Introdução ao SQL

O SQL (Structured Query Language) é uma linguagem de programação usada para gerenciar e manipular dados em bancos de dados. Vamos descobrir suas principais características e aprender como usá-lo para extrair informações valiosas.

Características Fundamentais do SQL

- **Sintaxe Clara e Legível:** O SQL possui uma sintaxe fácil de entender, tornando-o acessível até mesmo para iniciantes.
- **Operações CRUD:** Permite realizar quatro operações principais: Create (Inserir), Read (Consultar), Update (Atualizar) e Delete (Excluir) dados em um banco de dados.
- **Consulta de Dados:** Facilita a consulta de informações específicas em grandes conjuntos de dados.
- **Junção de Tabelas:** Permite combinar dados de várias tabelas, tornando possível analisar relações complexas.
- **Funções Agregadas:** Oferece funções como AVG, SUM, MAX e MIN para realizar cálculos em dados.

Exemplos de Código Básico com Múltiplas Tabelas

Selecionando dados

```sql
SELECT alunos.nome, disciplinas.nome
FROM alunos
JOIN matriculas ON alunos.id = matriculas.aluno_id
JOIN disciplinas ON matriculas.disciplina_id = disciplinas.id;
```

Neste exemplo, estamos selecionando nomes de alunos e disciplinas de duas tabelas (alunos e disciplinas) usando a junção (JOIN) da tabela matriculas.

Exemplos de Código Básico com Múltiplas Tabelas

Agregando dados

```sql
SELECT departamentos.nome, COUNT(funcionarios.id) AS total_funcionarios
FROM departamentos
LEFT JOIN funcionarios ON departamentos.id = funcionarios.departamento_id
LEFT JOIN cargos ON funcionarios.cargo_id = cargos.id
GROUP BY departamentos.nome;
```

Aqui, estamos contando o número de funcionários em cada departamento, mesmo que alguns departamentos não tenham funcionários (usando LEFT JOIN).

Exemplos de Código Básico com Múltiplas Tabelas

Filtrando dados

```sql
SELECT clientes.nome, pedidos.numero, produtos.nome
FROM clientes
JOIN pedidos ON clientes.id = pedidos.cliente_id
JOIN itens_pedido ON pedidos.numero = itens_pedido.numero_pedido
JOIN produtos ON itens_pedido.produto_id = produtos.id
WHERE produtos.categoria = 'Eletrônicos';
```

Este código recupera informações de clientes, pedidos e produtos, filtrando apenas produtos da categoria 'Eletrônicos'.

Exemplos de Código Básico com Múltiplas Tabelas

Atualizando registros

```
UPDATE pedidos
JOIN clientes ON pedidos.cliente_id = clientes.id
JOIN produtos ON pedidos.produto_id = produtos.id
SET pedidos.status = 'Enviado'
WHERE clientes.regiao = 'Norte' AND produtos.estoque > 0;
```

Aqui, estamos atualizando o status de pedidos para 'Enviado' apenas para clientes da região Norte cujos produtos têm estoque disponível.

Exemplos de Código Básico com Múltiplas Tabelas

Excluindo com junção de tabelas

```
DELETE FROM alunos
WHERE NOT EXISTS (
    SELECT 1 FROM matriculas
    WHERE matriculas.aluno_id = alunos.id
);
```

Este código exclui registros de alunos que não estão associados a nenhuma matrícula.

Exemplos de Código Avançado com Múltiplas Tabelas

Consulta envolvendo subconsultas

```sql
SELECT funcionarios.nome, departamentos.nome AS departamento, (
    SELECT MAX(salario) FROM funcionarios AS f
    WHERE f.departamento_id = departamentos.id
) AS maior_salario
FROM funcionarios
JOIN departamentos ON funcionarios.departamento_id = departamentos.id;
```

Neste exemplo, calculamos o maior salário em cada departamento usando uma subconsulta.

Exemplos de Código Avançado com Múltiplas Tabelas

Utilizando funções agregadas

```sql
SELECT departamentos.nome, AVG(salario) AS salario_medio, MAX(salario) AS sal
FROM departamentos
JOIN funcionarios ON departamentos.id = funcionarios.departamento_id
GROUP BY departamentos.nome
HAVING AVG(salario) > 50000;
```

Aqui, calculamos a média e o máximo de salários em cada departamento, filtrando apenas os departamentos com uma média salarial superior a $50.000.

Exemplos de Código Avançado com Múltiplas Tabelas

Consulta utilizando tabelas com junção e subconsultas

```sql
SELECT funcionarios.nome, cargos.nome AS cargo, departamentos.nome AS depar
FROM funcionarios
JOIN cargos ON funcionarios.cargo_id = cargos.id
JOIN departamentos ON cargos.departamento_id = departamentos.id
WHERE funcionarios.id NOT IN (
    SELECT funcionario_id FROM avaliacoes WHERE nota < 5
);
```

Neste exemplo, recuperamos funcionários que não têm avaliações com notas abaixo de 5.

Exemplos de Código Avançado com Múltiplas Tabelas

Utilizando consulta recursiva

```
WITH RECURSIVE org_chart AS (
    SELECT id, nome, chefe_id FROM funcionarios WHERE id = 1
    UNION ALL
    SELECT f.id, f.nome, f.chefe_id FROM funcionarios f
    JOIN org_chart o ON f.chefe_id = o.id
)
SELECT * FROM org_chart;
```

Aqui, usamos uma consulta recursiva para construir uma estrutura hierárquica de organograma de funcionários.

Aplicação do SQL no Mundo Real

- **Sistema de Gerenciamento de Vendas**: Rastrear pedidos, clientes, produtos e pagamentos.
- **Sistema de Recursos Humanos**: Gerenciar informações de funcionários, cargos e departamentos.
- **Plataforma de Mídia Social**: Armazenar, recuperar e relacionar dados de usuários, postagens e interações.
- **Análise de Dados de Vendas**: Extrair insights valiosos de grandes conjuntos de dados de vendas para orientar estratégias de negócios.
- **Sistema de Logística**: Rastrear remessas, inventários e entregas.

Conclusão

O SQL é uma ferramenta poderosa para manipular dados em bancos de dados relacionais. Compreender suas características fundamentais e como aplicá-las em consultas envolvendo múltiplas tabelas é essencial para qualquer pessoa que trabalhe com dados.

Continue aprimorando suas habilidades e explore o potencial ilimitado do SQL em seu trabalho e projetos.

Conteúdo gerado por: ChatGPT e revisões humanas

Ilustrações: gerada pela lexica.art