Gerenciamento de Bancos de Dados com SQL

Comandos Básicos de Consulta, Inserção e Joins

Página 1: Capa

Autor: Manus AI

Novembro de 2025

Página 2: Introdução a Bancos de Dados Relacionais

Um **Banco de Dados Relacional (RDBMS)** é um sistema de gerenciamento de dados que armazena informações em tabelas, onde os dados são organizados em linhas e colunas. A característica fundamental desses bancos é a relação entre as tabelas, estabelecida por meio de chaves.

Conceitos Fundamentais:

Conceito	Descrição
Tabela	Estrutura básica que armazena dados. É composta por linhas e colunas.
Coluna (Campo)	Representa um atributo específico do dado (ex: nome, idade, preço).
Linha (Registro)	Representa uma única ocorrência de dados na tabela (ex: um cliente, um produto).
Chave Primária (PK)	Coluna ou conjunto de colunas que identifica unicamente cada linha na tabela.
Chave Estrangeira (FK)	Coluna que estabelece um vínculo entre duas tabelas, referenciando a Chave Primária de outra tabela.

SQL (Structured Query Language) é a linguagem padrão utilizada para gerenciar e manipular dados em RDBMS. É a linguagem que permite consultar, inserir, atualizar e excluir dados.


Diagrama de Banco de Dados Relacional

Página 3: Comandos DDL e DML (Visão Geral)

A linguagem SQL é dividida em subconjuntos, sendo os mais importantes o DDL e o DML.

DDL (Data Definition Language)

Usado para **definir** a estrutura do banco de dados (`CREATE` , `ALTER` , `DROP`).

DML (Data Manipulation Language)

Usado para **manipular** os dados dentro das tabelas. Nosso foco principal será nos comandos DML:

Comando	Função
`SELECT`	Recupera dados de uma ou mais tabelas.
`INSERT`	Adiciona novos registros (linhas) a uma tabela.
`UPDATE`	Modifica dados existentes em uma tabela.
`DELETE`	Remove registros (linhas) de uma tabela.

Página 4: O Comando SELECT (Consulta Básica)

O comando `SELECT` é o mais fundamental do SQL, utilizado para recuperar dados do banco.

Sintaxe Básica

Para selecionar colunas específicas: `SELECT coluna1, coluna2 FROM tabela;` Para selecionar **todas** as colunas: `SELECT * FROM tabela;`

Exemplo Prático

Suponha uma tabela `Produtos` (`ID`, `Nome`, `Preco`, `Estoque`).

Consulta 1: Selecionar todas as informações dos produtos.

```
SELECT *
FROM Produtos;
```

Consulta 2: Selecionar apenas o nome e o preço dos produtos.

```
SELECT Nome, Preco
FROM Produtos;
```

Página 5: Filtragem e Ordenação de Dados

Filtragem com WHERE

A cláusula `WHERE` é usada para filtrar os registros, retornando apenas as linhas que satisfazem uma condição.

Sintaxe: `SELECT ... FROM tabela WHERE condicao;`

Operadores Comuns: `=`, `>`, `<`, `AND`, `OR`, `NOT`, `BETWEEN`, `LIKE`.

Consulta 3: Selecionar produtos com preço superior a 50.00 e em estoque.

```
SELECT Nome, Preco
FROM Produtos
WHERE Preco > 50.00 AND Estoque > 0;
```

Ordenação com ORDER BY

A cláusula `ORDER BY` classifica o resultado em ordem crescente (`ASC`, padrão) ou decrescente (`DESC`).

Consulta 4: Selecionar todos os produtos e ordená-los pelo preço, do mais caro para o mais barato.

```
SELECT Nome, Preco
FROM Produtos
ORDER BY Preco DESC;
```

Página 6: Manipulação de Dados: INSERT

O comando `INSERT` é usado para adicionar novos registros (linhas) a uma tabela.

Sintaxe

```
INSERT INTO nome_da_tabela (coluna1, coluna2, ...)
VALUES (valor1, valor2, ...);
```

Exemplo Prático

Tabela: `Clientes` (`ID`, `Nome`, `Email`, `Cidade`)

Consulta 5: Inserir um novo cliente.

```
INSERT INTO Clientes (Nome, Email, Cidade)
VALUES ('Ana Silva', 'ana.silva@email.com', 'São Paulo');
```

Página 7: Manipulação de Dados: UPDATE e DELETE

O Comando UPDATE

Modifica dados existentes em uma tabela.

Sintaxe:

```
UPDATE nome_da_tabela
SET coluna1 = novo_valor1, coluna2 = novo_valor2, ...
WHERE condicao;
```

Consulta 6: Atualizar a cidade do cliente ‘Ana Silva’.

```
UPDATE Clientes
SET Cidade = 'Rio de Janeiro'
WHERE Nome = 'Ana Silva';
```

O Comando DELETE

Remove registros (linhas) de uma tabela.

Sintaxe:

```
DELETE FROM nome_da_tabela
WHERE condicao;
```

Consulta 7: Excluir o cliente ‘Ana Silva’.

```
DELETE FROM Clientes
WHERE Nome = 'Ana Silva';
```

⚠ AVISO DE SEGURANÇA CRÍTICO

SEMPRE utilize a cláusula `WHERE` com os comandos `UPDATE` e `DELETE`. A omissão do `WHERE` afeta **TODOS** os registros.

Página 8: Introdução aos JOINS

O comando `JOIN` é essencial para combinar linhas de duas ou mais tabelas com base em uma coluna relacionada (Chave Estrangeira).

Tipos de JOINS Principais

Tipo de JOIN	Descrição
INNER JOIN	Retorna apenas as linhas que têm correspondência em **ambas** as tabelas.
LEFT JOIN	Retorna todas as linhas da tabela da esquerda e as correspondentes da direita (NULL se não houver).
RIGHT JOIN	Retorna todas as linhas da tabela da direita e as correspondentes da esquerda (NULL se não houver).
FULL JOIN	Retorna todas as linhas quando há uma correspondência em uma das tabelas.

Página 9: O Comando INNER JOIN

O `INNER JOIN` retorna apenas os registros que possuem valores correspondentes em ambas as tabelas.

Sintaxe

```
SELECT coluna1, coluna2, ...
FROM TabelaA
INNER JOIN TabelaB
ON TabelaA.coluna_comum = TabelaB.coluna_comum;
```

Exemplo Prático: Listando Pedidos com Nomes de Clientes

- **TabelaA:** `Pedidos` (`ID_Pedido`, `ID_Cliente`)
- **TabelaB:** `Clientes` (`ID_Cliente`, `Nome_Cliente`)

Consulta 8: Listar o ID do pedido e o nome do cliente que o fez.

```
SELECT P.ID_Pedido, C.Nome_Cliente
FROM Pedidos AS P
INNER JOIN Clientes AS C
ON P.ID_Cliente = C.ID_Cliente;
```

Página 10: O Comando LEFT JOIN

O `LEFT JOIN` retorna todos os registros da tabela da esquerda, mesmo que não haja correspondência na tabela da direita.

Sintaxe

```
SELECT coluna1, coluna2, ...
FROM TabelaA
LEFT JOIN TabelaB
ON TabelaA.coluna_comum = TabelaB.coluna_comum;
```

Exemplo Prático: Clientes e Seus Pedidos (Incluindo Clientes Sem Pedidos)

Consulta 9: Listar todos os clientes e, se houver, seus pedidos.

```
SELECT C.Nome_Cliente, P.ID_Pedido
FROM Clientes AS C
LEFT JOIN Pedidos AS P
ON C.ID_Cliente = P.ID_Cliente;
```

Resultado: Clientes sem pedidos terão `NULL` na coluna `ID_Pedido`.

Página 11: Funções de Agregação e Agrupamento

As **Funções de Agregação** (`COUNT`, `SUM`, `AVG`, `MIN`, `MAX`) realizam um cálculo em um conjunto de linhas.

Agrupamento com GROUP BY

A cláusula `GROUP BY` agrupa linhas com os mesmos valores em colunas especificadas.

Consulta 10: Contar quantos produtos existem em cada categoria.

```sql
SELECT Categoria, COUNT(ID) AS Total_Produtos
FROM Produtos
GROUP BY Categoria;
```

Filtragem de Grupos com HAVING

A cláusula `HAVING` filtra grupos criados pelo `GROUP BY`.

Consulta 11: Contar categorias que possuem mais de 10 produtos.

```sql
SELECT Categoria, COUNT(ID) AS Total_Produtos
FROM Produtos
GROUP BY Categoria
HAVING COUNT(ID) > 10;
```

Página 12: Subconsultas (Subqueries)

Uma **Subconsulta** é uma consulta `SELECT` aninhada dentro de outra consulta SQL, usada para retornar dados que serão usados pela consulta principal.

Consulta 12: Selecionar todos os produtos cujo preço é superior à média de preço de todos os produtos.

```sql
SELECT Nome, Preco
FROM Produtos
WHERE Preco > (SELECT AVG(Preco) FROM Produtos);
```

Página 13: Conclusão e Próximos Passos

O SQL é a linguagem essencial para a manipulação de dados em RDBMS. Dominar `SELECT`, `INSERT`, `UPDATE`, `DELETE` e os `JOINS` é o alicerce para qualquer profissional de TI que lida com dados.

Próximos Passos para Aprofundamento:

1. **Otimização de Consultas:** Aprenda a usar `EXPLAIN` para analisar e otimizar a performance.
2. **Stored Procedures e Functions:** Crie blocos de código SQL reutilizáveis.
3. **Transações:** Estude o conceito de transações (ACID) para garantir a integridade dos dados.

A prática constante é a chave para a fluência em SQL.