Modelo relacional e introdução aos Joins

<LAB365>

SENAI

OBJETIVOS

- Fornecer uma visão geral inicial sobre modelo relacional
- Entender o uso do Join

INTRODUÇÃO

- O modelo relacional organiza dados em tabelas (ou relações) com linhas (tuplas) e colunas (atributos). A modelagem relacional define como as tabelas estão relacionadas umas com as outras. Existem três tipos principais de relacionamentos:
  - Um-para-Muitos (1)
  - Um-para-Um (1:1)
  - Muitos-para-Muitos (M)

RELACIONAMENTO UM PARA MUITOS (1)

- Um relacionamento um-para-muitos é quando uma linha de uma tabela está associada a várias linhas de outra tabela.

```
CREATE TABLE clientes (
    cliente_id SERIAL PRIMARY KEY,
    nome VARCHAR(50),
    cidade VARCHAR(50),
    idade INT
);
```

```
CREATE TABLE pedidos (
pedido_id SERIAL PRIMARY KEY,
cliente_id INT,
data_pedido DATE,
valor DECIMAL(10, 2),
FOREIGN KEY (cliente_id) REFERENCES
clientes(cliente_id)
);
```

RELACIONAMENTO UM-PARA-UM

- Um relacionamento um-para-um é quando uma linha de uma tabela está associada a, no máximo, uma linha de outra tabela.

```
CREATE TABLE perfis_clientes (
    perfil_id SERIAL PRIMARY KEY,
    cliente_id INT UNIQUE,
    biografia TEXT,
    FOREIGN KEY (cliente_id) REFERENCES clientes(cliente_id)
);
```

RELACIONAMENTO MUITOS-PARA-MUITOS (M)

- Um relacionamento muitos-para-muitos é quando várias linhas de uma tabela estão associadas a várias linhas de outra tabela. Para implementar isso, usamos uma tabela intermediária (tabela de junção/satélite).

```
CREATE TABLE itens_pedidos (
    item_id SERIAL PRIMARY KEY,
    pedido_id INT,
    produto_id INT,
    quantidade INT,
    FOREIGN KEY (pedido_id) REFERENCES pedidos(pedido_id),
    FOREIGN KEY (produto_id) REFERENCES produtos(produto_id)
);
```

```
CREATE TABLE produtos (
    produto_id SERIAL PRIMARY KEY,
    nome_produto VARCHAR(100),
    preco DECIMAL(10, 2)
);
```

INSERÇÃO DE DADOS

```
INSERT INTO clientes (nome, cidade, idade) VALUES
('Alice', 'São Paulo', 30),
('Bob', 'Rio de Janeiro', 25),
('Carlos', 'Belo Horizonte', 35),
('Diana', 'Curitiba', 28);

INSERT INTO perfis_clientes (cliente_id, biografia) VALUES
(1, 'Alice é uma entusiasta de tecnologia.'),
(2, 'Bob gosta de viajar e conhecer novas culturas.'),
(3, 'Carlos é um amante de esportes e fitness.');
INSERT INTO pedidos (cliente_id, data_pedido, valor) VALUES
(1, '2023-06-01', 150.50),
(2, '2023-06-02', 200.00),
(NULL, '2023-06-04', 50.00);
```

```
INSERT INTO produtos
(nome_produto, preco) VALUES
('Notebook', 1500.00),
('Smartphone', 800.00),
('Tablet', 600.00);


INSERT INTO itens_pedidos
(pedido_id, produto_id,
quantidade) VALUES
(1, 1, 1),
(1, 2, 2),
(2, 2, 1),
(3, 3, 3);
```

OPERADOR JOIN

- Resumo dos Joins
  - INNER JOIN: Retorna linhas quando há uma correspondência em ambas as tabelas.
  - LEFT JOIN: Retorna todas as linhas da tabela à esquerda, e as linhas correspondentes da tabela à direita.
  - RIGHT JOIN: Retorna todas as linhas da tabela à direita, e as linhas correspondentes da tabela à esquerda.
  - FULL OUTER JOIN: Retorna todas as linhas quando há uma correspondência em uma das tabelas.

JOIN/INNER JOIN

```
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor
FROM clientes c
INNER JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

- **SELECT c.nome, p.pedido_id, p.data_pedido, p.valor**: Seleciona as colunas nome da tabela clientes, pedido_id, data_pedido, e valor da tabela pedidos.

- **FROM clientes c**: Define a tabela clientes com o alias c.

- **INNER JOIN pedidos p ON c.cliente_id = p.cliente_id**: Junta a tabela clientes (c) com a tabela pedidos (p) onde cliente_id é igual em ambas as tabelas. Somente retorna as linhas onde há correspondência em ambas as tabelas.

LEFT JOIN

```
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor
FROM clientes c
LEFT JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

- **SELECT c.nome, p.pedido_id, p.data_pedido, p.valor**: Seleciona as colunas nome da tabela clientes, pedido_id, data_pedido, e valor da tabela pedidos.

- **FROM clientes c**: Define a tabela clientes com o alias c.

- **LEFT JOIN pedidos p ON c.cliente_id = p.cliente_id**: Junta a tabela clientes (c) com a tabela pedidos (p) onde cliente_id é igual em ambas as tabelas. Retorna todas as linhas da tabela clientes e as correspondentes da tabela pedidos. Para clientes sem pedidos, os valores das colunas de pedidos serão NULL.

RIGHT JOIN

```
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor
FROM clientes c
RIGHT JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

- **SELECT c.nome, p.pedido_id, p.data_pedido, p.valor**: Seleciona as colunas nome da tabela clientes, pedido_id, data_pedido, e valor da tabela pedidos.

- **FROM clientes c**: Define a tabela clientes com o alias c.

- **RIGHT JOIN pedidos p ON c.cliente_id = p.cliente_id**: Junta a tabela clientes (c) com a tabela pedidos (p) onde cliente_id é igual em ambas as tabelas. Retorna todas as linhas da tabela pedidos e as correspondentes da tabela clientes. Para pedidos sem clientes (caso existam), os valores das colunas de clientes serão NULL.

FULL OUTER JOIN

```
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor
FROM clientes c
FULL OUTER JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

- **SELECT c.nome, p.pedido_id, p.data_pedido, p.valor**: Seleciona as colunas nome da tabela clientes, pedido_id, data_pedido, e valor da tabela pedidos.

- **FROM clientes c**: Define a tabela clientes com o alias c.

- **FULL OUTER JOIN pedidos p ON c.cliente_id = p.cliente_id**: Junta a tabela clientes (c) com a tabela pedidos (p) onde cliente_id é igual em ambas as tabelas. Retorna todas as linhas de ambas as tabelas, correspondendo ou não. Para clientes sem pedidos, os valores das colunas de pedidos serão NULL. Para pedidos sem clientes, os valores das colunas de clientes serão NULL.

UTILIZANDO O RELACIONAMENTO UM-PARA-UM

```
SELECT c.nome, pc.biografia
FROM clientes c
INNER JOIN perfis_clientes pc ON c.cliente_id = pc.cliente_id;
```

UTILIZANDO O RELACIONAMENTO UM PARA MUITOS (1)

```sql
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor
FROM clientes c
INNER JOIN pedidos p ON c.cliente_id = p.cliente_id;
```

UTILIZANDO O RELACIONAMENTO MUITOS-PARA-MUITOS (M)

```sql
SELECT c.nome, pr.nome_produto, ip.quantidade
FROM clientes c
INNER JOIN pedidos p ON c.cliente_id = p.cliente_id
INNER JOIN itens_pedidos ip ON p.pedido_id = ip.pedido_id
INNER JOIN produtos pr ON ip.produto_id = pr.produto_id;
```

INTERVALO

DEV!

Finalizamos o nosso primeiro período de hoje.
Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

MATERIAL COMPLEMENTAR

https://medium.com/@aneuk3/sql-joins-defcf817e8cf

https://www.w3schools.com/postgresql/postgresql_intro.php

CONTATO

- Discord: Pedro Henrique - phbs#2006
- Email: pedro.barroso@edu.sc.senai.br
- Linkedin: https://www.linkedin.com/in/pedro-h-b-da-silva/
- Github: https://github.com/pedrohbsilva/

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

OBRIGADO!

<LAB365>