# Descrição do Sistema Gerenciador de Banco de Dados

# MigSports
## Artigos Esportivos

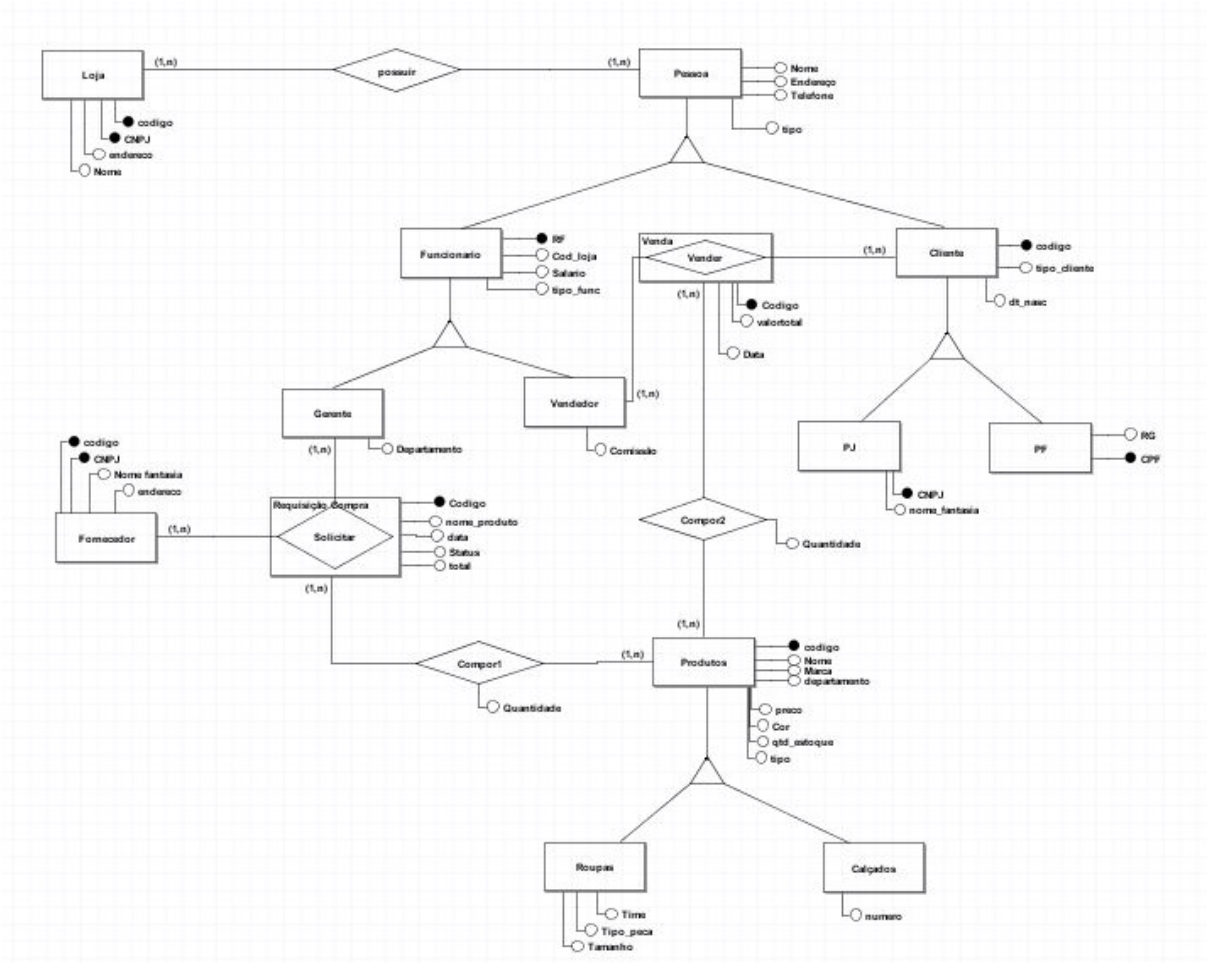
**Filipe Bernardo - 197054**
**Henrique Campiotti - 198760**
**Leonardo Furone - 201192**
**Lucas Watanuki - 202143**
**Rodrigo Zanco - 205541**

**Introdução**

Projeto de elaboração de um SGBD relacionado a loja de artigos esportivos. Nele, estão contidos a parte conceitual (ME-R), a parte lógica (Modelo Relacional), o projeto físico e os scripts de Procedimentos Armazenados e Gatilhos.

**Projeto Conceitual da Base de Dados:**

Para a modelagem do projeto, primeiramente pensamos em que produtos são mais específicos para uma loja de artigos esportivos, e também o que deve ser considerado desde a modelagem para destacar a derivação de um modelo genérico de loja para um modelo mais elaborado. Para especificar o modelo loja, precisamos detalhar o conjunto de vendas, fornecedores, pessoas, e até o tipo do produto vendido.



**Projeto Lógico da Base de Dados:**

No mapeamento, foi usado o modelo relacional para direcionar as restrições de integridade, e assim organizar as entidades para a transição e criação do banco. Organizamos o mapeamento de acordo com as especificações de uma loja, o que resultou em mapeamentos de controle de venda.

**MAPEAMENTO:**

Loja = {codigo, cnpj, endereco, nome}

Pessoa = {codigo, cod_loja, nome, endereco, telefone, tipo}
cod_loja - chave estrangeira referenciando Loja
tipo: 0 - funcionario 1 - cliente

Funcionario = {RF, cod_pessoa, cod_loja, salario, tipo_func}
cod_pessoa - chave estrangeira referenciando Pessoa
cod_loja - chave estrangeira ref. Loja

Gerente = {RF, departamento}
RF – chave estrangeira referenciando Funcionario

Vendedor = {RF, comissao}
RF – chave estrangeira referenciando Funcionario

Cliente = {codigo, cod_pessoa, dt_nasc, tipo_cliente}
cod_pessoa – chave estrangeira referenciando Pessoa
tipo_cilente: 0 - pessoa_juridica 1- pessoa_fisica

pessoa_juridica = {cod_cliente, cnpj, nome_fantasia}
cod_cliente – chave estrangeira referenciando Cliente

pessoa_fisica = {cod_cliente, RG, cpf}
cod_cliente - chave estrangeira referenciando Cliente

Fornecedor = {codigo, cnpj, nome_fantasia, endereco}

Produtos = {codigo, nome, marca, departamento, preco, cor, qtd_estoque, tipo}
tipo: 0 - Roupa 1 - Calçado

Roupa = {cod_produto, tipo_peca, tamanho, time}
cod_produto - chave estrangeira referenciando Produtos

calcado = {cod_produto, numero}
cod_produto - chave estrangeira referenciando Produtos

RequisicaoCompra = {codigo, cod_fornecedor, cod_produto, RF, data, status, total}
cod_fornecedor – chave estrangeira referenciando Fornecedor
RF – chave estrangeira referenciando Gerente

Venda = {codigo, cod_cliente, RF, data, total}
RF – chave estrangeira referenciando Vendedor
cod_cliente – chave estrangeira referenciando Cliente

item_requisicao = {cod_req, cod_produto, quantidade}
cod_req – chave estrangeira referenciando RequisicaoCompra
cod_produto – chave estrangeira referenciando Produtos

item_venda = {cod_venda, cod_produto, quantidade}
cod_venda – chave estrangeira referenciando Venda
cod_produto – chave estrangeira referenciando Produtos

**Projeto Físico da Base de Dados:**

No projeto físico, implantamos todas as tabelas e restrições oriundas do mapeamento, além dos índices. Está contido também os procedimentos armazenados de cadastro de cada entidade, além de gatilhos para a atualização do estoque através do cadastro de venda e exclusão de itens da venda. Além disso, há procedimentos armazenados para cadastrar itens específicos da loja, como por exemplo, roupas.

**SCRIPT DE CRIACAO DE TABELAS**

```
--Para resetar códigos de identificação:
DBCC CHECKIDENT('nome tabela', RESEED, 0)

CREATE TABLE loja
(
   codigo int not null IDENTITY(1,1),
   cnpj VARCHAR(18) not NULL,
   endereco VARCHAR(100),
   numero int not NULL,
   cidade VARCHAR(20) NOT NULL,
   UF CHAR(2) NOT NULL,
   nome VARCHAR(50),
   PRIMARY KEY(codigo),
   UNIQUE(cnpj)
)

create TABLE pessoa
(
   codigo int not null IDENTITY(1,1),
   cod_loja int NOT NULL,
```

```
   nome VARCHAR(50) NOT NULL,
   endereco VARCHAR(100),
   numero int not NULL,
   cidade VARCHAR(20) NOT NULL,
   UF CHAR(2) NOT NULL,
   telefone VARCHAR(14) NOT NULL,
   PRIMARY KEY(codigo),
   FOREIGN KEY(cod_loja) REFERENCES loja
)

CREATE TABLE funcionario
(
   RF int NOT NULL,
   cod_pessoa INT NOT NULL,
   cod_loja INT NOT NULL,
   salario money not NULL,
   tipo_func SMALLINT NOT NULL,
   PRIMARY KEY(RF),
   UNIQUE(cod_pessoa),
   FOREIGN KEY(cod_pessoa) REFERENCES pessoa,
   FOREIGN KEY(cod_loja) REFERENCES loja
)

CREATE TABLE gerente
(
   RF int not NULL,
   departamento VARCHAR(25) not NULL,
   PRIMARY KEY(RF),
   FOREIGN KEY(RF) REFERENCES funcionario
)

CREATE TABLE vendedor
(
   RF int not NULL,
   comissao money,
   PRIMARY KEY(RF),
   FOREIGN KEY(RF) REFERENCES funcionario
)
```

```
CREATE TABLE cliente
(
   codigo int not null IDENTITY(1,1),
   cod_pessoa INT NOT NULL,
   dt_nasc DATE not NULL,
   tipo_cliente SMALLINT NOT NULL,
   PRIMARY KEY(codigo),
   FOREIGN KEY(cod_pessoa) REFERENCES pessoa
)

CREATE TABLE pessoa_juridica
(
   cod_cliente int not NULL,
   cnpj VARCHAR(18) not NULL,
   nome_fantasia VARCHAR(50) NOT NULL,
   PRIMARY KEY(cod_cliente),
   FOREIGN KEY(cod_cliente) REFERENCES cliente,
   UNIQUE(cnpj)
)

CREATE TABLE pessoa_fisica
(
   cod_cliente int not NULL,
   cpf VARCHAR(14) NOT NULL,
   rg VARCHAR(13),
   PRIMARY KEY(cod_cliente),
   FOREIGN KEY(cod_cliente) REFERENCES cliente,
   UNIQUE(cpf)
)

CREATE TABLE fornecedor
(
   codigo int not null IDENTITY(1,1),
   cnpj VARCHAR(18) not NULL,
   nome_fantasia VARCHAR(50) not NULL,
   endereco VARCHAR(100),
   numero int not NULL,
   cidade VARCHAR(20) NOT NULL,
   UF CHAR(2) NOT NULL,
   PRIMARY KEY(codigo),
```

```
   UNIQUE(cnpj)
)

CREATE TABLE produtos
(
   codigo int not null IDENTITY(1,1),
   nome VARCHAR(50) not NULL,
   marca VARCHAR(15),
   departamento VARCHAR(15) not NULL,
   preco money not NULL,
   cor VARCHAR(15),
   qtd_estoque INT NOT NULL,
   tipo SMALLINT NOT NULL,
   PRIMARY KEY(codigo),
)

CREATE TABLE roupa
(
   cod_produto INT NOT NULL,
   tipo_peca VARCHAR(15) NOT NULL,
   tamanho char(3) NOT NULL,
   _time VARCHAR(15),
   FOREIGN KEY(cod_produto) REFERENCES produtos
)

CREATE TABLE calcado
(
   cod_produto INT NOT NULL,
   numero INT NOT NULL,
   FOREIGN KEY(cod_produto) REFERENCES produtos
)

drop TABLE requisicao_compra
(
   codigo int not null IDENTITY(1,1),
   cod_fornecedor INT NOT NULL,
   cod_produto INT NOT NULL,
   RF INT not NULL,
   _data DATE not NULL,
   total money NOT NULL,
```

```
   PRIMARY KEY(codigo),
   FOREIGN KEY(cod_fornecedor) REFERENCES fornecedor,
   FOREIGN KEY(cod_produto) REFERENCES produtos,
   FOREIGN KEY(RF) REFERENCES gerente,
)

CREATE TABLE venda
(
   codigo int not null IDENTITY(1,1),
   cod_cliente int not NULL,
   RF int not NULL,
   _data DATE not NULL,
   total money NOT NULL,
   PRIMARY KEY(codigo),
   FOREIGN KEY(cod_cliente) REFERENCES cliente,
   FOREIGN KEY(RF) REFERENCES vendedor
)

Create TABLE item_requisicao
(
   cod_req INT NOT NULL,
   cod_produto INT NOT NULL,
   quantidade INT NOT NULL,
   FOREIGN KEY(cod_req) REFERENCES requisicao_compra,
   FOREIGN KEY(cod_produto) REFERENCES produtos
)

CREATE TABLE item_venda
(
   cod_produto INT NOT NULL,
   cod_venda INT NOT NULL,
   quantidade INT NOT NULL,
   FOREIGN KEY(cod_produto) REFERENCES produtos,
   FOREIGN KEY(cod_venda) REFERENCES venda
)

/* ÍNDICES PARA CHAVES ESTRANGEIRAS*/

CREATE INDEX indice_loja
on pessoa(cod_loja)
```

```
CREATE INDEX indice_funcionario
on funcionario(cod_pessoa, cod_loja)

CREATE INDEX indice_pessoa_cliente
on cliente(cod_pessoa)

CREATE INDEX indice_cod_roupa
ON roupa(cod_produto)

CREATE INDEX indice_cod_calcado
ON calcado(cod_produto)

CREATE INDEX indice_gerente
ON requisicao_compra(RF)

CREATE INDEX indice_produto
ON requisicao_compra(cod_produto)

CREATE INDEX indice_fornecedor
ON requisicao_compra(cod_fornecedor)

CREATE INDEX indice_cliente_venda
ON venda(cod_cliente)

CREATE INDEX indice_vendedor
ON venda(RF)

Create INDEX item_RC
ON item_requisicao(cod_req, cod_produto)

CREATE INDEX indice_itemvenda
ON item_venda(cod_venda)
```

**Manipulação da Base de Dados:**

```
/*CADASTRAR LOJA*/

CREATE PROCEDURE cadastrar_loja
@cnpj VARCHAR(18),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(20),
@uf char(2),
@nome VARCHAR(50)
AS
INSERT into loja VALUES(@cnpj, @endereco, @numero, @cidade, @uf, @nome)
   IF @@ROWCOUNT>0
       BEGIN
           PRINT 'Loja cadastrada com sucesso!'
           RETURN 1
       END
   ELSE
       BEGIN
           PRINT 'Erro ao cadastrar loja'
           RETURN 0
       END

-----------------------------------------------------------------------------------------------------------------------

/*CADASTRAR CAlÇADO*/

ALTER PROCEDURE cadastrar_calcado
@nome VARCHAR(50),
@marca VARCHAR(15),
@departamento VARCHAR(15),
@preco money,
@cor VARCHAR(15),
@qtd_estoque INT,
@tipo SMALLINT,
@numero INT
as
BEGIN TRANSACTION
INSERT into produtos VALUES(@nome, @marca, @departamento, @preco, @cor,
@qtd_estoque, @tipo)
DECLARE  @cod_produto  INT  =  (select  IDENT_CURRENT('produtos')  AS
[IDENT_CURRENT('produtos')])
IF @@ROWCOUNT>0 AND @tipo = 1
```

```
    BEGIN
        INSERT into calcado VALUES(@cod_produto, @numero)
        IF @@ROWCOUNT>0
                 BEGIN PRINT 'Produto cadastrado com sucesso!' COMMIT
TRANSACTION RETURN 1 END
          ELSE PRINT 'Erro ao cadastrar produto!' ROLLBACK TRANSACTION
RETURN 0
    END
ELSE PRINT 'Erro ao cadastrar produto!' ROLLBACK TRANSACTION RETURN 0

-----------------------------------------------------------------------------------------------------------

/*CADASTRAR CLIENTE PF*/

ALTER PROCEDURE cadastrar_cliente_PF
@codloja int,
@nome VARCHAR(50),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(50),
@UF CHAR(2),
@telefone VARCHAR(14),
@tipo_pessoa int,
@dt_nasc date,
@tipo_cliente INT,
@cpf VARCHAR(14),
@rg VARCHAR(13)
AS
BEGIN TRANSACTION
      INSERT into pessoa VALUES(@codloja, @nome, @endereco, @numero,
@cidade, @UF, @telefone, @tipo_pessoa)
       DECLARE @cod_pessoa INT = (select IDENT_CURRENT('pessoa') AS
[IDENT_CURRENT('pessoa')])
    IF @@ROWCOUNT>0 AND @tipo_pessoa = 1
        BEGIN
                    INSERT into cliente VALUES(@cod_pessoa, @dt_nasc,
@tipo_cliente)
             DECLARE @cod_cliente INT = (select IDENT_CURRENT('cliente')
AS [IDENT_CURRENT('cliente')])
            IF @@ROWCOUNT > 0 AND @tipo_cliente = 1
                BEGIN
                    INSERT into pessoa_fisica VALUES(@cod_cliente, @cpf,
@rg)
                   if @@ROWCOUNT>0
                        BEGIN
                            COMMIT TRANSACTION
                            PRINT 'Cliente cadastrado com sucesso!'
                            RETURN 1
```

```
                    END
                ELSE
                    BEGIN
                        PRINT 'Erro ao cadastrar cliente!'
                        ROLLBACK TRANSACTION
                        RETURN 0
                    END
            END
        ELSE
            PRINT 'Erro ao cadastrar cliente!'
            ROLLBACK TRANSACTION
    END
ELSE
    PRINT 'Erro ao cadastrar cliente!'
    ROLLBACK TRANSACTION

-------------------------------------------------------------------------------------------------------------------

/*CADASTRAR CLIENTE PJ*/

Create PROCEDURE cadastrar_cliente_PJ
@codloja int,
@nome VARCHAR(50),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(50),
@UF CHAR(2),
@telefone VARCHAR(14),
@tipo_pessoa int,
@dt_nasc date,
@tipo_cliente INT,
@cnpj VARCHAR(14),
@nome_fantasia VARCHAR(13)
AS
BEGIN TRANSACTION
      INSERT into pessoa VALUES(@codloja, @nome, @endereco, @numero,
@cidade, @UF, @telefone, @tipo_pessoa)
       DECLARE @cod_pessoa INT = (select IDENT_CURRENT('pessoa') AS
[IDENT_CURRENT('pessoa')])
    IF @@ROWCOUNT>0 AND @tipo_pessoa = 1
        BEGIN
                    INSERT into cliente VALUES(@cod_pessoa, @dt_nasc,
@tipo_cliente)
            DECLARE @cod_cliente INT = (select IDENT_CURRENT('cliente')
AS [IDENT_CURRENT('cliente')])
            IF @@ROWCOUNT > 0 AND @tipo_cliente = 0
                BEGIN
                        INSERT into pessoa_juridica VALUES(@cod_cliente,
```

```
@cnpj, @nome_fantasia)
                    if @@ROWCOUNT>0
                         BEGIN
                             COMMIT TRANSACTION
                             RETURN 1
                         END
                     ELSE
                         BEGIN
                             ROLLBACK TRANSACTION
                             RETURN 0
                         END
                END
            ELSE
                ROLLBACK TRANSACTION
        END
    ELSE
        ROLLBACK TRANSACTION
```

---

```
/*CADASTRAR FORNECEDOR*/

create PROCEDURE cadastrar_fornecedor
@cnpj VARCHAR(18),
@nome_fantasia VARCHAR(50),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(20),
@UF CHAR(2)
AS
INSERT into fornecedor VALUES(@cnpj, @nome_fantasia, @endereco,
@numero, @cidade, @UF)
IF @@ROWCOUNT>0
    BEGIN
        PRINT 'Fornecedor cadastrado com sucesso'
        RETURN 1
    END
ELSE
    BEGIN
        PRINT 'Erro'
        RETURN 0
    END
```

---

```
/*CADASTRAR GERENTE*/
```

```
ALTER PROCEDURE cadastrar_gerente
@codloja int,
@nome VARCHAR(50),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(50),
@UF CHAR(2),
@telefone VARCHAR(14),
@tipo_pessoa int,
@RF INT,
@salario money,
@tipo_func INT,
@departamento VARCHAR(25)
AS
BEGIN TRANSACTION
INSERT into pessoa VALUES(@codloja, @nome, @endereco, @numero, @cidade,
@UF, @telefone, @tipo_pessoa)
DECLARE   @cod_pessoa   INT   =   (select   IDENT_CURRENT('pessoa')   AS
[IDENT_CURRENT('pessoa')])
IF @@ROWCOUNT>0 AND @tipo_pessoa = 0
    BEGIN
            INSERT into funcionario VALUES(@RF, @cod_pessoa, @codloja,
@salario, @tipo_func)
        IF @@ROWCOUNT>0 AND @tipo_func = 0
            BEGIN
            INSERT into gerente VALUES(@RF, @departamento)
                IF @@ROWCOUNT>0
                    BEGIN
                        COMMIT TRANSACTION
                        RETURN 1
                    END
                ELSE ROLLBACK TRANSACTION RETURN 0
            END
        ELSE ROLLBACK TRANSACTION
    END

-----------------------------------------------------------------------------------------------------------------------

/*CADASTRAR REQUISIÇÃO DE COMPRA*/

CREATE PROCEDURE cadastrar_RC
@cod_fornecedor INT,
@RF_gerente INT,
@cod_produto INT,
@quantidade INT,
@data date
AS
BEGIN TRANSACTION
```

```
INSERT INTO requisicao_compra VALUES(@cod_fornecedor, @cod_produto,
@RF_gerente, @data, 0)
DECLARE @cod_RC INT = (select IDENT_CURRENT('requisicao_compra') AS
[IDENT_CURRENT('requisicao_compra')])
IF @@ROWCOUNT>0
    BEGIN
             INSERT into item_requisicao VALUES(@cod_RC, @cod_produto,
@quantidade)
        IF @@ROWCOUNT>0
            BEGIN
                COMMIT TRANSACTION
                RETURN 1
            END
        ELSE ROLLBACK TRANSACTION RETURN 0
    END
ELSE ROLLBACK TRANSACTION RETURN 0

-----------------------------------------------------------------------------------------------------------------

/*CADASTRAR ROUPA*/

ALTER PROCEDURE cadastrar_roupa
@nome VARCHAR(50),
@marca VARCHAR(15),
@departamento VARCHAR(15),
@preco money,
@cor VARCHAR(15),
@qtd_estoque INT,
@tipo SMALLINT,
@tipo_peca VARCHAR(15),
@tamanho CHAR(3),
@time VARCHAR(15)
as
BEGIN TRANSACTION
INSERT into produtos VALUES(@nome, @marca, @departamento, @preco, @cor,
@qtd_estoque, @tipo)
DECLARE @cod_produto INT = (select IDENT_CURRENT('produtos') AS
[IDENT_CURRENT('produtos')])
IF @@ROWCOUNT>0 AND @tipo = 0
    BEGIN
           INSERT into roupa VALUES(@cod_produto, @tipo_peca, @tamanho,
@time)
        IF @@ROWCOUNT>0
                  BEGIN PRINT 'Produto cadastrado com sucesso!'COMMIT
TRANSACTION RETURN 1 END
          ELSE PRINT 'Erro ao cadastrar produto!' ROLLBACK TRANSACTION
RETURN 0
    END
```

```
ELSE PRINT 'Erro ao cadastrar produto!' ROLLBACK TRANSACTION RETURN 0

-----------------------------------------------------------------------------------------------------------------------

/*CADASTRAR VENDA*/

Create PROCEDURE cadastrar_venda
@cod_cliente int,
@rf_vendedor int,
@data DATE,
@cod_produto INT,
@quantidade INT
AS
BEGIN TRANSACTION
INSERT into venda VALUES(@cod_cliente, @rf_vendedor, @data, 0)
DECLARE   @cod_venda   INT   =   (select   IDENT_CURRENT('venda')   AS
[IDENT_CURRENT('venda')])
IF @@ROWCOUNT>0
    BEGIN
               INSERT  into  item_venda  VALUES(@cod_produto,@cod_venda,
@quantidade)
        IF @@ROWCOUNT>0
            BEGIN
                COMMIT TRANSACTION
                RETURN 1
            END
        ELSE ROLLBACK TRANSACTION RETURN 0
    END
ELSE ROLLBACK TRANSACTION RETURN 0

-----------------------------------------------------------------------------------------------------------------------

/*CADASTRAR VENDEDOR*/

CREATE PROCEDURE cadastrar_vendedor
@codloja int,
@nome VARCHAR(50),
@endereco VARCHAR(100),
@numero int,
@cidade VARCHAR(50),
@UF CHAR(2),
@telefone VARCHAR(14),
@tipo_pessoa int,
@RF INT,
@salario money,
@tipo_func INT
```

```
AS
BEGIN TRANSACTION
INSERT into pessoa VALUES(@codloja, @nome, @endereco, @numero, @cidade,
@UF, @telefone, @tipo_pessoa)
DECLARE   @cod_pessoa   INT   =   (select   IDENT_CURRENT('pessoa')   AS
[IDENT_CURRENT('pessoa')])
IF @@ROWCOUNT>0 AND @tipo_pessoa = 0
   BEGIN
            INSERT into funcionario VALUES(@RF, @cod_pessoa, @codloja,
@salario, @tipo_func)
       IF @@ROWCOUNT>0 AND @tipo_func = 1
           BEGIN
           INSERT into vendedor VALUES(@RF, 0)
               IF @@ROWCOUNT>0
                   BEGIN
                       COMMIT TRANSACTION
                       RETURN 1
                   END
               ELSE ROLLBACK TRANSACTION RETURN 0
           END
       ELSE ROLLBACK TRANSACTION
   END

-----------------------------------------------------------------------------------------------------------------

/*ADICIONAR ITEM À VENDA*/

Create TRIGGER adicionar_itemvenda
on item_venda
after INSERT
as
BEGIN TRANSACTION
UPDATE venda SET total = total + ((select quantidade from inserted) *
(select preco from produtos where codigo = (select cod_produto from
inserted)))
WHERE venda.codigo = (SELECT cod_venda from inserted)
UPDATE vendedor SET comissao = comissao + (0.2 * (select total FROM
venda where codigo = (select cod_venda from inserted) AND RF = (SELECT
RF from venda where codigo = (select cod_venda from inserted))))

IF @@ROWCOUNT>0
    BEGIN
             UPDATE produtos SET qtd_estoque = qtd_estoque - (select
```

```
quantidade from inserted)
        WHERE codigo = (select cod_produto from inserted)
         DECLARE @estoque INT = (select qtd_estoque from produtos where
codigo = (select cod_produto from inserted))
        IF @@ROWCOUNT>0 AND @estoque >= 0
            BEGIN
                PRINT 'Produto(s) adicionado(s) com sucesso!'
                COMMIT TRANSACTION
            END
        ELSE
            BEGIN
                PRINT 'Produto indisponível!'
                ROLLBACK TRANSACTION
            END
    END
ELSE ROLLBACK TRANSACTION

-----------------------------------------------------------------------------------------------------------------

/*ADICIONAR ITEM À REQUISIÇÃO DE COMPRA*/

Create TRIGGER add_item_requisicao
ON item_requisicao
FOR insert
AS
BEGIN TRANSACTION
UPDATE requisicao_compra SET total = total + ((select quantidade from
inserted) * (0.8 *(select preco from produtos

where codigo = (select cod_produto from inserted))))
WHERE requisicao_compra.codigo = (select cod_req from inserted)
IF @@ROWCOUNT>0
    BEGIN
        --Inserção dos produtos fornecidos ao estoque
             UPDATE produtos SET qtd_estoque = qtd_estoque + (select
quantidade from inserted)
        WHERE codigo = (select cod_produto from inserted)
        IF @@ROWCOUNT>0
            BEGIN
                PRINT 'Produto(s) adicionado(s) com sucesso!'
                COMMIT TRANSACTION
            END
        ELSE
            BEGIN
                ROLLBACK TRANSACTION
            END
    END
ELSE ROLLBACK TRANSACTION
```

```
-----------------------------------------------------------------------------------------------------------------

/*ATUALIZAR REQUISIÇÃO DE COMPRA*/

Create TRIGGER atualizar_RC
ON item_requisicao
AFTER UPDATE
AS
IF UPDATE(quantidade)
   BEGIN TRANSACTION
      DECLARE @qtd_final int = (select quantidade from inserted)
      DECLARE @qtd_inicial int = (select quantidade from deleted)
      IF @qtd_final > @qtd_inicial
           BEGIN
                      UPDATE produtos SET qtd_estoque = qtd_estoque +
(@qtd_final - @qtd_inicial)
                     WHERE produtos.codigo = (select cod_produto from
inserted)
           END
      ELSE
           BEGIN
                      UPDATE produtos SET qtd_estoque = qtd_estoque -
(@qtd_inicial - @qtd_final)
                     WHERE produtos.codigo = (select cod_produto from
inserted)
           END
       IF @@ROWCOUNT > 0
           BEGIN
                       UPDATE requisicao_compra SET total = (select
sum(item_requisicao.quantidade* (0.8*produtos.preco))
            from item_requisicao inner join produtos on produtos.codigo
= item_requisicao.cod_produto)
                WHERE requisicao_compra.codigo = (select cod_req from
inserted) OR requisicao_compra.codigo = (select cod_req from deleted)
           COMMIT TRANSACTION
           END
       ELSE
           BEGIN
               ROLLBACK TRANSACTION
           END


/*ATUALIZAR ITEM DE VENDA*/

CREATE TRIGGER atualizar_itemVenda
ON item_venda
AFTER UPDATE
AS
```

```
IF UPDATE(quantidade)
  BEGIN TRANSACTION
     DECLARE @qtd_final int = (select quantidade from inserted)
     DECLARE @qtd_inicial int = (select quantidade from deleted)
     IF @qtd_final > @qtd_inicial
          BEGIN
                      UPDATE produtos SET qtd_estoque = qtd_estoque -
(@qtd_final - @qtd_inicial)
                     WHERE produtos.codigo = (select cod_produto from
inserted) --se há mais pedidos em relação ao início, tira do estoque
          END
     ELSE
          BEGIN
                      UPDATE produtos SET qtd_estoque = qtd_estoque +
(@qtd_inicial - @qtd_final)
                     WHERE produtos.codigo = (select cod_produto from
inserted) --se há menos pedidos em relação ao início, arescenta ao
estoque
          END
      IF @@ROWCOUNT > 0
          BEGIN
                                    UPDATE venda SET total = (select
sum(item_venda.quantidade*produtos.preco)
                from item_venda inner join produtos on produtos.codigo =
item_venda.cod_produto)
                WHERE venda.codigo = (select cod_venda from inserted) OR
venda.codigo = (select cod_venda from deleted)
                    IF @@ROWCOUNT>0
                        BEGIN
                            UPDATE vendedor
                                SET comissao = comissao + (0.2 * (select
total FROM venda where codigo = (select cod_venda from inserted)
                                                        AND RF = (SELECT RF
from venda where codigo = (select cod_venda from inserted))))
                        COMMIT TRANSACTION
                        END
                   ELSE ROLLBACK TRANSACTION
          END
      ELSE
          BEGIN
```

```
            ROLLBACK TRANSACTION
        END

-------------------------------------------------------------------------------------------------------------------

/*EXCLUIR ITEM DE VENDA*/

Create TRIGGER excluir_itemvenda
on item_venda
after delete
as
BEGIN TRANSACTION
UPDATE venda SET total = total - ((select quantidade from deleted) *
(select preco from produtos

where codigo = (select cod_produto from deleted)))
WHERE venda.codigo = (select cod_venda from deleted)
IF @@ROWCOUNT>0
    BEGIN
            UPDATE produtos SET qtd_estoque = qtd_estoque + (select
quantidade from deleted)
        WHERE produtos.codigo = (select cod_produto from deleted)
        IF @@ROWCOUNT>0
            BEGIN
                PRINT 'Produto(s) removido(s) da venda com sucesso!'
                COMMIT TRANSACTION
            END
        ELSE
            BEGIN
                PRINT 'Não foi possível excluir o produto da venda!'
                ROLLBACK TRANSACTION
            END
    END
ELSE ROLLBACK TRANSACTION

-------------------------------------------------------------------------------------------------------------------

/*EXCLUIR ITEM REQUISIÇÃO*/

Create TRIGGER excluir_itemRequisicao
on item_requisicao
after delete
as
BEGIN TRANSACTION
UPDATE requisicao_compra SET total = total - ((select quantidade from
deleted) * (0.8*(select preco from produtos

where codigo = (select cod_produto from deleted))))
```

```
WHERE requisicao_compra.codigo = (select cod_req from deleted)
IF @@ROWCOUNT>0
    BEGIN
             UPDATE produtos SET qtd_estoque = qtd_estoque - (select
quantidade from deleted)
        WHERE produtos.codigo = (select cod_produto from deleted)
        IF @@ROWCOUNT>0
            BEGIN
                PRINT 'Produto(s) removido(s) da RC com sucesso!'
                COMMIT TRANSACTION
            END
        ELSE
            BEGIN
                PRINT 'Não foi possível excluir o produto da RC!'
                ROLLBACK TRANSACTION
            END
    END
ELSE ROLLBACK TRANSACTION
```