

BANCO DE DADOS RELACIONAL

Criando o Banco de Dados e Tabelas

Objetivos da aula



📌 Objetivos Gerais:

- Ensinar os **comandos essenciais** para criar bancos de dados e tabelas no PostgreSQL.
- Explicar como definir **tipos de dados** corretamente.
- Demonstrar a importância das **chaves primárias (PK) e estrangeiras (FK)**.
- Relacionar o conteúdo com a **estruturação do banco de dados do projeto ABP**.

☐ 📌 Objetivos Específicos:

- ✓ Criar um banco de dados no **PostgreSQL**.
- ✓ Criar tabelas corretamente usando **tipos de dados adequados**.
- ✓ Aplicar **constraints** para garantir integridade dos dados.
- ✓ Relacionar tabelas utilizando **chaves primárias e estrangeiras**.

O que é um Banco de Dados?

❑ Definição:

- Um **banco de dados** é um ambiente onde os dados são **armazenados, organizados e acessados de forma eficiente**.
- Em sistemas modernos, o **PostgreSQL** é amplamente utilizado por empresas e desenvolvedores.

📌 Exemplo prático:

Imagine um sistema de **cadastro de alunos**. Como organizar as informações corretamente?

❌ **Errado:** Criar um arquivo do Excel com os dados desorganizados.

✅ **Certo:** Criar um **banco de dados relacional** para armazenar alunos, cursos e matrículas de forma estruturada.

Criando um Banco de Dados no PostgreSQL

❑ Passo a passo:

1 Abrir o terminal do PostgreSQL ou pgAdmin

2 Criar um banco de dados usando SQL:

```
CREATE DATABASE escola;
```

3 Conectar ao banco de dados criado:

```
\c escola;
```

4 Pronto! Agora podemos criar tabelas dentro do banco.

O que são Tabelas no Banco de Dados?

❑ Definição:

- Uma **tabela** é a estrutura principal onde os dados são armazenados no PostgreSQL.
- Ela é organizada em **colunas (atributos)** e **linhas (registros)**.

Exemplo de tabela "Alunos":

id_aluno (PK)	nome	idade	curso
1	Ana Souza	20	ADS
2	João Oliveira	22	DSM

Criando Tabelas no PostgreSQL

Sintaxe básica:

```
CREATE TABLE alunos (  
    id_aluno SERIAL PRIMARY KEY,  
    nome VARCHAR(100) NOT NULL,  
    idade INT,  
    curso VARCHAR(50)  
);
```

Explicação:

SERIAL PRIMARY KEY → Cria um identificador único para cada aluno.

VARCHAR(100) → Define um campo de texto com limite de 100 caracteres.

NOT NULL → Garante que o campo não pode ficar vazio.

INT → Define um campo numérico para armazenar idade.

Tipos de Dados no PostgreSQL

- ❑ Principais tipos de dados usados em tabelas:

Tipo	Descrição	Exemplo
INTEGER	Números inteiros	18, 100, 9999
VARCHAR(n)	Texto limitado a "n" caracteres	"João", "Maria"
TEXT	Texto longo sem limite	"Descrição completa"
DATE	Armazena datas	"2024-03-01"
BOOLEAN	Verdadeiro/Falso	TRUE, FALSE

 **Escolher o tipo correto melhora o desempenho do banco de dados!**

Chaves Primárias e Estrangeiras

Chave Primária (PK)

- ✓ Identifica cada registro de forma **única** dentro da tabela.
- ✓ Exemplo: **id_aluno** na tabela "Alunos".

Chave Estrangeira (FK)

- ✓ Conecta tabelas diferentes criando **relacionamentos**.
- ✓ Exemplo: Um **curso** pode ter vários alunos, então usamos FK para relacioná-los.

Chaves Primárias e Estrangeiras

Exemplo prático:

```
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)  
);
```

BANCO DE DADOS RELACIONAL

Abrindo o Terminal do PostgreSQL (psql)

Abrindo o Terminal do PostgreSQL (psql)

- ❑ Abra o **Prompt de Comando** (*cmd*) ou **PowerShell**.

- ❑ Digite:

psql -U postgres

- ❑ Insira a senha do usuário.

- ❑ Se conectar a um banco específico:

\c nome_do_banco

Comandos Básicos no psql

- ❑ **Listar bancos de dados:**

\l

- ❑ **Conectar a um banco:**

\c nome_do_banco

- ❑ **Listar tabelas do banco:**

\dt

- ❑ **Ver estrutura de uma tabela:**

\d nome_da_tabela

- ❑ **Sair do psql:**

\q



BANCO DE DADOS RELACIONAL

DDL (*Data Definition Language*)

Principais Comandos DDL com Exemplos

- ❑ **Criando um Banco de Dados:**

```
CREATE DATABASE bd_sistema_bancario;
```

- ❑ O comando acima cria um banco chamado bd_sistema_bancario.

- ❑ **Para listar bancos criados:**

```
\l
```

- ❑ **Para conectar ao banco criado:**

```
\c bd_sistema_bancario
```

O que é DDL?

- ❑ DDL (*Data Definition Language*) é um conjunto de comandos SQL usados para criar, modificar e excluir estruturas de banco de dados, como tabelas e esquemas.
- ❑ Principais comandos DDL:
 - ❑ **CREATE** → Criar banco de dados, tabelas e esquemas
 - ❑ **ALTER** → Modificar tabelas existentes
 - ❑ **DROP** → Excluir bancos de dados ou tabelas

Criando Tabelas (CREATE TABLE)

- Ao criar uma tabela, é preciso definir suas colunas e os tipos de dados:

```
CREATE TABLE cliente2 (  
    id_cliente SERIAL PRIMARY KEY, -- Chave primária autoincrementada  
    nome VARCHAR(100) NOT NULL,   -- Nome obrigatório  
    cpf VARCHAR(11) UNIQUE,        -- CPF único  
    idade INTEGER CHECK (idade > 18) -- Restringe idade maior que 18  
);
```

- Aqui, temos diferentes restrições:
 - **PRIMARY KEY**: Identificador único
 - **NOT NULL**: campo não pode ser nulo
 - **UNIQUE**: Evita valores duplicados
 - **CHECK**: Restringe valores de uma coluna

Exemplo de Sistema Bancário

Criando as tabelas conforme a aula:

```
CREATE TABLE agencia (  
    nome_agencia VARCHAR(50) PRIMARY KEY,  
    cidade_agencia VARCHAR(50),  
    depositos DECIMAL(10,2)  
);
```

```
CREATE TABLE conta (  
    numero_conta SERIAL PRIMARY KEY,  
    nome_agencia VARCHAR(50) REFERENCES agencia(nome_agencia),  
    saldo DECIMAL(10,2) DEFAULT 0.00  
);
```

*Aqui, usamos chaves estrangeiras
REFERENCES para relacionar tabelas.*

Exemplo de Sistema Bancário

Criando as tabelas conforme a aula:

```
CREATE TABLE emprestimo (  
    numero_emprestimo SERIAL PRIMARY KEY,  
    nome_agencia VARCHAR(50) REFERENCES agencia(nome_agencia),  
    valor DECIMAL(10,2) CHECK (valor > 0)  
);
```

*Aqui, usamos **chaves estrangeiras**
REFERENCES para relacionar tabelas.*

```
CREATE TABLE cliente (  
    nome_cliente VARCHAR(100),  
    cidade_cliente VARCHAR(50),  
    endereco_cliente TEXT,  
    idade INTEGER,  
    cpf VARCHAR(11) UNIQUE,  
    PRIMARY KEY (cpf)  
);
```

Excluindo Bancos e Tabelas (DROP)

- ❑ Se precisar excluir um banco ou tabela:

DROP DATABASE *bd_sistema_bancario;*


DROP TABLE *cliente;*

⚠ **Cuidado!** Esses comandos removem os dados permanentemente.

Modificando Tabelas (ALTER)

- ❑ Se precisarmos alterar a estrutura de uma tabela:
ALTER TABLE cliente **ADD COLUMN** telefone VARCHAR(15); -- Adiciona uma nova coluna
ALTER TABLE cliente **RENAME COLUMN** telefone TO contato; -- Renomeia a coluna
ALTER TABLE cliente **DROP COLUMN** contato; -- Remove a coluna
ALTER TABLE cliente **RENAME TO** tbl_cliente; -- Renomeia a tabela
- ❑ **ALTER TABLE** permite modificar a estrutura sem recriar a tabela.

Atividade Prática (Individual)

- 1 **Crie um banco de dados chamado** empresa.
 - 2 **Crie uma tabela** funcionários com os seguintes campos: id_func, nome, cargo, salario.
 - 3 **Insira 3 registros na tabela.**
 - 4 **Faça uma consulta para exibir os funcionários cadastrados.**
-  **Agora os dados estão salvos no banco!**

Entrega do Requisito (Em Grupo)



O que deve ser entregue?



Criação do banco de dados do projeto ABP.



Estruturação das tabelas principais do projeto.



Relacionamentos entre as tabelas usando FK.



Requisito atendido: BDR.01 - Junção de tabelas.



Como será avaliado?

✓ Nome adequado das tabelas e colunas.

✓ Uso correto de **tipos de dados**.




✓ Aplicação de **chaves primárias e estrangeiras** corretamente.



Prazo de entrega: 15/04 - Sprint 1.

Entrega do Requisito (BDR.01)

O que deve ser entregue?





-  **Modelo de banco de dados** para o desafio da ABP.
-  **Diagrama Entidade-Relacionamento (DER)** com tabelas do projeto.
-  **Requisito atendido: BDR.01 - Junção de tabelas.**

Como será avaliado?

- ✓ Estrutura correta do modelo relacional.
- ✓ Uso adequado de **chaves primárias e estrangeiras**.
- ✓ Clareza e organização do diagrama.

 **Prazo de entrega:** 15/04 - Sprint 1.

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.

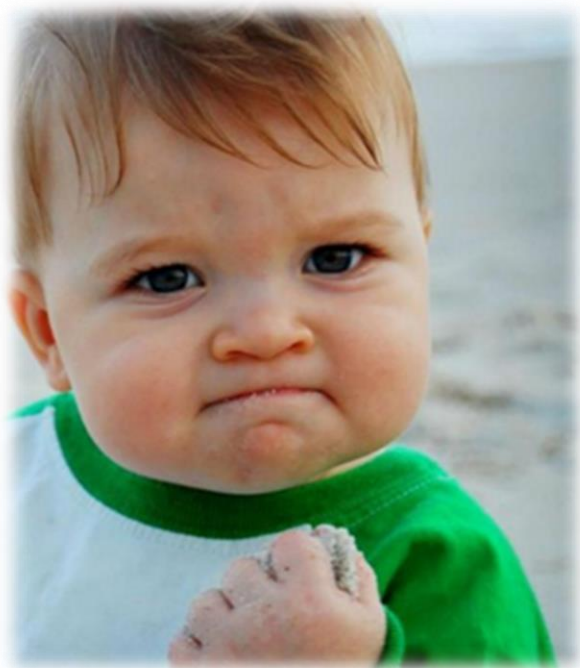
Bibliografia Complementar

- ❑ 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



**Professor(a):
Lucineide Pimenta**

Bom semestre à todos!

