BD - DCL

Linguagem de Controle de Dados

📌 Linguagem DCL (Data Control Language)

A **DCL (Data Control Language)** é uma sublinguagem do SQL usada para **controlar permissões e segurança no banco de dados**. Com ela, podemos conceder e revogar acessos a usuários, garantindo que apenas pessoas autorizadas possam manipular determinados dados.

🔹 Principais Comandos

A DCL possui **dois comandos principais**:

1️⃣ GRANT – Conceder Permissões

O comando `GRANT` permite conceder permissões para usuários ou funções específicas dentro do banco de dados.

📌 Exemplo 1: Concedendo permissão para selecionar dados de uma tabela

```sql
GRANT SELECT ON funcionarios TO usuario1;
```

✅ **Explicação:** O usuário **usuario1** poderá executar comandos `SELECT` na tabela **funcionarios**.

📌 Exemplo 2: Concedendo múltiplas permissões

```sql
GRANT INSERT, UPDATE, DELETE ON pedidos TO gerente;
```

✅ **Explicação:** O usuário **gerente** poderá **inserir, atualizar e deletar** registros na tabela **pedidos**.

📌 **Exemplo 3: Concedendo todas as permissões**

```sql
GRANT ALL PRIVILEGES ON clientes TO admin;
```

✅ **Explicação:** O usuário **admin** terá **todos os privilégios** sobre a tabela **clientes**.

2️⃣ REVOKE – Revogar Permissões

O comando `REVOKE` remove permissões anteriormente concedidas a um usuário ou função.

📌 **Exemplo 1: Revogando permissão de SELECT**

```sql
REVOKE SELECT ON funcionarios FROM usuario1;
```

✅ **Explicação:** O usuário **usuario1 não poderá mais** executar consultas `SELECT` na tabela **funcionarios**.

📌 **Exemplo 2: Removendo todas as permissões de um usuário**

```sql
REVOKE ALL PRIVILEGES ON pedidos FROM gerente;
```

✅ **Explicação:** O usuário **gerente** perderá **todas as permissões** sobre a tabela **pedidos**.

📌 **Exemplo 3: Revogando apenas uma permissão específica**

```sql
REVOKE DELETE ON clientes FROM admin;
```

✅ **Explicação:** O usuário **admin não poderá mais excluir registros** da tabela **clientes**, mas manterá outras permissões.

🎯 Comandos

	Comando	Função
1	`GRANT`	Concede permissões a usuários ou funções
2	`REVOKE`	Revoga permissões concedidas anteriormente

▼ CRIAR usuário, revogar e conceder acesso

Seção vazia. Clique para adicionar conteúdo.

1. Criar um novo usuário

```sql
CREATE USER 'novo_usuario'@'localhost' IDENTIFIED BY 'senha123';
```

📌 **Explicação:**

- `'novo_usuario'@'localhost'` → Usuário só pode conectar localmente.
- **Se quiser que acesse de qualquer lugar:**

```sql
CREATE USER 'novo_usuario'@'%' IDENTIFIED BY 'senha123';
```

(`%` permite conexões remotas)

---

2. Conceder permissões ao usuário (GRANT)

```sql
GRANT ALL PRIVILEGES ON meu_banco.* TO 'novo_usuario'@'localhost';
```

📌 **Explicação:**

- `ALL PRIVILEGES` → Dá todas as permissões ao usuário.
- `meu_banco.*` → Concede para todas as tabelas do banco `meu_banco`.
- Para dar apenas algumas permissões, use:

```sql
GRANT SELECT, INSERT, UPDATE ON meu_banco.* TO 'novo_usuario'@'localhost';
```

---

3. Remover permissões do usuário (REVOKE)

```sql
REVOKE INSERT, UPDATE ON meu_banco.* FROM 'novo_usuario'@'localhost';
```

📌 **Explicação:**

- Remove apenas `INSERT` e `UPDATE`, mantendo outras permissões.

---

4. Excluir um usuário

```sql
DROP USER 'novo_usuario'@'localhost';
```

📌 **Explicação**:

- Remove o usuário e todas as permissões associadas.

---

5. Aplicar as mudanças

Depois de `GRANT` ou `REVOKE`, rode:

```sql
FLUSH PRIVILEGES;
```

📌 **Explicação**:

- Atualiza as permissões no MySQL sem precisar reiniciar o servidor.

Atividade: Gerenciamento de Usuários no MySQL

Objetivo:

Criar, gerenciar e excluir usuários no MySQL, além de conceder e revogar permissões utilizando os comandos **CREATE USER, GRANT, REVOKE** e **DROP USER**.

Siga as instruções:

1. **Criando um banco de dados para a prática:**
   - No MySQL, crie um banco de dados chamado `escola` e uma tabela `alunos` com alguns registros.

```sql
CREATE DATABASE escola;
USE escola;

CREATE TABLE alunos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    idade INT
);

INSERT INTO alunos (nome, idade) VALUES
('Ana Souza', 18),
('Carlos Lima', 19),
('Mariana Costa', 17);
```

2. **Criando um usuário com acesso restrito:**
   - Crie um usuário chamado `professor` com a senha `senha123`, mas sem permissões iniciais.

```sql
CREATE USER 'professor'@'localhost' IDENTIFIED BY 'senha123';

```

3. **Concedendo permissões específicas:**
   - Agora, conceda ao usuário `professor` permissão para **ler os dados da tabela alunos**.

```sql
GRANT SELECT ON escola.alunos TO 'professor'@'localhost';

```

   - **Teste a permissão:**
     - Faça login como `` `professor` `` e tente executar:

```sql
SELECT * FROM escola.alunos;

```

     - **Veja o que acontece se tentar fazer um `INSERT` - não é permitido, porque professor não tem acesso a INSERT**

4. **Ampliando as permissões:**
   - Agora, conceda ao `professor` permissão para adicionar e modificar registros na tabela `alunos`:

```sql
GRANT INSERT, UPDATE ON escola.alunos TO 'professor'@'localhost';

```

   - **Teste novamente:**
     - Insira um novo aluno e observe que agora é possível

```sql
INSERT INTO escola.alunos (nome, idade) VALUES ('Pedro Silva', 20);

```

5. **Removendo permissões:**
   - O administrador da escola decide que o professor **não pode mais modificar** os alunos.
   - Revogue a permissão de `UPDATE`:

```sql
REVOKE UPDATE ON escola.alunos FROM 'professor'@'localhost';

```

- **Teste novamente:**
  - Tente atualizar um aluno e veja o que acontece: não será permitido, porque ele não tem permissão de UPDATE

```sql
UPDATE escola.alunos SET idade = 21 WHERE nome = 'Pedro Silva';

```

6. **Excluindo o usuário**
   - Caso o colaborador não faça mais parte do quadro, remova seu acesso:

```sql
DROP USER 'professor'@'localhost';

```