Serviço Nacional de Aprendizagem Comercial – SENAC

Atividade Projeto Integrador

Curso: Administrador de Banco de Dados

UC4 – Instalar e configurar o sistema de banco de dados

Professor: Edgar Oliveira Cardoso

Aluno(a)(s): Lucas Araújo de Carvalho, Daniele Anjos e João Victor dos Santos

Execução: Senac
Realização: GOVERNO DO ESTADO | SECRETARIA DA EDUCAÇÃO

Sumário

1. Introdução

O projeto se refere a um banco fictício, de um minimercado fictício, chamado Pravda. Por se tratar de um banco fictício, dados e situações serão meramente ilustrativas e, portanto, não se faz necessário as adequações as normas da Lei Geral de Proteção de Dados (LGPD).

As situações que o banco terá que suprir – limitadas ao imaginário dos discentes – variam entre um fluxo de clientes que tem um aumento significativo aos fins de semana até a relação do quadro de funcionários. Temos o estoque do minimercado que é reposto a cada 15 dias e/ou operando a um estoque mínimo, através de pedidos diretos às empresas. No banco há dois funcionários que trabalharão no estabelecimento, são eles: Lucas e Daniele. Os horários para os funcionários variam.

Este trabalho tem como requisito fundamental a demonstração dos conhecimentos obtidos durante os quatro (04) meses do curso Administrador de Banco de Dados, ofertado pelo SENAC através do programa Educar Para Trabalhar, iniciativa do Governo do Estado da Bahia. O mesmo abrangerá as funcionalidades necessárias para o armazenamento de dados do estabelecimento, a manutenção do banco, o monitoramento e permitindo o CRUD (Create, Read, Update and Delete [Criar, Ler, Atualizar e Excluir]) de várias partes do sistema através da integração com o Power BI.

Será feito a manipulação dos dados de funcionários, fornecedores e clientes, compras, estoque, pedidos e relatórios contendo informações sobre as compras e pedidos de um determinado período a fim de ter um controle das finanças, um balanço. As compras se referem ao que entra para as vendas aos clientes. Os pedidos são produtos e mercadorias que chegarão ao estabelecimento as quais foram previamente solicitadas.

1.1. Objetivo

Desenvolver um sistema de banco de dados, integrando ao PowerBI – como uma interface – a fim de suprir as necessidades dos usuários e facilitar o controle de estoque do supermercado, visando substituir métodos arcaicos de controle.

1.2. Público Alvo

Os profissionais que gerenciam o caixa, o estoque e os que gerenciam os funcionários, no caso, o departamento pessoal.

2. Regras de Negócio

Por se tratar de um minimercado e para efeito de materialidade, optamos por uma máquina fraca para o armazenamento dos dados. Devido a esse fator, as colunas da maioria das tabelas serão *smallint,* permitindo mais de 30 mil tuplas. Consideramos o fluxo de clientes, fornecedores, armazenamento e dentre outros dados que serão armazenados. As relações de “pedidos” e “compras”, entretanto, terão a chave primária do tipo *int,* pois terão uma quantidade ligeiramente maior de tuplas.

Colunas que estão como "*varchar*" em vez de “*char*” é pela necessidade de não ter um valor predefinido/limitado. Portanto, só teremos o padrão de “*char*” em valores que não fujam de um padrão de caracteres, como no caso do telefone, por exemplo, em que terá X quantidade de dígitos, também por questões de desempenho. Para dados como “nome” nas relações “pessoa” e “produto”, ficará como “*varchar*”

Colunas em *unique* servirão tanto como um indexador, ou seja, será uma coluna de valor único, não repetido. As colunas que servirão como chave secundária (*unique*) serão: “RG”, “CPF”, “CNPJ” e “telefone”.

O código das chaves primárias seguirá um padrão em que se chamará como “cod” e serão auto-incrementáveis. Também serão não-nulas, garantindo um nível de segurança, evitando conflitos e unicidade. Contudo teremos como exceção da automatização de incrementação do código na tabela produto em que será definido – por funcionário – o valor do código do produto. Por exemplo: 1000, 1001, 5908…

Tabelas deverão estar normalizadas.

Poderá ter saída de estoque mesmo que o estoque do Produto esteja zerado ou negativo no sistema. Isso acontece, pois, é possível haver estoque do produto (no físico) e não ter sido inserido.

A relação de compra/pedido terá a data do pedido bem como a referência do produto e a quantidade do produto no item.

Em triggers adotamos o padrão de começar sempre com “tr_”.

Teremos relações que não terão Stored Procedures ou Triggers pois alterá-las ou excluí-las significa tirar a integridade, consistência e recuperação dos dados. Isso ocorreria, por exemplo, na tabela compra que tem o código do cliente que comprou e por este motivo, clientes não poderão ser excluídos pois há um campo em compra que consome essa tupla. Por isso alteração ou remoção não serão aceitos ou serão limitados. No caso de aderir à LGPD, será “*null*” em dados sensíveis, contudo não será excluído a tupla.

“Triggers” serão usados. Gatilhos automáticos para modificação ANTES ou DEPOIS de uma inserção, alteração ou exclusão de dados, em pedidos e compras, a fim de atualizar os relatórios, como no caso do estoque do Produto e o valor total de cada Item, além do valor total do Pedido/compra.

Teremos três usuários para as tabelas, são eles: DBA, Lucas e Dani. O DBA terá acesso total, inalienável e imprescritível em todas tabelas. O usuário Lucas terá a opção de INSERÇÃO, ATUALIZAÇÃO e DELEÇÃO de dados. O mesmo vale para o usuário Dani. Todos usuários são locais.

Preferivelmente utilizaremos o MySQL, integrando ao PowerBI.

3. Requisitos Funcionais e Não Funcionais.

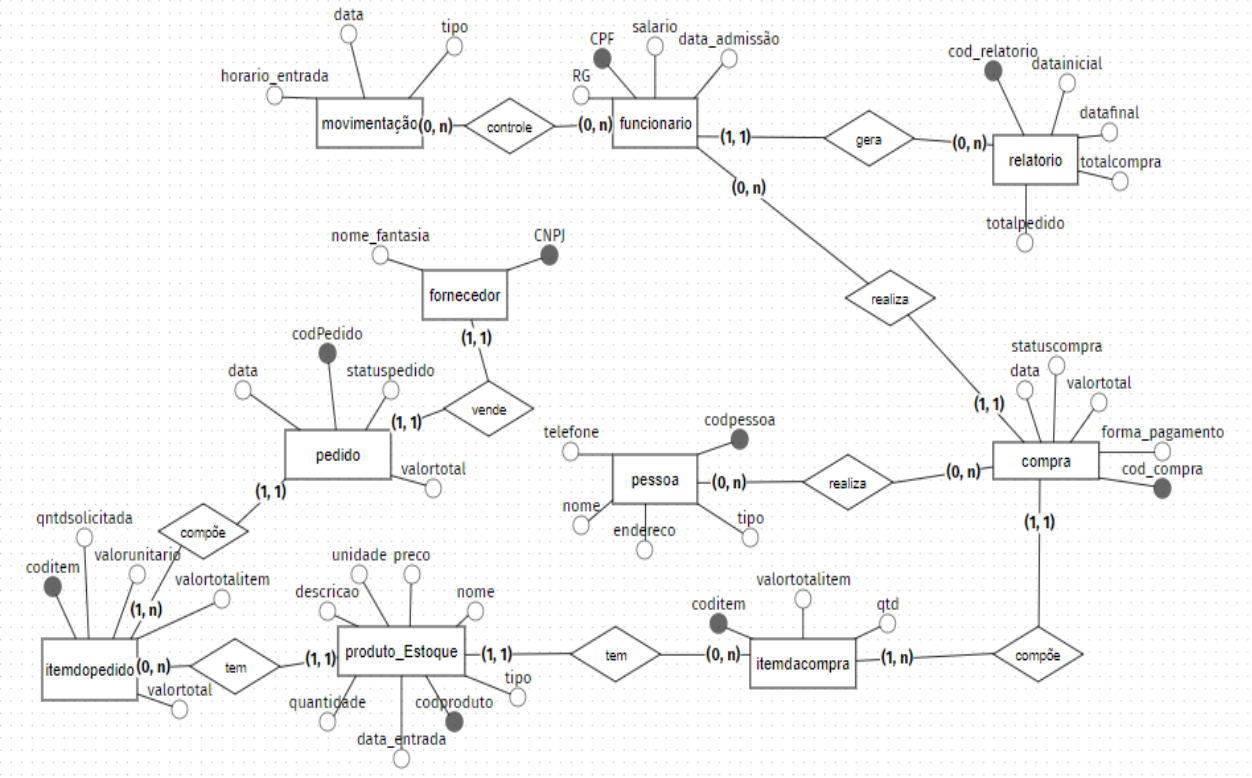
Tabela 1 – Requisitos Funcionais

Código	Descrição
RF001	Cadastro, exclusão e alteração das tabelas serão feitos por Lucas e Dani
RF002	A maioria das tabelas permitirão mais de 30 mil tuplas.
RF003	Permite um grande fluxo de compras e pedidos
RF004	Permite consulta dinâmica baseado em chaves secundárias
RF005	Permite mais do que os 2 funcionários iniciais, visando expansão
RF006	Permite manter cadastros de clientes e fazer a manutenção quando necessário.
RF007	Permite subtração do estoque ainda que esteja negativo ou zerado.
RF008	Permite atualização automática dos relatórios a partir de alguma inserção, alteração ou exclusão.
RF009	Permite cadastrar, atualizar, excluir e consultar em todas tabelas através de Stored Procedures
RF010	Criação de relatórios com as Stored Procedures

Tabela 2 – Requisitos Não-Funcionais

Código	Descrição
RNF001	Software MySQL
RNF002	Software PowerBI integrado ao MySQL
RNF003	O sistema deve estar disponível de segunda a sábado no horário comercial.
RNF004	O sistema deve seguir um padrão de nomenclatura das colunas da seguinte forma: InicialdoTipo_NomeColuna_NomeTabela, ex: i_cod_produto

4. Modelo Conceitual



Os relacionamentos ligam as classes/objetos entre si, criando relações lógicas entre estas as entidades. Os relacionamentos podem ser dos seguintes tipos: Associação: é uma conexão entre classes, e em UML, uma associação é definida com um relacionamento que descreve uma série de ligações. Generalização: É um relacionamento de um elemento mais geral e outro mais específico. O elemento mais específico pode

conter apenas informações adicionais. Dependência e Refinamentos: Dependência é um relacionamento entre elementos, um independente e outro dependente

5. Modelo Relacional

Pessoa (**codpessoa,**telefone,nome,endereco,tipo)

Funcionário (**codfuncionario**,cpf,rg,salário,data_admissão, id_pessoa) id_pessoa refere-se a **pessoa**

Movimentação (**cod_mov**.horario_entrada,data,horário_saída) id_func refere-se a **funcionário**

Fornecedor (**codfornecedor**,CNPJ)

Produto(**codproduto**, nome, descricao, tipo, quantidade, unidade, preco, data_entrada)

Pedido (**codpedido,** id_fornecedor, valortotal, data_pedido, statuspedido) id_fornecedor refere-se a tabela **fornecedor**

itemdopedido(**coditem**, id_produto, id_pedido, qntdsolicitada, preco, valortotalitem) id_produto refere-se a **produto**, id_pedido refere-se a **pedido**.

Compra (**codcompra**, idfuncionario, idcliente_pessoa, data_compra, valortotal, formapagamento, statuscompra) idfuncionario refere-se a **funcionario**, idcliente_pessoa refere-se a **pessoa**

itemdacompra(**coditem**, id_produto, id_compra, qntdsolicitada, valortotalitem) id_produto refere-se a **produto**, id_compra refere-se a **compra**

Relatório (**coditem**,cod_relatorio,datainicial,datafinal,totalcompra,totalpedido) id_funcionario refere-se a **codcompra**

6. Script estrutura do banco (DDL)

NOTA: essa tabela tem a função de armazenar dados de pessoas em que se diferem entre clientes e funcionários.

```
create table pessoa(
codpessoa smallint NOT NULL primary key auto_increment,
nome varchar(20) NOT NULL,
sobrenome varchar(40) NOT NULL,
telefone char(18) NOT NULL,
rua varchar(30),
cidade varchar(20),
estado char(2),
CEP char(10),
bairro varchar(20),
numero tinyint,
tipo ENUM ('cliente', 'funcionario')
);
```

NOTA: essa tabela tem a função de armazenar dados de funcionários. Detalhe: eles estão também na tabela pessoa, o que sugere uma chave estrangeira nesta tabela.

```
create table funcionario(
codfuncionario smallint NOT NULL primary key auto_increment,
cpf char(14) unique not null,
rg varchar(16) unique not null,
salario decimal(7,2),
datacontratacao date,
id_pessoa smallint REFERENCES pessoa(codpessoa),
index(cpf)
);
```

NOTA: essa tabela tem a função de armazenar dados de controle de ponto diários de um determinado funcionário.

```
create table movimentacao(
cod_mov int primary key not null auto_increment,
```


```
horarioentrada datetime default current_timestamp not null,
horariosaida datetime,
id_func smallint REFERENCES funcionario(codfuncionario));
```

NOTA: essa tabela tem a função de armazenar dados de fornecedores.

```
create table fornecedor(
codfornecedor smallint NOT NULL primary key auto_increment,
cnpj char(18) unique not null,
index(cnpj)
);
```

NOTA: essa tabela tem a função de armazenar dados de produtos em estoque. Em resumo é uma tabela de estoque.

```
create table produto(
codproduto smallint NOT NULL primary key auto_increment,
nome varchar(40) NOT NULL,
descricao varchar(80),
tipo char(20) NOT NULL,
quantidade decimal(7,3) NOT NULL,
unidade char(2) NOT NULL,
preco DOUBLE NOT NULL,
data_entrada date  NOT NULL
);
```

NOTA: essa tabela tem a função de armazenar dados de pedidos de produtos, fornecidos por fornecedores.

```
create table pedido(
codpedido int NOT NULL primary key auto_increment,
id_fornecedor smallint NOT NULL,
data_pedido timestamp NOT NULL DEFAULT current_timestamp(),
valortotal DOUBLE NOT NULL,
statuspedido varchar(12) default 'pendente' NOT NULL,
foreign key(id_fornecedor) references fornecedor(codfornecedor)
);
```

NOTA: essa tabela tem a função de armazenar dados de itens de pedidos (em que cada tupla representa um item em si) que irão alimentar a tabela estoque/produtos. Preco refere-se ao valor unitário, valortotalitem refere-se a quantidade solicitava multiplicado pelo valor unitário.

```
create table itemdopedido(
coditem smallint NOT NULL primary key auto_increment,
id_produto smallint NOT NULL,
id_pedido int NOT NULL,
qntdsolicitada decimal(6,3) NOT NULL,
preco DOUBLE NOT NULL,
valortotalitem decimal(7,2) NOT NULL,
foreign key(id_produto) references produto(codproduto),
foreign key(id_pedido) references pedido(codpedido)
);
```

NOTA: essa tabela tem a função de armazenar dados de movimentação de compras feitas por funcionários ou clientes, em que justificam entrada e saída do caixa.

```
create table compra(
codcompra int NOT NULL primary key auto_increment,
idfuncionario smallint,
idcliente_pessoa smallint,
data_compra timestamp NOT NULL DEFAULT NOW(),
valortotal DOUBLE NOT NULL,
formapagamento varchar(17) NOT NULL,
statuscompra varchar(10) default 'pendente' NOT NULL,
foreign key(idfuncionario) references funcionario(codfuncionario),
foreign key(idcliente_pessoa) references pessoa(codpessoa)
);
```

NOTA: essa tabela tem a função de armazenar dados de itens da compra em que compõe o que saíra do estoque. Ou ainda que componha a lista de compra de um cliente.

```
create table itemdacompra(
coditem int NOT NULL primary key auto_increment,
id_produto smallint NOT NULL,
```


```
id_compra int NOT NULL,
qntdsolicitada decimal(7,3) NOT NULL,
valortotalitem decimal(7,2) NOT NULL,
foreign key(id_produto) references produto(codproduto),
foreign key(id_compra) references compra(codcompra)
);

create table relatorio(
codrelatorio smallint NOT NULL primary key auto_increment,
id_funcionario smallint,
datainicial timestamp NOT NULL DEFAULT now(),
datafinal date,
totalpedido decimal(7,2) NOT NULL,
totalcompra decimal(7,2) NOT NULL
);
```

6.1. Segurança do banco - usuários

Nome	Função
Lucas	Inserir, deletar e atualizar em todas tabelas.
DBA	Todos privilégios.
Dani	Inserir, deletar e atualizar em todas tabelas.

7. Script manipulação no banco (DML)

INSERT INTO pessoa (nome, sobrenome, telefone, rua, cidade, estado, cep, bairro, numero, tipo) VALUES ('Pamela', 'Oliveira', '73 915925370', 'california' , 'coaraci' , 'BA', '49400-186', 'california' , '15', 'cliente'), ('lucas' , 'silva', '73 991503370', 'calitornia' , 'coaraci' , 'BA', '19460-171', 'california' , '107', 'cliente'), ('lucas' , 'silva', '73 968103582', 'california' , 'coaraci' , 'BA', '56099-179', 'california' , '95', 'cliente'), ('Galileu', 'Victor', '73 961072370', 'california' , 'coaraci' , 'BA', '20519-126' , 'calisto filho', '102', 'cliente'),

('Mohamed', 'afrodi' , '73 999251835', 'california' , 'itabuna' , 'BA', '42090-193', 'Inácio costa', '103', 'funcionario'), ('Neymar', 'Júnior', '73 968411370' , 'manguinho' , 'coaraci' , 'BA' , '91639-176' , 'horizonte', '92' , 'cliente'), ('luandeson' , 'silva' , '73 968499740' , 'california' , 'coaraci' , 'BA' , '48060-116' , 'santo enes ' , '75' , 'funcionario'),('lula' , 'souza', '73 988498725' , 'california' , 'coaraci' , 'BA' , '48876-116' , 'Meneses', '55' , 'funcionario'),('Geronimo' , 'azevedo' , '73 988997970', 'california' , 'coaraci' , 'BA' , '48000-176' , 'california' , '02', 'funcionario'), ('Bolsonaro', 'caetano' , '73 988479680', 'california' , 'coaraci' , 'BA' , '48090-175' , 'santo Antônio' , '66' , 'cliente');

INSERT INTO funcionario (cpf,rg,salario,datacontratacao,id_pessoa) VALUES

('082.395.185-39', '11111111-11', '1200.00', '2021/03/12', '1'), ('074.305.638-30', '10584942-37', '1200.00', '2009/11/05', '32'), ('289.314.185-09', '14307387-09', '1200.00', '2011/09/09', '1'), ('079.399.297-69', '19502511-71', '1200.00', '2010/03/11', '1'),('917.765.186-01', '10161937-31', '3600.00', '2019/06/02', '1'),('017.376.155-28', '11110360-11', '1200.00', '2012/09/01', '1'),('028.389.115-89', '01831129-11', '1200.00', '2017/01/05', '2'),('172.069.985-17', '0163961-11', '1200.00', '2020/11/28', '9'),('193.395.185-39', '11072419-11', '6600.00', '2015/01/15', '8'),('103.017.125-79', '21103611-11', '2400.00', '2014/07/19', '7');

INSERT INTO movimentacao (horarioentrada, horariosaida, id_func) VALUES

(2011-12-18 13:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 19:17:17, 1), (2011-12-18 12:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 11:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 14:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 16:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 18:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 20:17:17, 1), (2011-12-18 13:17:17, 2011-12-18 18:17:17, 1);

insert into fornecedor(cnpj) VALUES

('71.915.463/0001-21'), ('44.386.066/0001-37'), ('46.232.121/0001-60'), ('86.463.163/0001-26'), ('17.744.465/0001-50'), ('62.172.188/0001-02'), ('39.168.114/0001-80'),('73.107.870/0001-38'), ('62.777.763/0001-09'), ('35.025.173/0001-49');

INSERT INTO produto(nome, descricao, tipo, quantidade, unidade, preco, data_entrada) VALUES

('Arroz', '', 'Graos', 100, 'kg', 4.00, '2022-01-10'), ('Feijao', '', 'Graos', 200, 'kg', 7.00, '2022-01-10'), ('Macarrao', '', 'Massa', 150, 'kg', 3.50, '2022-01-10'), ('Aveia', '', 'Graos', 50, 'pc', 5.00, '2022-01-10'), ('Bolacha', '', 'doce', 70, 'cx', 2.50, '2022-01-10'), ('Presunto', '', 'Carne', 100, 'kg', 4.00, '2022-01-10'), ('Frango', '', 'Carne', 100, 'kg', 20.00, '2022-01-10'), ('Iorgute', '', 'Liquido', 40, 'L', 4.00, '2022-01-10'), ('Mortandela', '', 'geral', 10, 'kg', 4.00, '2022-01-10'), ('Queijo', '', 'geral', 10, 'kg', 4.00, '2022-01-10');

insert into pedido(id_fornecedor,valortotal,statuspedido) values

(1,987.09,'concluido'), (2,505.42,'pendente'),(3,54.05,'concluido'), (4,841.60,'pendente'), (5,255.12,'concluido'), (6,125.65,'pendente'), (7,646.56,'concluido'), (8,648.32,'pendente'), (9,321.87,'concluido'), (10,981.54,'concluido'), (1,987.09,'concluido'),(2,505.42,'pendente'), (3,54.05,'concluido'), (4,841.60,'pendente'), (5,255.12,'concluido'),(6,125.65,'pendente'), (7,646.56,'concluido'),(8,648.32,'pendente'), (9,321.87,'concluido'), (10,981.54,'concluido');

insert into itemdopedido(id_produto,id_pedido,qntdsolicitada,preco,valortotalitem) values

(1,1,65,398,3654.4),(2,2,1,557,557.0),(3,3,8,117,854.14),(4,4,9,54,425.05),(5,5,4,122,458.25),(6,6,4,49,154.21), (7,7,9,37,478.79),(8,8,4,565,5344.45),(9,9,6,45,556.65),(10,10,5,157,2136.45);

insert into compra(valortotal,formapagamento,statuscompra) values

(454.54,'debito','concluido'),(124,'credito','concluido'),(200,'pix','pendente'),(154.50,'debito','concluido'),(173.50,'pix','pendente'),(833.33,'credito','pendente'),(511.16,'debito','pendente'),(687.69,'credito','concluido'),(852.36,'pix','concluido'),(687.41,'dinheiro','concluido');

insert into itemdacompra(id_produto,id_compra,qntdsolicitada,valortotalitem) values

(1,1,5,564),(2,2,3,845.54),(3,3,6,897.79),(4,4,9,1542.14),(5,5,2,645),(6,6,4,486.54),(7,7,4,225.50),(8,8,9,987.54),(9,9,1,596),(10,10,954.58);

insert into relatorio(id_funcionario,datafinal,totalpedido,totalcompra) values

(1,'2002-05-01',5,954.41),(2,'2015-10-09','2',754.41),(3,'2015-07-10',1,144.21),(4,'2013-07-15',6,984.58),(5,'2021-02-08',2,147.58),(6,'2022-03-21',7,397.58),(7,'2021-10-19',3,654.45),(8,'2023-02-01',12,3843.67),(9,'2013-03-12',5,198.25),(10,'2019-06-01',06,184.12);

8. Transactions, stored procedures e triggers.

8.1 Procedimento para cadastrar pessoas.

```
DELIMITER $$
create procedure sp_cadastrarpessoa(
IN INome varchar(20),
IN Isobrenome varchar(40),
IN Itelefone char(18),
IN Irua varchar(30),
IN Icidade varchar(20),
```


```
IN Iestado char(2),
IN ICEP char(10),
IN Ibairro varchar(20),
IN Inumero tinyint,
IN Itipo ENUM ('cliente', 'funcionario'))
begin
START transaction;
insert into pessoa values (INome, Isobrenome, Itelefone, Irua, Icidade, Iestado, ICEP, Ibairro, Inumero, Itipo);
if (row_count() > 0)
then commit ;
else SELECT "Algum valor inserido é inválido. Verifique os valores digitados."
rollback ;
END IF ;
END ;
END $$
DELIMITER ;
```

8.2 Procedimento para cadastrar funcionário.

```
DELIMITER $$
create procedure sp_cadastrarfuncionario(
IN Icpf char(14),
IN Irg varchar(16),
IN Isalario decimal(7,2),
IN Idatacontratacao date,
IN Iid_pessoa smallint)
begin
START transaction ;
insert into funcionario values (Icpf, Irg, Isalario, Idatacontratacao, Iid_pessoa);
if (row_count() > 0)
then commit ;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados. "
rollback ;
END IF ;
END ;
end $$
```


```
DELIMITER ;
```

8.3 Procedimento para cadastrar movimentação rotineira de entrada e saída.

```
DELIMITER $$
create procedure sp_cadastrarmovimentacao(
IN id_func smallint)
begin
START transaction ;
insert into movimentacao values (id_func);
if (row_count() > 0)
then commit ;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados. "
rollback ;
END IF ;
END ;
end $$
DELIMITER ;
```

8.4 Procedimento para cadastrar fornecedores.

```
DELIMITER $$
create procedure sp_cadastrarfornecedor(
IN Icnpj char(18))
begin
START transaction;
insert into fornecedor values (Icnpj);
if (row_count() > 0)
then commit;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.5 Procedimento para cadastrar produtos.

```
DELIMITER $$
create procedure sp_cadastrarproduto(
IN Inome varchar(40),
IN Idescricao varchar(80),
IN Itipo char(20),
IN Iquantidade decimal(7,3),
IN Iunidade char(2),
IN Ipreco decimal(7,2),
IN Idata_entrada date)
begin
START transaction;
insert into produto values (Inome, Idescricao, Itipo, Iquantidade, Iunidade, Ipreco, Idata_entrada);
if (row_count() > 0)
then commit;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.6 Procedimento para cadastrar pedidos.

```
DELIMITER $$
create procedure sp_cadastrarpedido(
IN id_fornecedor smallint,
IN valortotal DOUBLE,
IN statuspedido varchar(12))
begin
START transaction;
insert into pedido values (id_fornecedor, valortotal, statuspedido);
if (row_count() > 0)
then commit;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados."
```


```
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.7 Procedimento para cadastrar item dos pedidos.

```
DELIMITER $$
create procedure sp_cadastraritemdopedido(
IN Iid_produto smallint,
IN Iid_pedido int,
IN Iqntdsolicitada decimal(6,3),
IN Ipreco DOUBLE,
IN Ivalortotalitem decimal(7,2))
begin
START transaction;
insert into itemdopedido values (Iid_produto, Iid_pedido, Iqntdsolicitada, Ipreco, Ivalortotalitem);
if (row_count() > 0)
then commit;
else SELECT " Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.8 Procedimento para cadastrar compra.

```
DELIMITER $$
create procedure sp_cadastrarcompra(
IN Iidfuncionario smallint,
IN Iidcliente_pessoa smallint,
IN Ivalortotal DOUBLE,
IN Iformapagamento varchar(17),
IN Istatuscompra varchar(10))
begin
```


```
START transaction;
insert into compra values (Iid_produto, Iid_pedido, Iqntdsolicitada, Ipreco, Ivalortotalitem);
if (row_count() > 0)
then commit;
else SELECT "Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.9 Procedimento para cadastrar item da compra.

```
DELIMITER $$
create procedure sp_cadastraritemdacompra(
IN Iid_produto smallint,
IN Iid_compra int,
IN Iqntdsolicitada decimal(7,3),
IN Ivalortotalitem decimal(7,2))
begin
START transaction;
insert into itemdacompra values (Iid_produto, Iid_compra, Iqntdsolicitada, Ivalortotalitem);
if (row_count() > 0)
then commit;
else SELECT "Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.10 Procedimento para cadastrar relatório.

```
DELIMITER $$
create procedure sp_cadastrarrelatorio(
IN Iid_funcionario smallint,
IN Idatafinal date,
```


```
IN Itotalpedido decimal(7,2),
IN Itotalcompra decimal(7,2))
begin
START transaction;
insert into relatorio values (Iid_funcionario, Idatafinal, Itotalpedido, Itotalcompra);
if (row_count() > 0)
then commit;
else SELECT "Algum valor inserido é inválido. Verifique os valores digitados."
rollback;
END IF;
END;
end$$
DELIMITER ;
```

8.11 Procedimento para atualizar funcionário.

```
DELIMITER $$
create procedure sp_atualizarfuncionario(
IN Icodfuncionario smallint,
IN Icpf char(14),
IN Irg varchar(16),
IN Isalario decimal(7,2),
IN Idatacontratacao date,
IN Iidpessoa smallint,
IN Inome varchar(20),
IN Isobrenome varchar(40),
IN Itelefone char(18),
IN Irua varchar(30),
IN Icidade varchar(20),
IN Iestado char(2),
IN ICEP char(10),
IN Ibairro varchar(20),
IN Inumero tinyint,
IN Itipo ENUM ('cliente', 'funcionario'))
begin
START transaction;
```


```
update funcionario set cpf = Icpf, rg = Irg, salario = Isalario, Idatacontratacao = datacontratacao, Iidpessoa = idpessoa
where codfuncionario = Icodfuncionario;

if (row_count() > 0)

then begin

update pessoa set nome=Inome, telefone=Itelefone, rua=Irua, cidade=Icidade, estado=Iestado, ICEP=CEP,
bairro=Ibairro, numero=Inumero, tipo=Itipo

where pessoa.codpessoa=Icodfuncionario;

if (row_count() > 0)

THEN COMMIT;

else SELECT "Erro. Verifique o valor inserido."

rollback;

END IF;

END;

END IF;

END$$

*/
```

8. 12 Procedimento para atualizar cliente.

```
DELIMITER $$

create procedure sp_atualizarcliente(

IN Icodpessoa smallint,

IN Inome varchar(20),

IN Isobrenome varchar(40),

IN Itelefone char(18),

IN Irua varchar(30),

IN Icidade varchar(20),

IN Iestado char(2),

IN ICEP char(10),

IN Ibairro varchar(20),

IN Inumero tinyint,

IN Itipo ENUM ('cliente', 'funcionario')

)

begin

START transaction;

update pessoa

        set nome=Inome, telefone=Itelefone, rua=Irua, cidade=Icidade, estado=Iestado, ICEP=CEP, bairro=Ibairro,
numero=Inumero, tipo=Itipo
```


```
        where codpessoa=Icodpessoa;
if (row_count() > 0)
THEN COMMIT;
else SELECT "Erro. Verifique o valor inserido."
rollback;
END IF;
END;
END $$
```

8.13 Procedimento para atualizar pedido.

```
create procedure sp_atualizarpedido(
IN Icodpedido int,
IN Iid_fornecedor smallint,
IN Idata_pedido timestamp,
IN Istatuspedido varchar(12))
update pedido
set id_fornecedor=Iid_fornecedor, data_pedido=Idata_pedido,statuspedido=Istatuspedido
where codpedido=Icodpedido;
if (row_count() = 0) SELECT "Algum valor está errado. Reveja"
END $$

create procedure sp_atualizaritemdopedido(
IN Icoditem smallint,
IN Ipreco decimal(7,2),
IN Iqntdsolicitada decimal(6,3))
update itemdopedido
set qntdsolicitada=Iqntdsolicitada, preco=Ipreco
where coditem=Icoditem;
if (row_count() = 0) SELECT "Algum valor está errado. Reveja."
END $$
```

8.14 Procedimento para atualizar compra.

```
create procedure sp_atualizarcompra(
IN Icodcompra int,
IN Iidfuncionario smallint,
IN Iidcliente_pessoa smallint,
```


```
IN Idata_compra timestamp,
IN Iformapagamento varchar(17),
IN Istatuscompra varchar(10))
update compra
set idfuncionario=Iidfuncionario, idcliente_pessoa=Iidcliente_pessoa, data_compra=Idata_compra,
formapagamento=Iformapagamento, statuscompra=Istatuscompra
where codcompra=Icodcompra;
if (row_count() = 0) SELECT "Algum valor está errado. Reveja."
END $$
```

8.15 Procedimento para atualizar item da compra.

```
create procedure sp_atualizaritemdacompra(
IN Icoditem int,
IN Iqntdsolicitada decimal(7,3))
update ItemDaCompra
set qntdsolicitada=Iqntdsolicitada
where coditem=@coditem;
if (row_count() = 0) SELECT "Algum valor está errado. Reveja."
END $$
```

8.16 Procedimento para atualizar status da compra.

```
create procedure sp_atualizarstatuscompra(
IN Iidcliente_pessoa smallint)
update compra
set statuscompra='paga'
where idcliente_pessoa=Iidcliente_pessoa;
if (row_count() = 0) SELECT "Algum valor está errado. Reveja."
END $$
```

8.17 Procedimento para excluir item do pedido.

```
create procedure sp_excluiritemdopedido(
IN Icoditem int)
delete from itemdopedido
where coditem=Icoditem;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
```


```
END$$
```

8.18 Procedimento para excluir compra.

```
create procedure sp_excluircompra(
IN Icodcompra int)
delete from compra
where codcompra=Icodcompra;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.19 Procedimento para excluir fornecedor.

```
create procedure sp_excluirfornecedor(
IN Icodfornecedor smallint)
delete from fornecedor
where codfornecedor=Icodfornecedor;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.20 Procedimento para excluir funcionário.

```
create procedure sp_excluirfuncionario(
IN Icodfuncionario smallint)
delete from funcionario
where codfuncionario=Icodfuncionario;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.21 Procedimento para excluir item da compra.

```
create procedure sp_excluiritemdacompra(
IN Icoditem int)
delete from itemdacompra
where coditem=Icoditem;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.22 Procedimento para excluir movimentação.

```
create procedure sp_excluirmovimentacao(
```


```
IN Icod_mov int)
delete from movimentacao
where cod_mov=Icod_mov;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.23 Procedimento para excluir pedido.

```
create procedure sp_excluirpedido(
IN Icodpedido int)
delete from pedido
where codpedido=Icodpedido;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.24 Procedimento para excluir pessoa.

```
create procedure sp_excluirpessoa(
IN Icodpessoa smallint)
delete from pessoa
where codpessoa=Icodpessoa;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.25 Procedimento para excluir produto.

```
create procedure sp_excluirproduto(
IN Icodproduto smallint)
delete from produto
where codproduto=Icodproduto;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END$$
```

8.26 Procedimento para excluir relatorio.

```
create procedure sp_excluirrelatorio(
IN Icodrelatorio smallint)
delete from relatorio
where codrelatorio=Icodrelatorio;
if (row_count() = 0) SELECT "Não existe esse código. Reveja."
END $$
```

8.1.1 Visualizador para consultar funcionário.

```
DELIMITER ;
CREATE VIEW vw_consultarfuncionario AS
select * from funcionario inner join pessoa
on funcionario.id_pessoa=codpessoa;
```

8.1.2 Visualizador para consultar cliente.

```
CREATE VIEW vw_consultarcliente AS
select * from pessoa
where tipo = 'cliente';
```

8.2.1 Gatilho que atualiza o valor total de um pedido quando um item do pedido é criado.

```
DELIMITER $$
create trigger t_inclusaodeitemdopedido AFTER INSERT ON itemdopedido
FOR EACH ROW
BEGIN
update pedido
   set pedido.valortotal = pedido.valortotal + (new.itemdopedido.valortotalitem * new.itemdopedido.qntdsolicitada)
   where valortotal.codpedido = itemdopedido.id_pedido;
END $$
```

8.2.2 Gatilho que atualiza o valor total de uma compra quando um item da compra é criado, onde há o relacionamento das duas tabelas.

```
DELIMITER $$
#--Atualizar o valor total de uma compra quando um item da compra é criado
create trigger t_inclusaodeitemdacompra AFTER insert ON itemdacompra
FOR EACH ROW
BEGIN
        update compra
set compra.valortotal = compra.valortotal + (new.valortotalitem * new.qntdsolicitada)
where codcompra = new.id_compra;
```


```
END $$
```

8.2.3 Gatilho que atualiza o valor total de um pedido quando um item do pedido for excluído.

```
DELIMITER $$
#-Atualizar o valor total de um pedido e atualizar o estoque do produto quando um item do pedido for excluído
CREATE TRIGGER t_exclusaoitempedido BEFORE DELETE ON itemdopedido
FOR EACH ROW
BEGIN
update pedido
set valortotal = valortotal - (old.valortotalitem * old.qntdsolicitada) where codpedido = old.id_pedido;
END $$
```

8.2.4 Gatilho que atualiza a quantidade de um pedido quando um item do pedido é excluído.

```
DELIMITER $$
CREATE TRIGGER t_exclusaoitempedido2 BEFORE DELETE ON itemdopedido
FOR EACH ROW
BEGIN
update produto set quantidade = quantidade - old.qntdsolicitada where codproduto = old.id_produto;
END $$
```

8.2.5 Gatilho que atualiza o valor total da tabela compra quando um item da compra é excluído.

```
DELIMITER $$
#--Atualizar o valor total de uma compra e atualizar o estoque do produto quando um item da compra for excluído
#pois nao vai entrar receita e nem sair produto
CREATE TRIGGER t_exclusaoitemcompra BEFORE DELETE ON itemdacompra
FOR EACH ROW
BEGIN
update compra
set valortotal = valortotal - (old.valortotalitem * old.qntdsolicitada) where codcompra = old.id_compra;
END $$
```

8.2.6 Gatilho que atualiza a quantidade de uma tupla da tabela produto (Baixa de estoque) quando um item da compra for excluído.

```
DELIMITER $$
#--Atualizar o valor total de uma compra e atualizar o estoque do produto quando um item da compra for excluído
#pois nao vai entrar receita e nem sair produto
CREATE TRIGGER t_exclusaoitemcompra2 BEFORE DELETE ON itemdacompra
FOR EACH ROW
BEGIN
update produto
set quantidade = quantidade + old.qntdsolicitada where codproduto = old.id_produto;
END $$
```

8.2.7 Gatilho que atualiza o valor total de uma tupla da tabela pedido quando um item do pedido for atualizado.

```
DELIMITER $$
CREATE TRIGGER t_atualizacaoitemdopedido AFTER UPDATE ON itemdopedido
FOR EACH ROW
BEGIN
update pedido
set valortotal = (select sum(i.preco*i.qntdsolicitada) from itemdopedido AS i, produto AS p, pedido AS ped
where ped.codpedido=i.id_pedido and p.codproduto=i.id_produto) where codpedido = new.id_pedido;
END $$
```

8.2.8 Gatilho que atualiza o estoque do produto quando um item da compra for alterado.

```
DELIMITER $$
CREATE TRIGGER t_atualizacaoitemdopedido2 AFTER UPDATE ON itemdopedido
FOR EACH ROW
BEGIN
update produto set quantidade = quantidade + (select i.qntdsolicitada-d.qntdsolicitada
from NEW.itemdopedido AS i INNER JOIN OLD.itemdopedido AS d ON i.id_produto=d.id_produto)
where codproduto = new.id_produto;
END $$
```

8.2.9 Gatilho que atualiza o valor total de um item do pedido quando um item do pedido for alterado.

```
DELIMITER $$
CREATE TRIGGER t_atualizacaoitemdopedido3 AFTER UPDATE ON itemdopedido
FOR EACH ROW
BEGIN
UPDATE itemdopedido SET valortotalitem = NEW.preco * NEW.qntdsolicitada where coditem = new.coditem;
END $$
```