



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?



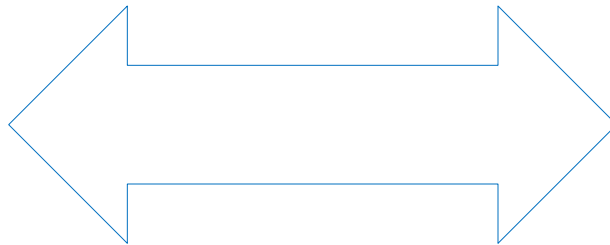
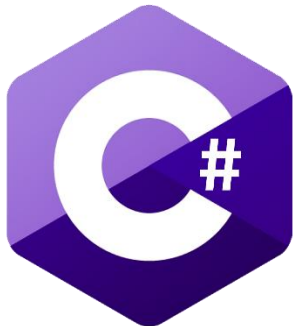
Colégio
Pedro II

- 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



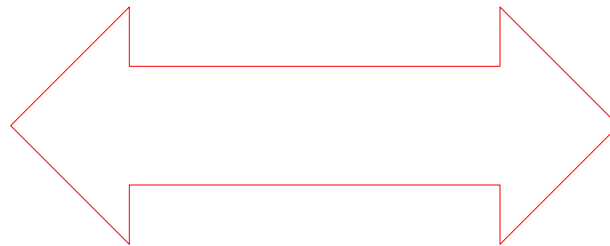
- Problemas:
 - Redundância não controlada e inconsistência de dados;
 - Isolamento dos dados;
 - Falta de integridade;
 - Anomalias de acesso concorrente;
 - Segurança;



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**.

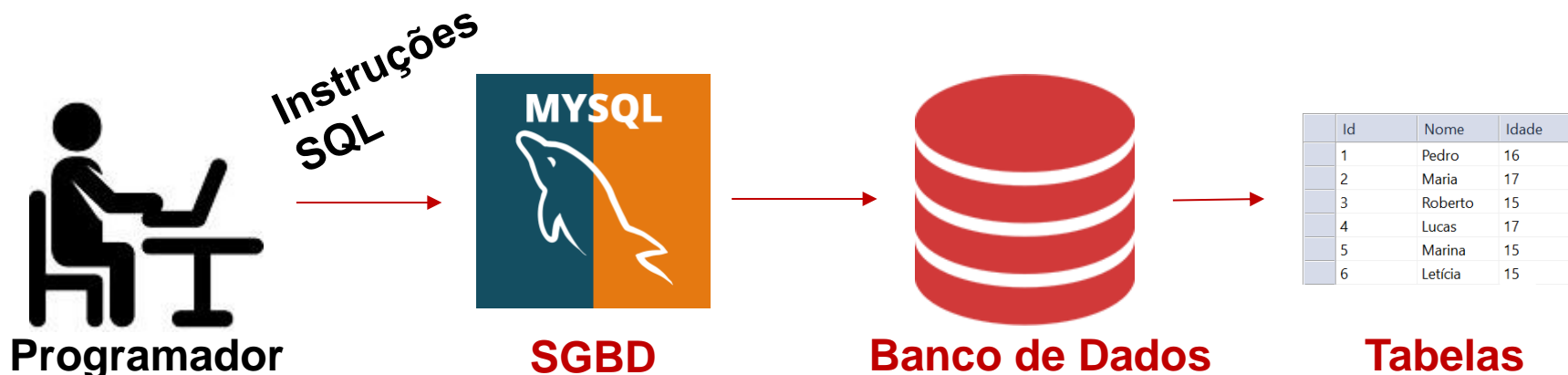


SGBD

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

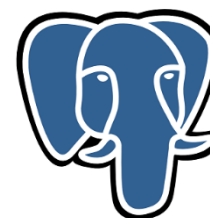
Sistema Gerenciador de Banco de Dados



- 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



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).

MySQL

SGBD mais acessível e abrangente



- Criado pela empresa sueca MySQL AB e atualmente mantido pela Oracle Corporation, o MySQL é um dos principais SGBDs relacionais do mercado, amplamente utilizado por aplicações web e empresariais.
- Interface gráfica leve no navegador:
 - O phpMyAdmin é uma ferramenta web escrita em PHP que fornece uma interface gráfica para gerenciar bancos de dados **MySQL**. Ele simplifica tarefas como criar tabelas, executar consultas SQL e fazer backups, sendo ideal para usuários que preferem evitar o terminal.

SQL

Linguagem de Consulta Estruturada

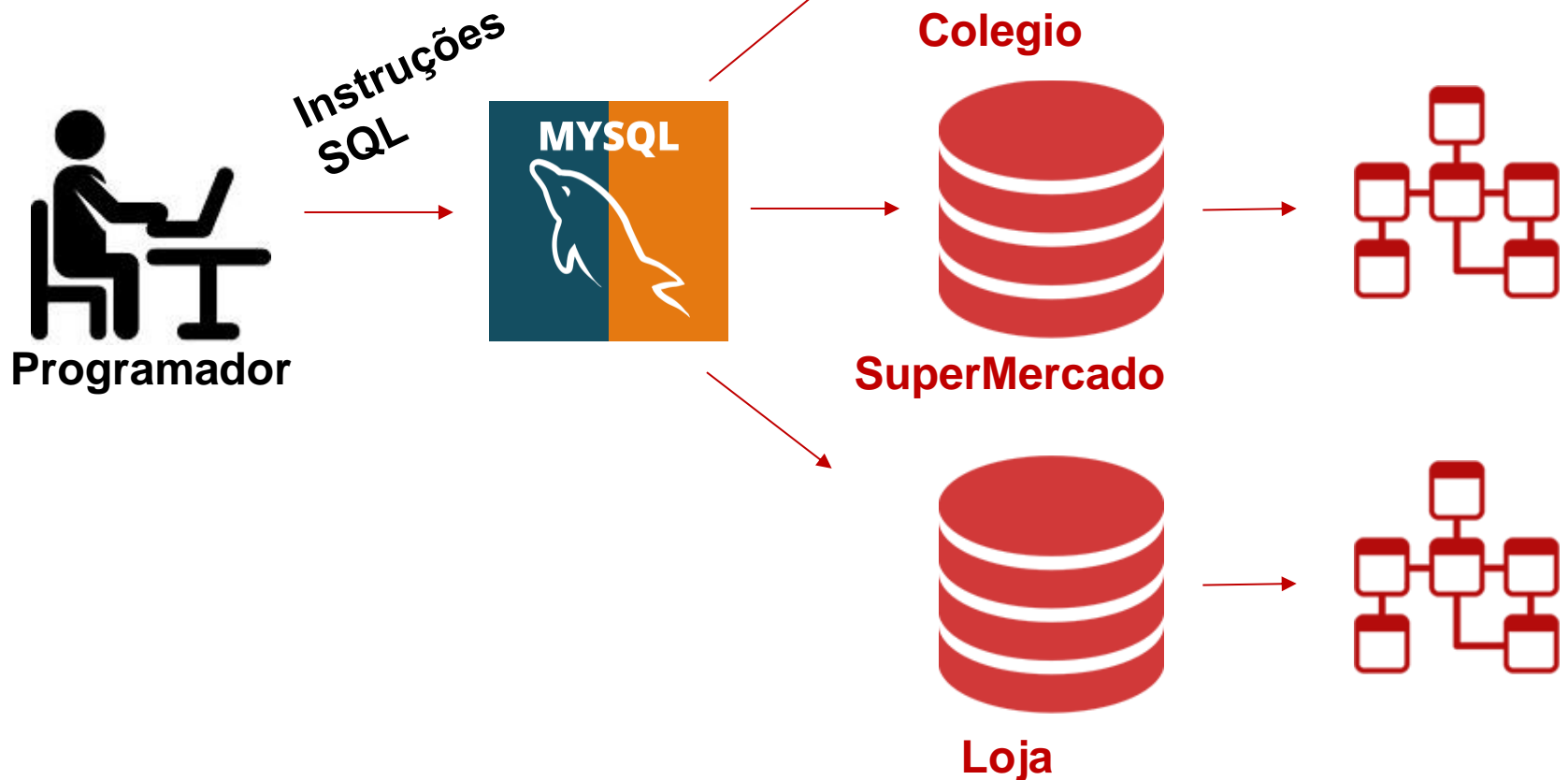


- “*Structured Query Language*” (SQL) é uma linguagem declarativa padronizada que está associada aos princípios de banco de dados no modelo relacional (modelo de tabelas).
- Os SGBD’s atualmente implementam SQL mas cada um deles pode apresentar certas particularidades. De qualquer forma, existe um grande 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

Comunicação

- Exemplo:



Sistema de Banco de Dados



Como se dá a comunicação (Teoria)

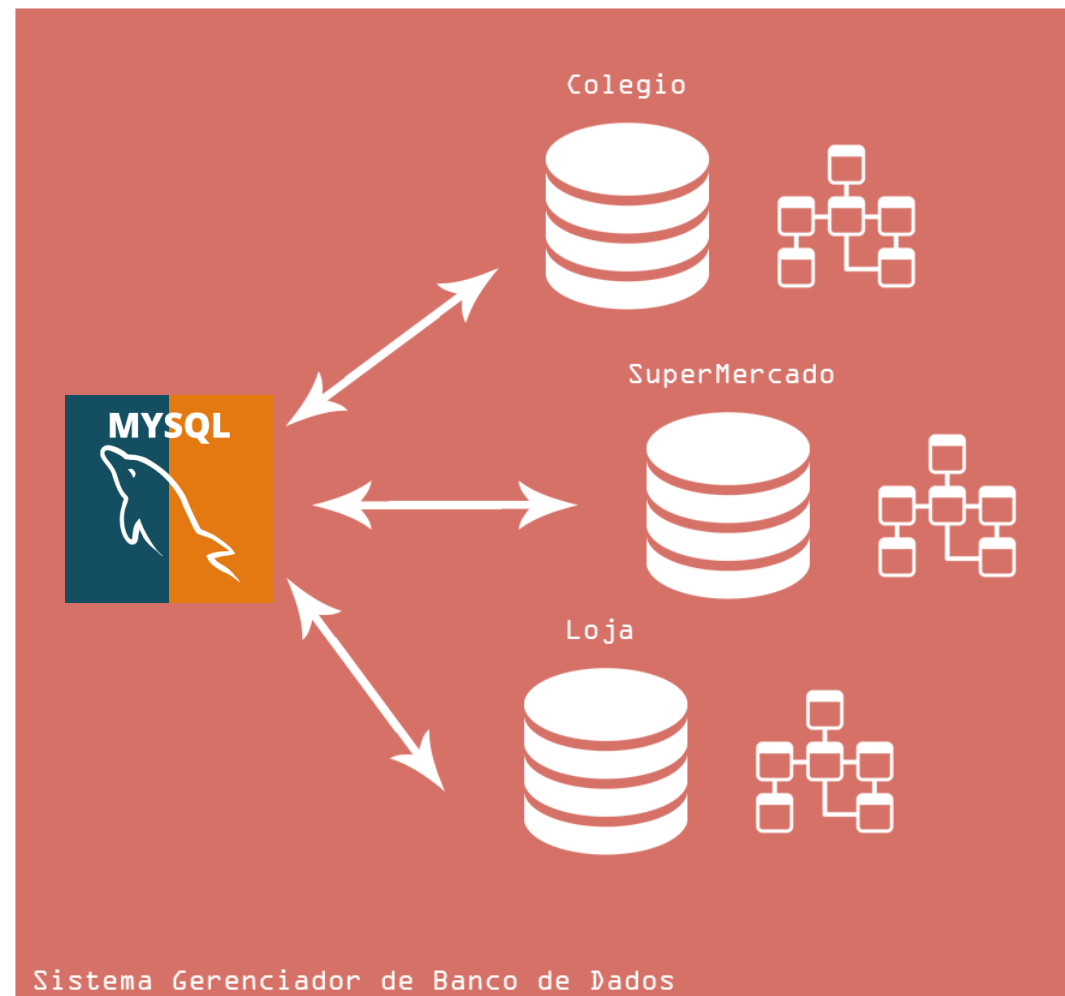
1. A sua aplicação deve se conectar com o Banco de Dados gerenciado por algum SGBD.
2. Comandos SQL são enviados através da conexão estabelecida no passo anterior para o SGBD.
3. O SGBD executa a consulta SQL e devolve para a aplicação uma resposta. Por exemplo: o resultado de uma consulta SELECT.
4.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

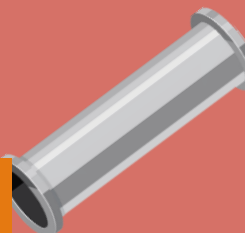


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)



1. Estabelecer conexão



Colegio



SuperMercado



Loja



Sistema de Banco de Dados

Como se dá a comunicação (Teoria)

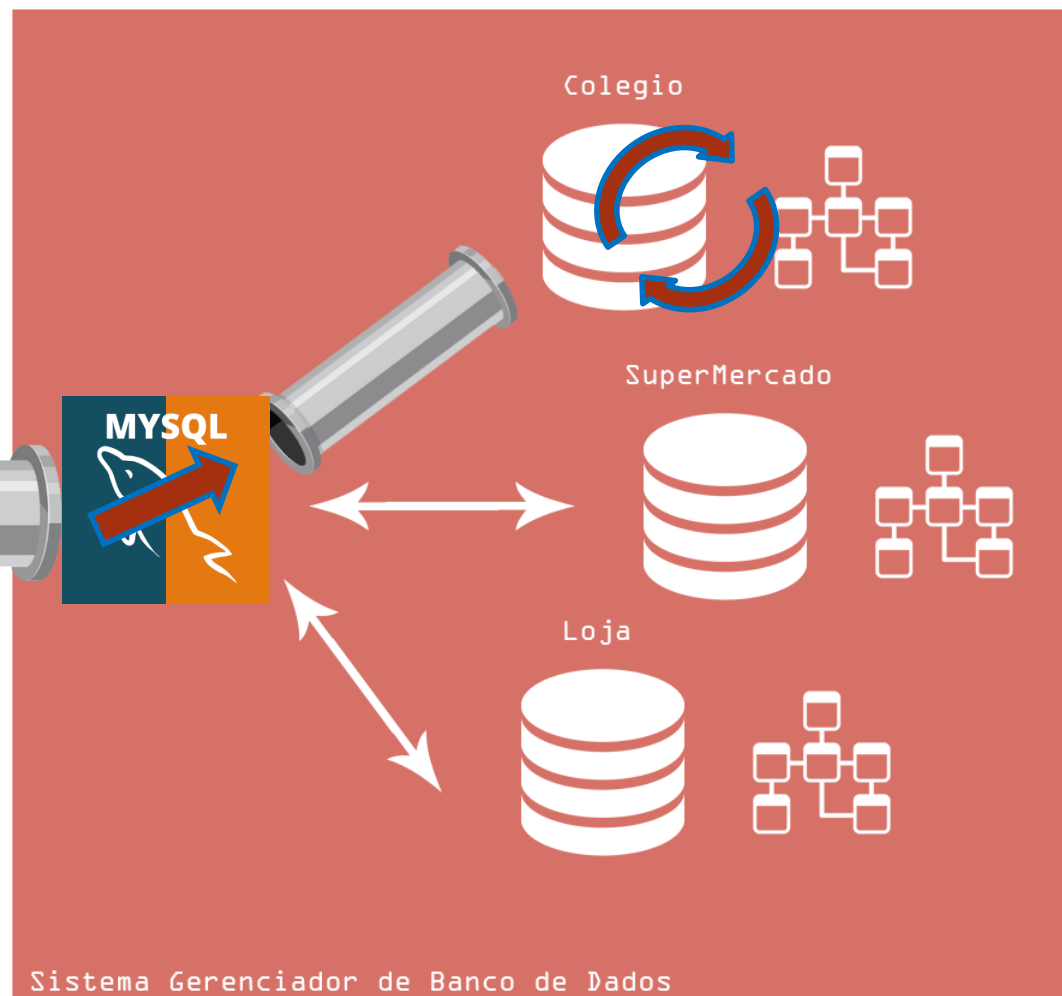


2. Enviar comandos SQL

```
SELECT * FROM Aluno
```



Programa

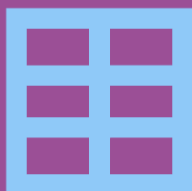


Sistema de Banco de Dados

Como se dá a comunicação (Teoria)



3. Responder consulta SQL



Programa



Sistema Gerenciador de Banco de Dados

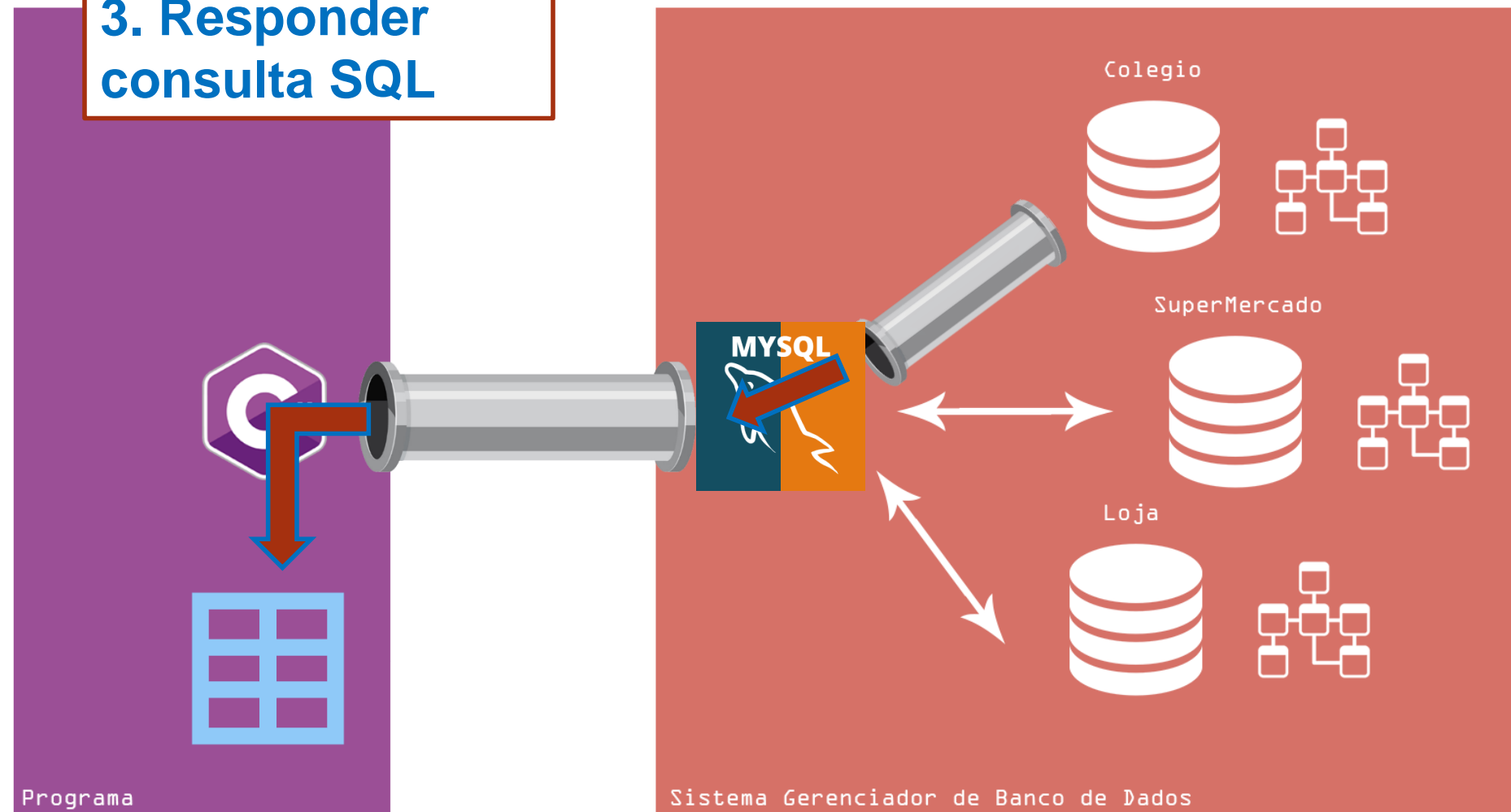
Colegio



SuperMercado



Loja



Sistema de Banco de Dados

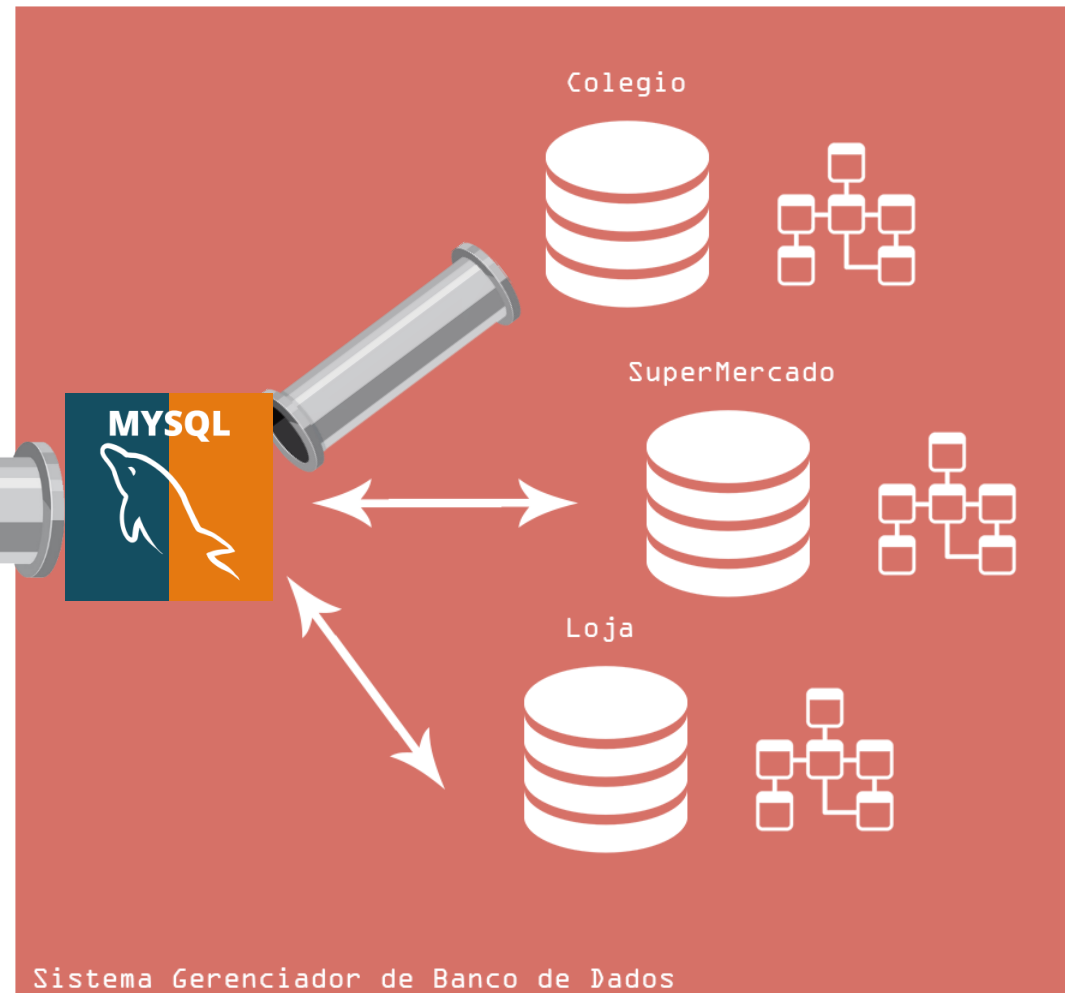
Como se dá a comunicação (Teoria)



4. Aplicação
utiliza a resposta



Programa



Sistema de Banco de Dados

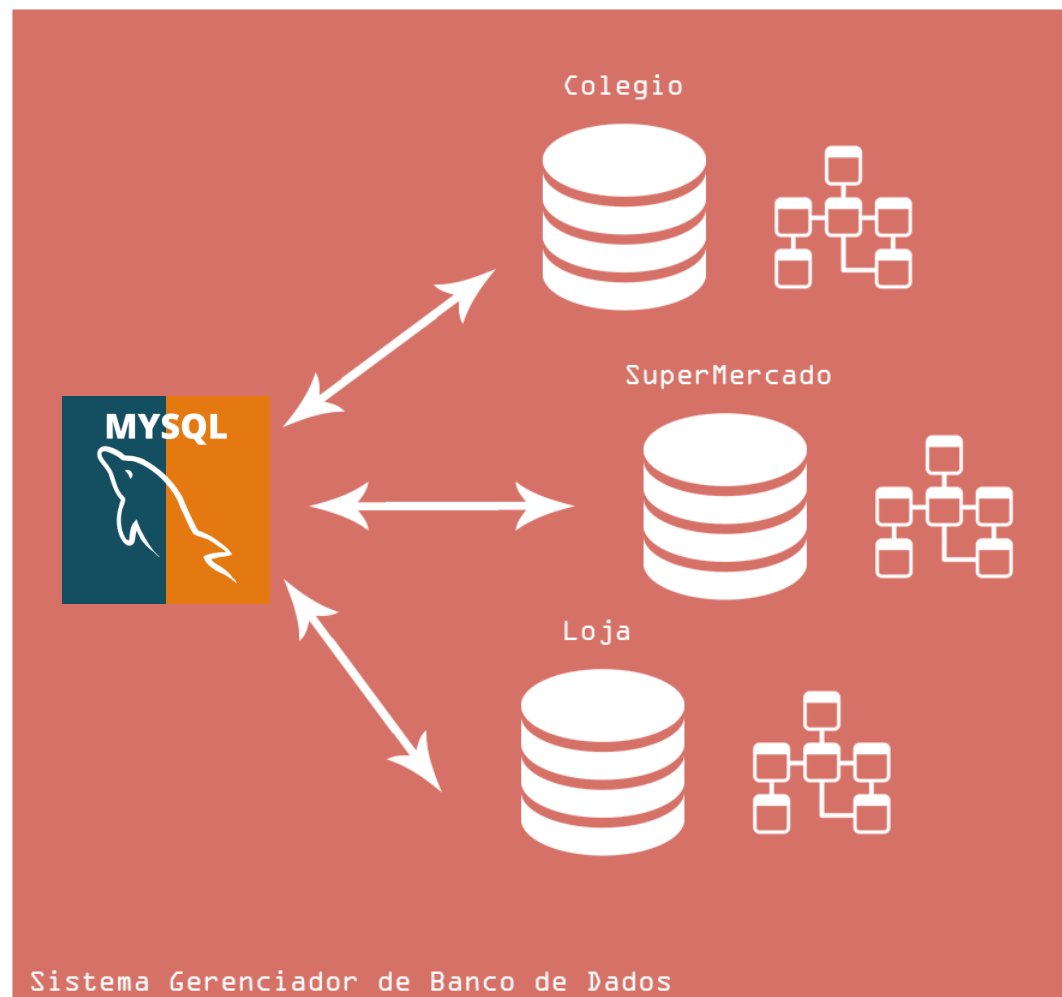
Como se dá a comunicação (Teoria)



5. Fechar conexão

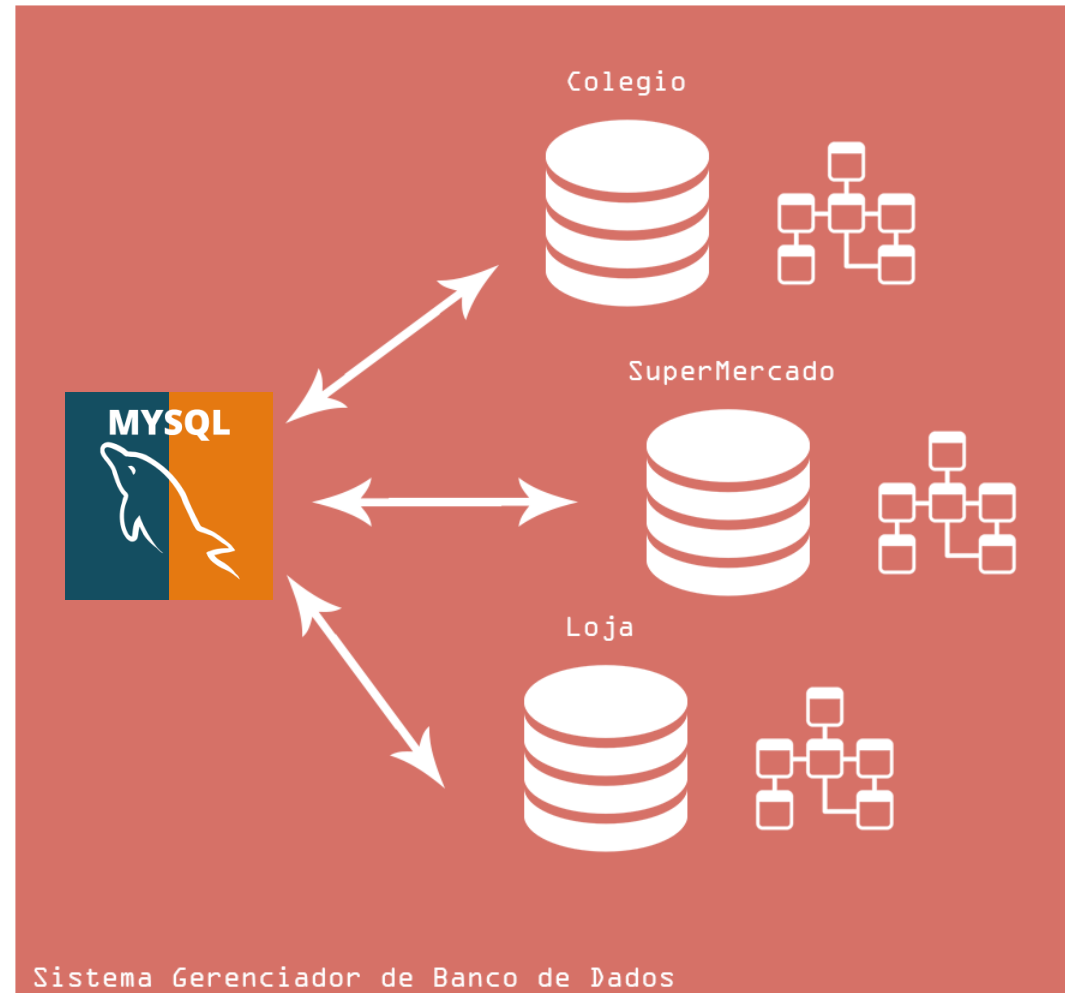


Programa



Sistema de Banco de Dados

Como se dá a comunicação (Teoria)



Como acessar o banco MySQL?



```
using MySql.Data.MySqlClient;
```

MySqlConnection

MySqlCommand

MySqlDataReader

- Dentro do namespace **MySql.Data.MySqlClient**, há uma variedade de classes que são utilizadas para se comunicar com um banco de dados MySQL.

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

Observação: Certifique-se de instalar o pacote **MySql.Data** através do NuGet caso não esteja disponível no seu projeto.

MySqlConnection: Realizando a conexão



Comunicar sua aplicação com o banco de dados.

- A classe MySqlConnection é responsável por realizar a conexão entre a aplicação C# e o Banco de Dados MySQL.
- No seu construtor, ela espera receber uma string de conexão que contém todas as informações necessárias para acessar o banco de dados. O valor de retorno é um objeto que representa a conexão.
- Um objeto MySqlConnection 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.

MySqlConnection: Realizando a conexão

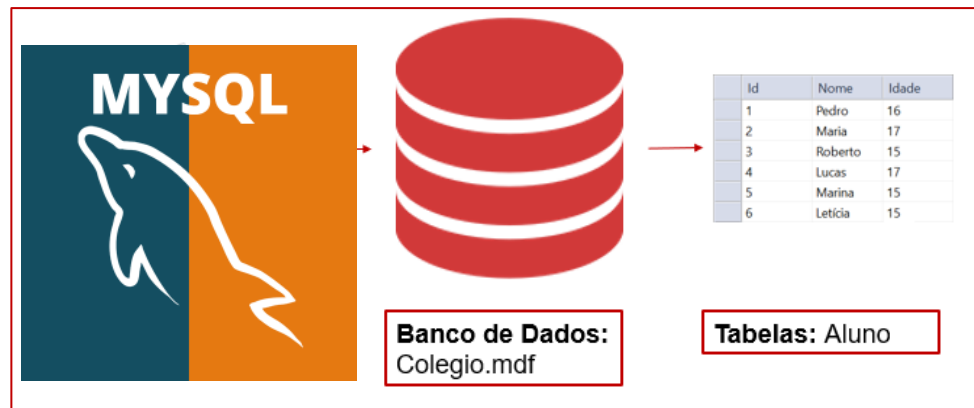
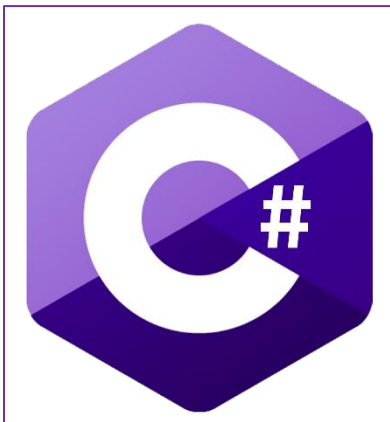
Comunicar sua aplicação com o banco de dados.



```
string connectionString =
"Server=localhost;Database=colegio;Uid=root;Pwd=";

using (var connection = new MySqlConnection(connectionString))
{
    connection.Open();
}
```

Exemplo



MySqlConnection: Realizando a conexão

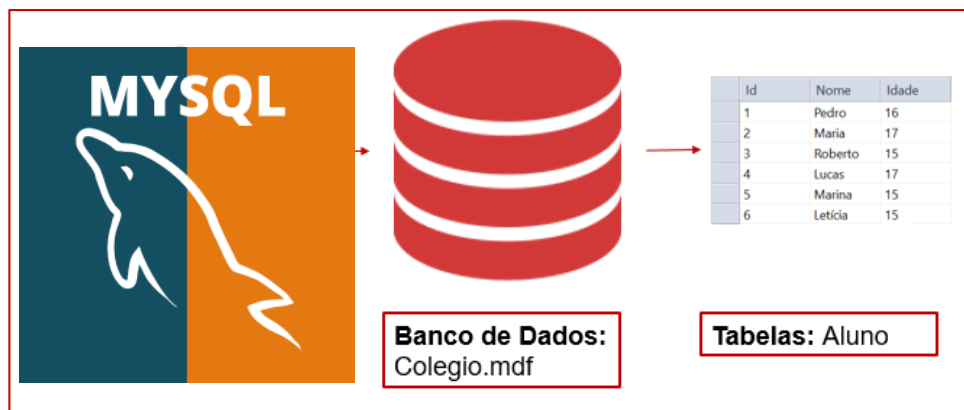
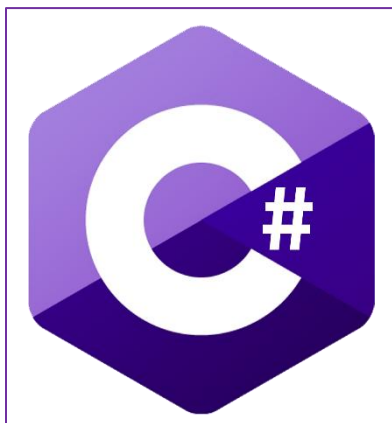
Comunicar sua aplicação com o banco de dados.



```
string connectionString =
"Server=localhost;Database=colegio;Uid=root;Pwd=";

using (var connection = new MySqlConnection(connectionString))
{
    connection.Open();
}
```

Exemplo



MySqlConnection: Realizando a conexão

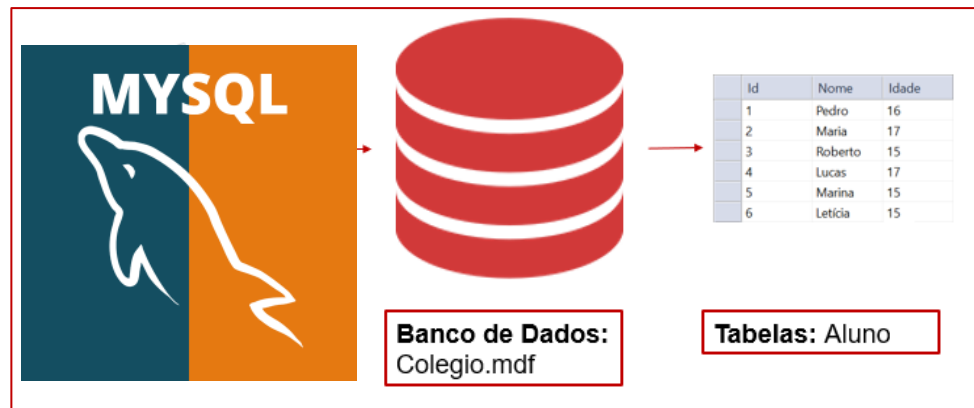
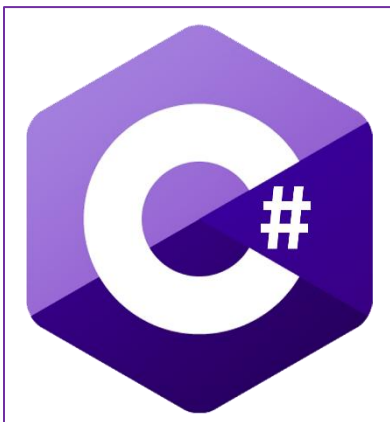
Comunicar sua aplicação com o banco de dados.



```
string connectionString =
"Server=localhost;Database=colegio;Uid=root;Pwd=";

using (var connection = new MySqlConnection(connectionString))
{
    connection.Open();
}
```

Exemplo



MySqlConnection: Realizando a conexão

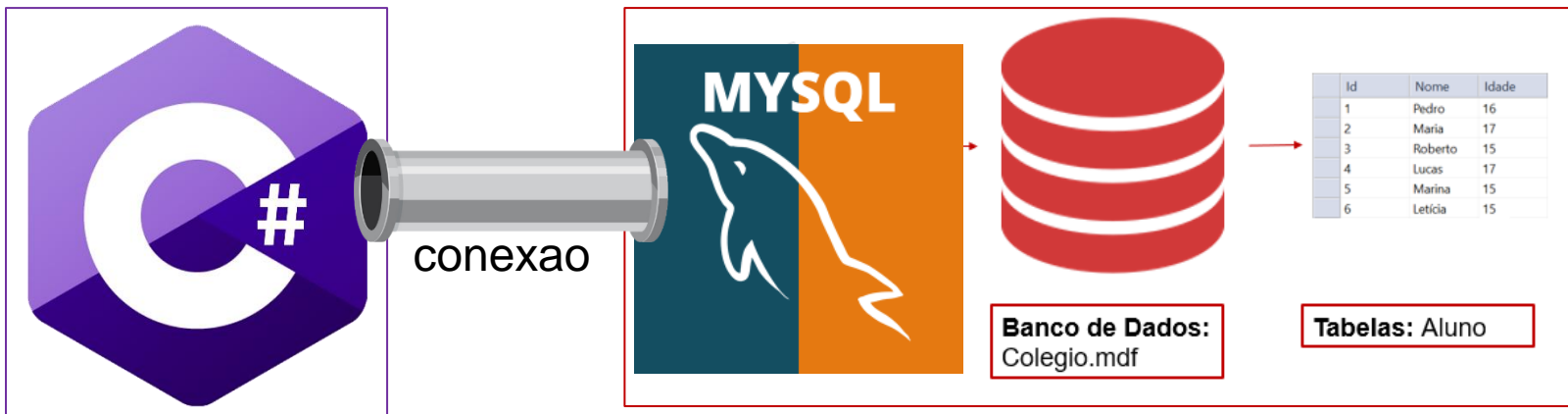
Comunicar sua aplicação com o banco de dados.



```
string connectionString =
"Server=localhost;Database=colegio;Uid=root;Pwd=";

using (var connection = new MySqlConnection(connectionString))
{
    connection.Open();
}
```

Exemplo



MySqlCommand: Enviando um comando



Comunicar sua aplicação com o banco de dados.

- A classe MySqlCommand é responsável por codificar uma instrução SQL e enviá-la, através de um objeto MySqlConnection, para o banco de dados.
- O construtor de MySqlCommand espera receber dois parâmetros: uma string contendo uma instrução SQL e um objeto MySqlConnection para que a aplicação saiba para onde ele deve enviar a instrução SQL.
- Um objeto MySqlCommand 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 MySqlDataReader, que nada mais é que uma abstração de uma resposta SQL (uma tabela).

MySqlCommand: Enviando um comando

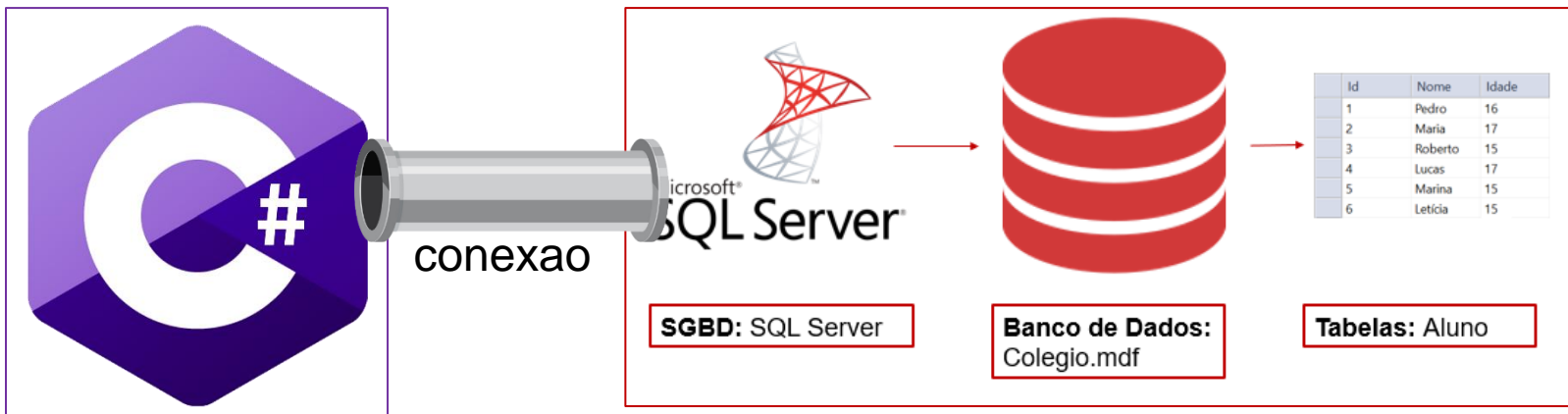
Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



MySqlCommand: Enviando um comando

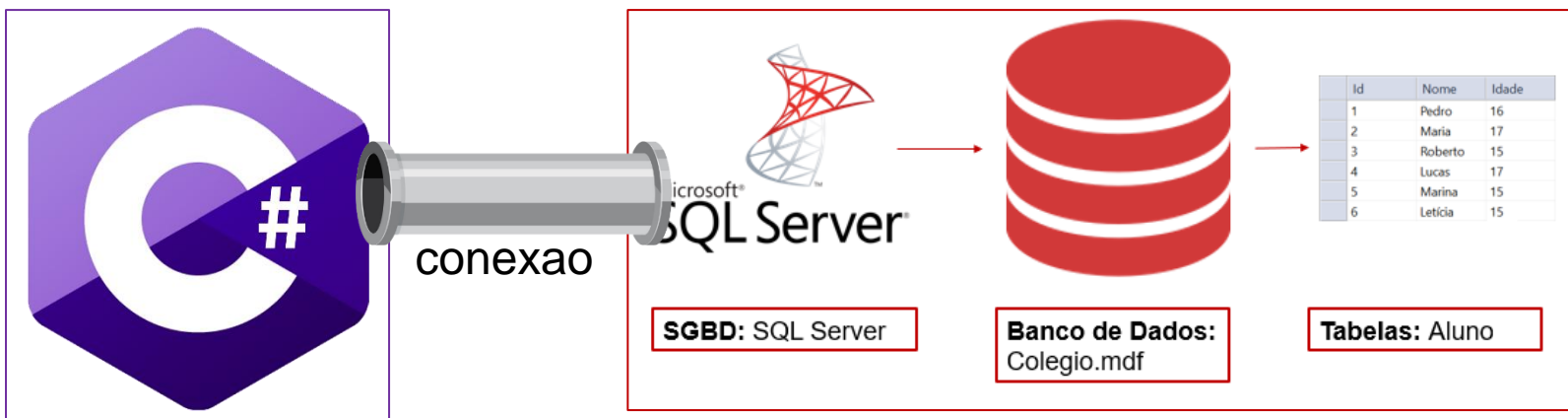
Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



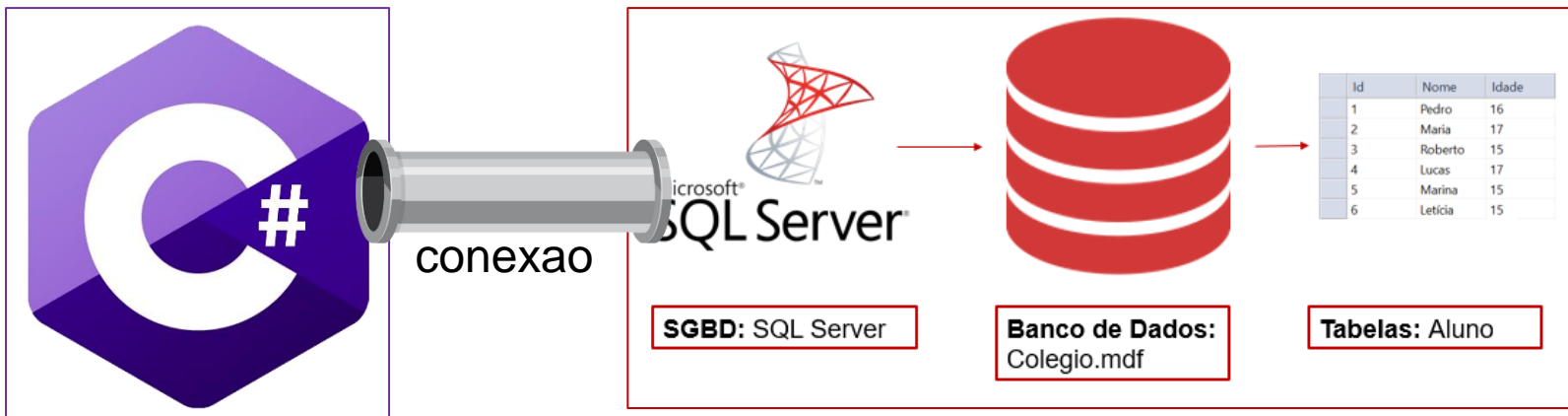
MySqlCommand: Enviando um comando

Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";
using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



MySqlCommand: Enviando um comando

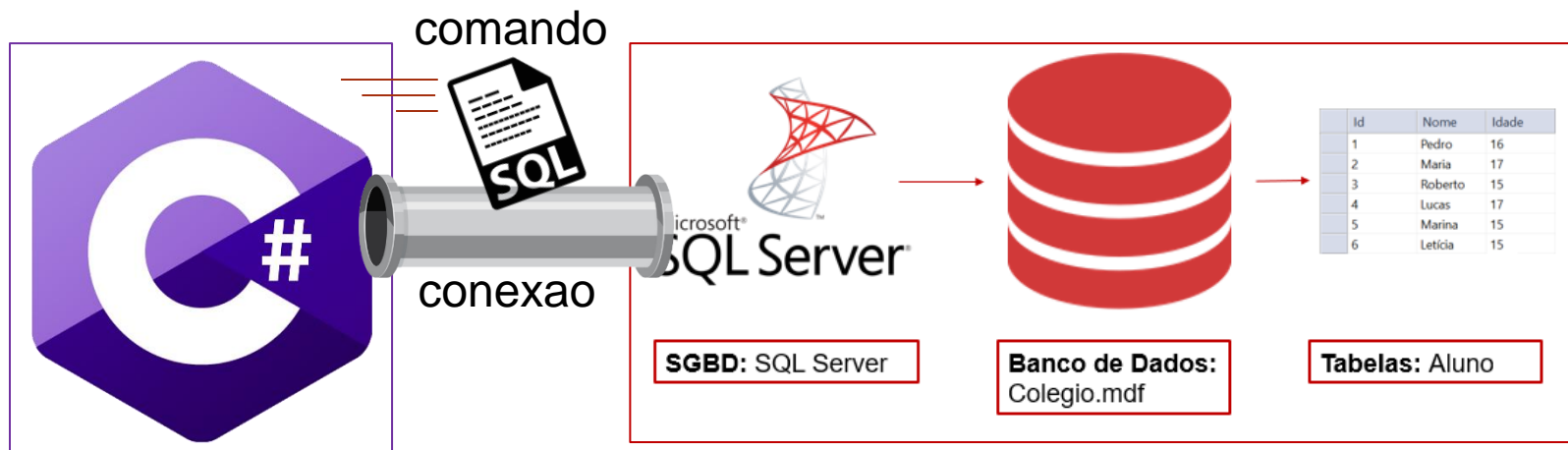
Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



MySqlCommand: Enviando um comando

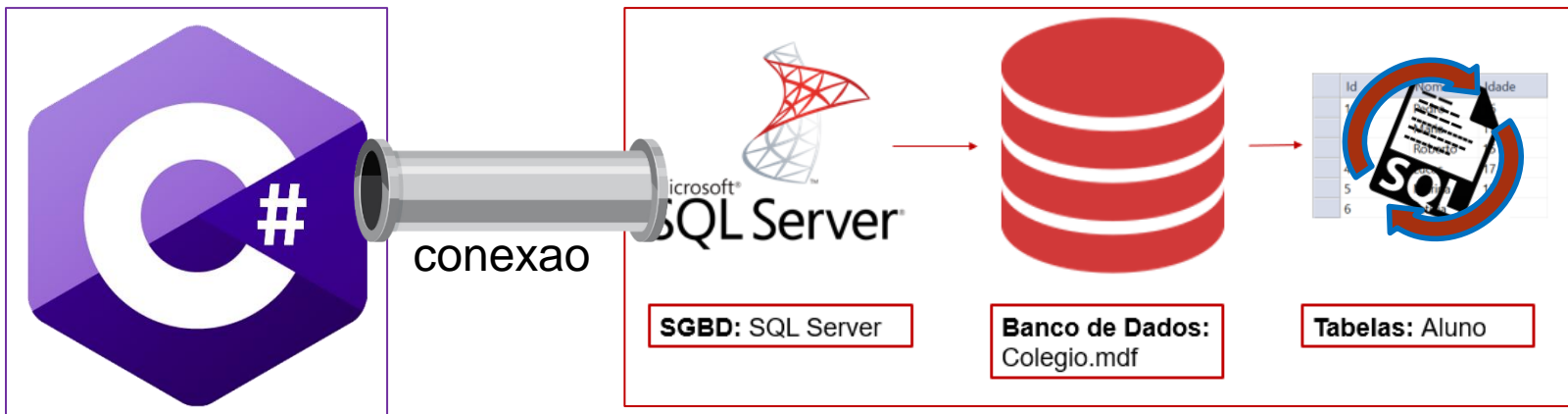
Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



MySqlCommand: Enviando um comando

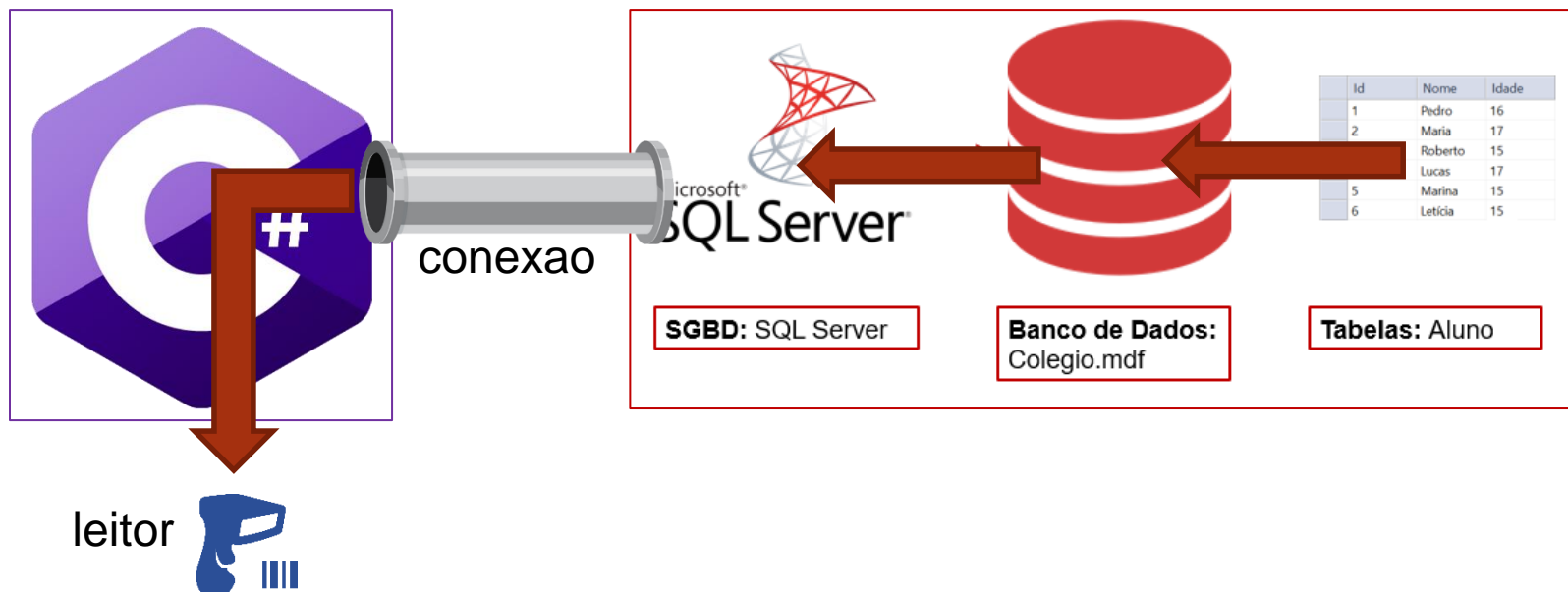
Comunicar sua aplicação com o banco de dados.



```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
```

Exemplo



MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

- A classe `MySqlDataReader` 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 `MySqlDataReader` para se criar uma instância. O método `ExecuteReader()` de um objeto `MySqlCommand` retorna uma instância de `SqlDataReader` de acordo com a instrução SQL que foi executada.
- Um objeto `MySqlDataReader` 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.

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())  
{  
    // Lê os resultados da consulta  
    while (reader.Read())  
    {  
        string nome = reader.GetString("Nome");  
        Console.WriteLine(nome);  
    }  
}
```

leitor



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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("Nome");
        Console.WriteLine(nome);
    }
}
```

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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())  
{  
    // Lê os resultados da consulta  
    while (reader.Read())  
    {  
        string nome = reader.GetString("Nome");  
        Console.WriteLine(nome);  
    }  
}
```

leitor



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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("Nome");
        Console.WriteLine(nome);
    }
}
```

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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())  
{  
    // Lê os resultados da consulta  
    while (reader.Read())  
    {  
        string nome = reader.GetString("Nome");  
        Console.WriteLine(nome);  
    }  
}
```

leitor



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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("Nome");
        Console.WriteLine(nome);
    }
}
```

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

MySqlDataReader: Lendo a resposta do Banco de Dados



Colégio
Pedro II

Comunicar sua aplicação com o banco de dados.

```
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("Nome");
        Console.WriteLine(nome);
    }
}
```

leitor["Nome"]

leitor["Id"]

leitor["Idade"]

leitor



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

O leitor 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



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd="";  
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

Passo a passo

Resumo do código



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd=";
```

```
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

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

Passo a passo

Resumo do código



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd=";
```

→

```
using (var connection = new MySqlConnection(connectionString))
{
    connection.Open();

    // Consulta SQL para buscar os dados
    string query = "SELECT * FROM colegio";

    using (var command = new MySqlCommand(query, connection))
    using (var reader = command.ExecuteReader())
    {
        // Lê os resultados da consulta
        while (reader.Read())
        {
            string nome = reader.GetString("nome");
            Console.WriteLine(nome);
        }
    }
}
```


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

Passo a passo

Resumo do código



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd=";
```

 `connection.Open();`

```
// Consulta SQL para buscar os dados
string query = "SELECT * FROM colegio";

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("nome");
        Console.WriteLine(nome);
    }
}
```

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

Passo a passo

Resumo do código



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd=";
```

```
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

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

Passo a passo

Resumo do código



```
string connectionString = "Server=lo
using (var connection = new MySqlConnection
{
    connection.Open();

    // Consulta SQL para buscar os d
    string query = "SELECT * FROM co

using (var command = new MySqlCommand(query, connection))
using (var reader = command.ExecuteReader())
{
    // Lê os resultados da consulta
    while (reader.Read())
    {
        string nome = reader.GetString("nome");
        Console.WriteLine(nome);
    }
}
}
```

O método `ExecuteReader()` do objeto comando retorna um objeto preparado da classe `MySqlDataReader`. 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.

```
string connectionString = "Server=localhost;Database=colegio;User=root;Password=";  
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

Passo a passo

Resumo do código



```
string connectionString = "Server=lo
using (var connection = new MySqlConnection
{
    connection.Open();

    // Consulta SQL para buscar os dados
    string query = "SELECT * FROM colegio";

    using (var command = new MySqlCommand(query, connection))
    using (var reader = command.ExecuteReader())
    {
        // Lê os resultados da consulta
        while (reader.Read())
        {
            string nome = reader.GetString("nome");
            Console.WriteLine(nome);
        }
    }
}
```

Para acessar um elemento da linha onde o leitor está posicionado, basta indicar o nome da coluna e o tipo de dado que se deseja obter.



Passo a passo

Resumo do código



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd="";  
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

Como estamos usando a construção using, não é necessário liberar os recursos alocados.

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 `MySqlConnection` e o método `ExecuteReader()` de um `MySqlCommand`, podem lançar exceções facilmente. Basta a conexão de string estar errada ou a consulta SQL estar errada, por exemplo.

Banco de Dados e Exceções



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd=";
```

```
using (var connection = new MySqlConnection(connectionString))  
{  
    connection.Open();  
  
    // Consulta SQL para buscar os dados  
    string query = "SELECT * FROM colegio";  
  
    using (var command = new MySqlCommand(query, connection))  
    using (var reader = command.ExecuteReader())  
    {  
        // Lê os resultados da consulta  
        while (reader.Read())  
        {  
            string nome = reader.GetString("nome");  
            Console.WriteLine(nome);  
        }  
    }  
}
```

Código de risco



É importante preparar seu código para lidar com esses métodos.

Banco de Dados e Exceções



```
string connectionString = "Server=localhost;Database=colegio;Uid=root;Pwd="";

try
{
    using (var connection = new MySqlConnection(connectionString))
    {
        connection.Open();

        // Consulta SQL para buscar os dados
        string query = "SELECT * FROM colegio";

        using (var command = new MySqlCommand(query, connection))
        using (var reader = command.ExecuteReader())
        {
            // Lê os resultados da consulta
            while (reader.Read())
            {
                string nome = reader.GetString("nome");
                Console.WriteLine(nome);
            }
        }
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Erro: {ex.Message}");
}
```

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.