

# MySQL - Um Exemplo de Aplicação


Mário Leite

...

Dentre os dez maiores bancos de dados relacionais o MySQL ocupa a segunda posição; talvez por ser muito simples de ser configurado ou pelo fato de ser *open e free*. A verdade é que este SGBD é amplamente usado pelos programadores; e em aplicações para Web é, com certeza, o preferido. É um sistema de gerenciamento de banco de dados relacionais de código aberto e gratuito, cujo nome “MySQL” foi dado em homenagem à uma das filhas de um dos seus criadores: o finlandês Michael Widenius. Mais tarde ele criou, também, o banco MariaDB, em homenagem à outra filha: Maria. MySQL é um *software* gratuito sob os termos da GNU General Public License e também está disponível sob uma variedade de licenças proprietárias, e tem clientes autônomos que permite aos usuários interagirem diretamente com um banco de dados usando instruções e recursos da linguagem SQL. Frequentemente é usado com outros ambientes para implementar aplicativos que usam bancos de dados relacionais. Neste exemplo vamos usar um banco do servidor MySQL chamado “Locadora”, cujas tabelas foram concebidas a partir das seguintes entidades relacionadas:

- ✓ **Genero**      Dados sobre o gênero de um filme (*código e descrição: drama, terror, ação, etc*).
- ✓ **Filme**        Dados sobre um filme (*código, título, código-gênero, código-diretor*).
- ✓ **Ator**         Dados sobre os atores (*código, nome, nacionalidade*).
- ✓ **Diretor**      Dados sobre os diretores (*código, nome, nacionalidade*).
- ✓ **Participacao**   Dados sobre os tipos de participação (*código e descrição da participação*).
- ✓ **AtorFilme**    Dados sobre os atores nos filmes (*código-ator, código-filme, código-participação*).

Podemos extrair dados de várias tabelas do banco “Locadora”; e para isto vamos considerar o MER (Modelo Entidade-Relacionamento) da **figura 1**. O objetivo é mostrar os filmes com seus respectivos *gêneros e diretores* num *DataGridView*, fazendo os relacionamentos com as equi-junções entre as tabelas que contenham os dados necessários. Observe, nesta figura, que apesar de nem o campo “Gen\_desc” e nem o campo “Dir\_nome” estarem da tabela “Filmes”, é perfeitamente possível exibir os filmes com a “descrição do gênero” e o “nome do diretor” através do relacionamento entre estas tabelas. A instrução *sql* que torna isto possível faz os relacionamentos necessários usando o termo INNER JOIN, conforme mostrado no código do programa “PrExtraiFilmes” em C# no IDE do Visual Studio 2022,

Sobre o MER da **figura 1**: aqui foi empregada a notação “pé de galinha”  nos relacionamentos porque fica mais fácil mostrar as respectivas cardinalidades, fugindo daquela notação empregada nos Diagramas de Classes em sistemas Orientados a Objetos. Na notação aqui utilizada a entidade que contém o terminador “pé de galinha” é a entidade fraca; aquela que está relacionada com a entidade que contém a “chave primária” (entidade forte). É o caso, por exemplo, do relacionamento entre “Gênero” e “Filme”: para cadastrar um filme ele tem que ter um gênero; isto é, um *gênero* está relacionado a vários *filmes*, mas, um *filme* está relacionado a apenas um *gênero*; em outras palavras, a entidade fraca depende de outra entidade para existir. A **figura 2** mostra o banco “Locadora” no do **MySQL-Front**; um IDE muito utilizado para criar e gerenciar bancos de dados MySQL. A **figura 3** mostra a aplicação em tempo de projeto (*design*) e a **figura 4** em tempo de execução (*running*). Nesta última figura são exibidos todos os filmes em ordem alfabética do título em Português, com a descrição do gênero e nome do diretor extraídos das tabelas “Gêneros” e “Diretores”. Em seguida é apresentado o código da aplicação relacionado com o evento *\_click()* dos botões [Executar] e [Fechar]. E apenas para esclarecer aos programadores, quatro instâncias (objetos) foram criadas no código:

- objConn**    Objeto Connection: define a conexão com o banco de dados.
  - objAdpt**    Objeto Adapter: preenche um *DataSet* (cache de dados na RAM) e atualiza o banco de dados.
  - objTab**     Objeto Table: representa uma tabela, em memória, da classe *DataTable*.
  - objCmd**    Objeto Command: usado como uma instância de um comando que executa a instrução SQL.
-

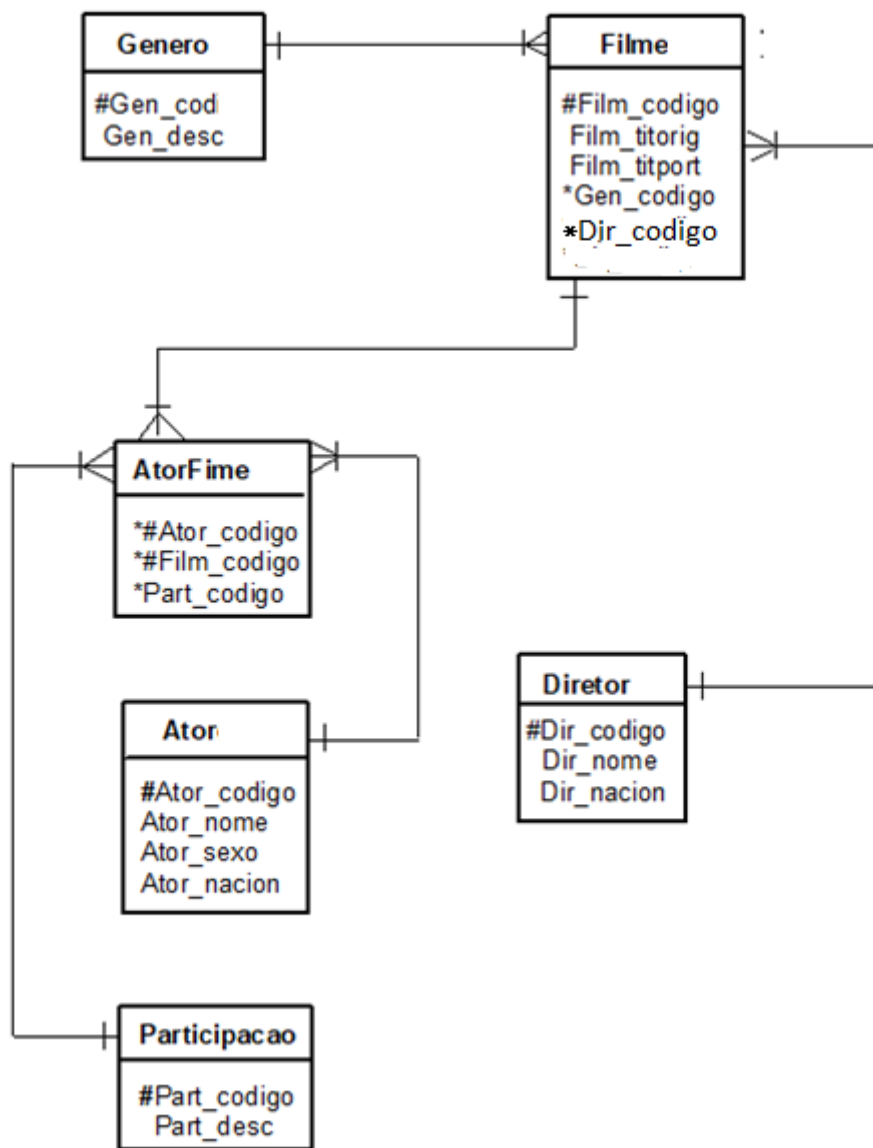


Figura 1 - MER do banco de dados "Locadora"

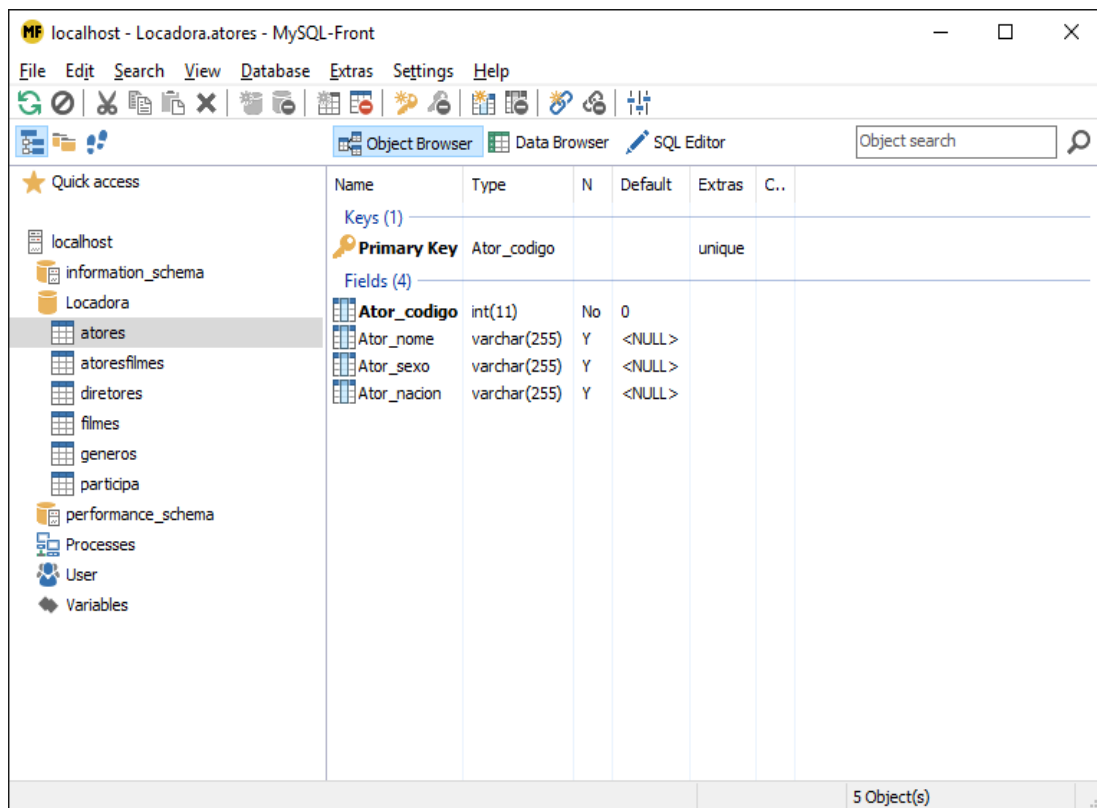


Figura 2 - O Banco "Locadora" no IDE no MySQL-Front

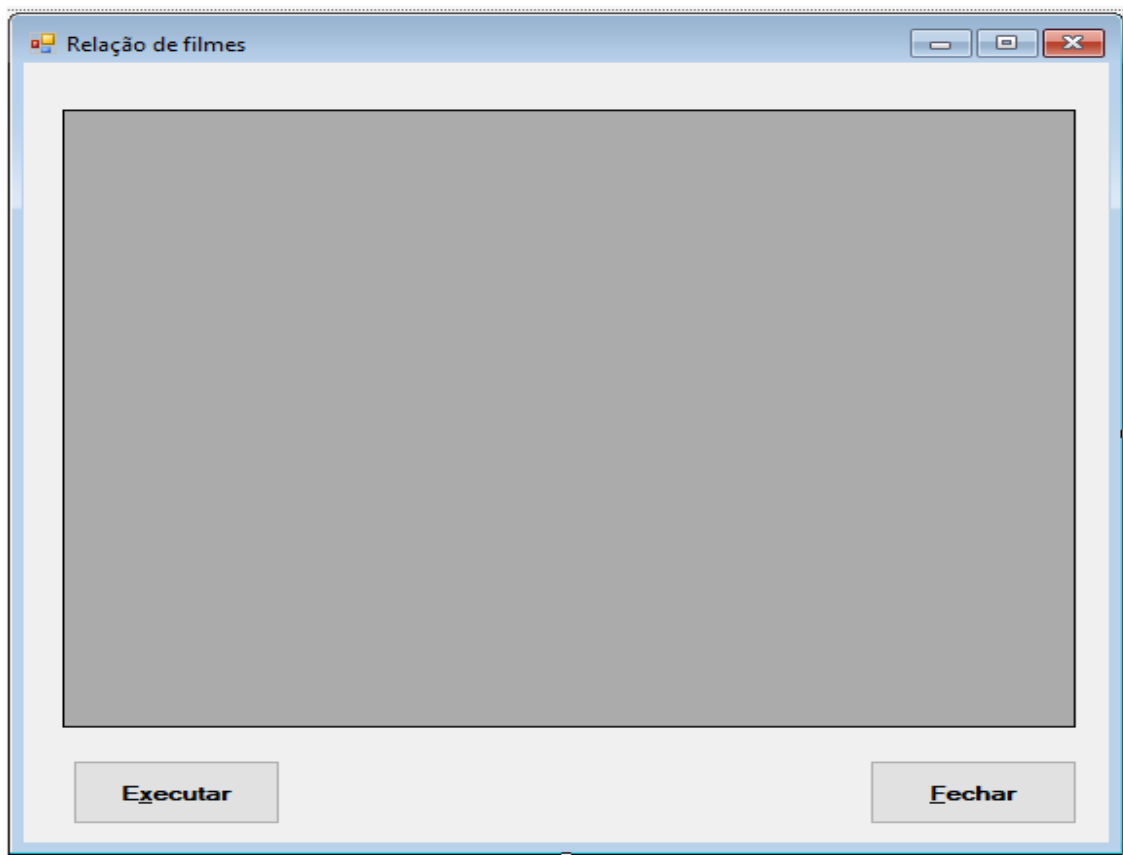


Figura 3 - Interface da aplicação em tempo de projeto

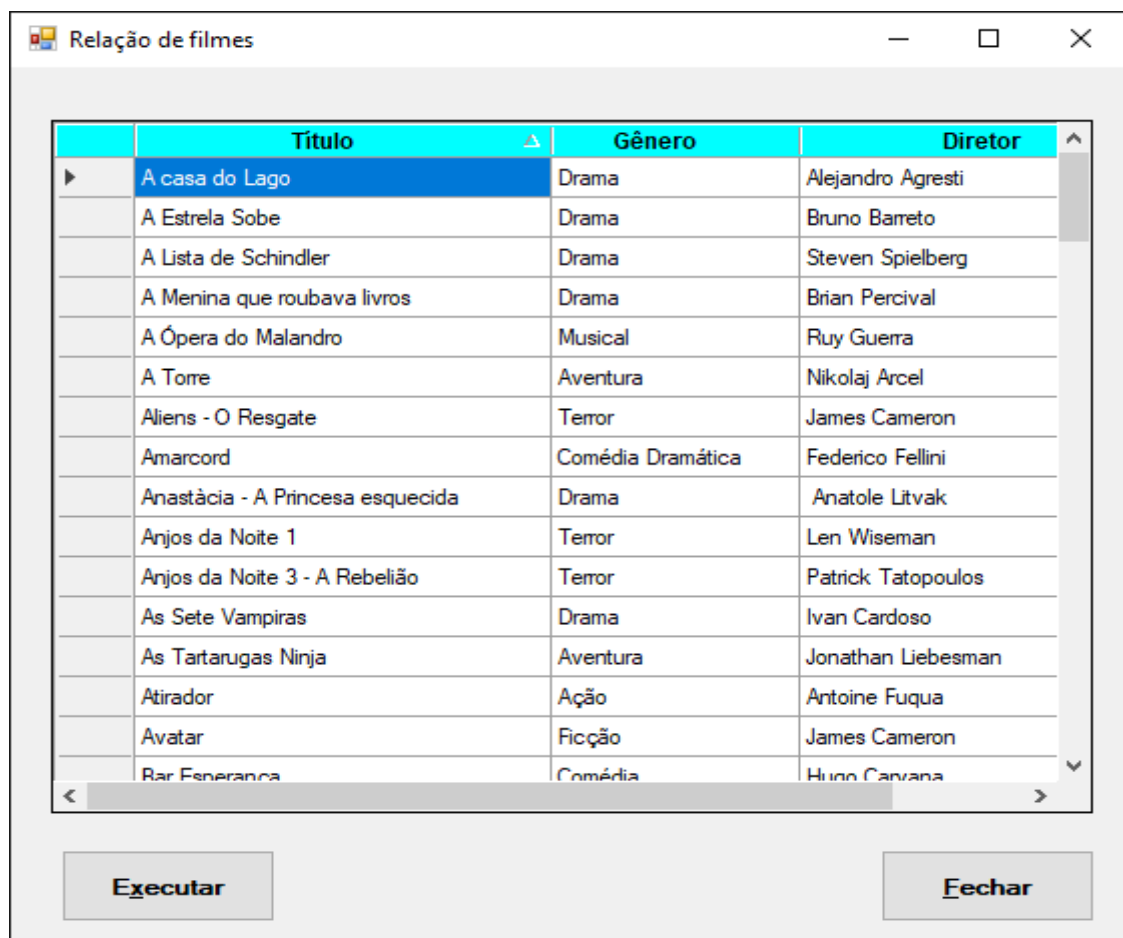


Figura 4 - Interface da aplicação em tempo de execução

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient; //assembly que contém a biblioteca de conexão com MySql

namespace PrExtraiFilmes
public partial class frmFilmes : Form
{
    string strCon = "Server=localhost;Database=Locadora;Uid=root;Pwd=mTe112358@@#";
    public frmFilmes() {
        InitializeComponent();
    }
    //-----
    private void btnExecutar_Click(object sender, EventArgs e)
    {
        MySqlConnection objCon = new MySqlConnection(strCon);
        try {
            objCon.Open(); //faz a conexão: tenta abrir o banco de dados
            //Extrai os dados através de instrução sql
            string strSql =
                "SELECT Filmes.Film_titulo,Generos.Gen_desc,Diretores.Dir_nome" +
                " FROM Filmes" +
                " INNER JOIN Diretores ON"
                "     Diretores.Dir_codigo=Filmes.Dir_codigo " +
                " INNER JOIN Generos ON Generos.Gen_cod=Filmes.Gen_cod";
            objCon.Close(); //fecha a conexão com a base de dados
            //Cria objetos para conter os dados
            MySqlCommand objCmd = new MySqlCommand(strSql, objCon);
            MySqlDataAdapter objAdp = new MySqlDataAdapter(objCmd);
            DataTable objTab = new DataTable();
            objAdp.Fill(objTab); //preenche objeto Adapter

            //Altera os nomes das colunas do objeto Tabela no grid
            objTab.Columns[0].ColumnName = "Título";
            objTab.Columns[1].ColumnName = "Gênero";
            objTab.Columns[2].ColumnName = "Diretor";
            dgrFilmes.DataSource = objTab; //preenche o DataGridView

            //Ordena os nomes dos filmes no DataGridView em ordem alfabética
            dgrFilmes.Sort(dgrFilmes.Columns[0], ListSortDirection.Ascending);

            //Altera estilo e cor dos títulos (headers) das colunas
            dgrFilmes.EnableHeadersVisualStyles = false;
            //Desabilita padrões
            dgrFilmes.ColumnHeadersDefaultCellStyle.Font = new
                Font("Arial", 9, FontStyle.Bold);
            dgrFilmes.ColumnHeadersDefaultCellStyle.BackColor = Color.Aqua;
            dgrFilmes.ColumnHeadersDefaultCellStyle.ForeColor= Color.Black;
            dgrFilmes.ColumnHeadersDefaultCellStyle.Alignment =
                DataGridViewContentAlignment.MiddleCenter
        }
        catch (Exception){
            MessageBox.Show("Falha na conexão!");
            return; //sai da rotina
        }
    }
    //-----
    private void btnFechar_Click(object sender, EventArgs e)
    {
        this.Close(); //fecha o formulário e encerra a aplicação
    }
} //fim da aplicação

```