



**Mestrado Integrado em Engenharia Informática e
Computação**

Projeto de Bases de Dados

**Esquema Relacional, criação de Base de Dados em SQL
com restrições e carregamento de dados**

Grupo 108

Composição do grupo:

- Francisco José Paiva Gonçalves (up201704790)
- Luís Pedro Viana Ramos (up201706253)
- Paulo Jorge Palhau Moutinho (up201704710)

14 de abril de 2018

Índice

Tema do trabalho	3
Contextualização	4
Introdução.....	4
Pessoa e cliente	4
Loja	4
Produtos e Secções	4
Encomenda	5
Suporte Online	5
Transportadora	5
Diagrama UML.....	6
Esquema Relacional, dependências funcionais e formas normais	7
Restrições	10
Preenchimento da Base de dados	14
Queries (Interrogações)	14
Triggers (Gatilhos).....	14

Tema do Trabalho

Um supermercado pretende criar uma base de dados para gerir as encomendas relativas a alguns setores da sua loja online. Para este efeito pediu ajuda a alunos do MIEIC para concretizar a tarefa. O supermercado requesitou que fosse necessário guardar a informação de cada pessoa que utiliza a sua loja online, quer seja cliente ou funcionário, sendo que é necessário saber o nome, morada, nif e nº de telemovel de cada pessoa. O site irá guardar os diversos produtos distribuidos em várias secções: charcutaria, limpeza, bebidas e lacticínios. Para este efeito cada produto tem o seu código de barras, nome, marca, validade, preço e um desconto, caso exista alguma promoção. As encomendas são limitadas a um stock que é mantido pelo supermercado e podem ser pagas por diversos tipos de pagamento, sendo estes dinheiro, multibanco e PayPal. As encomendas podem ser levantadas diretamente da loja ou podem ser enviadas por uma de várias empresas transportadoras. A loja pediu também que existisse um sistema de apoio ao cliente em que o cliente pode enviar um pedido de apoio e caso exista um funcionário disponível este será atendido.

Contextualização

A contextualização está descrita abaixo. Nem todas as classes têm um cabeçalho individual, mas estão ligadas entre si, estando algumas descritas dentro de outros cabeçalhos, como por exemplo o stock.

Introdução

Assumiu-se que a loja online a ser desenvolvida vende quatro tipos de produtos, organizados por secções: bebidas, limpeza, charcutaria, e lacticínios. O SuperMieicado tem uma loja física, mas pretende ter uma ferramenta online para as secções descritas acima.

Pessoa e cliente

A classe pessoa contém os seguintes atributos: nome, NIF, morada, nº telemóvel. Esta classe é subdividida em cliente e funcionário, sendo que o cliente terá de fornecer um endereço de email a fim de efetuar encomendas e pedidos de ajuda online. Em relação à subclasse funcionário será guardado se o funcionário se encontra disponível para atender um pedido ou não. Ambas as subclasses poderão ser identificadas através do NIF contido em Pessoa.

Loja

A classe loja representa a loja física do SuperMieicado, representando o local onde estão os produtos. A classe terá um nome, um contacto telefónico e uma morada. A esta classe estarão associados todos os artigos na loja online, que depois podem estar (ou não) disponíveis nesta mesma loja. A loja tem pelo menos um produto. O stock estará associado à loja.

Produtos e Secções

A classe produto remete não só para a classe loja, como para as de encomendas. Terá como atributos: nome, marca, código de barras, preço, data de validade e desconto. O produto será uma superclasse, que terá associadas a si subclasses, nomeadamente, as secções (de produtos). Estas contêm os diversos produtos característicos que necessitam de mais atributos para serem melhor classificados, nomeadamente:

Bebidas: quantidade (volume), tamanho de pack, tipo de bebida (inteiro, 0-água, 1-sumo, 2-refrigerante, 3-alcoólica);

Limpeza: tamanho de pack, peso;

Charcutaria: tipo de carne (inteiro, 0-carne branca, 1-vermelha), peso, embalado (sim/não);

Lacticínios: quantidade (volume), tamanho de pack, lactose (sim/não).

Encomenda

A classe encomenda terá como atributos: a data, o preço (calculado através do preço dos artigos e um possível desconto se houver), o método de pagamento (0-dinheiro, 1-multibanco, 2-PayPal) e transporte (sim/não). Cada encomenda será associada a apenas um cliente e a sua principal função será guardar a lista de artigos seleccionados das diferentes secções, através do NIF. Para este efeito, é necessário guardar a quantidade pedida desse artigo seleccionado através de uma classe de associação entre a encomenda e o produto. Esta classe é também associada à transportadora caso o valor de transporte seja sim, será necessário guardar a morada à qual se destina a encomenda.

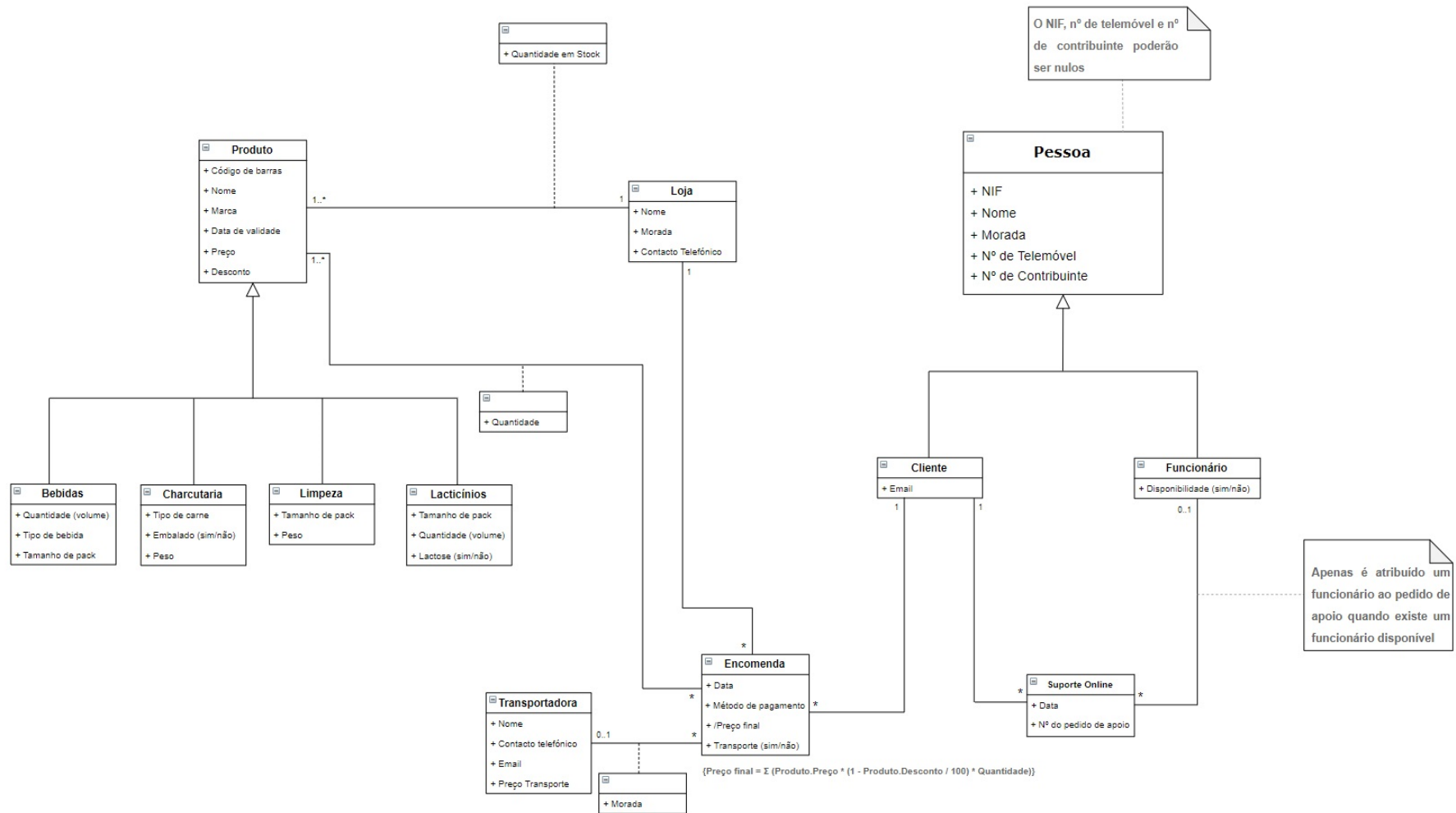
Suporte Online

Os funcionários estão encarregues dos pedidos de ajuda online. Estes têm os atributos data e nº de pedido. Como dentro da subclasse funcionário também há um parâmetro que indica se este tem disponibilidade para assistir o cliente. Caso o funcionário não esteja disponível, o cliente terá que aguardar.

Transportadora

A classe transportadora terá que guardar o nome, contato telefónico, email e preço do transporte, sendo que podem ser associadas várias encomendas a uma transportadora. A transportadora terá um id que a identifica.

Diagrama UML



Esquema relacional, dependências funcionais e formas normais

Pessoa (NIF, nome, morada, telemóvel)

Dependência funcional:

- NIF -> nome, morada, telemóvel
- ✓ 3NF
- ✓ BCNF

Cliente (NIF -> Pessoa, email)

Dependência funcional:

- NIF -> email
- ✓ 3NF
- ✓ BCNF

Funcionário (NIF -> Pessoa, disponível)

Dependência funcional:

- NIF -> disponível
- ✓ 3NF
- ✓ BCNF

Suporte (idPedido, data, NIF -> Funcionário, NIF -> Cliente)

Dependência funcional:

- idPedido -> data, idFuncionário, idCliente
- ✓ 3NF
- ✓ BCNF

Produto (Código de barras, nome, marca, validade, preço, desconto)

Dependência funcional:

- Código de barras -> nome, marca, validade, preço, desconto
- ✓ 3NF
- ✓ BCNF

Bebidas (Código de barras -> Produto, quantidade, tipo, tamanho de pack)

Dependência funcional:

- Código de barras -> quantidade, tipo, tamanho de pack
- ✓ 3NF
- ✓ BCNF

Charcutaria (Código de barras -> Produto, quantidade, tipo de carne, peso, embalado)

Dependência funcional:

- Código de barras -> quantidade, tipo de carne, peso, embalado
- ✓ 3NF
- ✓ BCNF

Limpeza (Código de barras -> Produto, tamanho de pack, peso)

Dependência funcional:

- Código de barras -> tamanho de pack, peso
- ✓ 3NF
- ✓ BCNF

Lactícínios (Código de barras -> Produto, tamanho de pack, quantidade, lactose)

Dependência funcional:

- Código de barras -> tamanho de pack, quantidade, lactose
- ✓ 3NF
- ✓ BCNF

Quantidade Pedida (Código de barras -> Produto, idEncomenda -> Encomenda, quantidade)

Dependência funcional:

- Código de barras, idEncomenda -> quantidade
- ✓ 3NF
- ✓ BCNF

Loja (idLoja, nome, morada, telefone)

Dependência funcional:

- idLoja -> nome, morada, telefone
- ✓ 3NF
- ✓ BCNF

Stock (idLoja -> Loja, código de barras -> Produto, quantidade em stock)

Dependência funcional:

- IdLoja, código de barras -> quantidade em stock
- ✓ 3NF
- ✓ BCNF

Encomenda (idEncomenda, NIF -> Cliente, idLoja -> Loja, data, método de pagamento, preço final, idTransporte -> Transporte)

Dependência funcional:

- idEncomenda -> idCliente, idLoja, data, método de pagamento, preço final, idTransporte
- ✓ 3NF
- ✓ BCNF

Entrega (idTransporte -> Transportadora, idEncomenda -> Encomenda, morada)

Dependência funcional:

- idTransporte, idEncomenda -> morada
- ✓ 3NF
- ✓ BCNF

Transportadora (idTransporte, nome, telefone)

Dependência funcional:

- idTransporte -> nome, telefone
- ✓ 3NF
- ✓ BCNF

Nem todas as relações se encontravam inicialmente na 3ª Forma Normal (3NF) e Boyce-Codd Normal Form (BCNF). Uma relação para estar na BCNF tem de ter **A** como chave numa dependência funcional não trivial **A → B**.

Para estar na 3NF precisa de cumprir este pré-requisito, ou se B apenas consiste de atributos primos. Todas as relações só têm apenas uma dependência funcional e as chaves costumam ser IDs ou números.

Foram feitas alterações às relações cliente e funcionário. Removemos o idCliente e idFuncionário, de maneira a ser mais fácil obter dependência funcional em 3NF e BCNF, reduzindo também alguma redundância, já que passamos a usar o NIF como identificador quer para cliente ou funcionário (pessoa). Todas as relações cumprem os requisitos mencionados acima, exceto as relações cliente e funcionário. Estas relações podem agora ser transferidas diretamente para a criação da base de dados em SQL.

Restrições

Relação	Descrição	Implementação em SQLite
Pessoa	Não podem existir duas pessoas com o mesmo NIF ou com o mesmo nº de telemóvel. Ambos estão entre 0 e o maior número representado com 9 algarismos. Ambos não nulos.	NIF INT PRIMARY KEY CHECK (NIF >= 0 and NIF <= 999999999) NOT NULL, telemovel INT CHECK (telemovel > 0 and NIF < 1000000000)
	Todos as pessoas têm de ter um nome e podem ter uma morada.	nome STRING NOT NULL, morada STRING
Cliente	Não podem existir dois clientes com o mesmo NIF.	NIF INT REFERENCES Pessoa (NIF) ON DELETE CASCADE ON UPDATE CASCADE PRIMARY KEY NOT NULL,
	email é string e tem de incluir os caracteres '@' e '.' Não há mais que um email para cada cliente.	email STRING CHECK (email LIKE '%@%.%') UNIQUE
Funcionário	Não podem existir dois clientes com o mesmo NIF.	NIF INT REFERENCES Pessoa (NIF) ON DELETE CASCADE ON UPDATE CASCADE PRIMARY KEY NOT NULL,
	O funcionário deve indicar sempre a sua disponibilidade	disponivel BOOLEAN DEFAULT [no] NOT NULL
Suporte	Não podem existir dois pedidos com idPedido igual.	idPedido INT PRIMARY KEY AUTOINCREMENT NOT NULL
	Data não pode ser nula e tem por defeito a data atual	data DATETIME NOT NULL DEFAULT(datetime('now'))
	NIF de cliente e funcionário fazem referência a Funcionário e Cliente, tendo de ser coerentes com os valores nessas tabelas. clienteNIF nunca nulo.	funcionarioNIF INT REFERENCES Funcionario (NIF), ON DELETE CASCADE ON UPDATE CASCADE clienteNIF INT REFERENCES Cliente (NIF) NOT NULL ON DELETE CASCADE ON UPDATE CASCADE

Produto	Não podem existir dois produtos com o mesmo código de barras, código de barras maior que 0.	codigoBarras INT NOT NULL PRIMARY KEY AUTOINCREMENT CHECK (codigoBarras > 0) DEFAULT (1)
	Todos os produtos têm que ter nome, preço e desconto. Podem ou não ter marca. Por defeito desconto é 0 e pode ir até 100 (%). Preço por defeito é 1. Preço e desconto são número reais.	nome STRING NOT NULL, marca STRING, preço REAL (2, 2) CHECK (preço > 0) DEFAULT (1) NOT NULL, desconto REAL CHECK (desconto >= 0 AND desconto <= 100) DEFAULT (0) NOT NULL
Bebidas	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos). Não ha codigos de barras repetidos.	codigoBarras INTEGER REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE PRIMARY KEY NOT NULL,
	Quantidade é maior ou igual que 0, não pode ser nula. Valor por defeito é 330mL ("em par" com default pack = 1)	quantidade INT DEFAULT (330) CHECK (quantidade >= 0) NOT NULL,
	Tipo de bebida pode assumir 4 valores: Água, sumo, refrigerante ou alcoólica, sendo que por defeito é água. O tamanho do pack varia entre 0 e 32 e por defeito é 1.	tipo INTEGER DEFAULT (3) NOT NULL CHECK (tipo >= 0 AND tipo <= 3), tamanhoPack INT DEFAULT (1) CHECK (tamanhoPack > 0 AND tamanhoPack <= 32)
Charcutaria	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos) . Não ha codigos de barras repetidos.	codigoBarras INTEGER REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE PRIMARY KEY NOT NULL,
	Tipo de bebida pode assumir 2 valores: Branca ou vermelha. O peso é maior que 0 e por defeito é 100 gramas.	tipoCarne INTEGER DEFAULT (0) NOT NULL CHECK (tipoCarne >= 0 AND tipoCarne <= 1), peso INTEGER CHECK (peso > 0) DEFAULT (100) NOT NULL,
	Embalado é booleano, por defeito é não e não pode ser nulo	embalado BOOLEAN DEFAULT [no] NOT NULL
Limpeza	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos) . Não ha codigos de barras repetidos.	codigoBarras INTEGER REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE PRIMARY KEY NOT NULL,
	Tamanho do pack não pode ser nulo, por defeito é 1 e é maior que 0. O peso é maior que 0 e por defeito é 100 gramas.	tamanhoPack INTEGER CHECK (tamanhoPack > 0) DEFAULT (1) NOT NULL, peso INTEGER CHECK (peso > 0) NOT NULL DEFAULT (100)

Lacticínios	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos). Não ha codigos de barras repetidos.	codigoBarras INTEGER PRIMARY KEY REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL,
	Tamanho do pack não pode ser nulo, por defeito é 1 e é maior que 0. Quantidade (volume) é maior que 0, nunca nula e por defeito é 100mL. Lactose é booleana, não nula e por defeito é sim (contém lactose).	tamanhoPack INTEGER NOT NULL DEFAULT (1) CHECK (tamanhoPack > 0), quantidade INTEGER CHECK (quantidade > 0) NOT NULL DEFAULT (100), lactose BOOLEAN NOT NULL DEFAULT [yes]
Quantidade Pedida	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos). As chaves primárias são idEncomenda e código de barras.	codigoBarras INTEGER REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL, PRIMARY KEY (codigoBarras , idEncomenda ASC)
	ID da encomenda faz referência à Encomenda (tem de ser coerente com valores na tabela das encomendas)	idEncomenda INTEGER REFERENCES Encomenda (idEncomenda) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL,
	Quantidade é um inteiro, nunca nulo, maior ou igual que 0 e por defeito 0.	quantidade INTEGER NOT NULL DEFAULT(0) CHECK (quantidade > 0),
Loja	Não há duas lojas com o mesmo id. Id é obrigatório e por defeito é 1.	idLoja INTEGER PRIMARY KEY ASC AUTOINCREMENT DEFAULT (1) NOT NULL,
	Nome da loja é MIEICado por defeito e deve ser semelhante a MIEICado. Morada da loja tem de existir. Telefone por defeito é 252598694 e pode variar entre 0 e o maximo valor com 9 algarismos.	nome STRING DEFAULT MIEICado CHECK (nome LIKE '%MIEICado%'), morada STRING NOT NULL, telefone INTEGER CHECK (telefone > 0 AND telefone < 1000000000) DEFAULT (252598694)
Stock	Código de barras faz referência a produto (tem de ser coerente com valores na tabela dos produtos). Na tabela de stock não existem pares de código de barras e loja iguais.	codigoBarras INTEGER REFERENCES Produto (codigoBarras) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL, PRIMARY KEY (codigoBarras , idLoja ASC)
	idLoja faz referência a Loja (tem de ser coerente com valores na tabela da Loja).	idLoja INTEGER REFERENCES Loja (idLoja) ON DELETE CASCADE ON UPDATE RESTRICT NOT NULL,
	Stock é um inteiro, maior ou igual a 0 e por defeito toma o valor 1.	stock INTEGER DEFAULT (1) CHECK (stock >= 0) NOT NULL

Encomenda	Não há duas encomendas com o mesmo id, o id é superior a 0.	idEncomenda INTEGER PRIMARY KEY ASC AUTOINCREMENT CHECK (idEncomenda > 0),
	NIF faz reference a Cliente, tendo de ser coerente com os valores na tabela do cliente. Não nulo.	NIF INTEGER REFERENCES Cliente (NIF) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL,
	O idLoja faz referência à loja, tendo de ter ids compatíveis com os presentes na tabela da loja. (valores coerentes)	idLoja INTEGER REFERENCES Loja (idLoja) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL,
	O idTransportadora faz referência à Transportadora, tendo de ser coerente com os valores na tabela referida.	idTransportadora INTEGER REFERENCES Transportadora (idTransportadora) ON DELETE SET DEFAULT ON UPDATE SET DEFAULT,
	Data da encomenda nunca é nula e por defeito é no presente. Método de pagamento pode ser por dinheiro, multibanco ou PayPal. Por defeito é dinheiro e nunca é nulo. Preço final é superior a 0, número real com 2 casas decimais e nunca nulo.	data DATETIME NOT NULL DEFAULT (datetime('now')), metodoPagamento INTEGER NOT NULL DEFAULT (0) CONSTRAINT metodoPagamento_tipo CHECK (metodoPagamento >= 0 AND metodoPagamento <= 2), preçoFinal REAL (2) NOT NULL CHECK (preçoFinal > 0) DEFAULT (0)
Entrega	O idTransportadora faz referência à Transportadora, tendo de ser coerente com os valores na tabela referida.	idTransportadora INTEGER REFERENCES Transportadora (idTransportadora) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL,
	Não há duas encomendas com o mesmo id, o id é superior a 0.	idEncomenda INTEGER REFERENCES Encomenda (idEncomenda) ON DELETE CASCADE ON UPDATE CASCADE UNIQUE,
	Morada não pode ser nula.	morada STRING NOT NULL, PRIMARY KEY (idTransportadora , idEncomenda ASC)
Transportadora	Não há transportadores repetidas e são sempre maiores que 0 e não nulas. Nome nunca é nulo Telefone pode variar entre 0 e o máximo valor com 9 algarismos. Preço final é superior a 0, número real e nunca nulo.	idTransportadora INTEGER PRIMARY KEY AUTOINCREMENTCHECK (idTransportadora > 0) NOT NULL, nome STRING NOT NULL, telefone INTEGER UNIQUE NOT NULL CHECK (telefone > 0 AND telefone < 1000000000), preço REAL NOT NULL DEFAULT (1) CHECK (preço > 0)

Preenchimento da base de dados

Os dados que adicionámos foram de modo relativamente aleatório, com base na percepção da realidade de um supermercado com os produtos especificados para o SuperMIEICado (bebidas, limpeza, etc.).

Queries (Interrogações)

Foram implementadas as 10 interrogações a seguir descritas, com vista a obter mais facilmente determinadas informações acerca dos dados contidos na base de dados

1. Todos os clientes que já fizeram um pedido de apoio
2. Produto mais vendido do mês passado
3. Todos os clientes que já recorreram à transportadora 'Runner'
4. Todos os produtos com stock superior a 70 unidades
5. Todas as encomendas feitas para a morada de um cliente
6. Preço final médio das encomendas de cada cliente
7. Nome de todos os funcionários que já atenderam um pedido
8. Número de produtos vendidos por secção
9. Ordenar todos os clientes por número de encomendas
10. Nome do funcionário que respondeu a mais pedidos de apoio

Triggers (Gatilhos)

1. O preço inicial de qualquer encomenda (com transporte) terá o mesmo valor que o preço de transporte. Deste modo o preço da encomenda (preço final) será preço inicial (preço de transporte) + preço total dos produtos pedidos.
2. Incrementa o preço da encomenda de acordo com a quantidade pedida do produto comprado e decrementa o stock da loja
3. Insere o produto na tabela de stock e igualá-o a 0 para cada loja (só especificámos uma loja na nossa base de dados, com id=1, mas podem existir várias).

O programa deve ser corrido com `.read criar.sql`, seguido de `.read povoar.sql`

Após esta implementação podem ser corridos os triggers e queries.

Os gatilhos devem ser lidos pela ordem “adiciona, verifica, remove”.