CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Manoel Jorge de Lima Mieiro

&

Philippe Henry Marinho de Lagaye De Lanteuil

DOCUMENTO DE BANCO DE DADOS

módulo de vendas da DF Paper Company

Rio de Janeiro

2024

Sumário

1. Introdução

Este documento especifica detalhes arquiteturais, e de composição do banco de dados do time de vendas da empresa DF paper company; empresa fictícia objeto de estudo para a segunda avaliação da disciplina de Administração de Banco de Dados, ministrada pelo docente Diego Cardoso Borda Castro no Centro Federal de Educação Tecnológica da Celso Suckow da Fonseca durante o primeiro semestre de 2024.

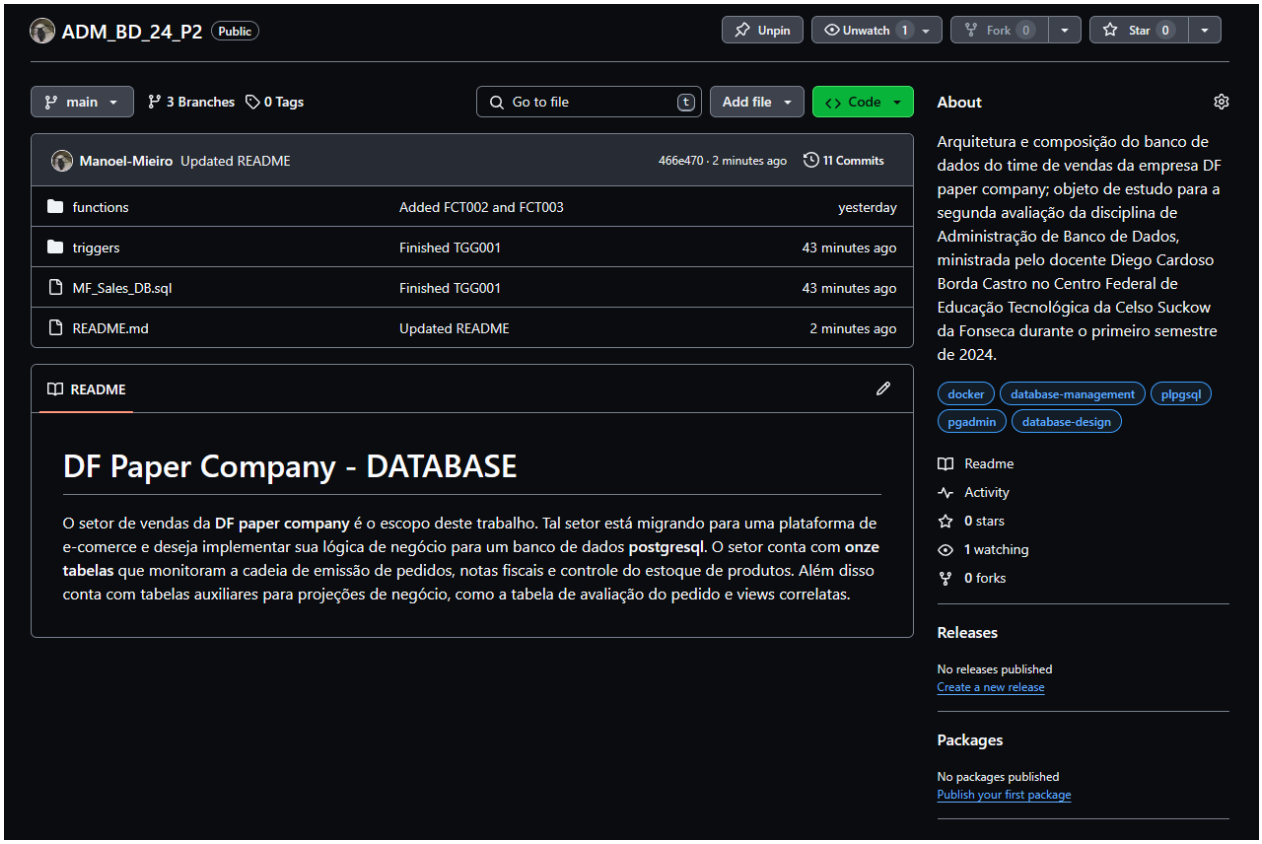
2. Escopo

O setor de vendas da DF paper company é o escopo deste trabalho. Tal setor está migrando para uma plataforma de e-comerce e deseja implementar sua lógica de negócio para um banco de dados postgresql. O setor conta com onze tabelas que monitoram a cadeia de emissão de pedidos, notas fiscais e controle do estoque de produtos. Além disso conta com tabelas auxiliares para projeções de negócio, como a tabela de avaliação do pedido e views correlatas.

3. Código Fonte

Os scripts, bem como infraestrutura do código estão disponíveis no GitHub, conforme imagem abaixo:

Figura 1 - Repositório do Projeto



4. Tabelas e Dicionário de Dados

4.1. [TB001] Produtos

Tabela	Produtos			
Descrição	Armazenará o catálogo de produtos da empesa			
Observações	Possui uma chave estrangeira na tabela Armazem e Fornecedor			
	As colunas cor, marca e tipo são ENUMs com valores pré-definidos			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
local	Id do armazém correspondente na tabela armazem	integer		FK
fornecedor	Id do fornecedor correspondente na tabela fornecedores	integer		FK
quantidade_estoque	Quantidade de itens no estoque	smallint		NOT NULL
codigo_barras	Código de barras do produto	varchar	13	NOT NULL
marca	Marca do produto	brands		NOT NULL
tipo	Tipo do produto	types		NOT NULL
descricao	Descrição do produto	varchar	100	NOT NULL
altura_embalagem	Altura da embalagem do produto	real		NOT NULL
largura_embalagem	Largura da embalagem do produto	real		NOT NULL
comprimento_embalagem	Comprimento da embalagem do produto	real		NOT NULL
peso	Peso do produto	real		NOT NULL
qtd_folhas	Quantidade de folhas de uma embalagem	smallint		NOT NULL
cor	Cor do papel	colors		NOT NULL
gramatura	Gramatura do papel	real		NOT NULL
altura_folha	Altura do papel	real		NOT NULL
comprimento_folha	Comprimento do papel	real		NOT NULL
valor_compra	Valor da compra com o fornecedor	real		NOT NULL
valor_venda	Valor de venda do produto	real		NOT NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.2. [TB002] Clientes

Tabela	Clientes			
Descrição	Armazenará cada cliente			
Observações	A coluna cartao é um ENUM com valores crédito e débito			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
nome	Nome completo	varchar	80	NOT NULL
cpf	Documento de identificação	varchar	11	NOT NULL
email	Email do cliente	varchar	80	NOT NULL
telefone	Telefone do cliente	varchar	11	NOT NULL
cartao	Cartao do cliente	varchar	30	
senha	Senha do cliente	varchar	30	NOT NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.3. [TB003] Pedidos

Tabela	Pedidos			
Descrição	Armazenará cada pedido feito por um cliente			
Observações	Possui uma chave estrangeira na tabela Clientes, Entregas e Carrinho			
	A coluna forma_pagamento é um ENUM com valores de pagamento pré-definidos			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id_pedido	Código de identificação da tabela	serial		PK/Identity
destino	Id da entrega correspondente na tabela entregas	integer		FK
cliente	Id do usuário correspondente na tabela clientes	integer		FK
carrinho	Id do carrinho correspondente na tabela carrinho	integer		FK
frete	Calculado baseado no tipo de entrega indicado da tabela entregas	real		NOT NULL
total_carrinho	Total a ser pago pelo carrinho	real		NOT NULL
forma_pagamento	Tipo de pagamento do pedido	payment		NOT NULL
cupom	Cupom de desconto aplicado ao pedido	varchar	12	

valor_desconto	Valor com base no cupom	real		NOT NULL
total_a_pagar	Valor total do pedido	real		NOT NULL
confirmação_pagamento	Flag se o pedido foi pago ou não	boolean		NOT NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.4. [TB004] Entrega

Tabela	Entrega			
Descrição	Armazenará as entregas feitas para cada pedido			
Observações	As colunas tipo_entrega, estado e uf são ENUMs com valores pré definidos			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
tipo_entrega	Tipo de entrega do pedido	shipping		NOT NULL
cep	Cep de destino	varchar	8	NOT NULL
adress	Endereço de destino	varchar	150	NOT NULL
complemento	Complemento do endereço do desino	varchar	50	NOT NULL
numero	Número do endereço do destino	smallint		NOT NULL
estado	Estado do endereço do destino	states		NOT NULL
uf	Unidade Federativa do endereço	UF		NOT NULL
bairro	Bairro do endeeço de destino	varchar	50	NOT NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.5. [TB005]Cartões

Tabela	Cartoes			
Descrição	Armazenará informações sobre os eventuais cartões do cliente			
Observações	Possui uma chave estrangeira na tabela Cliente			
	As coluna banco e tipo são ENUMs com valores pré-definidos			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
cliente	Id do cliente correspondente na tabela clientes	integer		FK
banco	Nome da instituição financeira	banks		NOT NULL
numero	Número do cartão	varchar	16	NOT NULL
tipo	Tipo do cartão	cards		NOT NULL
mes_validade	Mês de validade do cartão	smallint		NOT NULL
ano_validade	Ano de validade do cartão	smallint		NOT NULL
nome_titular	Nome do titular do cartão	varchar	40	NOT NULL
cod_verificacao	CVV do cartão	smallint		NOT NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.6. [TB006] Avaliação de Pedido

Tabela	Avaliação de Pedido			
Descrição	Armazenará informações de avaliação de um produto feita por um cliente			
Observações	Possui uma chave estrangeira na tabela Clientes e outra em Produtos			
	A coluna qtd_estrelas é um ENUM com valores 1, 2, 3, 4 e 5			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
usuario	Id do usuário correspondente na tabela clientes	integer		FK
produto	Id do produto na tabela produtos	integer		FK
qtd_estrelas	Quantidade de estrelas dada como avaliação ao pedido podendo ir de 1 a 5	qtd_estrelas		NOT NULL
descricao	Descrição verbal opcional dada ao produto	varchar	100	
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.7. [TB007] Fornecedores

Tabela	Fornecedores			
Descrição	Armazenará informações de uma entidade fornecedora de produtos para revenda por nossa empresa			
Observações	A coluna estado conta com um ENUM mapeando todos os estados brasileiros			
	A coluna uf é um ENUM mapeando todas as UFs brasileiras			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial	serial	PK/Identity
nome	Nome da entidade fornecedora	varchar	100	Not NULL
cnpj	CNPJ da entidade fornecedora, sem pontos	varchar	18	Not NULL
address	Endereço da entidade fornecedora	varchar	80	Not NULL
cep	CEP da entidade fornecedora, sem pontos	varchar	8	Not NULL
estado	Estado em que a entidade fornecedora se encontra, limitado ao Brasil	states		Not NULL
uf	Unidade Federativa em que a entidade fornecedora se encontra	UF		Not NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.8. [TB008] Carrinho

Tabela	Carrinho
Descrição	Armazenará o agrupamento de cada produto único de forma a alimentar posteriormente a tabela de pedido
Observações	Possui uma chave estrangeira na tabela Pedidos e outra em Produtos
	Adotou-se o tipo samllint para quantidade pelo valor máximo desse tipo de dado estar dentro da realidade do que podemos oferecer em nosso estoque e por ser um tipo de dados mais leve, com apenas 2 bytes de tamanho

Campos

Nome	Descrição	Tipo de Dado	Tamanho	Restrições
id	Código de identificação da tabela	serial		PK/Identity
pedido	Id do pedido na tabela pedidos	integer		FK
produto	Id do produto na tabela produtos	integer		FK
quantidade	Quantidade adicionada de cada produto ao carrinho	smallint		Not NULL
preco_total_produto	Cálculo do valor total a ser pago por cada conjunto de produto único, sendo o produto da quantidade pelo valor de venda obtido na tabela produto	real		Not NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.9. [TB009] Pedidos Arquivados

Tabela	Pedidos Arquivados			
Descrição	Armazenará pedidos cujo pagamento foi efetuado			
Observações	Possui uma chave estrangeira na tabela Produtos			
	Possui um ENUM na coluna forma_pagamento com 4 formas predefinidas			
Campos				
Nome	**Descrição**	**Tipo de Dado**	**Tamanho**	**Restrições**
id	Código de identificação da tabela	serial		PK/Identity
pedido	Id do pedido na tabela pedidos	integer		FK
nf	Id da nota fiscal gerada após pagamento do pedido na tabela nota_fiscal	integer		Not NULL
data_pedido	Data em que o pedido foi concluído, isto é, pago	datetime		Not NULL
cliente	Id do cliente na tabela pedidos	integer		Not NULL
carrinho	Id do carrinho na tabela pedidos	integer		Not NULL
forma_pagamento	Forma de pagamento usada na tabela pedido	payment		Not NULL
total_a_pagar	Valor a ser pago usado na tabela pedidos	real		Not NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.10. [TB010] Nota Fiscal

Tabela	Nota Fiscal
Descrição	Armazena informações pertinentes à Nota Fiscal de um dado pedido
Observações	Adotou-se o tipo samllint para a coluna numero pelo valor máximo desse tipo de dado estar dentro da realidade de números de logradouro e por ser um tipo de dados mais leve, com apenas 2 bytes de tamanho
	A coluna estado conta com um ENUM mapeando todos os estados brasileiros
	A coluna uf é um ENUM mapeando todas as UFs brasileiras
	A coluna natureza_operacao é um ENUM com valores predefinidos

Campos

Nome	Descrição	Tipo de Dado	Tamanho	Restrições
id	Código de identificação da tabela	serial		PK/Identity
nome_empresa	Nome da nossa empresa	varchar	70	Not NULL
entrada_saida	Indica a natureza da nota fiscal, sendo compra ou venda	boolean		Not NULL
numero_nf	Número da nota fiscal	varchar	43	Not NULL
chave_acesso	Chave de acesso à nota fiscal	varchar	40	Not NULL
natureza_operacao	Natureza da operação	op		Not NULL
protocolo	Protocolo da nota fiscal	varchar	40	Not NULL
cnpj	CNPJ da empresa emissora da nota fiscal	varchar	18	Not NULL
data_emissao	Data de emissão da nota	datetime		Not NULL
address	Endereço da empresa que emitiu	varchar	150	Not NULL
numero	Número do logradouro da empresa emissora	smallint		Not NULL
uf	UF da empresa emissora	UF		Not NULL
estado	Estado em que se encontra a empresa emissora	states		Not NULL
bairro	Bairro em que se encontra a empresa emissora	varchar	50	Not NULL
cep	CEP da empresa emissora	varchar	8	Not NULL
telefone	Telefone da empresa emissora	integer		Not NULL
base_icms	Cálculo base do ICMS a ser aplicado à nota fiscal	real		Not NULL
valor_icms	Valor do ICMS atribuído ao produto	real		Not NULL
valor frete	Valor do frete obtido na tabela pedidos	real		Not NULL
desconto	Valor do desconto aplicado ao pedido	real		Not NULL
total_tributos	Valor total de tributos a serem agregados à nota fiscal	real		Not NULL

total_produtos	Valor total dos produtos presentes na nota	real		Not NULL
total_nf	Valor total da NF, combinando total_produtos, total_tributos e desconto	real	100	Not NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

4.11. [TB011] Armazém

Tabela	Armazém
Descrição	Armazenará informações referentes a localização de armazenamento dos produtos para que a tabela Produtos faça o consumo delas
Observações	Usa um ENUM para a coluna uf com todas as UFs brasileiras predefinidas
	Usa um ENUM para a coluna estado com todos os estados brasileiros predefinidos
	Adotou-se o tipo samllint para a coluna numero pelo valor máximo desse tipo de dado estar dentro da realidade de números de logradouro e por ser um tipo de dados mais leve, com apenas 2 bytes de tamanho

Campos

Nome	Descrição	Tipo de Dado	Tamanho	Restrições
id	Código de identificação da tabela	serial		PK/Identity
cep	CEP em que se encontra o armazém, sem pontos	varchar	8	Not NULL
address	Endereço completo do armazém	address	150	Not NULL
numero	Número da casa/galpão onde se encontra o armazém	smallint		Not NULL
uf	Unidade federativa onde se encontra o armazém	UF		Not NULL
estado	Estado em que se encontra o armazém	states		Not NULL
bairro	Bairro em que se encontra o armazém	varchar	50	Not NULL
added_at	Coluna suporte para operações soft	datetime		
updated_at	Coluna suporte para operações soft	datetime		
deleted_at	Coluna suporte para operações soft	datetime		

5. Índices

5.1. [IDX001] Tabela Pedidos

Esse índice foi criado com o intuito de agilizar consultas a pedidos, usando seu id na tabela como parâmetro para tal. Além disso, é feita uma condição em que a forma de pagamento do pedido deve ser o PIX. Isso ocorre porque espera-se que haja muitas linhas com esse valor, já que nosso produto é bem comum e acessível no geral. Sendo assim, visando aumentar a performance das consultas, optou-se por selecionar o caso de consulta mais comum. A figura 2 indica o script usado para tal.

Figura 2 - Script de IDX001

```
CREATE INDEX idx_pedido ON pedidos (id_pedido) WHERE forma_pagamento = "PIX";
```

5.2. [IDX002] Tabela Produtos

Esse índice foi criado com o intuito de agilizar consultas a produtos, usando seu id na tabela como parâmetro para tal. Além disso, são feitas duas condições que indicam o caso mais comum de produto que temos cadastrado: papel cartão de cor branca. Isso ocorre porque espera-se que haja muitas linhas com esse valor. Sendo assim, visando aumentar a performance das consultas, optou-se por selecionar o caso de consulta mais comum. A figura 3 indica o script usado para tal.

Figura 3 - Script de IDX002

```
CREATE INDEX idx_produto ON produtos (id) WHERE cor="branco" AND tipo ="PapelCartão";
```

5.3. [IDX003] Tabela Clientes

Esse índice foi criado com o intuito de agilizar consultas a produtos, usando seu id na tabela como parâmetro para tal. A figura 4 indica o script usado para tal.

Figura 4 - Script de IDX003

```
CREATE INDEX idx_cliente ON clientes(id)
```

5.4. [IDX004] Tabela Carrinho

Esse índice foi criado com o intuito de agilizar a carrinhos de compra, usando seu id na tabela como parâmetro para tal. A figura 5 indica o script usado para tal.

Figura 5 - Script de IDX004

```
CREATE INDEX idx_carrinho ON carrinho(id);
```

5.5. [IDX005] Tabela Entrega

Para essa tabela, foram criados dois índices, um para identificar o tipo da entrega e outro para mapear unidades federativas. Primeiramente falando sobre o index no tipo de entrega, fizemos sua criação filtrada como “Standard” por ser uma modalidade bem popular no *e-comnerce* vide conforto e baixo preço. Já para as unidades federativas, fizemos o filtro apenas nas quatro que mais consomem os produtos, vide alta demanda nesses estados. A figura 6 indica o script usado para tal.

Figura 6 - Script de IDX005

```
CREATE INDEX idx_tipo_entrega ON entrega (tipo_entrega) WHERE tipo_entrega = "Standard";
--Espera-se que boa parte das linhas da tabela entrega tenham como tipo da entrega Standard
CREATE INDEX idx_uf ON entrega (uf) WHERE uf IN ('SP', 'MG', 'RJ', 'SC');
--As UFs em questão devem representar a maioria por serem as mais ricas e populosas, por co
```

5.6. [IDX006] Tabela Armazém

Esse índice foi criado com o intuito de agilizar consultas aos armazéns, usando o parâmetro uf e filtrando as três UFs com mais armazéns cadastrados, e consequentemente mais acessados no banco de dados. A figura 7 indica o script usado para tal.

Figura 7 - Script de IDX006

```
CREATE INDEX idx_uf ON armazem (uf) WHERE uf IN ('SP', 'MG', 'PR');
```

5.7. [IDX007] Tabela Nota Fiscal

Para essa tabela, foram criados quatro índices. No primeiro foi feito um simples índice pelo id da nota fiscal em sua tabela. Já no segundo, fez-se um índice pelo seu protocolo. Depois, no terceiro, fez-se outro índice com base na chave de acesso e por fim mais um índice para UF, com as quatro que mais emitem como parâmetro. Para os três primeiros índices dessa tabela a justificativa é que são campos com valores únicos e que portanto lotarão a tabela, o que pede um índice para fazer a consulta pela árvore B ao invés da própria tabela (mais pesada). Já no último caso, filtrou-se apenas os casos mais numerosos de UF para o índice fazer sentido do ponto de vista do custo da operação. A figura 8 indica o script usado para tal.

```
CREATE INDEX idx_nf ON nota_fiscal(id); --Agiliza a busca pela NF
CREATE INDEX idx_protocolo ON nota_fiscal(protocolo); --Agiliza a busca pelos
CREATE INDEX idx_chave_acesso ON nota_fiscal(chave_acesso); --Agiliza a busca
CREATE INDEX idx_uf ON nota_fiscal (uf) WHERE uf IN ('SP', 'MG', 'RJ', 'SC');
```

Figura 8 - Script de IDX007

5.8. [IDX008] Tabela Pedidos Arquivados

Para essa tabela foram criados dois índices, um para o id do pedido arquivado e outro para o número da nota fiscal. Isso porque ambos são únicos e lotam a tabela facilmente com várias linhas. Sendo assim, visando evitar consultas à tabela propriamente dita e sim à árvore B, empregamos esses dois índices. A figura 9 indica o script usado para tal.

Figura 9 - Script de IDX008

```
CREATE INDEX idx_pedido_arquivado ON pedidos_arquivados(id); --Agiliza a busca pelo pedido
CREATE INDEX idx_nf ON pedidos_arquivados(nf); --Agiliza a busca por NFs
```

6. Views

6.1. [VW001] Pedidos arquivados 2023.1

Exibe os pedidos arquivados com data_pedido dentro do primeiro semestre de 2023.

Figura 10 - Script de VW001

```sql
-- CRIAR UMA VIEW QUE EXIBA PEDIDOS ARQUIVADOS COM DATA_PEDIDO DENTRO DO PRIMEIRO SEMESTRE DE 2023
CREATE VIEW PEDIDOS_ARQUIVADOS_23_1 AS
SELECT
    *
FROM
    PEDIDOS
WHERE
    CONFIRMAR_PAGAMENTO = TRUE
    AND YEAR(DATA_PEDIDO) = 2023
    AND MONTH(DATA_PEDIDO) <= 6;
```

6.2. [VW002] Total gasto pelo cliente mais pedinte

Exibe o total gasto pelo cliente com mais pedidos feitos.

Figura 11 - Script de VW002

```sql
-- FAZER UMA VIEW EM PEDIDOS QUE RETORNA O TOTAL GASTO PELO CLIENTE COM MAIS PEDIDOS FEITOS
CREATE VIEW gastos_cliente_n1 AS
SELECT
    CLIENTE_ID,
    SUM(TOTAL_GASTO) AS TOTAL_GASTO
FROM
    PEDIDOS
WHERE
    CLIENTE_ID = (
        SELECT
            CLIENTE_ID
        FROM
            PEDIDOS
        GROUP BY
            CLIENTE_ID
        ORDER BY
            COUNT(*) DESC
        LIMIT
            1
    )
GROUP BY
    CLIENTE_ID;
```

6.3. [VW003] TOP 3 do armazém

Retorna os 3 produtos em maior quantidade no armazém.

Figura 12 - Script de VW003

```
-- FAZER UMA VIEW QUE RETORNA OS 3 PRODUTOS EM MAIOR QUANTIDADE NO ARMAZÉM.
CREATE VIEW TOP_3_ESTOQUE AS
SELECT
    PRODUTO_ID,
    TIPO_PRODUTO,
    QUANTIDADE_ESTOQUE
FROM
    ARMAZEM
ORDER BY
    QUANTIDADE_ESTOQUE DESC
LIMIT
    3;
```

7. Functions

7.1. [FCT001] Calcular frete

Baseado no tipo_entrega da tabela entrega calcula o frete e adiciona o valor à columa frete em pedidos (tabela). Sendo padrão retornando 0.00, expresso retornando 16.32 e premium retornando 26.50.

Figura 13 - Script de FCT001

```
CREATE
OR REPLACE FUNCTION CALCULAR_FRETE() RETURNS VOID AS $ $ BEGIN
UPDATE
    PEDIDOS AS P
SET
    FRETE = CASE
        E.TIPO_ENTREGA
        WHEN 'PADRÃO' THEN 0.00
        WHEN 'EXPRESSO' THEN 16.32
        WHEN 'PREMIUM' THEN 26.50
        ELSE 0.00
    END
FROM
    ENTREGA AS E
WHERE
    P.DESTINO = E.ID;

END;

$ $ LANGUAGE PLPGSQL;
```

7.2. [FCT002] Calcular Carrinho

Soma valores da tabela carrinho (preco_total_produto) e de forma a alimentar a columa total_carrinho da tabela pedidos.

Figura 14 - Script de FCT002

```
CREATE
OR REPLACE FUNCTION CALCULAR_CARRINHO() RETURNS VOID AS $ $ BEGIN
UPDATE
    PEDIDOS AS P
SET
    TOTAL_CARRINHO = (
        SELECT
            SUM(C.PRECO_TOTAL_PRODUTO)
        FROM
            CARRINHO AS C
        WHERE
            C.PEDIDO = P.ID_PEDIDO
    );

END $ $ LANGUAGE PLPGSQL;
```

7.3. [FCT003] Total a Pagar

Soma os valores de frete e total_carrinho e subtrai do valor_desconto. Depois, joga a saída dessa operação na coluna total_a_pagar na tabela pedidos.

Figura 15 - Script de FCT003

```
CREATE
OR REPLACE FUNCTION CALCULAR_TOTAL_A_PAGAR() RETURNS VOID AS
$ $
BEGIN
    UPDATE
        PEDIDOS
    SET
        TOTAL_A_PAGAR = (
            TOTAL_CARRINHO + FRETE - VALOR_DESCONTO
        );
END;
$ $ LANGUAGE PLPGSQL;
```

8. Triggers

8.1. [TGG001] Gerar NF e arquivar

Ao trocar o valor de confirmacao_pagamento em pedidos para true cria uma NF e depois transfere esse pedido para a tabela pedidos_arquivados.

Figura 16 - Script de TGG001 (1)

```
CREATE OR REPLACE FUNCTION END_ORDER() RETURNS TRIGGER AS
$$
DECLARE
    FLAG BOOLEAN
    BASEICMS REAL =: 28.0
    PERCENTUAL_ICMS REAL =: 0.12
BEGIN
    SELECT CONFIRMACAO_PAGAMENTO FROM PEDIDOS INTO FLAG;
    IF FLAG = TRUE THEN
        INSERT INTO NOTA_FISCAL(
            NOME_EMPRESA, ENTRADA_SAIDA, NUMERO_NF, CHAVE_ACESSO, NATUREZA_OPERACAO, PROTOCOLO, CNPJ, DATA_EMISSAO,
            ADDRESS, NUMERO, ESTADO, UF, BAIRRO, CEP, TELEFONE, BASE_ICMS, VALOR_ICMS, VALOR_FRETE, DESCONTO,
            TOTAL_TRIBUTOS, TOTAL_PRODUTOS, TOTAL_NF, ADDED_AT
        )
        VALUES (
            "DF PAPER COMPANY",
            1,
            CONCAT('N°.', NEW.ID),
            (SELECT STRING_AGG(TO_CHAR(FLOOR(RANDOM() * 10), '0000'), ' ') FROM GENERATE_SERIES(1, 11)),
            'VENDA',
            CONCAT(TO_CHAR(FLOOR(RANDOM() * 1000000000000000), '000000000000000'), ' ', TO_CHAR(NOW(), 'DD/MM/YYYY HH24:MI:SS')),
            '00750343000173' --PODE VIR A MUDAR A MANEIRA COMO É PREENCHIDO,
            NOW(),
            (SELECT ADDRESS FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            (SELECT NUMERO FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            (SELECT ESTADO FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            (SELECT UF FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            (SELECT BAIRRO FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            (SELECT CEP FROM ENTREGA AS E JOIN PEDIDOS AS P ON P.DESTINO = ENTREGA.ID),
            '35899337',
            BASEICMS,
            BASEICMS*PERCENTUAL_ICMS,
            (SELECT FRETE FROM PRODUTOS AS P WHERE OLD.ID_PEDIDO = NEW.ID_PEDIDO),
            (SELECT DESCONTO FROM PRODUTOS AS P WHERE OLD.ID_PEDIDO = NEW.ID_PEDIDO),
            BASEICMS*PERCENTUAL_ICMS,
            (SELECT TOTAL_A_PAGAR - (BASEICMS*PERCENTUAL_ICMS) - FRETE - DESCONTO FROM PEDIDOS),
            (SELECT TOTAL_A_PAGAR FROM PEDIDOS),
            NOW()
        )

        -- MOVER A LINHA CORRESPONDENTE DE PEDIDOS PARA PEDIDOS_ARQUIVADOS
        INSERT INTO PEDIDOS_ARQUIVADOS
        SELECT * FROM PEDIDOS WHERE ID = NEW.ID;

        -- DELETAR A LINHA DA TABELA PEDIDOS
        DELETE FROM PEDIDOS WHERE ID = NEW.ID;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
```

Figura 17 - Script de TGG001 (2)

```
CREATE OR REPLACE TRIGGER FECHAR_PEDIDO
AFTER UPDATE ON PEDIDOS
FOR EACH ROW
EXECUTE FUNCTION END_ORDER();
```

8.2. [TGG002] Resolver UFs

Faz a resolução de UF baseado na coluna estado.

Figura 18 - Script de TGG002 (1)

```
CREATE OR REPLACE FUNCTION LINK_UF_STATE() RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
BEGIN
    -- SUDESTE
    IF NEW.ESTADO = 'RIO DE JANEIRO' THEN
        NEW.UF := 'RJ';
    ELSE IF NEW.ESTADO = 'SÃO PAULO' THEN
        NEW.UF := 'SP';
    ELSE IF NEW.ESTADO = 'ESPÍRITO SANTO' THEN
        NEW.UF := 'ES';
    ELSE IF NEW.ESTADO = 'MINAS GERAIS' THEN
        NEW.UF := 'MG';
    -- SUL
    ELSE IF NEW.ESTADO = 'PARANÁ' THEN
        NEW.UF := 'PR';
    ELSE IF NEW.ESTADO = 'SANTA CATARINA' THEN
        NEW.UF := 'SC';
    ELSE IF NEW.ESTADO = 'RIO GRANDE DO SUL' THEN
        NEW.UF := 'RS';
    -- CENTRO OESTE
    ELSE IF NEW.ESTADO = 'MATO GROSSO DO SUL' THEN
        NEW.UF := 'MS';
    ELSE IF NEW.ESTADO = 'GOIÂNIA' THEN
        NEW.UF := 'GO';
    ELSE IF NEW.ESTADO = 'MATO GROSSO' THEN
        NEW.UF := 'MT';
    -- NORDESTE
    ELSE IF NEW.ESTADO = 'BAHIA' THEN
        NEW.UF := 'BA';
    ELSE IF NEW.ESTADO = 'SERGIPE' THEN
        NEW.UF := 'SE';
    ELSE IF NEW.ESTADO = 'ALAGOAS' THEN
        NEW.UF := 'AL';
    ELSE IF NEW.ESTADO = 'PERNAMBUCO' THEN
        NEW.UF := 'PE';
    ELSE IF NEW.ESTADO = 'PIAUÍ' THEN
        NEW.UF := 'PI';
    ELSE IF NEW.ESTADO = 'MARANHÃO' THEN
        NEW.UF := 'MA';
    ELSE IF NEW.ESTADO = 'PARAÍBA' THEN
        NEW.UF := 'PB';
    ELSE IF NEW.ESTADO = 'RIO GRANDE DO NORTE' THEN
        NEW.UF := 'RN';
    ELSE IF NEW.ESTADO = 'CEARÁ' THEN
        NEW.UF := 'CE';
    -- NORTE
    ELSE IF NEW.ESTADO = 'RONDÔNIA' THEN
        NEW.UF := 'RO';
    ELSE IF NEW.ESTADO = 'AMAZONAS' THEN
        NEW.UF := 'AM';
    ELSE IF NEW.ESTADO = 'TOCANTINS' THEN
        NEW.UF := 'TO';
    ELSE IF NEW.ESTADO = 'PARÁ' THEN
        NEW.UF := 'PA';
    ELSE IF NEW.ESTADO = 'PARAÍBA' THEN
        NEW.UF := 'PB';
    ELSE
        RAISE EXCEPTION 'ESTADO INVÁLIDO';
    END IF;

    RETURN NEW;
END;
$$;
```

Figura 19 - Script de TGG002 (2)

```
-- TRIGGER PARA A TABELA ENTREGA
CREATE TRIGGER RESOLVE_UF_ENTREGA
BEFORE INSERT OR UPDATE ON ENTREGA
FOR EACH ROW
EXECUTE FUNCTION LINK_UF_STATE();

-- TRIGGER PARA A TABELA FORNECEDORES
CREATE TRIGGER RESOLVE_UF_FORNECEDORES
BEFORE INSERT OR UPDATE ON FORNECEDORES
FOR EACH ROW
EXECUTE FUNCTION LINK_UF_STATE();

-- TRIGGER PARA A TABELA ARMAZEM
CREATE TRIGGER RESOLVE_UF_ARMAZEM
BEFORE INSERT OR UPDATE ON ARMAZEM
FOR EACH ROW
EXECUTE FUNCTION LINK_UF_STATE();
```

8.3. [TGG003] Impedir input inválido

Impede novas inserções em armazém quando a quantidade de produtos em um armazém superar sua capacidade total.

Figura 20 - Script de TGG003

```
CREATE OR REPLACE FUNCTION CHECK_CAPACIDADE_ARMAZEM() RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
DECLARE
    QTD_ARMAZENADOS SMALLINT;
BEGIN
    SELECT COUNT(ID) INTO QTD_ARMAZENADOS
    FROM ARMAZEM
    WHERE ID = NEW.ID;


    IF QTD_ARMAZENADOS >= 300 THEN -- 300 FOI ARBITRÁRIO, PODE SER MUDADO DEPOIS
        RAISE EXCEPTION 'CAPACIDADE DO ARMAZÉM NÃO PODE SER EXCEDIDA. FALHA AO INSERIR NOVO ITEM!';
    END IF;

    RETURN NEW;
END;
$$;



CREATE TRIGGER MONITORAR_CAPACIDADE_ESTOQUE
BEFORE INSERT ON ARMAZEM
FOR EACH ROW
EXECUTE FUNCTION CHECK_CAPACIDADE_ARMAZEM();
```

9. Otimização

10. Monitoramento

11. Conclusão

12. Bibliografia

Nasser, H. (30 de Setembro de 2020). *Database Indexing Explained (with PostgreSQL)*. Obtido em 17 de Maio de 2024, de Site do Youtube: https://www.youtube.com/watch?v=-qNSXK7s7_w&list=PL4utN2dh7VSpwUWkzwx6FVu-9u4cEO9N4