Guia Completo de Comandos DDL no MariaDB

Este guia aborda a criação, alteração e exclusão de objetos do banco de dados. Usaremos um cenário prático: a criação de um banco de dados empresa com tabelas para setores e funcionarios.

1. Comandos para Bancos de Dados (DATABASE)

CREATE DATABASE

Cria um novo banco de dados, que é o contêiner para todas as suas tabelas, views e outros objetos.

Sintaxe:

CREATE DATABASE [IF NOT EXISTS] nome_do_banco;

- **IF NOT EXISTS**: Esta é uma cláusula de segurança extremamente útil. Ela instrui o MariaDB a criar o banco de dados *somente se* um banco de dados com o mesmo nome ainda não existir. Se já existir, o comando é ignorado sem gerar um erro.

Exemplo Prático:

-- Cria o banco de dados 'empresa' apenas se ele não existir.

CREATE DATABASE IF NOT EXISTS empresa;

-- Para começar a usar o banco de dados criado:

USE empresa;

DROP DATABASE

Exclui permanentemente um banco de dados e **todos os seus objetos** (tabelas, dados, etc.). Esta ação é irreversível.

Atenção: Use este comando com o máximo de cuidado. Uma vez executado, todos os dados são perdidos.

Sintaxe:

DROP DATABASE [IF EXISTS] nome_do_banco;

- **IF EXISTS**: Cláusula de segurança que instrui o MariaDB a excluir o banco de dados *somente se* ele existir. Isso evita erros em scripts caso o banco de dados já tenha sido removido.

Exemplo Prático:

```
-- Exclui o banco de dados 'empresa_backup' apenas se ele existir.

DROP DATABASE IF EXISTS empresa_backup;
```

---

2. Comandos para Tabelas (TABLE)

CREATE TABLE

Cria uma nova tabela dentro do banco de dados atualmente em uso. É aqui que definimos as colunas, seus tipos de dados e as regras (constraints).

Sintaxe Básica:

```
CREATE TABLE [IF NOT EXISTS] nome_da_tabela (
  nome_coluna1 tipo_dado [CONSTRAINTS],
  nome_coluna2 tipo_dado [CONSTRAINTS],
  ...
);
```

Exemplo Prático (Tabela setores):

```
CREATE TABLE IF NOT EXISTS setores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL UNIQUE,
  orcamento DECIMAL(12, 2)
);
```

- **IF NOT EXISTS**: Mesma lógica do CREATE DATABASE, previne erros se a tabela já existir.
- **id INT AUTO_INCREMENT PRIMARY KEY**: Uma constraint que define a coluna id como chave primária (identificador único e não nulo) e faz com que seu valor seja gerado automaticamente.
- **nome VARCHAR(100) NOT NULL UNIQUE**: Duas constraints. NOT NULL torna o preenchimento obrigatório e UNIQUE garante que não haverá dois setores com o mesmo nome.

Constraints (Restrições)

Constraints são regras aplicadas às colunas para garantir a integridade, precisão e confiabilidade dos dados.

- **PRIMARY KEY**: Identificador único de cada linha na tabela. Não pode ser nulo.

- **FOREIGN KEY**: Cria um vínculo entre duas tabelas, garantindo a integridade referencial.
- **NOT NULL**: A coluna não pode conter valores nulos.
- **UNIQUE**: Todos os valores na coluna devem ser diferentes.
- **DEFAULT valor**: Define um valor padrão caso nenhum seja fornecido.
- **CHECK (condição)**: Garante que todos os valores na coluna satisfaçam uma condição específica.

Exemplo Prático (Tabela funcionarios com FOREIGN KEY):

```
CREATE TABLE IF NOT EXISTS funcionarios (
  matricula INT PRIMARY KEY,
  nome_completo VARCHAR(255) NOT NULL,
  data_admissao DATE NOT NULL,
  salario DECIMAL(10, 2) CHECK (salario > 1400.00),
  cargo VARCHAR(100) DEFAULT 'Não definido',
  setor_id INT,

  CONSTRAINT fk_setor
    FOREIGN KEY (setor_id)
    REFERENCES setores(id)
);
```

- CHECK (salario > 1400.00): Impede a inserção de um salário menor ou igual a 1400.00.
- DEFAULT 'Não definido': Se o cargo não for informado, ele receberá este valor.
- CONSTRAINT fk_setor FOREIGN KEY (setor_id) REFERENCES setores(id): Esta é a chave estrangeira. Ela garante que todo setor_id inserido na tabela funcionarios **deve** existir na coluna id da tabela setores.

Ações em Chaves Estrangeiras: ON DELETE

Estas regras definem o que acontece na tabela "filha" (funcionarios) quando um registro correspondente na tabela "pai" (setores) é excluído.

- **ON DELETE SET NULL**: Se um setor for excluído, o campo setor_id dos funcionários que pertenciam a ele será atualizado para NULL. Isso mantém o registro do funcionário, mas o deixa "sem setor".

Exemplo de Sintaxe:

```
-- (Alterando a tabela criada anteriormente para adicionar a regra)

-- Primeiro, precisamos remover a constraint antiga

ALTER TABLE funcionarios DROP FOREIGN KEY fk_setor;


-- Agora, adicionamos a nova com ON DELETE SET NULL

ALTER TABLE funcionarios ADD CONSTRAINT fk_setor

  FOREIGN KEY (setor_id)

  REFERENCES setores(id)

  ON DELETE SET NULL;
```

- **ON DELETE CASCADE**: Se um setor for excluído, **todos os funcionários** que pertenciam a esse setor também serão **automaticamente excluídos**. É uma ação em cascata.

Atenção: CASCADE é muito poderoso e pode levar à perda de dados em massa se não for usado com cuidado.

Exemplo de Sintaxe:

```
-- (Supondo que removemos a constraint anterior)

ALTER TABLE funcionarios ADD CONSTRAINT fk_setor

  FOREIGN KEY (setor_id)

  REFERENCES setores(id)

  ON DELETE CASCADE;
```

---

3. ALTER TABLE: Modificando Tabelas Existentes

Permite modificar a estrutura de uma tabela já criada.

Adicionar Campo (ADD COLUMN)

```
ALTER TABLE funcionarios

ADD COLUMN email VARCHAR(100);
```

Renomear Campo (CHANGE COLUMN)

Para renomear, você deve usar a cláusula CHANGE COLUMN, que exige que você especifique o nome antigo, o nome novo e a definição completa do tipo de dado da coluna.

ALTER TABLE funcionarios

CHANGE COLUMN email email_corporativo VARCHAR(150) NOT NULL;

- Note que além de renomear, também alteramos o tipo (VARCHAR(150)) e adicionamos uma constraint (NOT NULL).

Alterar/Modificar Campo (MODIFY COLUMN)

Para alterar apenas o tipo de dado ou as constraints de uma coluna (sem mudar o nome), use MODIFY COLUMN.

ALTER TABLE funcionarios

MODIFY COLUMN salario DECIMAL(11, 2) CHECK (salario >= 1412.00);

- Aqui, aumentamos a precisão do salário e atualizamos a regra CHECK.

Excluir Campo (DROP COLUMN)

ALTER TABLE funcionarios

DROP COLUMN cargo;

Adicionar e Remover Constraints

Você pode adicionar ou remover regras a qualquer momento.

- **Adicionar Constraint:** Vamos adicionar uma constraint UNIQUE ao email.

ALTER TABLE funcionarios

ADD CONSTRAINT uk_email_corporativo UNIQUE (email_corporativo);

- **Remover Constraint:** Para remover, você precisa saber o nome da constraint. Para PRIMARY KEY e FOREIGN KEY, os nomes são explícitos ou gerados pelo sistema.

-- Removendo a constraint UNIQUE que acabamos de criar

ALTER TABLE funcionarios DROP CONSTRAINT uk_email_corporativo;

-- Removendo a chave estrangeira (o nome foi 'fk_setor')

ALTER TABLE funcionarios DROP FOREIGN KEY fk_setor;

-- Removendo a chave primária (não precisa de nome, só pode haver uma)

ALTER TABLE funcionarios DROP PRIMARY KEY;

---

4. DROP TABLE e TRUNCATE TABLE

DROP TABLE

Exclui permanentemente uma tabela, sua estrutura e todos os seus dados. É irreversível.

Sintaxe:

DROP TABLE [IF EXISTS] nome_da_tabela;

Exemplo:

DROP TABLE IF EXISTS funcionarios_demitidos;

TRUNCATE TABLE

Remove **todos os registros** de uma tabela de forma muito rápida, mas **mantém a estrutura da tabela** (colunas, tipos, constraints). É uma operação DDL, o que significa que é muito mais rápida que DELETE FROM funcionarios (que é DML), mas geralmente não pode ser desfeita (rollback). O contador de AUTO_INCREMENT também é reiniciado.

Sintaxe:

TRUNCATE TABLE nome_da_tabela;

Exemplo: Imagine que você quer limpar a tabela de logs para começar um novo ciclo, sem ter que recriar a tabela.

TRUNCATE TABLE logs;

-- A tabela 'logs' agora está vazia, mas sua estrutura continua intacta.

Comando	O que faz?	Estrutura	Dados	Auto Increment	Velocidade
DROP TABLE	Exclui tudo	Removida	Removidos	Removido	Rápida
TRUNCATE TABLE	Limpa os dados	Mantida	Removidos	Reiniciado	Muito Rápida
DELETE FROM ...	Remove linhas (DML)	Mantida	Removidos	Mantido	Lenta (linha por linha)