UNIVERSIDADE FEDERAL DO MARANHÃO

BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

EECP0004 - BANCO DE DADOS

JOAO VICTOR OLIVEIRA SANTOS

P3 BANCO DE DADOS - MODELAGEM DE SISTEMA PARA SITE DE ATELIER

SÃO LUÍS - MA

JULHO/2025

JOÃO VICTOR OLIVEIRA SANTOS (2021024581)

P3 BANCO DE DADOS - MODELAGEM DE SISTEMA PARA SITE DE ATELIER

Documento apresentado como requisito parcial de avaliação da disciplina Banco de Dados, no curso Bacharelado Interdisciplinar em Ciência e Tecnologia da Universidade Federal do Maranhão.

Orientador: Profa. Dra. Vandecia Rejane Monteiro Fernandes.

SÃO LUÍS - MA

JULHO/2025

SUMÁRIO

1 PROBLEMÁTICA

O Atelier Goreth's é um negócio especializado em costura personalizada, consertos e confecção sob medida. Localizado em uma área de grande demanda e com uma clientela crescente, a gestão manual de pedidos, anotações de pagamento e cadastro de clientes tornou-se ineficiente e sujeita a falhas. O controle dos serviços, como o acompanhamento de pedidos e a organização dos pagamentos, está cada vez mais complexo e impacta negativamente a agilidade e a precisão no atendimento aos clientes.

Para atender a essa demanda crescente e melhorar a gestão de seus processos, o Atelier Goreth's decidiu investir na modernização de sua gestão com um sistema de banco de dados robusto e eficiente. A implementação de um sistema informatizado permitirá otimizar o controle de serviços, garantir a segurança das informações e melhorar a experiência do cliente.

Objetivo:

A missão é criar um sistema de banco de dados completo para o Atelier Goreth's, que abrange os seguintes requisitos:

- Cadastrar clientes (físicos e jurídicos), com informações detalhadas como nome, CPF/CNPJ, endereço e telefone;
- Registrar e associar pedidos de costura a clientes, com informações sobre tipo de serviço, data de entrega e status do pedido;
- Controlar os valores e status de pagamento de cada pedido, garantindo rastreabilidade e transparência;
- Vincular pedidos a consultores ou costureiras específicas, garantindo a organização do trabalho;
- Gerar relatórios de desempenho do negócio, incluindo o histórico de pedidos por cliente, tempo de execução e taxa de satisfação.

A criação desse sistema envolverá a modelagem de banco de dados, incluindo o modelo conceitual, lógico e físico, considerando a necessidade de garantir um controle ágil e eficiente, com segurança das informações e fácil acesso aos dados pelos gestores.

2 MODELO CONCEITUAL (DER)

A figura abaixo mostra o modelo conceitual do sistema de gestão do Atelier Goreth's, modelado a partir da problemática apresentada. O diagrama apresenta 5 entidades, sendo uma delas uma especialização (Cliente, que pode ser Pessoa Física ou Jurídica). Ele também apresenta diversos relacionamentos 1:N, como entre Cliente e Pedido (um cliente pode realizar vários pedidos), e entre Pedido e Pagamento (um pedido pode ter vários pagamentos).

Além disso, o diagrama inclui um relacionamento N:N entre Pedido e Consultor (um pedido pode ter múltiplos consultores e um consultor pode trabalhar em vários pedidos), implementado por meio de uma tabela associativa entre as duas entidades.

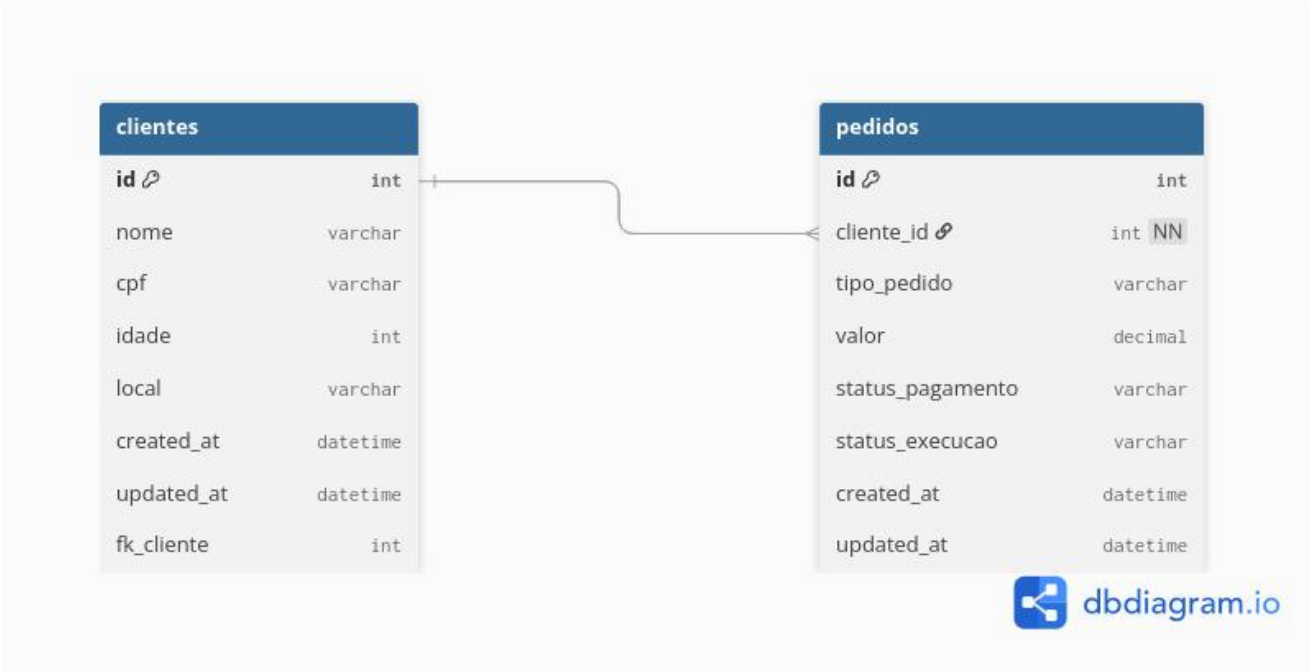

Figura 1 - Modelo conceitual DER.

3 MODELO LÓGICO RELACIONAL

A partir do modelo conceitual apresentado anteriormente, foi modelado o Modelo Lógico Relacional para o banco de dados:

CLIENTE

CLIENTE(id_cliente, nome, tipo_cliente, cpf_cnpj, endereco, telefone)

PEDIDO

PEDIDO(id_pedido, descricao, data_pedido, status_pedido, status_pagamento, valor_total, id_cliente [FK])

PAGAMENTO

PAGAMENTO(id_pagamento, data_pagamento, valor_pago, metodo, id_pedido [FK])

CONSULTOR

CONSULTOR(id_consultor, nome, especialidade)

PEDIDO_CONSULTOR

PEDIDO_CONSULTOR(id_pedido [FK], id_consultor [FK]) — tabela associativa

3 MODELO FÍSICO

Com os modelos relacional e lógico devidamente apresentados, é hora de transformar essas informações em código SQL. Abaixo, encontra-se o código necessário para a implementação do Banco de Dados do Atelier Goreth's, abrangendo desde a criação da database até a definição de todas as tabelas essenciais para o gerenciamento dos dados relacionados aos clientes, pedidos, pagamentos e consultores.

• Uma função:
```sql
CREATE FUNCTION get_nome_cliente(id_cliente integer) RETURNS text AS $$
DECLARE nome_cliente TEXT;
BEGIN
  SELECT nome INTO nome_cliente FROM clientes WHERE id = id_cliente;
  RETURN nome_cliente;
END;
$$ LANGUAGE plpgsql;
```

• Uma trigger:
```sql
CREATE FUNCTION on_update_current_timestamp_clientes() RETURNS trigger AS $$
BEGIN
  NEW.updated_at = now();
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

• Função agregada:
```sql
SELECT id_cliente, SUM(valor_total) AS total_gasto
FROM pedidos
GROUP BY id_cliente;
```

• Consulta com HAVING:
```sql
SELECT id_cliente, SUM(valor_total) AS total_gasto
FROM pedidos
GROUP BY id_cliente
HAVING SUM(valor_total) > 100;
```

3.1 3.1 EXEMPLOS DE CONSULTA AO BANCO DE DADOS

Com o modelo físico implementado e o banco de dados configurado, passaremos para as consultas que validarão o funcionamento do sistema. A seguir, serão realizadas algumas consultas utilizando funções agregadas, cláusulas HAVING, implementação de funções e triggers.

3.1.1 Adição de Dados

Para realizar as consultas, primeiro precisamos adicionar dados ao banco de dados. Abaixo estão as instruções SQL para povoar as tabelas com informações e permitir a execução das consultas.

```
-- Adicionar dados no Banco de Dados para testar consultas

-- Clientes
INSERT INTO Cliente (id_cliente, nome, cpf, endereco, telefone) VALUES
(1, 'Maria Silva', '123.456.789-00', 'Rua A, 1', '9999-9999'),
(2, 'José Souza', '987.654.321-00', 'Rua B, 2', '8888-8888'),
(3, 'Ana Paula', '555.555.555-55', 'Rua C, 3', '7777-7777');

-- Consultores
INSERT INTO Consultor (id_consultor, nome, especialidade) VALUES
(1, 'João Pereira', 'Costura Personalizada'),
(2, 'Ana Costa', 'Conserto de Roupas');

-- Pedidos
INSERT INTO Pedido (id_pedido, descricao, data_pedido, status_pagamento, valor_total, id_cliente)
    VALUES
(1, 'Conserto de Blusa', '2024-07-20', 'Pendente', 50.00, 1),
(2, 'Vestido Sob Medida', '2024-07-21', 'Pago', 150.00, 2),
(3, 'Ajuste de Calça', '2024-07-22', 'Pendente', 80.00, 3);

-- Pagamentos
INSERT INTO Pagamento (id_pagamento, data_pagamento, valor_pago, metodo, id_pedido) VALUES
(1, '2024-07-21', 150.00, 'Cartão', 2);

-- Pedido-Consultor (Tabela Associativa)
INSERT INTO Pedido_Consultor (id_pedido, id_consultor) VALUES
(1, 1),
(2, 2),
    (3, 1);
```

3.1.2 Consulta utilizando alguma Função Agregada

- a) Calcular o valor total de pedidos por cliente:

```
SELECT c.nome AS cliente, SUM(p.valor_total) AS total_gasto
FROM Pedido p
JOIN Cliente c ON p.id_cliente = c.id_cliente
    GROUP BY c.nome;
```

- b) Calcular a média de valores dos pedidos:

```
SELECT AVG(valor_total) AS media_valor_pedido
FROM Pedido;
```

3.1.3 Consulta utilizando HAVING

Aqui estão dois exemplos de consultas utilizando a cláusula HAVING.

- a) Listar clientes que gastaram mais de R$ 100,00:

```
SELECT c.nome AS cliente, SUM(p.valor_total) AS total_gasto
FROM Pedido p
JOIN Cliente c ON p.id_cliente = c.id_cliente
GROUP BY c.nome
HAVING SUM(p.valor_total) > 100;
```

- b) Listar consultores que têm mais de 2 pedidos atribuídos:

```
SELECT cons.nome, COUNT(p.id_pedido) AS total_pedidos
FROM Consultor cons
JOIN Pedido_Consultor pc ON cons.id_consultor = pc.id_consultor
JOIN Pedido p ON pc.id_pedido = p.id_pedido
GROUP BY cons.nome
HAVING COUNT(p.id_pedido) > 2;
```

3.1.4 Função

Aqui está a criação de uma função para calcular o valor total de um pedido, somando os valores de cada item associado ao pedido.

- Criar função para calcular o total de um pedido:
-

```
CREATE OR REPLACE FUNCTION calcular_total_pedido(p_id_pedido INTEGER)
RETURNS DECIMAL AS $$
DECLARE
  total DECIMAL;
BEGIN
  SELECT SUM(valor_total) INTO total
  FROM Pedido
  WHERE id_pedido = p_id_pedido;

  RETURN total;
END;
$$ LANGUAGE plpgsql;
```

- Rodar a função para calcular o total de um pedido específico:

SELECT calcular_total_pedido(1); -- Substitua 1 pelo ID do pedido desejado

3.1.5 Trigger

Criamos uma trigger que será executada após a inserção de um novo item em **Pedido_Consultor**, garantindo que o valor total do pedido seja atualizado automaticamente.

- Criar trigger para atualizar o valor total do pedido:

```
CREATE OR REPLACE FUNCTION atualizar_total_pedido() RETURNS TRIGGER AS $$
BEGIN
  UPDATE Pedido
  SET valor_total = calcular_total_pedido(NEW.id_pedido)
  WHERE id_pedido = NEW.id_pedido;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER gatilho_atualizar_total_pedido
AFTER INSERT ON Pedido_Consultor
FOR EACH ROW
EXECUTE PROCEDURE atualizar_total_pedido();
```

- Verificar a trigger após inserção de um novo item de pedido:

```
INSERT INTO Pedido_Consultor (id_pedido, id_consultor)
VALUES (1, 2);  -- Exemplo de atualização de pedido

SELECT * FROM Pedido WHERE id_pedido = 1;
```