Cheatsheet de Comandos PostgreSQL

1. Estrutura e Definição de Dados (DDL)

CREATE TABLE

O comando `CREATE TABLE` é a base para construir o seu banco de dados. Ele não apenas define as colunas, mas também estabelece regras e restrições para garantir a integridade dos dados.

O que você pode fazer:

- **Definir colunas e tipos de dados:** Especifique o nome de cada coluna e o tipo de dado que ela armazenará (e.g., `INT`, `VARCHAR(255)`, `DATE`).
- **Chaves Primárias (`PRIMARY KEY`):** Defina uma coluna como chave primária para garantir que cada registro seja único. É a espinha dorsal de qualquer tabela.
- **Chaves Estrangeiras (`FOREIGN KEY`):** Crie links entre tabelas, garantindo que os dados em uma tabela referenciem dados válidos em outra. Isso é fundamental para relações.
- **Restrições (`CONSTRAINT`):** Adicione regras, como `NOT NULL` (impedir valores nulos), `UNIQUE` (garantir valores únicos) e `CHECK` (validar dados com uma condição específica).
- **Valores Padrão (`DEFAULT`):** Defina um valor que será inserido automaticamente se nenhum valor for fornecido para a coluna.

Exemplo:CREATE TABLE produtos (

id SERIAL PRIMARY KEY,

nome VARCHAR(255) NOT NULL UNIQUE,

preco NUMERIC(10, 2) DEFAULT 0.00,

categoria_id INT,

CONSTRAINT fk_categoria

FOREIGN KEY (categoria_id)

REFERENCES categorias (id)

ON DELETE SET NULL

);

- `SERIAL`: Cria uma sequência de números inteiros para a chave primária `id`, incrementando-o automaticamente.
- `NOT NULL UNIQUE`: O nome do produto não pode ser nulo e deve ser único.
- `NUMERIC(10, 2)`: `preco` é um tipo numérico com um total de 10 dígitos, 2 dos quais estão após a vírgula.
- `FOREIGN KEY (...) REFERENCES ...`: `categoria_id` se conecta à tabela `categorias`.
- `ON DELETE SET NULL`: Se a categoria for excluída, o `categoria_id` dos produtos associados se tornará `NULL` em vez de apagar os produtos.

ALTER TABLE

O comando `ALTER TABLE` permite que você modifique a estrutura de uma tabela já existente. É extremamente útil para evoluir seu banco de dados sem precisar recriar tudo.

O que você pode fazer:

- **Adicionar/Remover colunas:** Adicione uma nova coluna (`ADD COLUMN`) ou remova uma existente (`DROP COLUMN`).
- **Alterar tipo de dados:** Mude o tipo de dado de uma coluna (`ALTER COLUMN TYPE`).
- **Adicionar/Remover/Alterar restrições:** Adicione uma chave primária, uma chave estrangeira, uma restrição de nulidade (`SET NOT NULL`), ou remova uma restrição (`DROP CONSTRAINT`).
- **Renomear:** Mude o nome de uma tabela (`RENAME TO`) ou de uma coluna (`RENAME COLUMN`).

Exemplos:-- Adicionar uma nova coluna 'data_cadastro'

ALTER TABLE clientes

ADD COLUMN data_cadastro DATE DEFAULT CURRENT_DATE;

-- Alterar o tipo da coluna 'preco'

ALTER TABLE produtos

ALTER COLUMN preco TYPE NUMERIC(12, 4);

-- Adicionar uma restrição de chave estrangeira

ALTER TABLE pedidos

ADD CONSTRAINT fk_cliente

FOREIGN KEY (cliente_id)

REFERENCES clientes (id);

2. Manipulação de Dados (DML)

INSERT INTO

O comando `INSERT INTO` é usado para adicionar novas linhas de dados a uma tabela.

Exemplo:INSERT INTO clientes (nome, email)

VALUES ('João da Silva', 'joao.silva@email.com');

UPDATE

O comando `UPDATE` modifica dados existentes. O uso da cláusula `WHERE` é crucial para especificar quais linhas devem ser alteradas; sem ela, todos os registros da tabela serão modificados.

Exemplo:UPDATE produtos

SET preco = 25.50

WHERE id = 3;

DELETE FROM

O comando `DELETE FROM` remove linhas de uma tabela. Assim como no `UPDATE`, o uso da cláusula `WHERE` é essencial para evitar a exclusão de todos os dados.

Exemplo:DELETE FROM clientes

WHERE nome = 'Maria';

SELECT

O comando `SELECT` é a ferramenta de consulta mais poderosa. Ele permite que você recupere dados e os combine de inúmeras formas.

O que você pode fazer:

- **Consultas simples:** Selecionar colunas específicas (`SELECT nome, preco FROM produtos`).
- **Filtrar dados:** Usar a cláusula `WHERE` para aplicar filtros (`WHERE preco > 100`).
- **Agrupar e agregar:** Usar `GROUP BY` e funções como `COUNT()`, `SUM()`, `AVG()` para gerar resumos de dados.
- **Combinar tabelas (`JOIN`):** Unir dados de múltiplas tabelas com `INNER JOIN`, `LEFT JOIN`, etc.
- **Ordenar resultados:** Usar `ORDER BY` para definir a ordem dos resultados (`ORDER BY preco DESC`).

Exemplo:-- Listar o nome do cliente e o título dos livros que ele pegou emprestado

SELECT c.nome, l.titulo

FROM clientes c

INNER JOIN emprestimos e ON c.id = e.cliente_id

INNER JOIN livros l ON e.livro_id = l.id

WHERE c.nome = 'João da Silva';