Universidade do Minho
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2022/2023

Dorlux

Ioannis Nezis (E10785), Miguel Ângelo Freitas (A91635), Hugo Rocha (A96463), Simão Quintela (A9744)

Novembro, 2022

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Dorlux

Ioannis Nezis (E10785), Miguel Ângelo Freitas (A91635), Hugo Rocha (A96463), Simão Quintela (A9744)

Novembro, 2022

Este trabalho é dedicado à mãe do Miguel.

Resumo

No âmbito da Unidade Curricular de Base de Dados, os docentes Orlando Belo e Regina Sousa propuseram a realização deste projeto, cujo objetivo principal passa pela análise, modelação e implementação de um sistema de Base de Dados.

No que a esta matéria diz respeito, pretende-se criar uma Base de Dados que efetue a gestão de um sistema de entregas ao domicílio para uma loja de comércio local - a Dorlux. Nesse sentido, a Base de Dados terá de ter a capacidade de suportar uma elevada quantidade de informação de forma organizada, permitindo uma fácil compreensão e utilização, enquanto, simultaneamente, se procede às entregas eficientemente.
Ao longo deste relatório iremos abordar as variadas etapas de desenvolvimento que resultaram na construção desta Base de Dados.

Numa primeira fase, começamos por analisar os possíveis requisitos a serem suportados pela Base de Dados para a prossecução dos seus objetivos. Posteriormente, demos início à modulação conceptual, procedendo ao esboço da Base de Dados no brModel, de acordo com os esses mesmos requisitos. Dada esta etapa por concluída e após a aprovação do cliente, iniciamos o processo da modelação lógica. Para tal, utilizamos o MySQL Workbench, igualmente utilizado na modulação física.

Todas as etapas e o próprio processo de desenvolvimento da nossa Base de Dados culminaram no presente trabalho.

Área de Aplicação: Desenho, Arquitetura, Desenvolvimento e Implementação de Sistemas de Bases de Dados.
Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Análise de Requisitos, Entidades, Atributos, Relacionamentos, Modelo Conceptual, Diagrama ER, Modelo Lógico, Modelo Físico, brModel, MySQL Workbench, SQL.

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

1.1. Contexto de aplicação e fundamentação do sistema

A Dorlux fica situada numa pequena vila de Caldas das Taipas, na cidade de Guimarães. Foi criada em 1995 pela D. Dores (Maria das Dores da Silva Alves Freitas), como meio de começar um negócio independente e sair de um trabalho pesado numa fábrica.

A loja começou por ser pequena, dentro do centro comercial da vila, onde primeiramente vendeu adornos para as casas e alguma bijuteria. Com o passar do tempo a loja foi-se transformando e mudando o seu conceito conforme os gostos e as "modas" da atualidade, o que exigiu a alteração para um espaço físico maior, no qual a D. Dores conseguisse expor todos os seus produtos, sem diminuir à oferta nem à qualidade.

O espaço comercial da D. Dores destaca-se dos demais, uma vez que ela procura ir ao encontro dos gostos e necessidades dos seus clientes. Um exemplo prático da amabilidade da dona D. Dores é o facto de os clientes se manterem durante todos os anos da existência da loja, assim como o cuidado em modificar as montras semanalmente.

Até aos dias de hoje, qualquer pessoa pode encontrar a simpatia da D. Dores e a sua loja no centro comercial Passerelle, na Vila das Taipas, aberta de segunda a sábado. De notar que este espaço tem uma dimensão total de 74 m², dividindo-se em loja (50 m²) e armazém (24 m²).

Face à panóplia de produtos que são vendidos nesta loja e à sua própria evolução, a D. Dores viu a necessidade de criar redes sociais (nomeadamente Facebook e Instagram), de forma a expor os seus produtos e dar a conhecer as novidades de forma mais eficaz àqueles que já eram clientes habituais. Felizmente, e para surpresa da D. Dores, as redes sociais permitiram dar a conhecer a Dorlux a pessoas que não residiam na Vila e até mesmo a pessoas que residiam foram do seu concelho. Ainda assim, a D. Dores não descurou da sua preocupação e carinho por todos aqueles que interagiam com a sua loja.

A D. Dores e o seu marido (Sr. Francisco) são os únicos funcionários da loja. No entanto, contam com o apoio dos seus filhos (Francisco, Miguel e Sofia) e de outros membros da família sempre que as dificuldades apertam. Neste sentido, é correto afirmar que a Dorlux é um negócio familiar, do qual todos os que colaboram partilham do mesmo carinho pelo espaço e pelos clientes.

As funções diárias daqueles que vêm ajudar a D. Dores e o seu marido passam pelo atendimento ao cliente, pela arrumação da loja e pela identificação dos preços dos produtos. Acrescem a estas funções o trabalho diário do Sr. Francisco, que é quem está encarregue da gestão das redes sociais da loja (tirar fotografias, criar os posts, enviar as encomendas, etc.). Por sua vez, a D. Dores é a principal responsável pelo atendimento ao cliente, quer no espaço físico (através da recomendação dos produtos, da fiscalização dos pagamentos, da faturação das vendas, etc.) quer à distância (resposta aos clientes que entram em contacto com a Dorlux através das redes sociais ou por contacto telefónico).

É ainda importante notar o trabalho levado a cabo por um contabilista, em prol da loja da Dona Dores, sendo esta última também considerada a manager. Assim, a D. Dores tem muita responsabilidade nas suas mãos, ao assumir dois cargos na sua própria loja.

Há alguns meses, a D. Dores descobriu que estava doente, tendo-lhe sido diagnosticado cancro da mama. Preocupada com o bem-estar dos seus clientes, tem procurado conciliar a sua saúde com o seu negócio. No entanto, respeitando todos os conselhos dos seus médicos, a D. Dores viu-se obrigada a encerrar o espaço física da Dorlux. Face a isto, a D. Dores e o seu marido procuraram uma forma de dar continuidade à sua loja, satisfazendo as necessidades dos seus clientes e mantendo o seu meio de subsistência.

Sendo que a D. Dores estará a realizar tratamentos durante um período alargado, serão vários os meses nos quais o espaço físico da Dorlux estará encerrado. Por isso, a D. Dores procurou uma forma de gerir todo o seu negócio sem sair de casa e sem fazer esforços físicos. Lembrou-se do trabalho que fazia juntamente com o seu marido nas redes socais e decidiu expandir a sua marca para o online. Assim, criou um site no qual poderia responder aos seus clientes mais rapidamente e de forma mais detalhada (melhorando consequentemente o próprio atendimento ao cliente); controlar todos os gastos inerentes à loja; e principalmente controlar todo o processo de realização de encomendas, nomeadamente no que diz respeito à sua personalização, ao tempo e aos gastos associados. Desta forma, poderia também receber o feedback dos clientes e consequentemente melhorar o seu serviço.

Para que isso fosse possível, a D. Dores pediu ajuda ao seu filho que estuda ciências da computação e que a aconselhou a criarem uma base de dados para que pudessem dar continuidade à atividade da loja dentro dos parâmetros da dita normalidade.

Numa época comum, para além do atendimento ao cliente, é da responsabilidade da D. Dores o controlo de stock, a comparação entre o preço de venda e o preço de compra ao fornecedor e a descrição detalhada dos itens.

O controlo de stock é realizado diariamente de forma manual, através do registo de todos os produtos vendidos, e complementado com a elaboração de um inventário anual, no qual a D. Dores conta com a ajuda da sua família para que o processo seja mais fácil e rápido. A comparação de preços (venda/compra) surge no seguimento da preocupação em garantir os melhores preços ao cliente, procurando igualmente manter os melhores fornecedores. Por último, a descrição detalhada dos itens cabe à D. Dores, que sabe de cor todos os pormenores dos produtos que vende.
Com esta base de dados, todos estes procedimentos serão mais rápidos, mais controlados e, principalmente, estarão à disposição do cliente a qualquer momento.

1.2. Motivação e Objetivos do trabalho

Tendo por base a finalidade do site, o Miguel criou um grupo de trabalho, de forma a definir a metodologia e os objetivos para a criação da base de dados, dos quais se destacam:

- Organização do modelo de negócio;
- Controlo de qualidade do produto;
- Controlo de stocks;
- Controlo de fornecedores (preço de compra, qualidade do produto, etc);
- Gestão das contas da loja (taxa de imposto, taxa de envio, vendas, lucros, etc);
- Personalização ao cliente, conhecendo os seus gostos através de compras passadas (profiling);
    - Estabelecimento de descontos personalizados ao cliente.
- Controlo de qualidade das encomendas, nomeadamente no que diz respeito ao tempo de envio;
- Indicação do estado da encomenda, permitindo ao cliente saber em que ponto do processo de envio se encontra;
- Comparação entre as diferentes transportadoras, de forma a escolher a mais eficiente (tempo vs. preço);
- Conhecimento dos produtos vendidos, para que seja possível estimar com mais/menos saída;
- Conhecimento do feedback dos clientes;
- Melhoraria qualidade de serviço da loja;

1.3. Análise da viabilidade do processo

Este projeto consiste na implementação de uma Base de Dados Relacional que permita analisar e relacionar os dados da futura loja online da D. Dores com o sistema de entregas ao domicílio vista a melhor averiguar as necessidades de melhoramento dos serviços, uma vez que esta deverá possibilitar a manipulação e consulta dos dados de uma forma rápido, ágil e segura.

Este será um processo trabalhoso, mas que trará grandes frutos no futuro, uma vez que será substancialmente mais eficiente que o método atual, especialmente na atualização, consulta e tratamento dos dados, o que permitirá que a D. Dores consiga alcançar os seus objetivos. Para além disso, será também um sistema mais fiável, visto que garantirá uma uniformização dos dados, sendo que qualquer interveniente que pretenda consultar ou alterar os dados o fará de uma forma mais segura.

1.4. Recursos

- Humanos
    - Pessoal da loja e outros ajudantes externos, clientes e grupo de trabalho de desenvolvimento.
- Materiais
    - Hardware (1 servidor).
    - Software (SGBD, website e programa de faturação implementado já na loja).

1.5. Equipa de Trabalho

- Pessoal interno
    - D.Dores, Sr.Francisco, filhos (Francisco, Miguel, Sofia), contabilista, outros familiares
        - Funcionamento da loja, atendimento a clientes, validação de serviços, (preencher o resto)
- Pessoal Externo
    - Equipa de trabalho especialista de bases de dados da universidade do Minho
        - Levantamento de requisitos, modelação do sistema, implementação do sistema, (preencher o resto)
- Outros
    - Clientes selecionados
        - Inquéritos de opinião e validação de serviços.

1.6. Plano de Execução do Projeto

Na Introdução dá-se a contextualização do projeto, seguido pela apresentação do caso de estudo bem como as motivações e objetivos.

Na segunda parte intitulada Levantamento e Análise de Requisitos, menciona-se a metodologia de levantamento e análise dos requisitos, seguida da apresentação dos requisitos de descrição, exploração e controlo, que serão analisados e validados.

No terceiro capítulo, são tratados os aspetos relativos à modelação das entidades que irão integrar o modelo conceptual. Estas serão identificadas e caracterizadas em termos de atributos, o mesmo acontecerá com os relacionamentos. Na próxima fase, é apresentado o diagrama ER que caracterizará o sistema que passará por um processo de validação, tendo em conta os requisitos apresentados no capítulo acima.

Ao longo do quarto capítulo, é feita a conversão do modelo conceptual para a sua versão lógica que será validado através de interrogações do utilizador.

Para concluir o desenvolvimento da base de dados é feita uma conversão do modelo lógico para uma versão física em SQL.

Para finalizar, na Conclusão e Trabalho Futuro, apresentam-se ilações resultantes da elaboração do trabalho e possíveis formas de aprimoramento do projeto realizado.

2. Levantamento e Análise de Requisitos

2.1. Método de levantamento e de análise de requisitos adotado

No que diz respeito ao método de levantamento de requisitos, procuramos reunir com a D. Dores, de forma a auscultar e perceber qual a realidade a ser tratada. No decorrer desta primeira fase, foi possível retirar todas as exigências inerentes, que serão elencadas no decorrer desta secção do relatório.
Realizamos ainda várias entrevistas com outras pessoas que integram a atividade comercial da Dorlux, nomeadamente o contabilista, para averiguar qual o procedimento legal relativo à loja e ao serviço que será implementado. Consideramos igualmente importante proceder a uma pesquisa dentro deste ramo do comércio, através da qual compreendemos quais as dificuldades que podem advir, assim como as necessidades inerentes.

2.2. Organização dos requisitos levantados

2.2.1 Requisitos de descrição

Nr	Description
RD01	A caracterização de um cliente na base de dados deve incluir um identificador único e opcionalmente um número fiscal associado a ele. Deve também, obrigatoriamente, incluir um contacto
RD02	Uma morada deve conter um identificador único, o nome da rua, o código postal e a cidade de residência.
RD03	Um contacto deve conter um identificador único, o nome da pessoa/instituição em questão, o email e o número de telemóvel.
RD04	Cada encomenda deve ter um identificador único, um status, um preço de envio da encomenda opcional, uma data de entrada da encomenda e uma data com a última alteração feita na encomenda. Deve ainda ter um identificador para o funcionário responsável pela encomenda, e um identificador para o cliente que fez a encomenda.

RD05	A entidade empregado deve ter um identificador único, um campo que identifique o manager, um campo que identifique o salário do empregado e um campo que contenha um identificador único para o contacto do empregado.
RD06	Um fornecedor deve conter o seu NIF como identificador único e um contacto
RD07	Uma categoria deve conter um indentificador único bem como o seu nome, uma descrição e a taxa associada a essa categoria
RD08	Um item contém um identificador único, o seu nome, uma descrição do item, um campo que indica o número de stock desse item, o seu preço de venda e de compra e ainda a categoria a que pertence

Tabela 1 – Estruturação dos requisitos de descrição.

2.2.2 Requisitos de exploração

Nr	Description
RE01	Deve ser possível alterar o estado do pedido e dar update à última operação efetuada.
RE02	Deve ser possível inserir novos contactos
RE03	Deve ser possível alterar informação de um contacto
RE04	Deve ser possível apagar um contacto
RE05	Deve ser possível listar todas as encomendas realizadas num certo dia do mês
RE06	Deve ser possível listar encomendas num dado estado duma dada categoria de items
RE07	Deve ser possível listar todas as encomendas com a morada de entrega do cliente correspondente à encomenda
RE08	Deve ser possível listar todas as encomendas atríbuidas a um dado funcionário
RE09	Deve ser possível listar todas as encomendas feitas por um cliente
RE10	Deve ser possível fazer um top X de clientes, em que o valor de X fica ao encargo do utilizador
RE11	Deve ser possível saber quantos e quais items compramos a um certo fornecedor
RE12	Deve ser possível listar o stock de items disponível
RE13	Deve ser possível ordenar os items por número de stock
RE14	Deve ser possível ordenar as categorias por nome
RE15	Deve ser possível ordenar os funcionários por ordem decrescente de salário

Tabela 2 – Estruturação dos requisitos de exploração.

2.2.3 Requisitos de controlo

Nr	Descrição
RC01	Um cliente apenas pode ter um contacto
RC02	Um fornecedor apenas pode ter um contacto
RC03	Um empregado apenas pode ter um contacto
RC04	Um cliente pode ter várias moradas e várias moradas podem estar associadas a vários clientes
RC05	Um cliente pode fazer várias encomendas mas uma encomenda está associada apenas a um cliente.
RC06	Uma encomenda pode ter vários items e vários items podem estar presentes em várias encomendas
RC07	Uma encomenda pode estar associada apenas a um empregado mas um empregado pode estar associado a várias encomendas
RC08	Um item pode ser fornecido por vários fornecedores e um fornecedor pode fornecer vários items
RC09	Um categoria incorpora vários items no entanto um item apenas pertence a uma categoria
RC10	Uma encomenda contém uma certa quantidade de um item.

Tabela 3 – Estruturação dos requisitos de controlo.

2.3. Análise e validação geral dos requisitos

O processo de análise e validação de requisitos deu-se no ato do desenho e consequente implementação do sistema de Bases de Dados, tudo isto relacionado com os critérios definidos.

Importa, nesta fase, proceder ao levantamento dos seguintes requisitos: de descrição, de exploração e de controlo.

No que diz respeito aos primeiros (requisitos de descrição), começamos por analisar os elementos imprescindíveis ao funcionamento da loja, assim como o forma como funcionam entre si.

Posteriormente (requisitos de exploração), selecionamos as informações a extrair, tendo por base as conclusões da fase anterior.

Numa última etapa de análise (requisitos de controlo), foram estabelecidas as restrições de administração da Base de Dados, sempre com vista a garantir que o sistema é coerente.

Por sua vez, no processo de validação procuramos analisar a consistência, precisão e contextualização de todos estes requisitos, de maneira que as falhas na sua modelação sejam evitadas. Recorremos às usuais técnicas de reconsideração, testagem e análise dos modelos que garantem os resultados mais realistas, tendo em consideração os objetivos impostos pelo cliente.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Findo o levantamento dos requisitos e respetiva análise, voltamos a reunir com a D. Dores, por forma a criar o modelo conceptual da Base de Dados. Neste sentido, recorremos ao software brModelo para elaborar um Diagrama ER que representará as entidades, os seus atributos e as respetivas relações, tendo em particular consideração as relações de N para N.

3.2. Identificação e caracterização das entidades

Entidade	Caracterização
Client	Termo que contém a informação sobre os clientes, mais especificamente o número de contribuinte.
Contact	Termo que contém a informação de todos os contactos referentes ao cliente
Address	Termo que contém a informação sobre a morada dos clientes.
Employee	Termo que contém a informação sobre os funcionários que trabalham na loja.
Suplier	Termo que contém a informação sobre os fornecedores e o seu número fiscal.
Item	Termo que contém a informação sobre tudo que o Item constitui e o seu stock.
Order	Termo que contém a informação sobre a encomenda feita pelo cliente e processada pelo empregado

Tabela 4 - Caracterização das Entidades

3.3. Identificação e caracterização dos relacionamentos

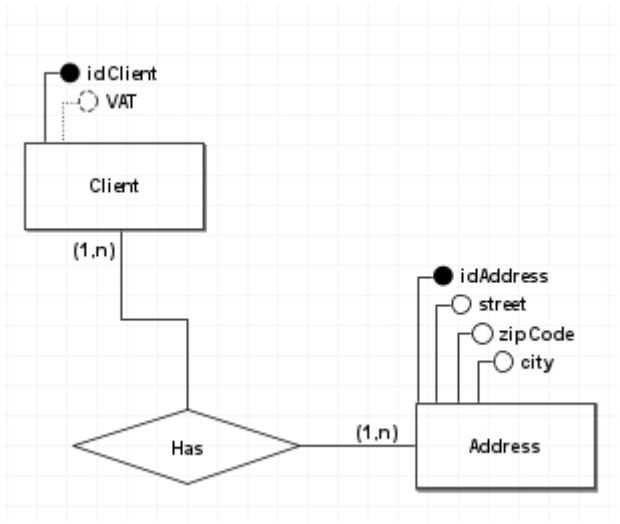

Figura 1 - Relação entre Alojamento e Edifício.

Um cliente pode ter 1 ou N Moradas e uma morada pode estar associada a mais do que um cliente. Logo temos uma cardinalidade de N para N

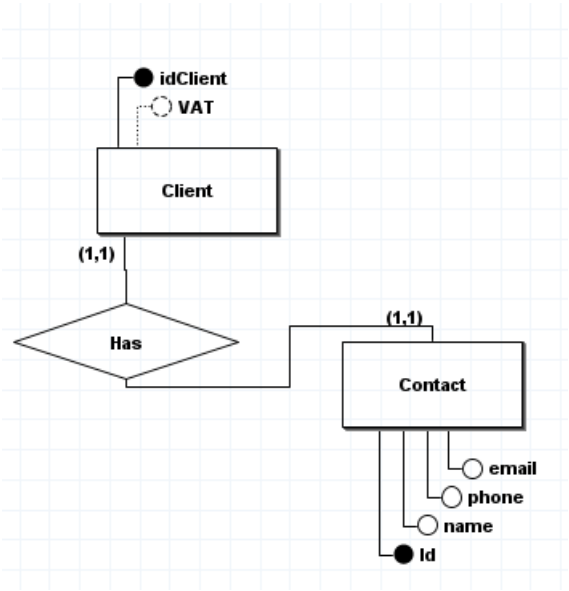

Figura 2 - Relação entre Cliente e Contacto.

Um cliente tem apenas um único contacto, e um contacto está associado a apenas 1 cliente. Logo temos uma cardinalidade de 1 para 1.

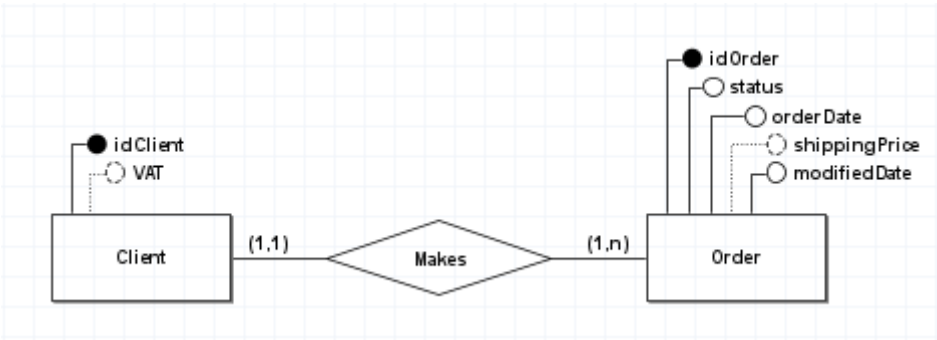

Figura 3 - Relação entre Cliente e Encomenda.

Um cliente pode ter 1 ou N encomendas, no entanto, uma encomenda está associada a uma e apenas uma encomenda. Logo a cardinalidade que temos é de 1 para N.

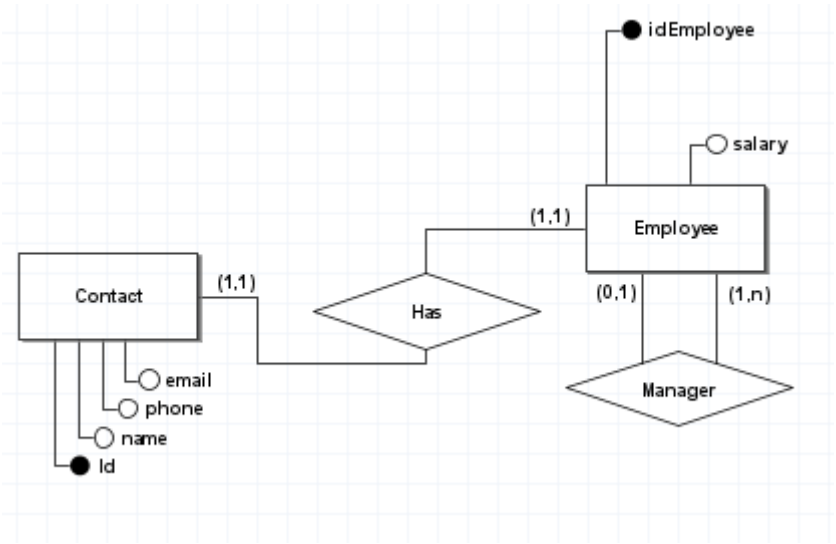

Figura 4 - Relação entre Contacto, Funcionário e Gerência.

Um cliente tem apenas um único contacto, e um contacto está associado a apenas 1 cliente. Logo temos uma cardinalidade de 1 para 1. E para além disso uma gerência pode ter 0 ou N Funcionários e um Funcionário pode ou não ter Gerência. Logo temos uma cardinalidade de 0 para N.

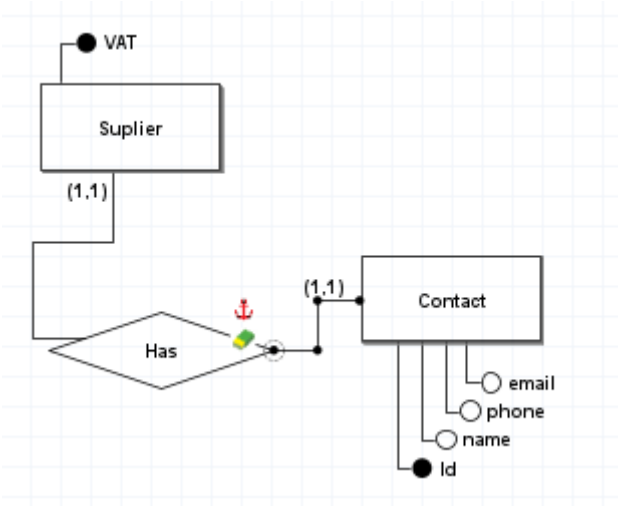

Figura 5 - Relação entre Fornecedor e Contacto.

Um cliente tem apenas um único contacto, e um contacto está associado a apenas 1 cliente. Logo temos uma cardinalidade de 1 para 1.

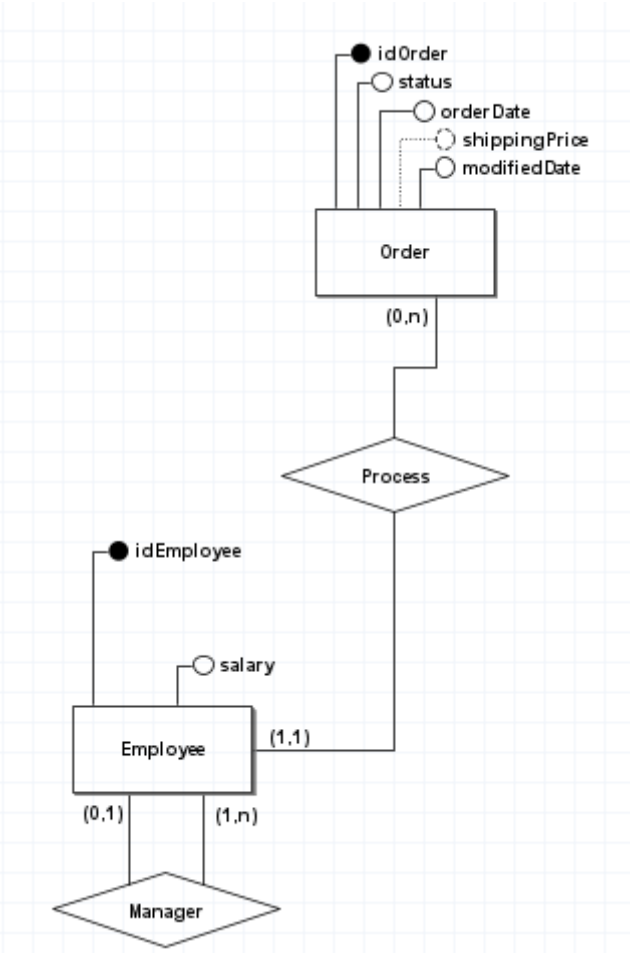

Figura 6 - Relação entre Funcionário e Encomenda.

Um Funcionário pode ter 0 ou N encomendas, no entanto, uma encomenda tem sempre um único Funcionário. Logo temos uma cardinalidade de 0 para N.

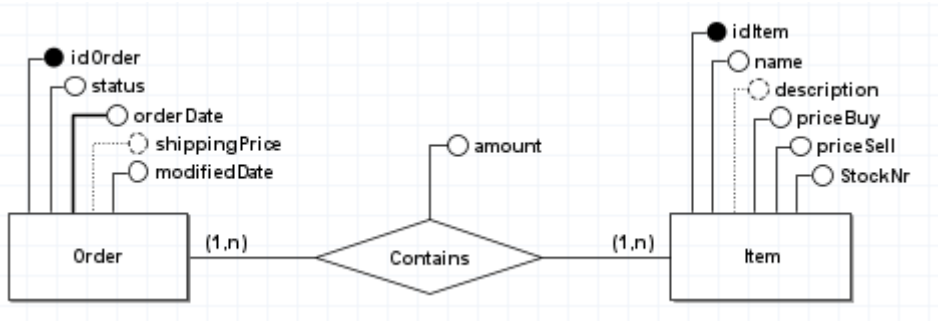

Figura 7 - Relação entre Encomenda e Items.

Uma encomenda pode ter 1 ou N items e um Item está presente em uma ou N encomendas. Uma encomenda contém uma certa quantidade de um item. Logo temos uma cardinalidade de N para N.

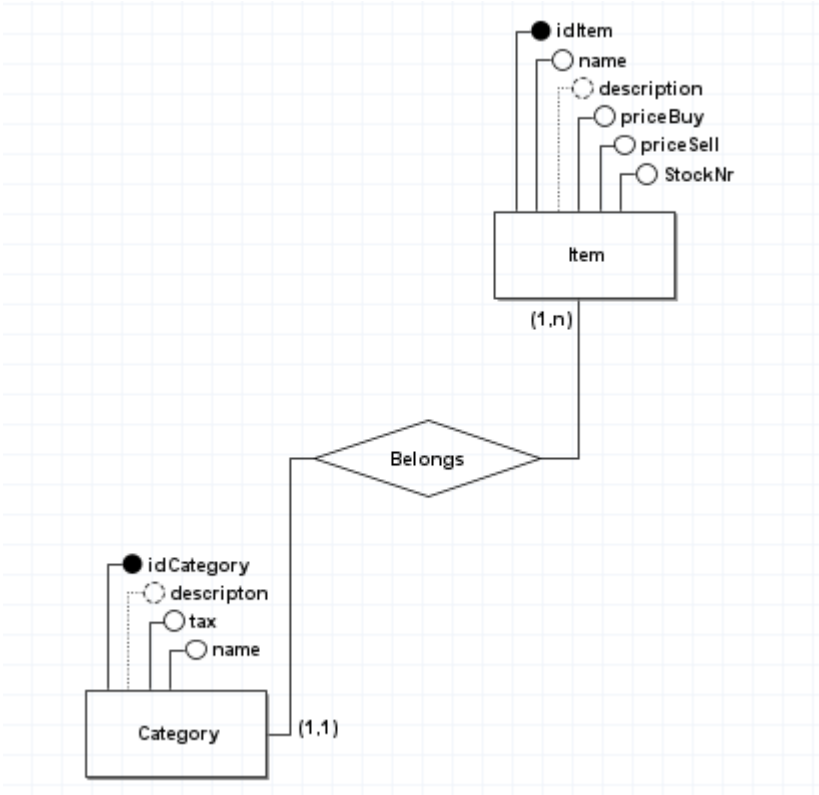

Figura 8 - Relação entre Item e Categoria.

Uma Categoria aporta 1 ou N Items, no entanto, um Item apenas pertence a uma Categoria. Logo temos uma cardinalidade de 1 para N.

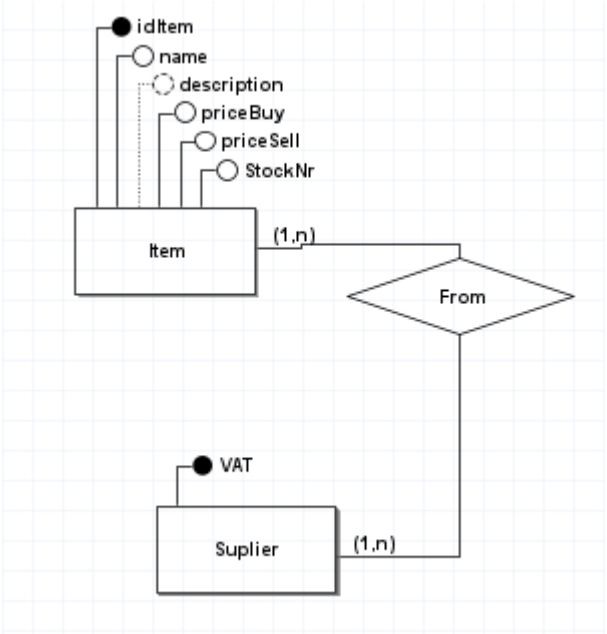

Figura 9 - Relação entre Fornecedor e Item.

Um fornecedor pode fornecer um ou N Items, assim como o mesmo Item pode ser fornecido por 1 ou N fornecedores. Logo temos uma cardinalidade de N para N.

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.

Entidade	**Atributo**	**Tipo**	Chave Primária	Nulo	Descrição	Exemplo
Client	idClient	INT	Sim	Não	Número único que identifica o cliente	1
	VAT	INT	Não	Sim	Número de contribuinte do cliente	999999999
Contact	idContact	INT	Sim	Não	Número único que identifica o contacto	1
	name	VARCHAR (255)	Não	Não	Nome da pessoa que está associado a este contacto	Joao
	email	VARCHAR (255)	Não	Não	E-mail da pessoa	joao@example.com
	phone	INT	Não	Não	Número de telemóvel da pessoa	939800001
Address	idAdress	INT	Sim	Não	Número único que identifica a morada	1
	street	VARCHAR (255)	Não	Não	Rua referente à morada registada pelo cliente	Avenida da Liberdade
	zipCode	VARCHAR (10)	Não	Não	Código postal da morada	4604-203
	city	VARCHAR (45)	Não	Não	Cidade referente à morada	Guimarães
Employee	idEmployee	INT	Sim	Não	Número único que identifica o empregado	1
	salary	DECIMAL (6,2)	Não	Não	salary referente a esse empregado	12.4
Suplier	VAT	INT	Sim	Não	Número único (neste caso número fiscal) que identifica o fornecedor	999999999
Category	idCategory	INT	Sim	Não	Número único que identifica a categoria que pertence o item	1
	name	VARCHAR (45)	Não	Não	Nome da categoria	Escolar
	descripton	VARCHAR (255)	Não	Sim	Descrição opcional sobre a categoria do item	Coisas da escola
	tax	DECIMAL (6,2)	Não	Não	Imposto sobre a categoria a que o item pertence	20.3
Item	idItem	INT	Sim	Não	Número único que identifica o item	1
	name	VARCHAR (45)	Não	Não	Nome do item	Mochila
	descripton	VARCHAR (255)	Não	Sim	Descrição do item (tamanho,cor,defeitos,…)	Preta
	stockNr	INT	Não	Não	Stock referente ao item	12
	priceBuy	DECIMAL (6,2)	Não	Não	Preço de compra do item	15
	PriceSell	DECIMAL (6,2)	Não	Não	Preço de venda do item	30

Order	idOrder	INT	Sim	Não	Número único que identifica o pedido de compra	1
	Status	VARCHAR (200)	Não	Não	Em que estado vai o pedido de compra ("PENDING","PROCESSED","CANCEL",...)	PENDING
	shippingPrice	DECIMAL (6,2)	Não	Sim	Preço associado ao custo de envio	5.60
	orderDate	DATETIME	Não	Não	Data a que o pedido foi submetido	'2023-01-15 17:00:48'
	upDate	DATETIME	Não	Não	Data da última modificação do pedido	2023-01-15 17:00:48'
Contains	amount	INT	Não	Não	Quantidade de items que a compra contém	12

Tabela 5 - Caracterização dos atributos das diversas entidades e alguns exemplos.

3.5. Apresentação e explicação do diagrama ER

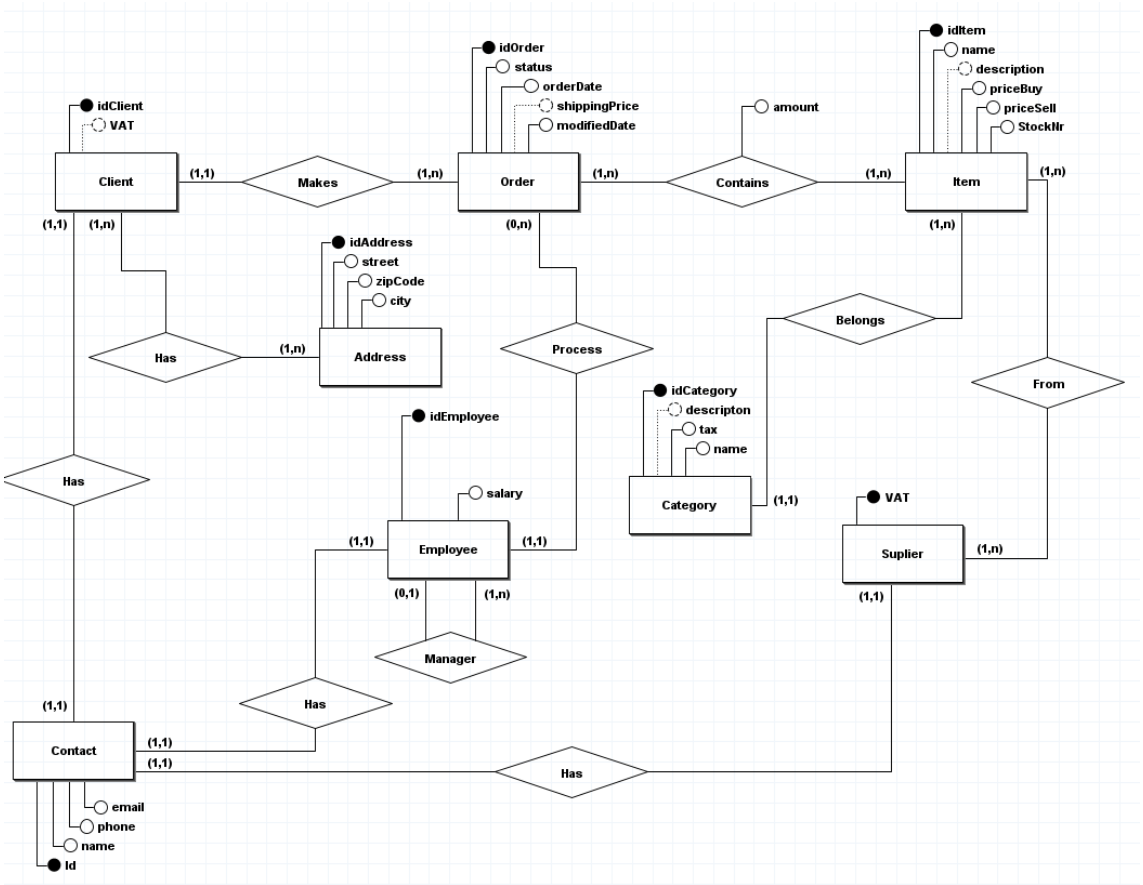

Figura 10 - Ilustração do Modelo Conceptual.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Para se dar conversão do modelo de dados conceptual para o modelo de dados lógico fizemos o agrupamento de cada entidade com os seus atributos numa tabela. Após este procedimento, prestou-se grande atenção às regras de derivação dos relacionamentos do modelo conceptual, concluindo-se assim a construção do modelo lógico.

Chave Primária	Atributos	Tipo	Chave Estrangeira
IdClient	idClient	INT	contact
	VAT	INT	
	contact	INT	

Tabela 6 - Ilustração referente à entidade Client.

Chave Primária	Atributo	Tipo
IdContact	idContact	INT
	name	VARCHAR (255)
	email	VARCHAR (255)
	phone	INT

Tabela 7 - Ilustração referente à entidade Contact.

Chave Primária	Atributo	Tipo
IdAdress	idAdress	INT
	name	VARCHAR (255)
	zipCode	VARCHAR (10)
	city	VARCHAR (45)

Tabela 8 - Ilustração referente à entidade Address.

Chave Primária	Atributo	Tipo	Chaves Estrangeiras
IdEmployee	idEmployee	INT	manager
	manager	INT	
	zipCode	DECIMAL (6,2)	contact
	contact	INT	

Tabela 9 - Ilustração referente à entidade Employee.

Chave Primária	Atributo	Tipo	Chave Estrangeira
VAT	VAT	INT	contact
	contact	INT	

Tabela 10 - Ilustração referente à entidade Suplier.

Chave Primária	Atributo	Tipo
idCategory	idCategory	INT
	name	VARCHAR (45)
	descripton	VARCHAR (255)
	tax	DECIMAL (6,2)

Tabela 11 - Ilustração referente à entidade Category.

Chave Primária	Atributo	Tipo	Chaves Estrangeiras
idItem	idItem	INT	category
	Name	VARCHAR (45)	
	descripton	VARCHAR (255)	
	stockNr	INT	
	priceBuy	DECIMAL (6,2)	
	PriceSell	DECIMAL (6,2)	
	category	INT	

Tabela 12 - Ilustração referente à entidade Item.

Chave Primária	Atributo	Tipo	Chaves Estrangeiras
idOrder	idOrder	INT	Client_idClient
	Status	VARCHAR (200)	
	shippingPrice	DECIMAL (6,2)	
	orderDate	DATETIME	
	upDate	DATETIME	Employee
	Employee	INT	
	Client_idClient	INT	

Tabela 13 - Ilustração referente à entidade Order.

A relação de N para N entre Client e Address origina a tabela:

Atributo	Tipo	Chaves Estrangeiras
Client_idClient	INT	Client_idClient
Adress_idAdress	INT	Adress_idAdress

Tabela 14 - Ilustração referente à relação Client e Adress.

A relação de N para N entre Suplier e Item origina a tabela:

Atributo	Tipo	Chaves Estrangeiras
Item_idItem	INT	Item_idItem
Suplier_VAT	INT	Suplier_VAT

Tabela 15 - Ilustração referente à relação Suplier e Item.

A relação de N para N entre Order e Item origina a tabela:

Atributo	Tipo	Chaves Estrangeiras
Order_idOrder	INT	Order_idOrder
Item_idItem	INT	Item_idItem
amount	INT	

Tabela 16 - Ilustração referente à relação Order_has_Item.

4.2. Desenho do modelo lógico

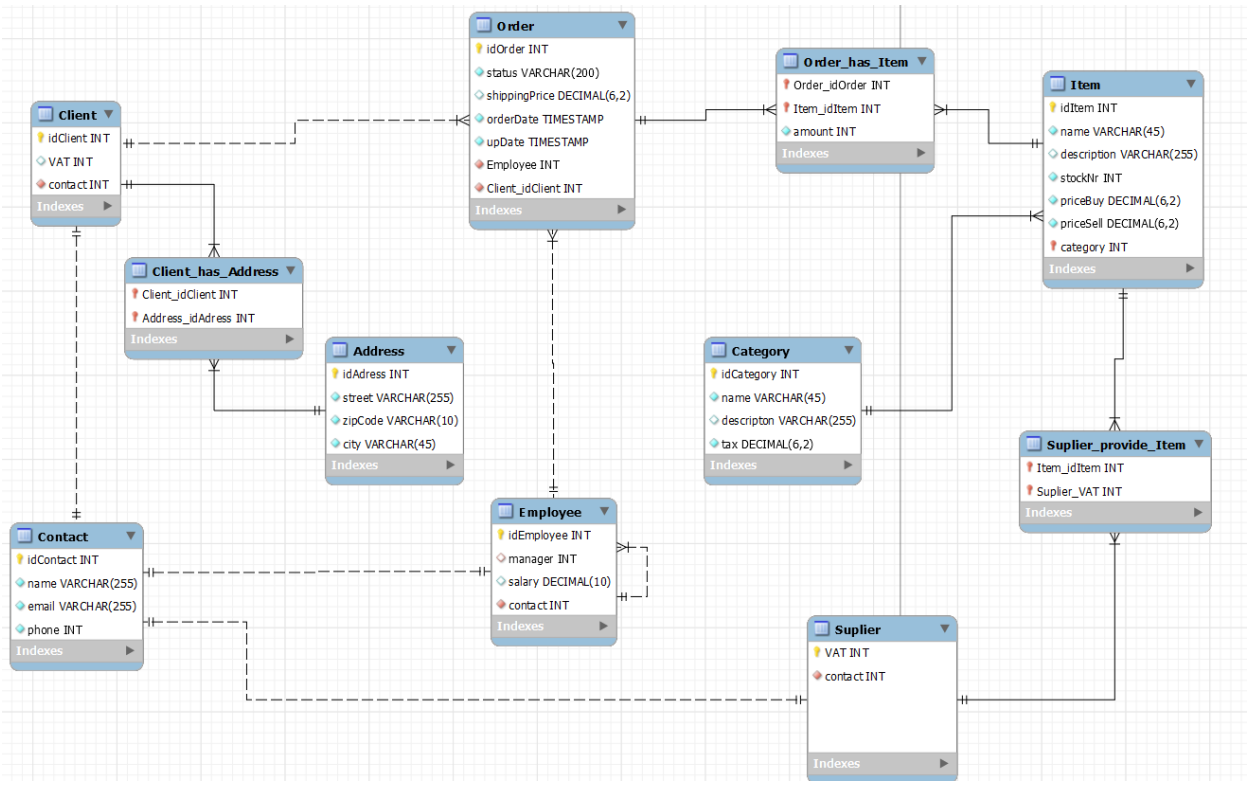

Figura 11 - Ilustração do Modelo Lógico.

4.3. Validação do modelo com interrogações do utilizador

Como forma de validação do modelo lógico, testamos se seria possível satisfazer as interrogações definidas sob forma de requisitos de exploração. Para tal, procedeu-se à transformação de algumas interrogações para álgebra relacional de modo que a ferramenta RelaX – Relational Algebra Calculator as conseguisse intrepertar. Mas este programa possui algumas limitações no que toca à disponibilização de operadores de lógica extendida, o que acabou por limitar a quantidade de interrogações que conseguimos modelar, visto que muitos operadores disponibilizados pelo MySQL não tinham equivalente lógico no RelaX.

```
-- Creating Clients View
-- 3- Listing orders
DROP PROCEDURE orders_on_year_month
    -- 3.1 - List all orders of a certain year in a certain month
DELIMITER $$
CREATE PROCEDURE orders_on_year_month(year_nr INT, month_nr INT)
BEGIN
SELECT * FROM dorlux.order
    WHERE year(orderDate) = year_nr AND month(orderDate) = month_nr;
END $$
CALL orders_on_year_month(2023, 1);
```

(σyear(orderDate)) = year_nr ∧ month(orderDate) = month_nr (Order))

```
DELIMITER $$
CREATE PROCEDURE list_order_in_category_status(category_nr INT, status_name VARCHAR(200))
BEGIN
SELECT * FROM
    dorlux.order AS Orders INNER JOIN order_has_Item AS Ord_has_item
        ON Orders.idOrder = Ord_has_item.Order_idOrder
    INNER JOIN Item AS It
        ON It.idItem = Ord_has_item.Item_idItem
    WHERE category = category_nr  AND status = status_name;
END
$$
```

σcategory = category_nr ∧ status = status_name (A)

A ← (Order ⋈idOrder=idOrder Order_has_Item) ⋈idItem=idItem Item

5. Implementação Física

5.1. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

Uma vez que o sistema de gestão de bases de dados utilizado durante as aulas foi o *MySQL*, decidimos que também deveríamos adotá-lo para a implementação deste projeto, já que estávamos mais familiarizados com o seu funcionamento e consideramos que é uma ferramenta bastante intuitiva, uma vez que foi utilizado durante as aulas.

O *MySQL* é um Sistema de Gestão de Bases de Dados *open-source* que assenta sobre um paradigma relacional. Sendo um dos sistemas mais utilizados em todo o mundo, devido à sua facilidade de utilização e elevado desempenho.

No MySQL, tivemos também a oportunidade de usufruir de mecanismos como o Forward Engineering, que, dado o modelo lógico da Base de Dados, produz o código da criação do sistema.

O *engine* utilizado será o *InnoDB*.

- **Contact**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Contact`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Contact` (
  `idContact` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `phone` INT NOT NULL,
  PRIMARY KEY (`idContact`))
ENGINE = InnoDB;
```

Figura 12 - Código SQL para criar tabela Contact.

- **Client**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Client`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Client` (
  `idClient` INT NOT NULL,
  `VAT` INT NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`idClient`),
  INDEX `fk_Client_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Client_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Figura 13 - Código SQL para criar tabela Client.

- **Employee**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Employee` (
  `idEmployee` INT NOT NULL,
  `manager` INT NULL,
  `salary` DECIMAL(10) NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`idEmployee`, `contact`),
  INDEX `fk_Employee_Employee1_idx` (`manager` ASC) VISIBLE,
  INDEX `fk_Employee_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Employee_Employee1`
    FOREIGN KEY (`manager`)
    REFERENCES `dorlux`.`Employee` (`idEmployee`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Employee_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 14 - Código SQL para criar tabela Employee.

- **Adress**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Address` (
  `idAdress` INT NOT NULL,
  `street` VARCHAR(255) NOT NULL,
  `zipCode` VARCHAR(10) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idAdress`))
ENGINE = InnoDB;
```

Figura 15 - Código SQL para criar tabela Address.

- **Category**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Category`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Category` (
  `idCategory` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `descripton` VARCHAR(255) NULL,
  `tax` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`idCategory`));
```

Figura 16 - Código SQL para criar tabela Category.

- **Item**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Item` (
  `idItem` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(255) NULL,
  `stockNr` INT NOT NULL,
  `priceBuy` DECIMAL(6,2) NOT NULL,
  `priceSell` DECIMAL(6,2) NOT NULL,
  `category` INT NOT NULL,
  PRIMARY KEY (`idItem`, `category`),
  INDEX `fk_Item_category1_idx` (`category` ASC) VISIBLE,
  CONSTRAINT `fk_Item_category1`
    FOREIGN KEY (`category`)
    REFERENCES `dorlux`.`Category` (`idCategory`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 17 - Código SQL para criar tabela Item.

- **Suplier**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Suplier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Suplier` (
  `VAT` INT NOT NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`VAT`),
  INDEX `fk_Suplier_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Suplier_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 18 - Código SQL para criar tabela Suplier.

- **Order**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Order` (
  `idOrder` INT NOT NULL,
  `status` VARCHAR(200) NOT NULL,
  `shippingPrice` DECIMAL(6,2) NULL,
  `orderDate` TIMESTAMP NOT NULL,
  `upDate` TIMESTAMP NOT NULL,
  `Employee` INT NOT NULL,
  `Client_idClient` INT NOT NULL,
  PRIMARY KEY (`idOrder`),
  INDEX `fk_Order_Employee1_idx` (`Employee` ASC) VISIBLE,
  INDEX `fk_Order_Client1_idx` (`Client_idClient` ASC) VISIBLE,
  CONSTRAINT `fk_Order_Employee1`
    FOREIGN KEY (`Employee`)
    REFERENCES `dorlux`.`Employee` (`idEmployee`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order_Client1`
    FOREIGN KEY (`Client_idClient`)
    REFERENCES `dorlux`.`Client` (`idClient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 19 - Código SQL para criar tabela Order.

- **Suplier_provide_Item**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Suplier_provide_Item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Suplier_provide_Item` (
  `Item_idItem` INT NOT NULL,
  `Suplier_VAT` INT NOT NULL,
  PRIMARY KEY (`Item_idItem`, `Suplier_VAT`),
  INDEX `fk_Item_has_Suplier_Suplier1_idx` (`Suplier_VAT` ASC) VISIBLE,
  INDEX `fk_Item_has_Suplier_Item1_idx` (`Item_idItem` ASC) VISIBLE,
  CONSTRAINT `fk_Item_has_Suplier_Item1`
    FOREIGN KEY (`Item_idItem`)
    REFERENCES `dorlux`.`Item` (`idItem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Item_has_Suplier_Suplier1`
    FOREIGN KEY (`Suplier_VAT`)
    REFERENCES `dorlux`.`Suplier` (`VAT`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 20 - Código SQL para criar tabela Suplier_provide_Item.

- **Client_has_Adress**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Client_has_Address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Client_has_Address` (
  `Client_idClient` INT NOT NULL,
  `Address_idAdress` INT NOT NULL,
  PRIMARY KEY (`Client_idClient`, `Address_idAdress`),
  INDEX `fk_Client_has_Address_Address1_idx` (`Address_idAdress` ASC) VISIBLE,
  INDEX `fk_Client_has_Address_Client1_idx` (`Client_idClient` ASC) VISIBLE,
  CONSTRAINT `fk_Client_has_Address_Client1`
    FOREIGN KEY (`Client_idClient`)
    REFERENCES `dorlux`.`Client` (`idClient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Client_has_Address_Address1`
    FOREIGN KEY (`Address_idAdress`)
    REFERENCES `dorlux`.`Address` (`idAdress`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 21 - Código SQL para criar tabela Client_has_Adress.

- **Order_has_Item**

```
-- -----------------------------------------------------
-- Table `dorlux`.`Order_has_Item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Order_has_Item` (
  `Order_idOrder` INT NOT NULL,
  `Item_idItem` INT NOT NULL,
  `amount` INT NOT NULL,
  PRIMARY KEY (`Order_idOrder`, `Item_idItem`),
  INDEX `fk_Order_has_Item_Item1_idx` (`Item_idItem` ASC) VISIBLE,
  INDEX `fk_Order_has_Item_Order1_idx` (`Order_idOrder` ASC) VISIBLE,
  CONSTRAINT `fk_Order_has_Item_Order1`
    FOREIGN KEY (`Order_idOrder`)
    REFERENCES `dorlux`.`Order` (`idOrder`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order_has_Item_Item1`
    FOREIGN KEY (`Item_idItem`)
    REFERENCES `dorlux`.`Item` (`idItem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 22 - Código SQL para criar tabela Order_has_Item.

5.2. Tradução das interrogações do utilizador para SQL (alguns exemplos)

```
# RM-13 - Orders items by stock number
SELECT * FROM Item
    ORDER BY stockNr DESC
```

Figura 23 - Código SQL para RM13.

idItem	name	description	stockNr	priceBuy	priceSell	category
13	Stickers	Stickers with funny/cute stuff	103	3.00	5.00	3
1	Teddy Bear, small	Toy to cuddle,	20	1.00	3.00	1
3	Birthday Card Nr. 1	A Card with happy birthday wishes.	20	0.10	1.00	3
14	Rubics Cube	Classic phisical riddle	20	10.00	15.00	3
15	Umbrella	Protection agains rain	18	6.00	9.00	3
4	Doll 1	A little toll for little kids	13	3.00	6.00	1
5	Doll 2	A little toll for little kids	13	3.00	6.00	1
6	Doll 3	A little toll for little kids	12	3.00	6.00	1
2	Black Pencil	A Pencile which is Black	10	0.54	2.00	2
8	Teddy Bear, XL	Toy to cuddle,	9	6.00	10.00	1
10	School Bag 1	A bag for studens	5	10.00	15.00	2
9	Teddy Bear, XXL	Toy to cuddle,	4	10.00	14.00	1
11	School Bag 2	A bag for studens	4	10.00	15.00	2
7	Teddy Bear, medium	Toy to cuddle	3	3.00	5.00	1
12	School Bag 3	A bag for studens	2	10.00	15.00	2

Figura 24 – Resultado da Query relativa RM13.

```sql
# RM08 - List all orders attached to a certain Employee
DELIMITER $$
CREATE PROCEDURE list_orders_attached_to_employee(employee_id INT)
BEGIN
SELECT idOrder, status, shippingPrice, orderDate, idEmployee, salary FROM
    dorlux.order AS Orders INNER JOIN employee as Emp
        ON Orders.Employee = Emp.idEmployee
    WHERE idEmployee = employee_id;
END
$$
CALL list_orders_attached_to_employee(2)
```

Figura 25 - Código SQL para RM08.

idOrder	status	shippingPrice	orderDate	idEmployee	salary
4	PENDING	3.43	2023-01-16 18:23:00	2	1000
6	PROCESSED	0.43	2023-01-16 18:23:00	2	1000
9	PROCESSED	2.10	2023-01-16 18:23:00	2	1000
12	PROCESSED	5.58	2023-01-16 18:23:00	2	1000
15	DELIVERED	1.39	2023-01-16 18:23:00	2	1000
17	DELIVERED	5.75	2023-01-16 18:23:00	2	1000
27	PENDING	5.62	2023-01-16 18:23:00	2	1000
28	PROCESSED	4.19	2023-01-16 18:23:00	2	1000
30	PENDING	2.23	2023-01-16 18:23:00	2	1000
33	PROCESSED	3.94	2023-01-16 18:23:00	2	1000
37	PROCESSED	2.10	2023-01-16 18:23:00	2	1000
40	PENDING	5.55	2023-01-16 18:23:00	2	1000
41	DELIVERED	3.66	2023-01-16 18:23:00	2	1000
45	PROCESSED	3.52	2023-01-16 18:23:00	2	1000
46	PROCESSED	3.88	2023-01-16 18:23:00	2	1000
47	DELIVERED	4.48	2023-01-16 18:23:00	2	1000
48	PENDING	1.98	2023-01-16 18:23:00	2	1000

Figura 26 – Resultado da Query relativa RM08.

```sql
# RM09 - What orders a client have made ?
DELIMITER $$
CREATE PROCEDURE orders_of_client(idClient_nr INT)
BEGIN
SELECT
    cont.name AS UserName,
    tab_ord.idOrder AS idOrder,
    tab_ord.status AS Status
    FROM
dorlux.Order AS tab_ord INNER JOIN Order_has_Item AS ord_has_it
        ON tab_ord.idOrder = ord_has_it.order_idOrder
    INNER JOIN Item AS it
        ON ord_has_it.Item_idItem = it.idItem
    INNER JOIN dorlux.Client AS Cl
        On tab_ord.Client_idClient = Cl.idClient
    INNER JOIN contact AS cont
        ON Cl.contact = cont.idContact
    WHERE Cl.idClient = idClient_nr;
END
$$
CALL orders_of_client(1)
```

Figura 27 - Código SQL para RM09.

UserName	idOrder	Status
Anita Barros	33	PROCESSED
Anita Barros	37	PROCESSED
Anita Barros	40	PENDING

Figura 28 – Resultado da Query relativa RM09.

```
# RM11 - How many items do we bought from a supplier
DELIMITER $$
CREATE FUNCTION  fuItemsCompradosForn(vatForn INT)
         RETURNS INT
         DETERMINISTIC
BEGIN
    DECLARE numeroitems INT;
    SELECT COUNT(I.idItem) INTO numeroitems FROM item AS I
         INNER JOIN suplier_provide_item AS SPI
             ON I.idItem = SPI.Item_idItem
         INNER JOIN suplier AS S
             ON SPI.Suplier_VAT = S.VAT
         WHERE S.VAT = vatForn;
    RETURN numeroitems;
END $$
SELECT fuItemsCompradosForn(795294490)
```

Figura 29 - Código SQL para RM11.

fuItemsCompradosForn(795294490)
4

Figura 30 – Resultado da Query relativa RM11.

5.3. Definição e caracterização das vistas de utilização em SQL (alguns exemplos)

```sql
-- Creating Clients View
CREATE VIEW Clients
AS
    SELECT idClient AS "Id", name AS "Name",  phone AS "PhoneNr", street AS "Street", zipCode AS "ZipCode", city AS "City"
    FROM client AS cl INNER JOIN contact AS cont
        ON cl.contact = cont.idContact
    INNER JOIN Client_has_Address AS Cl_has_addr
        ON cl.idClient = Cl_has_addr.Client_idClient
    INNER JOIN Address AS Addr
        ON Cl_has_addr.Address_idAdress = Addr.idAdress
    ORDER BY name ASC;

-- Select View Clients
SELECT * FROM Clients;

-- Drop View Clients
DROP VIEW Clients;
```

Figura 31 – Código SQL para a vista.

5.4. Cálculo do espaço das bases de dados (inicial e taxa de crescimento anual)

Para obtermos o espaço que ocupado pela nossa Base de Dados, é necessário calcularmos o espaço de cada entidade. Assim sendo, considerando que um Int corresponde a 4 bytes, um Date corresponde a 3 bytes e um Varchar (N) N+1 bytes caso a string esteja codificada em ASCII, 2*N + 1 caso a codificação seja mais complexa, mas o valor de 2*N seja menor que 255 e 2*N + 2 caso 2*N maior que 255.**

Tipo	Tamanho
Int	4 bytes
Float	4 bytes
Datetime	5 bytes
Date	3 bytes
Decimal (X, Y)	8 bytes*
Varchar (N)	N+1 ou 2xN+1 ou 2xN+2**

Tabela 17 - Descrição dos tamanhos consoante o tipo.

*Quando X – Y = 9 e Y <9, para valores superiores devemos consultar a tabela disponível na secção 11.7 do manual.

Atributo	Tipo	Tamanho
idClient	INT	4 bytes
VAT	INT	4 bytes
contact	INT	4 bytes

Tabela 18 – Tamanho (em bytes) dos atributos do Client.

Atributo	Tipo	Tamanho
idContact	INT	4 bytes
name	VARCHAR (255)	512 bytes
email	VARCHAR (255)	512 bytes
phone	INT	4 bytes

Tabela 19 – Tamanho (em bytes) dos atributos do Contact.

Atributo	Tipo	Tamanho
idAdress	INT	4 bytes
street	VARCHAR (255)	512 bytes
zipCode	VARCHAR (10)	21 bytes
city	VARCHAR (45)	91 bytes

Tabela 20 - Tamanho (em bytes) dos atributos do Address.

Atributo	Tipo	Tamanho
idEmployee	INT	4 bytes
manager	INT	4 bytes
salary	DECIMAL (6,2)	8 bytes
contact	INT	4 bytes

Tabela 21 - Tamanho (em bytes) dos atributos do Employee.

Atributo	Tipo	Tamanho
VAT	INT	4 bytes
contact	INT	4 bytes

Tabela 22 - Tamanho (em bytes) dos atributos do Suplier.

Atributo	Tipo	Tamanho
idCategory	INT	4 bytes
name	VARCHAR (45)	91 bytes
descripton	VARCHAR (255)	512 bytes
tax	DECIMAL (6,2)	8 bytes

Tabela 23 - Tamanho (em bytes) dos atributos da Category.

Atributo	Tipo	Tamanho
idItem	INT	4 bytes
Name	VARCHAR (45)	91 bytes
descripton	VARCHAR (255)	512 bytes
stockNr	INT	4 bytes
priceBuy	DECIMAL (6,2)	8 bytes
PriceSell	DECIMAL (6,2)	8 bytes
category	INT	4 bytes

Tabela 24 - Tamanho (em bytes) dos atributos do Item.

Atributo	Tipo	Tamanho
idOrder	INT	4 bytes
Status	VARCHAR (200)	402 bytes
shippingPrice	DECIMAL (6,2)	8 bytes
orderDate	DATETIME	5 bytes
upDate	DATETIME	5 bytes
Employee	INT	4 bytes
Client_idClient	INT	4 bytes

Tabela 25 - Tamanho (em bytes) dos atributos do Order.

Atributo	Tipo	Tamanho
Client_idClient	INT	4 bytes
Adress_idAdress	INT	4 bytes

Tabela 26 - Tamanho (em bytes) dos atributos da relação Client e Address.

Atributo	Tipo	Tamanho
Item_idItem	INT	4 bytes
Suplier_VAT	INT	4 bytes

Tabela 27 - Tamanho (em bytes) dos atributos da relação Suplier e Item.

Atributo	Tipo	Tamanho
Order_idOrder	INT	4 bytes
Item_idItem	INT	4 bytes
amount	INT	4 bytes

Tabela 28 - Tamanho (em bytes) dos atributos relação Order e Item.

Usando estes valores calculados, pode-se obter uma estimativa de espaço ocupado pela base de dados sem povoamento de 3312 bytes.

Tendo em conta que se trata de um sistema de entrega ao domicilio é de espera que exista um crescimento anual de clientes e consecutivamente de encomendas, por outro lado sabemos que vai existir mais crescimento nas encomendas, na base de dados.

Assumindo um crescimento de 13 % anual no número de clientes e tendo-se atualmente 30 clientes no sistema, o crescimento da base de dados será calculado por (30 * 0.13) * 1044 ≈ 4071,6 bytes, ou seja, é de esperar que num ano, relativamente aos clientes, a base de dados cresça 4.1 KB.

Quanto ao crescimento de encomendas estima-se que tenha um crescimento de 20 % anual (tendo em conta que os novos clientes e os eventuais clientes possam voltar a fazer compras), uma vez que existem atualmente 50 encomendas na base de dados o crescimento anual de encomendas é dado por (50*0.2) * 432 ≈ 4320 bytes, ou seja, durante um ano, relativamente às encomendas, é de esperar que a base de dados cresça 4.3KB.

Assim, uma boa estimativa do crescimento anual da base de dados, será somar o crescimento anual destas duas tabelas (visto serem as tabelas com maior projeção de crescimento). Então, esta estimativa é dada por 4.1KB + 4.3KB = 8.4KB ou seja, 8.4KB de crescimento anual.

Quanto às restantes tabelas, o seu crescimento não será denso o suficiente de modo a ter um impacto significativo na base de dados.

5.5. Plano de segurança e recuperação de dados

No que diz respeito à matéria da gestão de dados, precaver é a palavra-chave, uma vez que qualquer perda de dados pode dar aso a falhas no negócio. Desta forma, propomos várias ações e diretrizes que visam garantir o bom funcionamento da Base de Dados e, simultaneamente, diminuir o impacto de possíveis danos causados por variados motivos (por exemplo, ataques cibernéticos ou o próprio erro humano). Destacamos:

- Elaborar um documento com a listagem das prioridades no que toca à informação que se encontrava na Base de Dados;
- Elaborar de um documento onde constam os valores máximos de tolerância face à destruição de dados, assim como os principais meios afetados por este incidente;
- Guardar os sripts de criação e povoamento, por forma a amortizar os danos na eventualidade de a Base de Dados ter sido apagada;
- Proceder a Backups Periódicos numa Cloud fornecida por terceiros ou para um sistema físico dotado de um aglomerado de discos rígidos.
- Elaborar uma cópia de segurança num repositório privado do GitHub, o que permite um controlo de versões, garantindo também que todos os dados estão salvaguardados online.

6. Conclusões e Trabalho Futuro

Durante a realização deste trabalho sentimos um grande crescimento no que toca a conhecimento inerente às matérias relativas a bases de dados. Percebemos também que a recolha de requisitos é essencial para uma solução estruturada e correta. O facto deste trabalho ser baseado numa necessidade real duma pessoa fez com que abordássemos o problema duma maneira especial e com um toque de classe. Para finalizar, vamos enumerar alguns pontos que podem ser melhorados numa futura versão desta base de dados. A realização de mais queries será, claramente, algo a melhorar visto que ainda há muita margem de exploração da base de dados, por exemplo, saber qual a faturação realizada ao fim de cada dia. O desenvolvimento de gatilhos é também um ponto que gostaríamos de abordar visto que estes são essenciais para o bom funcionamento da base de dados. Em termos de desempenho, haverá sempre pontos a melhorar, quer seja através da gestão de índices ou através de alterações nos modelos criados.

Concluindo, o objetivo do projeto foi alcançado uma vez que cumprimos todos os pontos propostos e, principalmente, por termos oferecido à Dona Dores uma ferramenta que a vai ajudar no seu dia a dia e que vai, certamente, tornar a sua vida melhor, visto que, assim, a sua loja online passa a ser uma realidade mais provável que era o principal objetivo deste projeto.

Referências

RelaX - https://dbis-uibk.github.io/relax/landing

Connolly, T., Begg, C., 2014. Database Systems: A Practical Approach to Design, Implementation, and Management. Addison-Wesley, Global Edition.

Belo, O., 2021. Bases de Dados Relacionais: Implementação com MySQL. FCA, Editora de Informática.

Gouveia, F., 2021. Bases de Dados - Fundamentos e Aplicações. 2ª Ed. FCA, Editora de Informática.

MySQL, 2021. Reference Manual. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/ [Accessed 10 of January of 2023]

Lista de Siglas e Acrónimos

<<Apresentar uma lista com todas as siglas e acrónimos utilizados durante a realização do trabalho. O formato base para esta lista deverá ser da forma como abaixo se apresenta.>>

BD	Base de Dados
OLTP	*On-Line Analytical Processing*
SQL	*Structured Query Language*
ER	*Entity-Relationship*
D.	Dona
Sr.	Senhor

Anexos

I. Anexo 1 – *Script* de Criação da Base de Dados

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema dorlux
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema dorlux
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `dorlux` DEFAULT CHARACTER SET utf8 ;
-- -----------------------------------------------------

USE `dorlux` ;

-- -----------------------------------------------------
-- Table `dorlux`.`Contact`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Contact` (
  `idContact` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `phone` INT NOT NULL,
  PRIMARY KEY (`idContact`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `dorlux`.`Client`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Client` (
  `idClient` INT NOT NULL,
  `VAT` INT NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`idClient`),
  INDEX `fk_Client_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Client_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Employee` (
  `idEmployee` INT NOT NULL,
  `manager` INT NULL,
  `salary` DECIMAL(10) NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`idEmployee`, `contact`),
  INDEX `fk_Employee_Employee1_idx` (`manager` ASC) VISIBLE,
  INDEX `fk_Employee_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Employee_Employee1`
    FOREIGN KEY (`manager`)
    REFERENCES `dorlux`.`Employee` (`idEmployee`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Employee_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `dorlux`.`Order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Order` (
  `idOrder` INT NOT NULL,
  `status` VARCHAR(200) NOT NULL,
  `shippingPrice` DECIMAL(6,2) NULL,
  `orderDate` TIMESTAMP NOT NULL,
  `upDate` TIMESTAMP NOT NULL,
  `Employee` INT NOT NULL,
  `Client_idClient` INT NOT NULL,
  PRIMARY KEY (`idOrder`),
  INDEX `fk_Order_Employee1_idx` (`Employee` ASC) VISIBLE,
  INDEX `fk_Order_Client1_idx` (`Client_idClient` ASC) VISIBLE,
  CONSTRAINT `fk_Order_Employee1`
    FOREIGN KEY (`Employee`)
    REFERENCES `dorlux`.`Employee` (`idEmployee`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order_Client1`
    FOREIGN KEY (`Client_idClient`)
    REFERENCES `dorlux`.`Client` (`idClient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Category`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Category` (
  `idCategory` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `descripton` VARCHAR(255) NULL,
  `tax` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`idCategory`));


-- -----------------------------------------------------
-- Table `dorlux`.`Item`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Item` (
  `idItem` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(255) NULL,
  `stockNr` INT NOT NULL,
  `priceBuy` DECIMAL(6,2) NOT NULL,
  `priceSell` DECIMAL(6,2) NOT NULL,
  `category` INT NOT NULL,
  PRIMARY KEY (`idItem`, `category`),
  INDEX `fk_Item_category1_idx` (`category` ASC) VISIBLE,
  CONSTRAINT `fk_Item_category1`
    FOREIGN KEY (`category`)
    REFERENCES `dorlux`.`Category` (`idCategory`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Suplier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Suplier` (
  `VAT` INT NOT NULL,
  `contact` INT NOT NULL,
  PRIMARY KEY (`VAT`),
  INDEX `fk_Suplier_Contact1_idx` (`contact` ASC) VISIBLE,
  CONSTRAINT `fk_Suplier_Contact1`
    FOREIGN KEY (`contact`)
    REFERENCES `dorlux`.`Contact` (`idContact`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Suplier_provide_Item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Suplier_provide_Item` (
  `Item_idItem` INT NOT NULL,
```

```
  `Suplier_VAT` INT NOT NULL,
  PRIMARY KEY (`Item_idItem`, `Suplier_VAT`),
  INDEX `fk_Item_has_Suplier_Suplier1_idx` (`Suplier_VAT` ASC) VISIBLE,
  INDEX `fk_Item_has_Suplier_Item1_idx` (`Item_idItem` ASC) VISIBLE,
  CONSTRAINT `fk_Item_has_Suplier_Item1`
    FOREIGN KEY (`Item_idItem`)
    REFERENCES `dorlux`.`Item` (`idItem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Item_has_Suplier_Suplier1`
    FOREIGN KEY (`Suplier_VAT`)
    REFERENCES `dorlux`.`Suplier` (`VAT`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Address` (
  `idAdress` INT NOT NULL,
  `street` VARCHAR(255) NOT NULL,
  `zipCode` VARCHAR(10) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idAdress`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Client_has_Address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Client_has_Address` (
  `Client_idClient` INT NOT NULL,
  `Address_idAdress` INT NOT NULL,
  PRIMARY KEY (`Client_idClient`, `Address_idAdress`),
  INDEX `fk_Client_has_Address_Address1_idx` (`Address_idAdress` ASC) VISIBLE,
  INDEX `fk_Client_has_Address_Client1_idx` (`Client_idClient` ASC) VISIBLE,
  CONSTRAINT `fk_Client_has_Address_Client1`
    FOREIGN KEY (`Client_idClient`)
```

```
    REFERENCES `dorlux`.`Client` (`idClient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Client_has_Address_Address1`
    FOREIGN KEY (`Address_idAdress`)
    REFERENCES `dorlux`.`Address` (`idAdress`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `dorlux`.`Order_has_Item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dorlux`.`Order_has_Item` (
  `Order_idOrder` INT NOT NULL,
  `Item_idItem` INT NOT NULL,
  `amount` INT NOT NULL,
  PRIMARY KEY (`Order_idOrder`, `Item_idItem`),
  INDEX `fk_Order_has_Item_Item1_idx` (`Item_idItem` ASC) VISIBLE,
  INDEX `fk_Order_has_Item_Order1_idx` (`Order_idOrder` ASC) VISIBLE,
  CONSTRAINT `fk_Order_has_Item_Order1`
    FOREIGN KEY (`Order_idOrder`)
    REFERENCES `dorlux`.`Order` (`idOrder`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order_has_Item_Item1`
    FOREIGN KEY (`Item_idItem`)
    REFERENCES `dorlux`.`Item` (`idItem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

II. Anexo 1 – *Script* de povoação da Base de Dados

INSERT INTO Address (idAdress, street, zipCode, city)

VALUES

('1', 'Avenida de Abreu, 7', '4345-563', 'Lamego'),

('2', 'Alameda das Galeotas ao Parque das Na��es, 94', '3568-484', 'Santiago do Cac�m'),

('3', 'Pra�a de Marques, 65', '7642-981', 'Mirandela'),

('4', 'Avenida Em�lia Nunes, 27', '8791-324', 'Loul�'),

('5', 'Travessa de S. L�zaro, 611', '4910-335', 'Lourosa'),

('6', 'Travessa do Duro, 7', '0153-798', 'Lamego'),

('7', 'Alameda Ferreira, 7', '3256-661', 'Lisboa'),

('8', 'Rua Prof. Ant�nio Jos� Saraiva, 89', '7713-892', 'Odivelas'),

('9', 'Alameda de Gon�alves, S/N', '4857-708', 'S�o Mamede de Infesta');

INSERT INTO Contact (idContact, name, email, phone)

VALUES

('1', 'N�ria Brito', 'ysilva@example.org', '921805998'),

('2', 'Gabriela Ara�jo', 'baptistaadriana@example.com', '960923449'),

('3', 'Juliana Moreira', 'figueiredosimao@example.net', '916605242'),

('4', 'Pedro Lopes', 'goncalo75@example.net', '959794195'),

('5', 'Leonor S�', 'ivo24@example.com', '953352205'),

('6', 'M�rcio Vicente', 'margarida98@example.com', '954443287'),

('7', 'Beatriz da Pinheiro', 'iaraneves@example.com', '910743094'),

('8', 'Vera Paiva', 'zloureiro@example.org', '939644358'),

('9', 'Adriana Pacheco', 'soaressebastiao@example.org', '943779260'),

('10', 'Renata Machado', 'gsimoes@example.com', '912734665'),

('11', 'Petra Vicente', 'mgoncalves@example.net', '955810629'),

('12', 'L�cia Teixeira-S�', 'magalhaesnoah@example.org', '950322870'),

('13', 'M�rcio Cardoso', 'ismael70@example.com', '912478583'),

('14', 'Mafalda-Luciana Lopes', 'dmachado@example.com', '968467316'),

('15', 'Fabiana-Flor Henriques', 'raquel88@example.org', '912486414'),

('16', 'Tom�s Castro', 'saravalente@example.com', '914263744'),

('17', 'Carminho da Amorim', 'pachecomarcos@example.com', '954682866'),
('18', 'Mauro Campos-Vieira', 'brunaabreu@example.com', '923882213'),
('19', '�ris do Maia', 'soarescarolina@example.org', '951758184'),
('20', 'Bruna-Carlota Lima', 'micael71@example.net', '935518474'),
('21', 'Daniela Ferreira', 'rodrigomacedo@example.org', '960573639'),
('22', 'Francisco Figueiredo-Almeida', 'liamendes@example.com', '963270468'),
('23', 'N�ria Correia', 'vfonseca@example.net', '939952145'),
('24', 'V�tor Teixeira', 'coelhoangelo@example.net', '915738969'),
('25', 'Adriana Gomes', 'gramos@example.com', '963973244'),
('26', 'Mia Domingues', 'amorimluana@example.com', '940887616'),
('27', 'David Loureiro-Paiva', 'martim36@example.com', '940418528'),
('28', 'Erika Barros', 'macedolorena@example.com', '914957774'),
('29', 'Larissa Campos', 'qpinto@example.com', '932748652'),
('30', 'Anita Barros', 'xtorres@example.org', '929793833'),
('31', 'Matias Moreira', 'cbatista@example.org', '930112868'),
('32', 'Nuno Nogueira', 'jbatista@example.net', '943888356'),
('33', 'Igor Martins', 'lisandro71@example.org', '953456357'),
('34', 'M�rcio Machado', 'vazantonio@example.net', '916861243'),
('35', 'Brian Pinto', 'dnogueira@example.org', '918188808'),
('36', 'Leandro Borges', 'netomariana@example.org', '942785172'),
('37', 'Raquel Matos', 'bernardo05@example.net', '948310762'),
('38', 'Luna Santos', 'msantos@example.com', '943025262'),
('39', 'Jaime Marques', 'bmendes@example.com', '930420549'),
('40', 'Leonor Gaspar', 'nogueiraclara@example.com', '962073591'),
('41', 'Ariana Borges', 'joel00@example.com', '912742492'),
('42', 'Diana da Sim�es', 'piresmelissa@example.net', '925210125'),
('43', 'Diogo Vaz', 'bandrade@example.org', '934520468'),
('44', 'Eduardo Baptista', 'enogueira@example.net', '952690193');

INSERT INTO Category (idCategory, name, descripton, tax)
VALUES
('1', 'Toys', 'Toys mainly targeted at children.', '10'),
('2', 'School utilitys', 'Things students need is School, l', '5'),
('3', 'gifts for friends and family', 'Cards for Birthdays, Chrismas and so on', '9');

INSERT INTO Client (idClient, VAT, contact)
VALUES
(1, 551660888, 30),
(2, 101196568, 17),
(3, 980660414, 20),

```
    (4, 471652219, 10),
    (5, 524268937, 29),
    (6, 372595589, 26),
    (7, 307464196, 27),
    (8, 276971367, 22),
    (9, 331728268, 21),
    (10, 322735884, 25),
    (11, 218405266, 13),
    (12, 341886077, 28),
    (13, 381023641, 12),
    (14, 465413312, 19),
    (15, 974584613, 11),
    (16, 937695194, 1),
    (17, 785898283, 4),
    (18, 508785024, 15),
    (19, 953683195, 3),
    (20, 257134780, 14),
    (21, 884649266, 23),
    (22, 881856917, 2),
    (23, 574317553, 6),
    (24, 214720980, 5),
    (25, 847132177, 9),
    (26, 527469341, 8),
    (27, 256468436, 18),
    (28, 348205377, 16),
    (29, 678658091, 7),
    (30, 851632038, 24);

INSERT INTO Client_has_Address (Client_idClient, Address_idAdress)
    VALUES
    (6, 8),
    (1, 5),
    (8, 7),
    (4, 4),
    (9, 6),
    (5, 9),
    (7, 3),
    (2, 1),
    (3, 2),
    (10, 7),(14, 3),
```

```
(15, 1),
(12, 8),
(16, 7),
(13, 4),
(18, 2),
(17, 6),
(11, 9),
(19, 5),
(20, 7),(22, 1),
(26, 6),
(29, 8),
(21, 4),
(25, 7),
(27, 2),
(24, 3),
(23, 9),
(28, 5),
(30, 9);

INSERT INTO Item (idItem, name, description, stockNr, priceBuy, priceSell, category)
    VALUES
    ('1', 'Teddy Bear, small', 'Toy to cuddle,', '20', '1', '3', '1'),
    ('2', 'Black Pencil', 'A Pencile which is Black', '10', '0.54', '2', '2'),
    ('3', 'Birthday Card Nr. 1', 'A Card with happy birthday wishes.', '20', '0.1', '1', '3'),
    ('4', 'Doll 1', 'A little toll for little kids', '13', '3.0', '6', '1'),
    ('5', 'Doll 2', 'A little toll for little kids', '13', '3.0', '6', '1'),
    ('6', 'Doll 3', 'A little toll for little kids', '12', '3.0', '6', '1'),
    ('7', 'Teddy Bear, medium', 'Toy to cuddle', '3', '3', '5', '1'),
    ('8', 'Teddy Bear, XL', 'Toy to cuddle,', '9', '6', '10', '1'),
    ('9', 'Teddy Bear, XXL', 'Toy to cuddle,', '4', '10', '14', '1'),
    ('10', 'School Bag 1', 'A bag for studens', '5', '10', '15', '2'),
    ('11', 'School Bag 2', 'A bag for studens', '4', '10', '15', '2'),
    ('12', 'School Bag 3', 'A bag for studens', '2', '10', '15', '2'),
    ('13', 'Stickers', 'Stickers with funny/cute stuff', '103', '3', '5', '3'),
    ('14', 'Rubics Cube', 'Classic phisical riddle', '20', '10', '15', '3'),
    ('15', 'Umbrella', 'Protection agains rain', '18', '6', '9', '3');

INSERT INTO Employee (idEmployee, manager, salary, contact) VALUES
    ('1', '1', '1500', '32'),
    ('2', '1', '1000', '33'),
```

```
        ('3', '1', '1000', '34');

INSERT INTO Suplier (VAT, contact) VALUES
        ('112456995', '35'),
        ('965338790', '36'),
        ('477761945', '37'),
        ('462953975', '38'),
        ('781938810', '39'),
        ('174793255', '40'),
        ('338248591', '41'),
        ('721395173', '42'),
        ('795294490', '43'),
        ('746118362', '44');

INSERT INTO Suplier_provide_Item (Suplier_VAT,Item_idItem) VALUES
        ('721395173', '1'),
        ('795294490', '2'),
        ('965338790', '3'),
        ('721395173', '4'),
        ('781938810', '5'),
        ('795294490', '6'),
        ('795294490', '7'),
        ('112456995', '8'),
        ('746118362', '9'),
        ('795294490', '10'),
        ('746118362', '11'),
        ('965338790', '12'),
        ('462953975', '13'),
        ('462953975', '14'),
        ('721395173', '15');

INSERT INTO dorlux.Order (idOrder, status, shippingPrice, orderDate, dorlux.Order.upDate,
Employee, client_idClient)
        VALUES
        ('1', 'DELIVERED', '1.72', NOW(), NOW(), '3', '25'),
        ('2', 'PENDING', '3.2', NOW(), NOW(), '3', '28'),
        ('3', 'PENDING', '0.92', NOW(), NOW(), '1', '3'),
        ('4', 'PENDING', '3.43', NOW(), NOW(), '2', '11'),
        ('5', 'PENDING', '4.97', NOW(), NOW(), '3', '24'),
        ('6', 'PROCESSED', '0.43', NOW(), NOW(), '2', '20'),
```

('7', 'PROCESSED', '3.38', NOW(), NOW(), '3', '12'),
('8', 'DELIVERED', '3.89', NOW(), NOW(), '1', '29'),
('9', 'PROCESSED', '2.1', NOW(), NOW(), '2', '9'),
('10', 'DELIVERED', '5.22', NOW(), NOW(), '1', '11'),
('11', 'DELIVERED', '3.82', NOW(), NOW(), '1', '6'),
('12', 'PROCESSED', '5.58', NOW(), NOW(), '2', '2'),
('13', 'DELIVERED', '3.64', NOW(), NOW(), '1', '17'),
('14', 'PENDING', '1.88', NOW(), NOW(), '1', '4'),
('15', 'DELIVERED', '1.39', NOW(), NOW(), '2', '30'),
('16', 'PENDING', '1.17', NOW(), NOW(), '1', '28'),
('17', 'DELIVERED', '5.75', NOW(), NOW(), '2', '23'),
('18', 'PENDING', '4.87', NOW(), NOW(), '1', '16'),
('19', 'DELIVERED', '2.40', NOW(), NOW(), '1', '3'),
('20', 'PENDING', '0.99', NOW(), NOW(), '3', '3'),
('21', 'PROCESSED', '0.79', NOW(), NOW(), '1', '19'),
('22', 'PENDING', '3.15', NOW(), NOW(), '1', '16'),
('23', 'DELIVERED', '1.27', NOW(), NOW(), '3', '29'),
('24', 'PROCESSED', '3.49', NOW(), NOW(), '1', '10'),
('25', 'DELIVERED', '5.6', NOW(), NOW(), '1', '12'),
('26', 'PENDING', '4.69', NOW(), NOW(), '1', '19'),
('27', 'PENDING', '5.62', NOW(), NOW(), '2', '15'),
('28', 'PROCESSED', '4.19', NOW(), NOW(), '2', '6'),
('29', 'PENDING', '2.42', NOW(), NOW(), '3', '17'),
('30', 'PENDING', '2.23', NOW(), NOW(), '2', '21'),
('31', 'PENDING', '0.5', NOW(), NOW(), '3', '17'),
('32', 'DELIVERED', '4.87', NOW(), NOW(), '3', '9'),
('33', 'PROCESSED', '3.94', NOW(), NOW(), '2', '1'),
('34', 'PENDING', '1.44', NOW(), NOW(), '1', '8'),
('35', 'DELIVERED', '3.66', NOW(), NOW(), '1', '28'),
('36', 'PROCESSED', '3.94', NOW(), NOW(), '1', '22'),
('37', 'PROCESSED', '2.1', NOW(), NOW(), '2', '1'),
('38', 'DELIVERED', '0.96', NOW(), NOW(), '3', '4'),
('39', 'DELIVERED', '4.16', NOW(), NOW(), '3', '2'),
('40', 'PENDING', '5.55', NOW(), NOW(), '2', '1'),
('41', 'DELIVERED', '3.66', NOW(), NOW(), '2', '26'),
('42', 'DELIVERED', '2.18', NOW(), NOW(), '1', '20'),
('43', 'DELIVERED', '1.78', NOW(), NOW(), '3', '4'),
('44', 'DELIVERED', '1.53', NOW(), NOW(), '1', '27'),
('45', 'PROCESSED', '3.52', NOW(), NOW(), '2', '9'),
('46', 'PROCESSED', '3.88', NOW(), NOW(), '2', '21'),

```
('47', 'DELIVERED', '4.48', NOW(), NOW(), '2', '21'),
('48', 'PENDING', '1.98', NOW(), NOW(), '2', '13'),
('49', 'PENDING', '3.29', NOW(), NOW(), '1', '3'),
('50', 'PENDING', '0.37', NOW(), NOW(), '1', '7');

INSERT INTO Order_has_Item (Order_idOrder, Item_idItem, amount) VALUES
('40', '6', '11'),
('5', '4', '6'),
('39', '10', '4'),
('24', '9', '4'),
('26', '11', '20'),
('47', '10', '13'),
('15', '1', '14'),
('28', '9', '15'),
('21', '4', '20'),
('38', '8', '10'),
('36', '14', '7'),
('13', '6', '19'),
('22', '13', '20'),
('31', '5', '1'),
('2', '13', '12'),
('19', '10', '17'),
('6', '9', '15'),
('41', '5', '19'),
('42', '11', '18'),
('7', '4', '18'),
('3', '15', '15'),
('46', '7', '1'),
('44', '9', '9'),
('43', '12', '20'),
('35', '6', '17'),
('4', '1', '3'),
('18', '3', '11'),
('32', '15', '8'),
('30', '10', '5'),
('34', '1', '19'),
('25', '9', '4'),
('49', '3', '13'),
('20', '11', '15'),
('23', '11', '17'),
```

('1', '13', '1'),
('9', '5', '9'),
('33', '3', '6'),
('14', '1', '18'),
('27', '10', '14'),
('29', '6', '9'),
('11', '1', '12'),
('10', '3', '1'),
('8', '11', '13'),
('48', '15', '4'),
('50', '1', '5'),
('12', '14', '15'),
('37', '1', '12'),
('45', '12', '16'),
('17', '1', '1'),
('16', '13', '2');

III. Anexo 1 – *Queries* de exploração da Base de Dados

```
# FETCH ALL DATA FROM THE DATABASE
USE dorlux;

SELECT * FROM dorlux.order;
SELECT * FROM category;
SELECT * FROM client;
SELECT * FROM address;
SELECT * FROM client_has_address;
SELECT * FROM contact;
SELECT * FROM employee;
SELECT * FROM item;
SELECT * FROM order_has_item;
SELECT * FROM suplier;
SELECT * FROM suplier_provide_item;

DROP SCHEMA dorlux;

# RM01 - Anything to Processed
# RM02 - Anything to Delivering
# RM03 - Anything to Delivered
# RM04 - Anything to Canceled
DELIMITER $$
CREATE PROCEDURE change_order_status(status_name VARCHAR(200), idOrder_nr INT)
BEGIN
UPDATE dorlux.order
   SET dorlux.order.status = status_name , dorlux.order.upDate = NOW()
   WHERE idOrder = idOrder_nr;
END $$
CALL change_order_status('DELIVERED', 2)
CALL change_order_status('PROCESSED', 3)
CALL change_order_status('CANCELED', 4)
CALL change_order_status('DELIVERING', 5)
```

```
-- RM5 - List all orders of a certain year in a certain month
DROP PROCEDURE orders_on_year_month
-- -------------------------------------------------------
DELIMITER $$
CREATE PROCEDURE orders_on_year_month(year_nr INT, month_nr INT)
BEGIN
SELECT * FROM dorlux.order
	WHERE year(orderDate) = year_nr AND month(orderDate) = month_nr;
END $$
CALL orders_on_year_month(2023, 1);

-- RM6 - List Orders of a especific category in a specific status
DROP PROCEDURE orders_on_year_month;
DELIMITER $$
CREATE PROCEDURE list_order_in_category_status(category_nr INT, status_name
VARCHAR(200))
BEGIN
SELECT * FROM
	dorlux.order AS Orders INNER JOIN order_has_Item AS Ord_has_item
		ON Orders.idOrder = Ord_has_item.Order_idOrder
	INNER JOIN Item AS It
		ON It.idItem = Ord_has_item.Item_idItem
	WHERE category = category_nr  AND status = status_name;
END
$$

CALL list_order_in_category_status(2, 'PENDING');
CALL list_order_in_category_status(1, 'PROCESSED');

-- RM07 - List all orders Processed with address attached
SELECT idOrder, status, orderDate, idClient, VAT, street, zipCode, city FROM
	dorlux.order AS Orders INNER JOIN dorlux.client AS Cl
		ON Orders.Client_idClient = Cl.idClient
	INNER JOIN Client_has_Address AS cl_has_addr
		ON cl_has_addr.Client_idClient = Cl.idClient
	INNER JOIN Address AS addr
		ON cl_has_addr.Address_idAdress = addr.idAdress
	WHERE status = 'PROCESSED';
```

```
# RM08 - List all orders attached to a certain Employee
DELIMITER $$
CREATE PROCEDURE list_orders_attached_to_employee(employee_id INT)
BEGIN
SELECT idOrder, status, shippingPrice, orderDate, idEmployee, salary FROM
        dorlux.order AS Orders INNER JOIN employee as Emp
                ON Orders.Employee = Emp.idEmployee
        WHERE idEmployee = employee_id;
END
$$
CALL list_orders_attached_to_employee(2)

# RM09 - What orders a client have made ?
DELIMITER $$
CREATE PROCEDURE orders_of_client(idClient_nr INT)
BEGIN
SELECT
        cont.name AS UserName,
   tab_ord.idOrder AS idOrder,
   tab_ord.status AS Status
   FROM
dorlux.Order AS tab_ord INNER JOIN Order_has_Item AS ord_has_it
                ON tab_ord.idOrder = ord_has_it.order_idOrder
        INNER JOIN Item AS it
                ON ord_has_it.Item_idItem = it.idItem
        INNER JOIN dorlux.Client AS Cl
                On tab_ord.Client_idClient = Cl.idClient
        INNER JOIN contact AS cont
                ON Cl.contact = cont.idContact
        WHERE Cl.idClient = idClient_nr;
END
$$
CALL orders_of_client(1)

# RM10 - Top X clients
DELIMITER $$
CREATE PROCEDURE top_X_clients(top_limit INT)
BEGIN
SELECT Cl.idClient AS "idClient", cont.name AS "Nome", it.priceSell*ord_has_it.amount AS
ValorGasto FROM
```

```
        dorlux.Order AS tab_ord INNER JOIN Order_has_Item AS ord_has_it
            ON tab_ord.idOrder = ord_has_it.order_idOrder
        INNER JOIN Item AS it
            ON ord_has_it.Item_idItem = it.idItem
        INNER JOIN dorlux.Client AS Cl
            On tab_ord.Client_idClient = Cl.idClient
        INNER JOIN contact AS cont
            ON Cl.contact = cont.idContact

  GROUP BY Order_idOrder
  ORDER BY ValorGasto DESC
      LIMIT top_limit;
END
$$
CALL top_X_clients(10);
CALL top_X_clients(20);
CALL top_X_clients(30);

# RM11 - How many items do we bought from a supplier
DELIMITER $$
CREATE FUNCTION  fuItemsCompradosForn(vatForn INT)
      RETURNS INT
      DETERMINISTIC
BEGIN
  DECLARE numeroitems INT;
  SELECT COUNT(I.idItem) INTO numeroitems FROM item AS I
      INNER JOIN suplier_provide_item AS SPI
        ON I.idItem = SPI.Item_idItem
      INNER JOIN suplier AS S
        ON SPI.Suplier_VAT = S.VAT
      WHERE S.VAT = vatForn;
  RETURN numeroitems;
END $$
SELECT fuItemsCompradosForn(795294490)

# RM11 - Which items we bought from a supplier
DROP PROCEDURE spQuaisItemsForn;

DELIMITER $$
CREATE PROCEDURE spQuaisItemsForn(vatForn INT)
```

```
BEGIN
   SELECT I.name, I.description FROM item AS I
      INNER JOIN suplier_provide_item AS SPI
         ON I.idItem = SPI.Item_idItem
      INNER JOIN suplier AS S
         ON SPI.Suplier_VAT = S.VAT
      WHERE S.VAT = vatForn;
END $$
CALL spQuaisItemsForn(795294490);

# RM12 - List the current stock
SELECT I.name, I.description, I.stockNr FROM Item AS I



# RM-13 - Orders items by stock number
SELECT * FROM Item
         ORDER BY stockNr DESC

# RM-14 - Order categories by name
SELECT * FROM category
         ORDER BY name ASC

# RM-15 - Order employees by salary
SELECT * FROM employee
         ORDER BY salary DESC

-- Creating Clients View
CREATE VIEW Clients
AS
         SELECT idClient AS "Id", name AS "Name",  phone AS "PhoneNr", street AS "Street",
zipCode AS "ZipCode", city AS "City"
   FROM client AS cl INNER JOIN contact AS cont
                  ON cl.contact = cont.idContact
         INNER JOIN Client_has_Address AS Cl_has_addr
                  ON cl.idClient = Cl_has_addr.Client_idClient
         INNER JOIN Address AS Addr
                  ON Cl_has_addr.Address_idAdress = Addr.idAdress
         ORDER BY name ASC;

-- Select View Clients
```

```
SELECT * FROM Clients;

-- Drop View Clients
DROP VIEW Clients;
```