UniSENAI

Módulo 2: Estruturação de Bancos de Dados

Módulo 2: Estruturação de Bancos de Dados

1. Modelo de Entidade e Relacionamento

O **Modelo de Entidade e Relacionamento (MER)** é fundamental para a estruturação de bancos de dados. Ele permite planejar e organizar as informações que serão armazenadas e suas relações, garantindo integridade e eficiência na implementação.

1.1 Entidades

- **Exemplo Prático**:
  - Entidade: `Cliente`
    - **Atributos**:
      - `id_cliente` (Chave Primária)
      - `nome` (VARCHAR)
      - `email` (VARCHAR)
      - `telefone` (VARCHAR)
  - Entidade: `Produto`
    - **Atributos**:
      - `id_produto` (Chave Primária)
      - `nome` (VARCHAR)
      - `preco` (DECIMAL)
      - `estoque` (INT)
- **Boas Práticas**:
  - Sempre defina uma chave primária única e significativa.
  - Evite atributos redundantes, como `preco_produto` e `valor_produto` se ambos significarem o mesmo.

1.2 Atributos

- **Exemplo Prático**:
  - Atributos de `Pedido`:
    - `id_pedido` (Chave Primária)
    - `data_pedido` (DATE)
    - `id_cliente` (Chave Estrangeira)
- **Boas Práticas**:
  - Utilize nomes que reflitam a natureza do dado.
  - Escolha tipos de dados que otimizem armazenamento e operações:
    - Exemplo: Use `VARCHAR(100)` para nomes ao invés de `TEXT`.

1.3 Relacionamentos

- **Exemplo Prático**:
    - Relacionamento: `Cliente realiza Pedido`.
        - Tipo: 1:N (Um cliente pode realizar vários pedidos).
    - Relacionamento: `Produto pertence a Categoria`.
        - Tipo: N:1 (Muitos produtos pertencem a uma categoria).
- **Boas Práticas**:
    - Nomeie os relacionamentos de forma clara e descritiva.
    - Defina se o relacionamento é obrigatório ou opcional (exemplo: nem todos os clientes podem ter pedidos).

1.4 Cardinalidades

- **Exemplo Prático**:
    - **1:1**: Um `Cliente` possui um `Endereço Fiscal`.
    - **1:N**: Um `Cliente` pode realizar vários `Pedidos`.
    - **N:M**: Um `Produto` pode estar em vários `Pedidos` e cada `Pedido` pode conter vários `Produtos`.
- **Boas Práticas**:
    - Certifique-se de que as cardinalidades refletem os requisitos do sistema.
    - Para N:M, sempre use uma tabela intermediária (exemplo: `pedido_produto`).

1.5 Modelo Lógico e Físico

- **Modelo Lógico**:
    - Inclui todas as entidades, atributos e relacionamentos.
    - Exemplo: Representação gráfica no MySQL Workbench.
- **Modelo Físico**:
    - Define a implementação no SGBD, incluindo chaves primárias e estrangeiras, tipos de dados e restrições.
    - Exemplo: Criação das tabelas com SQL, como:

```sql
CREATE TABLE clientes (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    telefone VARCHAR(15)
);
```

Atividade Prática: Pesquisa sobre Normalização

Objetivo: Compreender o conceito de normalização e sua aplicação no projeto de bancos de dados.

1. **Tarefa do Acadêmico**:
    - Pesquisar sobre o processo de normalização, incluindo:
        - Conceitos das formas normais (1ª, 2ª, 3ª e BCNF).
        - Vantagens da normalização na eliminação de redundâncias e na garantia de integridade.
        - Exemplos práticos de como normalizar um banco de dados.
2. **Entrega**:
    - Elaborar uma apresentação explicando:
        - O que é normalização.
        - As formas normais e seus requisitos.
        - Um exemplo prático mostrando como transformar uma tabela não normalizada em 3ª forma normal.
3. **Formato da Apresentação**:
    - **Slides** (PowerPoint ou Google Slides) com:
        - Títulos claros.
        - Uso de gráficos ou diagramas para ilustrar o processo.
    - Duração sugerida: 10 minutos.
4. **Critérios de Avaliação**:
    - Clareza na explicação dos conceitos.
    - Qualidade visual e organização dos slides.
    - Aplicação correta do processo de normalização no exemplo.

2. Criação e Gerenciamento de Bancos de Dados

Para gerenciar um banco de dados é essencial dominar comandos básicos que permitem criar, excluir e selecionar um banco de dados. Vamos entender cada um deles.

2.1 Comando `CREATE DATABASE`

O comando `CREATE DATABASE` é usado para criar um novo banco de dados no MySQL. Ele é uma das primeiras etapas ao iniciar o desenvolvimento de um sistema.

Sintaxe:

```
CREATE DATABASE nome_do_banco;
```

Exemplo prático:

```
CREATE DATABASE loja_virtual;
```

Nesse exemplo, estamos criando um banco de dados chamado `loja_virtual`, que poderá armazenar tabelas relacionadas a uma loja online.

Comando com definição de caracteres:

```
CREATE DATABASE loja_virtual CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

Neste caso, especificamos o conjunto de caracteres e a collation (regras de classificação) para o banco de dados.

Boas Práticas:

- Dê nomes significativos ao banco de dados para facilitar a identificação.
- Utilize `CHARACTER SET` e `COLLATE` para evitar problemas de acentuação e caracteres especiais.

2.2 Comando `DROP DATABASE`

O comando `DROP DATABASE` é utilizado para excluir um banco de dados e todo o seu conteúdo (tabelas, registros, índices, etc.). Cuidado! Este comando é irreversível.

Sintaxe:

```
DROP DATABASE nome_do_banco;
```

Exemplo prático:

```
DROP DATABASE loja_virtual;
```

Este comando excluirá permanentemente o banco de dados `loja_virtual` e todos os seus dados.

Boas Práticas:

- Antes de usar `DROP DATABASE`, certifique-se de ter um backup dos dados.
- Evite executar este comando em ambientes de produção sem uma validação rigorosa.

2.3 Comando `USE` para Selecionar Banco de Dados

O comando `USE` é usado para selecionar um banco de dados para ser manipulado. Depois de selecionado, todas as operações subsequentes (CREATE TABLE, INSERT, SELECT, etc.) serão realizadas no banco de dados escolhido.

Sintaxe:

```
USE nome_do_banco;
```

Exemplo prático:

```
USE loja_virtual;
```

Neste exemplo, estamos selecionando o banco de dados `loja_virtual`. A partir desse momento, qualquer comando SQL será executado dentro desse banco.

Boas Práticas:

- Verifique se o banco foi selecionado corretamente antes de executar comandos.
- Use o comando `SHOW DATABASES;` para visualizar todos os bancos de dados disponíveis.Resumos dos Comandos

Comando	Função	Sintaxe
CREATE DATABASE	Cria um novo banco de dados	*CREATE DATABASE nome_do_banco;*
DROP DATABASE	Exclui um banco de dados	*DROP DATABASE nome_do_banco;*
USE	Seleciona um banco para uso	*USE nome_do_banco;*

Exercício Prático

1. Crie um banco de dados chamado `biblioteca`.
2. Selecione o banco `biblioteca` para realizar operações nele.
3. Depois de garantir que não há informações importantes, exclua o banco `biblioteca`.

Comandos Esperados:

```
CREATE DATABASE biblioteca;
USE biblioteca;
DROP DATABASE biblioteca;
```

3. Criação e Gerenciamento de Tabelas

---

1. Comando `CREATE TABLE`

O comando `CREATE TABLE` é usado para criar tabelas em um banco de dados. As tabelas são a estrutura principal de armazenamento de dados no MySQL. Cada tabela é composta por colunas (campos) que definem os tipos de dados e as restrições que serão aplicadas.

Sintaxe:

```
CREATE TABLE nome_da_tabela (
    nome_coluna1 TIPO_DADO(RESTRIÇÃO),
    nome_coluna2 TIPO_DADO(RESTRIÇÃO),
    ...
);
```

Exemplo prático:

```
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    telefone VARCHAR(15),
    data_cadastro DATE DEFAULT CURRENT_DATE
);
```

Neste exemplo, criamos uma tabela `clientes` com as seguintes colunas: `id`, `nome`, `email`, `telefone` e `data_cadastro`.

2. Definição de Colunas, Tipos de Dados e Restrições

Ao criar tabelas, é necessário definir as colunas, seus tipos de dados e as restrições que garantem a integridade dos dados.

2.1 Tipos de Dados

Tipo de Dado	Descrição
INT	Números inteiros
VARCHAR(n)	Texto de até n caracteres
DATE	Datas no formato AAAA-MM-DD

DECIMAL(x, y)	Números decimais com x digitos e y decimais
BOOLEAN	Valores booleanos (TRUE ou FALSE)

3.2 Restrições

Restrição	Descrição
PRIMARY KEY	Identifica unicamente cada registro
FOREIGN KEY	Cria relação com outra tabela
UNIQUE	Garante que o valor é único
NOT NULL	Impede valores nulos na coluna
DEFAULT	Define um valor padrão para a coluna

Exemplo prático:

```
CREATE TABLE pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT NOT NULL,
    data_pedido DATE DEFAULT CURRENT_DATE,
    valor_total DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);
```

Nesse exemplo, a tabela `pedidos` faz referência à tabela `clientes` através de uma chave estrangeira (`FOREIGN KEY`).

4. Alteração de Tabelas (`ALTER TABLE`)

O comando `ALTER TABLE` permite modificar uma tabela existente, adicionando, removendo ou alterando colunas, bem como mudando restrições.

4.1 Adição de Colunas Sintaxe:

```
ALTER TABLE nome_da_tabela ADD nome_da_coluna TIPO_DADO(RESTRIÇÃO);
```

Exemplo prático:

```
ALTER TABLE clientes ADD endereco VARCHAR(255);
```

4.2 Remoção de Colunas Sintaxe:

```
ALTER TABLE nome_da_tabela DROP COLUMN nome_da_coluna;
```

Exemplo prático:

```
ALTER TABLE clientes DROP COLUMN telefone;
```

4.3 Alteração de Colunas Sintaxe:

```
ALTER TABLE nome_da_tabela MODIFY nome_da_coluna TIPO_DADO(RESTRIÇÃO);
```

Exemplo prático:

```
ALTER TABLE clientes MODIFY nome VARCHAR(150) NOT NULL;
```

4.4 Alterar Restrições (PRIMARY KEY, FOREIGN KEY) Para adicionar ou remover chaves primárias e estrangeiras, utilizamos os seguintes comandos:

Adicionar chave estrangeira:

```
ALTER TABLE pedidos ADD FOREIGN KEY (cliente_id) REFERENCES clientes(id);
```

Remover chave estrangeira:

```
ALTER TABLE pedidos DROP FOREIGN KEY fk_cliente_id;
```

5. Exclusão de Tabelas (`DROP TABLE`)

O comando `DROP TABLE` exclui uma tabela e todo o seu conteúdo permanentemente. Cuidado! Esse comando é irreversível.

Sintaxe:

```
DROP TABLE nome_da_tabela;
```

Exemplo prático:

```
DROP TABLE pedidos;
```

Esse comando exclui a tabela `pedidos` e todos os seus dados.

Exercícios práticos

1. Crie uma tabela `produtos` com as colunas `id`, `nome`, `preco`, `quantidade_estoque` e `data_cadastro`. Use as restrições adequadas (chave primária, not null, etc.).
2. Adicione uma coluna `descricao` à tabela `produtos`.
3. Altere o tipo de dados da coluna `preco` para `DECIMAL(8, 2)`.
4. Exclua a coluna `quantidade_estoque` da tabela `produtos`.
5. Exclua a tabela `produtos`.