COMANDOS DE DDL E OPERAÇÃO JOIN

MÓDULO I

COMANDOS DE DDL

CREATE TABLE: este comando é usado para criar uma tabela no seu banco de dados. Você especifica o nome da tabela e, em seguida, lista as colunas que deseja incluir, juntamente com o tipo de dados que cada coluna pode armazenar. Por exemplo:

```
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(50),
    idade INT,
    email VARCHAR(100)
);
```

Neste exemplo, estamos criando uma tabela chamada “clientes” com colunas para o ID, nome, idade e e-mail.

ALTER TABLE: se precisar fazer alterações na estrutura de uma tabela existente, você usa o comando ALTER TABLE. Por exemplo, se quiser adicionar uma nova coluna à tabela “clientes”, pode fazer assim:

```
ALTER TABLE clientes
ADD COLUMN telefone VARCHAR(20);
```

Este comando adiciona uma nova coluna chamada “telefone” à tabela “clientes”.

DROP TABLE: se precisar remover uma tabela completamente do seu banco de dados, você pode usar o comando DROP TABLE. Por exemplo:

```
DROP TABLE clientes;
```

Este comando removerá a tabela “clientes” e todos os dados armazenados nela.

Esses são apenas alguns exemplos de como você pode usar a DDL para definir a estrutura do seu banco de dados. Com esses comandos, você pode criar e modificar suas tabelas conforme necessário para atender às necessidades do seu aplicativo ou sistema.

Agora você irá aprender os comandos de DML. Aqui estão eles:

INSERT INTO: este comando é usado para adicionar novos registros a uma tabela existente no seu banco de dados. Você especifica o nome da tabela e, em seguida, lista os valores que deseja adicionar para cada coluna. Por exemplo:

```
INSERT INTO clientes (nome, idade, email)
VALUES ('João', 30, 'joao@email.com');
```

Este comando adiciona um novo registro à tabela "clientes" com o nome "João", idade "30" e e-mail "joao@email.com".

UPDATE: se precisar modificar os dados de um registro existente em uma tabela, você pode usar o comando UPDATE. Por exemplo, se quiser alterar o e-mail de um cliente chamado "João", pode fazer assim:

```
UPDATE clientes
SET email = 'joao.novo@email.com'
WHERE nome = 'João';
```

Este comando atualiza o e-mail do cliente "João" para "joao.novo@email.com".

DELETE FROM: se precisar remover um registro específico de uma tabela, você pode usar o comando DELETE FROM. Por exemplo, se quiser excluir o registro do cliente "João" da tabela "clientes", pode fazer assim:

```
DELETE FROM clientes
WHERE nome = 'João';
Este comando removerá o registro do cliente "João" da tabela "clientes".
```

Com esses comandos DML, você pode manipular os dados dentro das suas tabelas, adicionando novos registros, atualizando informações existentes e excluindo registros conforme necessário para manter seus dados atualizados e precisos.

Agora, o último, mas não menos importante, **DQL**:

SELECT: este comando é usado para recuperar dados de uma ou mais tabelas do banco de dados. Você pode especificar quais colunas deseja recuperar e aplicar filtros para limitar os resultados. Por exemplo:

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

Este comando irá recuperar os nomes e idades de todos os clientes que moram em São Paulo, por exemplo. Além disso, você pode usar cláusulas adicionais para refinar ainda mais sua consulta:

- **WHERE:** permite aplicar condições para filtrar os resultados com base em critérios específicos.
- **ORDER BY:** permite classificar os resultados em ordem ascendente ou descendente com base em uma ou mais colunas.
- **GROUP BY:** agrupa os resultados com base nos valores de uma ou mais colunas.
- **HAVING:** permite aplicar condições de filtro a grupos criados pela cláusula GROUP BY.

Por exemplo, se quisermos agrupar os clientes por idade e contar quantos clientes existem em cada faixa etária, podemos fazer assim:

```
SELECT idade, COUNT(*)
FROM clientes
GROUP BY idade;
```

Este comando retornará o número de clientes em cada faixa etária.

ATENÇÃO

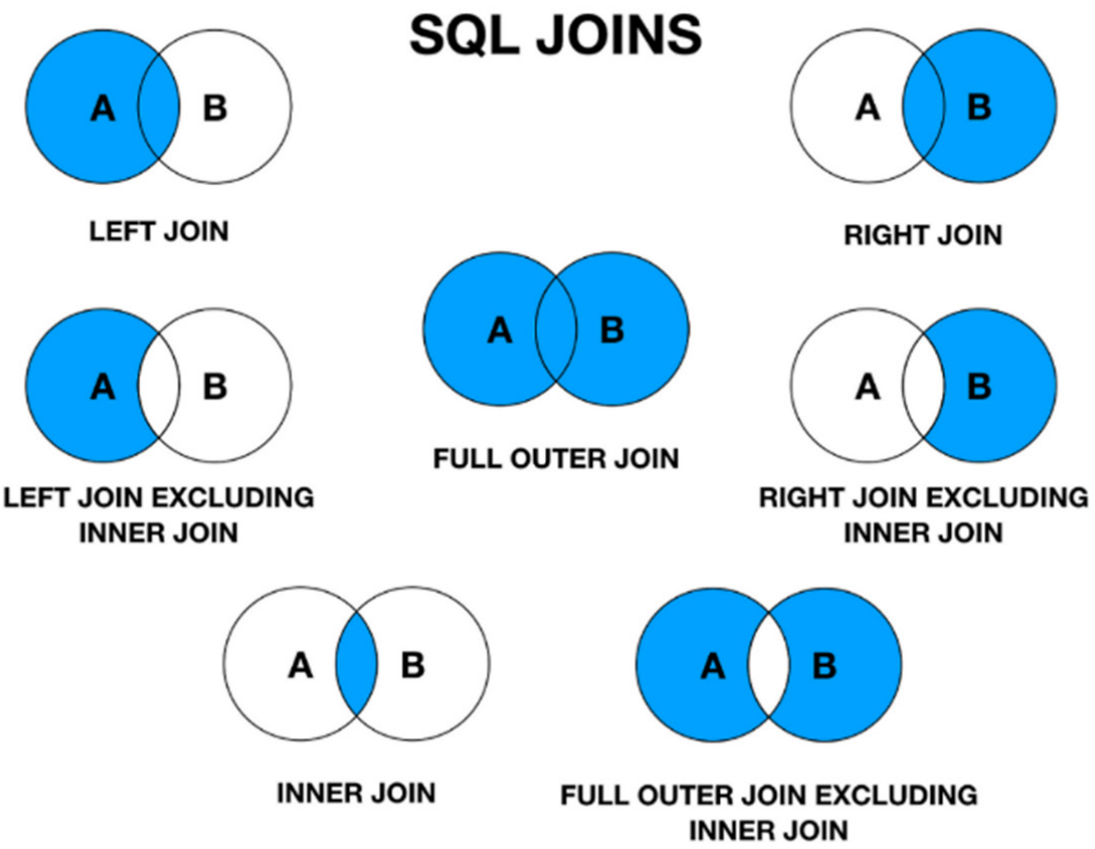
O número de clientes em cada faixa etária pode ser encontrado usando este comando. Você pode recuperar dados específicos do seu banco de dados e manipulá-los de acordo com suas necessidades, ajudando você a encontrar as informações precisas que está procurando usando o comando SELECT e suas cláusulas associadas.

Operação JOIN

Ah, meu caro, vamos falar sobre consultas de dados de múltiplas tabelas, algo bem interessante na linguagem SQL. Sabe quando você precisa juntar informações de várias fontes diferentes, como misturar ingredientes para uma receita especial?

É mais ou menos isso que fazemos com as consultas de múltiplas tabelas. O truque aqui é usar uma operação chamada JOIN, que é como uma cola que une os dados das tabelas. Vou lhe mostrar alguns tipos de JOIN que podemos usar:

- **INNER JOIN:** este é como um encontro perfeito! Ele retorna apenas as linhas que têm correspondências nas duas tabelas, ou seja, apenas os dados que têm algo em comum.
- **LEFT JOIN:** aqui é como abrir uma porta. Retorna todas as linhas da tabela à esquerda e as correspondentes da tabela à direita, ou NULL se não houver correspondência. É como convidar todos para a festa, mesmo que alguns convidados não apareçam.
- **RIGHT JOIN:** é parecido com o LEFT JOIN, mas inverte os papéis. Retorna todas as linhas da tabela à direita e as correspondentes da tabela à esquerda, ou NULL se não houver correspondência.
- **FULL JOIN:** este é como ter todos os ingredientes na receita, não importa de onde eles venham. Retorna todas as linhas quando há uma correspondência em uma das tabelas.



A imagem é a representação ilustrativa dos conceitos apresentados acima.

ATENÇÃO

Considere a situação em que você tem duas tabelas: “clientes” e “pedidos”. Todos os pedidos estão relacionados a um cliente, não é? Você pode usar um INNER JOIN para ver os detalhes do pedido, incluindo o nome do cliente. Juntar peças de quebra-cabeça para ver a imagem completa é semelhante a isso. Imagine que temos informações de clientes como nome, endereço e telefone em uma tabela. Além disso, a tabela “pedidos” contém informações como número, data e valor total, além de uma coluna “id_cliente” que representa o cliente. Você pode usar um INNER JOIN para unir essas duas tabelas se quiser ver o nome do cliente e os detalhes do pedido. É como ligar os clientes aos pedidos.

Dê uma olhada em como seria uma consulta assim:

```
SELECT clientes.nome, pedidos.numero_pedido, pedidos.data, pedidos.valor_total
FROM clientes
INNER JOIN pedidos ON clientes.id = pedidos.id_cliente;
```

Aqui, você está selecionando o nome do cliente da tabela “clientes” e o número do pedido, a data e o valor total da tabela “pedidos”. Depois, você usa o INNER JOIN para unir as duas tabelas com base na condição de que o “id” do cliente na tabela “clientes” seja igual ao “id_cliente” na tabela “pedidos”.

Assim, você obtém uma lista que mostra o nome de cada cliente junto com os detalhes de todos os pedidos que ele fez. É como se você estivesse montando um quebra-cabeça para ver toda a imagem!