Lista 02 – Sistema de Banco de Dados

Considere o diagrama relacional da Figura 1 e escreva os comandos SQL para executar os exercícios abaixo.

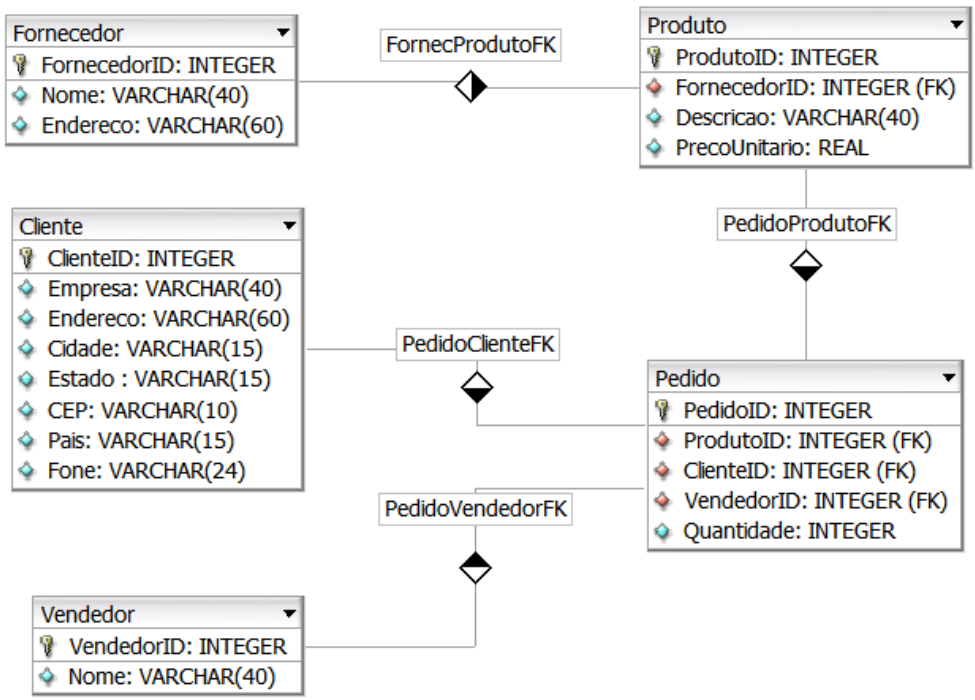

Figure 1 – Diagrama relacional.

Exercícios:

01) Crie as tabelas Fornecedor e Produto.

02) Use os comandos do Anexo I deste arquivo para criar as tabelas Cliente, Vendedor e Pedido.

03) Insira os registros nas tabelas criadas, usando os comandos "INSERT INTO" do Anexo I deste arquivo. O SGBD retornará um erro ao inserir os registros da tabela Cliente. Qual o comando SQL é utilizado para solucionar esse erro e inserir todos os registros da tabela Cliente?

04) Recupere as informações do catálogo:

   a. Quais esquemas existem nesse banco de dados?
   b. Recupere as informações sobre as tabelas do esquema "public".
   c. Recupere as informações sobre todas as colunas de todas as tabelas do esquema "public".

d. Recupere as informações sobre todas as restrições (*constraints*) de todas as tabelas do esquema “public”.

05) Selecione todos os vendedores ordenados pelo nome.
06) Quantos vendedores estão cadastrados no banco de dados?
07) Quantos pedidos foram feitos pelo vendedor 'Jose Marcio'?
08) Quantos itens foram vendidos pelo vendedor 'Jose Marcio'?
09) Quantos itens foram vendidos por cada vendedor?
10) Quais vendedores venderam mais que 300 itens?
11) Quanto foi o valor de cada pedido?
12) Quanto cada vendedor vendeu?
13) Quanto foi a comissão de cada vendedor (considerando que a comissão é 10% do valor da venda)
14) Crie uma view que contenha as seguintes informações sobre os pedidos: identificador do pedido, nome do vendedor, nome do cliente, endereco do cliente, telefone do cliente, descrição do produto, fornecedor do produto e a quantidade do produto.
15) Insira um novo pedido => vendedor: "Luis Claudio", cliente: "ACM", produto: "Tinta", quantidade: 112
16) Altere o nome do vendedor "Andre Carlos" para "Andre Carlos Garcia"
17) Reajuste o preço de todos os produtos em 10%
18) Remova o cliente "ACM" da tabela cliente

------ OBS: Observe o que acontece com os pedidos associados ao cliente "ACM"

19) Adicione um atributo "CNPJ" na tabela "Cliente" (os valores dos CNPJ devem ser únicos)
20) Mude o atributo "Fone" da tabela "Cliente" para um tipo numérico
21) Renomeie o atributo "CNPJ" para "CNPJ_Cliente"
22) Remova o atributo "CNPJ_Cliente"
23) Remova todos os registros da tabela "Vendedor"

------ OBS: Observe o que acontece com os registros das tabelas que recebem o atributo "VendedorID" como foreign key

24) Remova o atributo "VendedorID" da tabela "Vendedor"
25) Remova a tabela "Vendedor"

26) Remova todas as tabelas do banco (o esquema e o conteúdo)
27) Crie novamente as tabelas do banco de dados, usando os comandos CREATE TABLE já utilizados.
28) Crie uma nova tabela chamada "vendedor_comissao" para armazenar a comissão de cada vendedor. Essa tabela terá o id de cada vendedor, que será PRIMARY KEY e FOREIGN KEY, e o valor da comissão.
29) Use o recurso de trigger para popular automaticamente a tabela "vendedor_comissao", conforme vendedores e pedidos são inseridos no banco de dados. Crie duas triggers:
    a. Cada vez que um novo vendedor é inserido no banco de dados (tabela "vendedor"), a trigger deve inserir um novo registro na tabela "vendedor_comissao" contendo o id desse novo vendedor e o valor 0.0 para sua comissão. Após criar essa trigger, execute os comandos INSERT TABLE da tabela "vendedor" e veja o que acontece:

```
CREATE FUNCTION insert_vendedor_comissao() RETURNS trigger
AS $vend_comissao$
    BEGIN
         INSERT INTO vendedor_comissao(vendedorid, comissao)
         VALUES (NEW.vendedorid, 0.0);
         RETURN NEW;
    END;
$vend_comissao$ LANGUAGE plpgsql;

CREATE TRIGGER vendedor_comissao_trigger_1
AFTER INSERT ON vendedor
FOR EACH ROW
EXECUTE PROCEDURE insert_vendedor_comissao();
```

    b. Cada vez que um novo pedido é inserido ou atualizado no banco de dados (tabela "pedido"), a trigger deve atualizar a comissão na tabela "vendedor_comissao" do vendedor que efetuou o pedido. Após criar essa trigger, execute os comandos INSERT TABLE da tabela "pedido" e veja o que acontece.

30) Insira uma nova coluna na tabela vendedor com o salário de cada uma:
    a. Jose Marcio: R$ 3500.00
    b. Luis Claudio: R$ 3000.00
    c. Andre Carlos: R$ 2500.00
31) Altere a trigger acima para que nenhum vendedor ganhe uma comissão com o valor acima do seu salário. Ou seja, toda vez que o vendedor efetuar um pedido, a trigger deve checar se o novo valor da comissão é maior do que o salario do vendedor. Se for maior, o vendedor deve receber exatamente o valor do seu salário de comissão.

Anexo I – SQL

```
------------------------------- Create Table
CREATE TABLE Cliente (
ClienteID       INT             NOT NULL,
Empresa         VARCHAR( 40 )   NOT NULL,
Endereco        VARCHAR( 60 ),
Cidade          VARCHAR( 15 ),
Estado          VARCHAR( 15 ),
CEP             VARCHAR( 10 ),
Pais            VARCHAR( 15 ),
Fone            VARCHAR( 24 ),
CONSTRAINT ClientePK PRIMARY KEY (ClienteID)
);

CREATE TABLE Vendedor (
VendedorID  INT            NOT NULL,
Nome        VARCHAR( 40 )  NOT NULL,

CONSTRAINT VendedorPK PRIMARY KEY ( VendedorID )
);

CREATE TABLE Pedido (
PedidoID        INT  NOT NULL,
VendedorID      INT  NOT NULL,
ClienteID       INT  NOT NULL,
ProdutoID       INT  NOT NULL,
Quantidade      INT  NOT NULL   DEFAULT 1 CHECK(Quantidade>0),

CONSTRAINT PedidoPK PRIMARY KEY ( PedidoID ),

CONSTRAINT PedidoVendedorFK FOREIGN KEY (VendedorID)
REFERENCES Vendedor(VendedorID) ON DELETE CASCADE ON UPDATE
CASCADE,

CONSTRAINT PedidoClienteFK FOREIGN KEY (ClienteID) REFERENCES
Cliente(ClienteID) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT PedidoProdutoFK FOREIGN KEY (ProdutoID) REFERENCES
Produto(ProdutoID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

------------------------------ Insert Table

```
INSERT INTO Cliente Values ( 1, 'ACM', 'Rua das Flores, 10',
'Sao Paulo', 'SP', '1222000', 'Brasil', '112233445566');
INSERT INTO Cliente Values ( 2, 'VW', 'Rua do Comercio, 47',
'Sao Paulo', 'SP', '1222010', 'Brasil', '11298735566');
INSERT INTO Cliente Values ( 3, 'GM', 'Via Dutra, 1000', 'Sao
Jose dos Campos', 'SP', '1222560', 'Brasil', '122239876566');
INSERT INTO Cliente Values ( 4, 'TEX', 'AV Brasil, 1210', 'Rio
de Janeiro', 'RJ', '348890', 'Brasil', '212134567');

INSERT INTO Vendedor Values ( 1, 'Jose Marcio');
INSERT INTO Vendedor Values ( 2, 'Luis Claudio');
INSERT INTO Vendedor Values ( 3, 'Andre Carlos');

INSERT INTO Fornecedor Values ( 1, 'Ferragens Santa Lucia',
'Rua Catalao, 20, Goiania, GO');
INSERT INTO Fornecedor Values ( 2, 'Borracharia Campos', 'Rua
dos Ipes 1235, Presidente Prudente, SP');
INSERT INTO Fornecedor Values ( 3, 'Tintas Brasil',    'Avenida
dos Guararapes 44, Paulinia, SP');

INSERT INTO Produto Values ( 1, 2, 'Roda', 500.00);
INSERT INTO Produto Values ( 2, 1, 'Mola', 234.00);
INSERT INTO Produto Values ( 3, 1, 'Porca', 11.00);
INSERT INTO Produto Values ( 4, 1, 'Parafuso', 5.30);
INSERT INTO Produto Values ( 5, 2, 'Prego', 1.20);
INSERT INTO Produto Values ( 6, 3, 'Tinta', 234.00);

INSERT INTO Pedido Values ( 1, 2, 4, 2, 450);
INSERT INTO Pedido Values ( 2, 1, 2, 1, 123);
INSERT INTO Pedido Values ( 3, 2, 1, 2, 60);
```

```
INSERT INTO Pedido Values ( 4, 3, 2, 2, 121);
INSERT INTO Pedido Values ( 5, 3, 3, 6, 65);
INSERT INTO Pedido Values ( 6, 1, 3, 5, 36);
INSERT INTO Pedido Values ( 7, 2, 1, 5, 140);
INSERT INTO Pedido Values ( 8, 3, 4, 1, 200);
INSERT INTO Pedido Values ( 9, 3, 2, 3, 67);
INSERT INTO Pedido Values ( 10, 1, 2, 3, 89);
```

------------------------------ Trigger

```
CREATE OR REPLACE FUNCTION update_vendedor_comissao()
RETURNS trigger AS $vend_comissao$
DECLARE
     v_preco real := 0.;
     BEGIN
          SELECT precounitario INTO v_preco FROM produto
          WHERE produtoid = NEW.produtoid;

          UPDATE vendedor_comissao
          SET comissao = comissao + (NEW.quantidade * v_preco
          * 0.1) WHERE vendedorid = NEW.vendedorid;

          RETURN NEW;
     END;
$vend_comissao$ LANGUAGE plpgsql;

CREATE TRIGGER vendedor_comissao_trigger_2
AFTER INSERT OR UPDATE OF quantidade, vendedorid ON pedido
FOR EACH ROW
EXECUTE PROCEDURE update_vendedor_comissao();
```