Banco de Dados e Introdução a SQL

Labenu_

Introdução à SQL

Labenu_

Relembrando persistência no front

- Como lidamos com isso?
    - LocalStorage e APIs

- No back-end o LocalStorage **não existe** e nossas APIs **ainda não lidam** com persistência

Persistência no Backend

- **Bancos de Dados**
  - Relacional (tabelas e ids)
  - Linguagem SQL
  - Gerenciador de banco de dados MySQL
  - Aplicativos mySQL, BeeKeeper ou extensões VSCode

Exemplo de tabela

Pessoas			
id (PK)	nome	email	idade
1	Fulano	fulano@gmail.com	35
2	Ciclana	ciclana@gmail.com	24
3	Alice	alice@gmail.com	33
4	Bob	bob@gmail.com	52

Structure Query Language (SQL)

- **Linguagem de Consulta Estruturada**
  é a linguagem padrão de banco de dados Relacionais

- É amplamente utilizada no mercado

- Baixa curva de aprendizado

Sistema de Gerenciamento de Banco de dados (SGBD)

- **MySQL**
  - Código aberto (gratuito)
  - Popular e fácil de iniciar o uso
  - Já temos um servidor pessoal

Criando e Deletando Tabelas

Labenu_

Antes de começar

- **Deixar configurado para utilizarmos nosso Banco de Dados**

```
USE nome_do_banco;
```

- **Liberar uma configuração de segurança**

```
SET SQL_SAFE_UPDATES = 0;
```

Criando tabelas

- Usamos o comando **CREATE TABLE** para criar as tabelas
- O comando recebe:
  - O nome da tabela
  - Nome e Tipo de cada coluna
  - Restrições das colunas:
    - Chave primária; obrigatoriedade; valor único; valor incremental, entre outros

Tipos principais

- **INT** - Números inteiros
- **DOUBLE** - Números com ponto flutuante
- **VARCHAR(n)** - String com no máximo N caracteres
- **CHAR** - String com 1 caractere
- **TEXT** - String com quantidade quase ilimitada de caracteres
- **ENUM** - Objeto com strings pré-definidas
- **DATE** - Representa data (YYYY-MM-DD)
- **DATETIME** - Representa data e tempo (YYYY-MM-DD hh:mm:ss)

Para saber mais, consulte: SQL Data Types

Restrições principais

- **PRIMARY KEY** - Chave primária (chave única na tabela)
- **FOREIGN KEY** - Chave estrangeira (chave única na tabela)
- **NULL / NOT NULL** - Indica se a coluna pode ser ou não pode ser nula
- **UNIQUE** - Indica que o valor deve ser único
- **AUTO_INCREMENT** - Indica que o valor é auto incrementável
- **DEFAULT** - Define um valor padrão caso nenhum valor seja passado

Para saber mais, consulte: SQL Restrições

SINTAXE

```
CREATE TABLE nome_tabela (
    coluna1 INT PRIMARY KEY,
    coluna2 VARCHAR(255) UNIQUE,
    coluna3 CHAR NULL DEFAULT 'X',
    coluna4 ENUM('op1', 'op2') NOT NULL,
    .....
    colunaN TIPO RESTRIÇÃO
);
```

Coding together 👩🏽‍💻

```sql
CREATE TABLE Pessoas (
    id      INT             PRIMARY KEY,
    nome    VARCHAR(255)    NOT NULL,
    email   VARCHAR(255)    NOT NULL UNIQUE,
    idade   INT             DEFAULT 18
);
```

Resultado:

Pessoas			
id	nome	email	idade

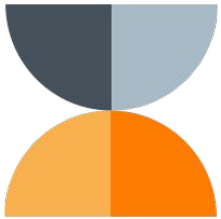
Deletando tabelas

- Usamos o comando **DROP TABLE** para deletar as tabelas
- O comando recebe o nome da tabela

```
DROP TABLE instrutores;
```

Pausa para relaxar 😴

10 min



- No backend, armazenamos os dados em **bancos** e utilizamos um **SGBD** para interagir com eles
- O banco relacional é formado por tabelas
- Toda tabela deve ter uma **chave primária** [PK]
- Os atributos das tabelas possuem tipos definidos

Inserindo e Removendo Valores

Labenu_

Inserindo valores na tabela

- Usamos o comando **INSERT** para adicionar valores na tabela
- O comando recebe:
  - O nome da tabela
  - Os nomes das colunas em ordem de inserção
  - Os valores

```
INSERT INTO nome_tabela (coluna1, coluna2, coluna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

Sintaxe

Se for adicionar valores para **todas** as colunas da tabela, pode-se omitir os nomes das colunas. No entanto, a ordem dos valores devem ser passados na mesma ordem das colunas na tabela.

```
INSERT INTO nome_tabela
VALUES (valor1, valor2, valor3, ...);
```

Coding Together 👩🏽‍💻

```
INSERT INTO Pessoas (id, nome, email, idade)
VALUES (1, “Fulano”,”fulano@gmail.com”, 35),
       (2, “Ciclana”,”ciclana@gmail.com”, 24);
```

Resultado:

Pessoas			
id	nome	email	idade
1	Fulano	fulano@gmail.com	27
2	Ciclana	ciclana@gmail.com	24

Removendo valores da tabela

- Usamos o comando **DELETE** para apagar uma ou mais linhas de uma tabela.
- O comando recebe:
  - O nome da tabela
  - Opcional: A condição **WHERE**
    - Se houver, apagará somente as linhas que atendam às condições.
    - Se **não** houver, apagará **todas as linhas** da tabela. ⚠️

Sintaxe

Deleção segura (RECOMENDADA)

```
DELETE FROM nome_tabela
WHERE condicao = valor;
```

Apaga todas as linhas da tabela, não apaga a tabela em si

```
DELETE FROM nome_tabela;
```

Coding Together 👩🏽‍💻

```sql
DELETE FROM Pessoas
WHERE id = 1;
```

Consultando valores na tabela

Labenu_

Consultando os valores na tabela

- Usamos o comando **SELECT** para consultar os valores
- O comando recebe:
  - Os nomes das colunas que serão buscadas
    - Para buscar todas as colunas, usa-se: ***** (asterisco)
  - O nome da tabela
  - Opcional: A condição **WHERE**
    - Se houver, retorna as linhas que atendem a condição
    - Se não houver, todas as linhas são retornadas

Sintaxe

Retorna todos os valores de todas as colunas da tabela

```
SELECT * FROM nome_tabela;
```

Retorna os valores que atendam a condição das colunas especificadas

```
SELECT coluna1, coluna2 FROM nome_tabela
WHERE condicao = valor;
```

Coding Together 👩🏽‍💻

```sql
SELECT * FROM Pessoas;
```

Pessoas			
id (PK)	nome	email	idade
1	Fulano	fulano@gmail.com	35
2	Ciclana	ciclana@gmail.com	24
3	Alice	alice@gmail.com	33
4	Bob	bob@gmail.com	52

Coding Together 👩🏽‍💻

```
SELECT email, idade
FROM Pessoas
WHERE nome=”Fulano”;
```

Resultado	
email	idade
fulano@gmail.com	35

Coding Together 👩🏽‍💻

Podemos renomear a coluna com um ALIAS (AS)

```
SELECT email, idade AS idade_em_anos
FROM Pessoas
WHERE nome=”Fulano”;
```

Resultado	
email	idade_em_anos
fulano@gmail.com	35

Coding Together 👩🏽‍💻

Use esses exemplos, ou solte a criatividade para testar novos!

```
SELECT *
FROM Pessoas
WHERE idade = 24 OR nome = "Fulano";
```

```
SELECT id, nome
FROM Pessoas
WHERE nome LIKE "%an%";
```

```
SELECT email
FROM Pessoas
WHERE nome LIKE "%an%" AND
idade > 30;
```

Leia sobre o **%** aqui

Extra: Visualizar a estrutura da tabela

- Usamos o comando **DESCRIBE** para conferir a estrutura da tabela e os tipos de dados de cada coluna

```
DESCRIBE Pessoas;
```

Resultado:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
nome	varchar(255)	NO		NULL	
email	varchar(255)	NO	UNI	NULL	
idade	int(11)	YES		NULL	

E é isso por hoje!

obrigado pela atenção

Labenu_

Obrigado!