Colégio Pedro II

PROGRAMAÇÃO O.O.

(*C#*)

{ Introdução aos Sistemas de Banco de Dados

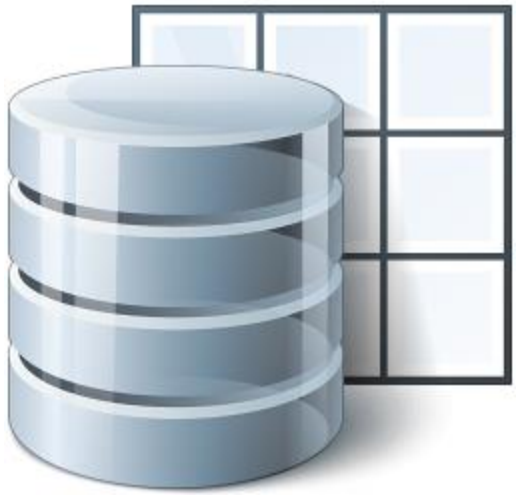
Professor: João Luiz Lagôas }

O que é um banco de dados?

Colégio Pedro II

- De modo geral, uma banco de dados é uma coleção de dados relacionados que tem por objetivo atender a uma comunidade de usuários.

- Há diversas formas de se modelar um banco de dados:
  - **Modelo Relacional (Tabelas)**
  - Modelo Hierárquico (Árvores)
  - Modelo em Rede (Grafos)
  - Modelo O.O. (Classes)



- Hoje, o modelo mais difundido e utilizado na implementação de bancos de dados é o **modelo relacional** que faz uso de tabelas e associações para armazenar dados.

Como era um banco de dados?

Colégio Pedro II

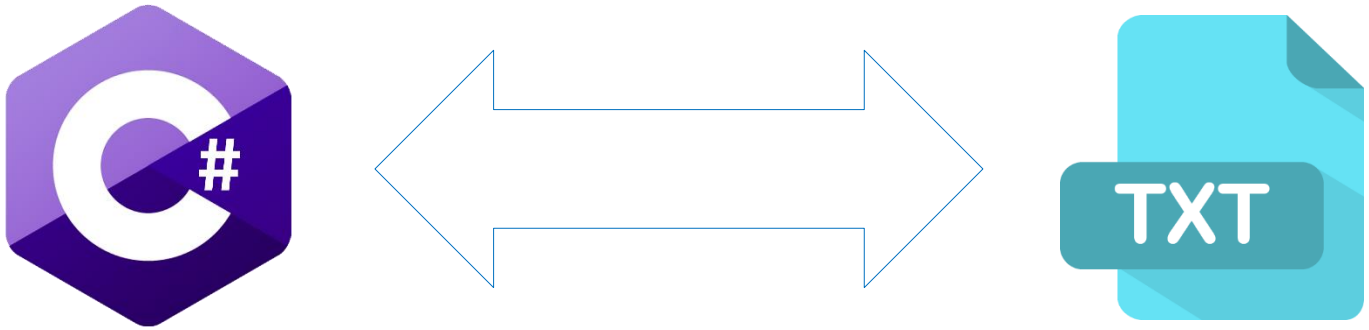
- Tradicionalmente, os dados que eram utilizados por aplicações/programas eram armazenadas em arquivos .txt ou .bin.

- No entanto, essa forma de se armazenar dados apresentava muitos problemas e se tornou imprópria para se gerenciar grandes quantidades de dados com o desenvolvimento da tecnologia da informação.

Abordagem Tradicional

Colégio Pedro II

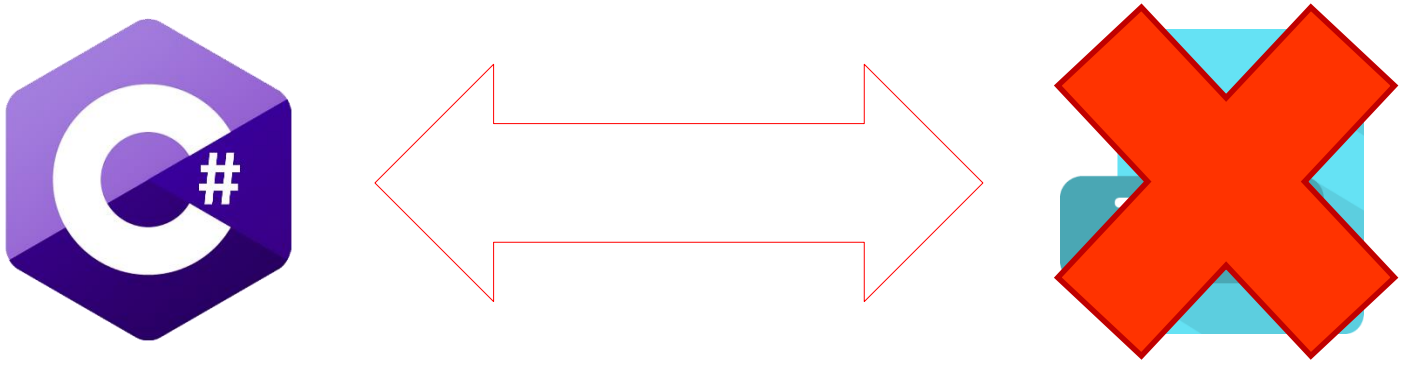
- Problemas:
  - Redundância não controlada e inconsistência de dados;
  - Isolamento dos dados;
  - Falta de integridade;
  - Anomalias de acesso concorrente;
  - Segurança;



Abordagem Tradicional

Colégio Pedro II

- A alternativa para se lidar com grande quantidade de dados e minimizar os problemas citados foi a criação de **Sistemas Gerenciadores de Bancos de Dados**.



SGBD

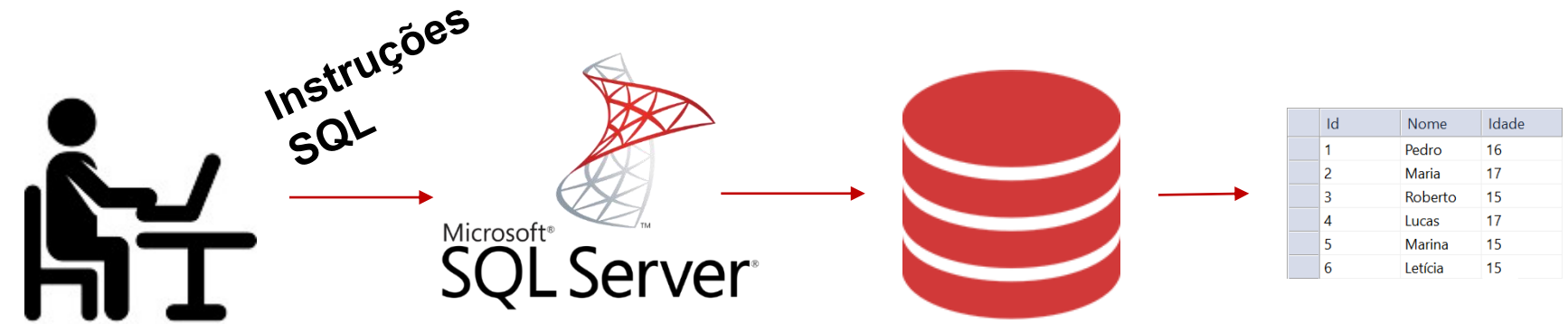
Sistema Gerenciador de Banco de Dados

Colégio Pedro II

- Um SGBD (Sistema Gerenciador de Banco de Dados) é um software que faz a interface entre o **programador ou aplicação** com o **banco de dados** propriamente.

- O usuário, por exemplo, se quiser realizar qualquer alteração em um banco de dados, ele envia instruções SQL para o SGBD e este irá realizar as alterações no banco de dados.

- O propósito geral de um SGBD é **definir**, **atualizar** e **recuperar** dados de um banco de dados, mantendo os dados sempre consistentes e íntegros.



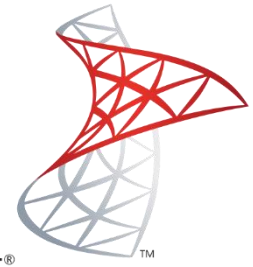
SGBD

Sistema Gerenciador de Banco de Dados

Colégio Pedro II

- Um banco de dados está geralmente armazenado em um formato específico do SGBD que geralmente não apresenta portabilidade, mas diferentes SGBD's podem compartilhar dados usando padrões de consulta a linguagem **SQL**.

- Exemplos de SGBD's:
  - MySQL,
  - Access,
  - SQL Server,
  - PostgreSQL,
  - Oracle, etc.



SQL

Linguagem de Consulta Estruturada

Colégio Pedro II

- "*Structured Query Language*" (SQL) é uma linguagem declarativa padronizada que está associada aos princípios de banco de dados no modelo relacional (modelo de tabelas).

- Os SGBD's atualmente implementam SQL mas cada um deles pode apresentar certas particularidades. De qualquer forma, existe um grande grupo de comandos comuns que são categorizados e existem em todos os SGBD's.

Propósito	Exemplo
Definição	CREATE TABLE
Recuperação	SELECT
Atualização	INSERT INTO, UPDATE, DELETE

Microsoft SQL Server

SGBD do Visual Studio

Colégio Pedro II

- Mantido pela Microsoft há anos e integrado à plataforma de desenvolvimento .NET, o SQL Server é um dos principais SGBDs relacionais do mercado.

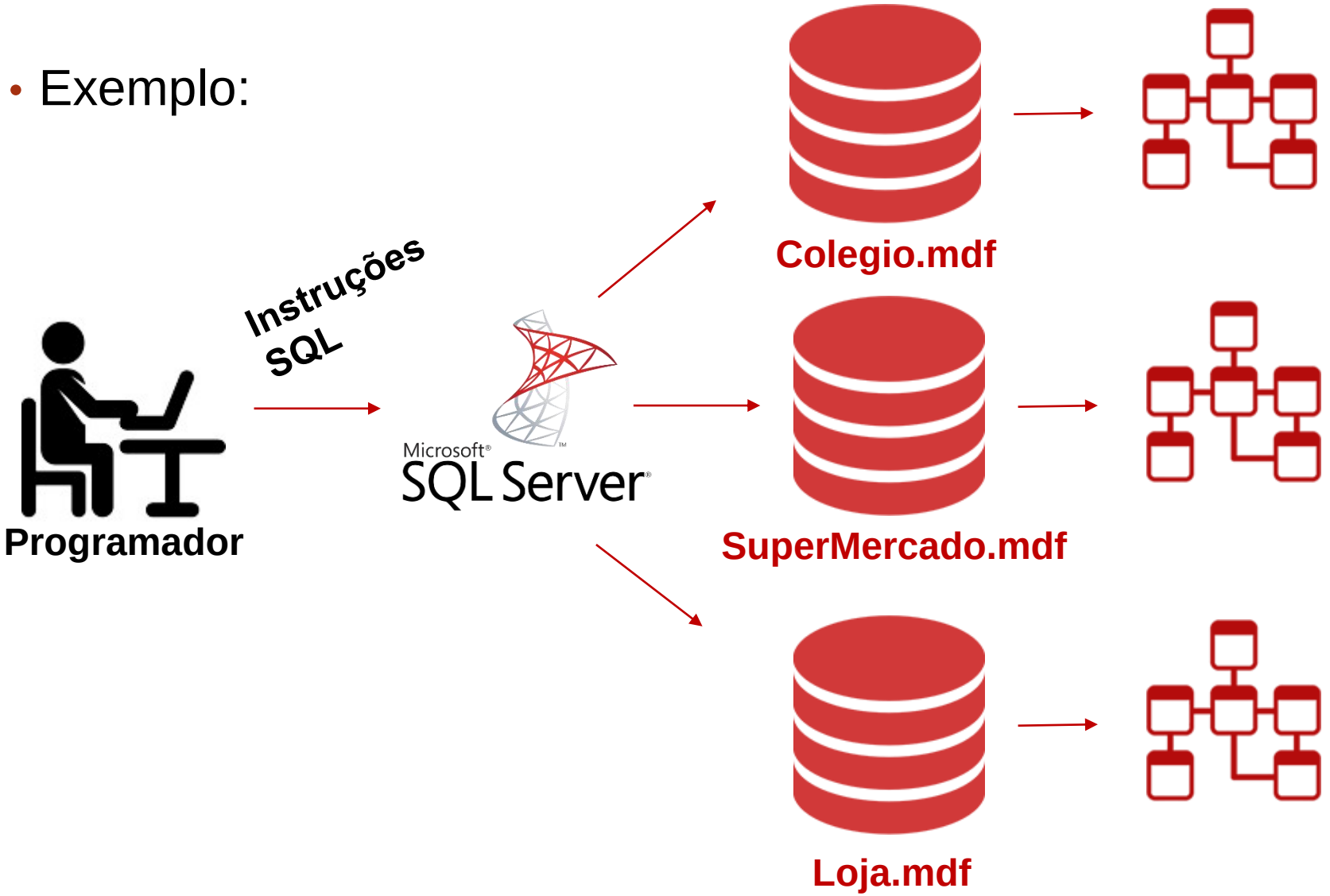
- Bancos de dados SQL Server usam dois arquivos:
  - Um .**mdf** conhecido como **arquivo de banco de dados primário**. Esse arquivo contém os esquemas, tabelas, registros e dados.
  - Um **.ldf** que consiste de um **arquivo de log** (log é uma espécie de registro de ocorrências no banco de dados).

Microsoft SQL Server

SGBD do Visual Studio

Colégio Pedro II

- Exemplo:



Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

1. A sua aplicação deve se conectar com o Banco de Dados gerenciado por algum SGBD.

2. Comandos SQL são enviados através da conexão estabelecida no passo anterior para o SGBD.

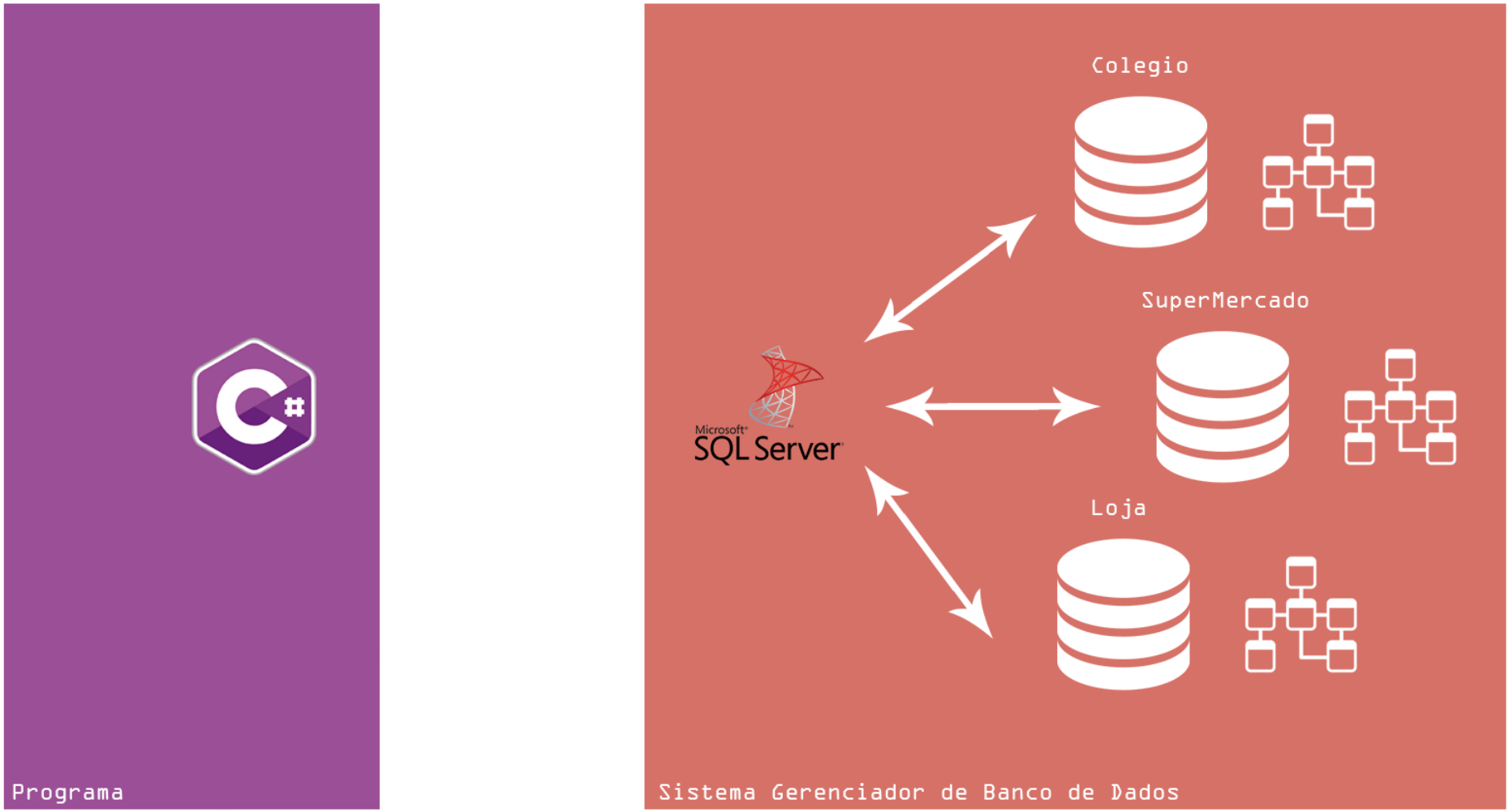
3. O SGBD executa a consulta SQL e devolve para a aplicação uma resposta. Por exemplo: o resultado de uma consulta SELECT.

4. A aplicação recebe a resposta da consulta em um formato específico e a utiliza normalmente na construção de seu código.

Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

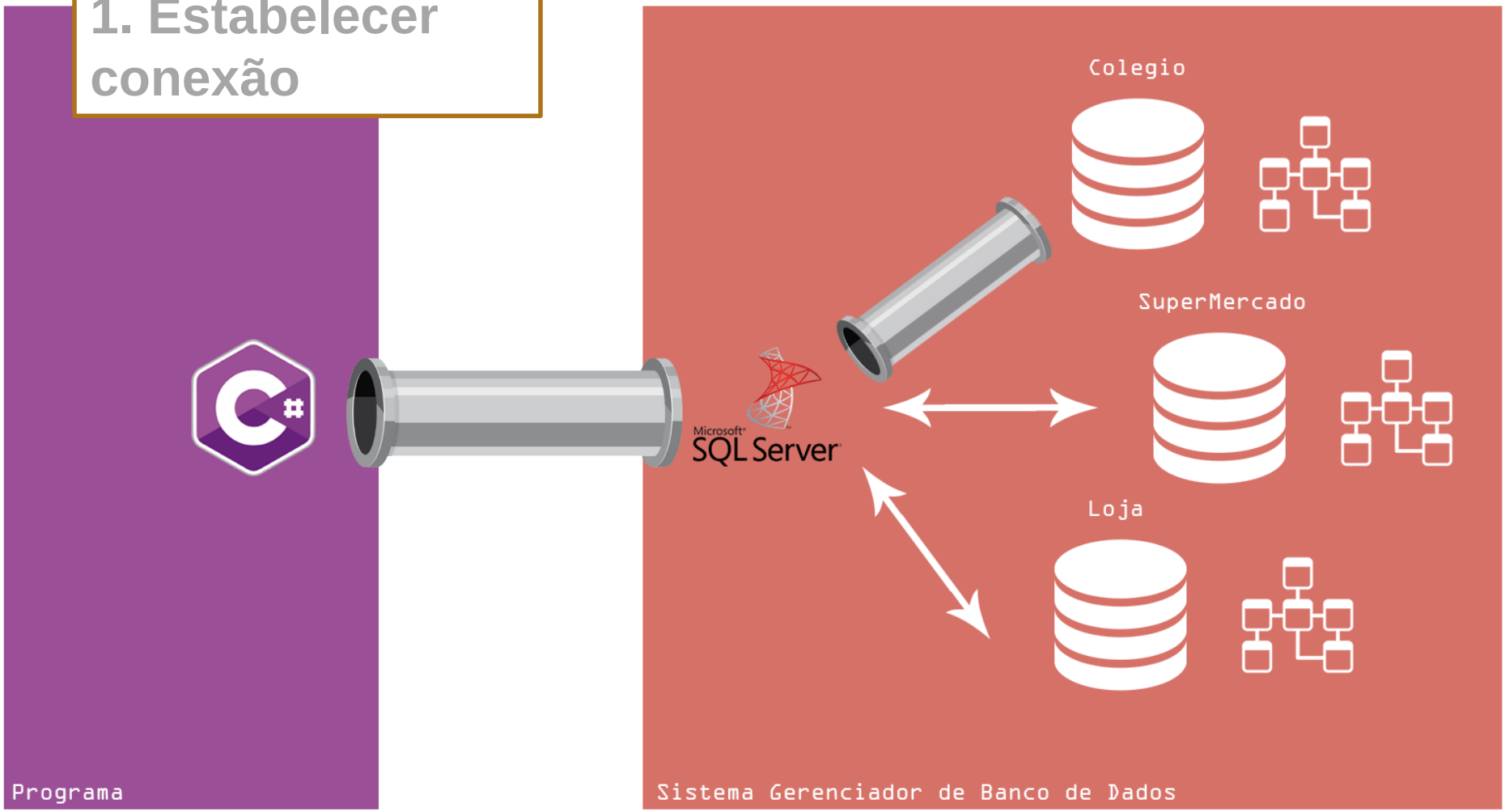


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

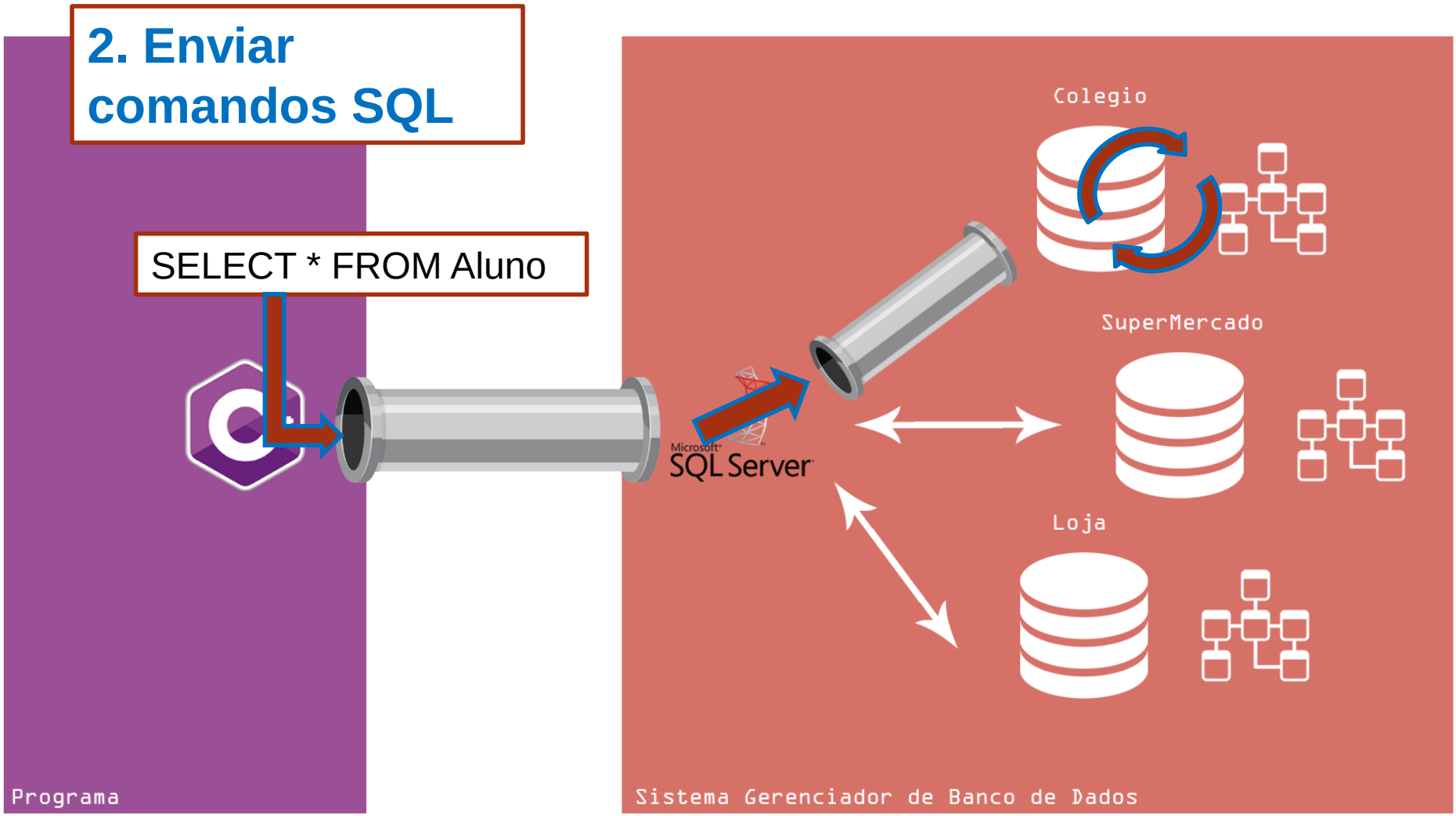
1. Estabelecer conexão



Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

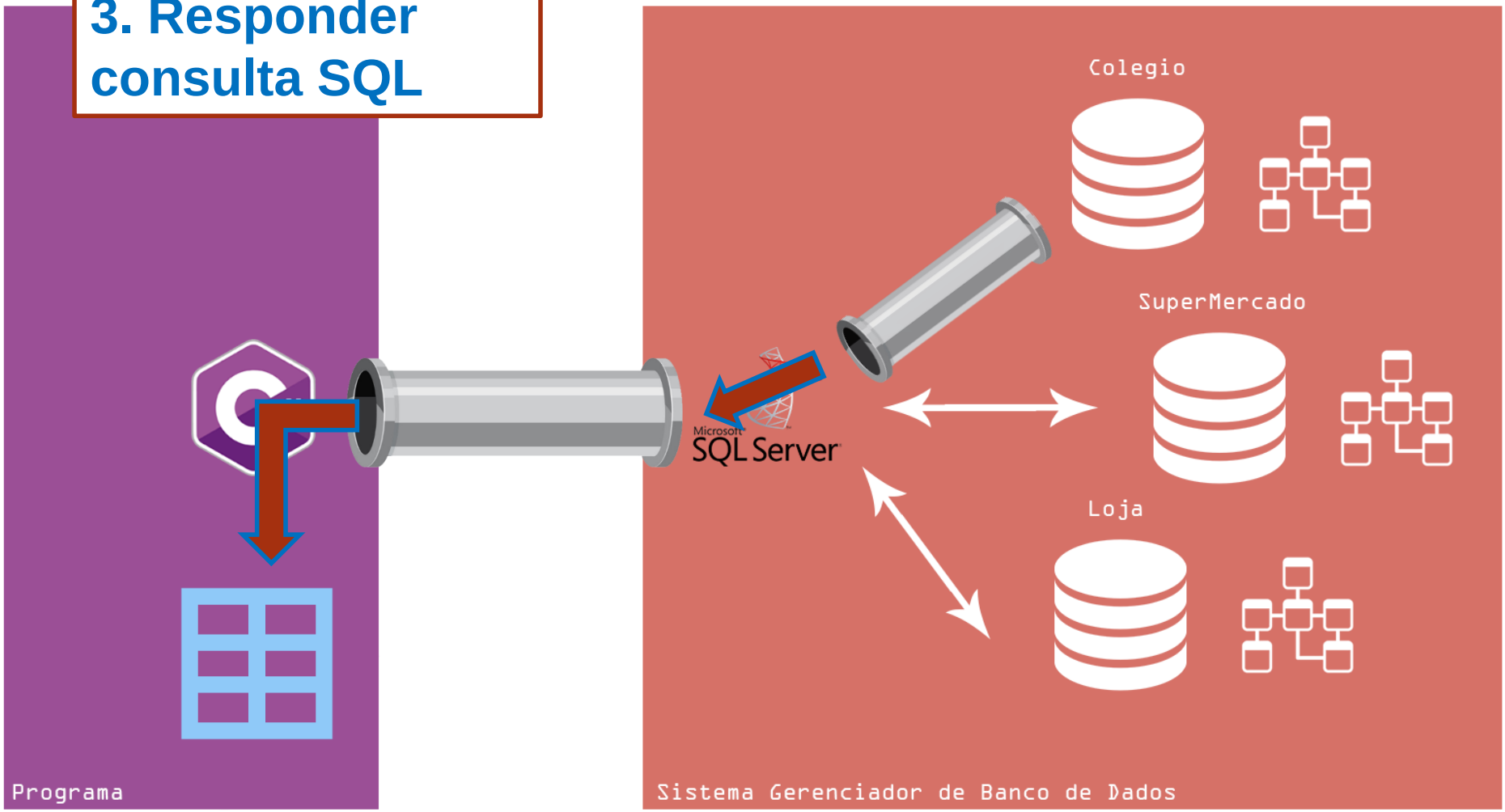


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

3. Responder consulta SQL

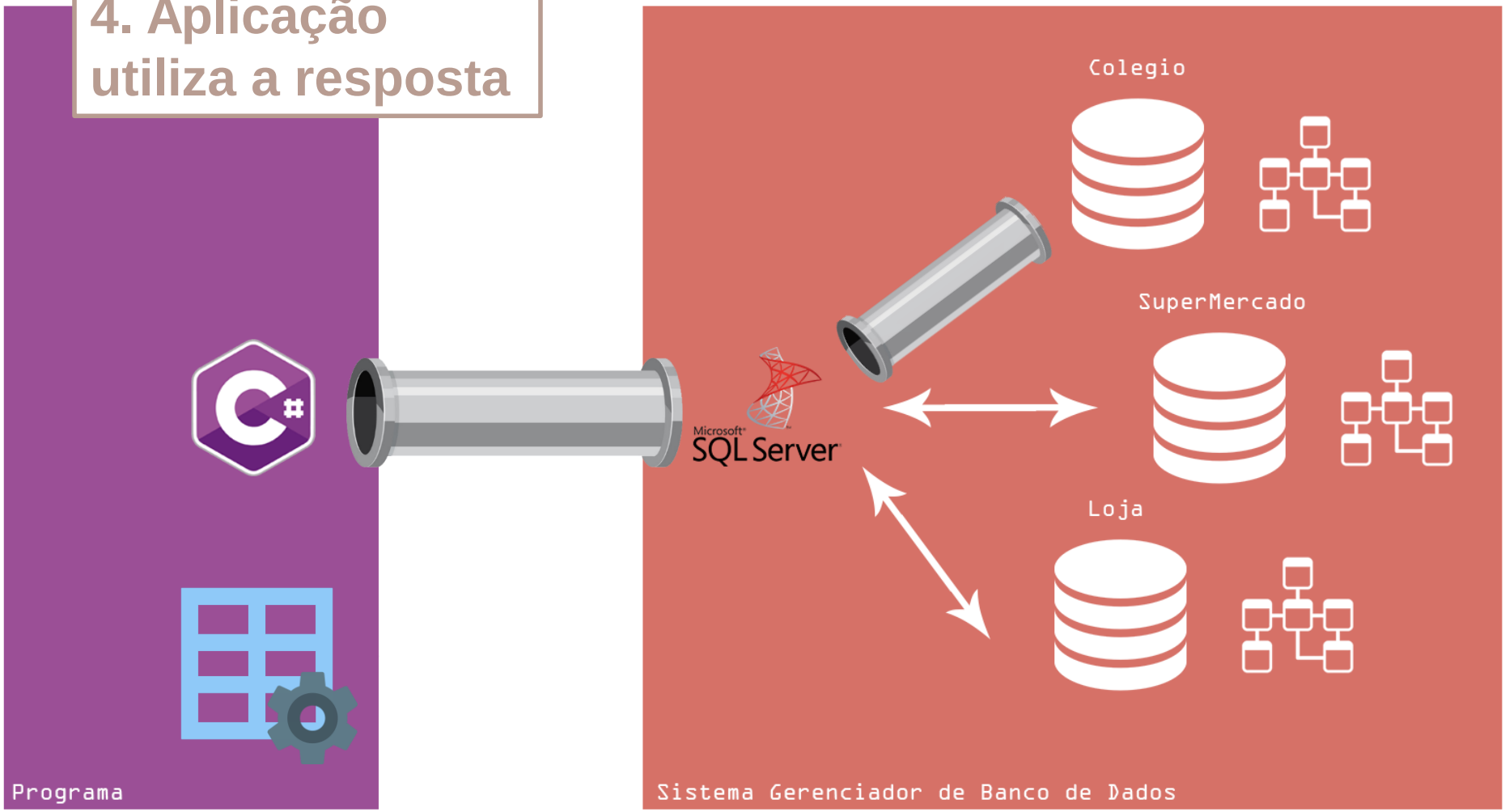


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

4. Aplicação utiliza a resposta

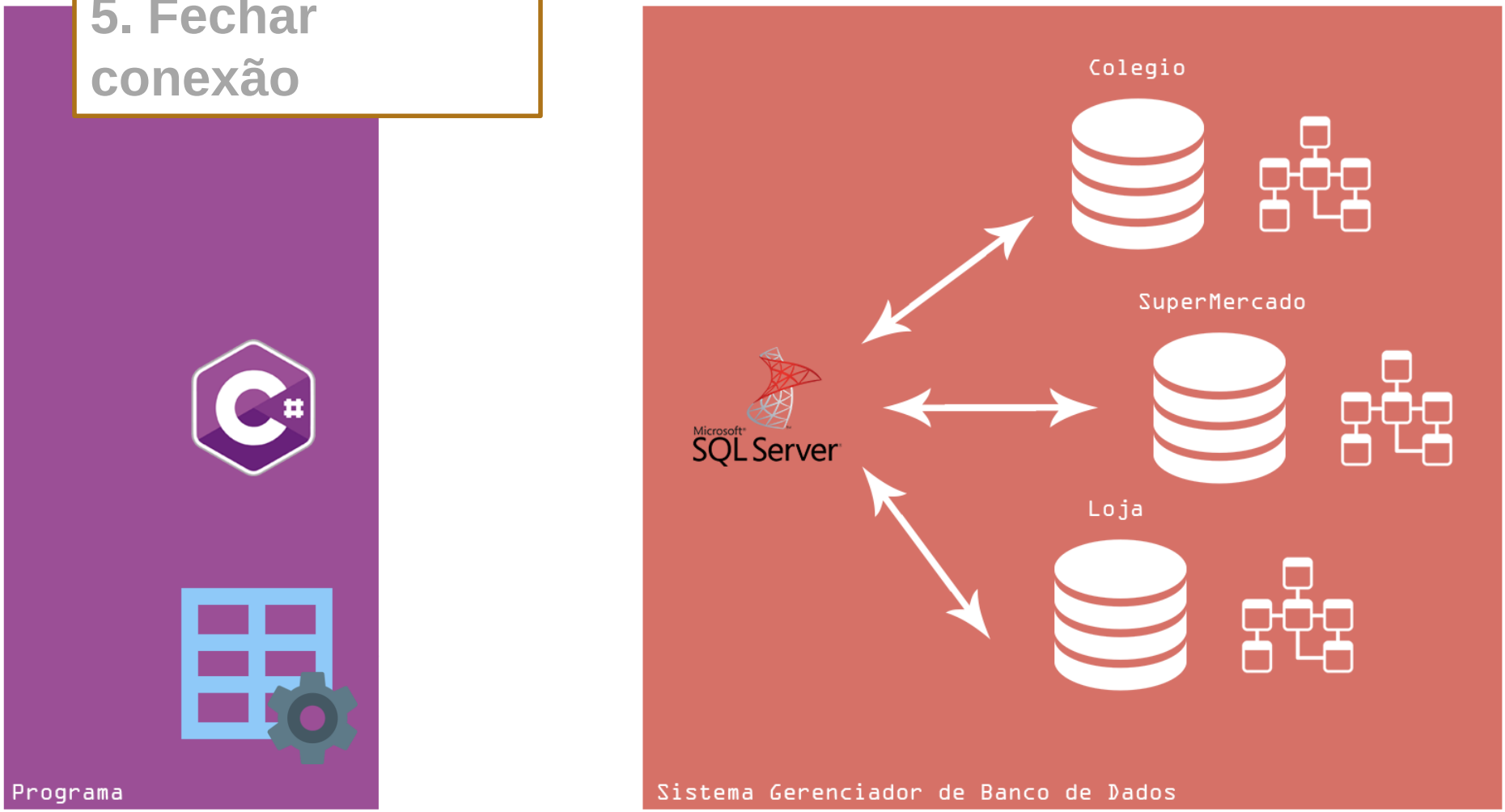


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II

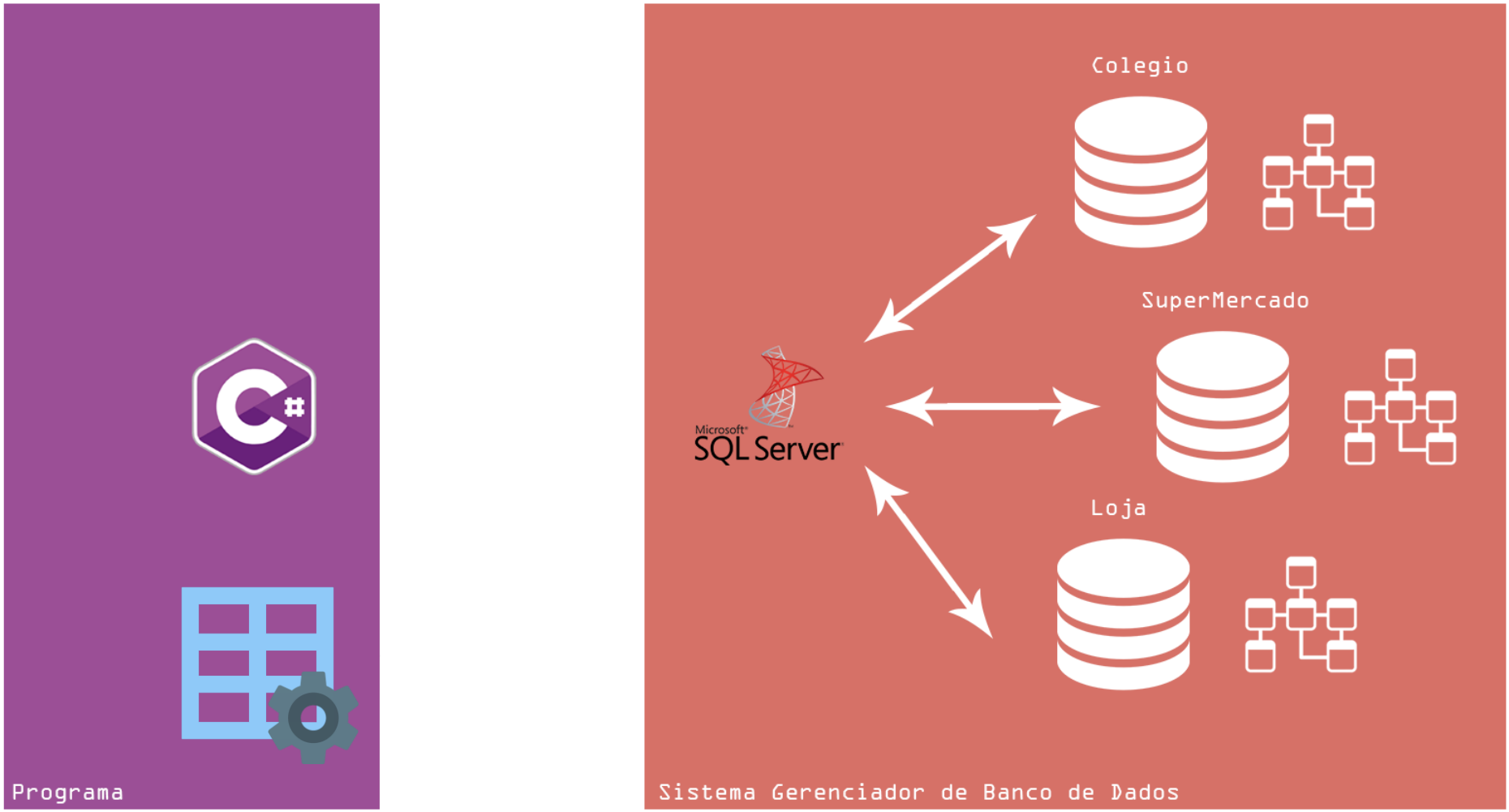
5. Fechar conexão



Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

Colégio Pedro II



Passo a passo

Criando um Sistema de Banco de Dados

Colégio Pedro II

- Para se criar um Sistema de Banco de Dados, iremos:

1. **Preparar o ambiente de banco de dados:**
    1. Criar o banco de dados (arquivo .mdf);
    2. Criar as tabelas do banco de dados;
    3. Popular as tabelas com alguns registros.

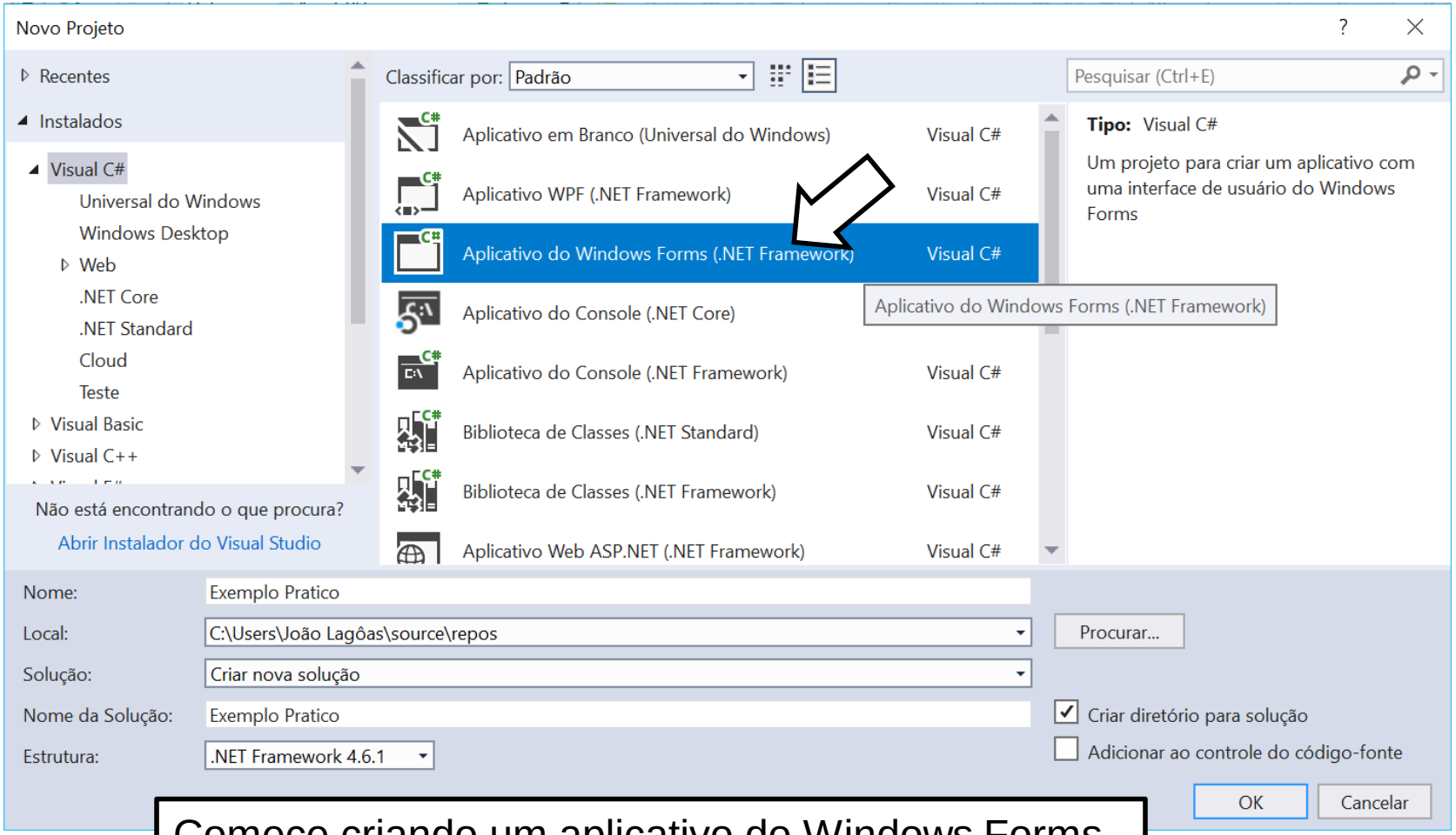
2. **Preparar a nossa aplicação:**
    1. Criar os formulários necessários;
    2. Criar as classes necessárias;
    3. Implementar os métodos em resposta aos eventos;

3. **Comunicar sua aplicação com o banco de dados.**
    1. Adicionar o namespace SqlClient que contém as classes de comunicação;
    2. Instanciar e usar as classes de conexão, comando e leitura de dados.

Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

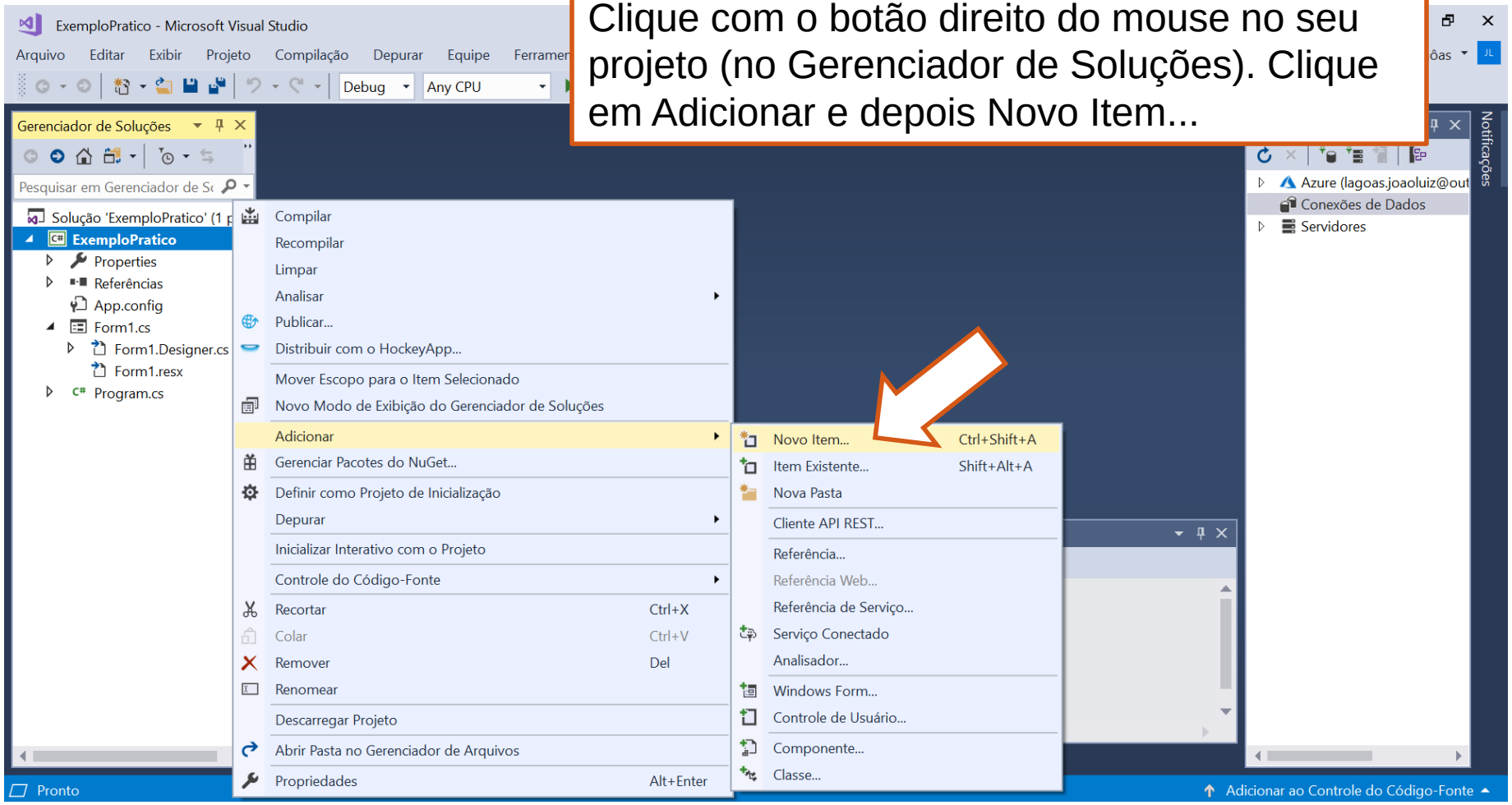


Comece criando um aplicativo do Windows Forms

Passo a passo

Colégio Pedro II

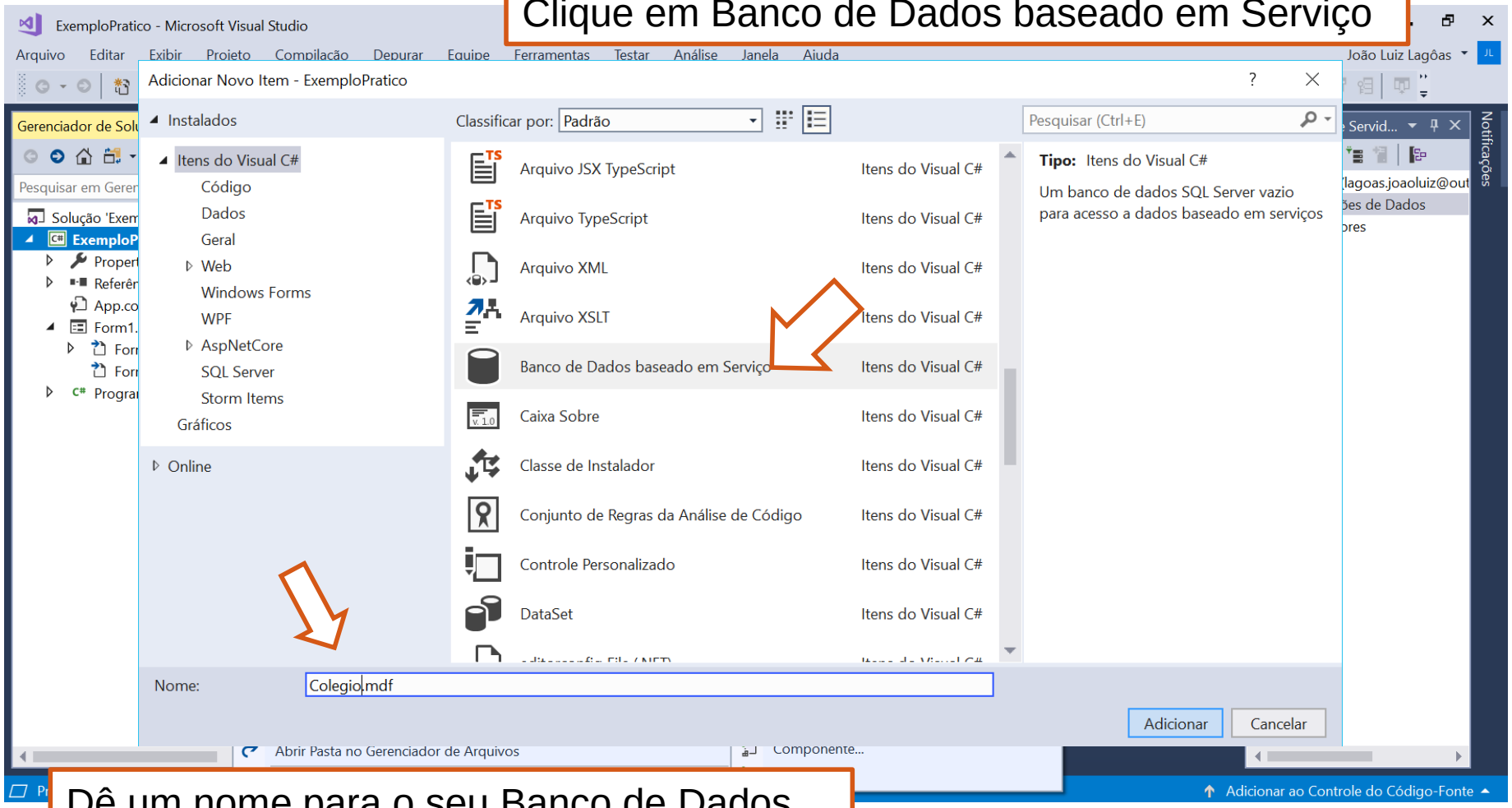
1. Preparar o ambiente de banco de dados:



Passo a passo

Colégio Pedro II

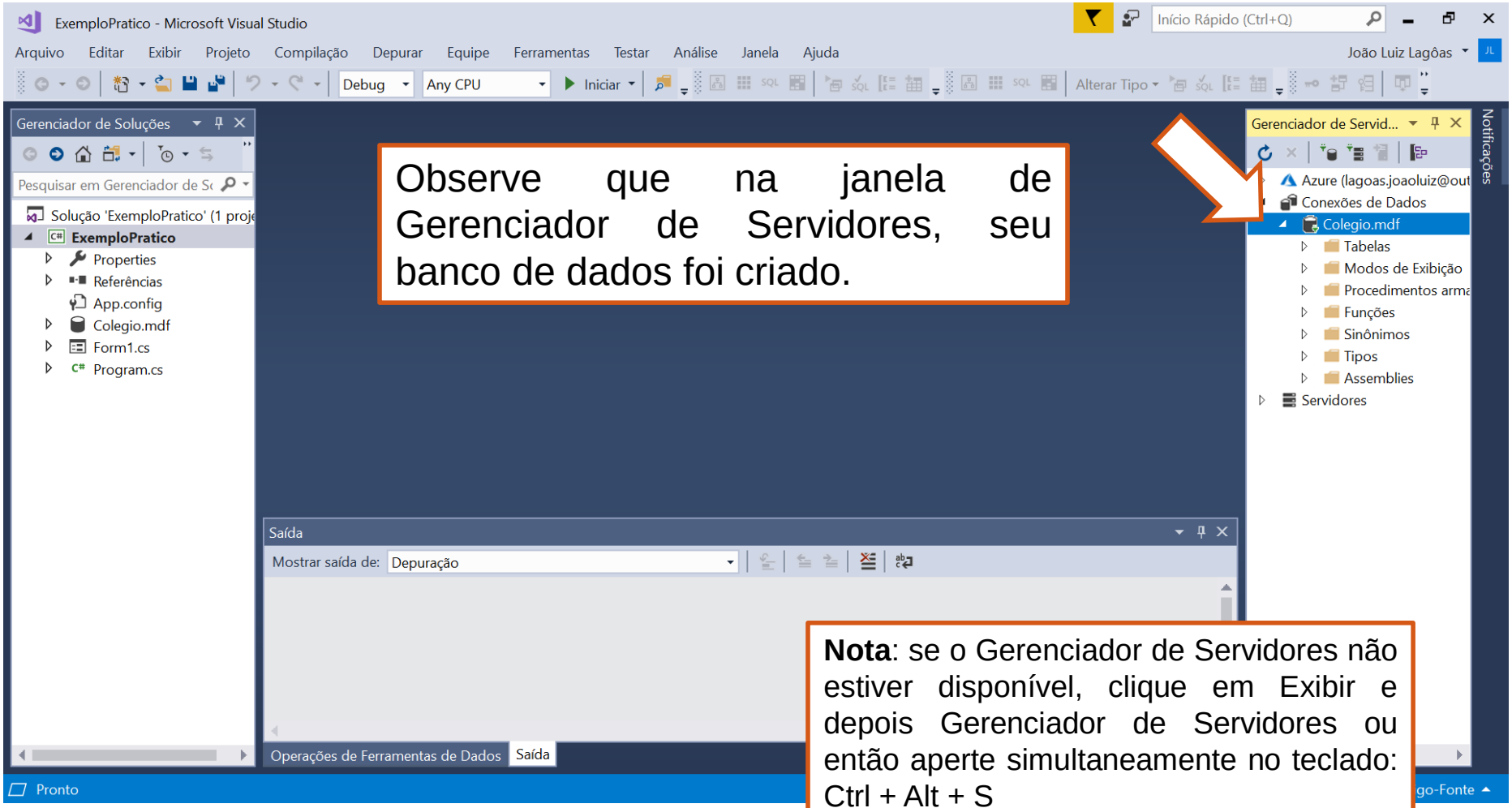
1. Preparar o ambiente de banco de dados:



Passo a passo

Colégio Pedro II

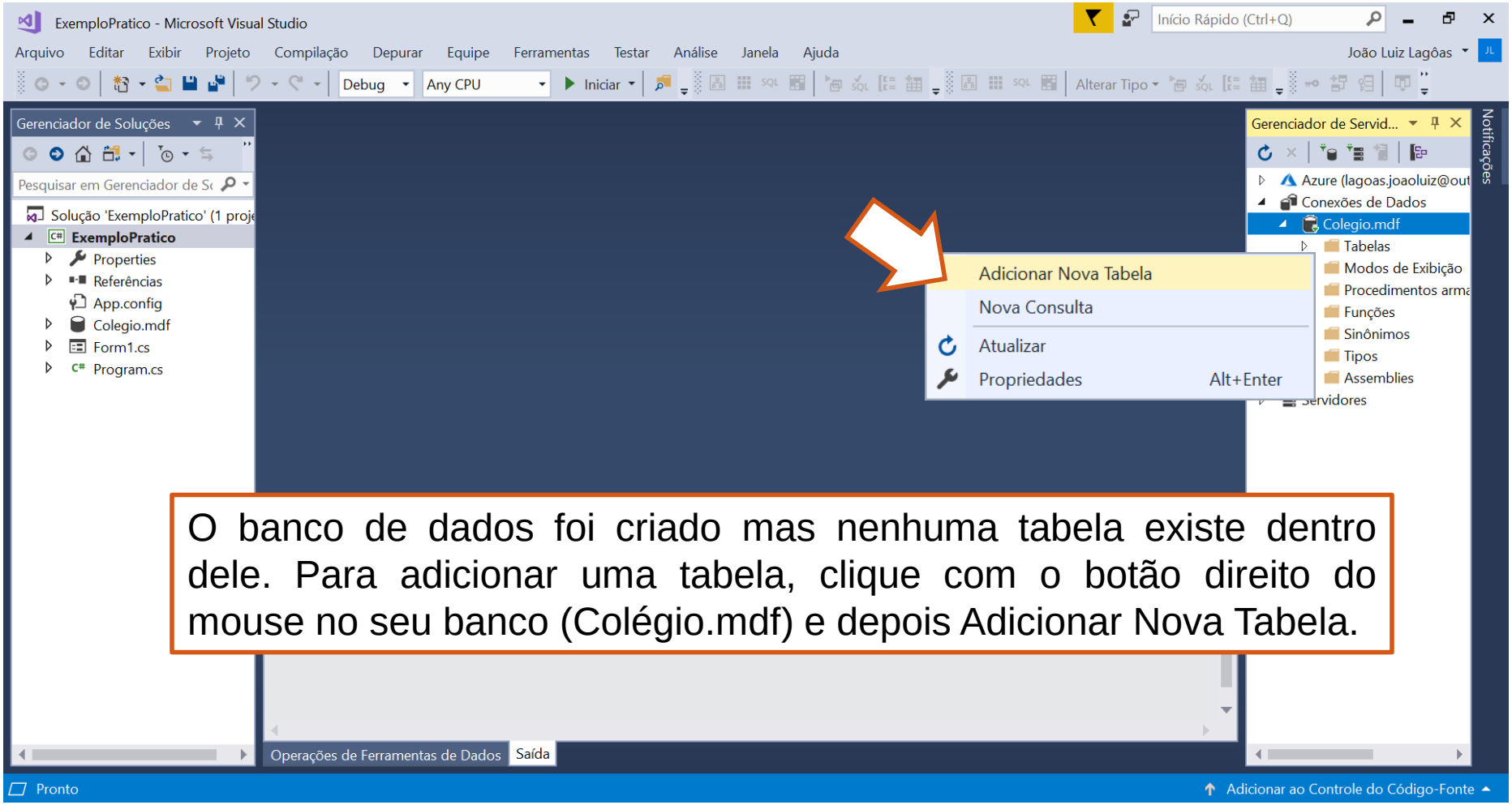
1. Preparar o ambiente de banco de dados:



Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

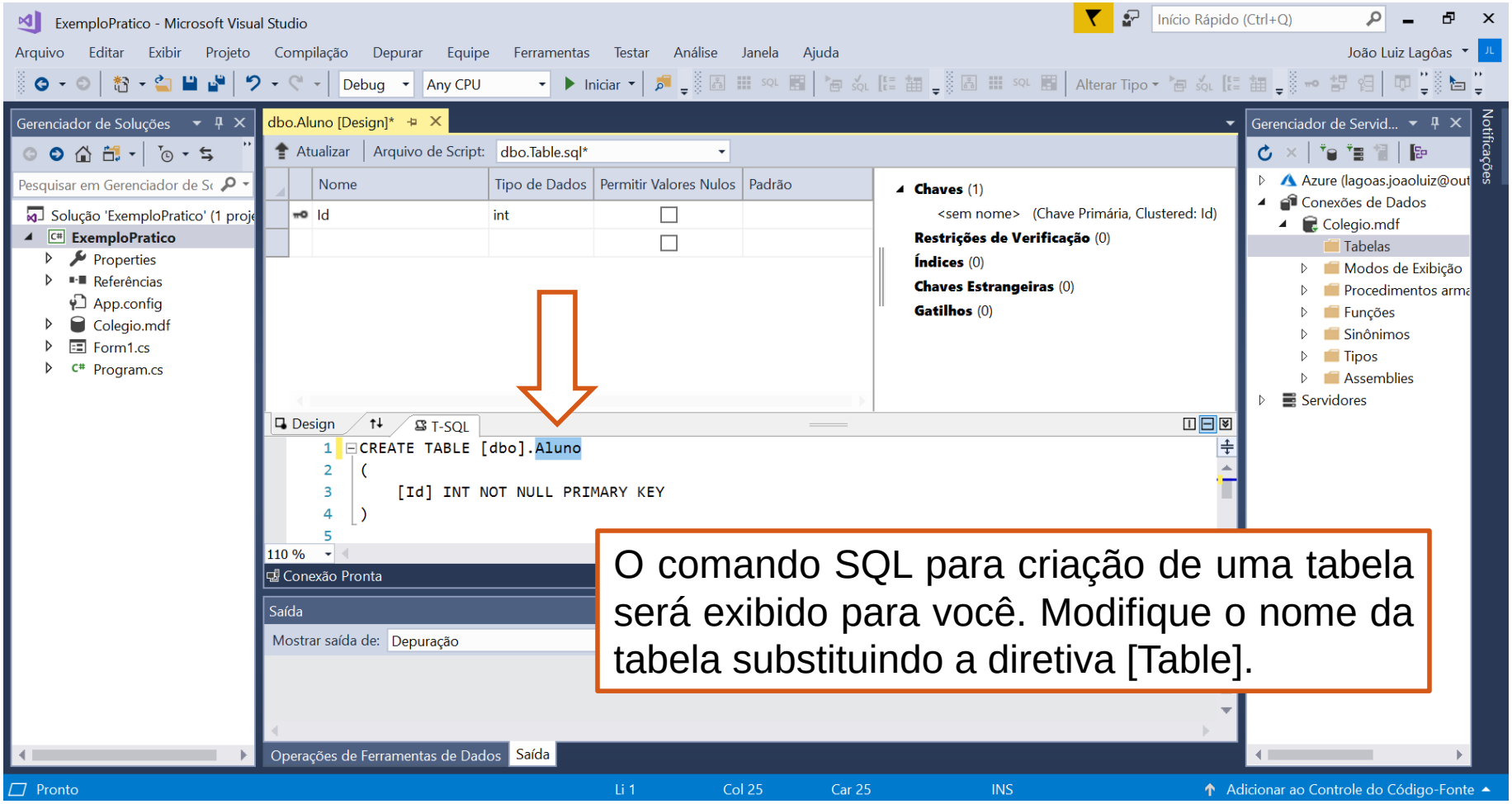


O banco de dados foi criado mas nenhuma tabela existe dentro dele. Para adicionar uma tabela, clique com o botão direito do mouse no seu banco (Colégio.mdf) e depois Adicionar Nova Tabela.

Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:



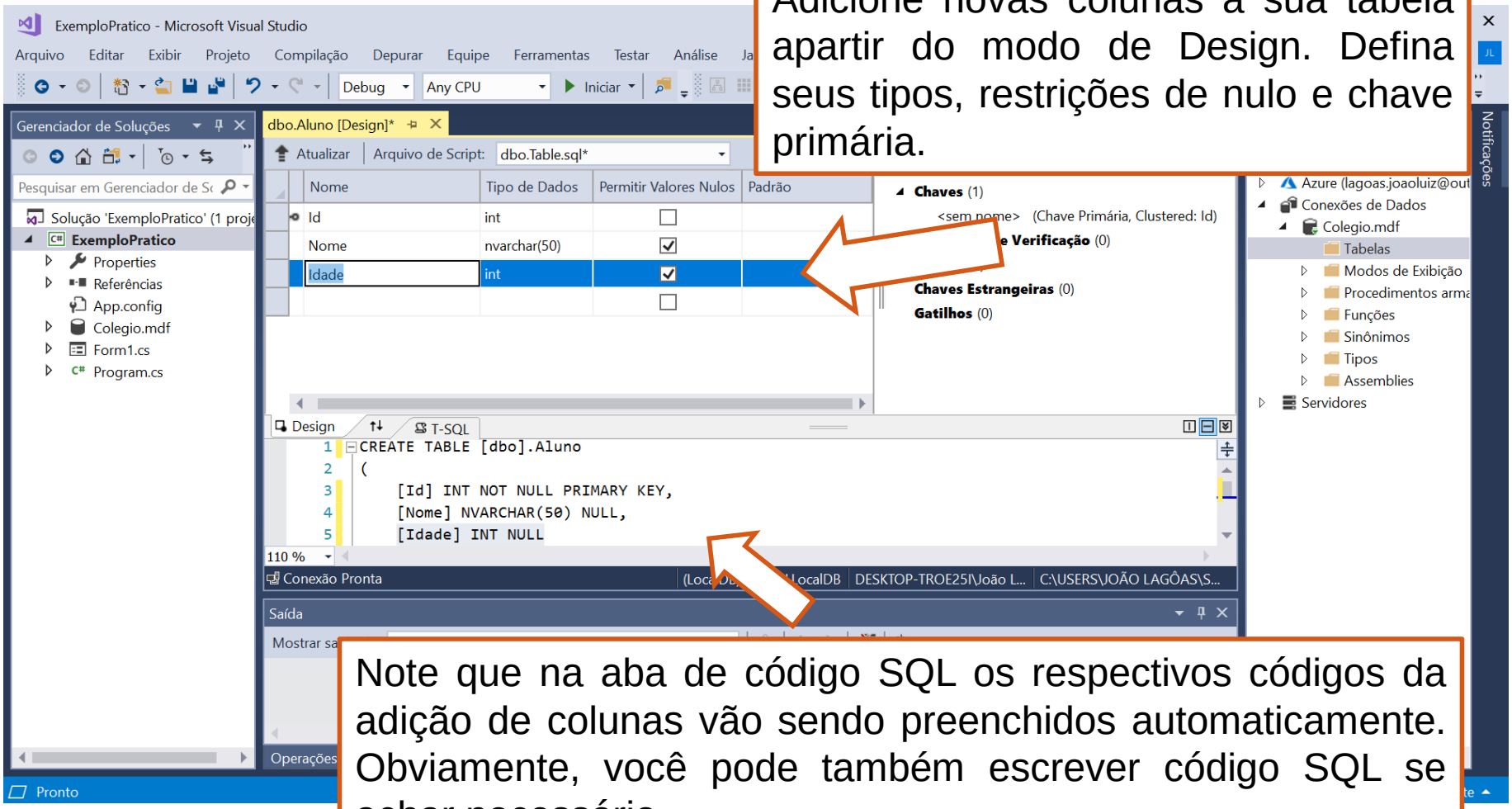
O comando SQL para criação de uma tabela será exibido para você. Modifique o nome da tabela substituindo a diretiva [Table].

Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

Adicione novas colunas à sua tabela apartir do modo de Design. Defina seus tipos, restrições de nulo e chave primária.



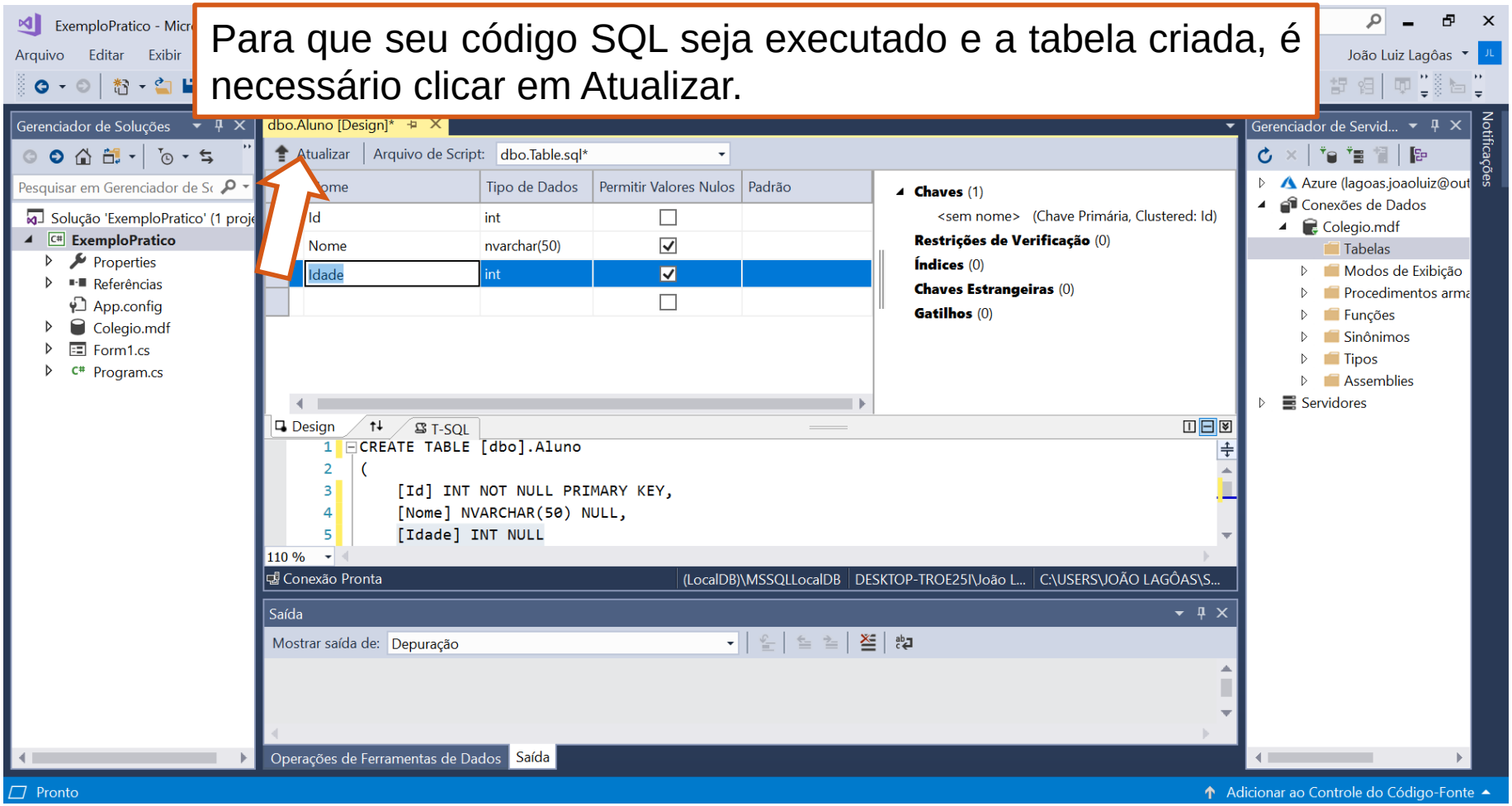
Note que na aba de código SQL os respectivos códigos da adição de colunas vão sendo preenchidos automaticamente. Obviamente, você pode também escrever código SQL se achar necessário.

Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

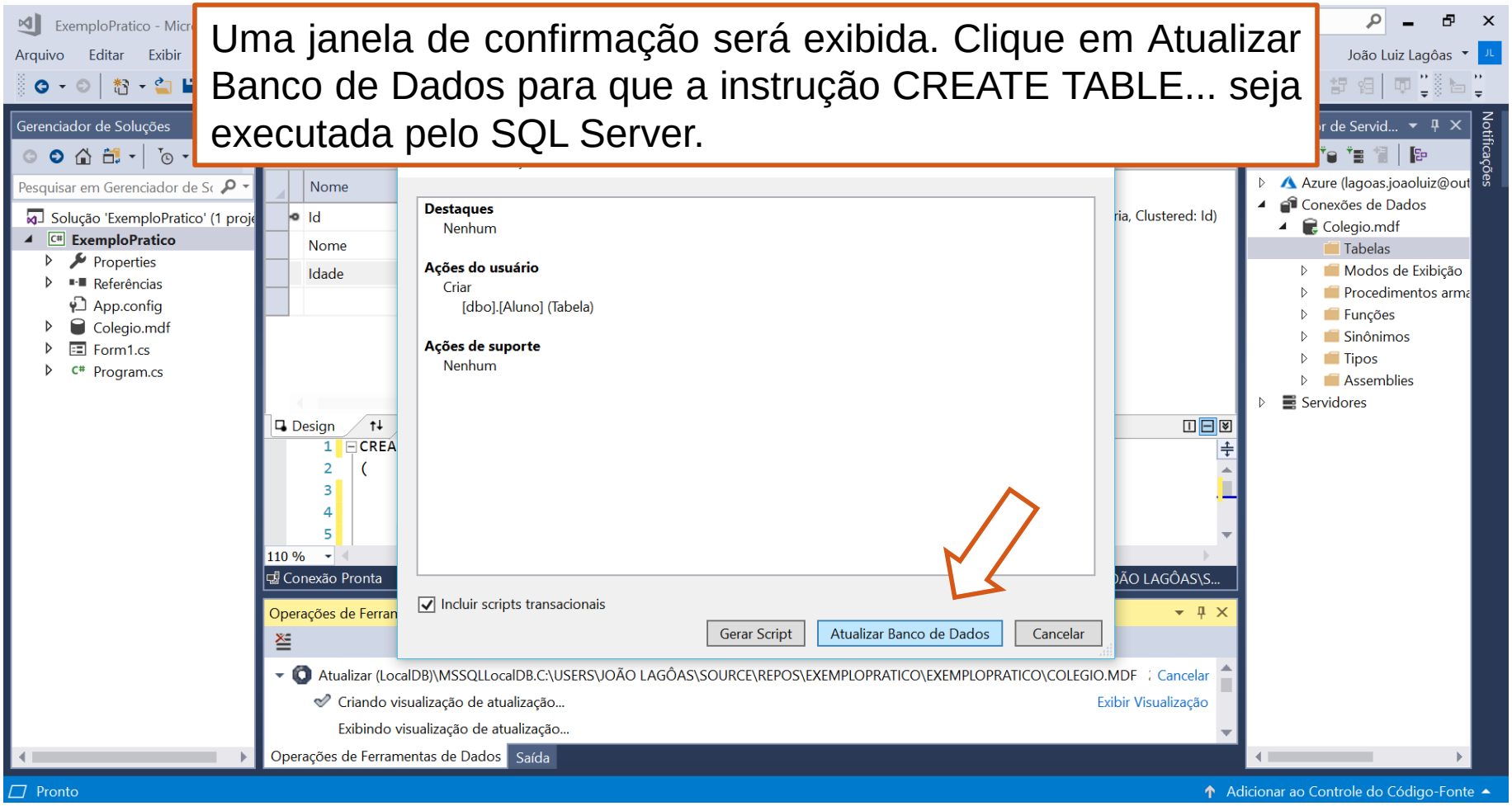
Para que seu código SQL seja executado e a tabela criada, é necessário clicar em Atualizar.



Passo a passo

Colégio Pedro II

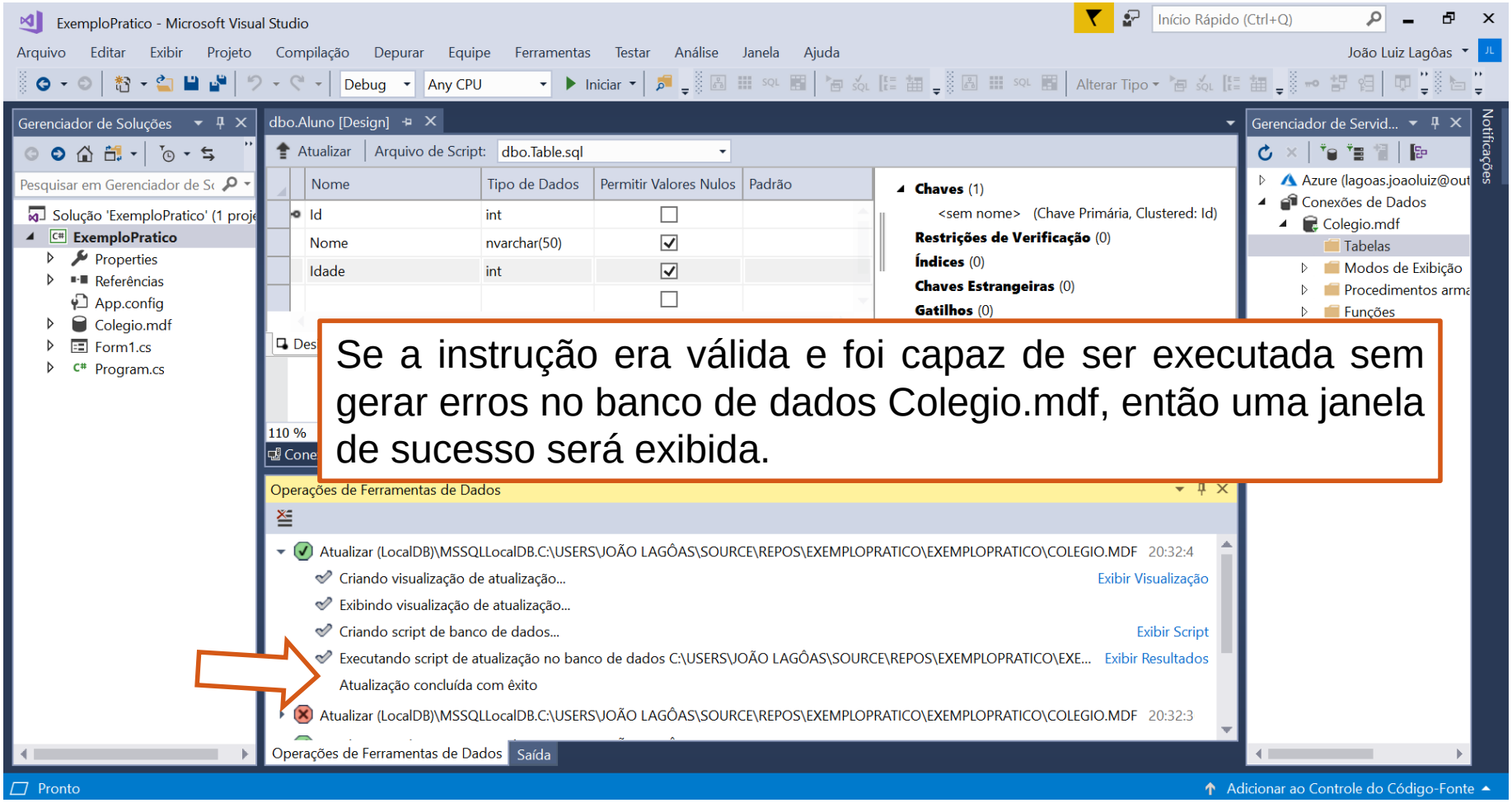
1. Preparar o ambiente de banco de dados:



Passo a passo

Colégio Pedro II

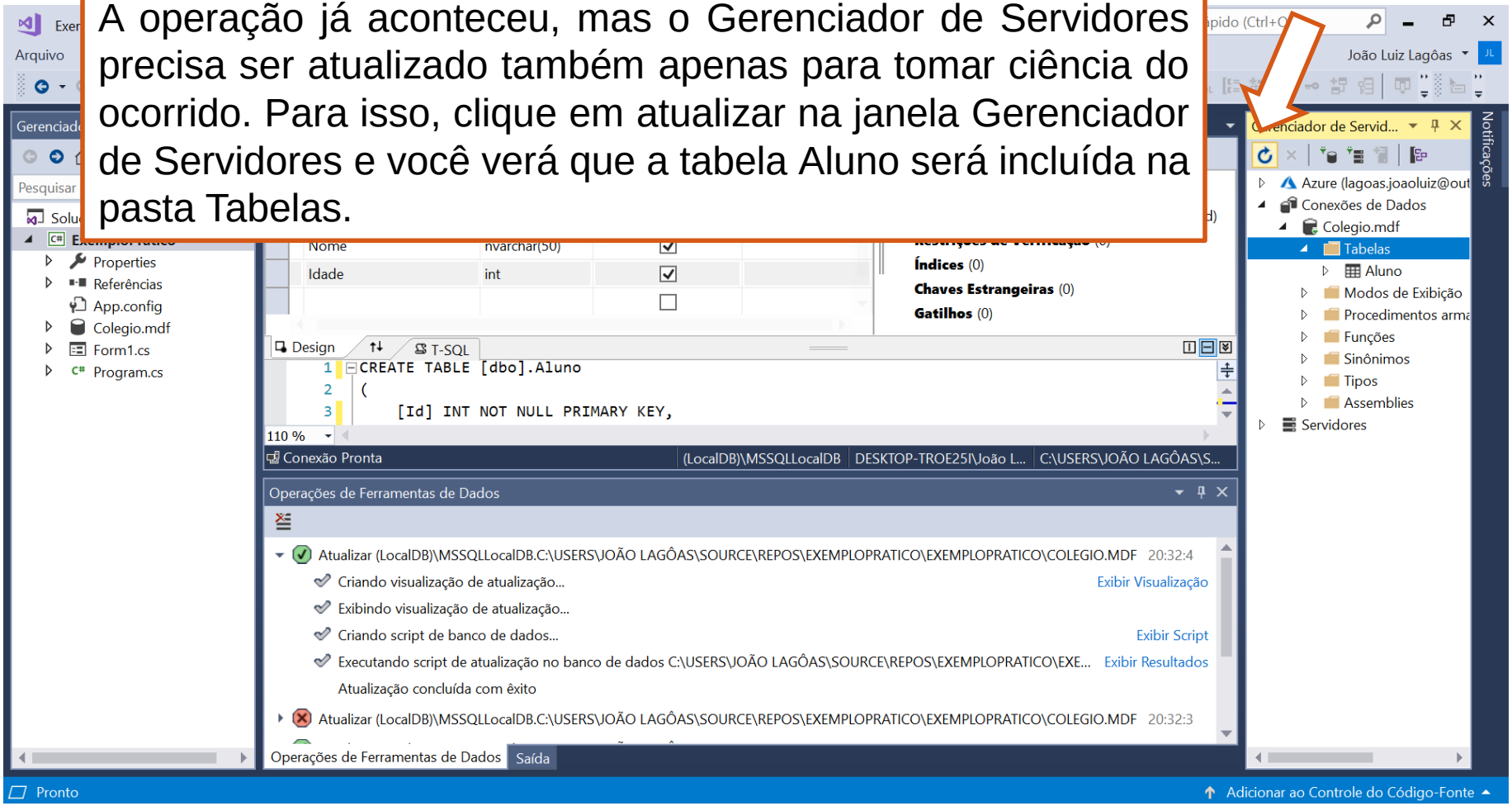
1. Preparar o ambiente de banco de dados:



Passo a passo

1. Preparar o ambiente de banco de dados:

A operação já aconteceu, mas o Gerenciador de Servidores precisa ser atualizado também apenas para tomar ciência do ocorrido. Para isso, clique em atualizar na janela Gerenciador de Servidores e você verá que a tabela Aluno será incluída na pasta Tabelas.

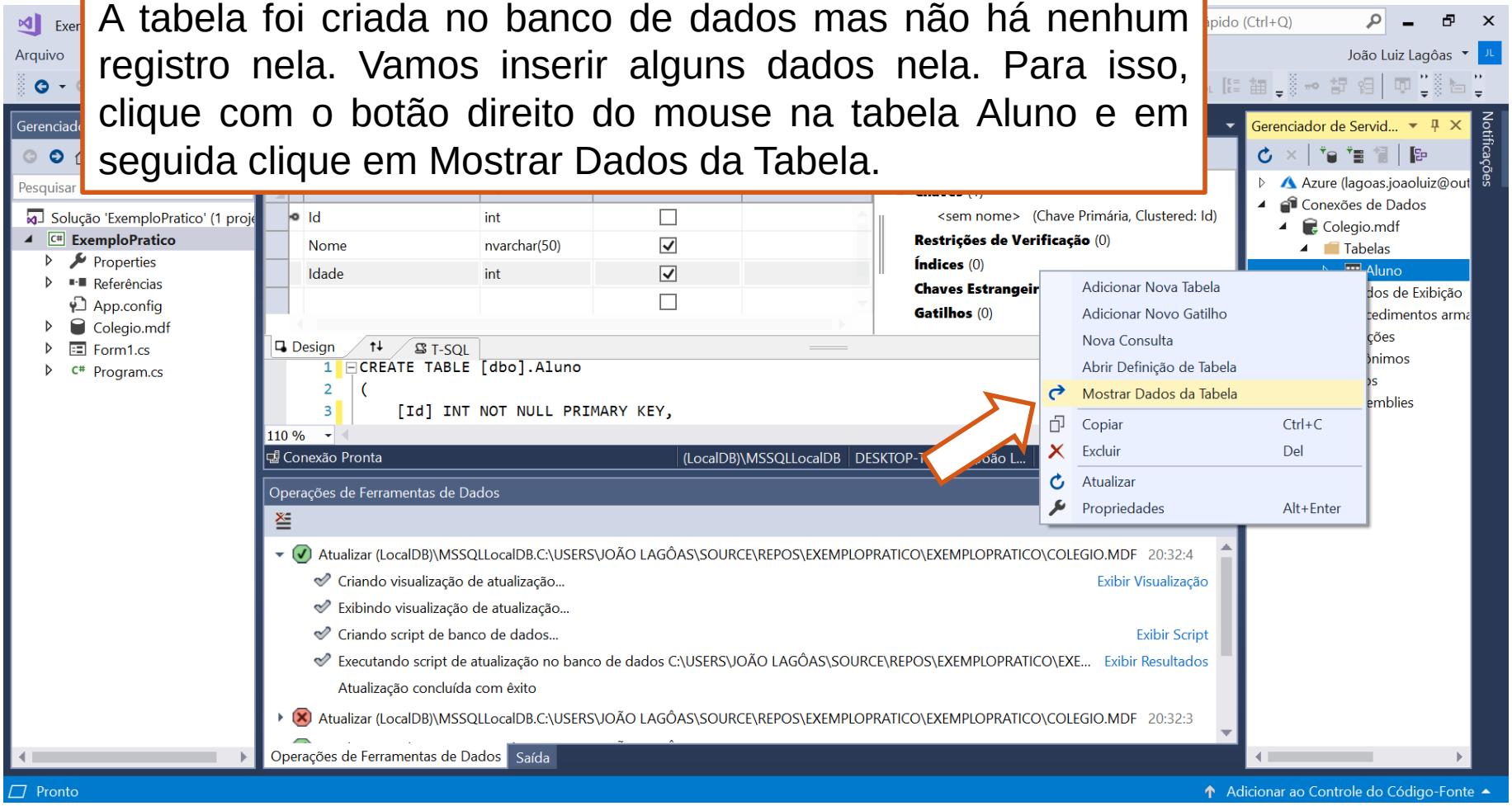


Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

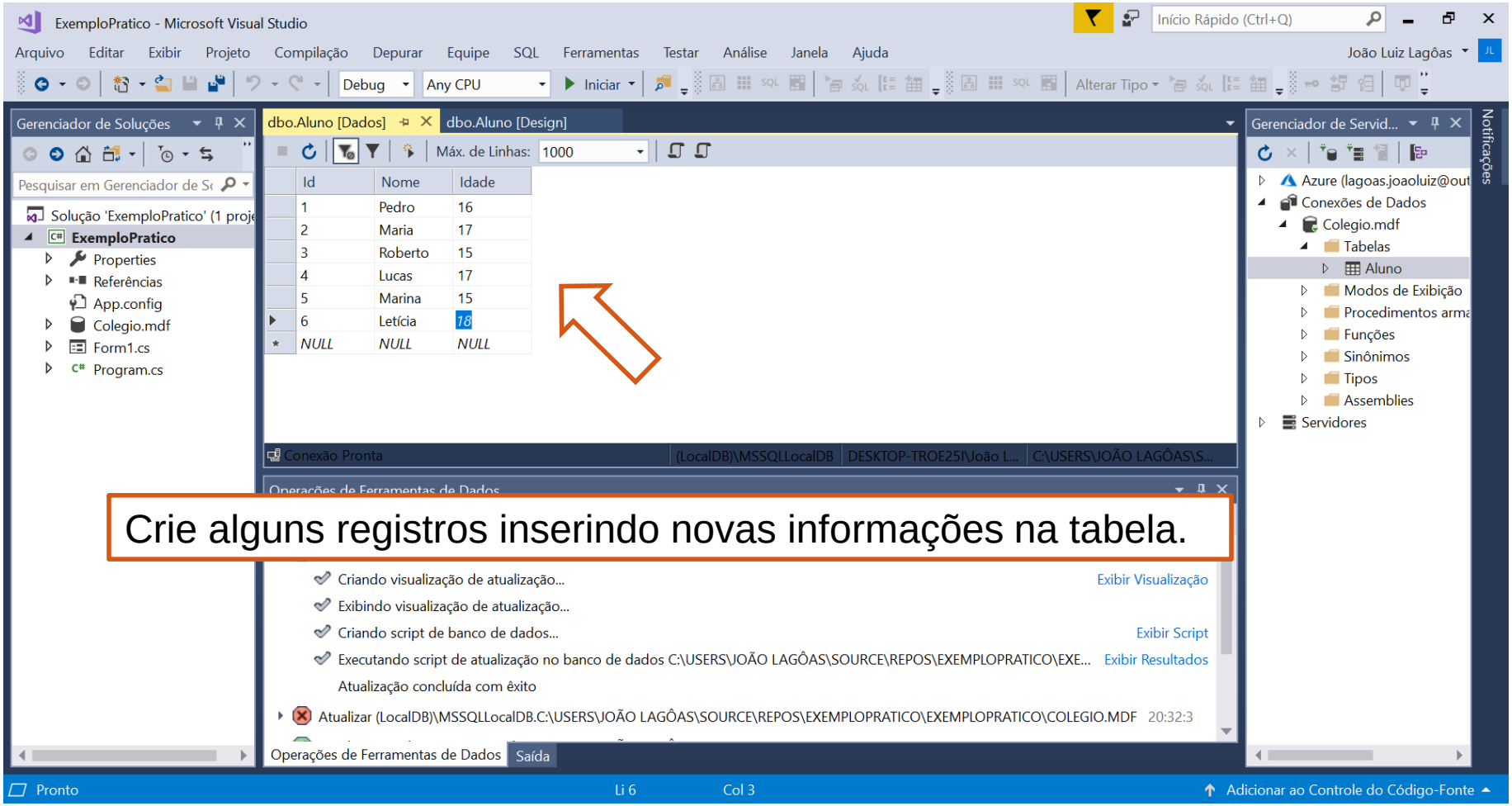
A tabela foi criada no banco de dados mas não há nenhum registro nela. Vamos inserir alguns dados nela. Para isso, clique com o botão direito do mouse na tabela Aluno e em seguida clique em Mostrar Dados da Tabela.



Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:



Passo a passo

Colégio Pedro II

1. Preparar o ambiente de banco de dados:

- A partir desse momento, o seu banco de dados está preparado. Observe o esquema abaixo:

SGBD: SQL Server

Banco de Dados: Colegio.mdf

Tabelas: Aluno

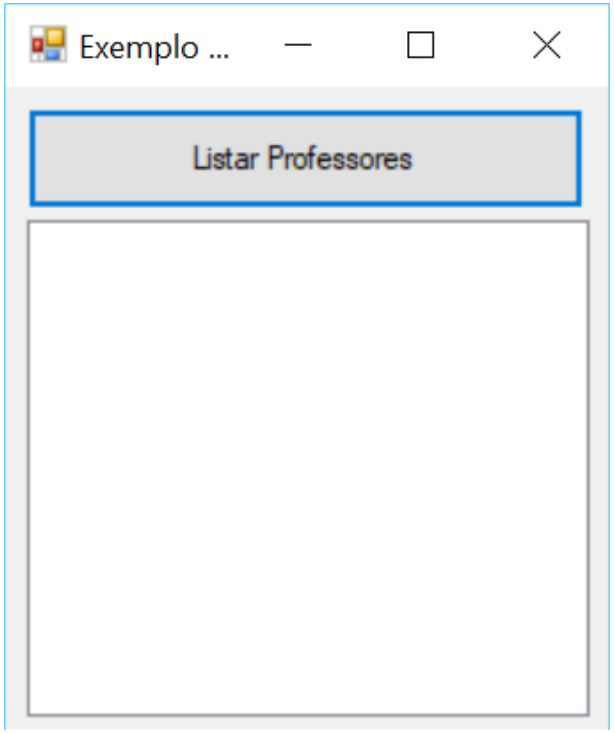
Passo a passo

Colégio Pedro II

2. Preparar a nossa aplicação:

- Essa etapa consiste da elaboração do seu formulário e implementação de classes que você venha a usar.

- Por exemplo, crie o formulário ao lado:

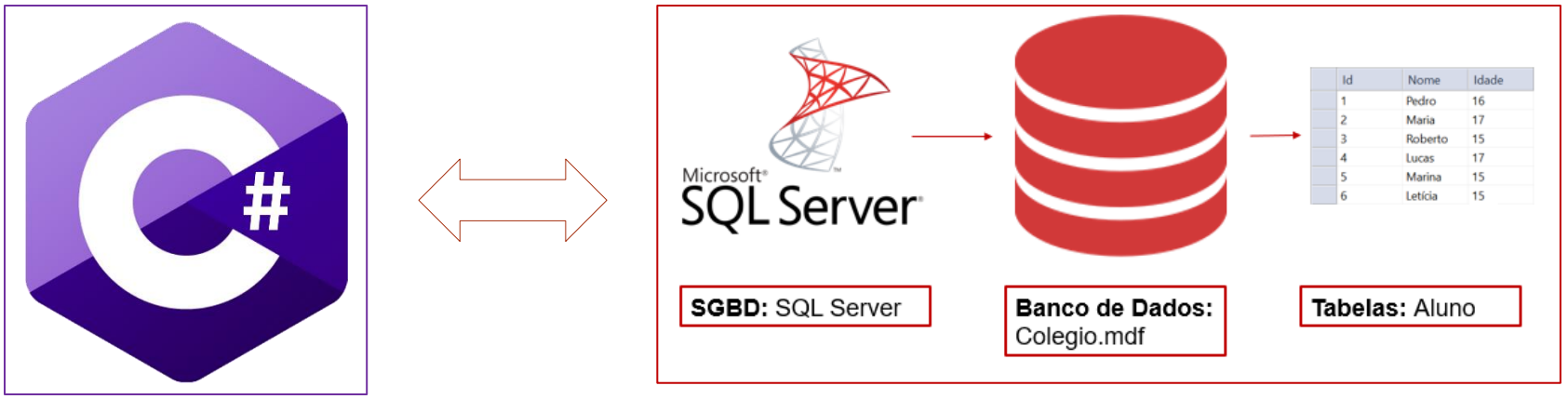


Passo a passo

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- Nosso próximo objetivo é fazer com que nossa aplicação se **comunique** com o SGBD que por sua vez vai gerenciar toda troca de dados entre nossa aplicação e o banco de dados.



Passo a passo

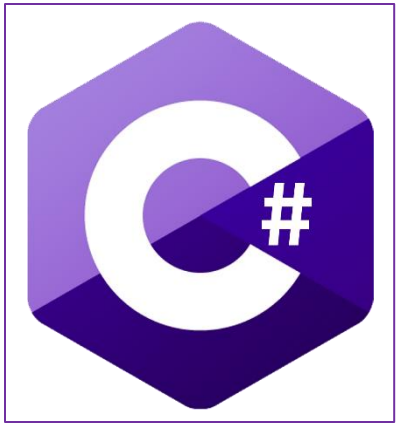
Colégio Pedro II

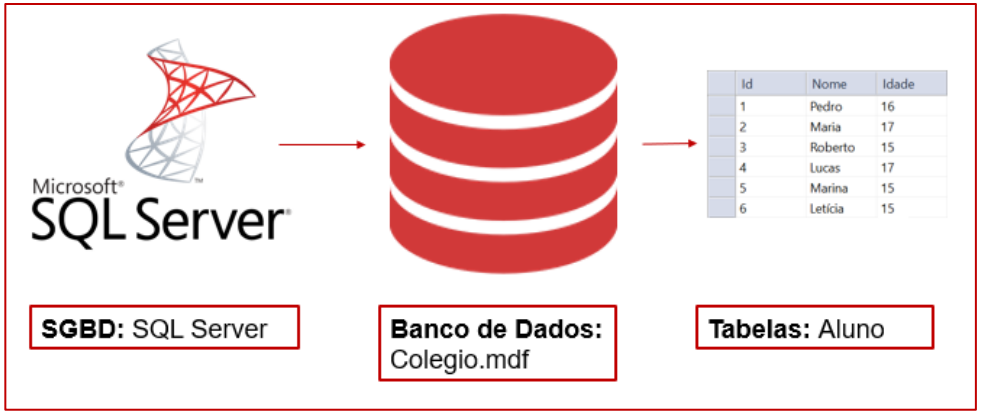
3. Comunicar sua aplicação com o banco de dados.

- Para fazermos essa **comunicação** é necessário:

1. Importar classes C# que são utilizadas para de fato se comunicar e trocar dados com o banco de dados (**namespace SqlClient**).

2. Saber o caminho de conexão dentro do computador para se encontrar o banco de dados (**connectionString**).





Passo a passo

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- Para saber quais são as informações necessárias para se localizar o banco de dados, é comum a gente procurar por uma string denominada connectionString (ou string de conexão).

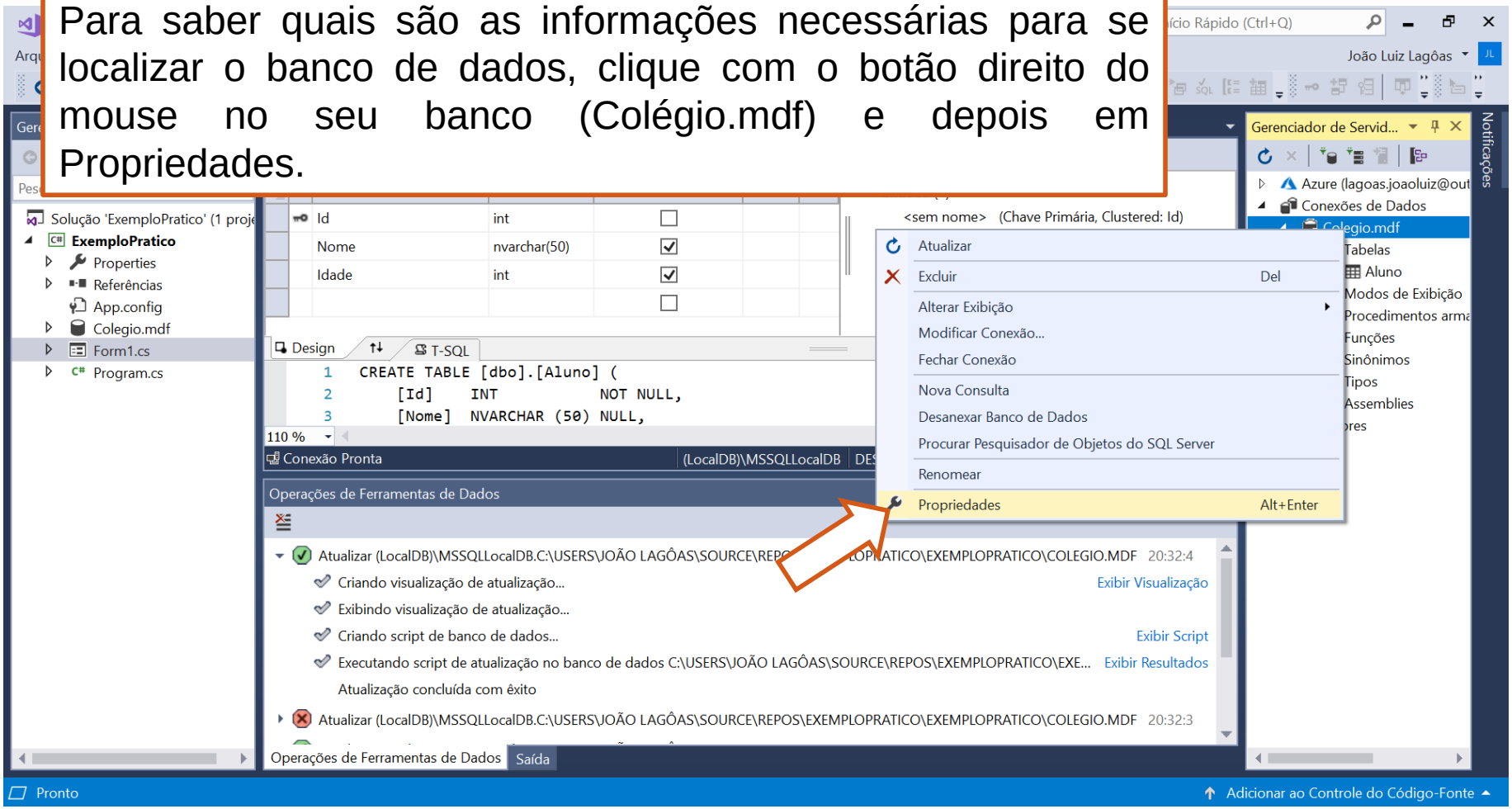
- Essa string, dentre outras coisas, contém as informações necessárias para que a sua aplicação encontre o banco de dados.

Passo a passo

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

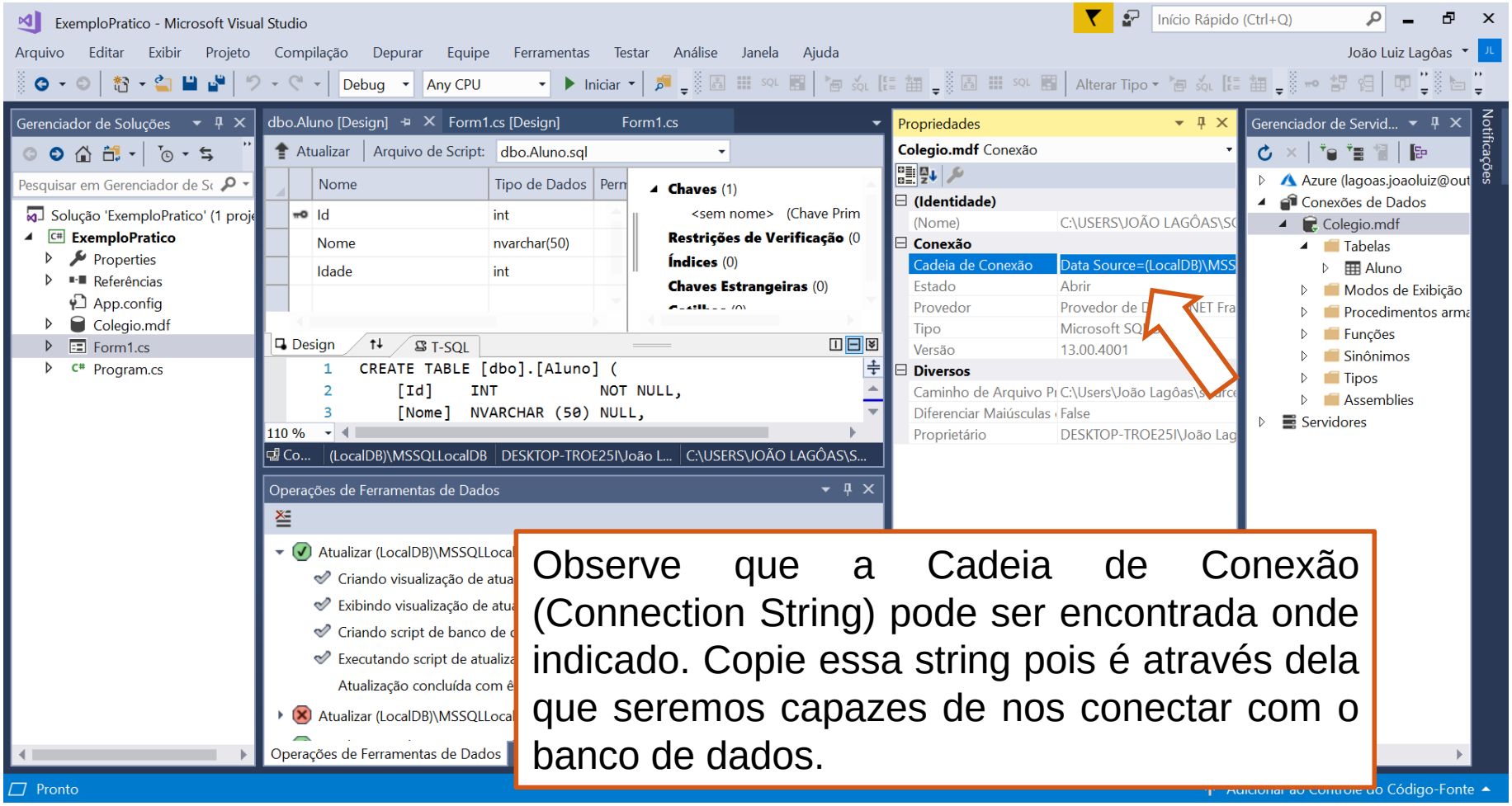
Para saber quais são as informações necessárias para se localizar o banco de dados, clique com o botão direito do mouse no seu banco (Colégio.mdf) e depois em Propriedades.



Passo a passo

3. Comunicar sua aplicação com o banco de dados.

Colégio Pedro II



Observe que a Cadeia de Conexão (Connection String) pode ser encontrada onde indicado. Copie essa string pois é através dela que seremos capazes de nos conectar com o banco de dados.

Passo a passo

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- **SqlConnection**
- **SqlCommand**
- **SqlDataReader**
- **SqlAdapter**

- Dentro do namespace System.Data.SqlClient, há uma variedade de classes que são utilizadas para se comunicar com um Banco de Dados SQL Server

```
using System.Data.SqlClient;
```

Não deixe de adicionar esse linha de instrução caso o seu programa faça comunicação com um banco de dados.

SqlConnection: Realizando a conexão

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- A classe SqlConnection é responsável por realizar a conexão entre a aplicação C# e o Banco de Dados SQL Server.

- No seu construtor, ela espera receber uma string de conexão que contém todas as informações necessárias para encontrar o banco de dados (o arquivo .mdf). O valor de retorno é um objeto que representa a conexão.

- Um objeto SqlConnection apresenta dois métodos muito importantes: Open() e Close(). O primeiro de fato cria o “túnel” entre a aplicação e o banco de dados enquanto que o segundo o fecha.

SqlConnection: Realizando a conexão

Colégio Pedro II

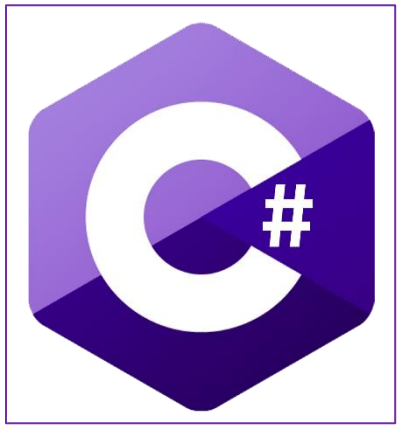
3. Comunicar sua aplicação com o banco de dados.

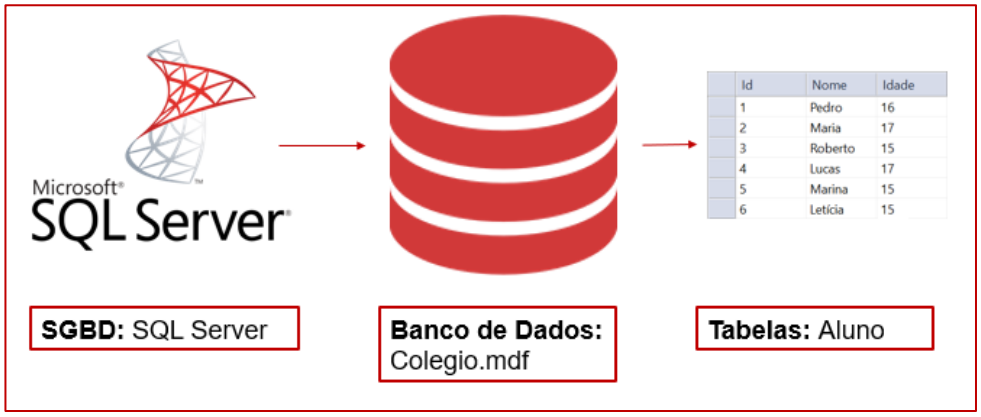
Exemplo

```
string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrated
Security=True";

SqlConnection conexao = new SqlConnection(stringDeConexao);

conexao.Open();
```





SqlConnection: Realizando a conexão

Colégio Pedro II

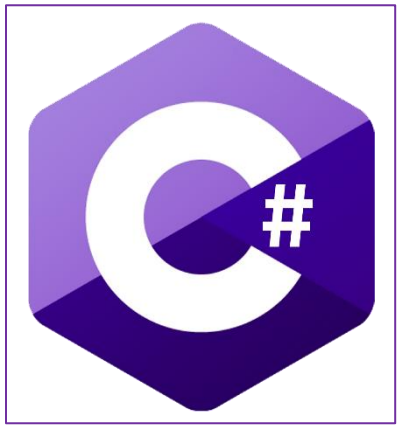
3. Comunicar sua aplicação com o banco de dados.

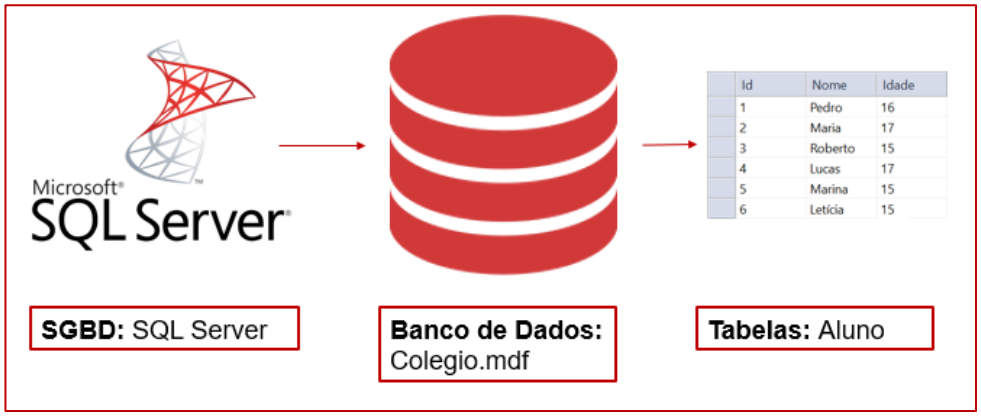
Exemplo

```
string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrated
Security=True";

SqlConnection conexao = new SqlConnection(stringDeConexao);

conexao.Open();
```





SqlConnection: Realizando a conexão

Colégio Pedro II

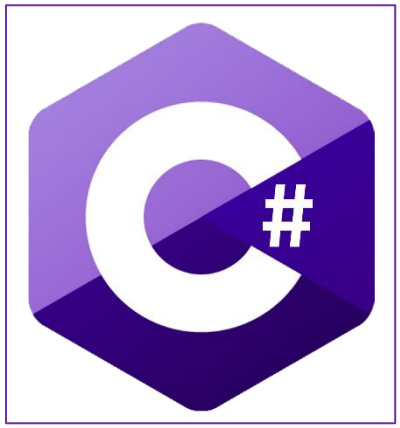
3. Comunicar sua aplicação com o banco de dados.

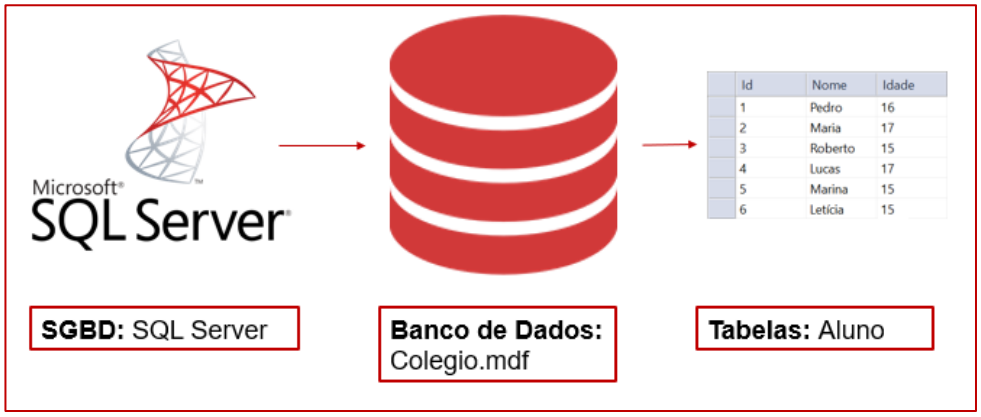
Exemplo

```
string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrated
Security=True";

SqlConnection conexao = new SqlConnection(stringDeConexao);

conexao.Open();
```





SqlConnection: Realizando a conexão

Colégio Pedro II

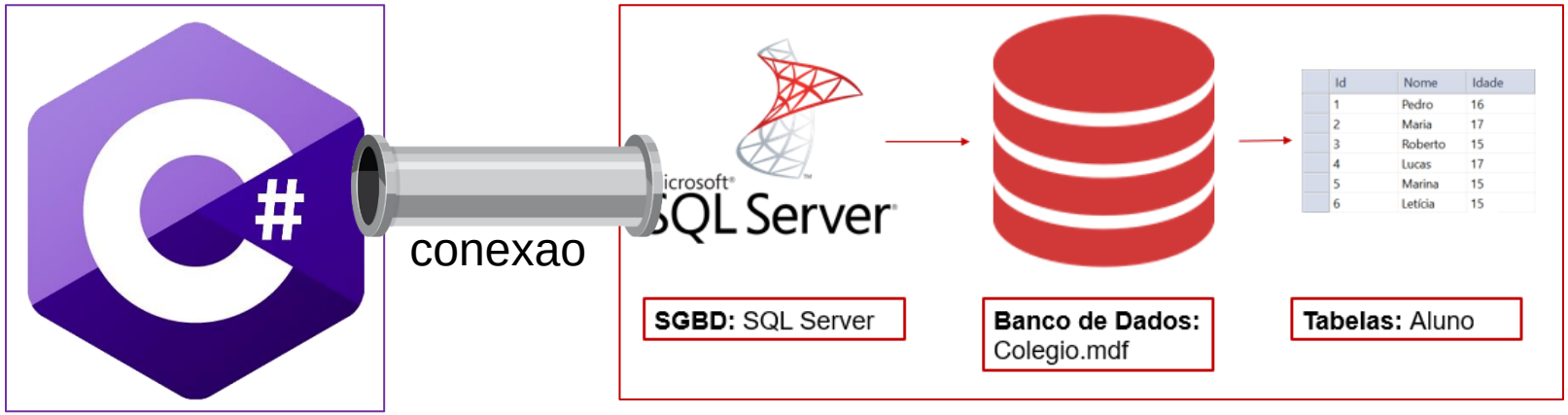
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrated
Security=True";

SqlConnection conexao = new SqlConnection(stringDeConexao);

conexao.Open();
```



SqlCommand: Enviando um comando

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- A classe SqlCommand é responsável por codificar uma instrução SQL e enviá-la, através de um objeto SqlConnection, para o banco de dados.

- O construtor de SqlCommand espera receber dois parâmetros: uma string contendo uma instrução SQL e um objeto SqlConnection para que a aplicação saiba para onde ele deve enviar a instrução SQL.

- Um objeto SqlCommand tem um método chamado ExecuteReader(). Esse método que de fato envia a instrução SQL pela conexão e recebe o retorno do Banco de Dados. Esse retorno vem como um objeto da classe SqlDataReader, que nada mais é que uma abstração de uma resposta SQL (uma tabela).

SqlCommand: Enviando um comando

Colégio Pedro II

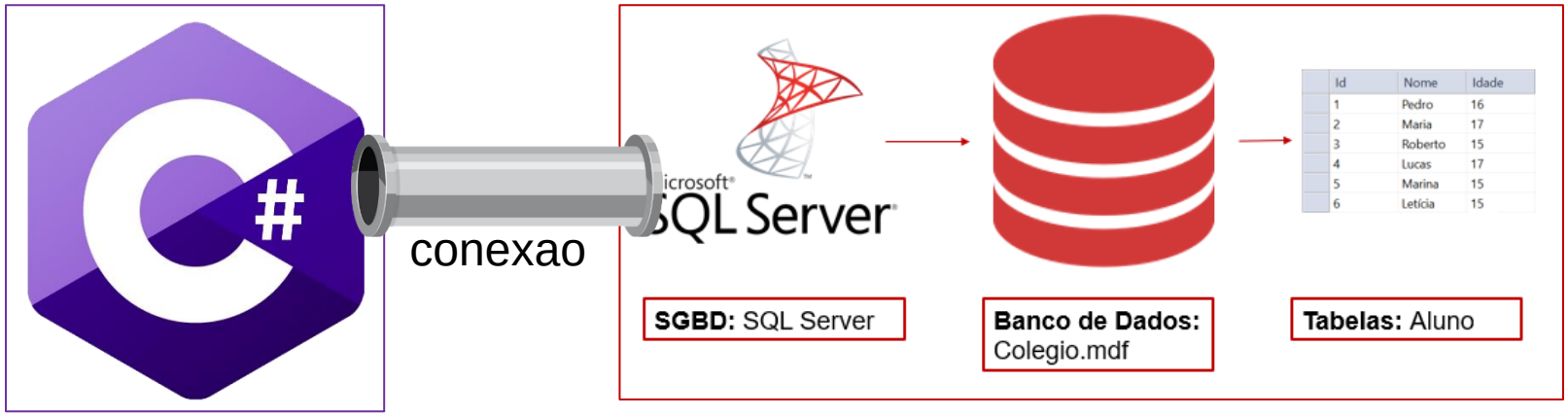
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlCommand: Enviando um comando

Colégio Pedro II

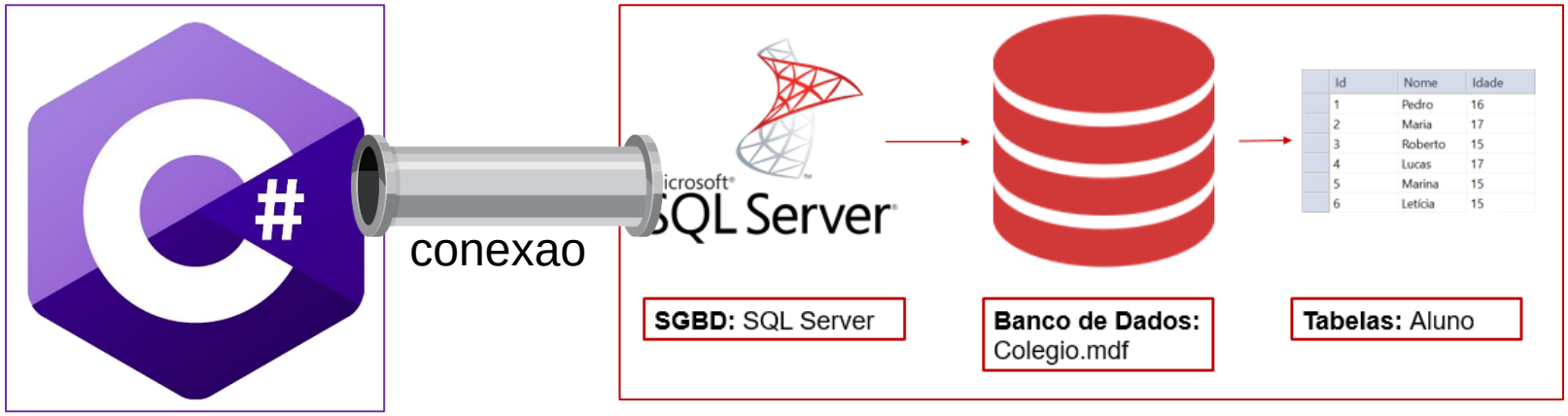
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlCommand: Enviando um comando

Colégio Pedro II

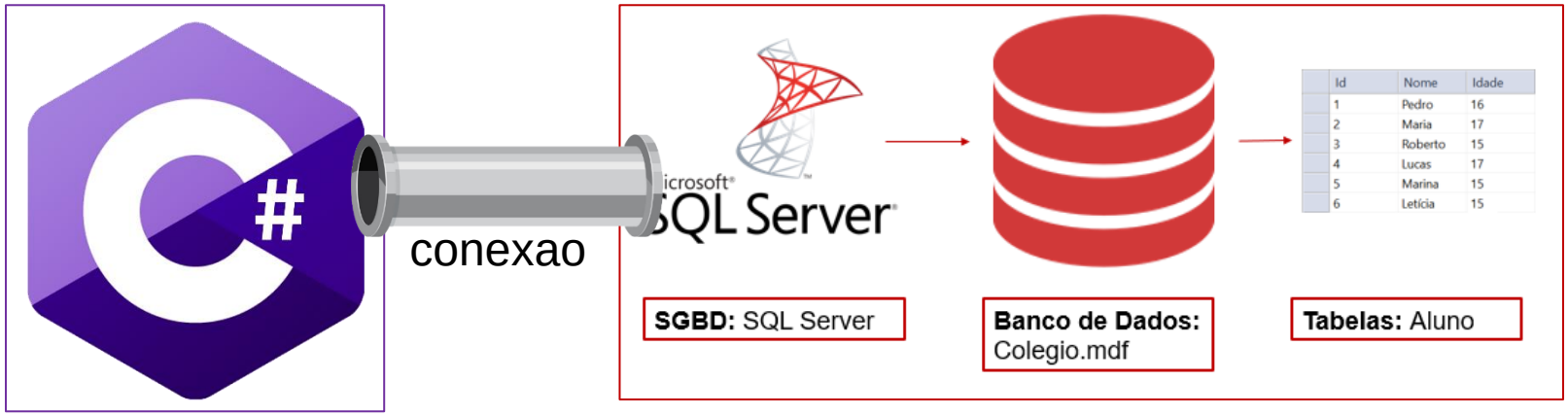
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlCommand: Enviando um comando

Colégio Pedro II

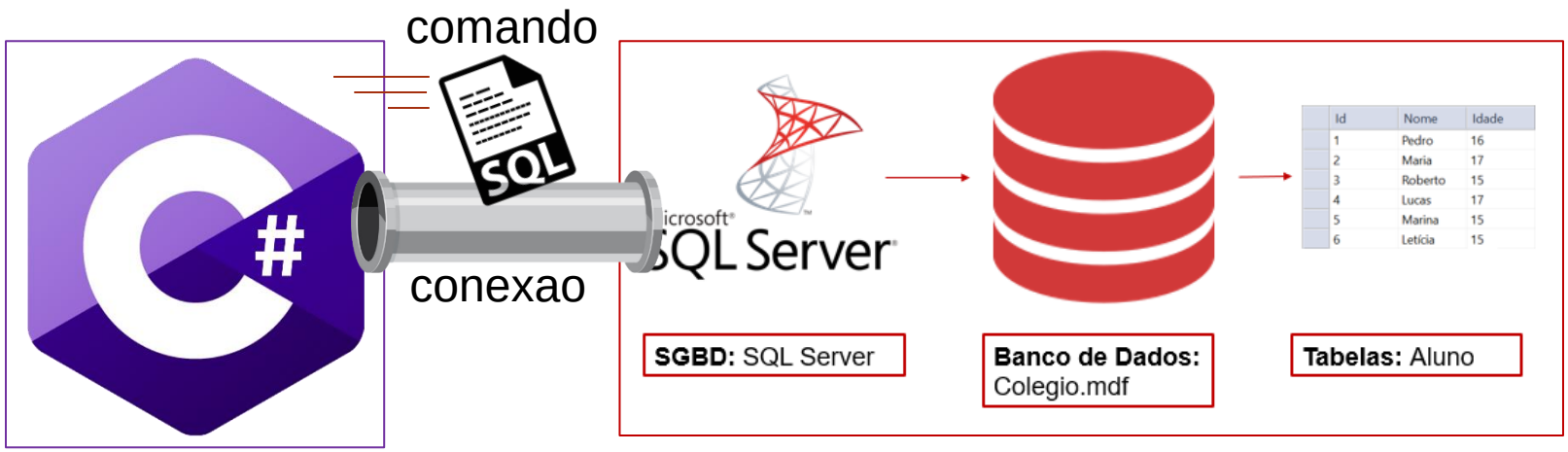
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlCommand: Enviando um comando

Colégio Pedro II

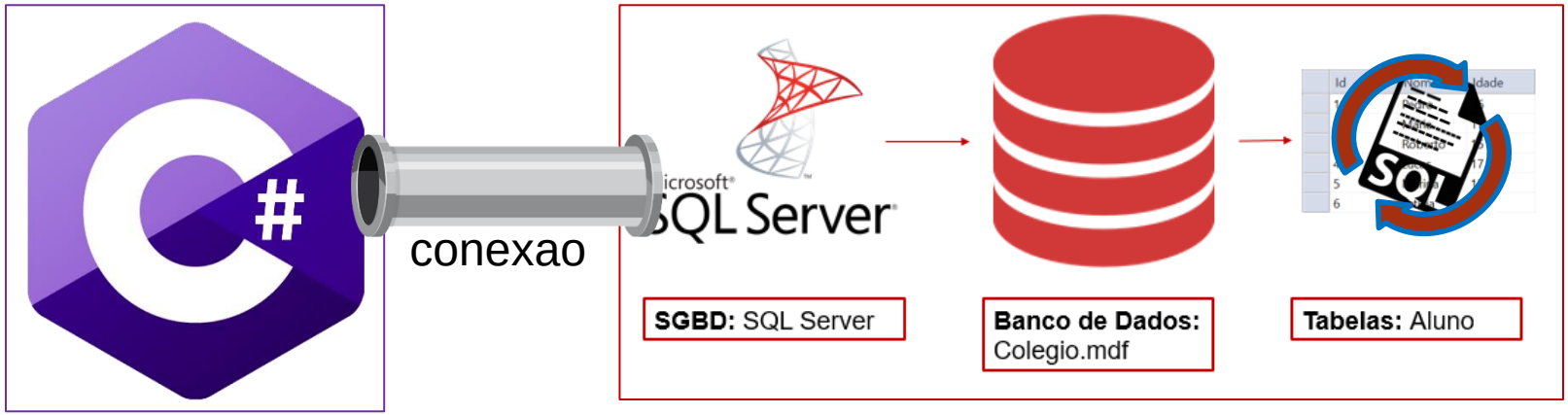
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlCommand: Enviando um comando

Colégio Pedro II

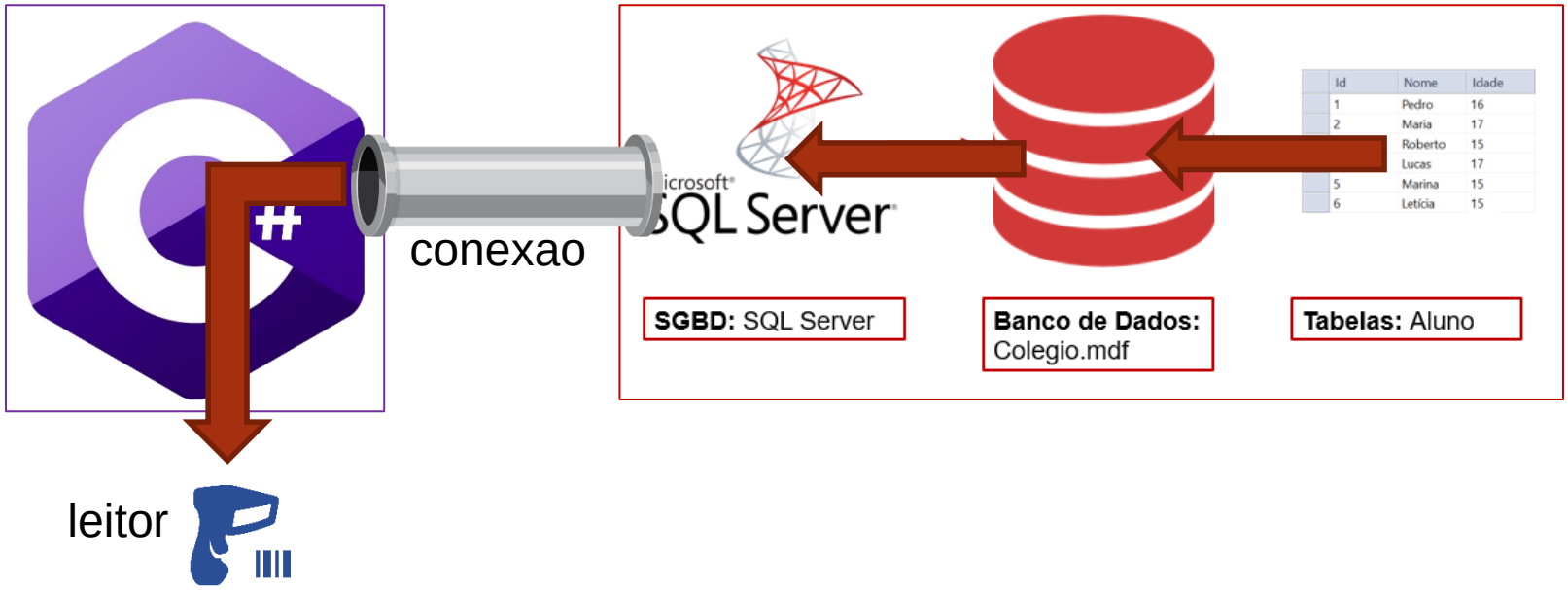
3. Comunicar sua aplicação com o banco de dados.

Exemplo

```
string consulta = "SELECT * FROM Aluno";

SqlCommand comando = new SqlCommand(consulta, conexao);

SqlDataReader leitor = comando.ExecuteReader();
```



SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

- A classe SqlDataReader é uma classe cuja instância representa uma espécie de leitor em uma tabela de um banco de dados (similar a uma leitor de arquivos).

- Você não precisa usar um construtor de SqlDataReader para se criar uma instância. O método ExecuteReader() de um objeto SqlCommand retorna uma instância de SqlDataReader de acordo com a instrução SQL que foi executada.

- Um objeto SqlDataReader tem um método chamado Read(). Esse método posiciona o cursor de leitura na próxima linha da tabela sempre que é chamado e retorna true se encontrou uma linha ou false se não há mais linhas para ler.

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor

	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor["Id"]

leitor["Nome"]

leitor["Idade"]

leitor

	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor

	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor["Id"]

leitor["Nome"]

leitor["Idade"]

leitor

	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor

	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```

leitor["Id"]

leitor["Nome"]

leitor[“Idade"]

leitor

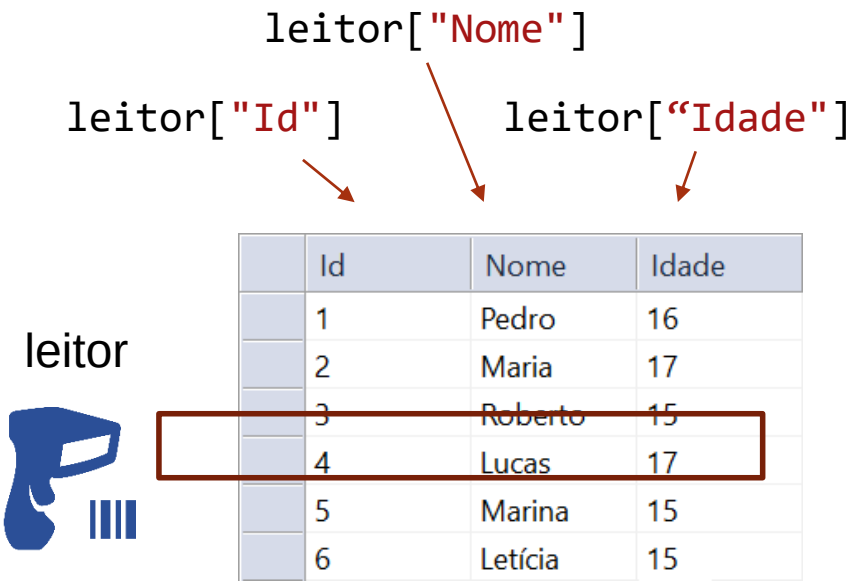
	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

SqlDataReader: Lendo a resposta do Banco de Dados

Colégio Pedro II

3. Comunicar sua aplicação com o banco de dados.

```
while(leitor.Read())
{
        listBox1.Items.Add(leitor["Nome"].ToString());
}
leitor.Close();
```



O continuará executando até que leitor.Read() retorne false. Isso irá acontecer quando não houver mais linhas na tabela para ler.

Tente interpretar o leitor como se fosse uma linha de uma tabela onde o acesso a cada elemento da linha pode ser dado pelo nome da coluna leitor[nome_da_coluna]

Passo a passo

Resumo do código

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(leitor["Nome"]);
            }

            leitor.Close();
            conexao.Close();
        }
```

Passo a passo

Resumo do código

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
{
    string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
d Security=True";

    SqlConnection conexao = new SqlConnection(stringDeConexao);

    conexao.Open();

    SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

    SqlDataReader leitor = comando.ExecuteReader();

    while(leitor.Read())
    {
        listBox1.Items.Add(leitor["Nome"]);
    }

    leitor.Close();
    conexao.Close();
}
```

string que contém as informações necessárias para encontrar o banco de dados Colegio.mdf na máquina.

Passo a passo

Resumo do código

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(l
            }

            leitor.Close();
            conexao.Close();
        }
```

Construção de um objeto conexao da classe SqlConnection. O construtor de SqlConnection espera receber uma string de conexao para saber onde (para qual banco de dados) a conexão deve ser estabelecida.

Passo a passo

Resumo do código

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(l
            }

            leitor.Close();
            conexao.Close();
        }
```

O método Open de conexao é responsável por criar um túnel entre a aplicação e o banco de dados.

Passo a passo

Resumo do código

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(l
            }

            leitor.Close();
            conexao.Close();
        }
```

Criacao de um objeto da classe SqlCommand. O construtor dessa classe espera receber dois parâmetros: a instrução SQL que se deseja enviar para o banco de dados e um objeto de SqlConnection, contendo o “túnel” pelo qual a instrução será enviada.

Passo a passo

Resumo do código

Colégio Pedro II

O método ExecuteReader() do objeto comando retorna um objeto preparado da classe SqlDataReader. Nesse comando, nós chamamos o método e atribuímos o seu valor de retorno a variável leitor.

```
private void button1_Click(object s
        {
            string stringDeConexao
         Source=(LocalDB)\MSSQLLoca
         Lagôas\source\repos\Exempl                                    e
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(leitor["Nome"]);
            }

            leitor.Close();
            conexao.Close();
        }
```

Passo a passo

Resumo do código

Colégio Pedro II

O método Read() de leitor posiciona o cursor de linhas na tabela.

```
private void button1_Click(object s
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(leitor["Nome"]);
            }

            leitor.Close();
            conexao.Close();
        }
```

Passo a passo

Resumo do código

Colégio Pedro II

Para acessar um elemento da linha onde o leitor está posicionado, basta indicar o nome da coluna como se fosse um vetor.

```
private void button1_Click(object s
        {
            string stringDeConexao
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(leitor["Nome"]);
            }

            leitor.Close();
            conexao.Close();
        }
```

Passo a passo

Resumo do código

Colégio Pedro II

Por fim, feche o leitor e a conexão.

```
private void button1_Click(object
        {
            string stringDeConexao = @"Data
         Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
         Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrate
         d Security=True";

            SqlConnection conexao = new SqlConnection(stringDeConexao);

            conexao.Open();

            SqlCommand comando = new SqlCommand("SELECT * FROM Aluno", conexao);

            SqlDataReader leitor = comando.ExecuteReader();

            while(leitor.Read())
            {
                listBox1.Items.Add(leitor["Nome"]);
            }

            leitor.Close();
            conexao.Close();
        }
```

Banco de Dados e Exceções

- Quando trabalhamos com banco de dados, vários trechos do código ficam sujeitos a lançar **exceções** (erros de execução). É importante então, nesses casos, utilizar os conceitos de tratamento de exceção para tornar o código mais robusto.

- Em particular, os métodos Open() de uma SqlConnection e o método ExecuteReader() de um SqlCommand, podem lançar exceções facilmente. Basta a conexão de string estar errada ou a consulta SQL estar errada, por exemplo.

Banco de Dados e Exceções

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"...";

            SqlDataReader leitor = null;
            SqlConnection conexao = null;
            SqlCommand comando = null;

            try
            {
                conexao = new SqlConnection(stringDeConexao);
                conexao.Open();

                string instrucaoSQL = "SELECT * FROM Aluno;";

                comando = new SqlCommand(instrucaoSQL, conexao);

                leitor = comando.ExecuteReader();

                while (leitor.Read())
                    listBox1.Items.Add(leitor["Nome"]);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                if (leitor != null)
                    leitor.Close();

                if (conexao != null)
                    conexao.Close();
            }
        }
```

Código de risco

É importante preparar seu código para lidar com esses métodos.

Banco de Dados e Exceções

- Sabemos que o comando **using** pode ser utilizado para liberar recursos sempre que necessário.

- Recursos são solicitados com certa frequência em aplicações que se comunicam com banco de dados. No exemplo que vimos, tanto a classe SqlConnection quanto a classe SqlDataReader fazem uso de recurso externo e portanto precisam ser fechadas.

- Podemos reescrever o código usando o comando **using** para liberar os recursos alocados sem que o programador tenha que se preocupar com métodos Close().

Banco de Dados e Exceções

Colégio Pedro II

```
private void button1_Click(object sender, EventArgs e)
        {
            string stringDeConexao = @"...

            try
            {
                using (SqlConnection conexao = new SqlConnection(stringDeConexao))
                {
                    conexao.Open();
                    string consultaSQL = "SELECT * FROM Aluno;";

                    SqlCommand comando = new SqlCommand(consultaSQL, conexao);

                    using (SqlDataReader leitor = comando.ExecuteReader())
                    {
                        while (leitor.Read())
                            listBox1.Items.Add(leitor["Nome"].ToString());
                    }
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Os recursos serão liberados automaticamente, sem precisar de um bloco finally.

Ainda sim é necessário usar bloco try/catch para capturar exceções que venham a ser lançadas.

Banco de Dados e Exceções

- Usar apenas try/catch/finally ou using com try/catch é uma escolha do programador.

- Essa escolha pode estar envolvida tanto com a demanda da aplicação quanto com as práticas que o desenvolvedor esteja usando.

- Não existe certo ou errado nesses casos, apenas o que realiza o que é esperado de forma simples.