Guia de Estudo Completo: Comandos SQL Fundamentais

Criado pelo Gemini

Setembro de 2025

Sumário

1 Linguagem de Definição de Dados (DDL) 3
- 1.1 CREATE TABLE . . . 3
- 1.2 ALTER TABLE . . . 3
- 1.3 DROP TABLE . . . 4

2 Linguagem de Manipulação de Dados (DML) 4
- 2.1 INSERT INTO . . . 4
- 2.2 UPDATE . . . 4
- 2.3 DELETE . . . 4

3 Consultas de Dados com SELECT 5
- 3.1 Estrutura Básica e Filtros (WHERE) . . . 5
- 3.2 Ordenação (ORDER BY) . . . 5
- 3.3 Junção de Tabelas (JOINs) . . . 5
- 3.4 Agrupamento e Funções de Agregação (GROUP BY) . . . 6

4 Functions (Funções) 6

5 Triggers (Gatilhos) 7

6 Exercícios Práticos 8
- 6.1 Nível Básico . . . 11
- 6.2 Nível Intermediário . . . 11
- 6.3 Nível Avançado . . . 12

7 Conclusão 12

Introdução ao SQL

SQL (Structured Query Language) é a linguagem padrão para gerenciar e manipular bancos de dados relacionais. Com ela, é possível criar, consultar, atualizar e administrar as estruturas de dados e a informação contida nelas.

Este guia aborda os comandos SQL mais importantes, desde a criação de tabelas e consultas básicas até conceitos mais avançados como procedures, triggers e functions, essenciais para qualquer profissional de dados.

Contexto dos Exemplos: Para os exemplos, usaremos um esquema de banco de dados para uma empresa fictícia, com as seguintes tabelas:

- `Departamentos`: `[ID_Depto]`, `[Nome_Depto]`
- `Funcionarios`: `[ID_Func]`, `[Nome]`, `[Sobrenome]`, `[ID_Depto]`, `[Salario]`, `[Data_Admissao]`
- `Projetos`: `[ID_Proj]`, `[Nome_Proj]`, `[ID_Depto_Resp]`

---

1 Linguagem de Definição de Dados (DDL)

DDL (Data Definition Language) lida com a estrutura do banco de dados, ou seja, a criação e modificação de objetos como tabelas, índices e constraints.

1.1 CREATE TABLE

Cria uma nova tabela no banco de dados. É preciso especificar o nome das colunas e seus respectivos tipos de dados.

Exemplo Prático: Criando a tabela de Funcionários

```sql
CREATE TABLE Funcionarios (
    ID_Func INT PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    Sobrenome VARCHAR(50) NOT NULL,
    Salario DECIMAL(10, 2),
    Data_Admissao DATE,
    ID_Depto INT,
    FOREIGN KEY (ID_Depto) REFERENCES Departamentos(ID_Depto)
);
```

1.2 ALTER TABLE

Modifica a estrutura de uma tabela existente. Pode ser usado para adicionar, remover ou modificar colunas.

Exemplo Prático: Adicionando uma coluna

```sql
-- Adiciona uma coluna para armazenar o email do funcion rio
ALTER TABLE Funcionarios
ADD COLUMN Email VARCHAR(100);
```

1.3 DROP TABLE

Remove permanentemente uma tabela, incluindo todos os seus dados.

Exemplo Prático: Removendo uma tabela

```sql
-- CUIDADO: Este comando    irrevers vel!
DROP TABLE Projetos;
```

2 Linguagem de Manipulação de Dados (DML)

DML (Data Manipulation Language) é usada para gerenciar os dados dentro das tabelas.

2.1 INSERT INTO

Adiciona novas linhas (registros) a uma tabela.

Exemplo Prático: Inserindo um novo departamento

```sql
INSERT INTO Departamentos (ID_Depto, Nome_Depto)
VALUES (1, 'Tecnologia'), (2, 'Recursos Humanos');
```

2.2 UPDATE

Modifica registros existentes em uma tabela. A cláusula `WHERE` é crucial para especificar quais linhas devem ser atualizadas.

Exemplo Prático: Aumentando um salário

```sql
-- Aumenta o sal rio do funcion rio com ID 101 em 10%
UPDATE Funcionarios
SET Salario = Salario * 1.10
WHERE ID_Func = 101;
```

2.3 DELETE

Remove registros de uma tabela. Assim como no `UPDATE`, a cláusula `WHERE` é fundamental para evitar a exclusão de todos os dados.

Exemplo Prático: Removendo um funcionário

```sql
DELETE FROM Funcionarios
WHERE ID_Func = 102;
```

3 Consultas de Dados com SELECT

O comando `SELECT` é usado para buscar dados de uma ou mais tabelas. É o comando mais utilizado em SQL.

3.1 Estrutura Básica e Filtros (WHERE)

Exemplo Prático: Selecionando funcionários de um departamento

```sql
-- Seleciona o nome e o sal rio dos funcion rios do departamento
   de Tecnologia (ID 1)
-- que ganham mais de 5000
SELECT Nome, Salario
FROM Funcionarios
WHERE ID_Depto = 1 AND Salario > 5000;
```

3.2 Ordenação (ORDER BY)

Exemplo Prático: Listando funcionários por salário

```sql
-- Lista todos os funcion rios em ordem decrescente de sal rio
SELECT Nome, Sobrenome, Salario
FROM Funcionarios
ORDER BY Salario DESC;
```

3.3 Junção de Tabelas (JOINs)

`JOIN`s são usados para combinar linhas de duas ou more tabelas com base em uma coluna relacionada entre elas.

- **INNER JOIN**: Retorna registros que têm valores correspondentes em ambas as tabelas.
- **LEFT JOIN**: Retorna todos os registros da tabela da esquerda e os registros correspondentes da tabela da direita. Se não houver correspondência, o resultado é NULL no lado direito.
- **RIGHT JOIN**: O oposto do LEFT JOIN.

Exemplo Prático: Listando funcionários e seus departamentos

```sql
-- Usa INNER JOIN para combinar as tabelas Funcionarios e
   Departamentos
SELECT
    F.Nome,
    F.Sobrenome,
    D.Nome_Depto
FROM
    Funcionarios F
INNER JOIN
    Departamentos D ON F.ID_Depto = D.ID_Depto;
```

3.4 Agrupamento e Funções de Agregação (GROUP BY)

`GROUP BY` agrupa linhas que têm os mesmos valores em colunas especificadas. É frequentemente usado com funções de agregação como `COUNT()`, `SUM()`, `AVG()`, `MAX()` e `MIN()`.

Exemplo Prático: Calculando o salário médio por departamento

```sql
SELECT
    D.Nome_Depto,
    AVG(F.Salario) AS Salario_Medio,
    COUNT(F.ID_Func) AS Numero_Funcionarios
FROM
    Funcionarios F
JOIN
    Departamentos D ON F.ID_Depto = D.ID_Depto
GROUP BY
    D.Nome_Depto
HAVING
    COUNT(F.ID_Func) > 5; -- HAVING filtra ap s o agrupamento
```

4 Functions (Funções)

Funções são blocos de código reutilizáveis que realizam uma operação específica e retornam um valor. Elas podem ser escalares (retornam um único valor) ou de tabela (retornam uma tabela).

Exemplo Prático: Criando uma função para calcular o bônus anual A sintaxe pode variar um pouco entre sistemas de banco de dados (SQL Server, PostgreSQL, MySQL). Este exemplo usa uma sintaxe comum.

```
CREATE FUNCTION CalcularBonusAnual (
    @Salario DECIMAL(10, 2),
    @AnosDeServico INT
)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @Bonus DECIMAL(10, 2);

    IF @AnosDeServico > 5
        SET @Bonus = @Salario * 0.20; -- 20% de b nus
    ELSE
        SET @Bonus = @Salario * 0.10; -- 10% de b nus

    RETURN @Bonus;
END;
```

5 Triggers (Gatilhos)

Um Trigger é um tipo especial de procedimento armazenado que é executado automaticamente quando um evento DML (`INSERT`, `UPDATE`, `DELETE`) ocorre em uma tabela específica.

São úteis para manter a integridade dos dados, auditar alterações ou automatizar lógica de negócios complexa.

Exemplo Prático: Trigger para auditar alterações de salário

```sql
-- Primeiro, crie uma tabela para armazenar o log de auditoria
CREATE TABLE AuditoriaSalarios (
    ID_Log INT PRIMARY KEY IDENTITY(1,1),
    ID_Func INT,
    Salario_Antigo DECIMAL(10, 2),
    Salario_Novo DECIMAL(10, 2),
    Data_Alteracao DATETIME DEFAULT GETDATE()
);

-- Agora, crie o Trigger na tabela Funcionarios
CREATE TRIGGER trg_AuditoriaSalario
ON Funcionarios
AFTER UPDATE
AS
BEGIN
    -- Verifica se a coluna Salario foi de fato atualizada
    IF UPDATE(Salario)
    BEGIN
        INSERT INTO AuditoriaSalarios (ID_Func, Salario_Antigo,
    Salario_Novo)
        SELECT
            d.ID_Func,      -- 'd' vem da tabela virtual 'deleted'
            d.Salario,      -- Sal rio antigo
            i.Salario       -- 'i' vem da tabela virtual 'inserted'
    (Sal rio novo)
        FROM
            deleted d
        JOIN
            inserted i ON d.ID_Func = i.ID_Func;
    END
END;
```

Explicação: As tabelas virtuais `inserted` e `deleted` contêm as linhas afetadas pela operação. Em um `UPDATE`, `deleted` contém os dados antigos e `inserted` contém os novos.

6 Exercícios Práticos

Use o esquema de banco de dados (`Departamentos`, `Funcionarios`, `Projetos`) para resolver os seguintes desafios.

Script SQL para Criação do Schema

Este documento contém o código SQL completo para gerar as tabelas e popular os dados necessários para a resolução dos exercícios dos guias de estudo.

Parte 1: Criação das Tabelas (DDL)

```sql
--
    ================================================================================

-- SCRIPT DE CRIACAO DAS TABELAS (DDL)
-- A ordem de criacao e importante devido as chaves estrangeiras.
--
    ================================================================================


-- 1. Tabela de Departamentos
-- Esta tabela armazena os diferentes departamentos da empresa.
-- E a tabela "pai" para Funcionarios e Projetos.
CREATE TABLE Departamentos (
    ID_Depto INT PRIMARY KEY,
    Nome_Depto VARCHAR(100) NOT NULL UNIQUE
);

-- 2. Tabela de Funcionarios
-- Armazena os dados dos funcionarios.
-- Possui uma chave estrangeira (FOREIGN KEY) que se conecta a
    Departamentos.
CREATE TABLE Funcionarios (
    ID_Func INT PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    Sobrenome VARCHAR(50) NOT NULL,
    Salario DECIMAL(10, 2) CHECK (Salario > 0),
    Data_Admissao DATE NOT NULL,
    ID_Depto INT,
    FOREIGN KEY (ID_Depto) REFERENCES Departamentos(ID_Depto)
);

-- 3. Tabela de Projetos
-- Armazena informacoes sobre os projetos da empresa.
-- Cada projeto e responsabilidade de um departamento.
CREATE TABLE Projetos (
    ID_Proj INT PRIMARY KEY,
    Nome_Proj VARCHAR(150) NOT NULL,
    ID_Depto_Resp INT,
    FOREIGN KEY (ID_Depto_Resp) REFERENCES Departamentos(ID_Depto)
);
```

Parte 2: Inserção de Dados de Exemplo (DML)

```sql
-- ============================================================================
-- SCRIPT DE INSERCAO DE DADOS (DML)
-- Populando as tabelas com dados ficticios para os exercicios.
-- ============================================================================

-- Inserindo dados na tabela Departamentos
INSERT INTO Departamentos (ID_Depto, Nome_Depto) VALUES
(1, 'Tecnologia'),
(2, 'Recursos Humanos'),
(3, 'Marketing'),
(4, 'Vendas'),
(5, 'Financeiro'),
(6, 'Diretoria'); -- Departamento sem funcionarios para o exercicio

-- Inserindo dados na tabela Funcionarios
INSERT INTO Funcionarios (ID_Func, Nome, Sobrenome, Salario,
    Data_Admissao, ID_Depto) VALUES
(101, 'Ana', 'Silva', 7500.00, '2022-03-15', 1),
(102, 'Bruno', 'Costa', 8200.00, '2021-11-20', 1),
(103, 'Carla', 'Martins', 5500.00, '2023-01-10', 2),
(104, 'Daniel', 'Pereira', 6200.00, '2023-05-25', 3),
(105, 'Eduarda', 'Lima', 9100.00, '2020-08-01', 4),
(106, 'Fabio', 'Ribeiro', 4800.00, '2023-02-18', 2),
(107, 'Gabriela', 'Alves', 11500.00, '2019-07-30', 1),
(108, 'Heitor', 'Mendes', 7100.00, '2022-09-05', 4),
(109, 'Isabela', 'Nunes', 6800.00, '2021-12-12', 3),
(110, 'Joao', 'Santos', 8900.00, '2022-06-01', 5);

-- Inserindo dados na tabela Projetos
INSERT INTO Projetos (ID_Proj, Nome_Proj, ID_Depto_Resp) VALUES
(1, 'Desenvolvimento do Novo App Mobile', 1),
(2, 'Campanha de Marketing Digital Q4', 3),
(3, 'Reestruturacao do Plano de Carreira', 2),
(4, 'Expansao de Vendas para a Regiao Sul', 4),
(5, 'Migracao do Servidor para a Nuvem', 1),
(10, 'Projeto a ser deletado', 5);
```

Verificação

Após executar os scripts, você pode rodar os seguintes comandos `SELECT` para verificar se os dados foram inseridos corretamente.

Comandos de Verificação

```sql
-- Verificar Departamentos
SELECT * FROM Departamentos;

-- Verificar Funcionarios
SELECT * FROM Funcionarios;

-- Verificar Projetos
SELECT * FROM Projetos;
```

6.1 Nível Básico

[label=**1.**]

1. **Listar Todos os Funcionários:** Escreva uma query para selecionar o nome completo (nome e sobrenome) e o salário de todos os funcionários, ordenados pelo nome em ordem alfabética.

2. **Inserir um Novo Departamento:** Escreva um comando para inserir um novo departamento chamado "Marketing"com `ID_Depto` = 3.

3. **Atualizar Salário:** Escreva um comando para dar um aumento de 15% para todos os funcionários do departamento de "Recursos Humanos"(`ID_Depto` = 2).

4. **Funcionários Recentes:** Selecione os funcionários que foram admitidos após '2023-01-01'.

5. **Deletar um Projeto:** Escreva um comando para deletar o projeto com `ID_Proj` = 10.

6.2 Nível Intermediário

[label=**2.**]

6. **Funcionários e Nomes dos Departamentos:** Escreva uma query que liste o nome completo do funcionário e o nome do seu respectivo departamento. Use um `JOIN`.

7. **Contagem de Funcionários por Departamento:** Escreva uma query que mostre o nome de cada departamento e o número de funcionários que trabalham nele.

8. **Salário Máximo por Departamento:** Mostre o maior salário pago em cada departamento. A saída deve conter o nome do departamento e o valor do salário máximo.

9. **Departamentos Sem Funcionários:** Escreva uma query para encontrar todos os departamentos que não possuem nenhum funcionário alocado. (Dica: um `LEFT JOIN` é ideal aqui).

10. **Funcionários com Salário Acima da Média:** Liste os funcionários cujo salário é maior que a média de todos os salários da empresa. (Dica: use uma subquery na cláusula `WHERE`).

6.3 Nível Avançado

[label=**3.**]

11. **Ranking de Salários:** Liste os 3 maiores salários da empresa, junto com o nome do funcionário e do departamento.

12. **Projetos e Departamentos Responsáveis:** Escreva uma query que liste o nome de cada projeto e o nome do departamento responsável por ele.

13. **Criar uma View:** Crie uma `VIEW` chamada `v_DetalhesFuncionario` que exiba o ID do funcionário, nome completo, salário, data de admissão e nome do departamento.

14. **Criar uma Function:** Crie uma função chamada `fn_TempoDeCasa` que recebe a data de admissão de um funcionário como parâmetro e retorna o número de anos completos que ele tem na empresa.

15. **Criar um Trigger:** Crie um trigger que impeça a exclusão (`DELETE`) de um departamento se ainda existirem funcionários alocados a ele. Se uma tentativa de exclusão ocorrer, o trigger deve gerar um erro com uma mensagem como "Não é possível excluir departamento com funcionários ativos."

7 Conclusão

Este guia cobriu os pilares fundamentais do SQL, desde a manipulação básica de dados até a automação com triggers e a modularização com functions. A maestria em SQL vem com a prática contínua. Utilize este material como referência para construir consultas cada vez mais eficientes e robustas em seus projetos de banco de dados.