Sistema de Informação para Empresa de Madeiras e Cerâmicas

2ª Entrega

2MIEIC06 - Grupo 602

Bases de Dados 2017/2018

Mestrado Integrado em Engenharia Informática e Computação

Beatriz de Henriques Martins - up201502858@fe.up.pt
Luisa Zambelli Artmann Rangel - up201710784@fe.up.pt
Vitor Miguel Medeiros Cordeiro - up201505087@fe.up.pt

15 de abril de 2018

Conteúdo

1 Descrição

O objectivo de uma *empresa* especializada em construir cozinhas é fabricar, montar e instalar.

A *empresa* mantém um registo dos seus clientes. É necessário armazenar alguns dados dos seus *clientes* como o nome, a morada, o telefone, o e-mail e o NIF (Número de Identificação Fiscal) para a faturação e uma lista de compras anteriores. Os *clientes* são *pessoas*.

Para continuar o trabalho exemplar que os seus *clientes* apreciam, a *empresa* precisa de *funcionários* especializados nas mais diversas áreas, como por exemplo marceneiros, vidreiros, pedreiros, designers, vendedores, administrativos entre tantas outras especialidades. Sobre estes, além da *especialidade*, é necessário guardar outros dados, como por exemplo, o cargo atual, o ordenado, o horário de trabalho e o agregado familiar (que representa o número de dependentes do trabalhador). Todos os que trabalham na *empresa* são *pessoas*, logo as informações básicas necessárias que esta necessita são iguais às do *cliente*. A empresa tem dois tipos de *estabelecimento*, a *fábrica* e a *loja*, cada um deles tem dados próprios, como morada, telefone, e-mail e um grupo de funcionários diversificados adaptados às necessidades de cada estabelecimento.

Associada a cada *fábrica* existe também uma lista de *fornecedores* (empresas externas que fornecem materiais). Estes partilham dados como o nome, a morada, o telefone, o e-mail e o NIF. A *empresa* guarda também informação sobre os *materiais* fornecidos por cada *fornecedor*. Sobre os *materiais* precisamos de saber o nome, o tipo, o preço, a cor e a quantidade necessária para cada produto.

A partir dos *materiais*, a *empresa* consegue fabricar seus *produtos*, as cozinhas. Cada *produto* exige *funcionários* especializados para sua construção. Cada um tem um conjunto de dados específico como o preço, a cor e as dimensões deste.

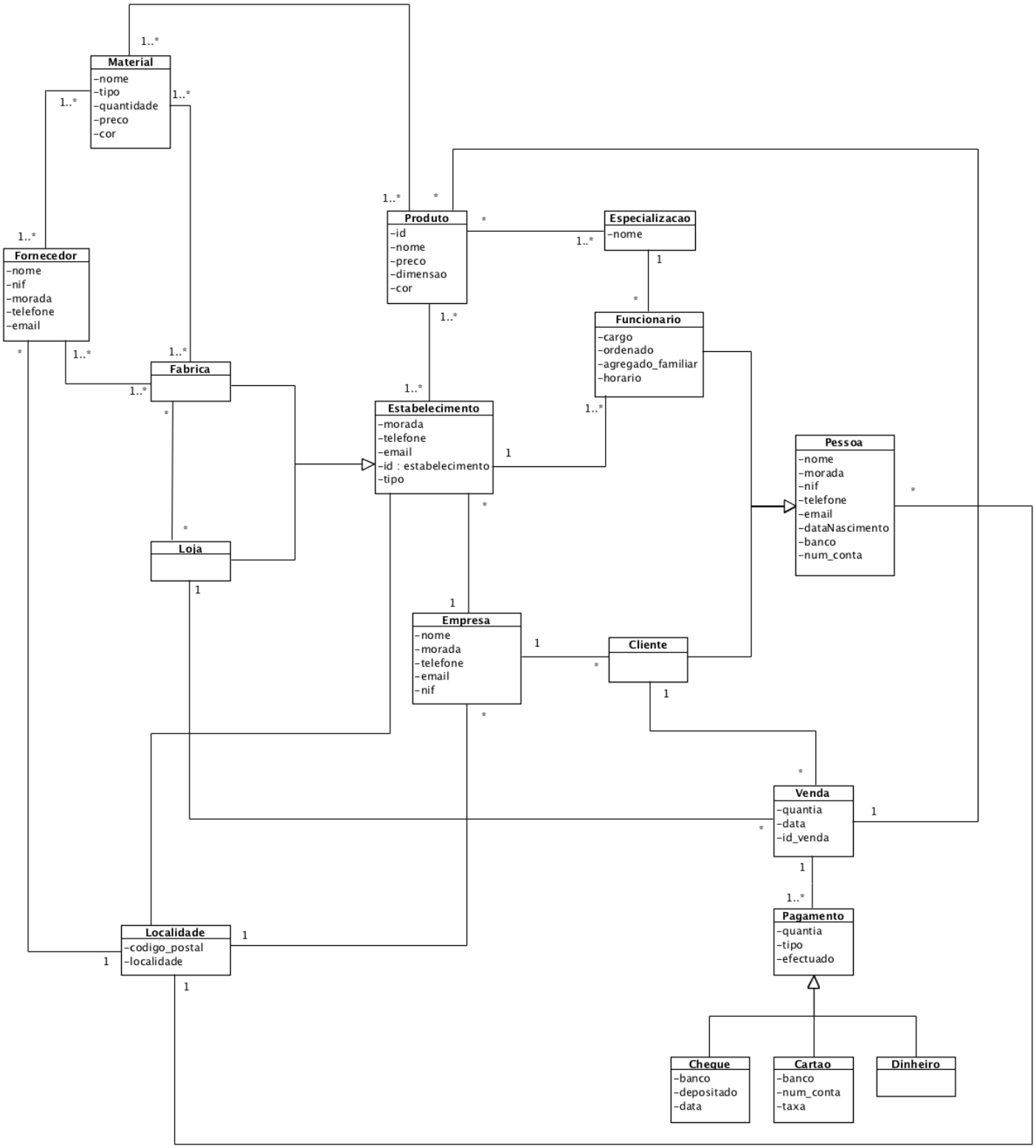
Na ocorrência de uma *venda*, que relaciona um *cliente* a um ou mais *produtos*, deve-se registar a data, a quantia final e a *loja* em que ocorreu. Sobre o *pagamento* temos de saber o estado em que se encontra e se foi realizado através de um cartão, um cheque ou dinheiro. Dependendo da forma de *pagamento*, precisamos de armazenar os dados bancários e tarifas aplicadas.

2 Classe e Atributos

Classe	Atributos
Empresa	nome morada telefone email nif
Cliente	id
Venda	idVenda quantia data
Dinheiro	
Cheque	banco depositado data
Estabelecimento	idEstabelecimento morada telefone email tipo
Fabrica	
Material	idMaterial tipo preco cor quantidade

Classe	Atributos
Pessoa	nome cc morada telefone email nif dataNascimento nomeBanco numeroConta
Funcionario	idEmpresa especializacao ordenado horario agregadoFamiliar cargo
Pagamento	quantia tipo efetuado
Cartao	banco numeroConta taxa
Localidade	codigoPostal localidade
Loja	
Fornecedor	idFornecedor nome morada telefone email
Produto	idProduto nome preco dimensao cor

3 Diagrama de UML



4 Modelo Relacional e Dependências Funcionais

Pessoa(<u>nif</u>, nome, morada, telefone, email, dataNasc, banco, numConta, codPostal->*Localidade*)

- <u>nif</u> → nome, morada, telefone, email, dataNasc, banco, numConta, codPostal

Cliente(<u>nif</u>->*Pessoa*, nomeEmpresa->*Empresa*)

- <u>nif</u> → *Pessoa*, nomeEmpresa

Funcionario(<u>nif</u>->*Pessoa*, cargo, ordenado, agregadoFamiliar, horario, idEstabelecimento->*Estabelecimento*, nomeEspecializacao->*Especializacao*)

- <u>nif</u> → *Pessoa*, cargo, ordenado, agregadoFamiliar, horario, idEstabelecimento, nomeEspecializacao

Especializacao(<u>Nome</u>)

Produto(<u>idProduto</u>, nome, preco, dimensao, cor, idVenda->*Venda*)

- <u>idProduto</u> → nome, preco, dimensao, cor, idVenda

Estabelecimento(<u>idEstabelecimento</u>, morada, telefone, email, tipo, nomeEmpresa->*Empresa*, codPostal->*Localidade*)

- <u>idEstabelecimento</u> → morada, telefone, email, tipo, nomeEmpresa, codPostal

Fabrica(<u>idEstabelecimento</u>->*Estabelecimento*)

Loja(<u>idEstabelecimento</u>->*Estabelecimento*)

Empresa(<u>nif</u>, nome, morada, telefone, email)

- <u>nif</u> → nome, morada, telefone, email

Material(<u>idMaterial</u>, nome, tipo, quantidade, preco, cor)

- <u>idMaterial</u> → nome, tipo, quantidade, preco, cor

Fornecedor(<u>nif</u>, nome, morada, telefone, email, codPostal->*Localidade*)

- <u>nif</u> → nome, morada, telefone, email, codPostal

Localidade(codPostal, localidade)

- codPostal → localidade

Venda(idVenda, quantia, data, idCliente->*Cliente*, idEstabelecimento->*Loja*)

- idVenda → quantia, data, idCliente, idEstabelecimento

Pagamento(idPagamento, idVenda->*Venda*, quantia, tipo, efetuado, dataPagamento)

- idPagamento → idVenda, quantia, tipo, efetuado, dataPagamento

Cheque(idPagamento->*Pagamento*, banco, depositado, dataDeposito)

- idPagamento → *Pagamento*, banco, depositado, dataDeposito

Cartao(idPagamento->*Pagamento*, banco, numConta, taxa)

- idPagamento → *Pagamento*, banco, numConta, taxa

Dinheiro(idPagamento->*Pagamento*)

- idPagamento → *Pagamento*

Material-Produto(idMaterial->*Material*, idProduto->*Produto*, quantidadeNecessaria)

- idMaterial, idProduto → *Material*, *Produto*, quantidadeNecessaria

Material-Forcedor(idMaterial->*Material*, nomeFornecedor->*Fornecedor*)

- idMaterial, nomeFornecedor → *Material*, *Fornecedor*

Material-Fabrica(idMaterial->*Material*, idEstabelecimento->*Fabrica*)

- idMaterial, idEstabelecimento → *Material*, *Fabrica*

Fornecedor-Fabrica(nomeFornecedor->*Fornecedor*, idEstabelecimento->*Fabrica*)

- nomeFornecedor, idEstabelecimento → *Fornecedor*, *Fabrica*

Loja-Fabrica(idEstabelecimento->*Loja*, idEstabelecimento->*Fabrica*)

- idEstabelecimento, idEstabelecimento → *Loja*, *Fabrica*

Produto-Estabelicemto(idProduto->*Produto*, idEstabelecimento->*Estabelecimento*)

- idProduto, idEstabelecimento → *Produto*, *Estabelecimento*

Produto-Especialização(idProduto->*Produto*, nomeEspecialização->*Especializacao*)

- idProduto, nomeEspecialização → *Produto*, *Especializacao*

5 Formas Normais

Para garantir a integridade de um banco de dados e evitarmos as repetições é importante a normalização. Uma relação pode se encontrar em diferentes formas de normalização, sendo a *Terceira Forma Normal* e a *Forma Normal de Boyce-Codd* as principais e mais utilizadas.

Para uma relação estar na *Terceira Forma Normal* (3FN) é necessário atender primeiramente a *Primeira* (1FN) e a *Segunda* (2FN) Formas Normais. A 1FN consiste em uma tabela apresentar apenas atributos atômicos, fato que é conferido em todas as relações do modelo relacional apresentado. As relações também atendem à 2FN pois além de atenderem a *Primeira Forma Normal*, nenhum atributo não primo é funcionalmente dependente de algum subconjunto de uma chave candidata. Um atributo pertencente a alguma chave é denominado primo.

Além disso, para estar na 3FN, para cada dependência funcional não trivial, o lado esquerdo deve ser uma super chave ou o lado direito consistir somente em atributos primos. Em outras palavras, todos os atributos não chave são completamente dependentes dos atributos chave e são independentes entre si.

A Forma Normal de Boyce-Codd (BCNF) é ainda um pouco mais restrita que a 3FN e garante a não existência de anomalias. As relações estão na BCNF pois todos os atributos são dependentes exclusivamente da chave primária. Dessa forma, também não apresentam nenhuma violação à Terceira Forma Normal.

6 Restrições

De acordo com o modelo de negócio, temos as seguintes restrições:

```
CREATE TABLE Localidade (
    codPostal int NOT NULL UNIQUE,
    localidade text NOT NULL UNIQUE,
    PRIMARY KEY (codPostal)
);
```

- O código postal (CodPostal) não pode ser nulo e é único;
- A localidade (Localidade) não pode ser nulo e é único;

```
CREATE TABLE Especializacao (
     nome text NOT NULL UNIQUE,
     PRIMARY KEY (nome),
);
```

- Todos os funcionários têm uma especialização;
- O nome (nome) é único e não nulo;

```
CREATE TABLE Pessoa (
    nif int NOT NULL UNIQUE,
    nome text NOT NULL,
    morada text,
    codPostal int,
    telefone text,
    email text,
    dataNasc date,
    banco text,
    numConta int,
    PRIMARY KEY (nif),
    FOREIGN KEY (codPostal) REFERENCES Localidade(codPostal)
);
```

- O nif (NIF) não pode ser nulo e é único;
- O nome (Nome) não pode ser nulo;
- A morada (Morada) não pode ser nulo;

```
CREATE TABLE Empresa (
    nif int NOT NULL UNIQUE,
    nome text NOT NULL,
    morada text,
    codPostal int,
    telefone text,
    email text,
    PRIMARY KEY (nif),
    FOREIGN KEY (codPostal) REFERENCES Localidade(codPostal)
);
```

- O nif (NIF) não pode ser nulo e é único;
- O nome (Nome) não pode ser nulo;

```
CREATE TABLE Estabelecimento (
    idEstabelecimento int NOT NULL UNIQUE,
    morada text NOT NULL,
    codPostal int NOT NULL,
    telefone text,
    tipo text NOT NULL,
    email text,
    nomeEmpresa text,
    PRIMARY KEY (idEstabelecimento),
    FOREIGN KEY (nomeEmpresa) REFERENCES Empresa(nome),
    FOREIGN KEY (codPostal) REFERENCES Localidade(codPostal)
);
```

- Cada estabelecimento tem um número de identificação (IDEstabelecimento) não nulo e único;
- A morada (Morada) não pode ser nulo;
- O email (Email) não pode ser nulo;
- O tipo de estabelecimento (tipo) não pode ser nulo;

```
CREATE TABLE Venda (
    idVenda int NOT NULL UNIQUE,
    quantia float NOT NULL,
    dataVenda date NOT NULL,
    idCliente int,
    idEstabelecimento int,
    PRIMARY KEY (idVenda),
    FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente),
    FOREIGN KEY (idEstabelecimento) REFERENCES Loja(idEstabelecimento),
);
```

- Cada venda tem um número de identificação (idVenda) único e não nulo;
- A quantia (quantia) não pode ser nulo;
- A data de venda (dataVenda) não pode ser nulo;

```
CREATE TABLE Produto (
    idProduto int NOT NULL UNIQUE,
    nome text NOT NULL,
    preco float NOT NULL,
    dimensao text,
    cor text,
    idVenda int,
    PRIMARY KEY (idProduto),
    FOREIGN KEY (idVenda) REFERENCES Venda(idVenda)
);
```

- Cada produto tem um número de identificação (idProduto) único e não nulo;
- O nome (nome) não pode ser nulo;
- O preço (preco) não pode ser nulo;

```
CREATE TABLE Pagamento (
    idPagamento int NOT NULL UNIQUE,
    idVenda int NOT NULL,
    quantia float NOT NULL,
    tipo text NOT NULL,
    efetuado text NOT NULL,
    dataPagamento date NOT NULL,
    PRIMARY KEY (idPagamento),
    FOREIGN KEY (idVenda) REFERENCES Venda(idVenda)
);
```

- Cada pagamento tem um número de identificação (idPagamento) único e não nulo;
- O número de identificação de venda (idVenda) não pode ser nulo;
- A quantia (quantia) não pode ser nula;
- O tipo de pagamento (tipo) não pode ser nulo;
- A data de nascimento (DataNasc) não pode ser nulo;
- A data de pagamento (dataPagamento) não pode ser nula;

```
CREATE TABLE Cheque (
    idPagamento int NOT NULL,
    banco text NOT NULL,
    depositado text NOT NULL,
    dataDeposito date NOT NULL,
    PRIMARY KEY (idPagamento),
    FOREIGN KEY (idPagamento) REFERENCES Pagamento(idPagamento)
);
```

- Cada pagamento tem um número de identificação (idPagamento) único e não nulo;
- O nome do banco (banco) não pode ser nulo;
- A data em que foi feito o depósito (dataDeposito) não pode ser nulo;
- A data de nascimento (DataNasc) não pode ser nulo;

```
CREATE TABLE Cartao (
    idPagamento int NOT NULL,
    banco text NOT NULL,
    numConta int NOT NULL,
    PRIMARY KEY (idPagamento),
    FOREIGN KEY (idPagamento) REFERENCES Pagamento(idPagamento)
);
```

- O número de idetificação do pagamento (idPagamento) não pode ser nulo;
- O nome do banco (banco) não pode ser nulo;
- O número de conta (numConta) não pode ser nulo;

```
CREATE TABLE Dinheiro (
    idPagamento int NOT NULL,
    PRIMARY KEY (idPagamento),
    FOREIGN KEY (idPagamento) REFERENCES Pagamento(idPagamento)
);
```

- O número de identificação do pagamento não pode ser nulo;

```
CREATE TABLE Funcionario (
    nif int,
    idEstabelecimento int,
    cargo text NOT NULL,
    ordenado float NOT NULL,
    agregadoFamiliar int NOT NULL,
    horario text NOT NULL,
    PRIMARY KEY (nif),
    FOREIGN KEY (nif) REFERENCES Pessoa(nif),
    FOREIGN KEY (idEstabelecimento) REFERENCES Estabelecimento(idEstabelecimento)
);
```

- O cargo (cargo) não pode ser nulo;
- O valor do ordenado (ordenado) não pode ser nulo;
- O agregado familiar (agregadoFamiliar) não pode ser nulo;
- O horário (horario) não pode ser nulo;

```
CREATE TABLE Material (
    idMaterial int NOT NULL UNIQUE,
    nome text NOT NULL,
    tipo text NOT NULL,
    estoque float NOT NULL,
    preco float,
    cor text,
    PRIMARY KEY (idMaterial)
);
```

- Cada material tem um número de identificação (idMaterial) único e não nulo;
- O nome do material (nome) não pode ser nulo;
- O tipo do material (tipo) não pode ser nulo;
- O stock do material (estoque) não pode ser nulo;

```
CREATE TABLE Fornecedor (
    idFornecedor int NOT NULL UNIQUE,
    nome text NOT NULL UNIQUE,
    morada text,
    codPostal int,
    nif int NOT NULL UNIQUE,
    telefone text,
    email text,
    PRIMARY KEY (idFornecedor),
    FOREIGN KEY (codPostal) REFERENCES Localidade(codPostal)
);
```

- Cada forncedor tem um número de identificação único (idFornecedor) e não nulo;
- O nome de cada fornecedor (nome) é único e não pode ser nulo;
- O número de identificação fiscal (nif) é único e não nulo;

7 Referências

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems. 3rd Edition.