

# BANCO DE DADOS RELACIONAL

**Junção de Tabelas (INNER JOIN, LEFT JOIN)**

***Professora:***

Lucineide Pimenta

# Objetivos da aula



## ✓ **Objetivos Gerais:**

- ✓ Ensinar **como relacionar dados de diferentes tabelas** usando comandos JOIN.
- ✓ Explicar os tipos de junção **INNER JOIN e LEFT JOIN**.
- ✓ Mostrar exemplos com **banco de dados de escola e sistema bancário**.
- ✓ Relacionar as junções com a estrutura do **projeto ABP**.

## ✓ **Objetivos Específicos:**

- ✓ Compreender a **importância dos relacionamentos** no banco de dados.
- ✓ Aprender a **combinar informações de tabelas diferentes**.
- ✓ Aplicar os comandos INNER JOIN e LEFT JOIN para consultas mais completas.
- ✓ Criar consultas que possam ser usadas no **projeto ABP**.

# Crie a estrutura do banco de dados

```
create database bd_escola;
```

```
CREATE TABLE cursos (id_curso SERIAL PRIMARY KEY, nome VARCHAR(100) NOT NULL);
```

```
CREATE TABLE alunos (id_aluno SERIAL PRIMARY KEY, nome VARCHAR(100) NOT NULL, idade INT, id_curso INT REFERENCES cursos(id_curso));
```

```
CREATE TABLE notas (id_nota PRIMARY KEY, disciplina VARCHAR(100) NOT NULL, nota float, id_aluno INT REFERENCES alunos(id_aluno));
```

# Crie a estrutura do banco de dados

```
INSERT INTO cursos (nome) VALUES ('Engenharia');
```

```
INSERT INTO cursos (nome) VALUES ('Análise de Sistemas'), ('Computação'), ('Matemática');
```

```
INSERT INTO alunos (nome, idade, id_curso) VALUES ('João Silva', 22, 1);
```

```
INSERT INTO alunos (nome, idade, id_curso) VALUES ('Marina Lima', 16, 3), ('Maria Souza', 20, 3),  
( 'Carlos Lima', 25, 4), ('Lucas Pereira', 18, 3);
```

```
INSERT INTO notas (id_nota, id_aluno, disciplina, nota) VALUES (101, 1, 'Matemática', 8.5), (102, 2,  
'História', 9.0);
```

# Crie a estrutura do banco de dados

*UPDATE alunos SET idade = 16 WHERE nome = 'João Silva';*

*UPDATE alunos SET idade = 17, id\_curso = 1 WHERE nome = 'Marina Lima';*

*Select \* from cursos;*

*Select \* from alunos;*

# Por que precisamos unir tabelas?

## ❑ Problema:

- Em um **banco de dados escolar**, temos uma tabela de **alunos** e outra de **notas**.
- Como saber quais alunos tiraram determinada nota em cada matéria?

## ❑ Solução:

- Usamos um **JOIN** para **combinar** as informações de ambas as tabelas.
- Isso evita **duplicação de dados** e melhora o desempenho das consultas.

## ❑ Exemplo:

Temos as tabelas:

### ◆ Alunos

id_aluno	nome	idade
1	João Silva	16
2	Maria Lima	17

### ◆ Notas

id_nota	id_aluno (FK)	disciplina	nota
101	1	Matemática	8.5
102	2	História	9.0

**Pergunta:** Como mostrar o nome do aluno junto com a nota e a disciplina?

**Resposta:** Usamos um JOIN!

# O que é INNER JOIN?

## Definição:

INNER JOIN retorna **apenas os registros que possuem correspondência** em ambas as tabelas.

## Sintaxe:

```
SELECT alunos.nome, notas.disciplina, notas.nota
FROM alunos
INNER JOIN notas ON alunos.id_aluno = notas.id_aluno;
```

nome	disciplina	nota
João Silva	Matemática	8.5
Maria Lima	História	9.0

## Resultado:

Só aparecem os alunos **que possuem notas registradas!**

# Exemplo com Sistema Bancário

- ❑ **Problema:**
  - No banco, temos **clientes** e **transações bancárias**.
  - Como saber quais clientes fizeram transações?

## ❑ Tabelas:

### ◆ Clientes

id_cliente	nome	saldo
1	Pedro Souza	5000
2	Ana Martins	3000

### ◆ Transações

id_transacao	id_cliente (FK)	tipo	valor
101	1	Saque	500
102	2	Depósito	1000



# Exemplo com Sistema Bancário

## ❑ Consulta para unir os dados:

*SELECT clientes.nome, transacoes.tipo, transacoes.valor*

*FROM clientes*

*INNER JOIN transacoes ON clientes.id\_cliente = transacoes.id\_cliente;*

## ❑ Resultado:

nome	tipo	valor
Pedro Souza	Saque	500
Ana Martins	Depósito	1000

Agora temos **os clientes e suas transações** juntas!

# O que é LEFT JOIN?

## Definição:

LEFT JOIN retorna **todos os registros da tabela da esquerda** e os **correspondentes** da tabela da direita.

Se **não houver correspondência**, exibe NULL.

## Exemplo no banco de dados escolar:

```
SELECT alunos.nome, notas.disciplina, notas.nota
FROM alunos
LEFT JOIN notas ON alunos.id_aluno = notas.id_aluno;
```

## Resultado:

nome	disciplina	nota
João Silva	Matemática	8.5
Maria Lima	História	9.0
Lucas Pereira	NULL	NULL

O aluno **Lucas Pereira aparece, mesmo sem notas registradas!**

# Atividade Prática (Individual)

## Exercícios práticos:

Crie as tabelas abaixo e relacione-as corretamente:

Insira os dados:

Liste os alunos e seus cursos (INNER JOIN):

## Resultado esperado:

nome	curso
Carlos Almeida	Engenharia
Fernanda Costa	Computação

Liste todos os alunos, incluindo os sem curso (LEFT JOIN):

**Agora o aluno Roberto aparece, mesmo sem curso!**

# BANCO DE DADOS RELACIONAL

Sistema Bancário - Modelagem e Implementação do Banco de Dados

- ❑ Uma instituição financeira deseja criar um **sistema bancário** para gerenciar **clientes, contas e transações**. O banco de dados precisa armazenar as seguintes informações:
  - ❑ **Clientes**: Nome, CPF, endereço e telefone.
  - ❑ **Contas Bancárias**: Número da conta, saldo e o cliente associado.
  - ❑ **Transações**: Registra saques, depósitos e transferências.
- ❑ **Requisitos**:
  - ❑ Cada cliente pode ter **uma ou mais contas bancárias**.
  - ❑ Cada conta pode ter **várias transações** registradas.
  - ❑ Apenas clientes cadastrados podem ter uma conta no banco.
  - ❑ O sistema deve permitir consultas para listar **clientes, contas e transações**.

# Criando o Banco de Dados

- ❑ *CREATE DATABASE sistema\_bancario;*

# Criando a Tabela de Clientes

## Tabela clientes

- Cada cliente tem um **ID único** (chave primária).
- O **CPF é único** e obrigatório.

```
CREATE TABLE clientes (  
    id_cliente SERIAL PRIMARY KEY,  
    nome VARCHAR(100) NOT NULL,  
    cpf VARCHAR(11) UNIQUE NOT NULL,  
    endereco TEXT,  
    telefone VARCHAR(15)  
);
```

## Exemplo de inserção de clientes:

```
INSERT INTO clientes (nome, cpf, endereco, telefone) VALUES  
('João Silva', '12345678900', 'Rua A, 123', '11999990000'),  
('Maria Oliveira', '98765432100', 'Rua B, 456', '11988887777');
```

# Criando a Tabela de Contas Bancárias

## Tabela contas

- Cada conta tem um **número único** e pertence a um **cliente**.
- O saldo inicial deve ser **zero por padrão**.

```
CREATE TABLE contas (  
    id_conta SERIAL PRIMARY KEY,  
    numero_conta VARCHAR(10) UNIQUE NOT NULL,  
    saldo DECIMAL(10,2) DEFAULT 0,  
    id_cliente INT REFERENCES clientes(id_cliente) ON DELETE CASCADE  
);
```

## Exemplo de inserção de contas:

```
INSERT INTO contas (numero_conta, saldo, id_cliente) VALUES  
( '000123', 1500.00, 1),  
( '000456', 2300.00, 2);
```



# Criando a Tabela de Transações

## Tabela transacoes

- Cada transação tem um **ID único** e está ligada a uma **conta bancária**.
- O tipo pode ser **saque, depósito ou transferência**.
- O campo destino\_transferencia é usado apenas para transferências.

```
CREATE TABLE transacoes (
    id_transacao SERIAL PRIMARY KEY,
    id_conta INT REFERENCES contas(id_conta) ON DELETE CASCADE,
    tipo VARCHAR(15) CHECK (tipo IN ('Depósito', 'Saque', 'Transferência')),
    valor DECIMAL(10,2) NOT NULL CHECK (valor > 0),
    data_transacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    destino_transferencia INT REFERENCES contas(id_conta)
);
```

## Exemplo de inserção de transações:

```
INSERT INTO transacoes (id_conta, tipo, valor) VALUES
(1, 'Depósito', 500.00),
(2, 'Saque', 200.00),
(1, 'Transferência', 300.00, 2);
```

# Consultas Básicas no Sistema Bancário

- ❑ **Listar todos os clientes cadastrados:**

```
SELECT * FROM clientes;
```

- ❑ **Listar todas as contas e seus respectivos clientes:**

```
SELECT contas.numero_conta, clientes.nome, contas.saldo
```

```
FROM contas
```

```
INNER JOIN clientes ON contas.id_cliente = clientes.id_cliente;
```

- ❑ **Listar todas as transações registradas:**

```
SELECT transacoes.tipo, transacoes.valor, transacoes.data_transacao,
```

```
contas.numero_conta AS origem,
```

```
c2.numero_conta AS destino
```

```
FROM transacoes
```

```
INNER JOIN contas ON transacoes.id_conta = contas.id_conta
```

```
LEFT JOIN contas c2 ON transacoes.destino_transferencia = c2.id_conta;
```

# Atualizações e Remoções

- ❑ **Atualizar o saldo de uma conta (exemplo de um depósito):**

*UPDATE contas*

*SET saldo = saldo + 500.00*

*WHERE id\_conta = 1;*

- ❑ **Excluir um cliente e suas contas (devido à regra ON DELETE CASCADE, as contas e transações associadas também serão excluídas automaticamente):**

*DELETE FROM clientes WHERE id\_cliente = 2;*

# Atividade Prática (Individual)

Insira um novo cliente no sistema.

Crie uma conta para esse novo cliente.

Realize uma transferência de R\$ 100,00 da conta 000123 para a conta 000789.

Liste todas as contas do banco, mostrando os saldos atualizados.

**Agora seu sistema bancário está funcional!**

# Referências Bibliográfica da Aula

## Livros:

**Elmasri & Navathe (2010).** Sistemas de Banco de Dados.

**Silberschatz et al. (2011).** Sistemas de Banco de Dados.

## Links úteis:

 [PostgreSQL Docs](#)

 [DBDiagram.io](#)

# Bibliografia Básica

- ❑ DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro, Elsevier: Campus, 2004.
- ❑ ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 7 ed. São Paulo: Pearson, 2018.
- ❑ SILBERSCHATZ, A.; SUNDARSHAN, S.; KORTH, H. F. **Sistema de banco de dados**. Rio de Janeiro: Elsevier Brasil, 2016.

# Bibliografia Complementar

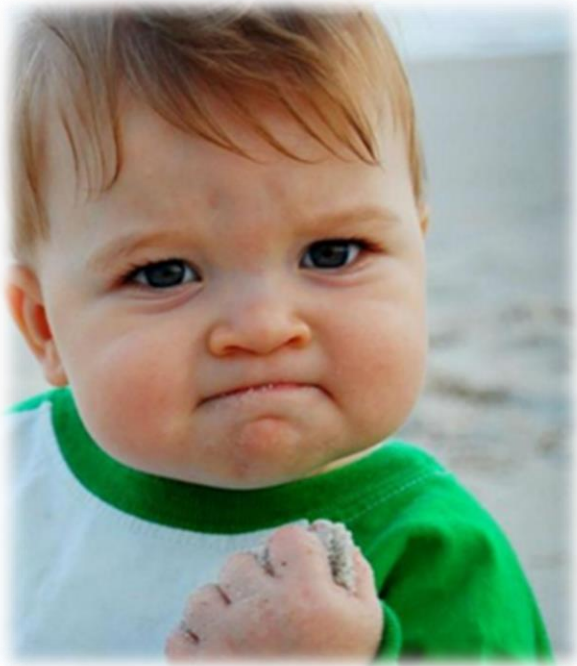
- ❑ BEAULIEU, A. **Aprendendo SQL**. São Paulo: Novatec, 2010.
- ❑ GILLENSON, M. L. **Fundamentos de Sistemas de Gerência de Banco de Dados**. Rio de Janeiro: LTC, 2006.
- ❑ MACHADO, F. N. R. **Banco de Dados: Projeto e Implementação**. São Paulo: Érica, 2005.
- ❑ OTEY, M; OTEY, D. **Microsoft SQL Server 2005: Guia do Desenvolvedor**. Rio de Janeiro: Ciência Moderna, 2007.
- ❑ RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de Gerenciamento de Bancos de Dados**. 3 ed. Porto Alegre: Bookman, 2008.
- ❑ ROB, P; CORONEL, C. **Sistemas de Banco de Dados: Projeto, Implementação e Gerenciamento**. 8 ed. São Paulo: Cengage Learning, 2011.
- ❑ TEOREY, T; LIGHTSTONE, S; NADEAU, T. **Projeto e Modelagem de Bancos de Dados**. São Paulo: Campus, 2006.

# Dúvidas?





# Considerações Finais



**Professora:  
Lucineide Pimenta**

**Bom descanso à todos!**

