FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Bases de Dados

Gestão de Supermercado

Professor João Pedro Carvalho Leal Mendes Moreira

2018/2019

Grupo 705

Liliana Almeida up201706908@fe.pu.pt

José Gomes up201707054@fe.up.pt

Francisco Rodrigues up201808909@fe.up.pt

domingo, 26 de maio de 2019

Descrição

Pretende-se guardar informação relativa a uma Cadeia de Supermercados.

De cada Loja é necessário armazenar a sua Morada, área (m^2) e telefone, bem como os seus clientes, funcionários e secções. De cada Secção pretende-se armazenar informação relativa ao seu nome, que Produtos lhe estão destinados e quais os funcionários que nela trabalham. Um funcionário tem um responsável e um funcionário responsável tem a seu comando vários funcionários. Cada loja tem um gestor.

Um funcionário realiza Encomendas pedidas por Clientes. Sobre o Cliente é necessário armazenar o seu nome, NIF e telefone e sobre a Encomenda é necessária informação sobre o seu preço total e produtos que a compõem (quantidade e desconto).

Um cliente pode ou não ter um Cartão de Cliente com determinado saldo e a esse Cartão estão associadas Promoções com certa data de início, data de fim e percentagem de promoção. A promoção pode ser usada como desconto direto ou não, sendo que, caso não seja utilizada como desconto direto, o saldo será acumulado no cartão. No ato do Pagamento, que tem como tipos possíveis “Dinheiro”, “Multibanco” ou “Cheque”, o cliente pode utilizar uma promoção, que esteja associada ao seu cartão, uma única vez. O saldo do cartão é sempre descontado na compra seguinte podendo cobrir o total da compra.

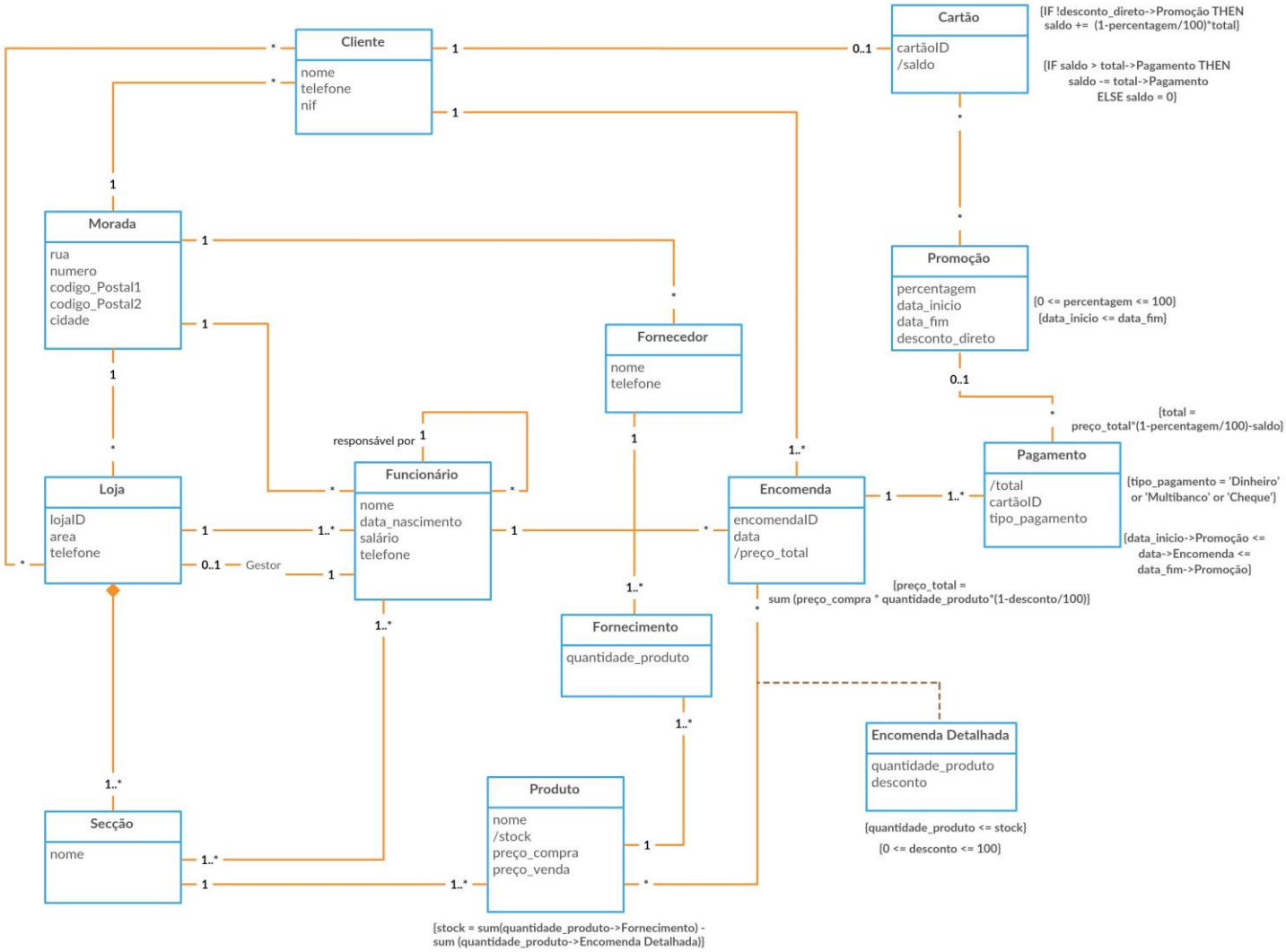
A Loja tem Fornecedores (que têm uma morada, nome e telefone) aos quais compra Produtos em determinada quantidade e mantêm registo do nome, stock, preço de compra e preço de venda desse mesmo produto.

Estrutura das classes

1. Loja
    a. lojaID
    b. area
    c. telefone
2. Secção
    a. nome
3. Funcionário
    a. nome
    b. data_nascimento
    c. salário
    d. telefone
4. Produto
    a. nome
    b. /stock
    c. preço_compra
    d. preço_venda
5. Fornecedor
    a. nome
    b. telefone
6. Fornecimento
    a. quantidade_produto
7. Cliente
    a. nome
    b. telefone
    c. nif
8. Cartão
    a. cartãoID
    b. /saldo
9. Encomenda
    a. encomendaID
    b. data
    c. /preço_total
10. Encomenda Detalhada
    a. quantidade_produto
    b. desconto
11. Promoção
    a. percentagem
    b. data_inicio
    c. data_fim

12. Pagamento
    a. /total
    b. cartãoID
    c. tipo_pagamento
13. Morada
    a. rua
    b. numero
    c. código_postal1
    d. código_postal2
    e. cidade

Diagrama de Classes – UML



Esquema Relacional e Dependências Funcionais

Morada (moradaID, rua, numero, codigo_postal1, codigo_postal2, cidade)

moradaID → rua, numero, codigo_postal1, codigo_postal2, cidade

codigo_postal1, codigo_postal2 → cidade, rua

cidade, rua, numero → codigo_postal1, codigo_postal2, moradaID

Loja (lojaID, area, telefone, moradaID->Morada)

lojaID → area, telefone, moradaID

telefone → lojaID, area, moradaID

Secção (secçãoID, nome, lojaID->Loja)

secçãoID → nome, lojaID

Funcionário (funcionárioID, nome, data_nascimento, salário, telefone, moradaID->Morada, lojaID->Loja)

funcionarioID → nome, data_nascimento, salário, telefone, moradaID, lojaID

telefone → funcionárioID, nome, data_nascimento, salário, moradaID, lojaID

Cliente (clienteID, nome, telefone, nif, moradaID->Morada)

clienteID → nome, telefone, nif, moradaID

nif → clienteID, nome, telefone, moradaID

telefone → clienteID, nome, nif, moradaID

Cartão (cartãoID, saldo, clienteID->Cliente)

cartãoID → saldo, clienteID

clienteID → cartãoID, saldo

Promoção (promoçãoID, percentagem, data_inicio, data_fim, desconto_direto)

promoçãoID → percentagem, data_inicio, data_fim, desconto_direto

Encomenda (encomendaID, data, preço_total, funcionárioID->Funcionário, clienteID->Cliente)

encomendaID → data, preço_total, funcionárioID, clienteID

Pagamento (pagamentoID, total, cartãoID->Cartão, tipo_de_pagamento, encomendaID-> Encomenda, promoção->Promoção)

pagamentoID → total, cartãoID, tipo_de_pagamento, encomendaID, promoção

encomendaID → pagamentoID, total, cartãoID, tipo_de_pagamento, promoção

Produto (produtoID, nome, stock, preço_compra, preço_venda, secçãoID->Secção)

produtoID → nome, stock, preço_compra, preço_venda, secçãoID

Fornecedor(fornecedorID, nome, telefone, moradaID->Morada)

fornecedorID → nome, telefone, moradaID

telefone → fornecedorID, nome, moradaID

Fornecimento (fornecimentoID, quantidade_produto, fornecedorID->Fornecedor, produtoID->Produto)

fornecimentoID → quantidade_produto, fornecedorID, produtoID

LojaCliente (lojaID->Loja, clienteID->Cliente)

FuncionárioSecção (funcionárioID->Funcionário, secçãoID->Secção)

EncomendaDetalhada(produtoID->Produto,encomendaID->Encomenda, quantidade_produto, desconto)

produtoID, encomendaID → quantidade_produto, desconto

CartãoPromoção (cartãoID->Cartão, promoçãoID->Promoção)

FuncionárioResponsável (funcionário->Funcionário, responsável->Funcionário)

funcionario → responsável

Gestor (loja->Loja , funcionário->Funcionário)

loja → funcionário

Formas Normais

As relações de uma base de dados devem seguir um conjunto de formas normais (1ª, 2ª, 3ª, *Boyce-Codd*, …) que permitem reduzir a redundância dos dados e melhorar a integridade dos mesmos. Neste capítulo, vamos apenas avaliar as relações definidas anteriormente quanto às duas últimas formas normais.

Uma relação está na **BCNF** (*Boyce-Codd Normal Form*) se as dependências funcionais que a caracterizam forem compostas, no seu lado esquerdo, por uma *superkey* dessa relação, ou seja, se o *closure* dos atributos da parte esquerda, das dependências funcionais, for igual a todos os atributos da relação.

No caso das relações descritas no capítulo anterior, apenas a relação **Morada** não cumpre este requisito, pois a seguinte dependência viola a BCNF:

codigo_postal1, codigo_postal2 → cidade, rua

Isto deve-se a: {codigo_postal1, codigo_postal2}$^{+}$ = {codigo_postal1, codigo_postal2, cidade, rua} que não possui todos os atributos da relação.

As restantes relações já estão todas na BCNF, podendo-se exemplificar a justificação para a relação **Cartão** que tem três atributos (cartãoID, saldo, clienteID):

{cartãoID}$^{+}$ = {cartãoID, saldo, clienteID}

{clienteID}$^{+}$ = {clienteID, cartãoID, saldo}

As relações também podem estar na **3ª Forma Normal** (3NF), se cumprirem a BCNF ou se a parte direita das suas dependências funcionais for composta apenas por atributos primos. Um atributo primo é um atributo que é membro de pelo menos uma chave da relação.

Assim, para a relação **Morada** apesar de esta não estar na BCNF, está na 3NF pois o conjunto {cidade, rua, número} é uma *superkey* da relação, sendo, por isso, os atributos "cidade" e "rua" atributos primos. Isto torna a dependência, que viola a BCNF, válida para a 3NF.

Como as restantes relações estão na forma BCNF, também estão inevitavelmente na 3ª Forma Normal.

Para que a base de dados se encontrasse toda na forma BCNF e na 3NF, ter-se-ia de criar uma classe para guardar os atributos do código-postal e relacioná-la com a classe **Morada**, em que cada **Morada** tem apenas um código-postal, mas este poderia estar associado a mais que uma **Morada**. Isto traduzir-se-ia na seguinte relação:

CodigoMorada (morada->Morada, codigo-postal→CodigoPostal)

morada → codigo-postal

Restrições

Para assegurar a fácil manutenção da integridade dos dados armazenados foram incluídas na implementação da base de dados restrições na criação de várias classes. Nomeadamente, foram introduzidas restrições do tipo chave, de integridade referencial e do tipo CHECK e NOT NULL.

No que diz respeito a restrições do tipo chave foram utilizadas *primary keys* em todas as classes. Foi ainda atribuída a restrição **UNIQUE** a atributos da classe que não se podem repetir, chegando até a ser possível usá-los para chegar aos restantes atributos dessa classe, já que são *superkey's* dessas relações:

- Atributo "telefone" nas classes **Loja, Funcionario** e **Cliente** -> não podem existir dois números de telefone iguais;
- Atributo "funcionario" na classe **Gestor** -> cada loja só pode ter um único gestor;
- Atributo "NIF" na classe **Cliente** -> não pode haver clientes com NIF's idênticos;
- Atributo "encomendaID" na classe **Pagamento** -> não pode ser feito mais do que um pagamento para cada encomenda;
- Na classe **Morada**, o conjunto de atributos "cidade", "rua" e "numero**"** não se pode repetir.

A restrição **NOT NULL** é empregue em ocasiões em que o acréscimo de um objeto de uma classe implica a existência obrigatória de um ou mais atributos dessa classe. Alguns exemplos simples desta restrição podem ser:

- Os atributos "nome", "telefone", "NIF" e "moradaID" na classe **Cliente** estão declarados com a restrição **NOT NULL**, pois são imprescindíveis para identificação intuitiva e tratamento de encomendas do cliente;
- Na classe **Morada**, todos os atributos, com a exceção de "moradaID" (primary key), foram identificados com a restrição **NOT NULL,** uma vez que são todos essenciais para a identificação de uma morada;
- Na classe **Loja**, os atributos "telefone" e "moradaID" estão marcados com esta restrição, pois uma loja não pode existir se não tiver um contacto ou uma localização;
- Na classe **Seccao**, os atributos "nome" e "lojaID" têm de estar definidos para que a secção seja valida.

As classes **Funcionario**, **Gestor**, **Cartao**, **Promocao**, **Encomenda**, **Pagamento**, **Produto**, **Fornecedor** e **Fornecimento** têm também atributos com a restrição **NOT NULL** por razões idênticas às que já foram anteriormente explicitadas.

A restrição **CHECK**, por seu lado, foi utilizada com o propósito de limitar os valores que certos atributos podem ou não tomar. Desta forma, é mais facilmente assegurada coerência nos dados.

- Na classe **Morada**, a restrição **CHECK** foi usada de forma a assegurar que o código postal da morada é válido, ou seja, o "codigo_postal1" está limitado entre 0 e 10000 e o "codigo_postal2" entre 0 e 1000. Assim, o código postal completo da morada será um valor do tipo '4405-123';
- Na classe **Loja**, esta restrição foi implementada para verificar que a área da loja é um valor positivo. O mesmo foi feito para os atributos "saldo" e "stock" nas classes **Cartao** e **Produto** respetivamente;
- O atributo "telefone", em qualquer classe onde esteja definido, está sempre limitado por uma restrição do tipo **CHECK**, de modo a que todos os números de telefone na base de dados tenham sempre nove algarismos.
- O atributo "NIF" na classe cliente tem uma restrição idêntica à do atributo **telefone**;
- Na classe **Promocao**, o atributo "percentagem" foi limitado de forma ter valores que se encontram sempre entre 0 e 100 (%). Uma restrição idêntica foi implementada no atributo "desconto" da classe **EncomendaDetalhada**;
- Por último, esta restrição foi ainda aplicada ao atributo "tipo_pagamento" da classe **Pagamento**, para assegurar que este era um dos três tipos de pagamento possíveis (dinheiro, multibanco ou cheque).

Por fim, no que diz respeito a restrições de integridade referencial, foram empregues chaves estrangeiras a classes relacionadas entre si. Um exemplo deste tipo de restrição pode ser visto na classe **Funcionario**. Uma vez que, a cada funcionário estão associadas uma loja e uma morada, justificando-se o emprego de duas chaves estrangeiras: uma em "lojaID" e outra em "moradaID". Este raciocínio foi também utilizado nas restantes classes da base de dados para a implementação destas restrições.

Interrogações

Como forma a interagir e perceber a base de dados, foram criadas dez interrogações consideradas pertinentes no contexto de uma Cadeia de Supermercados. As interrogações respondem às seguintes questões:

1 - Quais são os funcionários que trabalham na Secção de ID igual a 1 (Frescos)?

2 - Qual o número de encomendas realizadas por cada cliente?

3 - Atendendo a preços de compra, quantidades vendidas e preços de venda, de quanto é o lucro da cadeia de supermercados?

4 - Quais as promoções que estejam ativas entre "2019-04-06" e "2019-04-12"? Tabela ordenada ascendentemente pela data de fim da promoção.

5 - Qual a faixa etária dos funcionários pelos quais cada gestor está responsável? Tabela apresenta o número de funcionários, a idade do mais novo e do mais velho e a média das idades.

6 - Quais os 3 melhores funcionários da loja? Esta interrogação tem em conta o número de funcionários pelos quais é responsável e o valor das encomendas realizadas por ele.

7 - Quais as ruas onde moram simultaneamente clientes e funcionários?

8 - Quais os Fornecedores que forneceram produtos numa quantidade total superior a 10?

9 - Clientes com cartão ordenados por valor de desconto imediato usufruído num determinado mês (abril).

10 - Quais sãos os fornecedores que já realizaram pelo menos um fornecimento à cadeia de lojas?

Gatilhos

Os gatilhos implementados na base de dados foram os seguintes:

1 - Para garantir a integridade do stock de produtos aquando de uma compra, tendo sido criado um gatilho que subtrai a quantidade de produto comprado sempre que uma compra é inserida.

2 - De forma a garantir que não são feitos pagamentos com Promoções fora de prazo, foi criado um gatilho que verifica a data de início e fim da promoção e compara com a data da realização da encomenda.

3 – Este gatilho garante que a quantidade de produto encomendado não excede a quantidade em stock.

4 - Criou-se este gatilho para garantir que o stock de produtos é atualizado aquando de um fornecimento novo. Este gatilho foi feito para complementar o gatilho 1 e assim garantir que o stock está sempre atualizado.