BANCO DE DADOS ADMINISTRATIVO

Linguagem SQL e Manipulação de Dados

Conceitos Básicos de SQL

SQL, que significa Structured Query Language (Linguagem de Consulta Estruturada), é uma linguagem de programação utilizada para gerenciar e manipular bancos de dados relacionais. É amplamente adotada devido à sua simplicidade e eficiência na execução de operações complexas de banco de dados. Vamos explorar o que é SQL, a estrutura básica de uma consulta SQL e os comandos principais: SELECT, INSERT, UPDATE e DELETE.

O que é SQL?

SQL é uma linguagem padrão usada para interagir com bancos de dados relacionais. Ela permite que usuários criem, leiam, atualizem e excluam dados armazenados em tabelas de um banco de dados. SQL é essencialmente declarativa, o que significa que os usuários especificam o que desejam fazer com os dados, e o sistema de gerenciamento de banco de dados (SGBD) determina a melhor maneira de executar essas operações.

SQL é dividida em várias sub-linguagens, incluindo:

- **DDL (Data Definition Language):** Para definir e gerenciar a estrutura dos dados (e.g., CREATE, ALTER, DROP).
- **DML (Data Manipulation Language):** Para manipular os dados (e.g., SELECT, INSERT, UPDATE, DELETE).

- **DCL (Data Control Language):** Para controlar o acesso aos dados (e.g., GRANT, REVOKE).
- **TCL (Transaction Control Language):** Para gerenciar transações (e.g., COMMIT, ROLLBACK).

Estrutura Básica de uma Consulta SQL

A estrutura básica de uma consulta SQL envolve o uso de cláusulas que especificam as operações a serem realizadas nos dados. A consulta SQL mais comum é a consulta SELECT, que é usada para recuperar dados de uma ou mais tabelas. A estrutura básica de uma consulta SELECT é:

```
SELECT coluna1, coluna2, ...
FROM tabela
WHERE condição
ORDER BY coluna1, coluna2, ...;
```


- **SELECT:** Especifica as colunas que você deseja recuperar.
- **FROM:** Especifica a tabela de onde os dados serão recuperados.
- **WHERE:** (Opcional) Filtra os registros com base em uma condição.
- **ORDER BY:** (Opcional) Ordena os resultados com base em uma ou mais colunas.

Comandos Principais: SELECT, INSERT, UPDATE, DELETE

1. SELECT: O comando SELECT é utilizado para recuperar dados de um banco de dados. É a operação mais comum e poderosa em SQL.

```
SELECT nome, idade, cidade
FROM usuários
WHERE idade > 18
ORDER BY nome;
```

Este exemplo seleciona as colunas nome, idade e cidade da tabela usuários, onde a idade é maior que 18, e ordena os resultados pelo nome.

2. INSERT: O comando INSERT é utilizado para adicionar novos registros em uma tabela.

```
INSERT INTO usuários (nome, idade, cidade)
VALUES ('João', 25, 'São Paulo');
```

Este exemplo insere um novo registro na tabela usuários com os valores fornecidos para nome, idade e cidade.

3. UPDATE: O comando UPDATE é utilizado para modificar registros existentes em uma tabela.

```
UPDATE usuários
SET cidade = 'Rio de Janeiro'
WHERE nome = 'João';
```

Este exemplo atualiza o campo cidade para 'Rio de Janeiro' onde o nome do usuário é 'João'.

4. DELETE: O comando DELETE é utilizado para remover registros de uma tabela.

```
DELETE FROM usuários
WHERE idade < 18;
```

Este exemplo remove todos os registros da tabela usuários onde a idade é menor que 18.

Conclusão

SQL é uma linguagem poderosa e versátil para gerenciamento de bancos de dados relacionais. Com comandos simples como SELECT, INSERT, UPDATE e DELETE, os usuários podem executar uma ampla variedade de operações de dados, desde consultas básicas até manipulações complexas. Compreender esses conceitos básicos de SQL é fundamental para qualquer pessoa que trabalhe com gerenciamento de dados e desenvolvimento de aplicações que dependem de bancos de dados.

Consultas e Manipulação de Dados

A habilidade de consultar e manipular dados de forma eficiente é fundamental para o uso eficaz de SQL em qualquer banco de dados relacional. Três aspectos chave desse processo são a filtragem de dados com a cláusula WHERE, a ordenação de resultados com a cláusula ORDER BY, e o uso de operadores lógicos e aritméticos para refinar as consultas. Vamos explorar cada um desses tópicos em detalhe.

Filtragem de Dados com WHERE

A cláusula WHERE é usada para filtrar registros que atendem a condições específicas. Ela é uma parte essencial das consultas SQL, permitindo que você recupere apenas os dados que são relevantes para sua análise ou operação.

Sintaxe Básica:

```
SELECT coluna1, coluna2, ...
FROM tabela
WHERE condição;
```

Exemplo:

```
SELECT nome, idade, cidade
FROM usuários
WHERE cidade = 'São Paulo';
```

Este exemplo seleciona as colunas nome, idade e cidade da tabela usuários, mas apenas os registros onde a cidade é 'São Paulo' serão retornados. A cláusula WHERE pode usar uma variedade de operadores para definir as condições, incluindo igualdade (=), desigualdade (<> ou !=), maior que (>), menor que (<), maior ou igual (>=), e menor ou igual (<=).

Ordenação de Resultados com ORDER BY

A cláusula ORDER BY é usada para ordenar os resultados de uma consulta com base em uma ou mais colunas. Isso é útil para organizar os dados de maneira que facilite a interpretação e análise.

Sintaxe Básica:

```
SELECT coluna1, coluna2, ...
FROM tabela
ORDER BY coluna1 [ASC|DESC], coluna2 [ASC|DESC], ...;
```

- **ASC:** Ordena os resultados em ordem ascendente (padrão).
- **DESC:** Ordena os resultados em ordem descendente.

Exemplo:

```
SELECT nome, idade, cidade
FROM usuários
ORDER BY idade DESC, nome ASC;
```

Este exemplo ordena os resultados primeiro pela idade em ordem decrescente e, em caso de idades iguais, pelo nome em ordem crescente.

Uso de Operadores Lógicos e Aritméticos

Operadores lógicos e aritméticos são ferramentas poderosas em SQL que permitem a criação de condições complexas na cláusula WHERE e a realização de cálculos diretamente nas consultas.

Operadores Lógicos:

- **AND:** Combina duas ou mais condições, retornando verdadeiro se todas as condições forem verdadeiras.
- **OR:** Combina duas ou mais condições, retornando verdadeiro se pelo menos uma condição for verdadeira.
- **NOT:** Inverte o valor lógico de uma condição.

Exemplo de Uso de Operadores Lógicos:

```
SELECT nome, idade, cidade
FROM usuários
WHERE idade > 18 AND cidade = 'São Paulo';
```

Este exemplo retorna os usuários que têm mais de 18 anos e que moram em 'São Paulo'.

Operadores Aritméticos:

- **+ (Adição):** Soma valores.
- **- (Subtração):** Subtrai valores.
- *** (Multiplicação):** Multiplica valores.
- **/ (Divisão):** Divide valores.

Exemplo de Uso de Operadores Aritméticos:

```
SELECT nome, salario, salario * 1.1 AS salario_com_bonus
FROM funcionarios
WHERE salario < 5000;
```

Este exemplo seleciona o nome e o salário dos funcionários, além de calcular um novo valor de salário com um bônus de 10%, mas apenas para aqueles cujo salário é menor que 5000.

Conclusão

A capacidade de filtrar, ordenar e manipular dados eficientemente é crucial para maximizar o potencial do SQL em qualquer ambiente de banco de dados. A cláusula WHERE permite a filtragem precisa dos registros, a cláusula ORDER BY organiza os resultados de forma útil, e os operadores lógicos e aritméticos possibilitam a criação de condições complexas e a realização de cálculos diretamente nas consultas. Compreender e utilizar esses elementos de forma eficaz pode melhorar significativamente a qualidade e a utilidade das suas análises de dados.

Funções e Agrupamentos

No SQL, as funções agregadas e os agrupamentos são ferramentas essenciais para realizar análises de dados complexas e sumarização. Elas permitem que você calcule estatísticas e resuma informações de maneira eficiente. Neste texto, vamos explorar as funções agregadas mais comuns (SUM, AVG, COUNT, MAX e MIN), o agrupamento de dados com a cláusula GROUP BY, e a filtragem de grupos com a cláusula HAVING.

Funções Agregadas: SUM, AVG, COUNT, MAX, MIN

As funções agregadas são usadas para executar cálculos em um conjunto de valores e retornar um único valor. Elas são frequentemente usadas em consultas SQL para gerar resumos de dados.

1. SUM: A função SUM calcula a soma de um conjunto de valores. É útil para totalizar valores numéricos, como somas de vendas ou orçamentos.

```
SELECT SUM(salario) AS total_salarios
FROM funcionarios;
```

Este exemplo calcula a soma de todos os salários dos funcionários.

2. AVG: A função AVG calcula a média de um conjunto de valores. É usada para determinar o valor médio em um conjunto de dados.

```
SELECT AVG(salario) AS media_salarios
FROM funcionarios;
```

Este exemplo calcula o salário médio dos funcionários.

3. COUNT: A função COUNT conta o número de registros em um conjunto de dados. Pode contar todas as linhas ou apenas aquelas que não são NULL em uma coluna específica.

```
SELECT COUNT(*) AS total_funcionarios
FROM funcionarios;
```

Este exemplo conta o total de funcionários na tabela.

4. MAX: A função MAX retorna o valor máximo de um conjunto de valores. É usada para encontrar o maior valor em um conjunto de dados.

```
SELECT MAX(salario) AS maior_salario
FROM funcionarios;
```

Este exemplo encontra o maior salário entre os funcionários.

5. MIN: A função MIN retorna o valor mínimo de um conjunto de valores. É usada para encontrar o menor valor em um conjunto de dados.

```
SELECT MIN(salario) AS menor_salario
FROM funcionarios;
```

Este exemplo encontra o menor salário entre os funcionários.

Agrupamento de Dados com GROUP BY

A cláusula GROUP BY é usada para agrupar linhas que têm valores iguais em colunas especificadas. Isso é útil para gerar resumos de dados agregados, como somas, contagens e médias, para cada grupo distinto de valores.

Sintaxe Básica:

SELECT coluna1, coluna2, ..., função_agregada(coluna)

FROM tabela

GROUP BY coluna1, coluna2, ...;

Exemplo:

SELECT departamento, AVG(salario) AS media_salario

FROM funcionarios

GROUP BY departamento;

Este exemplo calcula o salário médio para cada departamento, agrupando os funcionários por departamento.

Filtragem de Grupos com HAVING

A cláusula HAVING é usada para filtrar os resultados de grupos criados pela cláusula GROUP BY. Enquanto a cláusula WHERE filtra linhas antes do agrupamento, a cláusula HAVING filtra grupos após o agrupamento.

Sintaxe Básica:

SELECT coluna1, função_agregada(coluna)

FROM tabela

GROUP BY coluna1

HAVING condição;

Exemplo:

```
SELECT departamento, COUNT(*) AS total_funcionarios
FROM funcionarios
GROUP BY departamento
HAVING COUNT(*) > 10;
```

Este exemplo conta o número de funcionários em cada departamento, mas só retorna os departamentos que têm mais de 10 funcionários.

Conclusão

As funções agregadas, o agrupamento de dados com GROUP BY e a filtragem de grupos com HAVING são ferramentas poderosas no SQL para realizar análises de dados e gerar resumos significativos. Com essas funcionalidades, você pode calcular somas, médias, contagens, valores máximos e mínimos, além de agrupar e filtrar dados de maneira eficiente para obter insights valiosos a partir de seus conjuntos de dados. Compreender e utilizar essas técnicas pode melhorar significativamente a capacidade de análise e tomada de decisão baseada em dados.