Items.Clear();
            sqlConn.Close();
        }
    }
}
```



# Resumo de aula

---

- Mandem os exercícios de aula para a tarefa no moodle.