



CCM310

# Arquitetura de Software e Programação Orientada a Objetos

Profa. Dra. Gabriela Biondi

Prof. Dr. Isaac Jesus

Prof. Dr. Luciano Rossi

Banco de Dados

# Banco de Dados

- São **coleções organizadas de dados** que se **relacionam** de forma a criar algum sentido e dar mais eficiência durante uma pesquisa

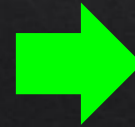
# Banco de Dados

- Antigamente as empresas armazenavam informações em **arquivos físicos**



# Banco de Dados

- O surgimento e a evolução dos computadores possibilitaram o **armazenamento de dados de modo digital**



# Banco de Dados

- Os Banco de Dados (*Database - DB*) são de **vital importância para empresas** e se tornaram uma das principais peças dos sistemas de informação



# Banco de Dados

Banco de Dados trata do problema de armazenamento e como recuperar eficientemente uma informação quando ela for necessária

# Principais Modelos de Banco de Dados

**Relacional**

**Não relacional**



# Modelos de Banco de Dados

- **Relacional**

- Um banco de dados relacional organiza os dados em tabelas ou relações
- As linhas da tabela são chamadas de registros ou tuplas
- As colunas são chamadas de campo ou atributo
- SQL (*Structured Query Language*) Linguagem de Consulta Estruturada

# Modelos de Banco de Dados

- Relacional

Produtos			← Entidade
Código	Descrição	Valor	← Atributos
01	Teclado	110,00	← Tupla
02	Monitor	430,00	
03	Mouse	45,00	

# Modelos de Banco de Dados

- **Relacional - Limitações**

- O modelo relacional foi desenvolvido em torno de 1970 por Edgar F. Codd (pesquisador da IBM)
- Contudo, a quantidade de informações que precisam ser armazenadas tem crescido muito (*principalmente em aplicações Web*) e os bancos relacionais podem sofrer com problemas de *escalabilidade\**

*\*estar preparado para crescer / funcionar da mesma forma com mais trabalho*

# Modelos de Banco de Dados

- **Não Relacional**

- A principal motivação para banco de dados não relacionais é a busca para **resolução do problema de escalabilidade**
- São modelados de **formas diferentes** das relações tabulares usadas nos bancos de dados relacionais
- **NoSQL** (*Not only SQL*)

# Sistemas de Gerenciamento de Banco de Dados (SGBD) (SGBDs)

- Um **SGBD** (*Data Base Management System - DBMS*) é o **conjunto de softwares** responsáveis pelo **gerenciamento de um banco de dados**
- O **SGBD** disponibiliza uma interface para que seus clientes possam **incluir, alterar, consultar** ou **remover** dados previamente armazenados - **CRUD** (*Create, Update, Read, Delete*)

# Sistemas de Gerenciamento de Banco de Dados (SGBDs)

## Relacionais



## Oracle Database



## Não Relacionais





# SGBDs Ranking

Rank			DBMS	Database Model	Score		
May 2022	Apr 2022	May 2021			May 2022	Apr 2022	May 2021
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1262.82	+8.00	-7.12
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1202.10	-2.06	-34.28
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	941.20	+2.74	-51.46
4.	4.	4.	PostgreSQL + 💬	Relational, Multi-model ⓘ	615.29	+0.83	+56.04
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	478.24	-5.14	-2.78
6.	6.	↑ 7.	Redis +	Key-value, Multi-model ⓘ	179.02	+1.41	+16.85
7.	↑ 8.	↓ 6.	IBM Db2	Relational, Multi-model ⓘ	160.32	-0.13	-6.34
8.	↓ 7.	8.	Elasticsearch +	Search engine, Multi-model ⓘ	157.69	-3.14	+2.34
9.	9.	↑ 10.	Microsoft Access	Relational	143.44	+0.66	+28.04
10.	10.	↓ 9.	SQLite +	Relational	134.73	+1.94	+8.04

# SQL

- **Structured Query Language**
  - Linguagem de Consulta Estruturada
  - SQL é uma **linguagem declarativa**, pois nela define-se o que deve ser retornado como resultado do processamento, sem especificar o como isso será feito

# SQL

- **Structured Query Language**
  - Alguns comandos:
    - **CREATE TABLE** - criar nova tabela

# SQL

- **Structured Query Language**

- Alguns comandos - CRUD:

- **INSERT INTO** <tabela> **VALUES** <valores>
- **SELECT** <atributos> **FROM** <tabela> **WHERE** <condições>
- **UPDATE** <tabela> **SET** <atributos e valores> **WHERE** <condições>
- **DELETE FROM** <tabela> **WHERE** <condições>

# SQL

- **Structured Query Language**
  - Vamos testar:
    - <https://sqliteonline.com/>

GUI + JDBC + PostgreSQL



# Instalando o PostgreSQL



[Home](#) [About](#) [Download](#) [Documentation](#) [Community](#) [Developers](#) [Support](#) [Donate](#) [Your account](#)



20th May 2021: [PostgreSQL 14 Beta 1 Released!](#)

<https://www.postgresql.org/>

PostgreSQL: The World's Most Advanced Open Source  
Relational Database

Download →

New to PostgreSQL?



[New to PostgreSQL?](#)



[Latest Releases](#)

# Criando um novo Banco de Dados - PostgreSQL

- pgAdmin 4:

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows a tree structure with 'Servers (1)' expanded to 'PostgreSQL 13', which further expands to 'Databases (3)'. A context menu is open over the 'aluno' database, with the 'Create' option selected. This opens a 'Database...' dialog box. An orange arrow points from the 'Database...' dialog box to the 'CREATE DATABASE' button in the main toolbar. The main pane shows a SQL editor with the following query:

```
1 SELECT * FROM public.alunos
2 ORDER BY usuario ASC
```

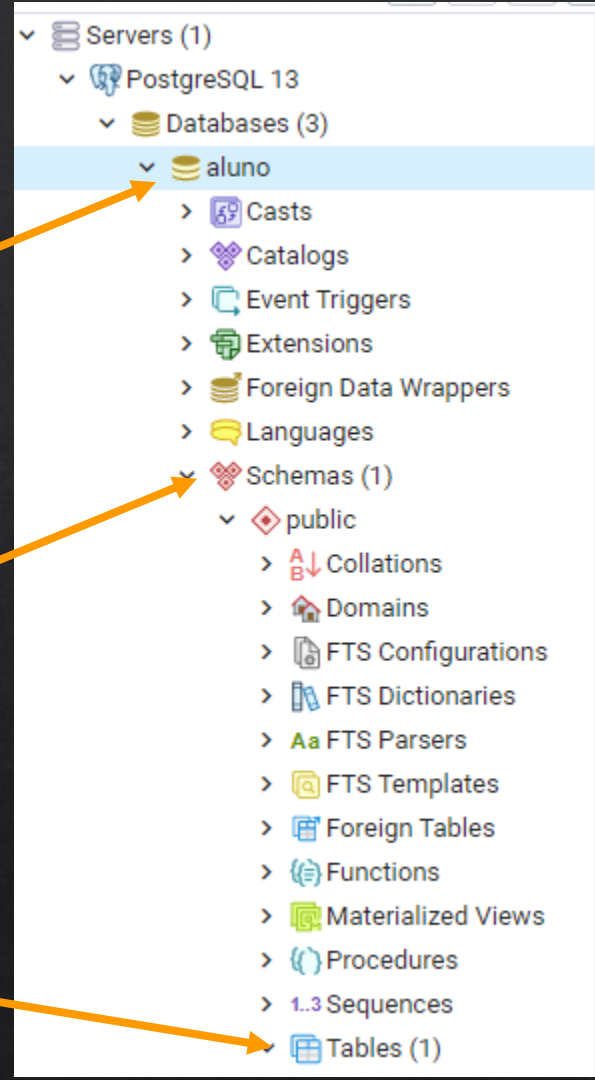
Below the SQL editor, the 'Data Output' tab is active, displaying a table with 8 rows of data. The table has three columns: 'nome', 'usuario', and 'senha'.

	nome	usuario	senha
1	Beltrano	ubeltrano	111111
2	Danilo	udanilo	123456789
3	Fulano ...	user_fulano	123456
4	Sicrano	usicrano	222222
5	Teste	uteste	99999
6	Teste11	uteste11	123456789
7	Teste2	uteste2	987
8	Teste2	uteste22	987

# Criando uma Tabela - PostgreSQL

- pgAdmin 4:

Clicar com o botão direito em  
Tables e Create Table



# Criando uma Tabela - PostgreSQL

Clicar em *columns* para cadastrar os campos

- pgAdmin 4:

Nome da tabela

The screenshot shows the 'Create - Table' dialog in pgAdmin 4. The 'General' tab is active. The 'Name' field is filled with 'tabela\_nova'. The 'Owner' is set to 'postgres' and the 'Schema' is 'public'. The 'Tablespace' dropdown shows 'Select an item...'. The 'Partitioned table?' checkbox is unchecked. The 'Comment' field is empty. At the bottom, there are buttons for 'Cancel', 'Reset', and 'Save'. An orange arrow points from the 'Columns' tab to the 'Name' field, and another orange arrow points from the 'Name' field to the 'Columns' tab.

# Criando uma Tabela - PostgreSQL

- pgAdmin 4:

Nome do campo

Tipo de dados

Não pode ser  
nulo



é chave primária

Create - Table

General Columns Advanced Constraints Partitions Parameters Security SQL

Inherited from table(s)

Columns +

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
 	nome	text			<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No

# Tabela - PostgreSQL

- pgAdmin 4: Como ver todos os dados da tabela? Clicar com o botão direito em cima da tabela e:

*View/Edit Data:*

*All Rows: lista tudo que está na tabela*

The screenshot shows the pgAdmin 4 interface. The 'Browser' pane on the left shows a tree structure with 'public' expanded, and 'alunos' selected under 'Tables'. A right-click context menu is open over the 'alunos' table. The menu options are: Create, Refresh..., Count Rows, Delete/Drop, Drop Cascade, Reset Statistics, Import/Export..., Maintenance..., Scripts, Truncate, Backup..., Restore..., View/Edit Data, Search Objects..., Query Tool..., and Properties... The 'View/Edit Data' option is highlighted, and its sub-menu is open, showing: All Rows, First 100 Rows, Last 100 Rows, and Filtered Rows... Two orange arrows point from the text on the left to the 'View/Edit Data' and 'All Rows' options in the menu.

Browser

public

- > Collate
- > Domain
- > FTS Config
- > FTS Dict
- > FTS Template
- > Foreign Data Wrapper
- > Function
- > Materialized View
- > Procedure
- > 1.3 Sequence
- > Table
- > Table Space
- > Trigger
- > Trigger Functions
- > Types
- > Views

Create

- Refresh...
- Count Rows
- Delete/Drop
- Drop Cascade
- Reset Statistics
- Import/Export...
- Maintenance...
- Scripts
- Truncate
- Backup...
- Restore...
- View/Edit Data
- Search Objects...
- Query Tool...
- Properties...

View/Edit Data

- All Rows
- First 100 Rows
- Last 100 Rows
- Filtered Rows...

Properties

Properties

SQL

State

alunos

Query History

```
SELECT * FROM public.alunos
ORDER BY usuario ASC
```

Explain

Messages

	usuario [PK] text	senha text
1	ubeltrano	111111

6	Teste11	uteste11	123456789
7	Teste2	uteste2	987
8	Teste2	uteste22	987

26

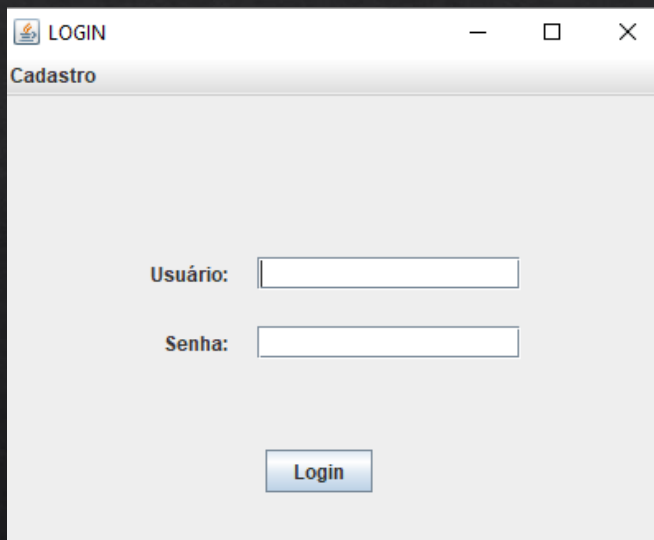


# Exemplo em Aula:

## Sistema de Login e Cadastro de Alunos

# Package - view

- Sistema de login e **cadastro** de alunos
  - 2 telas:



The screenshot shows a window titled "LOGIN" with a subtitle "Cadastro". It contains two input fields: "Usuário:" and "Senha:". Below the fields is a button labeled "Login".

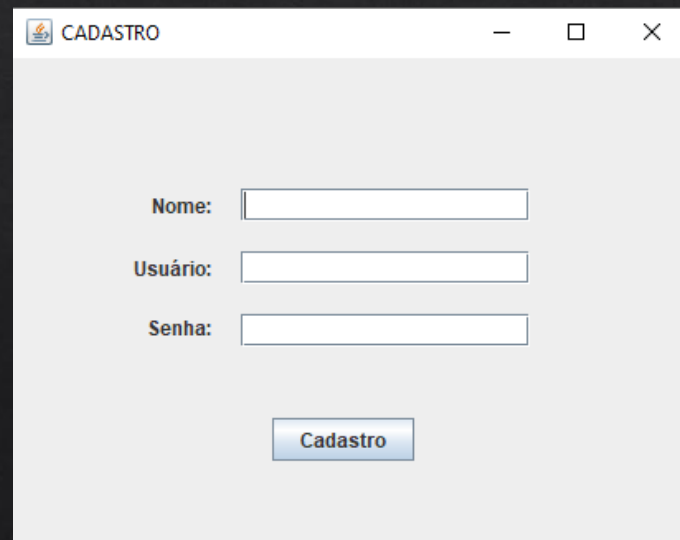
LOGIN

Cadastro

Usuário:

Senha:

Login



The screenshot shows a window titled "CADASTRO". It contains three input fields: "Nome:", "Usuário:", and "Senha:". Below the fields is a button labeled "Cadastro".

CADASTRO

Nome:

Usuário:

Senha:

Cadastro

# DAO - *Data Access Object*


- **Objeto de Acesso a Dados (DAO)**, é um padrão para aplicações que utilizam persistência de dados, onde existe a separação das regras de negócio e das regras de acesso ao banco de dados
- Implementada com linguagens de programação orientadas a objetos e arquitetura MVC, onde todas as funcionalidades de bancos de dados, tais como obter conexões ou executar comandos SQL, devem ser feitas por classes DAO

# JDBC

- **Java Database Connectivity** ou **JDBC** é um conjunto de classes e interfaces escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional

# Download do JDBC para PostgreSQL

<https://jdbc.postgresql.org/download.html>

 PostgreSQL JDBC Driver

The world's most advanced open source database.

Home | About | Download | Documentation | Community | Development

## Download

- [About](#)
- [Current Version](#)
- [Other Versions](#)
- [Archived Versions](#)

## About

Binary JAR file downloads of the JDBC driver are available here and the current version with [Maven Repository](#). Because Java is platform neutral, it is a simple process of just downloading the appropriate JAR file and dropping it into your classpath. Source versions are also available here for recent driver versions.

# Package - dao

- Classe Conexao

```
2  package dao;
3
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.SQLException;
7
8  public class Conexao {
9      public Connection getConnection() throws SQLException{
10         Connection conexao = DriverManager.getConnection(
11             "jdbc:postgresql://localhost:5432/aluno","postgres","123456");
12         return conexao;
13     }
14 }
```



# Package - dao

AlunoDAO

método:

inserir

```
1 package dao;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.SQLException;
6 import java.sql.ResultSet;
7 import model.Aluno;
8
9 public class AlunoDAO {
10     private Connection conn;
11
12     public AlunoDAO(Connection conn) {
13         this.conn = conn;
14     }
15
16     public void inserir(Aluno aluno) throws SQLException{
17         String sql = "INSERT INTO alunos(nome,usuario,senha) "
18             + "values('"+ aluno.getNome() + "', '"+ aluno.getUsuario() + "'"
19             + ", '"+ aluno.getSenha() + "')";
20         PreparedStatement statement = conn.prepareStatement(sql);
21         statement.execute();
22         conn.close();
23     }
24 }
```

# Package - dao

## AlunoDAO

### método: consultar

```
25      public ResultSet consultar(Aluno aluno) throws SQLException{  
26          String sql = "SELECT * FROM alunos WHERE usuario = ? AND senha = ?";  
27          PreparedStatement statement = conn.prepareStatement(sql);  
28          statement.setString(1, aluno.getUsuario());  
29          statement.setString(2, aluno.getSenha());  
30          statement.execute();  
31          ResultSet resultado = statement.getResultSet();  
32          conn.close();  
33          return resultado;  
34      }  
35  }  
36  }
```

# Package - model

Classe Aluno:

- nome, usuario e senha
- construtor e getters e setters

# Package - controller

## ControllerCadastro

```
2      package controller;
3
4      import dao.AlunoDAO;
5      import dao.Conexao;
6      import java.sql.Connection;
7      import java.sql.SQLException;
8      import javax.swing.JOptionPane;
9      import model.Aluno;
10     import view.CadastroForm;
11
12     public class ControllerCadastro {
13         private CadastroForm view;
14
15         public ControllerCadastro(CadastroForm view) {
16             this.view = view;
17         }
18     }
```

# Package - controller

## ControllerCadastro

```
19 public void salvarAluno(){
20     String nome = view.getCaixaNome().getText();
21     String usuario = view.getCaixaUsuario().getText();
22     String senha = view.getCaixaSenha().getText();
23     Aluno aluno = new Aluno(nome, usuario, senha);
24     Conexao conn = new Conexao();
25     try{
26         Connection connection = conn.getConnection();
27         AlunoDAO dao = new AlunoDAO(connection);
28         dao.inserir(aluno);
29         JOptionPane.showMessageDialog(view, "Usuário cadastrado com sucesso",
30                                     "Cadastrado!", JOptionPane.INFORMATION_MESSAGE);
31     }catch(SQLException e){
32         JOptionPane.showMessageDialog(view, "Falha no cadastro",
33                                     "Erro!", JOptionPane.ERROR_MESSAGE);
34         e.printStackTrace();
35     }
36 }
37
38 }
```

Obrigada pela sua  
participação, nos vemos na  
próxima aula! :)