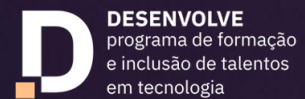


KORU





INTRODUÇÃO A DATABASES E SQL

AGENDA

O que vamos ver hoje:





- O que são bancos de dados e por que precisamos deles
- Bancos de dados relacionais - Tabelas, linhas e colunas
- SQLite vs MySQL - Quando usar cada um
- Comandos SQL básicos - SELECT, INSERT, UPDATE, DELETE
- Conexão com aplicações web - Como tudo se conecta
- Segurança - Evitando SQL Injection

A group of diverse young people, including men and women of various ethnicities, are smiling and waving. The image is overlaid with a purple gradient. The text "IMAGINE..." is written in white, bold, sans-serif font on the right side of the image.

IMAGINE...


... UM MUNDO SEM BANCO DE DADOS

Como o Instagram

-  1 bilhão de fotos por dia?
-  500 milhões de comentários diários?
-  4 bilhões de likes por dia?
-  Dados de 2 bilhões de usuários?

Problemas:

Buscar uma foto
Múltiplos usuários
Relacionar dados



O QUE É UM BANCO DE DADOS (DATABASE / DB)

O QUE É UM BANCO DE DADOS?

Sistema organizado para:

- Armazenar dados de forma estruturada
- Buscar informações rapidamente
- Relacionar dados entre si
- Proteger contra perda e acesso não autorizado

Imagine:

Um excel superpoderoso

ONDE USAMOS BANCO DE DADOS

Exemplos do dia a dia:

- E-commerce: Produtos, pedidos, clientes
- Redes sociais: Posts, curtidas, amigos
- Netflix: Filmes, séries, histórico
- Spotify: Músicas, playlists, artistas
- Bancos: Contas, transações, saldos
- Companhias aéreas: Voos, reservas, passageiros



BANCO DE DADOS RELACIONAIS SQL

BANCO DE DADOS RELACIONAIS

Organizam dados em TABELAS

- Tabela: users

```
- +---+-----+-----+
- | id | name   | email           |
- +---+-----+-----+
- | 1 | João   | joao@email.com |
- |   |       |                 |
- | 2 | Maria  | maria@email.com |
- |   |       |                 |
- | 3 | Pedro  | pedro@email.com |
- |   |       |                 |
- +---+-----+-----+
```

Componentes:

Tabela: Conjunto de dados similares

Linha/Registro: Um item específico

Coluna/Campo: Uma característica

Chave Primária: ID único



SQL

SQL: A LINGUAGEM DOS BANCOS DE DADOS

Structured Query Language (SQL)

Linguagem padrão para bancos relacionais

Simples e poderosa

Usada há mais de 40 anos!

sql

SELECT - Buscar dados

INSERT - Adicionar dados

UPDATE - Modificar dados

DELETE - Remover dados

A group of diverse, smiling people in a purple-tinted background. The image features a group of seven individuals of various ethnicities and ages, all smiling and looking towards the camera. The background is a solid purple color with a subtle, darker purple geometric pattern. The text 'SQLITE VS MYSQL' is overlaid in the center in a bold, white, sans-serif font.

SQLITE VS MYSQL

SQLITE

Ideal para:

Desenvolvimento local

Aplicações desktop

Apps mobile

Sites pequenos/médios

Quem usa:

WhatsApp, Chrome, Android, iOS

Características:

Um único arquivo (database.db)

Zero config - Só usar!

Embutido/Embedded - Não precisa servidor

Leve - Apenas 600KB

MYSQL

Ideal para:

Aplicações em produção

Sites com muitos usuários

Sistemas corporativos

Quando precisa de performance

Quem usa:

Facebook, Twitter, YouTube,
Wikipedia

Características:













Servidor completo de banco de dados

Multi-usuário - Milhares de conexões

Escalável - De startup a enterprise

Rico em recursos - Replicação, clusters

SQLite VS MySQL

SQLite	MySQL
 Arquivo único	 Servidor separado
 Zero config	 Precisa configurar
 Um usuário por vez	 Múltiplos usuários
 Máx 281TB	 Sem limite prático
 Domínio público	 Open source (GPL)
 Mobile/Desktop	 Web/Enterprise

A group of diverse young people, including men and women of various ethnicities, are smiling and waving. The image is overlaid with a purple gradient. The word "SINTAXE" is written in large, bold, white capital letters on the right side.

SINTAXE

CRIANDO UMA TABELA

Explicando:

PRIMARY KEY: Identificador único

AUTOINCREMENT: ID automático

NOT NULL: Campo obrigatório

UNIQUE: Valor único na tabela

DEFAULT: Valor padrão

```
CREATE TABLE users (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    name TEXT NOT NULL,  
    email TEXT UNIQUE NOT NULL,  
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```


INSERT: ADICIONANDO DADOS

Explicando:

INTO users → Especifica em qual tabela inserir

(name, email) → Lista os campos que vamos preencher

VALUES → Indica os valores correspondentes aos campos

Separe múltiplos registros com vírgula

```
-- 1 Registro
INSERT INTO users (name, email)
VALUES ('Ana Silva', 'ana@email.com');

-- Múltiplos Registros
INSERT INTO users (name, email) VALUES
    ('Carlos Santos', 'carlos@email.com'),
    ('Julia Costa', 'julia@email.com'),
    ('Pedro Lima', 'pedro@email.com');
```

SELECT: BUSCANDO DADOS

Explicando:

***** significa "todos os campos"

FROM indica de qual tabela buscar

WHERE filtra os resultados

LIKE permite busca parcial

% é um coringa (qualquer coisa)

ORDER BY organiza os resultados

ASC = crescente (A-Z)

DESC = decrescente (Z-A)

```
-- Todos os Campos
SELECT * FROM users;

-- Campos específicos:
SELECT name, email FROM users;

-- Com filtro WHERE:
SELECT * FROM users
WHERE email LIKE '%gmail.com';

-- Ordenando:
SELECT * FROM users
ORDER BY name ASC;
```

UPDATE: MODIFICANDO DADOS

Explicando:

UPDATE indica qual tabela modificar

SET define o novo valor do campo

WHERE especifica QUAL registro atualizar

Separe múltiplos campos com vírgula

CUIDADO: Sempre use WHERE!

Sem **WHERE**, TODOS os registros são atualizados

Sempre teste com SELECT antes

Não há "desfazer" em SQL

```
-- Atualizar um campo:
```

```
UPDATE users
```

```
SET email = 'ana.silva@email.com'
```

```
WHERE id = 1;
```

```
-- Atualizar múltiplos campos:
```

```
UPDATE users
```

```
SET name = 'Ana Silva Santos',
```

```
    email = 'ana.santos@email.com'
```

```
WHERE id = 1;
```

DELETE: REMOVENDO DADOS

Explicando:

DELETE FROM indica a tabela

WHERE especifica o que deletar

Remove permanentemente o registro

Sem WHERE = deleta TODOS os registros

A tabela continua existindo, mas vazia

Sempre teste com SELECT antes de DELETE!

Faça backup antes de operações perigosas

```
-- Deletar registro específico:
```

```
DELETE FROM users WHERE id = 5;
```

```
-- ⚠ PERIGO - Deleta TUDO:
```

```
DELETE FROM users;
```



RELACIONAMENTO ENTRE TABELAS

RELACIONAMENTO ENTRE TABELAS

FOREIGN KEY:

Liga uma tabela a outra -

posts.author_id → users.id

Mantém integridade dos dados -

Impede posts de autores inexistentes

Base dos bancos "relacionais" - Permite conectar informações

author_id armazena o ID do usuário que criou o post

REFERENCES users(id) garante que só aceita IDs válidos

Permite fazer JOINS para buscar dados relacionados

```
-- Tabela de posts
CREATE TABLE posts (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT NOT NULL,
  content TEXT NOT NULL,
  author_id INTEGER NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (author_id) REFERENCES users(id)
);
```

JOIN: UNINDO TABELAS

Como funciona:

JOIN combina dados de duas tabelas relacionadas

ON posts.author_id = users.id
define a conexão entre elas

as author renomeia a coluna para ficar mais clara

```
SELECT
    posts.title,
    posts.content,
    users.name as author
FROM posts
JOIN users ON posts.author_id = users.id;
```

title	content	author
Meu primeiro post	Olá mundo!	João
Aprendendo SQL	SQL é incrível!	Maria



FLUXO TÍPICO DE APLICAÇÃO FULLSTACK

FLUXO TÍPICO DE APLICAÇÃO FULLSTACK

Fluxo típico de aplicação Fullstack

Usuário clica em "Ver posts"

Frontend envia requisição

Backend recebe e processa

Backend monta query SQL

Banco executa e retorna dados

Backend formata resposta

Frontend exibe os dados



SEGURANÇA: SQL & INJECTION

FLUXO TÍPICO DE APLICAÇÃO FULLSTACK

Risco de Injeção de Query:

Imagine que o usuário ao invés de colocar seu email coloque a seguinte string:

'; DROP TABLE users;`

O que aconteceria? **Solução:**
prepared statements.

```
const query = `SELECT * FROM users  
WHERE email = '${userInput}'`;
```

Risco de Injection



A group of diverse young people, including men and women of various ethnicities, are smiling and waving. The image is overlaid with a purple gradient. The word "RESUMO" is written in large, white, bold, sans-serif capital letters on the right side of the image.

RESUMO

RESUMO

O que aprendemos hoje:

Bancos de dados são essenciais para aplicações modernas

SQL é a linguagem universal para bancos relacionais

SQLite é perfeito para desenvolvimento e apps pequenas

MySQL é ideal para produção e múltiplos usuários

Comandos SQL fundamentais:

CREATE TABLE → Criar estrutura

INSERT → Adicionar dados

SELECT → Buscar informações

UPDATE → Modificar registros

DELETE → Remover dados

Conceitos-chave:

🔑 Primary Key = ID único

🔗 Foreign Key = Conexão entre tabelas

🤝 JOIN = Unir dados relacionados

🔒 SQL Injection = Sempre use prepared statements!

A group of diverse young people, including men and women of various ethnicities, are smiling and waving. The image is overlaid with a purple gradient. The word "PRÁTICA" is written in large, bold, white capital letters on the right side.

PRÁTICA

ATIVIDADE PRÁTICA

Preparação de Ambiente

Instalando SQLite

Instalando DB Browser for SQLite (ou outro gerenciador de DB como DBeaver, Sequel Ace, etc)

Manipulando o banco de dados

Criando o Banco via terminal

Criando tabela de Usuários

Criando tabela de Posts

Inserindo dados

Usuários

Posts

Consultando Dados

Consulta básica

Consulta com JOIN

Atualizando Dados

Atualização simples

! UPDATE sem WHERE

Deletando Dados


Outras Queries

Criando sistema de comentário

Query complexa: Post completo com comentários



**NOS VEMOS NA
PRÓXIMA AULA!**



**Momento de
avaliar a aula
de hoje!**