Faculdade de Tecnologia
Universidade Estadual de Campinas

Sistema de controle de Pessoas, Estoque, Vendas e Pedidos para o estabelecimento mini mercado Empório Nações

Breno Martins

Jéssica Pereira

Matheus Evers

Renan Inoue

Tales Abdo

Limeira

2016

Sumário

1. Escopo 1
2. Regras de Negócio e Observações Importantes 1
3. Lista de Comandos 3
4. Lista de Ações Executadas Automaticamente (gatilhos/triggers) 4
5. Projeto Conceitual da Base de Dados 5
6. Projeto Lógico da Base de Dados 6
7. Projeto Físico da Base de Dados 7
Criação do Banco de Dados e suas Relações 7
Índices 9
8. Manipulação da Base de Dados 10
Stored Procedures de Cadastro 10
Stored Procedures de Atualização 14
Stored Procedures de Exclusão 18
Stored Procedures de Consulta por Código 19
Stored Procedures de Consulta por Outras Colunas 20
Triggers 23

1. Escopo

O estabelecimento Empório Nações é um mini mercado que tem como público alvo os moradores do bairro em que está inserido ou de bairros vizinhos. O fluxo varia de acordo com o dia da semana, porém é maior durante os fim de semana. Uma boa parte dos clientes vão ao estabelecimento diariamente para fazer compras. O estoque do mini mercado é abastecido por fornecedores que vão a cada 15 dias oferecer produtos ou através de pedidos diretos às empresas. Normalmente, há sempre dois funcionários trabalhando no estabelecimento, podendo ter horários de trabalho diferentes.

Este trabalho e toda a sua parte de manipulação de dados abrange as funcionalidades necessárias para o armazenamento de dados importantes do estabelecimento, permitindo o CRUD (Create, Read, Update and Delete ou Criar, Ler, Atualizar e Excluir) de várias partes do sistema.

É possível manipular dados de pessoas como funcionários, fornecedores e clientes, compras, pedidos e também criar relatórios contendo informações sobre as compras e pedidos de um determinado período para um melhor controle das finanças. As compras são as vendas para os clientes, que podem aliás fazer várias compras e pagar no seu dia definido. Os pedidos são produtos e mercadorias que o estabelecimento solicita aos fornecedores.

2. Regras de Negócio e Observações Importantes

Foi adotada uma padronização de nomenclatura no projeto. Apenas as tabelas têm letras maiúsculas em seu nome, sendo no início de cada palavra que compõe o nome. O restante do sistema possui nomes apenas com letras minúsculas. É importante notar que todos os nomes foram projetados para indicar exatamente o que aquele trecho de código realiza, para evitar confusões e facilitar a manutenção do mesmo.

Pensando na armazenagem, principalmente a longo prazo, todas as colunas foram devidamente analisadas, para que pudessem suprir a necessidade do sistema sem exigir muito espaço. As chaves primárias das relações são em sua maioria *smallint*, permitindo mais de 30 mil tuplas. Considerando o fluxo de clientes, fornecedores, relatórios, etc., esta quantidade está mais que adequada para vários anos de uso. Já as relações “Pedido” e “Compra”, assim como seus itens, possuem a chave primária do tipo *int*, pois terão uma quantidade consideravelmente maior de tuplas.

As colunas “nome” nas relações “Pessoa” e “Produto” estão com os seus tipos definidos como *varchar* para possibilitar a consulta por nome. Se fosse do tipo *char*, não seria possível procurar nenhum nome, pois colunas deste tipo preenchem a parte não usada com espaços.

As colunas “rg”, “cpf” e “cnpj” foram declaradas como chaves secundárias (*Unique*). Isso foi feito pois mesmo que estas colunas não sejam chaves primárias, elas não devem ter tuplas com valores repetidos.

As colunas que armazenam texto em sua maioria foram declaradas como *char* para aumentar o desempenho de buscas de tuplas pelo SGBD, tornando o uso do sistema

mais rápido e eficiente. Algumas colunas como “telefone” da relação “Pessoa” e “cnpj” da relação “Fornecedor” foram devidamente analisadas e definidas para permitir o armazenamento correto dessas informações que provêm do mundo real, sem deixar sobras de espaços desnecessários.

Não é possível alterar o código das chaves primárias das relações para evitar conflitos envolvendo chaves estrangeiras.

Para evitar possíveis erros e facilitar a vida do usuário, as colunas que fornecem a chave primária das relações estão como *identity*, ou seja, o próprio SGBD se encarrega deste campo e o preenche com números inteiros, de 1 até o limite do tipo de valor. Há uma exceção para isto que se encontra na relação “Produto”. Foi analisada a situação desta relação e foi decidido que a melhor opção é permitir que os usuários do sistema escolham o valor desta coluna para cada tupla.

As relações especializadas vindas da relação “Pessoa” compartilham a mesma chave primária. A especialização tem participação total e exclusão mútua, ou seja, só é possível ser uma das três opções (Funcionário, Fornecedor e Cliente) ao mesmo tempo e além disso não há tuplas na relação “Pessoa” que não se liguem com alguma tupla das três relações especializadas.

É possível realizar uma compra mesmo que o estoque do Produto esteja zerado ou negativo no sistema. Isso foi permitido pois é possível que haja estoque do Produto fisicamente e ainda não tenha sido inserido através de um Pedido.

Só é possível ter uma vez um Produto nos itens de uma Compra/Pedido, ou seja, um Produto só pode aparecer uma vez por Compra/Pedido. Caso precise de mais daquele Produto em determinada Compra/Pedido, basta alterar a quantidade do Produto no item.

Para evitar erros de execução e mal funcionamento dos triggers, todas as Stored Procedures manipulam no máximo uma tupla por vez quando a tupla manipulada é de uma relação que possui triggers. Isto foi feito pois os triggers aceitam apenas uma modificação por vez.

A Stored Procedure “cadastraritemdacompra”, usada para cadastrar um item de uma compra, possui uma condição a mais que as outras Stored Procedures. Foi necessário usar “If (@@trancount > 0) Commit TRAN” para encerrar as transações desta Stored procedure. O funcionamento ocorre como deveria e todas as atualizações/inserções são devidamente realizadas.

Alguns comandos/triggers não foram implementados ou foram limitados pelo seguinte motivo: integridade e consistência dos dados armazenados. Algumas relações, como “Compra”, armazenam em suas tuplas dados providos de outras relações, como a coluna “codcliente”, que é uma coluna que vem da relação “Cliente”. Por este motivo, não é possível remover uma tupla da relação “Cliente”, pois o seu campo da coluna “codcliente” está sendo usado pela relação “Compra”. Está situação se repete mais algumas vezes pelo sistema, por isso foi decidido que pela boa integridade, consistência

e recuperação dos dados, os comandos de remoção e atualização não fossem criados ou fossem limitados para evitar problemas futuros.

O sistema tem atualizações automáticas de várias partes devido aos triggers. Inclusão, atualização e exclusão de Pedidos e Compras atualizam os Relatórios. Inclusão, atualização e exclusão de Itens do Pedido ou de Itens da Compra atualizam devidamente o estoque do Produto e o valor total de cada Item, além do valor total do Pedido/Compra. Os triggers são melhores detalhados nos tópicos 4 e 8.

Caso for preciso é possível incrementar o sistema de forma que for julgada necessária. Para isso, basta criar novas Stored Procedures ou Triggers de acordo com o que foi pedido. Qualquer pedido será avaliado e feito assim que possível.

3. Lista de Comandos

É possível realizar os seguintes comandos através de execuções de Stored Procedures:

- Cadastrar Pessoa
- Cadastrar Funcionário
- Cadastrar Fornecedor
- Cadastrar Cliente
- Cadastrar Produto
- Cadastrar Pedido
- Cadastrar Item Do Pedido
- Cadastrar Compra
- Cadastrar Item Da Compra
- Cadastrar Relatório
- Atualizar Funcionário
- Atualizar Fornecedor
- Atualizar Cliente
- Atualizar Produto
- Atualizar Pedido
- Atualizar Item Do Pedido
- Atualizar Compra
- Atualizar Item Da Compra
- Atualizar Status de Compras
- Atualizar Relatório
- Excluir Item Do Pedido
- Excluir Item Da Compra
- Excluir Relatório
- Consultar Funcionário por código
- Consultar Fornecedor por código
- Consultar Cliente por código
- Consultar Produto por código
- Consultar Pedido por código

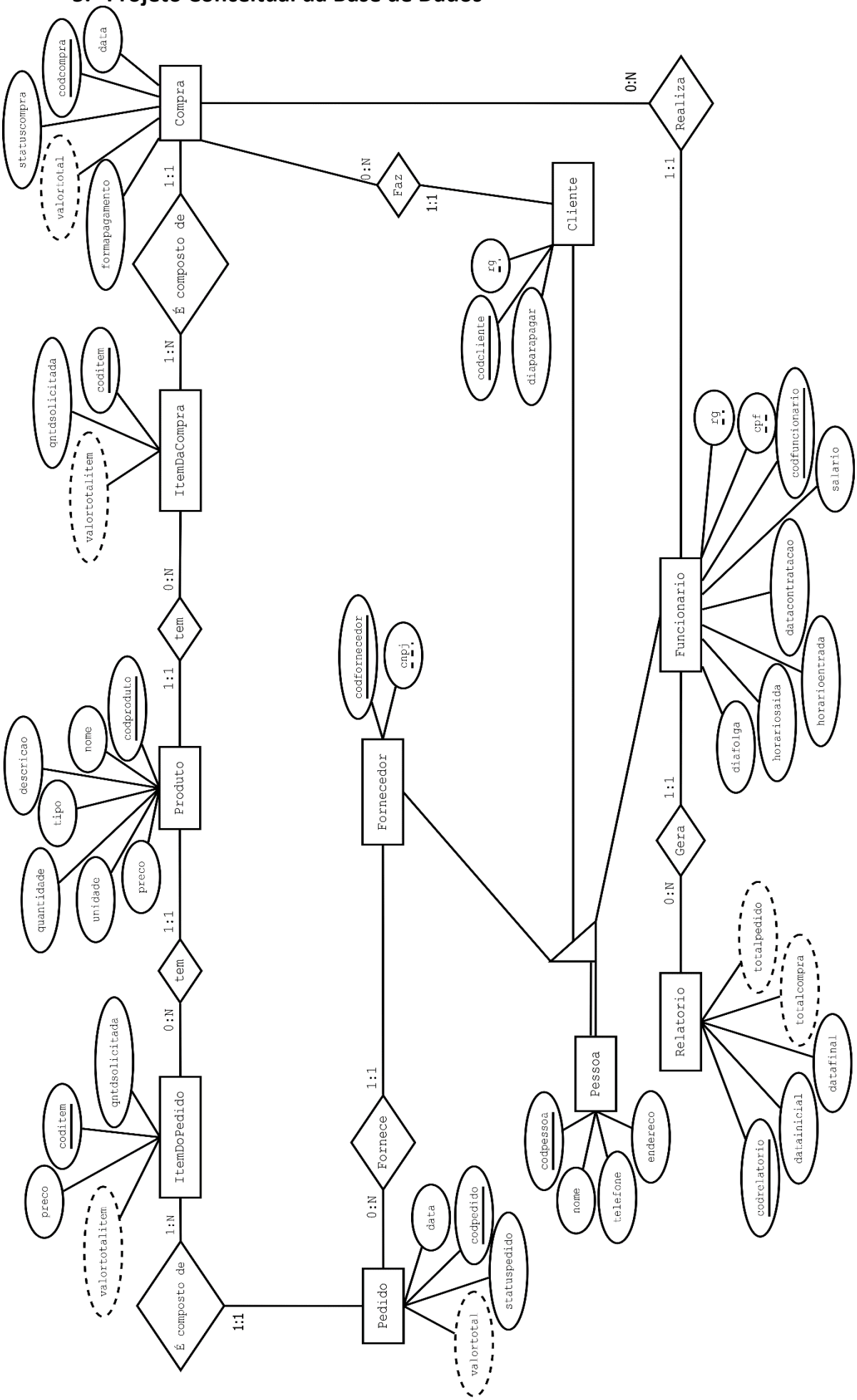
- Consultar Item Do Pedido por código
- Consultar Compra por código
- Consultar Item Da Compra por código
- Consultar Relatório por código
- Consultar Funcionário por nome
- Consultar Fornecedor por nome
- Consultar Cliente por nome
- Consultar Funcionário por data de contratação
- Consultar Produto por nome
- Consultar Produto por preço
- Consultar Pedidos acima de um valor
- Consultar Compras acima de um valor
- Consultar todos os Itens de um Pedido
- Consultar todos os itens de uma Compra
- Consultar todos os Itens Do Pedido por produto
- Consultar todos os Itens Da Compra por produto
- Consultar Pedidos feitos em um dia
- Consultar Compras feitas em um dia

4. Lista de Ações Executadas Automaticamente (gatilhos/triggers)

Estas ações são realizadas por efeito cascata, ou seja, quando um comando específico acontece em uma determinada relação, estes gatilhos(triggers) são acionados e realizam outros comandos:

- Atualizar o “valortotalitem” da relação “Itemdopedido” e “valortotal” da relação "Pedido" quando um “Item Do Pedido” é criado
- Atualizar o “valortotalitem” da relação “Itemdacompra’ e “valortotal” da relação "Compra" quando um “Item Da Compra” é criado
- Atualizar a “quantidade” da relação "Produto", “valortotalitem” da relação “Itemdopedido” e “valortotal” da relação "Pedido" quando um “Item Do Pedido” é excluído
- Atualizar a “quantidade” da relação “Produto”, “valortotalitem” da relação “Itemdacompra” e “valortotal” da relação "Compra" quando um “Item Da Compra” é excluído
- Atualizar a "quantidade” da relação “Produto”, “valortotalitem” da relação “Itemdopedido” e “valortotal” da relação "Pedido" quando um “Item Do Pedido” é atualizado
- Atualizar a “quantidade” da relação "Produto", “valortotalitem” da relação “Itemdacompra” e “valortotal” da relação "Compra" quando um “Item Da Compra” é atualizado
- Atualizar o “totalpedido” da relação “Relatório” quando uma tupla da relação "Pedido" for criada, atualizada ou excluída
- Atualizar o “totalcompra” da relação “Relatório” quando uma tupla da relação "Compra" for criada, atualizada ou excluída

5. Projeto Conceitual da Base de Dados



6. Projeto Lógico da Base de Dados

Pessoa (codpessoa, nome, telefone, endereco)

Funcionario (codfuncionario, cpf, rg, salario, datacontratacao, horarioentrada, horariosaida, diafolga)

codfuncionario - chave estrangeira referenciando Pessoa

Fornecedor (codfornecedor, cnpj)

codfornecedor - chave estrangeira referenciando Pessoa

Cliente (codcliente, rg, diaparapagar)

codcliente - chave estrangeira referenciando Pessoa

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

Pedido (codpedido, codfornecedor, data, valortotal, statuspedido)

codfornecedor - chave estrangeira referenciando Fornecedor

ItemDoPedido (coditem, codproduto, codpedido, qntdsolicitada, preco, valortotalitem)

codproduto - chave estrangeira referenciando Produto

codpedido - chave estrangeira referenciando Pedido

Compra (codcompra, codfuncionario, codcliente, data, valortotal, formadepagamento, statuscompra)

codfuncionario - chave estrangeira referenciando Funcionário

codcliente - chave estrangeira referenciando Cliente

ItemDaCompra (coditem, codproduto, codcompra, qntdsolicitada, valortotalitem)

codproduto - chave estrangeira referenciando Produto

codcompra - chave estrangeira referenciando Compra

Relatorio (codrelatorio, codfuncionario, datainicial, datafinal, totalpedido, totalcompra)

codfuncionario – chave estrangeira referenciando Funcionario

7. Projeto Físico da Base de Dados

```
create database Trabalho

use Trabalho

--Relação contendo todas as colunas comuns as pessoas especializadas
create table Pessoa
(
	codpessoa smallint NOT NULL identity,
	nome varchar(40),
	telefone char(18),
	endereco char(60),
	primary key(codpessoa)
)

create table Funcionario
(
	codfuncionario smallint NOT NULL,
	cpf char(14),
	rg char(12),
	salario numeric(7,2),
	datacontratacao date,
	horarioentrada time,
	horariosaida time,
	diafolga char(7),
	primary key(codfuncionario),
	unique(cpf),
	unique(rg),
	foreign key(codfuncionario) references Pessoa
)

create table Fornecedor
(
	codfornecedor smallint NOT NULL,
	cnpj char(18),
	primary key(codfornecedor),
	unique(cnpj),
	foreign key(codfornecedor) references Pessoa
)

create table Cliente
(
	codcliente smallint NOT NULL,
	rg char(12),
	diaparapagar tinyint,
	primary key(codcliente),
	unique(rg),
	foreign key(codcliente) references Pessoa
)
```

```sql
create table Produto
(
	codproduto smallint NOT NULL,
	nome varchar(40),
	descricao char(60),
	tipo char(20),
	quantidade numeric(6,3),
	unidade char(2),
	preco numeric(5,2),
	primary key(codproduto)
)

--Um pedido contém vários itens do pedido
create table Pedido
(
	codpedido int NOT NULL identity,
	codfornecedor smallint,
	data date,
	valortotal numeric(6,2),
	statuspedido char(8) default 'pendente',
	primary key(codpedido),
	foreign key(codfornecedor) references Fornecedor
)

create table ItemDoPedido
(
	coditem int NOT NULL identity,
	codproduto smallint,
	codpedido int,
	qntdsolicitada numeric(6,3),
	preco numeric(5,2),
	valortotalitem numeric(6,2)
	primary key(coditem,codproduto),
	foreign key(codproduto) references Produto,
	foreign key(codpedido) references Pedido
)

--Uma compra contém vários itens da compra
create table Compra
(
	codcompra int NOT NULL identity,
	codfuncionario smallint,
	codcliente smallint,
	data date,
	valortotal numeric(6,2),
	formapagamento char(8),
	statuscompra char(8) default 'pendente',
	primary key(codcompra),
	foreign key(codfuncionario) references Funcionario,
	foreign key(codcliente) references Cliente
)
```

```
create table ItemDaCompra
(
	coditem int NOT NULL identity,
	codproduto smallint,
	codcompra int,
	qntdsolicitada numeric(6,3),
	valortotalitem numeric(6,2)
	primary key(coditem,codproduto),
	foreign key(codproduto) references Produto,
	foreign key(codcompra) references Compra
)

--Esta relação utiliza a soma dos pedidos e compras para compor
algumas de suas colunas
create table Relatorio
(
	codrelatorio smallint NOT NULL identity,
	codfuncionario smallint,
	datainicial date,
	datafinal date,
	totalpedido numeric(8,2),
	totalcompra numeric(8,2)
	primary key(codrelatorio),
	foreign key(codfuncionario) references Funcionario
)

--Todos os índices das chaves estrangeiras que não são chaves
primárias
create index index_p_codfornecedor on Pedido (codfornecedor)
create index index_idp_codproduto on ItemDoPedido (codproduto)
create index index_idp_codpedido on ItemDoPedido (codpedido)
create index index_c_codfuncionario on Compra (codfuncionario)
create index index_c_codcliente on Compra (codcliente)
create index index_idc_codproduto on ItemDaCompra (codproduto)
create index index_idc_codcompra on ItemDaCompra (codcompra)
create index index_r_codfuncionario on Relatorio (codfuncionario)
```

8. Manipulação da Base de Dados

```
--Cadastrar uma pessoa. Como o codpessoa está sendo gerado automaticamente, o
return retorna o código de maior valor, ou seja, o último código cadastrado. Isso
permite a integridade entre a relação genérica(Pessoa) e as especializadas
(Funcionário, Fornecedor e Cliente)
create procedure cadastrarpessoa
@nome varchar(40),
@telefone char(18),
@endereco char(60)
as
	begin transaction
		insert into Pessoa
		values (@nome, @telefone, @endereco)
		if  (@@rowcount > 0)  --Inserção de pessoa bem sucedida
		begin
			commit transaction
			return (select max(codpessoa) from Pessoa)
		end
		else
		begin
			rollback transaction
			print 'Algum valor inserido é inválido. Verifique os valores
				digitados.'
			return 0
		end


--Cadastrar um funcionário. Para isso chama a SP "cadastrarpessoa"
create procedure cadastrarfuncionario
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@cpf char(14),
@rg char(12),
@salario numeric(7,2),
@datacontratacao date,
@horarioentrada time,
@horariosaida time,
@diafolga char(7)
as
	declare @retorno int
	begin transaction
		exec @retorno = cadastrarpessoa @nome, @telefone, @endereco
		if (@retorno > 0) --Inserção de pessoa bem sucedida
		begin
			insert into Funcionario
			values (@retorno, @cpf, @rg, @salario, @datacontratacao,
@horarioentrada, @horariosaida, @diafolga)
			if (@@rowcount > 0) --Inserção de funcionário bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print 'Algum valor inserido é inválido. Verifique os
valores digitados e confira se este RG e CPF já estão cadastrados.'
			end
		end
		else
			rollback transaction
```

```
--Cadastrar um fornecedor. Para isso chama a SP "cadastrarpessoa"
create procedure cadastrarfornecedor
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@cnpj char(18)
as
	declare @retorno int
	begin transaction
		exec @retorno = cadastrarpessoa @nome, @telefone, @endereco
		if (@retorno > 0) --Inserção de pessoa bem sucedida
		begin
			insert into Fornecedor
			values (@retorno, @cnpj)
			if  (@@rowcount > 0) --Inserção de fornecedor bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados e confira se este CNPJ já
					está cadastrado.'
			end
		end
		else
			rollback transaction


--Cadastrar um cliente. Para isso chama a SP "cadastrarpessoa"
create procedure cadastrarcliente
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@rg char(12),
@diaparapagar tinyint
as
	declare @retorno int
	begin transaction
		exec @retorno = cadastrarpessoa @nome, @telefone, @endereco
		if (@retorno > 0) --Inserção de pessoa bem sucedida
		begin
			insert into Cliente
			values (@retorno, @rg, @diaparapagar)
			if  (@@rowcount > 0) --Inserção de cliente bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados e confira se este RG já está
					cadastrado.'
			end
		end
		else
			rollback transaction
```

```
--Cadastrar um produto
create procedure cadastrarproduto
@codproduto smallint,
@nome varchar(40),
@descricao char(60),
@tipo char(20),
@quantidade numeric(6,3),
@unidade char(2),
@preco numeric(5,2)
as
	insert into Produto
	values (@codproduto, @nome, @descricao, @tipo, @quantidade, @unidade,
		 @preco)
	if  (@@rowcount = 0)  --Inserção de produto mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Cadastrar um pedido
create procedure cadastrarpedido
@codfornecedor smallint,
@data date
as
	insert into Pedido
	values (@codfornecedor, @data, 0, 'pendente')
	if  (@@rowcount = 0)  --Inserção de pedido mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Cadastrar um item do pedido e automaticamente incrementar o estoque
create procedure cadastraritemdopedido
@codproduto smallint,
@codpedido int,
@qntdsolicitada numeric(6,3),
@preco numeric(5,2)
as
	begin transaction
		insert into ItemDoPedido
		values (@codproduto, @codpedido, @qntdsolicitada, @preco,
			@qntdsolicitada*@preco)
		if  (@@rowcount > 0)  -- Inserção de item do pedido bem sucedida
		begin
			update Produto
			set quantidade = quantidade+@qntdsolicitada
			where codproduto=@codproduto
			if (@@rowcount > 0) --Incremento bem sucedido
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados.'
			end
		end
		else
		begin
			rollback transaction
			print  'Algum valor inserido é inválido. Verifique os valores
				digitados.'
		end
```

```
--Cadastrar uma compra
create procedure cadastrarcompra
@codfuncionario smallint,
@codcliente smallint,
@data date,
@formapagamento char(8),
@statuscompra char(8)
as
	insert into Compra
	values (@codfuncionario, @codcliente, @data, 0, @formapagamento,
		@statuscompra)
	if  (@@rowcount = 0)  --Inserção de compra mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Cadastrar um item da compra e automaticamente decrementar o estoque
create procedure cadastraritemdacompra
@codproduto smallint,
@codcompra int,
@qntdsolicitada numeric(6,3)
as
	begin transaction
		insert into ItemDaCompra
		values (@codproduto, @codcompra, @qntdsolicitada,
			@qntdsolicitada*(select preco from Produto
					where codproduto=@codproduto))
		if  (@@rowcount > 0)  --Inserção de item da compra bem sucedida
		begin
			update Produto
			set quantidade = quantidade-@qntdsolicitada
			where codproduto=@codproduto
			if (@@rowcount > 0) --Decremento bem sucedido
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido.
					Verifique os valores digitados.'
			end
		end
		else
		begin
			rollback transaction
			print  'Algum valor inserido é inválido. Verifique os
				valores digitados.'
		end
		if (@@trancount > 0)
			commit tran --isso garante que todas as transações sejam
					fechadas. É importante notar que não modifica
					o funcionamento das Procedures e Triggers
```

```
--Cadastrar um relatório, que contém a soma dos pedidos e compras de um
determinado período fornecido pelo funcionário
create procedure cadastrarrelatorio
@codfuncionario smallint,
@datainicial date,
@datafinal date
as
	insert into Relatorio
	values (@codfuncionario, @datainicial, @datafinal,
		(select sum(valortotal) from Pedido where data between @datainicial
		and @datafinal),
		(select sum(valortotal) from Compra where data between @datainicial
		and @datafinal))
	if (@@rowcount = 0) --Inserção de relatório mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores.'


--Atualizar os dados de um funcionário e de sua respectiva pessoa
create procedure atualizarfuncionario
@codfuncionario smallint,
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@cpf char(14),
@rg char(12),
@salario numeric(7,2),
@datacontratacao date,
@horarioentrada time,
@horariosaida time,
@diafolga char(7)
as
	begin transaction
		update Funcionario
		set cpf=@cpf, rg=@rg, salario=@salario,
		    datacontratacao=@datacontratacao,
		    horarioentrada=@horarioentrada, @horariosaida=@horariosaida,
		    diafolga=@diafolga
		where codfuncionario=@codfuncionario
		if  (@@rowcount > 0)  --Atualização do funcionário bem sucedida
		begin
			update Pessoa
			set nome=@nome, telefone=@telefone, endereco=@endereco
			where codpessoa=@codfuncionario
			if (@@rowcount > 0) --Atualização do fornecedor bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados.'
			end
		end
		else
		begin
			rollback transaction
			print  'Algum valor inserido é inválido. Verifique os valores
				digitados e confira se este RG e CPF já estão
				cadastrados.'
		end
```

```
--Atualizar os dados de um fornecedor e de sua respectiva pessoa
create procedure atualizarfornecedor
@codfornecedor smallint,
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@cnpj char(18)
as
	begin transaction
		update Fornecedor set cnpj=@cnpj
		where codfornecedor=@codfornecedor
		if  (@@rowcount > 0)  --Atualização do fornecedor bem sucedida
		begin
			update Pessoa
			set nome=@nome, telefone=@telefone, endereco=@endereco
			where codpessoa=@codfornecedor
			if (@@rowcount > 0) --Atualização da pessoa bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados.'
			end
		end
		else
		begin
			rollback transaction
			print  'Algum valor inserido é inválido. Verifique os valores
				digitados e confira se este CNPJ já está cadastrado.'
		end

--Atualizar os dados de um funcionário e de sua respectiva pessoa
create procedure atualizarcliente
@codcliente smallint,
@nome varchar(40),
@telefone char(18),
@endereco char(60),
@rg char(12),
@diaparapagar tinyint
as
	begin transaction
		update Cliente
		set rg=@rg, diaparapagar=@diaparapagar
		where codcliente=@codcliente
		if  (@@rowcount > 0)  --Atualização do funcionário bem sucedida
		begin
			update Pessoa
			set nome=@nome, telefone=@telefone, endereco=@endereco
			where codpessoa=@codcliente
			if (@@rowcount > 0) –Atualização da pessoa bem sucedida
				commit transaction
			else
			begin
				rollback transaction
				print  'Algum valor inserido é inválido. Verifique os
					valores digitados.'
			end
		end
		else
		begin
			rollback transaction
```

```
                print  'Algum valor inserido é inválido. Verifique os valores
                       digitados e confira se este RG já está cadastrado.'
        end

--Atualizar os dados de um produto, assim como os valores dos itens da compra que
possuem tal produto e as compras que possuem tais itens
create procedure atualizarproduto
@codproduto smallint,
@nome varchar(40),
@descricao char(60),
@tipo char(20),
@unidade char(2),
@preco numeric(5,2)
as
    begin transaction
        update produto
        set nome=@nome, descricao=@descricao, tipo=@tipo, unidade=@unidade,
            preco=@preco
        where codproduto=@codproduto
        if (@@rowcount>0) -Atualização do produto bem sucedida
        begin
            update ItemDaCompra
            set valortotalitem=@preco*qntdsolicitada
            where codproduto=@codproduto
            if (@@rowcount > 0) --Atualização dos itens bem sucedida
            begin
                update compra
                set valortotal=(select sum (p.preco*i.qntdsolicitada)
                                from produto p, ItemDaCompra i,
                                     compra c
                                where c.codcompra = i.codcompra
                                and p.codproduto = i.codproduto)
                where codcompra = (select i.codcompra
                                   from produto p
                                   inner join ItemDaCompra i
                                   on p.codproduto=i.codproduto
                                   inner join compra c
                                   on c.codcompra=i.codcompra
                                   and p.codproduto=@codproduto)
                if  (@@rowcount > 0)  --Atualização das compras bem
                                        sucedida
                    commit transaction
                else
                begin
                    rollback transaction
                    print  'Algum valor inserido é inválido.
                           Verifique os valores digitados.'
                end
            end
            else
            begin
                rollback transaction
                print  'Algum valor inserido é inválido. Verifique os
                       valores digitados.'
            end
        end
        else
        begin
            rollback transaction
            print  'Algum valor inserido é inválido. Verifique os valores
                   digitados.'
        end
```

```
--Atualizar os dados básicos de um pedido
create procedure atualizarpedido
@codpedido int,
@codfornecedor smallint,
@data date,
@statuspedido char(8)
as
	update Pedido
	set codfornecedor=@codfornecedor, data=@data,statuspedido=@statuspedido
	where codpedido=@codpedido
	if  (@@rowcount = 0)  --Atualização mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Atualizar o valor de "qntdsolicita" e "preco" de um item do pedido
create procedure atualizaritemdopedido
@coditem int,
@qntdsolicitada numeric(6,3),
@preco numeric(5,2)
as
	update ItemDoPedido
	set qntdsolicitada=@qntdsolicitada, preco=@preco
	where coditem=@coditem
	if (@@rowcount = 0) --Atualização mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Atualizar os dados básicos de uma compra
create procedure atualizarcompra
@codcompra int,
@codfuncionario smallint,
@codcliente smallint,
@data date,
@formapagamento char(8),
@statuscompra char(8)
as
	update Compra
	set codfuncionario=@codfuncionario, codcliente=@codcliente,
	    data=@data, formapagamento=@formapagamento,
	    statuscompra=@statuscompra
	where codcompra=@codcompra
	if  (@@rowcount = 0)  --Atualização mal sucedida
		print  'Algum valor inserido é inválido. Verifique os valores
			digitados.'


--Atualizar o valor de "qntdsolicita" de um item da compra
create procedure atualizaritemdacompra
@coditem int,
@qntdsolicitada numeric(6,3)
as
	update ItemDaCompra
	set qntdsolicitada=@qntdsolicitada
	where coditem=@coditem
	if (@@rowcount = 0) --Atualização mal sucedida
		print 'Algum valor inserido é inválido. Verifique os valores
			digitados.'
```

```
--Atualizar o status das compras “pendente” para “paga” quando chega o dia do mês
para pagar do cliente.
create procedure atualizarstatuscompra
@codcliente smallint
as
	update Compra
	set statuscompra='paga'
	where codcliente=@codcliente
	if (@@rowcount = 0) --Atualização mal sucedida
		print  'Não há este código de cliente cadastrado.'


--Atualizar o funcionário e as datas de um relatório, assim como os valores
totais dos pedidos e das compras
create procedure atualizarrelatorio
@codrelatorio smallint,
@codfuncionario smallint,
@datainicial date,
@datafinal date
as
	update Relatorio
	set codfuncionario=@codfuncionario, datainicial=@datainicial,
	    datafinal=@datafinal,
	    totalpedido=(select sum(valortotal) from Pedido
	                 where data between @datainicial and @datafinal),
	    totalcompra=(select sum(valortotal) from Compra
	                 where data between @datainicial and @datafinal)
	where codrelatorio=@codrelatorio
	if (@@rowcount = 0) --Atualização mal sucedida
		print  'Não há este código de relatório cadastrado.'


--Excluir um item do pedido com base no valor de "coditem" fornecido
create procedure excluiritemdopedido
@coditem int
as
	delete from ItemDoPedido
	where coditem=@coditem
	if (@@rowcount = 0) --Exclusão mal sucedida
		print  'Não há este código de item do pedido cadastrado.'


--Excluir um item da compra com base no valor de "coditem" fornecido
create procedure excluiritemdacompra
@coditem int
as
	delete from ItemDaCompra
	where coditem=@coditem
	if (@@rowcount = 0) --Exclusão mal sucedida
		print  'Não há este código de item da compra cadastrado.'


--Excluir um relatório
create procedure excluirrelatorio
@codrelatorio smallint
as
	delete from Relatorio
	where codrelatorio=@codrelatorio
	if (@@rowcount = 0) --Exclusão mal sucedida
		print  'Não há este código de relatório cadastrado.'
```

```
--Consultar um funcionário com base no código fornecido
create procedure consultarfuncionario
@codfuncionario smallint
as
	select * from Funcionario inner join Pessoa
	on codfuncionario=codpessoa
	where codfuncionario=@codfuncionario
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de funcionário cadastrado.'


--Consultar um fornecedor com base no código fornecido
create procedure consultarfornecedor
@codfornecedor smallint
as
	select * from Fornecedor inner join Pessoa
	on codfornecedor=codpessoa
	where codfornecedor=@codfornecedor
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de fornecedor cadastrado.'


--Consultar um cliente com base no código fornecido
create procedure consultarcliente
@codcliente smallint
as
	select * from Cliente inner join Pessoa
	on codcliente=codpessoa
	where codcliente=@codcliente
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de cliente cadastrado.'

--Consultar um produto com base no código fornecido
create procedure consultarproduto
@codproduto smallint
as
	select * from Produto
	where codproduto=@codproduto
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de produto cadastrado.'

--Consultar um pedido com base no código fornecido
create procedure consultarpedido
@codpedido int
as
	select * from Pedido
	where codpedido=@codpedido
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de pedido cadastrado.'


--Consultar um item do pedido com base no código fornecido
create procedure consultaritemdopedido
@coditem int
as
	select * from Itemdopedido
	where coditem=@coditem
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de item do pedido cadastrado.'
```

```
--Consultar uma compra com base no código fornecido
create procedure consultarcompra
@codcompra int
as
	select * from Compra
	where codcompra=@codcompra
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de compra cadastrado.'


--Consultar um item da compra com base no código fornecido
create procedure consultaritemdacompra
@coditem int
as
	select * from Itemdacompra
	where coditem=@coditem
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de item da compra cadastrado.'


--Consultar um relatório com base no código fornecido
create procedure consultarrelatorio
@codrelatorio smallint
as
	select * from Relatorio
	where codrelatorio=@codrelatorio
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de relatório cadastrado.'


--Consultar todos os funcionários que tenham o nome ou parte dele igual a parte
fornecida
create procedure consultarfuncionariopornome
@nome varchar(40)
as
	select * from Funcionario inner join Pessoa
	on codfuncionario=codpessoa
	where nome like '%'+@nome+'%'
	order by nome asc --Ordena os resultados em ordem alfabética
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há funcionários cadastrados com este nome ou parte de
			nome.'


--Consultar todos os fornecedores que tenham o nome ou parte dele igual a parte
fornecida
create procedure consultarfornecedorpornome
@nome varchar(40)
as
	select * from Fornecedor inner join Pessoa
	on codfornecedor=codpessoa
	where nome like '%'+@nome+'%'
	order by nome asc --Ordena os resultados em ordem alfabética
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há fornecedores cadastrados com este nome ou parte de
			nome.'
```

```
--Consultar todos os clientes que tenham o nome ou parte dele igual a parte
fornecida
create procedure consultarclientepornome
@nome varchar(40)
as
	select * from Cliente inner join Pessoa
	on codcliente=codpessoa
	where nome like '%'+@nome+'%'
	order by nome asc --Ordena os resultados em ordem alfabética
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há clientes cadastrados com este nome ou parte de
			nome.'


--Consultar todos os funcionários contratados após certa data
create procedure consultarfuncionariopordatacontratacao
@datacontratacao date
as
	select * from Pessoa
	inner join Funcionario
	on codfuncionario=codpessoa
	where datacontratacao>=@datacontratacao
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há funcionários contratados após a data fornecida.'


--Consultar todos os produtos que tenham o nome ou parte dele igual a parte
fornecida
create procedure consultarprodutopornome
@nome varchar(40)
as
	select * from Produto
	where nome like '%'+@nome+'%'
	order by nome asc --Ordena os resultados em ordem alfabética
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há produtos cadastrado com este nome ou parte de nome.'


--Consultar todos os produtos com preço igual ou maior que o valor fornecido
create procedure consultarprodutoporpreco
@preco numeric(5,2)
as
	select * from Produto
	where preco>=@preco
	order by preco asc --Ordena os resultados do menor para o maior preço
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há produtos com preço maior que o inserido.'


--Consultar todos os pedidos com valor total igual ou maior que o valor fornecido
create procedure consultarpedidosporvalor
@valortotal numeric(6,2)
as
	select * from Pedido
	where valortotal>=@valortotal
	order by valortotal asc --Ordena os resultados do menor para o maior valor
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há pedidos com valor total maior ou igual ao inserido.'
```

```
--Consultar todas as compras com valor total igual ou maior que o valor fornecido
create procedure consultarcomprasporvalor
@valortotal numeric(6,2)
as
	select * from Compra
	where valortotal>=@valortotal
	order by valortotal asc --Ordena os resultados do menor para o maior valor
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há compras com valor total maior ou igual ao inserido.'


--Consultar todos os itens de um pedido para exibição. Útil para quando o usuário
deseja saber tudo que foi requisitado em um pedido
create procedure consultarpedidocompleto
@codpedido int
as
	select coditem,codproduto,qntdsolicitada,preco,valortotalitem
	from ItemDoPedido
	where codpedido=@codpedido
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de pedido cadastrado.'


--Consultar todos os itens de uma compra para exibição. Útil para quando o
usuário deseja saber tudo que foi requisitado em uma compra
create procedure consultarcompracompleta
@codcompra int
as
	select coditem,codproduto,qntdsolicitada,valortotalitem
	from ItemDaCompra
	where codcompra=@codcompra
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há este código de compra cadastrado.'


--Consultar todos os itens do pedido que possuem determinado produto
create procedure consultaritemdopedidoporproduto
@codproduto smallint
as
	select coditem,codpedido,qntdsolicitada,preco,valortotalitem
	from ItemDoPedido
	where codproduto=@codproduto
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há itens de pedido cadastrados com esse produto.'


--Consultar todos os itens da compra que possuem determinado produto
create procedure consultaritemdacompraporproduto
@codproduto smallint
as
	select coditem,codcompra,qntdsolicitada,valortotalitem
	from ItemDaCompra
	where codproduto=@codproduto
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há itens de compra cadastrados com esse produto.'
```

```
--Consultar todos os pedidos de uma data fornecida
create procedure consultarpedidosdeumdia
@data date
as
	select * from Pedido
	where data=@data
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há pedidos neste dia.'


--Consultar todas as compras de uma data fornecida
create procedure consultarcomprasdeumdia
@data date
as
	select * from Compra
	where data=@data
	if (@@rowcount = 0) --Não há resultados para esta consulta
		print  'Não há compras neste dia.'


--Atualizar o valor total de um pedido quando um item do pedido é criado
create trigger inclusaodeitemdopedido
on ItemDoPedido for insert
as
	update Pedido
	set valortotal = valortotal + (select qntdsolicitada * preco
					from inserted)
	where codpedido = (select codpedido from inserted)

--Atualizar o valor total de uma compra quando um item da compra é criado
create trigger inclusaodeitemdacompra
on ItemDaCompra for insert
as
	update Compra
	set valortotal = valortotal + (select i.qntdsolicitada * p.preco
					from produto p, inserted i
					where p.codproduto = i.codproduto)
	where codcompra = (select codcompra from inserted)

--Atualizar o valor total de um pedido e atualizar o estoque do produto quando um
item do pedido for excluído
create trigger exclusaoitemdopedido
on ItemDoPedido for delete
as
	begin transaction
		update Pedido
		set valortotal = valortotal - (select qntdsolicitada *preco
						from deleted)
		where codpedido = (select codpedido from deleted)
		if (@@rowcount > 0) --Atualização do pedido bem sucedida
		begin
			update Produto
			set quantidade = quantidade-(select qntdsolicitada
							from deleted)
			where codproduto= (select codproduto from deleted)
			if (@@rowcount > 0) --Atualização do produto bem sucedida
				commit transaction
			else
				rollback transaction
		end
		else
			rollback transaction
```

```
--Atualizar o valor total de uma compra e atualizar o estoque do produto quando
um item da compra for excluído
create trigger exclusaoitemdacompra
on ItemDaCompra for delete
as
	begin transaction
		update Compra
		set valortotal = valortotal - (select d.qntdsolicitada * p.preco
						from produto p, deleted d
						where p.codproduto = d.codproduto)
		where codcompra = (select codcompra from deleted)
		if (@@rowcount > 0) --Atualização da compra bem sucedida
		begin
			update Produto
			set quantidade = quantidade+(select qntdsolicitada
							from deleted)
			where codproduto= (select codproduto from deleted)
			if (@@rowcount > 0) --Atualização do produto bem sucedida
				commit transaction
			else
				rollback transaction
		end
		else
			rollback transaction

--Atualizar o valor total de um pedido, atualizar o estoque do produto e
atualizar o valor total do item do pedido quando um item do pedido for atualizado
create trigger atualizacaoitemdopedido
on ItemDoPedido for update
as
	begin transaction
		update pedido
		set valortotal = (select sum(i.preco*qntdsolicitada)
				from ItemDoPedido i,produto p, pedido ped
				where ped.codpedido=i.codpedido
				and p.codproduto=i.codproduto)
		where codpedido = (select codpedido from inserted)
		if (@@rowcount > 0) --Atualização do pedido bem sucedida
		begin
			update Produto
			set quantidade = quantidade + (select i.qntdsolicitada-
								d.qntdsolicitada
						from inserted i
						inner join deleted d
						on i.codproduto=d.codproduto)
			where codproduto= (select codproduto from inserted)
			if (@@rowcount > 0) --Atualização do produto bem sucedida
			begin
				update ItemDoPedido
				set valortotalitem = (select i.preco*i.qntdsolicitada
							from inserted i)
				where coditem = (select coditem from inserted)
				if (@@rowcount > 0) --Atualização do item bem sucedida
					commit transaction
				else
					rollback transaction
			end
			else
				rollback transaction
		end
		else
			rollback transaction
```

```
--Atualizar o valor total de uma compra, atualizar o estoque do produto e o valor
de total do item da compra quando um item da compra for alterado
create trigger atualizacaoitemdacompra
on ItemDaCompra for update
as
	begin transaction
		update Compra
		set valortotal = valortotal + (select p.preco * (i.qntdsolicitada -
                                                        d.qntdsolicitada)
                                       from produto p inner join inserted i
                                       on p.codproduto = i.codproduto
                                       inner join deleted d
                                       on i.codproduto = d.codproduto
                                       and i.codcompra = d.codcompra)
		where codcompra = (select codcompra from inserted)
		if (@@rowcount > 0) --Atualização da compra bem sucedida
		begin
			update Produto
			set quantidade = quantidade - (select i.qntdsolicitada-
                                                  d.qntdsolicitada
                                           from inserted i
                                           inner join deleted d
                                           on i.codproduto=d.codproduto)
			where codproduto= (select codproduto from inserted)
			if (@@rowcount > 0) --Atualização do produto bem sucedida
			begin
				update ItemDaCompra
				set valortotalitem = (select p.preco*i.qntdsolicitada
                                      from produto p
                                      inner join inserted i
                                      on p.codproduto=i.codproduto)
				where coditem = (select coditem from inserted)
				if (@@rowcount > 0) --Atualização do item bem sucedida
					commit transaction
				else
					rollback transaction
			end
			else
				rollback transaction
		end
		else
			rollback transaction



--Atualizar o valor total de pedido quando um pedido é
adicionado/alterado/excluído
create trigger atualizarrelatoriocompedido
on Pedido for insert, delete, update
as
	update Relatorio
	set totalpedido=(select sum(valortotal)
                     from Pedido p, Relatorio r
                     where p.data between r.datainicial
                     and r.datafinal)
	where datainicial >= (select data from inserted)
	and datafinal <= (select data from inserted)
```

```sql
--Atualizar o valor total de compra quando uma compra é
adicionada/alterada/excluída
create trigger atualizarrelatoriocomcompra
on Compra for insert, delete, update
as
	update Relatorio
	set totalcompra=(select sum(valortotal)
			from Compra c, Relatorio r
			where c.data between r.datainicial
			and r.datafinal)
	where datainicial >= (select data from inserted)
	and datafinal <= (select data from inserted)
```