



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?



- De modo geral, um 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?



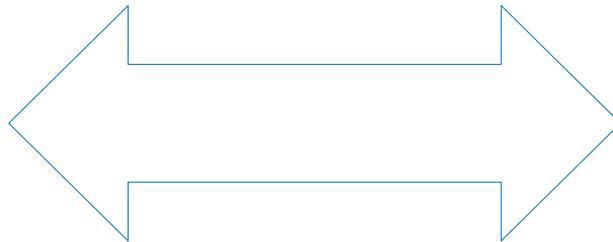
Colégio  
Pedro II

- 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



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



# Abordagem Tradicional



- 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**.

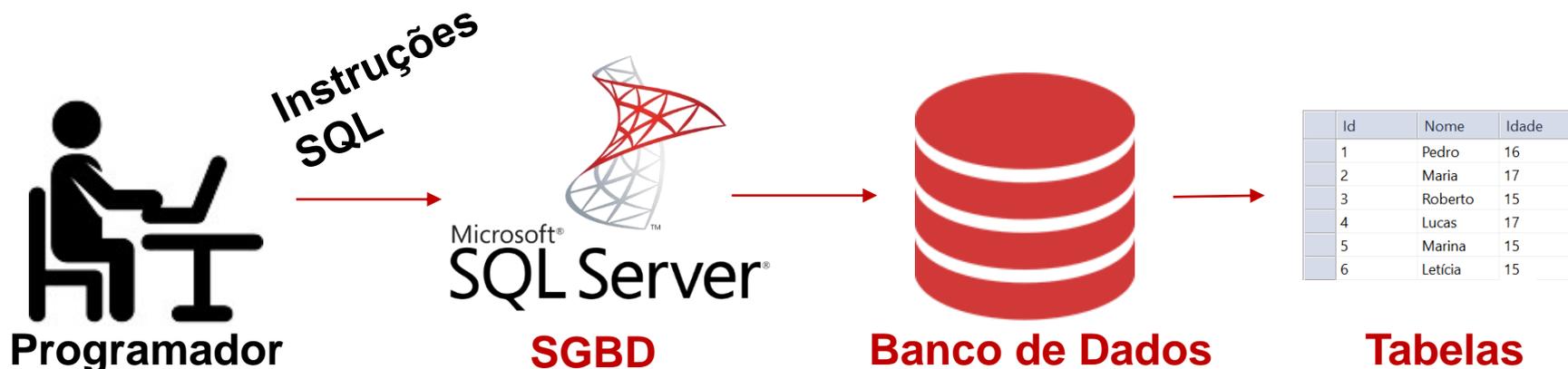


# SGBD

## Sistema Gerenciador de Banco de Dados



- 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

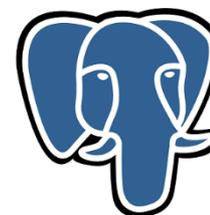
## Sistema Gerenciador de Banco de Dados



- 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.



PostgreSQL



Microsoft®  
SQL Server®

ORACLE



# SQL

## Linguagem de Consulta Estruturada



- “*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



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

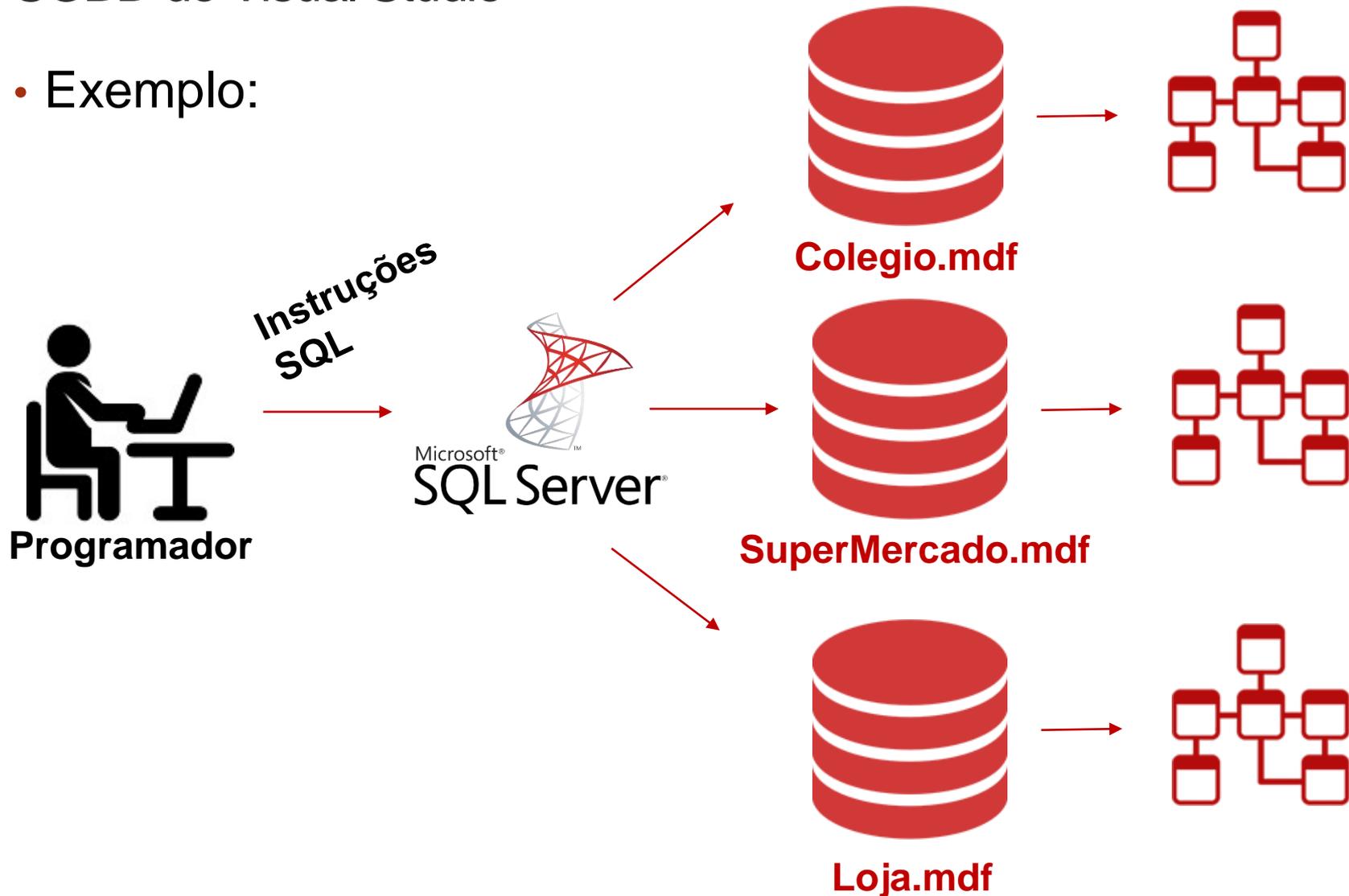
- Bancos de dados SQL Server usam dois arquivos:
  - Um **.mdf** conhecido como **arquivo de banco de dados primário**. Esse arquivo contém os esquemas, tabelas, registros e dados.
  - Um **.ldf** que consiste de um **arquivo de log** (log é uma espécie de registro de ocorrências no banco de dados).

# Microsoft SQL Server

## SGBD do Visual Studio



- Exemplo:



# Sistema de Banco de Dados



Como se dá a comunicação (Teoria)

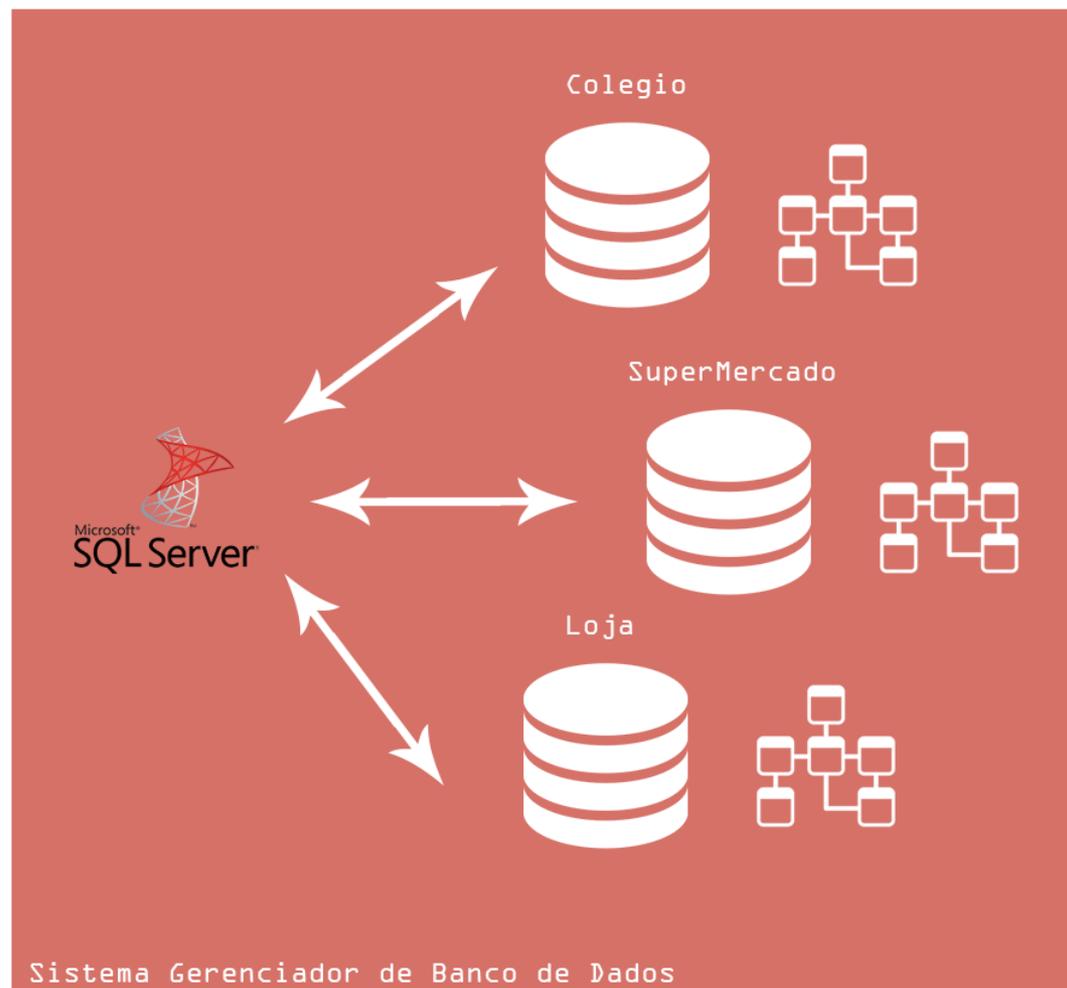
1. A sua aplicação deve se conectar com o Banco de Dados gerenciado por algum SGBD.
2. Comandos SQL são enviados através da conexão estabelecida no passo anterior para o SGBD.
3. O SGBD executa a consulta SQL e devolve para a aplicação uma resposta. Por exemplo: o resultado de uma consulta SELECT.
4. 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)



Programa



# Sistema de Banco de Dados

Como se dá a comunicação (Teoria)



## 1. Estabelecer conexão



Colegio



SuperMercado



Loja



# Sistema de Banco de Dados

Como se dá a comunicação (Teoria)



## 2. Enviar comandos SQL

```
SELECT * FROM Aluno
```



Microsoft  
SQL Server

Colegio



SuperMercado



Loja

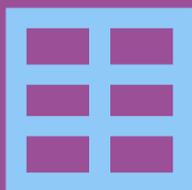


# Sistema de Banco de Dados

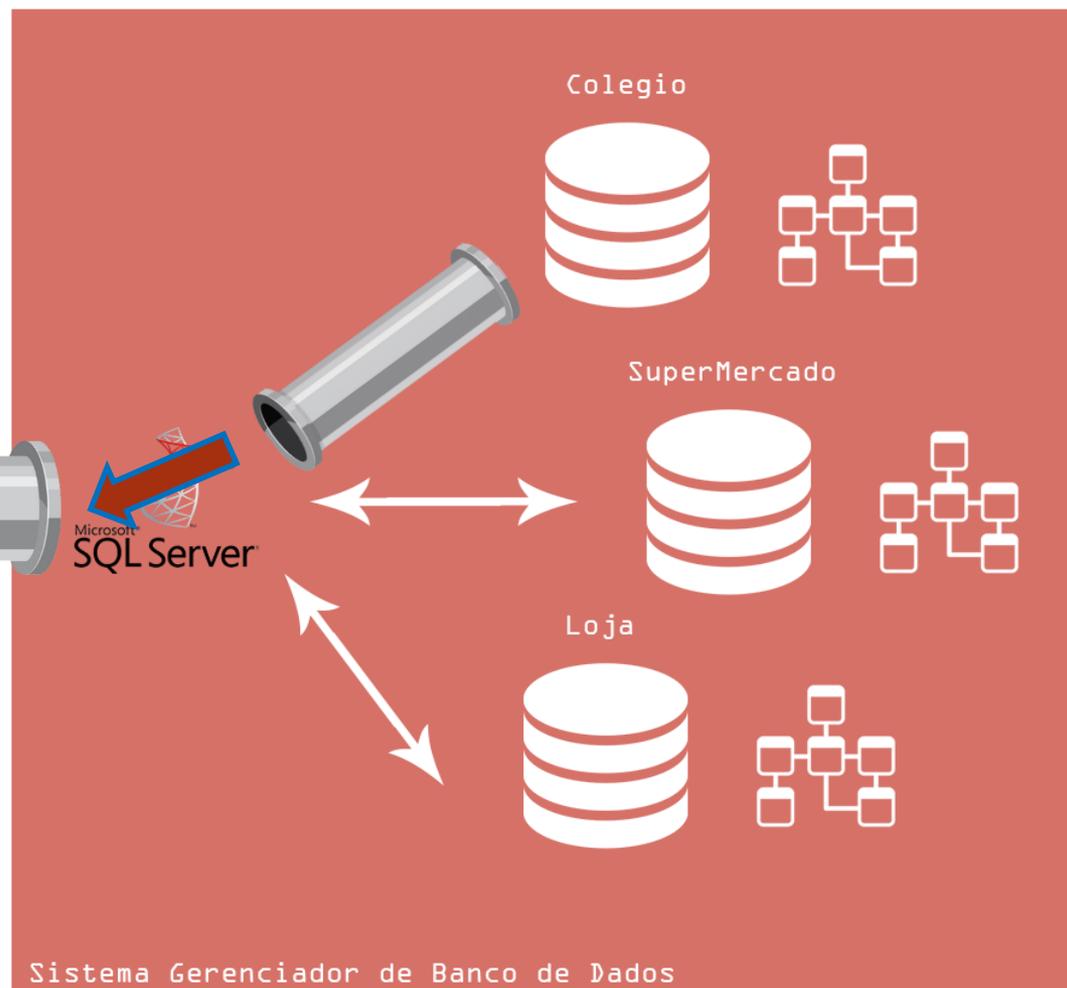
Como se dá a comunicação (Teoria)



## 3. Responder consulta SQL



Programa

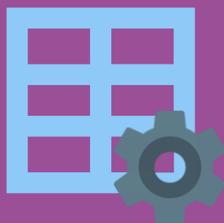


# Sistema de Banco de Dados

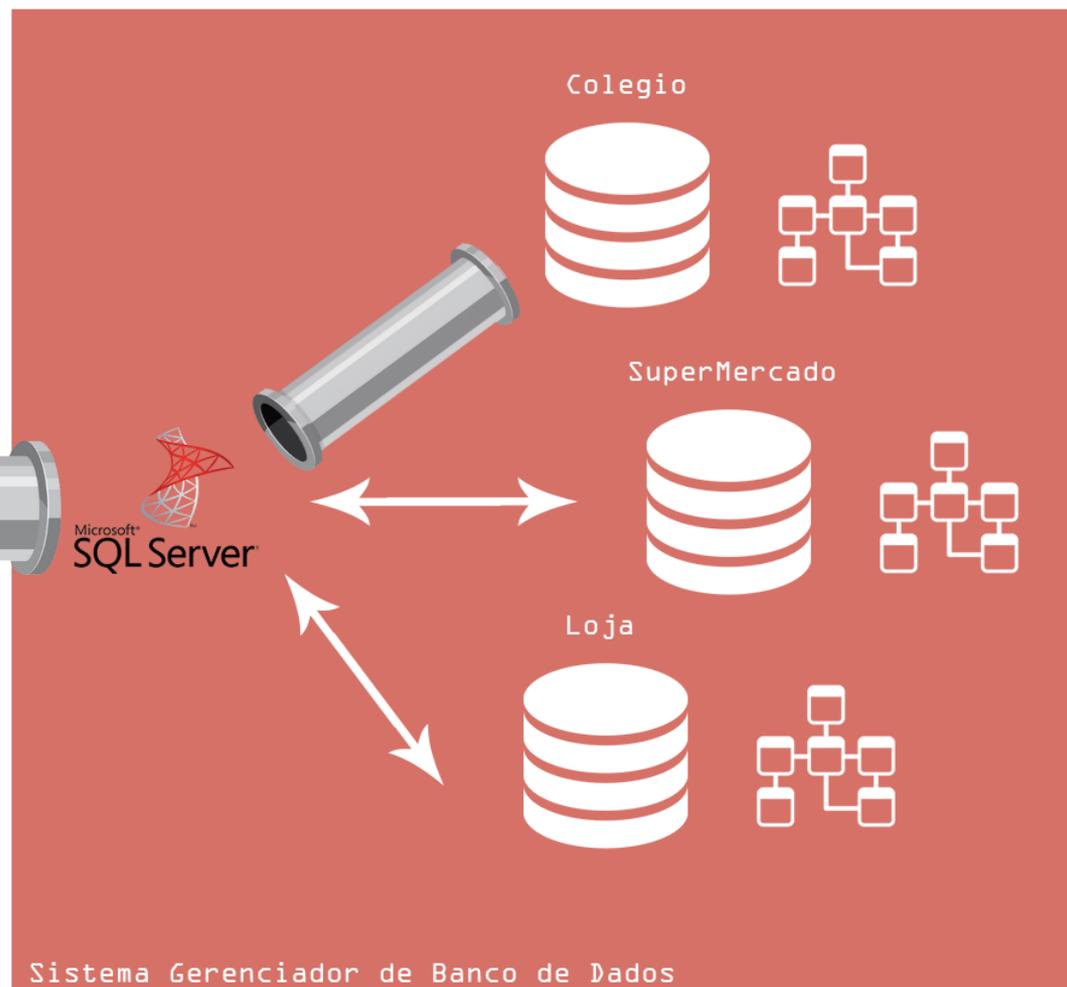
Como se dá a comunicação (Teoria)



4. Aplicação  
utiliza a resposta



Programa

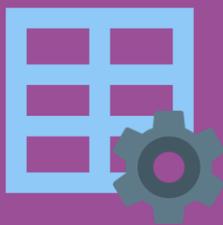


# Sistema de Banco de Dados

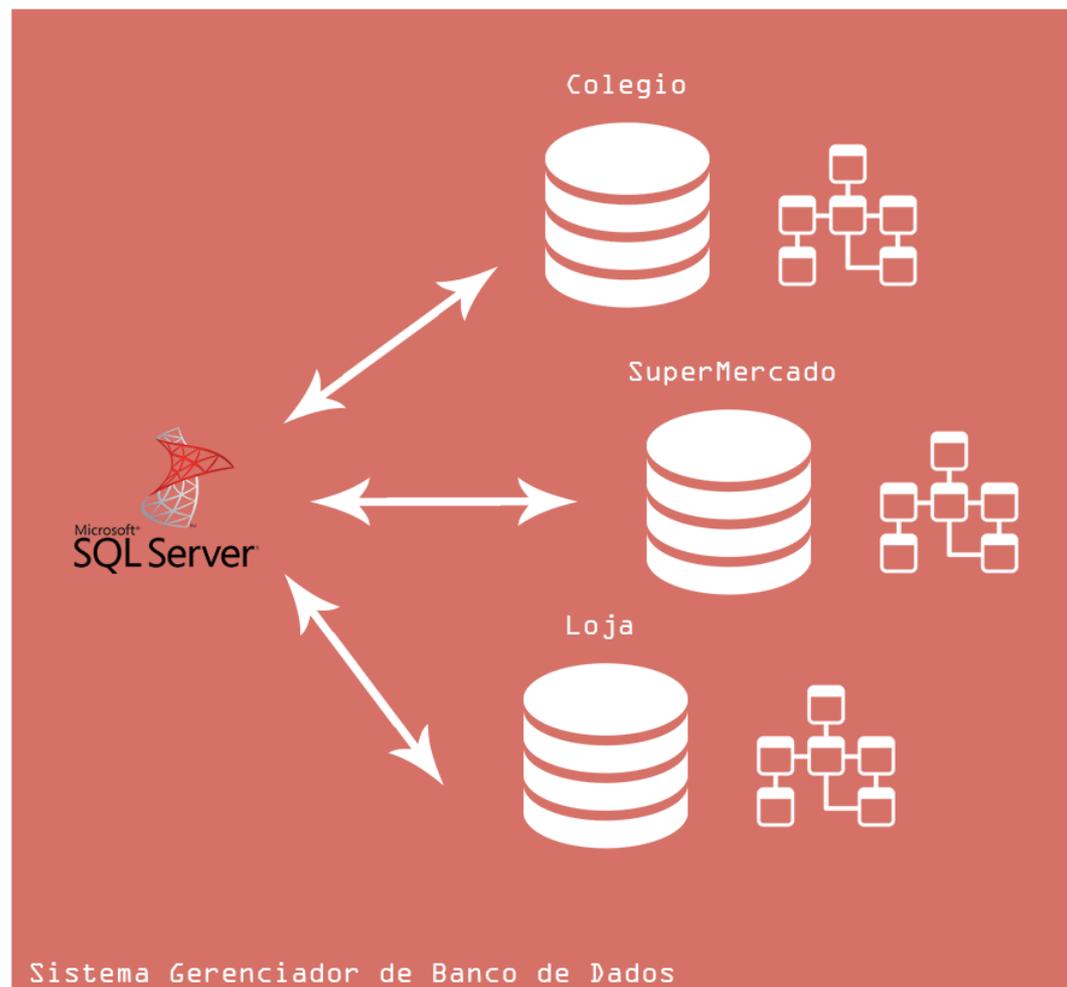
Como se dá a comunicação (Teoria)



## 5. Fechar conexão

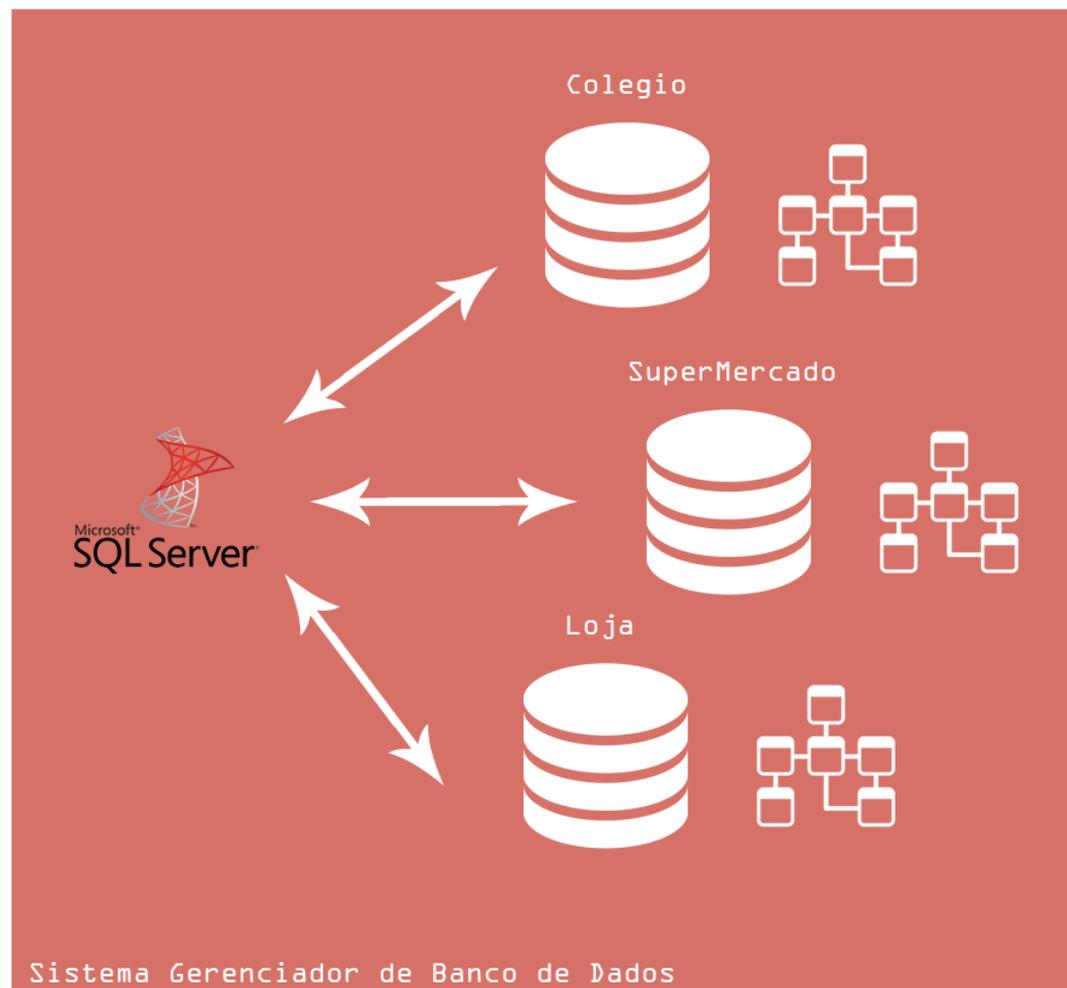
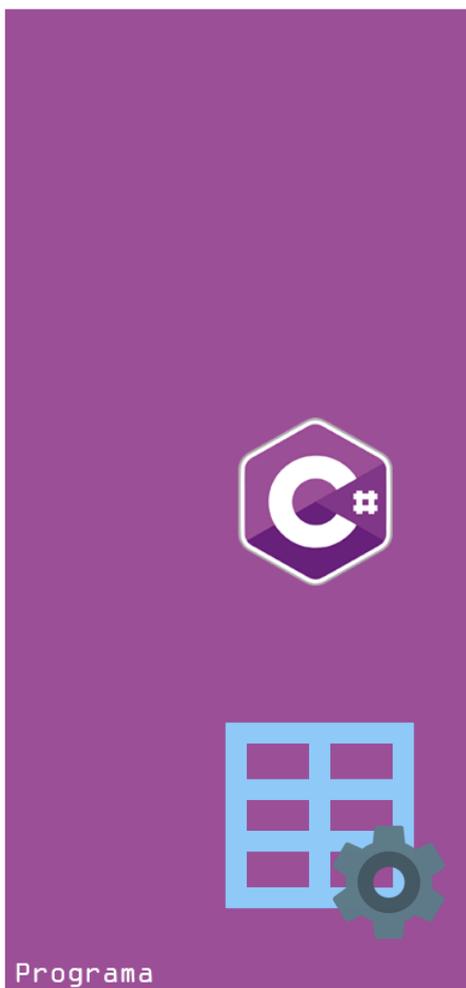


Programa



# Sistema de Banco de Dados

## Como se dá a comunicação (Teoria)



# Passo a passo

## Criando um Sistema de Banco de Dados

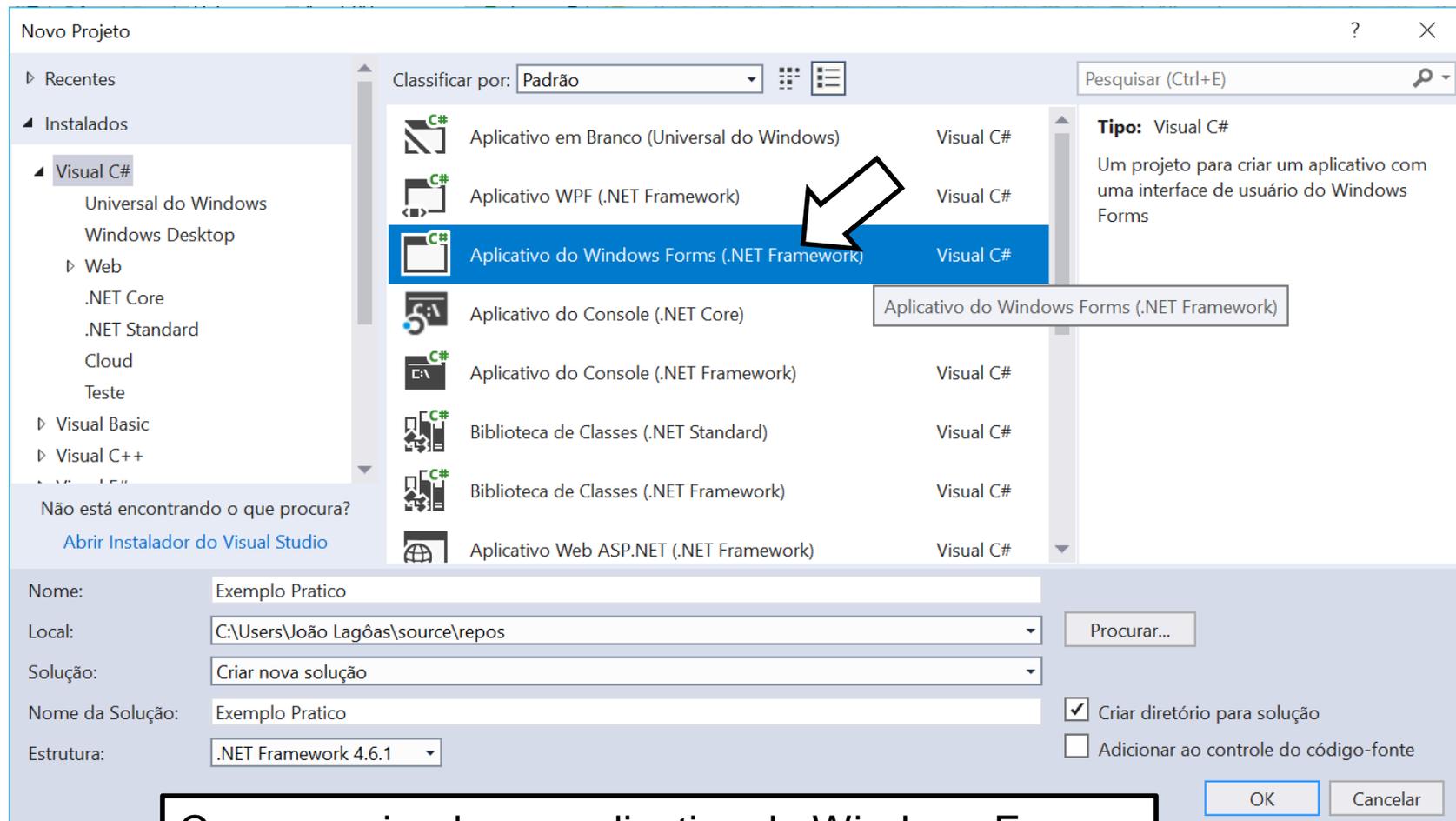


- Para se criar um Sistema de Banco de Dados, iremos:
  - 1. Preparar o ambiente de banco de dados:**
    1. Criar o banco de dados (arquivo .mdf);
    2. Criar as tabelas do banco de dados;
    3. Popular as tabelas com alguns registros.
  - 2. Preparar a nossa aplicação:**
    1. Criar os formulários 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



## 1. Preparar o ambiente de banco de dados:



Comece criando um aplicativo do Windows Forms

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

Clique com o botão direito do mouse no seu projeto (no Gerenciador de Soluções). Clique em Adicionar e depois Novo Item...

The screenshot shows the Microsoft Visual Studio interface. The Solution Explorer on the left displays a project named 'ExemploPratico' with files like 'Form1.cs', 'Form1.Designer.cs', and 'Program.cs'. The context menu is open over the project, and the 'Adicionar' option is selected, opening a sub-menu. In this sub-menu, 'Novo Item...' is highlighted, and an orange arrow points to it. The sub-menu also includes options like 'Item Existente...', 'Nova Pasta', 'Cliente API REST...', 'Referência...', 'Referência Web...', 'Referência de Serviço...', 'Serviço Conectado', 'Analisador...', 'Windows Form...', 'Controle de Usuário...', 'Componente...', and 'Classe...'. The status bar at the bottom indicates 'Pronto' and 'Adicionar ao Controle do Código-Fonte'.

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

Clique em Banco de Dados baseado em Serviço

ExemploPratico - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Adicionar Novo Item - ExemploPratico

Gerenciador de Soluções

Instalados

- Itens do Visual C#
  - Código
  - Dados
  - Geral
  - Web
  - Windows Forms
  - WPF
  - AspNetCore
  - SQL Server
  - Storm Items
  - Gráficos
- Online

Classificar por: Padrão

Pesquisar (Ctrl+E)

Ícone	Nome do Item	Categoria
TS	Arquivo JSX TypeScript	Itens do Visual C#
TS	Arquivo TypeScript	Itens do Visual C#
XML	Arquivo XML	Itens do Visual C#
XSLT	Arquivo XSLT	Itens do Visual C#
Banco de Dados	Banco de Dados baseado em Serviço	Itens do Visual C#
Caixa Sobre	Caixa Sobre	Itens do Visual C#
Classe de Instalador	Classe de Instalador	Itens do Visual C#
Conjunto de Regras da Análise de Código	Conjunto de Regras da Análise de Código	Itens do Visual C#
Controle Personalizado	Controle Personalizado	Itens do Visual C#
DataSet	DataSet	Itens do Visual C#

Tipo: Itens do Visual C#

Um banco de dados SQL Server vazio para acesso a dados baseado em serviços

Nome: Colegio.mdf

Adicionar Cancelar

Abrir Pasta no Gerenciador de Arquivos

Componente...

Adicionar ao Controle do Código-Fonte

Dê um nome para o seu Banco de Dados.

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

Observe que na janela de Gerenciador de Servidores, seu banco de dados foi criado.

**Nota:** se o Gerenciador de Servidores não estiver disponível, clique em Exibir e depois Gerenciador de Servidores ou então aperte simultaneamente no teclado: Ctrl + Alt + S

# Passo a passo



## 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



## 1. Preparar o ambiente de banco de dados:

ExemploPratico - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Gerenciador de Soluções

Solução 'ExemploPratico' (1 projeto)

- ExemploPratico
  - Properties
  - Referências
  - App.config
  - Colegio.mdf
  - Form1.cs
  - Program.cs

Atualizar Arquivo de Script: dbo.Table.sql\*

Nome	Tipo de Dados	Permitir Valores Nulos	Padrão
Id	int	<input type="checkbox"/>	

Design T-SQL

```
1 CREATE TABLE [dbo].[Aluno]
2 (
3     [Id] INT NOT NULL PRIMARY KEY
4 )
5
```

110 %

Conexão Pronto

Saída

Mostrar saída de: Depuração

Operações de Ferramentas de Dados Saída

Gerenciador de Servid...

- Azure (lagoas.joao Luiz@out)
- Conexões de Dados
  - Colegio.mdf
    - Tabelas
      - Modos de Exibição
      - Procedimentos arm
      - Funções
      - Sinônimos
      - Tipos
      - Assemblies
    - Servidores

Notificações

Chaves (1)  
<sem nome> (Chave Primária, Clustered: Id)

Restrições de Verificação (0)

Índices (0)

Chaves Estrangeiras (0)

Gatilhos (0)

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

Pronto Li 1 Col 25 Car 25 INS Adicionar ao Controle do Código-Fonte

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

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

Nome	Tipo de Dados	Permitir Valores Nulos	Padrão
Id	int	<input type="checkbox"/>	
Nome	nvarchar(50)	<input checked="" type="checkbox"/>	
Idade	int	<input checked="" type="checkbox"/>	

```
1 CREATE TABLE [dbo].[Aluno]
2 (
3     [Id] INT NOT NULL PRIMARY KEY,
4     [Nome] NVARCHAR(50) NULL,
5     [Idade] INT NULL
```

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



## 1. Preparar o ambiente de banco de dados:

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

The screenshot displays the Microsoft SQL Server Enterprise Designer interface. The main window shows the design view of a table named 'Aluno' in the 'dbo' schema. The table has three columns: 'Id' (int, primary key, not null), 'Nome' (nvarchar(50), nullable), and 'Idade' (int, nullable). The 'Atualizar' button is highlighted with an orange arrow. The T-SQL view shows the following code:

```
1 CREATE TABLE [dbo].[Aluno]
2 (
3     [Id] INT NOT NULL PRIMARY KEY,
4     [Nome] NVARCHAR(50) NULL,
5     [Idade] INT NULL
```

The interface also shows the 'Gerenciador de Soluções' (Solution Explorer) on the left, the 'Gerenciador de Servidores' (Server Enterprise Explorer) on the right, and the 'Saída' (Output) window at the bottom.

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

Uma janela de confirmação será exibida. Clique em Atualizar Banco de Dados para que a instrução CREATE TABLE... seja executada pelo SQL Server.

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. A confirmation dialog box is open, titled 'Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF'. The dialog contains the following information:

- Destaques:** Nenhum
- Ações do usuário:** Criar [dbo].[Aluno] (Tabela)
- Ações de suporte:** Nenhum

At the bottom of the dialog, there are three buttons: 'Gerar Script', 'Atualizar Banco de Dados', and 'Cancelar'. An orange arrow points to the 'Atualizar Banco de Dados' button. The background shows the 'Colegio.mdf' database structure with a table named 'Aluno'.

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

Se a instrução era válida e foi capaz de ser executada sem gerar erros no banco de dados Colegio.mdf, então uma janela de sucesso será exibida.

Nome	Tipo de Dados	Permitir Valores Nulos	Padrão
Id	int	<input type="checkbox"/>	
Nome	nvarchar(50)	<input checked="" type="checkbox"/>	
Idade	int	<input checked="" type="checkbox"/>	

Operações de Ferramentas de Dados

- ✓ Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:4
  - ✓ Criando visualização de atualização... [Exibir Visualização](#)
  - ✓ Exibindo visualização de atualização...
  - ✓ Criando script de banco de dados... [Exibir Script](#)
  - ✓ Executando script de atualização no banco de dados C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXE... [Exibir Resultados](#)
  - Atualização concluída com êxito
- ✗ Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:3

# 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.

The screenshot displays the SQL Server Enterprise Manager interface. The 'Operações de Ferramentas de Dados' (Data Tools Operations) pane shows a list of operations. The top operation, 'Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:4', is marked with a green checkmark and shows a progress bar. Below it, the 'Atualizar' button is highlighted with an orange arrow. The 'Tabelas' (Tables) folder in the right-hand pane is expanded, showing the 'Aluno' table. The 'Criando script de banco de dados...' (Creating database script...) step is also visible in the progress bar.

```
1 CREATE TABLE [dbo].[Aluno]
2 (
3     [Id] INT NOT NULL PRIMARY KEY,
```

# Passo a passo



## 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.

The screenshot displays the SQL Server Enterprise Manager interface. The central pane shows the 'Aluno' table structure with columns: Id (int, primary key), Nome (nvarchar(50)), and Idade (int). A context menu is open over the table, with 'Mostrar Dados da Tabela' highlighted. An orange arrow points to this option. The background shows the SQL script for creating the table and the 'Operações de Ferramentas de Dados' pane with a successful update message.

```
1 CREATE TABLE [dbo].[Aluno]
2 (
3     [Id] INT NOT NULL PRIMARY KEY,
```

Operações de Ferramentas de Dados

- Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:4
  - ✓ Criando visualização de atualização... [Exibir Visualização](#)
  - ✓ Exibindo visualização de atualização...
  - ✓ Criando script de banco de dados... [Exibir Script](#)
  - ✓ Executando script de atualização no banco de dados C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXE... [Exibir Resultados](#)Atualização concluída com êxito
- Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÔAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:3

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

ExemploPratico - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe SQL Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Gerenciador de Soluções

Solução 'ExemploPratico' (1 projeto)

- ExemploPratico
  - Properties
  - Referências
  - App.config
  - Colegio.mdf
  - Form1.cs
  - Program.cs

dbo.Aluno [Dados] | dbo.Aluno [Design]

Máx. de Linhas: 1000

Id	Nome	Idade
1	Pedro	16
2	Maria	17
3	Roberto	15
4	Lucas	17
5	Marina	15
6	Leticia	18
*	NULL	NULL

Gerenciador de Servid...

- Azure (lagoas.joaoluiz@out)
- Conexões de Dados
  - Colegio.mdf
    - Tabelas
      - Aluno
    - Modos de Exibição
    - Procedimentos arm...
    - Funções
    - Sinônimos
    - Tipos
    - Assemblies
  - Servidores

Operações de Ferramentas de Dados

Atualização concluída com êxito

Atualizar (LocalDB)\MSSQLLocalDB.C:\USERS\JOÃO LAGÓAS\SOURCE\REPOS\EXEMPLOPRATICO\EXEMPLOPRATICO\COLEGIO.MDF 20:32:3

Operações de Ferramentas de Dados Saída

Pronto Li 6 Col 3 Adicionar ao Controle do Código-Fonte

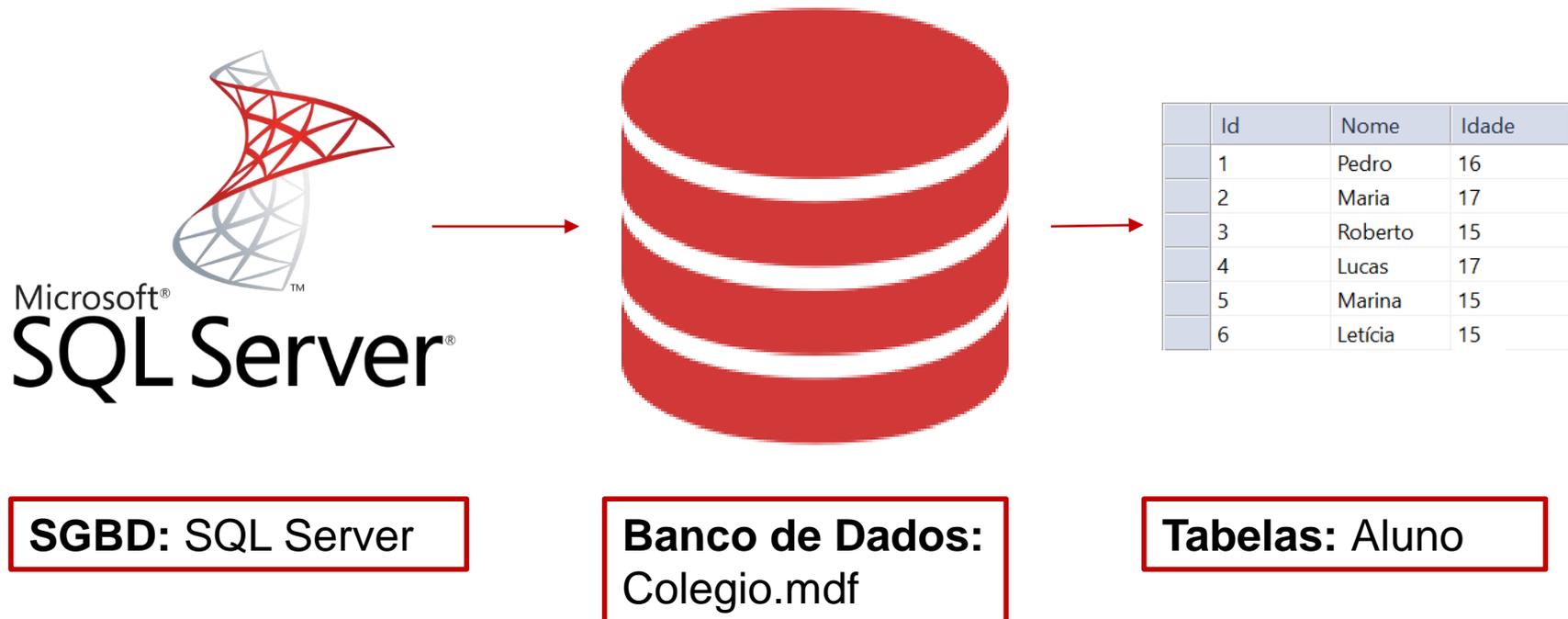
**Crie alguns registros inserindo novas informações na tabela.**

# Passo a passo



## 1. Preparar o ambiente de banco de dados:

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

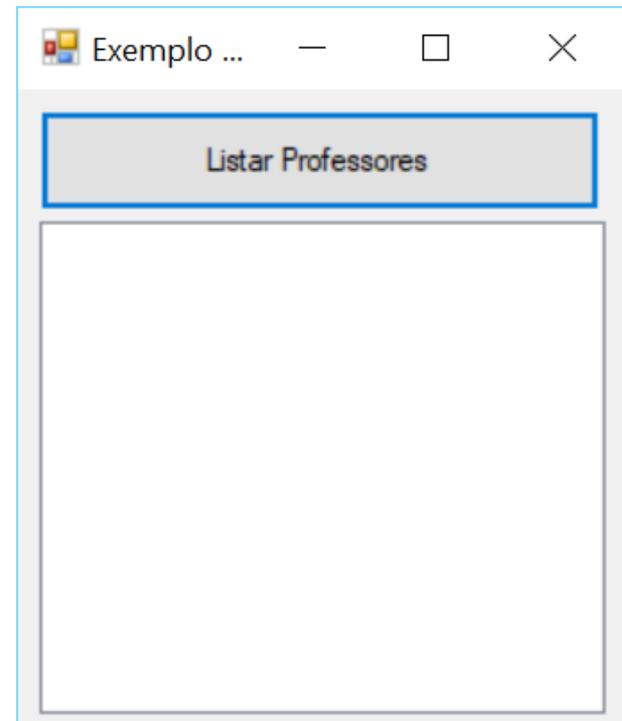


# Passo a passo



## 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:

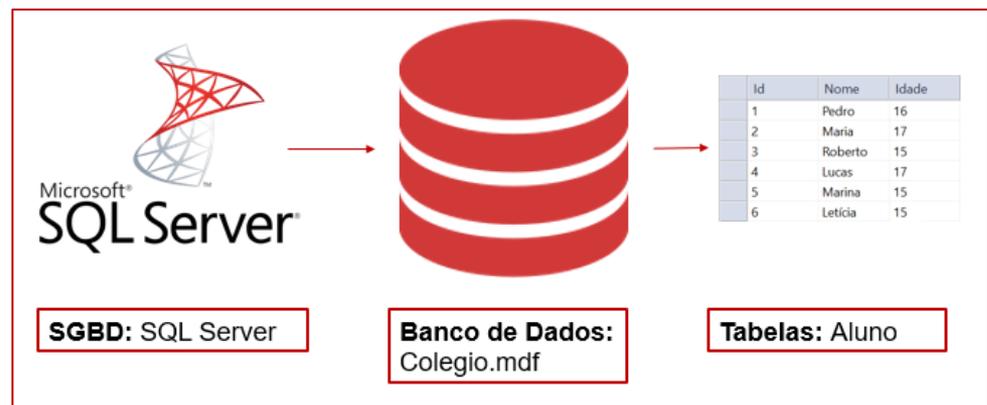
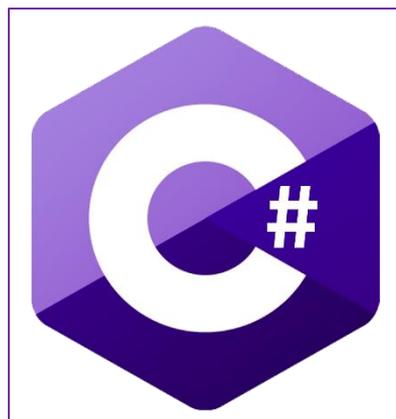


# Passo a passo



## 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.



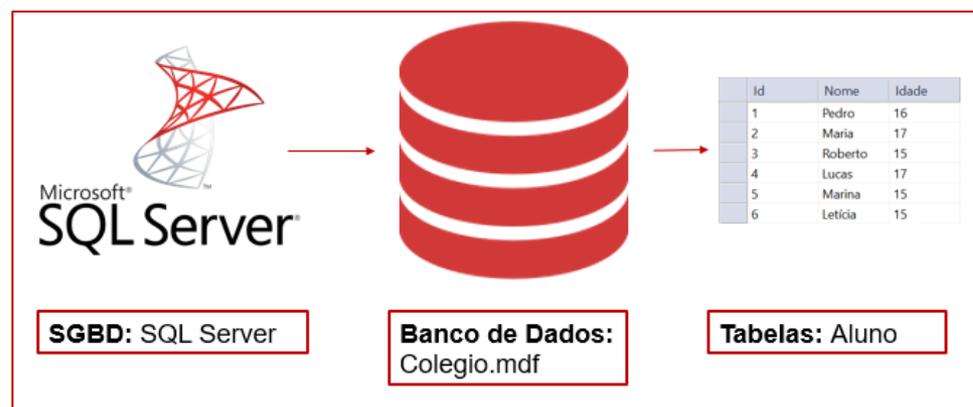
# Passo a passo



## 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



## 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



## 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.

The screenshot displays the SQL Server Enterprise Manager interface. In the 'Gerenciador de Servidor' tree, the 'Colégio.mdf' database file is selected. A context menu is open, showing options like 'Atualizar', 'Excluir', 'Alterar Exibição', etc. The 'Propriedades' option is highlighted, and an orange arrow points to it. The background shows a table design for 'Aluno' with columns 'Id', 'Nome', and 'Idade', and a T-SQL script window with a 'CREATE TABLE' statement.

Id	Nome	Idade
int	nvarchar(50)	int

```
1 CREATE TABLE [dbo].[Aluno] (  
2 [Id] INT NOT NULL,  
3 [Nome] NVARCHAR (50) NULL,
```

# Passo a passo

## 3. Comunicar sua aplicação com o banco de dados.



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



## 3. Comunicar sua aplicação com o banco de dados.

**SqlConnection**

**SqlCommand**

**SqlDataReader**

**SqlAdapter**

namespace  
SqlClient

- 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



## 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



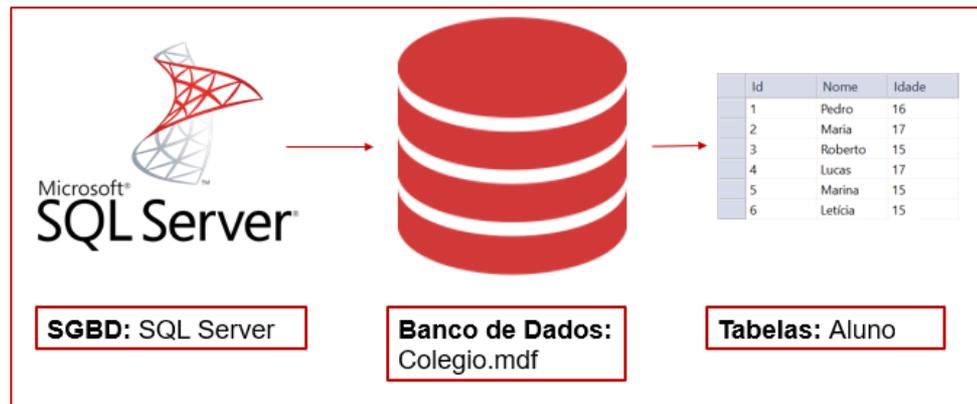
## 3. Comunicar sua aplicação com o banco de dados.

```
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();
```

Exemplo



# SqlConnection: Realizando a conexão



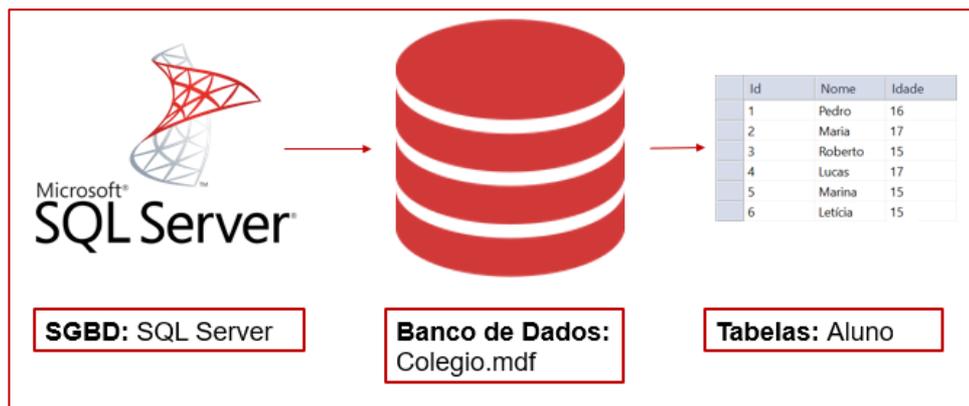
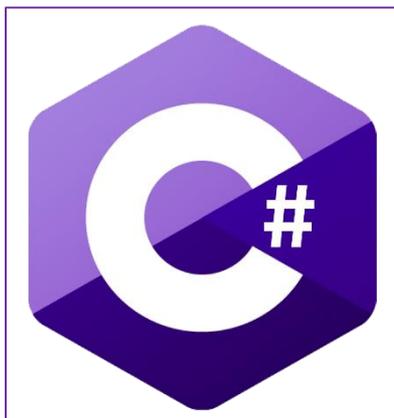
## 3. Comunicar sua aplicação com o banco de dados.

```
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();
```

Exemplo



# SqlConnection: Realizando a conexão



## 3. Comunicar sua aplicação com o banco de dados.

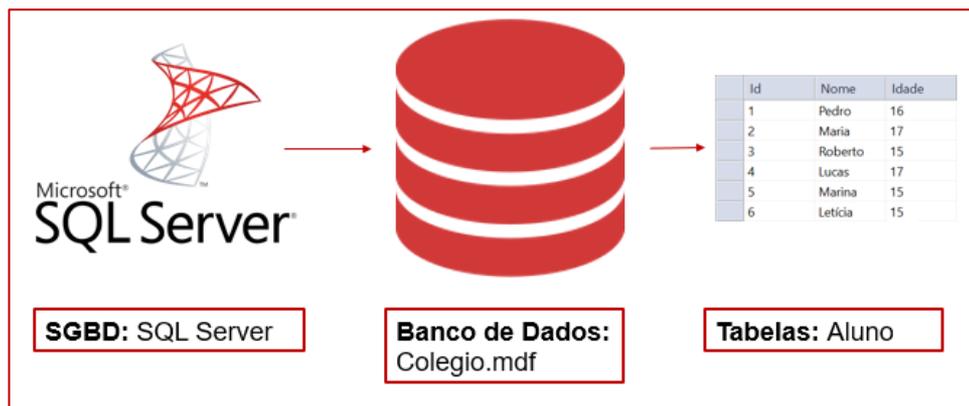
```
string stringDeConexao = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\João
Lagôas\source\repos\ExemploPratico\ExemploPratico\Colegio.mdf';Integrated
Security=True";
```



```
SqlConnection conexao = new SqlConnection(stringDeConexao);
```

```
conexao.Open();
```

Exemplo



# SqlConnection: Realizando a conexão



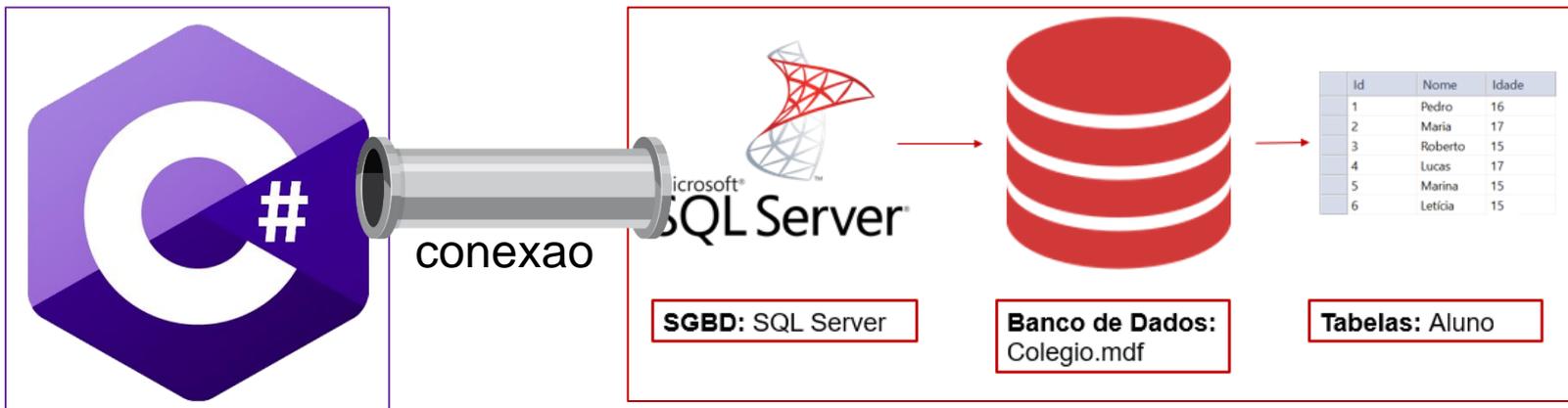
## 3. Comunicar sua aplicação com o banco de dados.

```
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();
```

Exemplo



## SqlCommand: Enviando um comando



### 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).

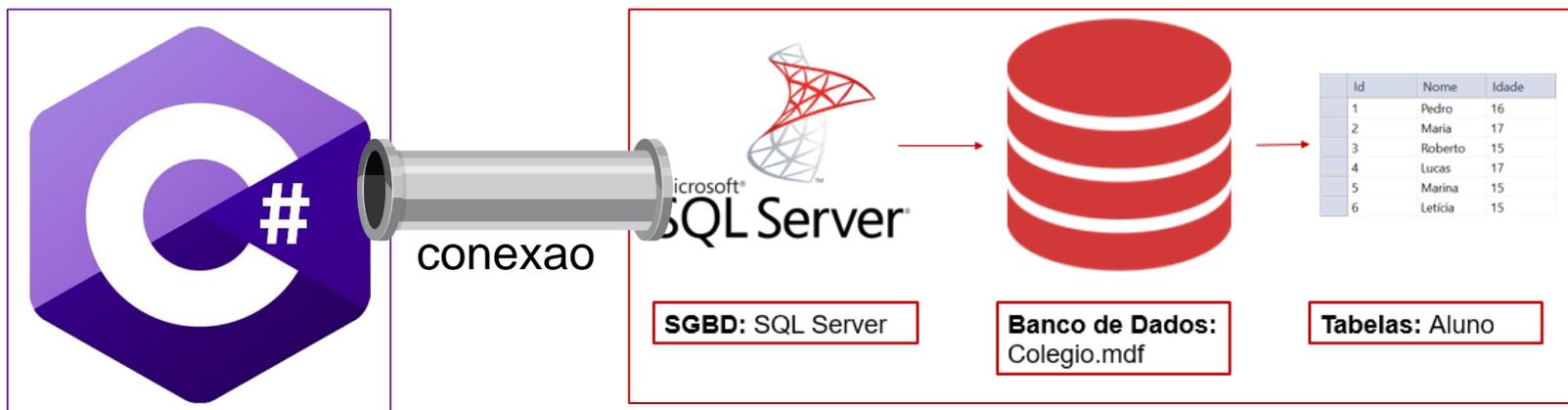
# SqlCommand: Enviando um comando



## 3. Comunicar sua aplicação com o banco de dados.

```
string consulta = "SELECT * FROM Aluno";
SqlCommand comando = new SqlCommand(consulta, conexao);
SqlDataReader leitor = comando.ExecuteReader();
```

Exemplo



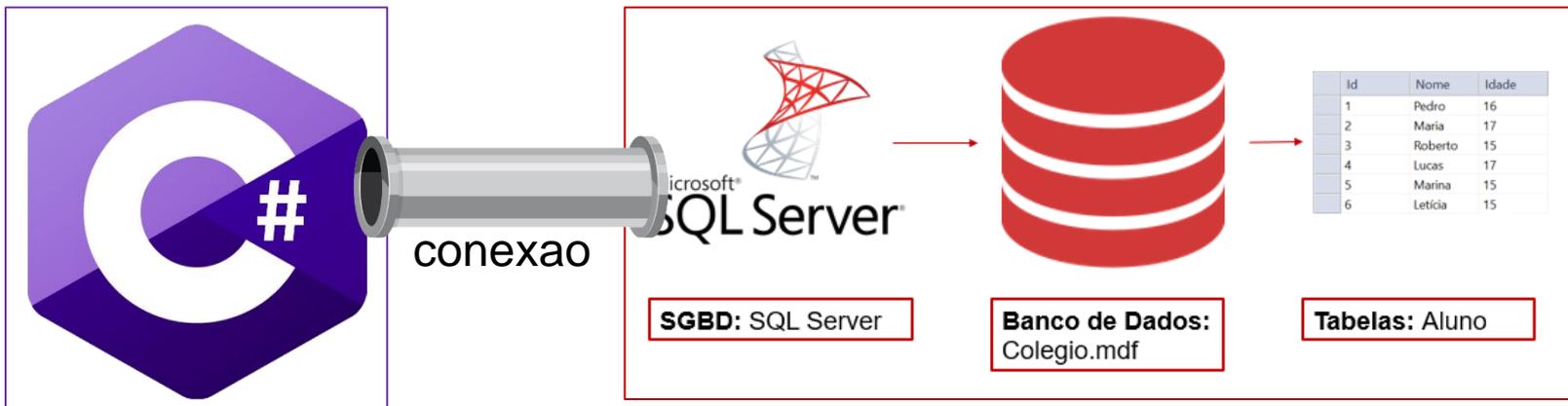
# SqlCommand: Enviando um comando



## 3. Comunicar sua aplicação com o banco de dados.

```
string consulta = "SELECT * FROM Aluno";
SqlCommand comando = new SqlCommand(consulta, conexao);
SqlDataReader leitor = comando.ExecuteReader();
```

Exemplo



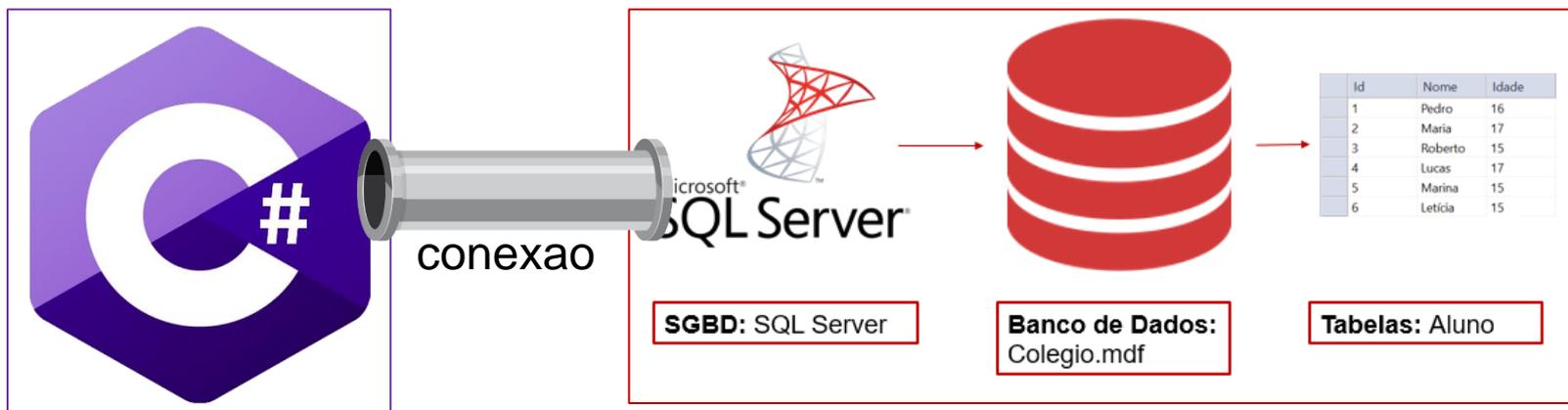
# SqlCommand: Enviando um comando



## 3. Comunicar sua aplicação com o banco de dados.

```
string consulta = "SELECT * FROM Aluno";
SqlCommand comando = new SqlCommand(consulta, conexao);
SqlDataReader leitor = comando.ExecuteReader();
```

Exemplo



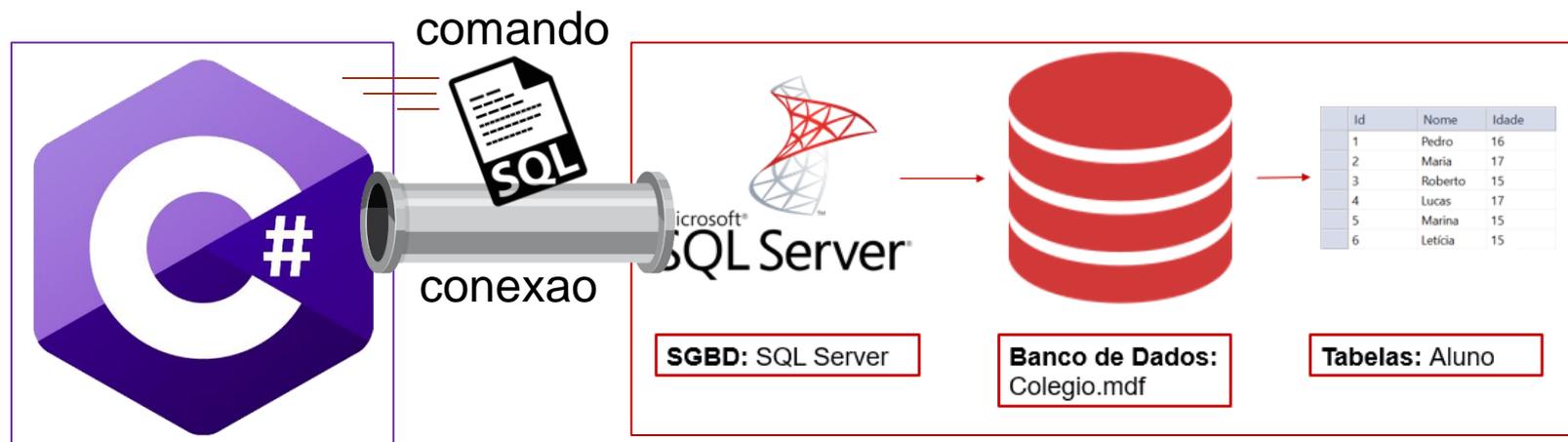
# SqlCommand: Enviando um comando



## 3. Comunicar sua aplicação com o banco de dados.

```
string consulta = "SELECT * FROM Aluno";
SqlCommand comando = new SqlCommand(consulta, conexao);
SqlDataReader leitor = comando.ExecuteReader();
```

Exemplo



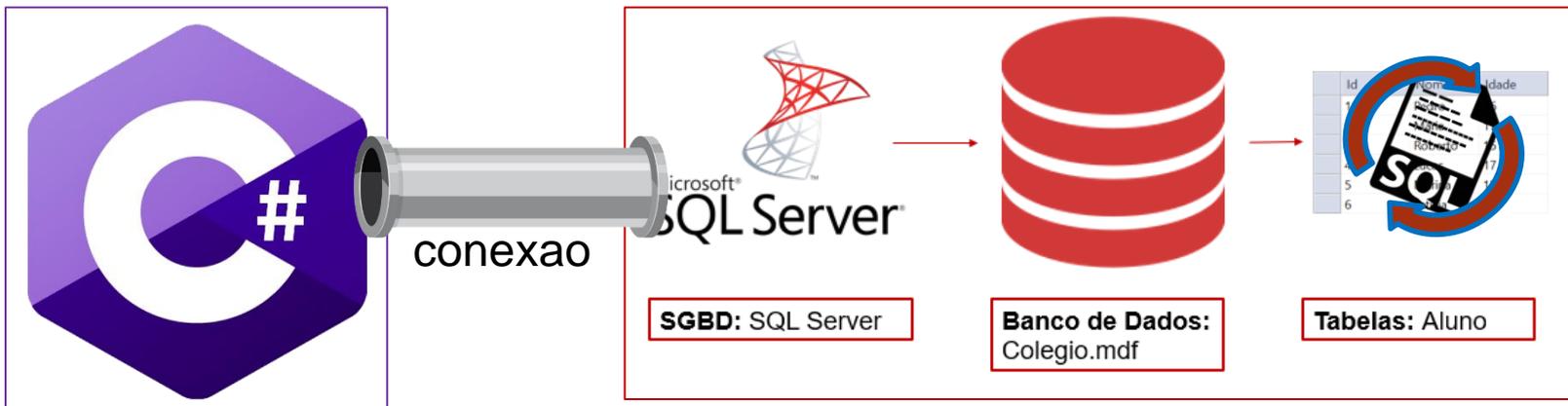
# SqlCommand: Enviando um comando



## 3. Comunicar sua aplicação com o banco de dados.

```
string consulta = "SELECT * FROM Aluno";
SqlCommand comando = new SqlCommand(consulta, conexao);
SqlDataReader leitor = comando.ExecuteReader();
```

Exemplo



# SqlCommand: Enviando um comando



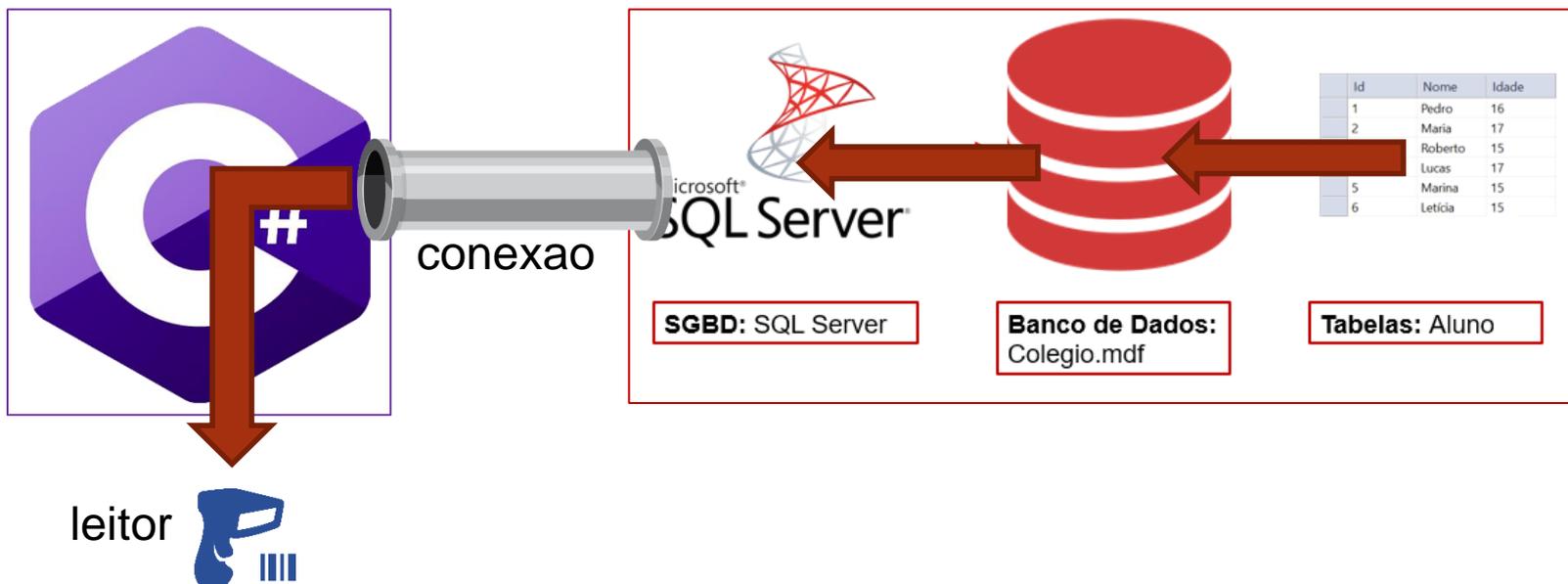
## 3. Comunicar sua aplicação com o banco de dados.

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

SqlCommand comando = new SqlCommand(consulta, conexao);

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

Exemplo



## SqlDataReader: Lendo a resposta do Banco de Dados



### 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



## 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



## 3. Comunicar sua aplicação com o banco de dados.

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



leitor



leitor["Id"]

leitor["Nome"]

leitor["Idade"]

	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



## 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

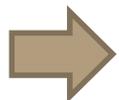


	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

# SqlDataReader: Lendo a resposta do Banco de Dados



## 3. Comunicar sua aplicação com o banco de dados.



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

leitor

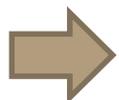


	Id	Nome	Idade
	1	Pedro	16
	2	Maria	17
	3	Roberto	15
	4	Lucas	17
	5	Marina	15
	6	Letícia	15

# SqlDataReader: Lendo a resposta do Banco de Dados



## 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

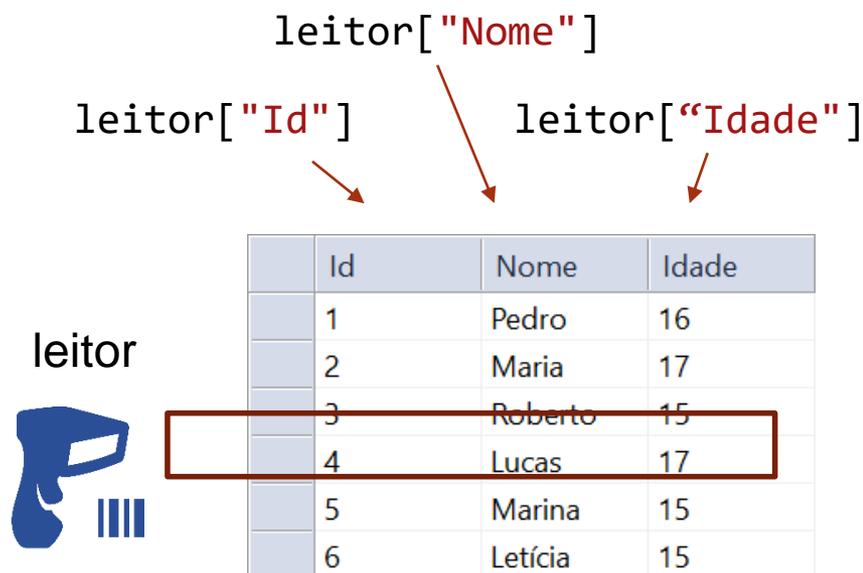
# SqlDataReader: Lendo a resposta do Banco de Dados



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



```
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



```
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



```
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(1)
    }

    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



```
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(1)
    }

    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



```
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(1)
    }

    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



```
private void button1_Click(object sender, EventArgs e)
{
    string stringDeConexao
    Source=(LocalDB)\MSSQLLocalDB;
    Lagôas\source\repos\Exemplo1
    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();
```

```
}
```

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`.



# Passo a passo

## Resumo do código



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

```
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



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 sender, EventArgs e)
{
    string stringDeConexao = "Server=(LocalDB)\\MSSQLLocalDB;AttachDbFilename='C:\\Users\\Joao Lagôas\\source\\repos\\ExemploPratico\\ExemploPratico\\Colegio.mdf';Integrated 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



Por fim, feche o leitor e a conexão.

```
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();
}
```



# 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



```
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



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

```
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);
    }
}
```

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.