INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DA PARAÍBA - CAMPUS CAJAZEIRAS
CURSO: ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAMILA FERREIRA DE CARVALHO
EDUARDO LUCAS CRISPIM DE CASTRO

SISTEMA PARA GERENCIAMENTO DE LOJA DE VESTUÁRIO

Cajazeiras –PB
2018

CAMILA FERREIRA DE CARVALHO
EDUARDO LUCAS CRISPIM DE CASTRO

SISTEMA PARA GERENCIAMENTO DE LOJA DE VESTUÁRIO

Trabalho apresentado por alunos do Curso Superior Tecnológico em Análise e Desenvolvimento de Sistemas do IFPB – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba- Campus Cajazeiras, como nota na disciplina Banco de Dados,
Professor Fabio Gomes de Andrade.

Cajazeiras –PB
2018

Sumário

1 Introdução

A correria do dia a dia no comércio acaba, por muitas vezes, deixando o controle de informações de último plano. Sistemas de informação beneficiam evidentemente a prestação de serviços realizada por um estabelecimento, já que automatizam buscas, inserção de dados já armazenados e agilizam controle e realização de atividades. Neste trabalho promovemos o gerenciamento da She He, uma loja de vestuário. Lojas como essa são caracterizadas pelo controle de entrada e saída de produtos. Assim, facilitaremos a consulta de clientes e produtos cadastrados, analisando problemas e mantendo a praticidade no uso da ferramenta. O objetivo é evitar perdas de informação, mau uso de investimentos e prejuízos.

2 Modelo Conceitual

2.1 Levantamento de Requisitos

A loja de vestuário contrata funcionários, dos quais serão armazenados: nome, CPF, telefone, endereço, cargo, data de admissão, salário atual e o status. Um fornecedor disponibiliza produtos, que são comprados pela loja. Para cada fornecedor, armazena-se o nome, telefone, endereço e o CNPJ. Para cada produto, são armazenados dados como: código, preço atual de venda, nome, tamanho da peça e o status. Para cada compra realizada ao fornecedor, são salvos a hora e a quantidade adquirida pela loja.

A venda de produtos a clientes será armazenada. Clientes cadastrados têm seu nome, CPF, telefone e e-mail armazenados. As informações da venda também precisam ser salvas, registra-se a hora. Caso o cliente pague à vista, nenhuma informação é acrescentada, porém, se for no cartão, o número do cartão e a bandeira são coletados, junto com a quantidade de prestações. Esse processo de venda é atendido pelo funcionário. O pagamento do salário de cada funcionário também será controlado, para cada pagamento serão armazenados a hora.

As despesas e entradas financeiras da loja serão armazenadas para que promova um melhor controle. Despesas podem ser determinadas pela compra de produtos a fornecedores e pelo pagamento do salário a funcionários. Entradas são determinadas pela venda de produtos a clientes. Para cada ocorrência de um destes, armazena-se a data e o valor.

De acordo com os dados citados acima, a entrada financeira do estabelecimento pode ser dada pelo valor adquirido em cada venda. Como a data é salva, a entrada de cada mês pode ser consultada, ajudando a criar estimativas e comparar dados entre os meses. Assim, a dona da loja pode analisar quais os motivos para tal ganho, evitando prejuízos nos próximos meses.

Os funcionários devem estar agrupados para facilitar a identificação de seus dados. É interessante para o estabelecimento que periodicamente sejam levantados quais produtos atraem mais clientes, consequentemente serão os mais pedidos aos fornecedores.

Ainda no que se diz respeito a funcionários, o status do mesmo é necessário para manter dados históricos da loja. Assim, caso um funcionário seja recontratado, os dados podem ser resgatados.
A respeito dos clientes cadastrados, é importante também ter acesso rápido a seu e-mail ou seu telefone, por tanto, ao enviar o código de cadastro do cliente, deve-se retornar os dados do mesmo.
Deve-se manter um monitoramento a respeito dos produtos, o atributo status é importante para saber se o mesmo está ou não disponível na loja. Caso o produto esteja disponível, a venda ocorre mais rapidamente, caso contrário, há o conhecimento acerca de futuras compras a fornecedores.

A dona do estabelecimento preza muito pelo bom atendimento ao cliente, pensando nisso, procura-se sempre analisar as vendas de cada funcionário. Além disso, dados como tamanho de uma peça, podem ser usados para definir qual tamanho será mais pedido em roupas, já que buscaremos qual vende mais. Desta forma, o banco de dados será usado

como um meio para traçar estratégias de vendas, controlar entradas e saídas financeiras e analisar clientes e funcionários.

2.2Diagrama

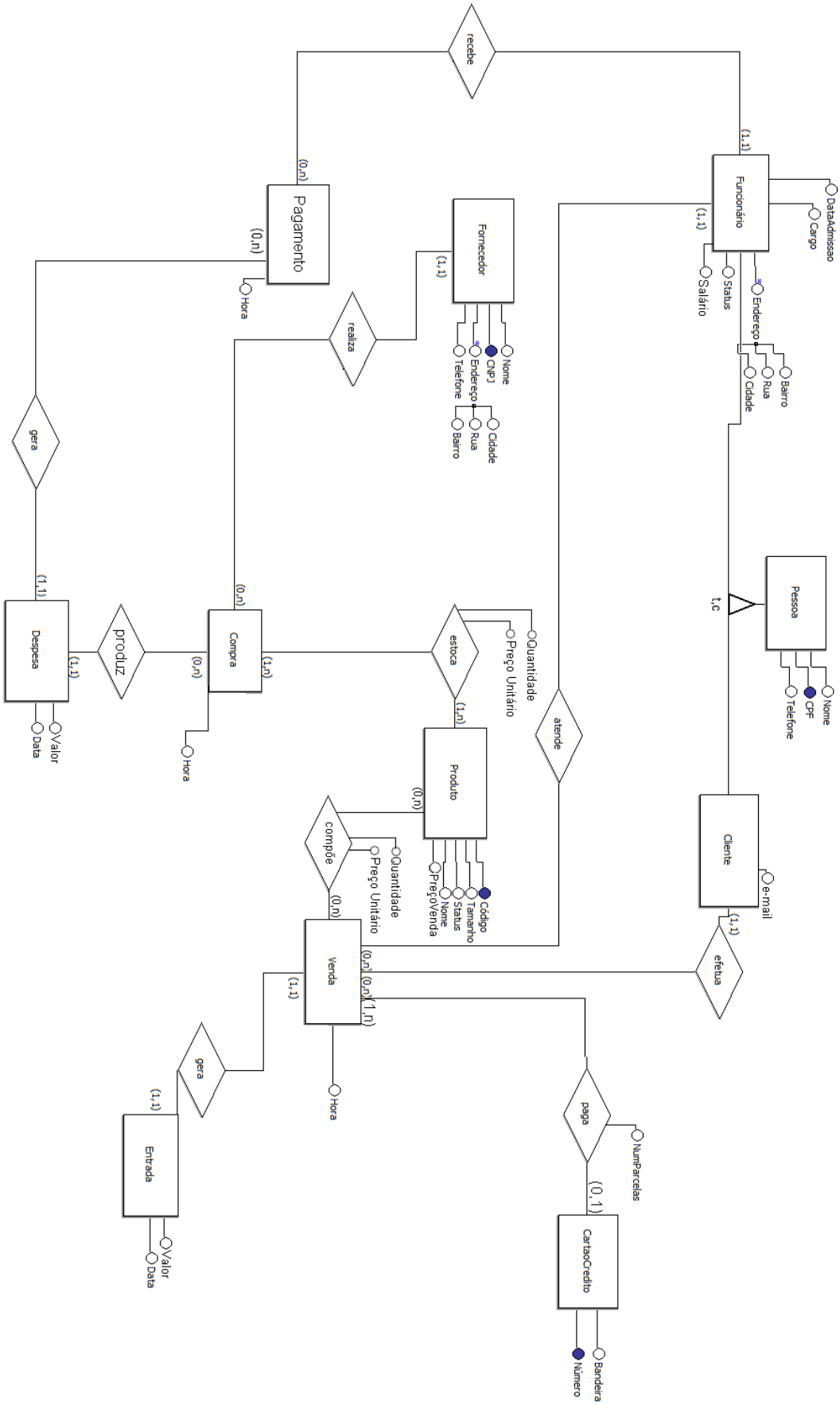

Figura 1: Diagrama de entidade e relacionamento

2.3 Dicionário conceitual de dados

Entidade Pessoa:
Generalização total e compartilhada para Cliente e Funcionário, armazena informações em comum entre essas duas entidades, ou seja, dados gerais.
Atributos:
• Nome: armazena o nome da pessoa.
• CPF: armazena o número do CPF da pessoa.
• Telefone: armazena o número principal da pessoa.
Relacionamentos:
Não há relacionamentos.

Entidade Funcionário:
Se trata de uma especialização da entidade Pessoa, armazena dados específicos de um funcionário.
Atributos:
• Cargo: salva o cargo exercido pelo funcionário na loja.
• Salário: salva o valor atual de salário.
• Status: armazena se o funcionário está trabalhando no estabelecimento ou se já foi demitido.
• DataAdmissao: armazena a data de admissão do funcionário.
• Endereço: armazena o endereço do funcionário, é composto por: rua, bairro e cidade.
Relacionamentos:
• recebe: Uma relação entre as entidades Funcionário e Pagamento. Um funcionário cadastrado no banco recebe zero ou vários pagamentos, mas um pagamento só pode ser recebido apenas por um funcionário.
• atende: Relação entre Funcionário e Venda. Um funcionário cadastrado atende zero ou muitas vendas, e cada venda só pode ser atendida por um funcionário.

Entidade Cliente:
Especificação da entidade Pessoa, armazena dados que especificam um cliente cadastrado.
Atributos:
• E-mail: armazena o e-mail do cliente.
Relacionamentos:
• efetua: Uma relação entre Cliente e Venda. Um cliente cadastrado pode realizar zero ou muitas vendas, mas uma venda só pode ser realizada por um cliente.

Entidade Fornecedor:
Armazena os dados que serão tratados de um fornecedor.
Atributos:
• Nome: armazena o nome do fornecedor.
• CNPJ: armazena o número do CNPJ do fornecedor.
• Endereço: armazena o endereço do fornecedor, composto por: rua, bairro e cidade.
• Telefone: armazena o número principal de telefone do fornecedor.
Relacionamentos:
• realiza: Relação entre Fornecedor e Compra. Um fornecedor pode realizar zero ou várias compras da loja, e cada compra cadastrada será apenas de um fornecedor.

Entidade Venda
Armazena os dados gerais de uma venda.

Atributos:
• Hora: armazena a hora da venda.
Relacionamentos:
• gera: Relação entre Venda e Entrada. Cada venda gera uma entrada, e cada entrada é gerada por uma venda.
• efetua: Uma relação entre Venda e Cliente. Um cliente cadastrado pode realizar zero ou muitas vendas, mas uma venda só pode ser realizada por um cliente.
• paga: Relação entre Venda e CartaoCredito, armazena o número de parcelas. Uma venda pode ser paga por zero ou um cartão, um cartão pode pagar uma ou várias vendas.
• atende: Relação entre Funcionário e Venda. Um funcionário cadastrado atende zero ou muitas vendas, e cada venda só pode ser atendida por um funcionário.
•compõe: Relacionamento entre Produto e Venda. Armazena a quantidade e o preço unitário de cada produto. Um produto pode compor zero ou muitas vendas, e uma venda pode ser comporta por zero ou muitos produtos.

Entidade Produto
Armazena os dados de um produto.
Atributos:
• Código: armazena o código do produto.
• Tamanho: armazena o tamanho do produto.
• Status: armazena o status do produto, se está disponível ou não.
• Nome: armazena o nome do produto.
• PreçoVenda: armazena o preço atualizado para venda do produto.
Relacionamentos:
• estoca: Relação entre Compra e Produto. A compra pode ser de um a muitos produtos, de mesma forma, o produto provém de uma ou mais compras. Armazena quantidade e preço unitário de cada produto.
•compõe: Relacionamento entre Produto e Venda. Armazena a quantidade e o preço unitário de cada produto. Um produto pode compor zero ou muitas vendas, e uma venda pode ser comporta por zero ou muitos produtos.

Entidade Compra
Armazena os dados da compra de produtos de um fornecedor.
Atributos:
• Hora: salva hora em que a compra foi realizada
Relacionamento:
• estoca: Relação entre Compra e Produto. A compra pode ser de um a muitos produtos, de mesma forma, o produto provém de uma ou mais compras. O relacionamento armazena a quantidade e o preço unitário.
• realiza: Relação entre Fornecedor e Compra. Um fornecedor pode realizar zero ou várias compras, e cada compra cadastrada será apenas de um fornecedor.
• produz: Relação entre Compra e Despesa. Uma compra pode produzir apenas uma despesa, mas uma despesa pode ser produto de zero ou várias compras.

Entidade CartaoCredito
Armazena os dados da compra por um cartão de crédito
Atributos:
• Bandeira: armazena a bandeira do cartão.
• Número: armazena o número do cartão.
Relacionamentos:
•paga: Relação entre Venda e CartaoCredito, armazena o número de parcelas. Uma venda pode ser paga por zero ou um cartão, um cartão pode pagar uma ou várias vendas.

Entidade Despesa
Armazena as despesas da loja em compras e salários.
Atributos:
• Data: armazena as datas das despesas.
• Valor: armazena o valor de cada despesa.
Relacionamentos:
• produz: Relação entre Compra e Despesa. Uma compra pode gerar apenas uma despesa, mas uma despesa pode ser gerada por várias compras.
• gera: Relação entre Despesa e Pagamento. Um pagamento gera uma despesa, uma despesa é gerada por zero ou mais salários.

Entidade Entrada
Armazena a entrada de uma venda.
Atributos:
• Data: armazena a data da entrada da venda.
• Valor: armazena o valor de cada entrada.
Relacionamentos:
• gera: Relação entre Venda e Entrada. Cada venda gera uma entrada, e cada entrada é gerada por uma venda.

Entidade Pagamento
Armazena dados de um pagamento de salário a um funcionário.
Atributos:
• Hora: armazena o horário em que o pagamento foi realizado.
Relacionamentos:
• recebe: Relação entre Pagamento e a entidade Funcionário. Cada pagamento só pode ser a um funcionário, e cada funcionário cadastrado pode receber zero ou mais pagamentos.
• gera: Relação entre Pagamento e Despesa. Cada pagamento gera uma despesa, e cada despesa é gerada por zero ou vários pagamentos.

3 Modelo Logico

3.1 Mapeamento Entidade-Relacionamento

Pessoa= (CPF, Nome, Telefone);
Funcionario= (CPFPessoa, Cargo, DataAdmissao, Salario, Status, Bairro, Rua, Cidade);
Cliente= (CPFPessoa, email);
Pagamento= (CodPagamento, Funcionario, Hora, CodDespesa);
Fornecedor= (CNPJ, Nome, Telefone, Cidade, Rua, Bairro);
Despesa= (CodDespesa,Valor, Data);
Compra= (CodCompra, Fornecedor, Hora, CodDespesa);
CompraProduto= (CodProduto, CodCompra, Quantidade, PrecoUnitario);
Produto= (Codigo, Tamanho, Status, Nome, PrecoVenda);
Venda= (CodVenda, Hora, Funcionario, Cliente, CodEntrada);
VendaProduto= (CodProduto, CodVenda, Quantidade, PrecoUnitario);
Entrada= (CodEntrada, Valor, Data);
CartaoCredito= (Numero, Bandeira);
VendaCartao= (NumCartao, CodVenda, NumParcelas);

3.2 Dicionário Lógico de Dados

PESSOA : Relação que armazena dados generalizados de funcionários e clientes.

Atributo	Descrição	Tipo	Domínio	Restrição
Nome	Representa o nome de uma pessoa, seja funcionário ou cliente	VARCHAR(100)	VARCHAR(100)	Não nulo
CPF	Representa o número do CPF de uma pessoa, seja funcionário ou cliente	VARCHAR(20)	VARCHAR(20)	Chave Primaria
Telefone	Representa o número do telefone de uma pessoa, seja funcionário ou cliente	VARCHAR(20)	VARCHAR(20)	Não nulo

Tabela 1 – Relação PESSOA

FUNCIONARIO: Relação que armazena dados específicosde funcionários.

Atributo	Descrição	Tipo	Domínio	Restrição
CPFPessoa	Representa o CPF do funcionário	VARCHAR(20)	VARCHAR(20)	-Chave estrangeira referenciando o CPF na entidade PESSOA -Chave Primaria
Cargo	Representa o cargo de um funcionário no estabelecimento	VARCHAR(20)	VARCHAR(20)	Não nulo
DataAdmissao	Representa a data em que o funcionário foi admitido no estabelecimento	Date	Date	Não nulo
Salário	Representa valor atual do	REAL	Números reais positivos	Não nulo

	salário de funcionários			
Status	Representa o status do funcionário no estabelecimento, se está ou não efetivo	VARCHAR(20)	Efetivo ou Não Efetivo	Não nulo
Bairro	Representa o bairro onde mora o funcionario	VARCHAR(100)	VARCHAR(100)	Não nulo
Rua	Representa a rua onde vive o funcionário	VARCHAR(100)	VARCHAR(100)	Não nulo
Cidade	Representa a cidade do funcionário	VARCHAR(100)	VARCHAR(100)	Não nulo

Tabela 2 – Relação FUNCIONARIO

CLIENTE: Entidade que armazena os dados de um cliente.

Atributo	Descrição	Tipo	Domínio	Restrição
CPFPessoa	Representa o CPF do cliente	VARCHAR(20)	VARCHAR(20)	-Chave Primária -Chave estrangeira que referencia CPFPessoa na tabela CLIENTE
Email	Representa o email do cliente	VARCHAR(100)	VARCHAR(100)	Não nulo

Tabela 3 – Entidade CLIENTE

PAGAMENTO: Relação que armazena dados sobre o pagamento do salário de um funcionário.

Atributo	Descrição	Tipo	Domínio	Restrição
CodPagamento	Representa o número do código do pagamento do salário	INT	Números inteiros positivos	-Chave primária -Chave substituta
Funcionario	Representa o CPF do funcionário que irá receber o pagamento	VARCHAR(20)	String(20)	-Não nulo -Chave estrangeira referenciando CPF em Pessoa

Hora	Representa a hora do pagamento	Time	Time	Não nulo
CodDespesa	Representa o número do código da despesa financeira	Int	Números inteiros positivos	-Chave Estrangeira referenciando CodDespesa em DESPESA -Não nulo

Tabela 4 – Entidade SALARIO

Fornecedor: Entidade que armazena os dados de fornecedores do estabelecimento.

Atributo	Descrição	Tipo	Domínio	Restrição
CNPJ	Representa o CNPJ do fornecedor	VARCHAR(20)	VARCHAR(20)	Chave Primária
Nome	Representa o nome do fornecedor	VARCHAR(100)	VARCHAR(100)	Não nulo
Telefone	Representa o número de telefone do fornecedor	VARCHAR(20)	VARCHAR(20)	Não nulo
Cidade	Representa a cidade onde se encontra o fornecedor	VARCHAR(50)	VARCHAR(50)	Não nulo
Rua	Representa a rua onde se encontra o fornecedor	VARCHAR(100)	VARCHAR(100)	Não nulo
Bairro	Representa o bairro onde se encontra o fornecedor	VARCHAR(100)	VARCHAR(100)	Não nulo

Tabela 5 – Entidade FORNECEDOR

Despesa: Entidade que armazena as despesas da loja.

Atributo	Descrição	Tipo	Domínio	Restrição
CodDespesa	Representa o código da despesa	INT	Números inteiros positivos	-Chave Primária -Chave substituta
Valor	Representa o valor total da despesa	Real	Números reais positivos	Não nulo
Data	Representa a data em que a	Date	Date	Não nulo

	despesa aconteceu			

Tabela 6 – Entidade DESPESA

Compra: Entidade que armazena dados de uma compra.

Atributo	Descrição	Tipo	Domínio	Restrição
CodCompra	Representa o código da compra	INT	Números inteiros positivos	-Chave Primária -Chave substituta
Fornecedor	Representa o CNPJ do fornecedor	VARCHAR(20)	VARCHAR(20)	-Chave Estrangeira representando NOME em FORNECEDOR -Não nulo
Hora	Representa a hora da compra	Time	Time	Não nulo
CodDespesa	Representa o código da despesa	Int	Números inteiros positivos	-Chave Estrangeira representando CodDespesa em DESPESA -Não nulo

Tabela 7 – Entidade COMPRA

CompraProduto: Relação que armazena o produto comprado ao fornecedor.

Atributo	Descrição	Tipo	Domínio	Restrição
CodProduto	Representa o código do produto	Int	Números inteiros positivos	-Chave Estrangeira referenciando CODIGO em PRODUTO -Forma a chave primária junto com CodCompra
CodCompra	Representa o código da compra	Int	Números inteiros positivos	-Chave Estrangeira referenciando CodCompra em COMPRA -Não nulo -Forma a chave primária junto com CodProduto

Quantidade	Reepresenta a quantidade de itens comprados	Int	Int	Não nulo
PrecoUnitario	Representa o preço de cada unidade comprada	Real	Números reais positivos	Não nulo

Tabela 8 – Relação COMPRAPRODUTO

Produto: Entidade que armazena os dados de um produto.

Atributo	Descrição	Tipo	Domínio	Restrição
Código	Representa o código do produto	INT	Números inteiros positivos	Chave Primária
Tamanho	Representa o tamanho do produto comprado	VARCHAR(5)	PP, P, M, G, GG ou Único	Não nulo
Status	Representa o status do produto	VARCHAR(12)	Disponível ou Indisponível	Não nulo
Nome	Representa o nome do produto	VARCHAR(20)	VARCHAR(20)	Não nulo
PrecoVenda	Representa o preço atualizado de venda do produto	Real	Números reais positivos	Não nulo

Tabela 9 – Entidade PRODUTO

Venda: Entidade que armazena os dados de uma venda.

Atributo	Descrição	Tipo	Domínio	Restrição
CodVenda	Representa o código da venda	INT	Números inteiros positivos	-Chave Primária -Chave Substituta
Hora	Representa a hora em que a venda foi realizada	Time	Time	Não nulo
Funcionario	Representa o CPF do funcionário	VARCHAR(20)	VARCHAR(20)	-Não nulo -Chave estrangeira referenciando o CPF do funcionário na tabela PESSOA

Cliente	Representa o CPF do cliente	VARCHAR(20)	VARCHAR(20)	-Não nulo -Chave estrangeira que referencia CPF do cliente na tabela PESSOA
CodEntrada	Representa o código da entrada	Int	Int	-Não nulo -Chave estrangeira que referencia CodEntrada na tabela Entrada

Tabela 10 – Entidade VENDA

VendaProduto: Entidade que armazena os dados da venda de cada produto.

Atributo	Descrição	Tipo	Domínio	Restrição
CodVenda	Representa o código da venda	INT	Números inteiros positivos	-Chave estrangeira referenciando CodVenda na tabela VENDA -Forma a chave primária, junto com CodProduto
CodProduto	Representa o código do produto	INT	INT	-Chave estrangeira referenciando Codigo na tabela PRODUTO -Forma a chave primária, junto com CodVenda
Quantidade	Representa a quantidade de produtos vendidos	Int	Int	-Não nulo
PrecoUnitario	Representa o preço de cada unidade do produto	REAL	Valores reais positivos	-Não nulo

Tabela 11 – Entidade VENDAPRODUTO

Entrada: Entidade que armazena os dados de um produto.

Atributo	Descrição	Tipo	Domínio	Restrição

CodEntrada	Representa o código da entrada financeira lançada no sistema	INT	Números inteiros positivos	-Chave Primária -Chave Substituta
Valor	Representa o valor total da entrada	Real	Números inteiros reais	Não nulo
Data	Representa a data em que a entrada foi lançada no sistema	Date	Date	Não nulo

Tabela 12 – Entidade ENTRADA

CartaoCredito: Entidade que armazena os dados de um produto.

Atributo	Descrição	Tipo	Domínio	Restrição
Numero	Representa o número do cartão	VARCHAR(50)	VARCHAR(50)	Chave Primária
Bandeira	Representa a bandeira do cartão usado	VARCHAR(20)	VARCHAR(20)	Não nulo

Tabela 13 – Entidade CARTAOCREDITO

VendaCartao: Entidade que armazena os dados de um produto.

Atributo	Descrição	Tipo	Domínio	Restrição
NumCartao	Representa o código do produto	VARCHAR(50)	VARCHAR(50)	Chave Primária
CodVenda	Representa o código da venda realizada com cartão	Int	Int	-Não nulo -Chave estrangeira referenciando CodVenda na tabela VENDA
NumParcelas	Representa o número de parcelas	Int	Números inteiros positivos	Não nulo

Tabela 14 – Entidade VENDACARTAO

4 Modelo Físico
4.1 Scripts SQL

Criando relação Pessoa

```
CREATE TABLE Pessoa(
    cpf VARCHAR(20),
    nome VARCHAR(100) NOT NULL,
    telefone VARCHAR(20) NOT NULL,
    CONSTRAINT PKPessoa PRIMARY KEY(cpf)
    );
```

Criando a relação Funcionario

```
    CREATE TABLE Funcionario(
    cpfPessoa VARCHAR(20),
    cargo VARCHAR(20) NOT NULL,
    dataAdmissao DATE NOT NULL,
    salario REAL NOT NULL,
    status VARCHAR(20) NOT NULL,
    bairro VARCHAR(100) NOT NULL,
    rua VARCHAR(100) NOT NULL,
    cidade VARCHAR(100) NOT NULL,
    CONSTRAINT SalarioPositivo CHECK(salario>0),
    CONSTRAINT PKFuncionario PRIMARY KEY (cpfPessoa),
    CONSTRAINT FKFuncionario FOREIGN KEY (cpfPessoa) REFERENCES
Pessoa(CPF)
    );
```

Criando a relação Cliente

```
    CREATE TABLE Cliente(
    CPFPessoa VARCHAR(20),
    email VARCHAR(100) UNIQUE,
    CONSTRAINT PKCliente PRIMARY KEY(CPFPessoa),
    CONSTRAINT FKCliente FOREIGN KEY(CPFPessoa) REFERENCES Pessoa(CPF)
    );
```

Criando a relação Despesa

```
    CREATE TABLE Despesa(
    CodDespesa INT,
    Valor REAL NOT NULL,
    Data DATE NOT NULL,
    CONSTRAINT PKDespesa PRIMARY KEY(CodDespesa),
    CONSTRAINT DespPositiva CHECK(Valor>0)
    );
```

Criando a relação Pagamento

```
    CREATE TABLE Pagamento(
    CodPagamento INT,
    Funcionario VARCHAR(20) NOT NULL,
```

```
    Hora TIME NOT NULL,
    CodDespesa INT NOT NULL,
    CONSTRAINT PKPagamento PRIMARY KEY(CodPagamento),
    CONSTRAINT FK1Pagamento FOREIGN KEY(CodDespesa) REFERENCES
    Despesa(CodDespesa),
    CONSTRAINT FK2Pagamento FOREIGN KEY(Funcionario) REFERENCES
Pessoa(CPF)
    );
```

Criando a relação Fornecedor

```
    CREATE TABLE Fornecedor(
    CNPJ VARCHAR(20),
    Nome VARCHAR(100) NOT NULL,
    Telefone VARCHAR(20) NOT NULL,
    Cidade VARCHAR(50) NOT NULL,
    Rua VARCHAR(100) NOT NULL,
    Bairro VARCHAR(100) NOT NULL,
    CONSTRAINT PKFornecedor PRIMARY KEY(CNPJ)
    );
```

Criando a relação Compra

```
    CREATE TABLE Compra(
    CodCompra INT,
    Fornecedor VARCHAR(20) NOT NULL,
    Hora TIME NOT NULL,
    CodDespesa INT NOT NULL,
    CONSTRAINT PKCompra PRIMARY KEY (CodCompra),
    CONSTRAINT FK1Compra FOREIGN KEY(Fornecedor) REFERENCES
Fornecedor(CNPJ),
    CONSTRAINT FK2Compra FOREIGN KEY(CodDespesa) REFERENCES
Despesa(CodDespesa)
    );
```

Criando a relação Produto

```
    CREATE TABLE Produto(
    Codigo INT,
    Tamanho VARCHAR(5) NOT NULL,
    Status VARCHAR(12) NOT NULL,
    Nome VARCHAR(20) NOT NULL,
    PrecoVenda REAL NOT NULL,
    CONSTRAINT PKProduto PRIMARY KEY (Codigo),
    CONSTRAINT PrecoPositivo CHECK(PrecoVenda>0)
    );
```

Criando a relação CompraProduto

```
    CREATE TABLE CompraProduto(
    CodProduto INT,
    CodCompra INT,
    quantidade INT NOT NULL,
    precoUnitario REAL NOT NULL,
    CONSTRAINT PKCompraProduto PRIMARY KEY(CodProduto, CodCompra),
```

```
CONSTRAINT FK1CompraProduto FOREIGN KEY(CodProduto) REFERENCES
Produto(Codigo),
CONSTRAINT FK2CompraProduto FOREIGN KEY(CodCompra) REFERENCES
Compra(CodCompra),
CONSTRAINT CompraProdutoPositiva CHECK (precoUnitario>0)
);
```

Criando a relação Entrada

```
CREATE TABLE Entrada(
CodEntrada INT,
Valor REAL NOT NULL,
Data DATE NOT NULL,
CONSTRAINT PKEntrada PRIMARY KEY (CodEntrada),
CONSTRAINT EntradaPositiva CHECK(Valor>0)
);
```

Criando a relação Venda

```
CREATE TABLE Venda(
CodVenda INT,
Hora TIME NOT NULL,
Funcionario VARCHAR(20) NOT NULL,
Cliente VARCHAR(20) NOT NULL,
codEntrada INT NOT NULL,
CONSTRAINT PKVenda PRIMARY KEY(CodVenda),
CONSTRAINT FKVenda1 FOREIGN KEY(Funcionario) REFERENCES Pessoa(CPF),
CONSTRAINT FKVenda2 FOREIGN KEY(Cliente) REFERENCES Pessoa(CPF),
CONSTRAINT FKVenda3 FOREIGN KEY (codEntrada) REFERENCES
Entrada(CodEntrada)
);
```

Criando a relação VendaProduto

```
CREATE TABLE VendaProduto(
CodProduto INT,
CodVenda INT,
Quantidade INT NOT NULL,
precoUnitario REAL NOT NULL,
CONSTRAINT PKVendaProduto PRIMARY KEY(CodProduto, CodVenda),
CONSTRAINT FK1VendaProduto FOREIGN KEY(CodProduto) REFERENCES
Produto(Codigo),
CONSTRAINT FK2VendaProduto FOREIGN KEY(CodVenda) REFERENCES
Venda(CodVenda),
CONSTRAINT VendaProdutoPositiva CHECK (precoUnitario>0)
);
```

Criando a relação CartaoCredito

```
CREATE TABLE CartaoCredito(
Numero VARCHAR(50),
Bandeira VARCHAR(20) NOT NULL,
CONSTRAINT PKCartaoCredito PRIMARY KEY(Numero)
);
```

Criando a relação VendaCartao

```
CREATE TABLE VendaCartao(
NumCartao VARCHAR(50),
CodVenda INT NOT NULL,
NumParcelas INT NOT NULL,
CONSTRAINT PKVendaCartao PRIMARY KEY(NumCartao),
CONSTRAINT FKVendaCartao1 FOREIGN KEY(NumCartao) REFERENCES
CartaoCredito(Numero),
CONSTRAINT FKVendaCartao2 FOREIGN KEY(CodVenda) REFERENCES
Venda(CodVenda)
);
```

Povoamentos

Povoamento da relação Pessoa

```
INSERT INTO Pessoa VALUES('154.288.064-51', 'Teofanes Ferreira', '99123456');
INSERT INTO Pessoa VALUES('358.947.104-21', 'Milena Carvalho', '98752374');
INSERT INTO Pessoa VALUES('354.514.294-99', 'Manoel de Sousa', '93098725');
INSERT INTO Pessoa VALUES('734.042.214-51', 'Amanda Ferreira', '93827460');
INSERT INTO Pessoa VALUES('559.629.914-60', 'Maria das Graças', '99097162');
INSERT INTO Pessoa VALUES('695.997.444-54', 'Antonia Lins', '98152674');
INSERT INTO Pessoa VALUES('690.418.134-62', 'Eduardo Martins', '99523850');
INSERT INTO Pessoa VALUES('846.101.944-06', 'Ana Isabel dos Santos', '99825556');
INSERT INTO Pessoa VALUES('146.287.484-38', 'Junior da Silva', '99123499');
INSERT INTO Pessoa VALUES('819.174.014-18', 'Ricardo Costa', '99553400');
```

Povoamento da relação Funcionário

```
INSERT INTO Funcionario VALUES('846.101.944-06','Caixa','2017-11-10', 954.00,
'Efetivo','JD Oasis','Jose Lira de Menezes','Cajazeiras');
INSERT INTO Funcionario VALUES('146.287.484-38', 'Vendedor', '2017-07-05', 1050.00,
'Efetivo', 'Centro', 'Bonifácio Moura', 'Cajazeiras');
INSERT INTO Funcionario VALUES('690.418.134-62', 'Faxineiro', '2017-12-01', 954.00,
'Efetivo', 'Centro', '13 de Maio', 'Cajazeiras');
INSERT INTO Funcionario VALUES('358.947.104-21', 'Vendedor', '2017-08-11', 1200.50,
'Efetivo', 'Centro', 'Bonifácio Moura', 'Cajazeiras');
INSERT INTO Funcionario VALUES('559.629.914-60', 'Marketing', '2018-08-05', 1500.50,
'Efetivo', 'Centro', 'Bonifácio Moura', 'Cajazeiras');
```

Povoamento da relação Cliente

```
INSERT INTO Cliente VALUES('154.288.064-51', 'teofanesferreira@hotmail.com');
INSERT INTO Cliente VALUES('354.514.294-99', 'manoeldesousa@hotmail.com');
INSERT INTO Cliente VALUES('734.042.214-51', 'amandaferrreira@hotmail.com');
INSERT INTO Cliente VALUES('695.997.444-54', 'antonialins@hotmail.com');
INSERT INTO Cliente VALUES('819.174.014-18', 'ricardo123@hotmail.com');
```

Povoamento da relação Despesa

```
INSERT INTO Despesa VALUES(1, 954.00, '2017-12-15');
INSERT INTO Despesa VALUES(2, 1050.00, '2017-08-15');
INSERT INTO Despesa VALUES(3, 954.00, '2018-01-15');
INSERT INTO Despesa VALUES(4, 1200.50, '2017-09-15');
INSERT INTO Despesa VALUES(5, 1500.50, '2018-08-15');
INSERT INTO Despesa VALUES(6, 2500.00, '2017-11-10');
INSERT INTO Despesa VALUES(7, 1300.00, '2017-12-10');
```

INSERT INTO Despesa VALUES(8, 600.00, '2017-11-09');
INSERT INTO Despesa VALUES(9, 100.00, '2018-01-10');
INSERT INTO Despesa VALUES(10, 700.00, '2018-03-10');

Povoamento da relação Pagamento
INSERT INTO Pagamento VALUES(1, '559.629.014-60', '06:30:35', 1);
INSERT INTO Pagamento VALUES(2, '358.947.104-21', '12:30:00', 2);
INSERT INTO Pagamento VALUES(3, '690.418.134-62', '09:20:05', 3);
INSERT INTO Pagamento VALUES(4, '146.287.484-38', '17:30:00', 4);
INSERT INTO Pagamento VALUES(5, '559.629.014-60', '10:29:00', 5);

Povoamento da relação Fornecedor
INSERT INTO Fornecedor VALUES('85.296.634/0001-96', 'Megaduran', '3561-4578', 'Sousa', 'Rua Vitória Alves da Silva', 'Centro');
INSERT INTO Fornecedor VALUES('64.615.575/0001-00', 'Indivíduo Clothing', '3531-7788', 'Sousa', 'Rua Arquimedes Souto Maior', 'Fagundes Vieira');
INSERT INTO Fornecedor VALUES('96.531.803/0001-41', 'Gata Selvagem', '3531-3456', 'Juazeiro', 'Rua Barbacena', 'Novo Centro');
INSERT INTO Fornecedor VALUES('59.547.777/0001-13', 'Geisandre Lingerie', '3561-0978', 'João Pessoa', 'Rua Doutor João Soares', 'Loteamento Ciro Costa');
INSERT INTO Fornecedor VALUES('74.263.193/0001-00', 'Medson Clothing', '3531-4554', 'João Pessoa', 'Popular Senhor Batista', 'Cidade Nova');
INSERT INTO Fornecedor VALUES('51.795.917/0001-60', 'Havair’, ‘3561-9976', 'Cajazeiras', 'Martins Monteiro', 'Centro');

Povoamento da relação Compra
INSERT INTO Compra VALUES(1, '85.296.634/0001-96', '08:50:00', 6);
INSERT INTO Compra VALUES(2, '64.615.575/0001-00', '09:45:00', 7);
INSERT INTO Compra VALUES(3, '96.531.803/0001-41', '11:20:00', 8);
INSERT INTO Compra VALUES(4, '59.547.777/0001-13', '08:23:00', 9);
INSERT INTO Compra VALUES(5, '74.263.193/0001-00', '10:34:00', 10);

Povoamento da relação Produto
INSERT INTO Produto VALUES(1001234, 'P’, 'Disponível', 'Calça Skinny em Sarja', 59.90);
INSERT INTO Produto VALUES(1001235, 'G', 'Disponível', 'Calça Skinny em Sarja', 45.50);
INSERT INTO Produto VALUES(1001236, 'GG', 'Indisponível', 'Camisa Estampada com Folhagem', 38.90);
INSERT INTO Produto VALUES(1001237, 'G', 'Disponível', 'Camisa Estampada com Amarração', 55.90);
INSERT INTO Produto VALUES(1001238, 'P', 'Disponível', 'Calça Mom Jeans', 75.90);
INSERT INTO Produto VALUES(1001239, 'Único', 'Disponível', 'Jaqueta de Frio Xadrez', 120.00);
INSERT INTO Produto VALUES(1001240, 'P', 'Indisponível', 'Cropped Azul Marinho', 35.50);
INSERT INTO Produto VALUES(1001241, 'PP', 'Disponível', 'Vestido Curto Florido', 65.00);
INSERT INTO Produto VALUES(1001242, 'GG', 'Disponível', 'Bermuda Jeans', 70.50);
INSERT INTO Produto VALUES(1001243, 'M', 'Disponível', 'Camisa Xadrez', 69.90);
INSERT INTO Produto VALUES(1001244, 'G', 'Indisponível', 'Camisa The Smiths', 45.50);
INSERT INTO Produto VALUES(1001245, 'P', 'Disponível', 'Saia Longa Branca', 60.80);

```
INSERT INTO Produto VALUES(1001246, 'PP', 'Disponível', 'Camisa Gola V', 60.00);
INSERT INTO Produto VALUES(1001247, 'M', 'Disponível', 'Bomber Jacket Nasa',
200.00);
INSERT INTO Produto VALUES(1001248, 'GG', 'Disponível', 'Saia Colegial', 60.00);
INSERT INTO Produto VALUES(1001249, 'Único', 'Disponível', 'Bikini Listrado', 59.90);
INSERT INTO Produto VALUES(1001250, 'M', 'Disponível', 'Blusão Estampado', 80.00);
INSERT INTO Produto VALUES(1001251, 'PP', 'Disponível', 'Camisa Social Preta',
100.00);
INSERT INTO Produto VALUES(1001252, 'M', 'Disponível', 'Vestido Longo Preto', 200.00);
INSERT INTO Produto VALUES(1001253, 'G', 'Disponível', 'Short Cintura Alta', 70.60);
```

Povoamento da relação CompraProduto

```
INSERT INTO CompraProduto VALUES(1001253, 1, 45, 55.00);
INSERT INTO CompraProduto VALUES(1001252, 2, 20, 170.00);
INSERT INTO CompraProduto VALUES(1001251, 3, 30, 80.00);
INSERT INTO CompraProduto VALUES(1001250, 4, 12, 60.00);
INSERT INTO CompraProduto VALUES(1001249, 5, 40, 40.00);
```

Povoamento da relação Entrada

```
INSERT INTO Entrada VALUES(12, 270.60, '2018-08-11');
INSERT INTO Entrada VALUES(13, 180.00, '2018-08-13');
INSERT INTO Entrada VALUES(14, 59.90, '2018-08-15');
INSERT INTO Entrada VALUES(15, 260.00, '2018-08-20');
INSERT INTO Entrada VALUES(16, 60.00, '2018-09-01');
INSERT INTO Entrada VALUES(17, 60.80, '2018-09-02');
```

Povoamento da relação Venda

```
INSERT INTO Venda VALUES(33, '11:09:00', '146.287.484-38', '154.288.064-51', 12);
INSERT INTO Venda VALUES(34, '09:34:55', '146.287.484-38', '354.514.294-99', 13);
INSERT INTO Venda VALUES(35, '08:23:40', '146.287.484-38', '734.042.214-51', 14);
INSERT INTO Venda VALUES(36, '15:02:22', '146.287.484-38', '695.997.444-54', 15);
INSERT INTO Venda VALUES(37, '17:20:00', '358.947.104-21', '695.997.444-54', 16);
INSERT INTO Venda VALUES(38, '08:10:12', '358.947.104-21', '819.174.014-18', 17);
```

Povoamento da relação VendaProduto

```
INSERT INTO VendaProduto VALUES(1001253, 33, 1, 70.60);
INSERT INTO VendaProduto VALUES(1001252, 33, 1, 200.00);
INSERT INTO VendaProduto VALUES(1001253, 34, 1, 100.00);
INSERT INTO VendaProduto VALUES(1001250, 34, 1, 80.00);
INSERT INTO VendaProduto VALUES(1001249, 33, 1, 59.90);
```

Povoamento da relação CartaoCredito

```
INSERT INTO CartaoCredito VALUES(‘5530330246021173’,'Master Card');
INSERT INTO CartaoCredito VALUES(‘4312092068980640’, 'Visa');
INSERT INTO CartaoCredito VALUES(‘6363686632535781’, 'Elo');
INSERT INTO CartaoCredito VALUES(‘5518757232660489’, 'Master Card');
INSERT INTO CartaoCredito VALUES(‘4532399086941049’, 'Visa');
```

Povoamento da relação VendaCartao

```
INSERT INTO VendaCartao VALUES(‘5530330246021173’, 33, 3);
INSERT INTO VendaCartao VALUES(‘4312092068980640’, 34, 2);
INSERT INTO VendaCartao VALUES(‘6363686632535781’, 35, 2);
```

```
INSERT INTO VendaCartao VALUES('5518757232660489', 36, 2);
INSERT INTO VendaCartao VALUES('4532399086941049', 37, 2);
```

Criação de Índices

```
CREATE INDEX cpfIndice ON Pessoa(CPF);
CREATE INDEX cpfFuncionarioIndice ON Funcionario(CPFPessoa);
CREATE INDEX cpfClienteIndice ON Cliente(CPFPessoa);
CREATE INDEX codSalarioIndice ON Pagamento(Codpagamento);
CREATE INDEX cnpjFornecedorIndice ON Fornecedor(CNPJ);
CREATE INDEX codigoDespesaIndice ON Despesa(CodDespesa);
CREATE INDEX codigoCompraIndice ON Compra(CodCompra);
CREATE INDEX codigoProdutoIndice ON Produto(Codigo);
CREATE INDEX codigoVendaIndice ON Venda(CodVenda);
CREATE INDEX codigoEntradaIndice ON Entrada(CodEntrada);
CREATE INDEX numeroCartaoIndice ON CartaoCredito(Numero);
```

Criando Visões

1. Essa visão auxilia na recuperação do produto mais comprado a fornecedores, já que haverá um fácil acesso aos dados principais.

```
CREATE VIEW pedidosVisao
AS(
   SELECT P.Nome, COUNT(*) AS Pedidos
   FROM CompraProduto CP, Produto P
   WHERE CP.CodProduto = P.Codigo
   GROUP BY CP.CodProduto, P.Nome
);
```

2. Essa visão auxilia na recuperação do tamanho de roupa mais vendido a clientes. Importante para saber quais tamanhos comprar em maior quantidade, visando lucros.

```
CREATE VIEW tamanhosVisao
AS(
   SELECT P.Tamanho, COUNT(*) AS Vendidos
   FROM VendaProduto VP, Produto P
   WHERE VP.CodProduto = P.Codigo
   GROUP BY P.Tamanho
);
```

3. Essa visão auxilia na recuperação do fornecedor que o estabelecimento compra mais vezes. Importante para ter noção de "fornecedor fiel".

```
CREATE VIEW fornecedoresVisao
AS(
   SELECT F.CNPJ, F.NOME, COUNT(*) AS Pedidos
   FROM Fornecedor F, Compra C
   WHERE F.CNPJ = C.Fornecedor
   GROUP BY F.CNPJ, F.Nome
);
```

4. Essa visão auxilia na recuperação do vendedor que atende em mais vendas. Importante caso a dona do estabelecimento queira dar aumento a algum funcionário, ou apenas manter controle sobre quais funcionários geram mais lucro para a loja.

```
CREATE VIEW vendedorVisao
AS(
   SELECT F.CPFPessoa, P.Nome, COUNT(*) AS Vendas
   FROM Funcionario F, Venda V, Pessoa P
   WHERE F.CPFPessoa = V.Funcionario AND F.Cargo = 'Vendedor' AND F.CPFPessoa =
P.CPF
   GROUP BY F.CPFPessoa, P.Nome
);
```

5. Essa visão auxilia na recuperação de vendas por funcionário. Importante para ter um acesso fácil aos dados de uma venda.

```
CREATE VIEW vendasVisao
AS(
   SELECT V.Funcionario, VP.CodProduto, VP.Quantidade, E.Valor
   FROM (Venda V NATURAL JOIN VendaProduto VP) NATURAL JOIN Entrada E
 );
```

Criando gatilhos

1. Esse gatilho garante que os salários dos funcionários não podem ser atualizados para um valor menor que o atual:

```
CREATE FUNCTION VerificaSalario () RETURNS TRIGGER
     AS $$
          BEGIN
               IF(NEW.salario < OLD.salario) THEN RETURN NULL;
                    ELSE RETURN NEW;
               END IF;
          END
     $$ LANGUAGE PLPGSQL;

CREATE TRIGGER VerificaSalarioTrigger BEFORE UPDATE ON Funcionario
     FOR EACH ROW
     EXECUTE PROCEDURE VerificaSalario();
```

2. Esse gatilho garante que nenhum funcionário ainda não efetivo tenha seus dados atualizados. Importante para que não haja um erro nos dados, já que funcionários demitidos só poderão ser atualizados caso sejam recontratados.

```
CREATE FUNCTION VerificaFuncionarioNaoEfetivo () RETURNS TRIGGER
     AS $$
          BEGIN
               IF((NEW.status = 'Não efetivo')AND(OLD.status = 'Não efetivo')) THEN
RETURN NULL;
```

```
            ELSE RETURN NEW;
        END IF;
    END
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER VerificaFuncionarioNaoEfetivo BEFORE UPDATE ON Funcionario
    FOR EACH ROW
    EXECUTE PROCEDURE VerificaFuncionarioNaoEfetivo();
```

Criando procedimentos

1. Esse procedimento armazenado classifica o salário de cada funcionário, importante para o controle de salários entre funcionários.

```
CREATE OR REPLACE FUNCTION ClassificaSalario (VARCHAR(20)) RETURNS
VARCHAR
    AS $$
        DECLARE
            SalFun funcionario.SALARIO%TYPE;
            CPFFun ALIAS FOR $1;
    BEGIN
            SELECT INTO SalFun SALARIO FROM Funcionario WHERE CPFFunc =
CPFPessoa;
            IF SalFun < 1000 THEN RETURN 'Salário baixo';
                ELSE IF SalFunc<2000 THEN RETURN 'Salário Médio';
                ELSE RETURN 'Salário Alto';
                END IF;
        END IF;
END
$$ LANGUAGE PLPGSQL;
```

2. Esse procedimento armazenado retorna os dias entre duas datas, importante para o estabelecimento ter controle entre datas.

```
CREATE OR REPLACE FUNCTION DiasEntre(DATE, DATE) RETURNS INTEGER
    AS $$
        DECLARE
            MaiorData DATE;
            MenorData DATE;
            quantDias INTEGER;
        BEGIN
            IF ($1<$2) THEN
                MenorData := $1;
                MaiorData := $2;
            ELSE
                MenorData := $2;
                MaiorData := $1;
            END IF;
            quantDias := MaiorData - MenorData;
            RETURN quantDias;
        END;
```

```
$$ LANGUAGE PLPGSQL;
```

Criando Consultas

1. Recuperar e-mail dos clientes atendidos por Milena.

```
SELECT C.email
FROM Cliente C JOIN Venda V ON C.CPFPessoa = V.Cliente
WHERE  V.Funcionario = '358.947.104-21'
```

2. Recuperar nome dos clientes que usaram cartão da Visa.

```
SELECT P.Nome
FROM CartaoCredito CC, VendaCartao VC, Venda V, Cliente C, Pessoa P
WHERE CC.Numero = VC.NumCartao AND V.CodVenda = VC.CodVenda AND
V.Cliente     = C.CPFPessoa AND C.CPFPessoa = P.CPF
```

3. Recuperar telefone de todos os fornecedores que venderam acima de 20 produtos para a loja.

```
SELECT F.CNPJ, F.Telefone
FROM (Compra C NATURAL JOIN CompraProduto CP) JOIN Fornecedor F ON
C.Fornecedor = F.CNPJ
GROUP BY F.CNPJ, F.Telefone, CP.Quantidade
HAVING CP.Quantidade > 20
```

4. Recuperar o nome de clientes que pagaram com cartão em pelo menos duas parcelas.

```
select p.nome
from pessoa p
where p.cpf in
     (select c.cpfpessoa
     from (VendaCartao VC NATURAL JOIN Venda V) JOIN Cliente C ON V.Cliente =
          C.CPFPessoa
     WHERE VC.NumParcelas >= 2)
```

5. Recuperar nome de todos os clientes já forneceram seu e-mail

```
SELECT P.Nome AS NomeCliente
FROM Cliente C JOIN Pessoa P ON C.CPFPessoa = P.CPF
WHERE C.email IS NOT NULL
```

6. Recuperar o cpf de funcionários que nunca atenderam nenhuma venda

```
SELECT F.CPFPessoa
FROM Funcionario F LEFT JOIN Venda V ON F.CPFPessoa = V.Funcionario
WHERE V.CodVenda IS NULL
```

7. Recuperar nome de todos os funcionários que começam com a letra M.

```
SELECT P.Nome
FROM Funcionario F JOIN Pessoa P ON F.CPFPessoa = P.CPF
WHERE P.Nome LIKE 'M%'
```

8. Recuperar email de clientes que usam o domínio @hotmail.com

```
SELECT email
FROM cliente
WHERE email LIKE '%@hotmail.com'
```

9. Recuperar nome e endereço de funcionários que moram na cidade de Cajazeiras.

```
SELECT P.Nome, F.Rua, F.Bairro
FROM Funcionario F JOIN Pessoa P ON F.CPFPessoa = P.CPF
WHERE F.Cidade = 'Cajazeiras'
ORDER BY P.Nome
```

10. Verificar quais produtos estão disponíveis na loja.

```
SELECT codigo, Nome
FROM Produto
WHERE Status = 'Disponível'
ORDER BY Nome
```

11. Recuperar valor de cada produto disponível se a loja toda declarar desconto de 20%.

```
SELECT Nome, PrecoVenda * 0.80 AS PreçoDesconto
FROM Produto
ORDER BY Nome
```

12. Recuperar e-mail e nome de todos os clientes que fizeram mais de uma compra.

```
SELECT C.email, P.Nome
FROM Cliente C JOIN Pessoa p ON P.CPF=C.CPFpessoa
WHERE C.CPFpessoa IN
(SELECT V.Cliente
FROM Venda V, Cliente C
WHERE V.Cliente = C.CPFPessoa
GROUP BY V.Cliente
HAVING Count(*)>1)
```

13. Recuperar nome e código de produtos que tiveram pelo menos 2 vendas contabilizadas.

```
SELECT P.Nome, P.Codigo
FROM Produto P
WHERE P.Codigo  IN
(SELECT VP.CodProduto
FROM VendaProduto VP
```

```
GROUP BY VP.CodProduto
HAVING Count(*)>=2 AND VP.CodProduto = P.Codigo)
```

14. Recuperar nome de funcionários que ainda não receberam pagamento.

```
SELECT P.Nome
FROM Pessoa P JOIN Funcionario F ON P.CPF = F.CPFPessoa
WHERE NOT EXISTS
(SELECT * FROM Pagamento PA
 WHERE P.CPF = PA.Funcionario)
```

15. Recuperar email de clientes atendidos por funcionários efetivos.

```
SELECT c.email
FROM Cliente C
WHERE EXISTS
(SELECT *
 FROM Venda V JOIN Funcionario F ON V.Funcionario = F.CPFPessoa
 WHERE F.Status = 'Efetivo')
```

16. Recuperar produtos disponíveis tanto em tamanho P quanto em tamanho G.

```
(SELECT Nome
FROM Produto
WHERE Status = 'Disponível' AND Tamanho = 'P')
INTERSECT
(SELECT Nome
FROM Produto
WHERE Status = 'Disponível' AND Tamanho = 'G')
```

17. Recuperar código de despesas que não provém de compras.

```
(SELECT CodDespesa
FROM Despesa)
EXCEPT
(SELECT CodDespesa
FROM Despesa NATURAL JOIN Compra)
```

18. Verificar o lucro de vendas entre 2018-08-11 até 2018-08-20.

```
SELECT SUM(Valor) as Lucro
FROM  Entrada
WHERE Data >= '2018-08-11' AND Data <= '2018-08-20'
```

19. Verificar o prejuízo com despesas entre 2018-07-30 e 2018-08-30.

```
SELECT SUM(Valor) as Prejuíjo
FROM Despesa
WHERE Data >= '2018-07-30' AND Data <= '2018-08-30'
```

20. Verificar quantos produtos estão disponíveis na loja.

```
SELECT COUNT(*) AS Disponíveis
FROM Produto
GROUP BY Status
HAVING Status = 'Disponível'
```

21. Verificar quantas vendas cada funcionário realizou.

```
SELECT V.Funcionario, P.Nome, COUNT(*) AS Vendas
FROM (Venda V JOIN Funcionario F ON V.Funcionario = F.CPFPessoa) JOIN Pessoa
P ON    P.CPF = F.CPFPessoa
GROUP BY V.Funcionario, P.Nome
```