Aula 12

Populando o Banco de Dados (DML - INSERT)

(Slide 1 de 4) - Lembrando da Aula Anterior

- **O que fizemos?**
  - Finalizamos a estrutura do nosso banco.
  - Aprendemos sobre **Constraints**, as regras do nosso banco (`NOT NULL` , `UNIQUE`).
  - Usamos `FOREIGN KEY` para criar as conexões essenciais entre as tabelas.
- **Resultado:** Nosso banco de dados está estruturalmente correto, mas **vazio**.
- **O objetivo de hoje:**
  - Aprender a **popular** nossas tabelas, ou seja, inserir os primeiros registros de dados.

(Slide 2 de 4) - Bem-vindo ao DML: Manipulando os Dados

- Até agora, usamos **DDL (Data Definition Language)** para *definir* a estrutura.
- A partir de hoje, começamos a usar **DML (Data Manipulation Language)** para *manipular* os dados dentro dessa estrutura.
- Os principais comandos DML são:
  - `INSERT` (Inserir novos dados) -> **Nosso foco hoje!**
  - `UPDATE` (Atualizar dados existentes)
  - `DELETE` (Remover dados existentes)

(Slide 3 de 4) - O Comando `INSERT INTO`

- É o comando que usamos para adicionar novas linhas (registros) a uma tabela.
- **Sintaxe Mais Segura (Recomendada):**
  - `INSERT INTO nome_da_tabela (coluna1, coluna2)`
  - `VALUES ('valor_para_coluna1', 'valor_para_coluna2');`
- **Pontos Importantes:**
  - A ordem das colunas e dos valores deve ser a mesma.
  - Valores de texto (`VARCHAR`) e data (`DATE`) devem estar entre aspas simples: `'texto'` .
  - Valores numéricos (`INT`) não usam aspas.
  - **Não** incluímos a coluna de chave primária com `AUTO_INCREMENT` . O MySQL cuida disso.

(Slide 4 de 4) - Inserindo Múltiplos Dados e a Regra da `FOREIGN KEY`

- **Para inserir várias linhas de uma vez (mais eficiente):**
  - `INSERT INTO tabela (coluna1, coluna2)`
  - `VALUES ('valorA1', 'valorA2'), ('valorB1', 'valorB2'), ('valorC1', 'valorC2');`
- **A Regra da Chave Estrangeira:**
  - Você **não pode** inserir um registro "filho" sem que o "pai" já exista.
  - Exemplo: Para inserir um `livro`, o `autor` correspondente já deve estar cadastrado na tabela `autores`. Você precisará usar o `autor_id` daquele autor já existente.

Prática

```sql
-- Garante que estamos usando o banco de dados correto.
USE biblioteca_curso;

-- pesquisando a tabela autores
SELECT * FROM autores;

-- 1. Inserindo na tabela "pai" (autores), que não depende de nenhuma outra.
--    Primeiro, um autor de cada vez.
INSERT INTO autores (nome, sobrenome, data_nascimento)
VALUES ('George', 'Orwell', '1903-06-25');

--    Agora, múltiplos autores de uma vez para sermos mais eficientes.
INSERT INTO autores (nome, sobrenome, data_nascimento)
VALUES
    ('J.R.R.', 'Tolkien', '1892-01-03'),
    ('Isaac', 'Asimov', '1920-01-02');

-- Dica: Como saber os IDs que acabaram de ser criados?
-- Rode um `SELECT * FROM autores;` para ver a tabela com os IDs 1, 2 e 3.

-- selecionando livros
SELECT * FROM livros;
```

```sql
-- 2. Agora, inserindo na tabela "filha" (livros), que depende de autores.
--    Note que precisamos usar os IDs que já existem na tabela 'autores'.
INSERT INTO livros (titulo, isbn, ano_publicacao, autor_id)
VALUES ('1984', '978-0451524935', 1949, 1); -- '1984' foi escrito por George Orwell (ID=1)

--    Inserindo múltiplos livros.
INSERT INTO livros (titulo, isbn, ano_publicacao, autor_id)
VALUES
    ('O Hobbit', '978-0345339683', 1937, 2), -- O Hobbit foi escrito por J.R.R. Tolkien (ID=2)
    ('A Revolução dos Bichos', '978-0451526342', 1945, 1); -- Outro livro de George Orwell (ID=1)

-- 3. Verificando o resultado final.
--    Agora nossas tabelas têm dados e estão conectadas!
SELECT * FROM autores;
SELECT * FROM livros;
```

Exercícios

Insira mais 3 valores em cada uma das duas tabelas, autores e livros