Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Loja BracaTECH

Filipa Correia Parente - A82145

José André Martins Pereira – A82880

Rafaela Maria Soares da Silva – A79034

Ricardo André Gomes Petronilho – A81744

Novembro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Loja BracaTECH

Filipa Correia Parente - A82145

José André Martins Pereira – A82880

Rafaela Maria Soares da Silva – A79034

Ricardo André Gomes Petronilho – A81744

1. Resumo

O Sistema de Base de Dados, cada vez mais, parece afirmar-se como uma alternativa fiável, mais informativa e simplificada, face aos registos em papel. Posto isto, no âmbito da Unidade Curricular de Base de Dados, do 3º ano do Mestrado Integrado em Engenharia Informática, decidimos desenvolver a criação de um SBD para uma empresa de serviços informáticos.

Neste relatório, será apresentada a definição do sistema, o levantamento e análise de requisitos, a modelação conceptual, a modelação lógica e a implementação física, pela ordem respetiva. Para cada uma das etapas, procedemos à sua validação.

Para além de elaborarmos a modulação conceptual através do *brModelo*, recorremos ao *MySQL*, para a implementação do modelo lógico, físico e desenvolvimento das *queries* que consideramos relevantes para a extração de informação que a nossa Base de Dados tem de ser capaz de responder.

Por fim, expomos a nossa análise crítica do desenvolvimento do projeto, onde apresentamos as nossas dificuldades e conclusões.

Área de Aplicação: Desenho e arquitectura de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados Relacionais, Modelação Conceptual, Modelação Lógica, mySQL, SELECT, UPDATE, INSERT, PROCEDURE, TRANSACTION, Álgebra Relacional.

2. Índice

1. Definição do Sistema

1.1. Contexto da aplicação do sistema

A empresa BracaTECH é a empresa de serviços informáticos mais antiga de Braga. Esta atua em diversos setores e marcas, desde 1980.

Devido ao avanço tecnológico, a firma dá primazia à adaptação das tecnologias que vão surgindo no mercado. Nesse intuito, disponibiliza produtos e serviços de última geração.

A sua ambição é destacar-se, não só pelo cariz inovador, mas também pela qualidade de serviço.

1.2. Fundamentação da aplicação da base de dados

A administração do departamento de comunicação estratégica e do departamento financeiro decidiram implementar um SBD para gerir informações relativas aos produtos, serviços, funcionários e clientes. A decisão surgiu pela impossibilidade de obter uma análise rápida e simplificada atrás de registos em papel.

A empresa investe quantias significativas em tecnologias de última geração, o que implica a necessidade de retorno breve. No entanto, tal não tem acontecido na medida desejada. Tanto o produto como o serviço têm um custo que tem de ser compensado pelo preço de venda. Para além disso, é fulcral conquistar novos clientes continuamente.

Assim, o objetivo do departamento financeiro é percecionar os produtos e serviços que não têm lucro significativo, de forma a maximizar a sua rentabilidade. Já a finalidade do departamento de comunicação estratégica é ter acesso facilitado às informações dos clientes, com o intento de padronizar as suas compras e de facilitar a análise swot.

1.3. Análise de viabilidade do processo

A implementação da Base de Dados apresentada à loja BracaTECH oferece viabilidade, visto que as vantagens que esta integra à loja, superam o orçamento necessário para a sua implementação. Os grandes benefícios da Base de Dados são a organização da informação dos clientes, que anteriormente era bastante difícil e impraticável, a possibilidade de gerir funcionários, produtos, serviços e vendas de forma rápida e robusta. A utilização da Base de Dados, tem a implicação do registo, de certas informações, que são obrigatórias para o bom funcionamento da mesma, tais como o nome do cliente, o seu NIF, entre outras informações.

Com a finalidade de manter a viabilidade da Base de Dados, sugere-se à loja BracaTECH uma verificação anual da mesma, pela mesma equipa que a desenvolveu, ou por outra, pois a mesma está apta para ser controlada por qualquer equipa, visto que contém todos os modelos necessários para sua perceção.

Deste modo, as grandes utilidades que a base de dados oferece à BracaTECH são a consulta instantânea de dados de clientes, funcionários, produtos, serviços e vendas, fazer análises estatísticas, filtrar clientes através dos seus campos de informação, com o objetivo de publicitar determinados produtos mais adequados, e provavelmente procurados pelos mesmos, como por exemplo: sexo, profissão, idade, estado civil, etc.., aumentando assim as vendas. De forma a conquistar a preferência do cliente, é possível ao mesmo, consultar o valor total poupado com descontos, entre outras funcionalidades.

Ao administrador da BracaTECH é possível a monitorização dos lucros, stocks, salários, valor total em descontos, produtos mais/menos vendidos, funcionários com mais/menos vendas/lucros, contribuindo para a gestão dos mesmos. É possível também ao administrador selecionar o número adequado de funcionários para um determinado dia da semana, face às estatísticas de vendas desse dia.

2. Levantamentos e analise de requisitos

2.1. Método de Levantamento e Análise de Requisitos adotado

Quando a equipa tomou conhecimento do trabalho proposto para a conceção de uma Base de Dados, imediatamente listamos possíveis métodos de levantamentos de requisitos.

O método mais intuitivo e eficiente é entrar em contacto direto com as necessidades da loja no próprio local de trabalho. Desta forma, a equipa disponibilizou três horas diárias para observar, analisar o fluxo de trabalho dos funcionários no chão da loja e detetar que dados realmente são importantes recolher.

Simultaneamente, achamos pertinente realizar questionários a cada funcionário, normais e administradores, de forma a obter indicadores estatísticos das exigências de cada um para o funcionamento ideal da empresa.

Assim, foi proposto o seguinte questionário aos funcionários normais:

1. Na existência de um perfil de cliente individualizado que dados acha relevante armazenar?
2. Na existência de um perfil de cliente individualizado que dados acha obrigatório armazenar?
3. Existe necessidade de armazenar da data de registo do perfil do cliente?
4. No caso de ser necessário armazenar o contacto do cliente que dados acha úteis associar (ex: tipo de contacto telefónico: móvel, fixo, pessoal ou trabalho) e indique a obrigatoriedade dos mesmos?
5. Existe necessidade obrigatória de armazenar o endereço físico completo do cliente?

De seguida, apresenta-se o questionário fornecido aos administradores da loja:

1. Na existência de um perfil de funcionário individualizado que dados acha relevante armazenar?
2. Na existência de um perfil de funcionário individualizado que dados acha obrigatório armazenar?
3. Existe necessidade de armazenar a data de registo do perfil do funcionário?

4. No caso de ser necessário armazenar o contacto do funcionário que dados acha úteis associar (ex: tipo de contacto telefónico: móvel, fixo, pessoal ou trabalho) e indique a obrigatoriedade dos mesmos?
5. Existe necessidade obrigatória de armazenar o endereço físico completo do funcionário?
6. Que tipo de estatísticas são importantes para a BracaTECH?
7. Quais as propostas para o aumento das vendas, isto é, o que deve integrar a base de dados, com o objetivo de as aumentar?

Existem, contudo, questões pertinentes a todos os funcionários e setores da empresa:

1. Quais são os ofícios distintos realizados e vendidos pela loja (exemplos: produtos, serviços ou aluguer)?
2. Quais os dados que acha relevante armazenar sobre cada um desses ofícios separadamente?
3. No momento de venda dos diferentes ofícios que dados acha relevante associar ao registo da respetiva venda?
4. Existe interesse na expansão ou escalabilidade da informação armazenada em relação à loja?

Analisadas as respostas aos questionários e levantadas as nossas próprias exigências detetas no chão da loja conseguimos estruturar e elaborar uma lista de requisitos suficiente para responder aos vários departamentos da loja.

2.2. Requisitos levantados

Como foi referido na secção anterior, a nossa equipa utilizou diversas técnicas de levantamentos de requisitos que permitiram listar as exigências para melhorar a qualidade de serviço e eficácia financeira da empresa.
Nos próximos subcapítulos, abordamos com detalhe, por categorias, os requisitos levantados.

2.2.1 Requisitos de descrição

No início do processo de levantamento de requisitos, verificamos a necessidade de descrever os atores envolventes no sistema.

A primeira exigência que foi verificada é a criação de uma ficha individualizada para cada cliente com os dados pessoais do mesmo e outras informações resultantes da interação do cliente com a loja.

O cliente é caracterizado pelas suas informações pessoais como: nome completo, sexo, morada, email, estado profissional, e data de nascimento; mas também por dados relativos à relação do mesmo com a loja, tais como: o valor monetário total que o cliente poupou em descontos na empresa, o valor monetário total que gastou na empresa quer em produtos ou serviços e a data de registo das suas informações no sistema.

Para além disso, é associado ao cliente a classificação (de 1 a 10) que este pode atribuir à loja (assumimos o valor 0 como ausência de classificação).

Para identificar cada cliente utiliza-se o número de identificação fiscal (NIF) do mesmo.

Da mesma forma, todos os funcionários necessitam de serem registados no sistema.

O funcionário é também caracterizado pelas suas informações pessoais, tais como: nome completo, sexo, morada, email, e data de nascimento; no entanto, é registado também o tipo de funcionário (normal ou administrador), data de registo das suas informações no sistema, valor monetário total das vendas por si responsáveis e o seu salário atual.

Todas as vendas da loja são registadas no sistema.

Quando é efetuada uma venda é registada a sua data e hora, o preço total dado pelo preço individual de cada produto vendido, o NIF do cliente que participou na venda, o número de identificação do funcionário que assistiu à mesma e o valor de descontos efetuados. A venda é identificada por um número de identificação interno da empresa.

As informações sobre as gamas de produtos vendidos na loja são armazenadas na base de dados.

A gama de produtos por motivos de simplificação nominal denomina-se por produto no nosso modelo conceptual. O produto é caracterizado por designação, descrição, categoria, stock, preço unitário e desconto associado. O produto é identificado pelo número de identificação interno atribuído pela empresa.

Os serviços prestados pela loja são inventariados no sistema.

Quando é efetuado um serviço, são registados dados referentes à descrição do mesmo, data de início e de fim, estado do equipamento e o número de identificação do

funcionário e da venda. O serviço é identificado pelo número de identificação interno atribuído pela empresa.

As informações dos produtos vendidos em cada venda individualmente são distintas e registadas.

De forma a possibilitar as informações dos produtos vendidos em cada venda, todos os produtos são registados individualmente. São, assim, catalogados o identificador da venda, o identificador do produto, quantidade, preço unitário final, preço total, desconto unitário e desconto total.

Os dados acerca do contacto telefónico do cliente e do funcionário são memorizados no sistema.

É permitido que o cliente e o funcionário tenham mais de um contacto telefónico, de forma a possibilitar mais possibilidades de contacto. Assim, é registado o número de telefone, o respetivo tipo de número e o nif do cliente(no caso de ser cliente) ou identificador do funcionário(no caso de ser funcionário).

2.2.2 Requisitos de exploração

1. Conhecimento da área geográfica mais abrangida.
2. Conhecimento dos clientes que mais gastaram na loja.
3. Conhecimento dos produtos que necessitam de adição de stock.
4. Conhecimento dos produtos mais comprados.
5. Conhecimento do número de vendas num determinado ano.
6. Conhecimento do número de serviços feitos por cada funcionário num determinado ano.
7. Conhecimento do número de vendas feitas por cada funcionário num determinado ano.
8. Conhecimento dos clientes com as maiores quantias em descontos.
9. Calcular a classificação geral da loja.
10. Verificar as vendas de um determinado dia.
11. Calcular o valor total gasto em salários.
12. Procurar a ficha do cliente pelo número de telefone.
13. Conhecimento das informações de todos produtos comprados pelo cliente.
14. Detalhar a informação dos produtos de uma determinada venda.
15. Indica valor total de descontos do cliente.
16. Ter o conhecimento do valor em descontos adquiridos pelos clientes.

2.2.3 Requisitos de controlo

1. O administrador do SBD tem todas as permissões.
2. O funcionário não tem acesso à ficha de outros funcionários nem pode manipular informação.

2.3. Análise geral de requisitos

Após levantamento dos requisitos com a administração da empresa e posterior análise dos mesmos, apresentamos à mesma a nossa proposta para cada tipo de requisito (Descrição Controlo, Exploração). Após um pequeno debate chegou-se a um consenso entre ambas as partes e a proposta de requisitos apresentada foi aprovada. Prosseguiu-se assim para a fase da Modelação Conceptual.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Na modelação conceptual do sistema de gestão do BracaTECH, a equipa responsável teve de considerar determinados fatores relacionados com o funcionamento da empresa e os requisitos exigidos pela mesma.

Primeiramente, começamos por identificar as possíveis entidades principais e mais óbvias no sistema como o Cliente e o Funcionário. Tanto nos requisitos de descrição como de exploração estas duas entidades são claramente identificadas.

Para cada Cliente e Funcionário, é necessário armazenar o seu contacto, os vários números de telefone, caso tenham. Desta forma, criamos um atributo multi-valorado denominado telefones para ambas as entidades. Este atributo possibilita a resposta ao requisito procurar o perfil do cliente ou funcionário, através do seu contacto sendo bastante útil.

De seguida, procedeu-se ao reconhecimento das entidades que caracterizam os produtos vendidos e serviços prestados pela loja, onde obtemos as entidades Produto e o Serviço.

A entidade Produto, inicialmente, foi mal concebida. A equipa identificou a entidade como representação individual de todos os produtos, mesmo dos produtos da mesma gama. Ainda antes da fase de validação do modelo conceptual, reparamos que dois produtos da mesma gama (ex: portátil Asus Zenbook ux430) têm informação em comum como: a sua designação, descrição, categoria, preço unitário, desconto aplicado, entre outros; informação que caso seja armazenada individualmente para cada produto é repetitiva.

Identificado este erro de conceção, imediatamente desenvolvemos um novo conceito para corrigirmos esta redundância de informação. O Produto passou a representar o conceito de gama de produtos invés de um produto individual, desta forma foi adicionado o atributo stock que indica a quantidade de produtos existentes da própria gama.

Tendo estas quatro entidades que caracterizam todos os requisitos enunciados, necessitamos de uma nova entidade que interligue as mesmas de forma a haver um registo da venda indicando quem participou na mesma (tanto o cliente como funcionário) e em que data foi efetuada, entre outros mais à frente especificados. Desta forma, originou-se a entidade Venda.

Note-se que esta entidade é completamente genérica, ou seja, é possível identificar uma nova entidade no futuro que esteja relacionada com a Venda (qualquer entidade que possa ser "vendida") e à mesma é relacionada uma Venda sem necessidade de quaisquer modificações no modelo conceptual, mais concretamente na entidade Venda. Esta característica vantajosa é possível observar posteriormente na fase de modelação lógica, uma vez que na entidade Venda não existe nenhuma chave estrangeira associada ao Produto ou Serviço vendido.

3.2. Identificação e caracterização das entidades

Tendo em conta a abordagem de modelação efetuada, chegou-se à conclusão da existência de cinco entidades preponderantes no funcionamento do Sistema de Bases de Dados da empresa:

- A Venda, que pode ser de um serviço, ou a venda de um produto (geral à entidade Produto e Serviço);
- O Produto, correspondente, neste caso, a uma peça de hardware ou a um computador;
- O Serviço, que diz respeito a um atendimento ao cliente prestado no caso de avaria ou por vontade do cliente, caso este pretenda;
- O Cliente, entidade que solicita um serviço e/ou compra um ou vários produtos;
- O Funcionário, entidade que presta o atendimento ao cliente tanto no caso de venda de produto(s) como na venda de um serviço.

3.3. Identificação e caraterização dos relacionamentos

Na tabela seguinte, encontram-se os relacionamentos a que se chegaram depois de uma análise detalhada dos requisitos:

Entidade origem	Relacionamento	Relacionamento	Entidade Destino
Cliente	Participa	1..1	Venda
Venda	Associada	1..1	Servico
Venda	Contem	N..N	Produto
Funcionario	Realiza	1..N	Servico
Funcionario	Efetua	1..N	Venda

Tabela 1 – Relacionamentos entre entidades.

Dos relacionamentos existentes no modelo, talvez o que pode suscitar dúvidas é o relacionamento entre o Serviço e a Venda, uma vez que à partida deveria ser idêntico ao relacionamento entre o Produto e a Venda.
Entre o Produto e a Venda, estabelece-se uma relação de N para N, uma vez que uma Venda contém vários produtos e várias vendas contêm o mesmo produto.
No entanto, entre o Serviço e a Venda não existe a mesma cardinalidade na relação. Mesmo que num único serviço sejam realizadas diversas tarefas independentes na ótica da loja, contam apenas como uma tarefa ou serviço. Desta forma, tem todo o sentido que a relação seja de 1 Serviço para 1 Venda.

3.4. Identificação e caracterização das Associações dos Atributos com as entidades e relacionamentos

Após a identificação de cada entidade procedemos à identificação e caracterização dos atributos de cada entidade.
Desta forma começamos por descrever a entidade Cliente.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
Cliente	nif	Número de Identificação Fiscal	x		x	INT
	nome_completo	Nome completo			x	VARCHAR
	sexo	sexo				VARCHAR
	valor_total_descontos	Somatório do valor de todos os descontos aplicados a produtos				DOUBLE
	valor_total_gasto	Somatório do valor de todas as compras				DOUBLE
	data_nascimento	Data de nascimento			x	DATE
	data_registo_perfil	Data de registo dos dados do cliente no sistema			x	DATETIME
	email	Email			x	VARCHAR
	profissao	Profissão				VARCHAR
	codigo_postal	Código postal do endereço			x	VARCHAR
	cidade	Cidade do endereço			x	VARCHAR
	freguesia	Freguesia do endereço			x	VARCHAR
	rua	Rua do endereço			x	VARCHAR
	classificacao	Classificação geral que o cliente atribui ao serviço prestado pela loja				INT

Tabela 2 – Atributos da entidade Cliente.

O **número de identificação fiscal** do cliente é utilizado como chave primária da entidade uma vez que é uma variável única e permite ao cliente mais facilidade em memorizar o seu identificador na loja uma vez que, geralmente está familiarizado com a respetiva chave. Utiliza-se o tipo e dados INT e não VARCHAR() uma vez que este identificador tem 9 dígitos, desta forma utilizando VARACHAR() gastava-se 9 bytes enquanto que utilizando INT gasta-se 4 bytes.

O atributo sexo é representado apenas por um caracter variável, sendo possível existir três tipos de sexo: 'f' – feminino, 'm' – masculino ou 'o' – outro.

Note-se que o **valor total de descontos** e o **valor total gasto** são atributos **derivados** uma vez que dependem de outros atributos como o preço do produto e desconto percentual aplicado ao produto, da quantidade de produtos comprados no momento da venda, entre outros mais á frente detalhados. Desta forma estes atributos são calculados e validados posteriormente ao processo de venda.

A data de registo do perfil é obrigatória uma vez que é usada por questões informativas como saber quais os cliente mais antigos ou análise estatística sendo possível produzir indicadores como por exemplo a evolução do número de clientes num determinado período de tempo, entre outras vantagens não menos influentes.

O **email** é obrigatório ser informado uma vez que é necessário para o departamento de marketing sendo usado para fins publicitários. Pode também ser utilizado para enviar questionários ao cliente sobre a qualidade de serviço da loja.

As informações relativas ao **endereço** do cliente são obrigatórias para fins de faturação, identificação e análise estatística.

De seguida apresentam-se os atributos do **Funcionário**.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
Funcionario	id	Número de identificação	x		x	INT
	nome_completo	Nome completo			x	VARCHAR
	tipo	Tipo de Funcionário			x	VARCHAR
	sexo	Sexo				VARCHAR
	data_nascimento	Data de nascimento			x	DATE
	data_registo_perfil	Data de registo dos dados do cliente no sistema			x	DATETIME
	valor_total_vendas	Somatório do valor de todos as vendas que o funcionário efetuou				DOUBLE
	salario	Salário atual do funcionário			x	DOUBLE
	email	Email			x	VARCHAR
	codigo_postal	Código postal do endereço			x	VARCHAR
	cidade	Cidade do endereço			x	VARCHAR
	freguesia	Freguesia do endereço			x	VARCHAR

	rua	Rua do endereço			x	VARCHAR

Tabela 3 – Atributos da entidade Funcionario.

O **tipo de funcionário** representa a posição profissional do funcionário na loja sendo possível existir dois tipos de funcionários: 'n' – normal ou 'a' – administrador.
O **valor total das vendas** são atributos **derivados** uma vez que dependem de outros atributos relacionados com o momento de venda.

Na tabela abaixo descreve-se a entidade **Venda**.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
Venda	id	Número de identificação	x		x	INT
	data_venda	Data da venda			x	DATETIME
	preco_total	Somatório do preço de todos os produtos envolvidos na venda			x	DOUBLE
	nif_cliente	Número de identificação fiscal do cliente que participou na venda		x		INT
	Id_funcionario	Número de identificação do funcionário que efetuou a venda		x	x	INT
	valor_desconto	Somatório de descontos aplicados a todos os produtos			x	DOUBLE

Tabela 4 – Atributos da entidade Venda.

Tanto o **preço total da venda** como **valor de desconto total** são atributos derivados uma vez que são calculados através de outros atributos relacionados com a entidade Produto
De forma a identificar e obter informação sobre os participantes no momento de venda, os identificadores do funcionário e do cliente são ambos **chaves estrangeiras**. Note-se que o identificador do cliente não é obrigatório uma vez que o cliente pode participar numa venda e não ser registado no sistema.

De seguida evidencia-se a entidade **Produto**.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
Produto	id	Número de identificação	x		x	INT
	designacao	Designação/ nome			x	VARCHAR
	descricao	Descrição			x	VARCHAR
	stock	Número de produtos em stock			x	INT
	preco_unitario	Preço unitário			x	DOUBLE
	desconto	Desconto aplicado ao preço			x	DOUBLE

Tabela 5 – Atributos da entidade Produto.

Como se pode observar todos os atributos são obrigatórios uma vez que são absolutamente necessários ao funcionamento do sistema. Todos os atributos derivados referidos em cima são calculados a partir dos atributos da entidade Produto.

Em baixo é apresentada a entidade **Serviço**.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
Servico	id	Número de identificação	x		x	INT
	descricao	Descrição			x	VARCHAR
	data_inicio	Data do inico			x	DATETIME
	data_fim	Data do fim				DATETIME
	estado_equipamento	Descrição do estado do equipamento no momento de entrega á loja			x	VARCHAR
	id_funcionario	Número de identificação do funcionário que realizou o serviço		x	x	INT
	id_venda	Número de identificação da venda associada ao serviço		x	x	INT

Tabela 6 – Atributos da entidade Servico.

Note-se que a **data do fim** não é obrigatório no momento de criação de um Servico uma vez que é, inicialmente, indeterminada.

A **descrição do estado do equipamento no momento de entrega á loja** é extremamente útil uma vez que o cliente pode ter a necessidade de reclamar um possível risco no ecrã do seu equipamento entre outros fatores que apenas com esta descrição é possível provar que a causa do incidente foi por parte da loja.

Note-se que apesar do identificador do funcionário ser chave estrangeira da entidade Servico e Venda simultaneamente, **este pode não ser o mesmo funcionário**, isto é, o funcionário que realizou o serviço pode não ser o mesmo que regista a venda do serviço.

Finalmente é detalhado o relacionamento entre a Venda e o Produto denominada por VendaProduto.

Entidade	Atributos	Descrição	CP	CE	NN	Domínio
VendaProduto	id_venda	Número de identificação da venda	x	x	x	INT
	id_produto	Número de identificação do produto	x	x	x	INT
	quantidade	Quantidade			x	INT
	preco_unitario_final	Preço individual final após desconto aplicado			x	DOUBLE

	preco_total	Quantidade * Preço individual			x	DOUBLE
	desconto_unitario	Desconto individual			x	DOUBLE
	desconto_total	Quantidade * Desconto individual			x	DOUBLE

Tabela 7 – Atributos do relacionamento entre a Venda e o Produto.

O **preco_unitario_final** é um atributo derivado e é o preço individual de cada produto no momento de venda já com o desconto aplicado ao mesmo.
Tanto o **preco_total** como o **desconto_total** são também atributos derivados.

3.5. Detalhe ou generalização das entidades

Na realização da modelação conceptual, não foi aplicada o detalhe ou generalização das entidades.

3.6. Apresentação e explicação do diagrama ER

Na figura seguinte, apresenta-se o diagrama ER tendo em conta a abordagem de modelação realizada:

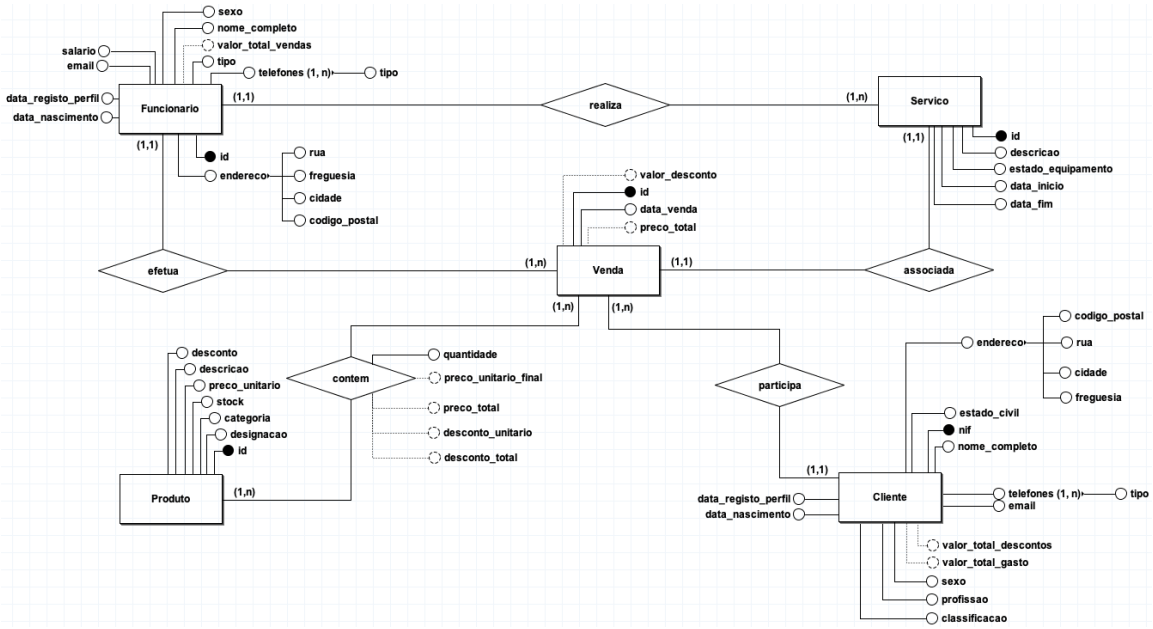

Figura 1 – Modelo Conceptual

Pelo que se pode observar no diagrama apresentado, um cliente participa numa venda, efetuada por um funcionário, que pode estar associada a um serviço e/ou conter produtos. O serviço é realizado por um funcionário especializado para o efeito.

Na entidade Cliente o NIF apresenta-se como uma chave primária, uma vez que este é único para cada cliente. Como atributos derivados tem-se o valor_com_descontos e o valor_total_gasto, visto que estes vão sendo atualizados conforme uma venda é efetuada. A morada é um atributo composto pelos atributos codigo_postal, rua e freguesia, pois fazem parte constitutiva desta. O telefone é um atributo multi-valorado, dando ao cliente a possibilidade de este ter mais do que um contato telefónico. O resto das informações pessoais estão inseridas em atributos simples.

Por sua vez, na entidade funcionário, o id é uma chave primária. Note-se que, como referido nos pontos anteriores, este não corresponde ao NIF, para dar a possibilidade ao mesmo de ser cliente da empresa. A informação relativa aos valores totais das vendas efetuadas(valor_das_vendas) está inserida num atributo derivado, visto que este também vai sendo atualizada conforme é efetuada uma venda.

Tal como no cliente, no funcionário a morada também é um atributo composto pelas mesmas razões apresentadas.

O resto das informações pessoais, estão inseridas em atributos simples.

Tanto a entidade Servico, como a Venda e o Produto também apresentam um ID como chave primária para distinguir diferentes serviços, vendas e produtos, respetivamente. Enquanto as entidades Servico e Produto apresentam apenas atributos simples para armazenar as informações correspondentes, a Venda apresenta um atributo derivado, o preço total (preco_total) da venda, pois depende do preço do produto ou do serviço, da quantidade do produto e do desconto aplicado.

Finalmente, tanto a entidade Venda como o Produto, como a relação binária é de N para N, foi inserido o atributo simples quantidade e os atributos derivados preco_final, e preco_total, correspondentes ao preço final do produto unitário já com o desconto e ao preço total a ser pago, já multiplicado pela quantidade. Estes dois últimos atributos derivados foram adicionados com o objetivo de evitar o acesso repetitivo à base de dados.

3.7. Validação do modelo de dados com o utilizador

Após a finalização do modelo conceptual e posterior apresentação do diagrama E-R à administração da empresa, obteve-se um *feedback* positivo, uma vez que todos os requisitos exigidos foram devidamente satisfeitos. Tendo em conta a aprovação do

modelo conceptual, prosseguiu-se para a fase seguinte do projeto, correspondente à Modelação Lógica.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

A elaboração do modelo lógico consistiu inicialmente em identificar as tabelas necessárias para a elaboração do mesmo. Para isso, a equipa baseou-se nas entidades identificadas no modelo conceptual, nos relacionamentos destas, atributos multivalorados e as respetivas cardinalidades.

No modelo conceptual existem diferentes tipos de relacionamentos, com cardinalidades N para N, onde se criou uma tabela para relacionar as entidades intervenientes, e nos de tipo de 1 para N, onde apenas se ligou as tabelas. Para a criação das tabelas, foi necessário a escolha de uma chave primária, que seja única, para sua identificação e que os atributos da mesma tabela sejam unicamente dependentes dessa chave. Do mesmo modo, inclui-se chaves estrangeiras em tabelas onde exista ligação entre as mesmas, para permitir a busca de informação de tabelas, a partir de outras.

Deste modo, começou-se por criar as tabelas Cliente e Funcionario, entidades da base de dados, e tal como o nome sugere, guardam a informação referente a um cliente e funcionário respetivamente. A chave primária do Cliente é o nif, e do **Funcionario** um id atribuído, sendo estas chaves únicas, e os restantes atributos não chave destas tabelas são dependentes unicamente destas chaves. Ambas as tabelas não possuem chaves estrangeiras.

Como se pode observar no modelo conceptual o atributo telefones é multivalorado na entidade **Cliente** e **Funcionario**, portanto, para garantir essa cardinalidade, no modelo lógico criou-se a tabela **ClienteTelefone** e **FuncionarioTelefone**, que integram como chave primária o número de telefone.

Na tabela **ClienteTelefone** tem-se como chave estrangeira, o nif do Cliente e na **FuncionarioTelefone,** o id do **Funcionario**, de forma a ser possível consultar a tabela do **Cliente** e **Funcionario** através do seu número de telefone. As tabelas também são constituídas por um atributo que identifica o tipo de número, ou seja, fixo ou móvel, identificado como ‘F’ e ‘M’, respetivamente.

Também é importante referir que o atributo composto endereço, presente nas entidades **Cliente** e **Funcionario**, foi colocado nas tabelas Cliente e **Funcionario**, visto que o código postal não é único para cada rua, em determinadas zonas de Portugal, sendo que para garantir a normalização, decidiu-se não isolar esta informação numa única tabela e mantê-la nas respetivas tabelas das entidades.

A entidade Venda também originou uma tabela, tendo como chave primária um id único. A tabela integra duas chaves estrangeiras que identificam o Cliente e Funcionario que participam nesta Venda, de modo a conseguir identificá-los e consultar a sua informação.

As entidades Produto e Servico, geraram as suas próprias tabelas, as quais contém um id como chave primária. Estas tabelas possuem uma relação com a tabela Venda, sendo que a relação de Produto com Venda tem cardinalidade de N para N, sendo necessário a criação de uma tabela desta relação, enquanto que a tabela Servico apenas contém a ligação à Venda, pois tem cardinalidade de 1 para N.

A razão da criação da tabela de relação VendaProduto, foi com o objetivo de que o modelo lógico fique apenas com cardinalidades de 1 para N, sendo que com a adição desta relação, a carnalidade de Venda para VendaProduto e de Produto para VendaProduto é de 1 para N.

A tabela Servico, ao contrário da tabela Produto, contém uma chave estrangeira do id do funcionário que realiza o Servico, que pode ser diferente do funcionário que realiza a Venda, daí a importância e inclusão desta chave estrangeira. A chave estrangeira id_venda, presente na tabela Servico, pode ser null, visto que, quando ocorre um serviço ele não tem uma venda imediatamente associada, só após a conclusão do mesmo é que ocorre essa associação.

A tabela VendaProduto, como já foi referido acima, relaciona as tabelas/entidades Produto e Venda, visto que esta tem uma relação com cardinalidade de N para N, portanto foi necessário a criação da mesma.

A VendaProduto integra uma chave composta, com o id do Produto e da Venda, sendo que é impossível achar/identificar esta tabela sem os dois campos da chave. O id do Produto e da Venda também são considerados chaves estrangeiras, na medida em que se pode consultar a informação dos mesmos a partir do id respetivo, na tabela VendaProduto.

Assim fica-se com:

Cliente(nif, nome_completo, sexo ,valor_total_descontos ,valor_total_gasto ,data_nascimento ,data_registo_perfil ,email ,profissao ,codigo_postal ,cidade ,
freguesia , rua ,classificacao)

PK: nif

Funcionario (id ,nome_completo ,tipo ,sexo ,data_nascimento ,data_registo_perfil ,valor_total_vendas ,salario ,email ,codigo_postal ,cidade ,freguesia ,rua)

PK: id

Venda (id ,data_venda ,preco_total ,nif_cliente ,id_funcionario ,valor_desconto)

PK: id

FK: Cliente referência Cliente(nif), Funcionario referência Funcionario(id)

Produto (id , designacao ,descricao ,categoria ,stock ,preco_unitario ,desconto)

PK: id

Servico (id

,descricao ,data_inicio ,data_fim ,estado_equipamento ,id_funcionario ,id_venda)

PK: id

FK: Funcionario referência Funcionario(id), Venda referência Venda(id)

VendaProduto(id_venda ,id_produto ,quantidade , preco_unitario_final ,preco_total ,de sconto_unitario , desconto_total)

PK: id_venda, id_produto (Chave Composta)

FK: Venda referência Venda(id), Produto referência Produto(id)

FuncionarioTelefone(nr_telefone ,tipo , id_funcionario)

PK: nr_telefone

FK: Funcionario referência Funcionrio(id)

ClienteTelefone(nr_telefone ,tipo ,nif_cliente)

PK: nr_telefone

FK: Cliente referência Cliente(nif)

4.2. Desenho do modelo lógico

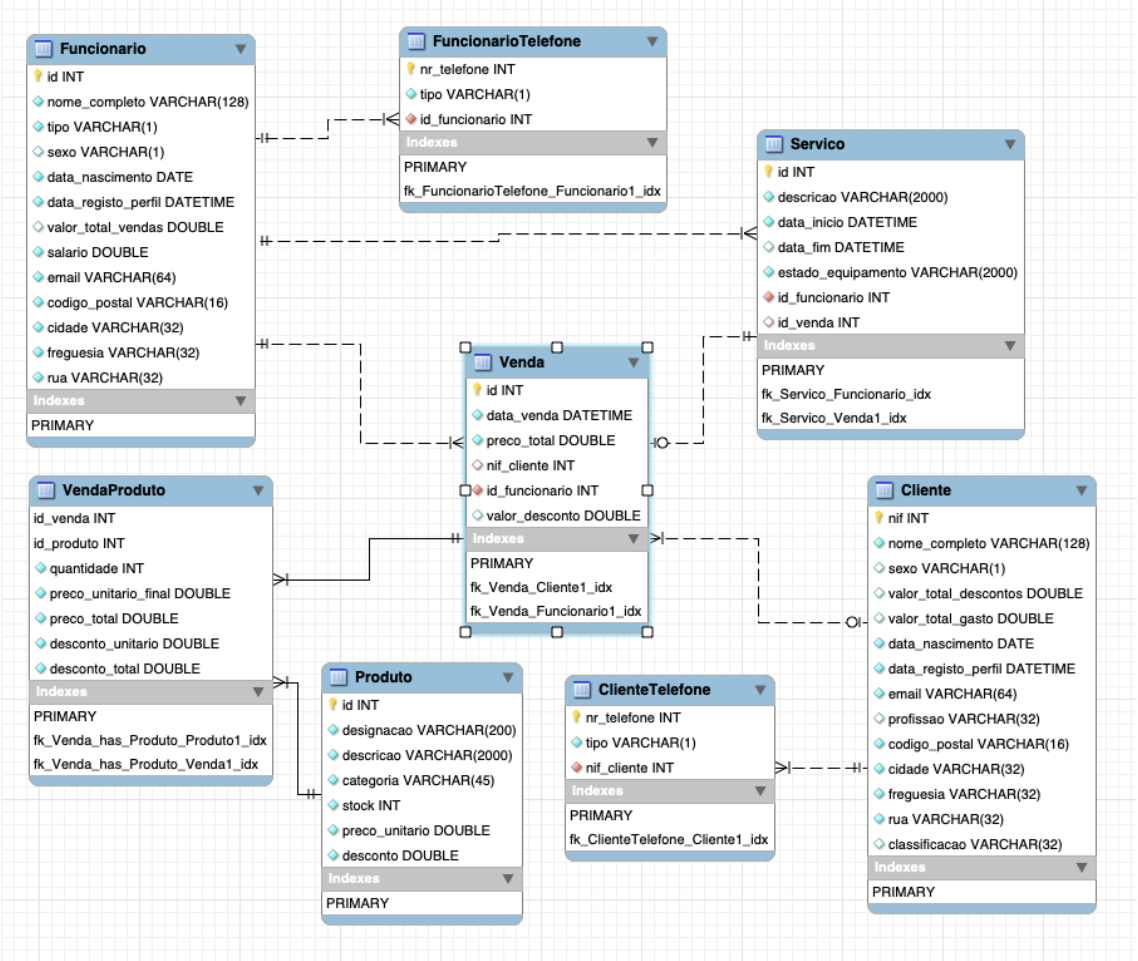

Figura 2 – Modelo lógico.

4.3. Validação do modelo através da normalização

A validação do modelo lógico começou pela listagem dos conjuntos/tabelas existentes e pela avaliação das dependências funcionais entre os atributos.

Assim temos que:

Cliente:

nif -> nome_completo,

sexo, valor_total_descontos, valor_total_gasto, data_nascimento, data_registo_perfil, email, profissao, codigo_postal, cidade, freguesia, rua, classificacao.

Funcionario:

id -> nome_completo, tipo,

sexo, data_nascimento, data_registo_perfil, valor_total_vendas, salario,

email, codigo_postal, cidade, freguesia, rua.

Produto:
id -> designacao, descricao, categoria, stock, preco_unitario, desconto.
Servico:
id -> descricao, data_inicio, data_fim, estado_equipamento, id_funcionario, id_venda.
Venda:
id -> data_venda, preco_total, nif_cliente, id_funcionario, valor_desconto.
VendaProduto:
id_venda, id_produto ->
quantidade, preco_unitario_final, preco_total, desconto_unitario, desconto_total.
FuncionarioTelefone:
nr_telefone -> tipo, id_funcionario.
ClienteTelefone:
nr_telefone -> tipo, nif_cliente.

As tabelas do modelo lógico respeitam a Primeira Forma Normal (1FN), pois todos os atributos das mesmas são atómicos, ou seja, não é possível dissociar qualquer um deles, e não existem atributos na mesma tabela, com informações repetidas ou idênticas.

A entidade Cliente e Funcionário, como já foi dito anteriormente, integra um atributo multivalorado, denominado por telefones. Deste modo, para garantir a Primeira Forma Normal(1FN), criou-se uma tabela ClienteTelefone e FuncionarioTelefone, para agregar todos os números de telefone destas entidades.

A Segunda Forma Normal (2FN), também é respeitada, visto que cumpre a Primeira Forma Normal (1FN), e todos os atributos não chave em todas as tabelas, dependem totalmente e unicamente da chave primaria da mesma.

Por fim, e não menos importante, a Terceira Forma Normal (3FN) é respeitada, visto que cumpre a Norma precedente e porque não existe nenhum atributo não chave que dependa de outro atributo não chave.

No caso da tabela Cliente e Funcionario, existem atributos relativos ao endereço, codigo_postal, freguesia, cidade e rua, os quais poderiam ser dependentes do codigo_postal, caso este fosse único por rua, mas a equipa verificou que existem locais em Portugal onde existe um único código postal, para diversas Ruas/Lugares, portanto, não foi necessário a separação destes atributos das respetivas tabelas, visto que, respeitam a Terceira Forma Normal (3FN).

4.4. Validação do modelo com interrogações do utilizado

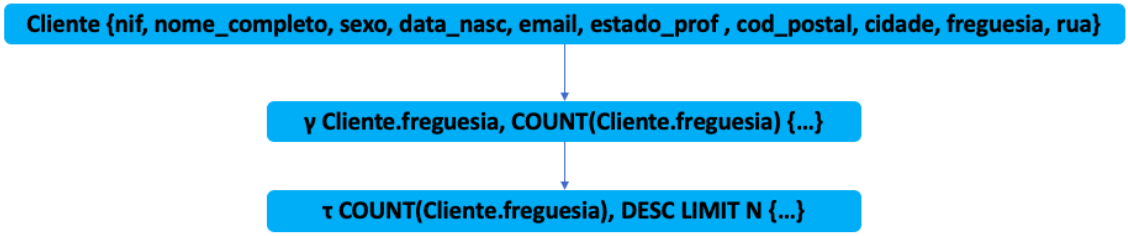

Figura 3 – Query 1.

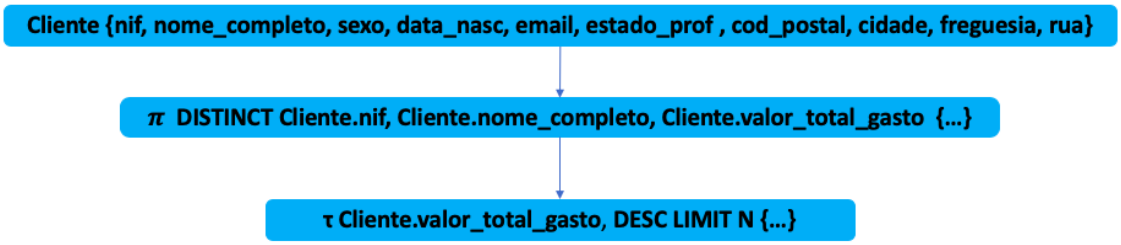

Figura 4 – Query 2.

Figura 5 – Query 3.

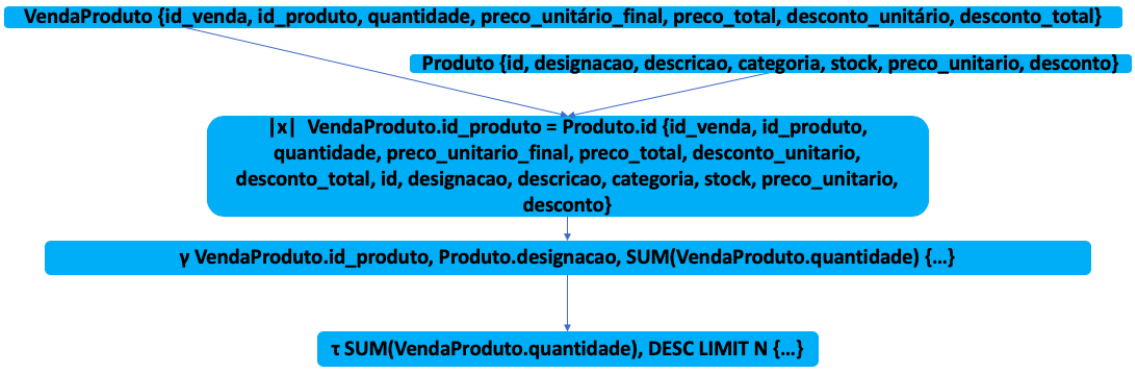

Figura 6 – Query 4.

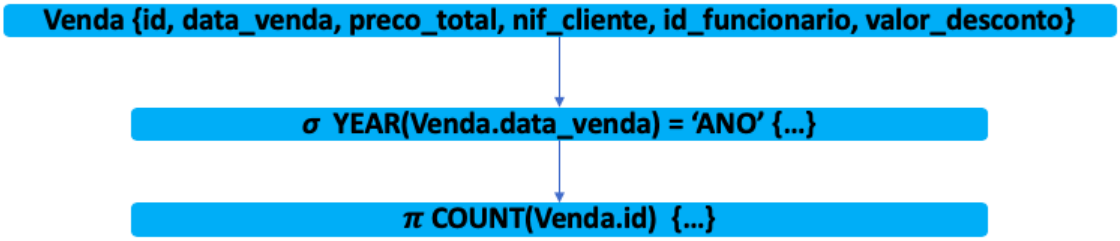

Figura 7 – Query 5.

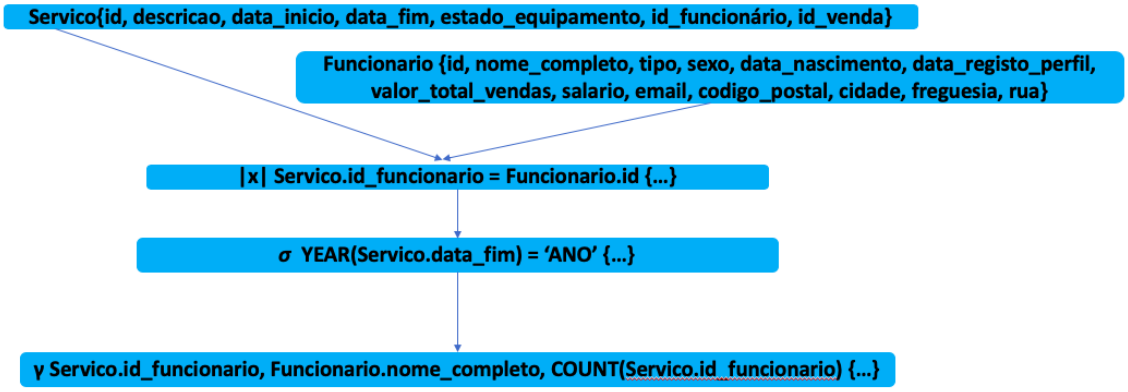

Figura 8 – Query 6.

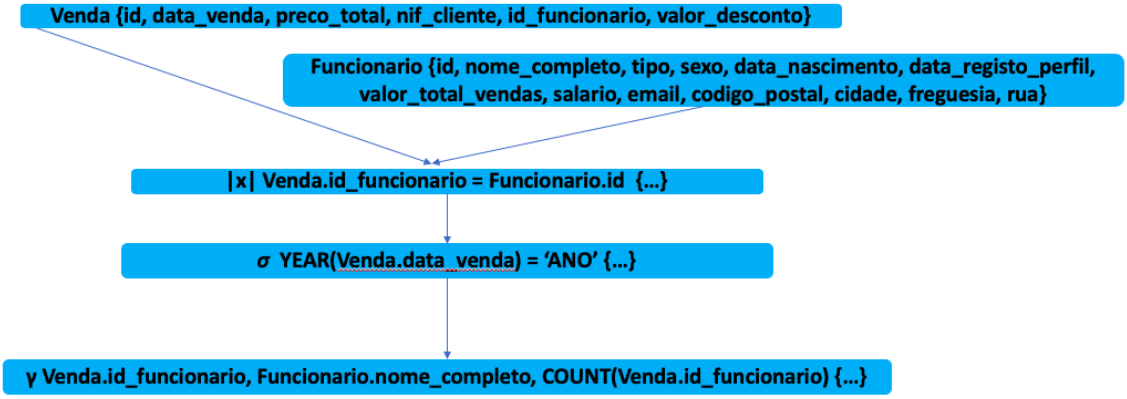

Figura 9 – Query 7.

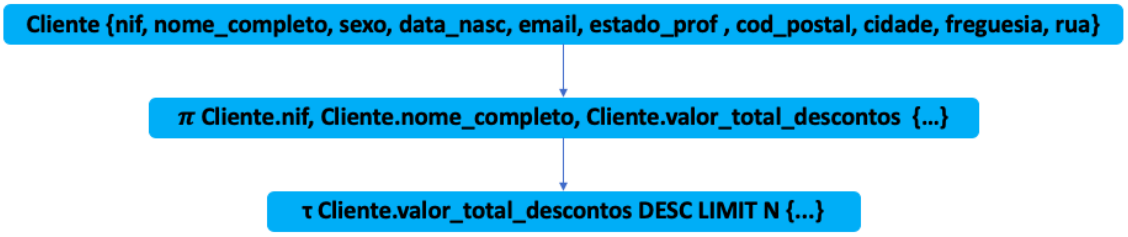

Figura 10 – Query 8.

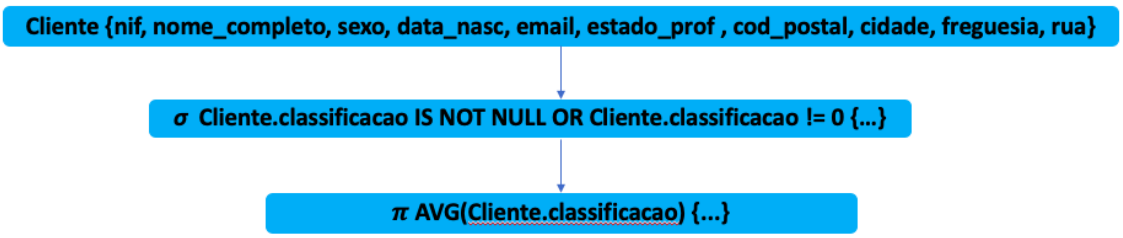

Figura 11 – Query 9.

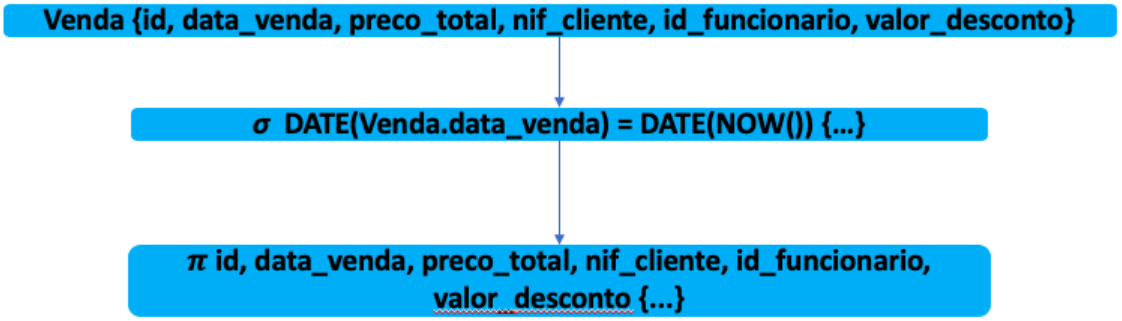

Figura 12 – Query 10.

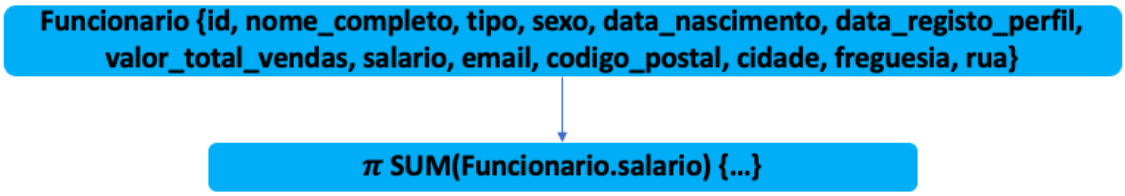

Figura 13 – Query 11.

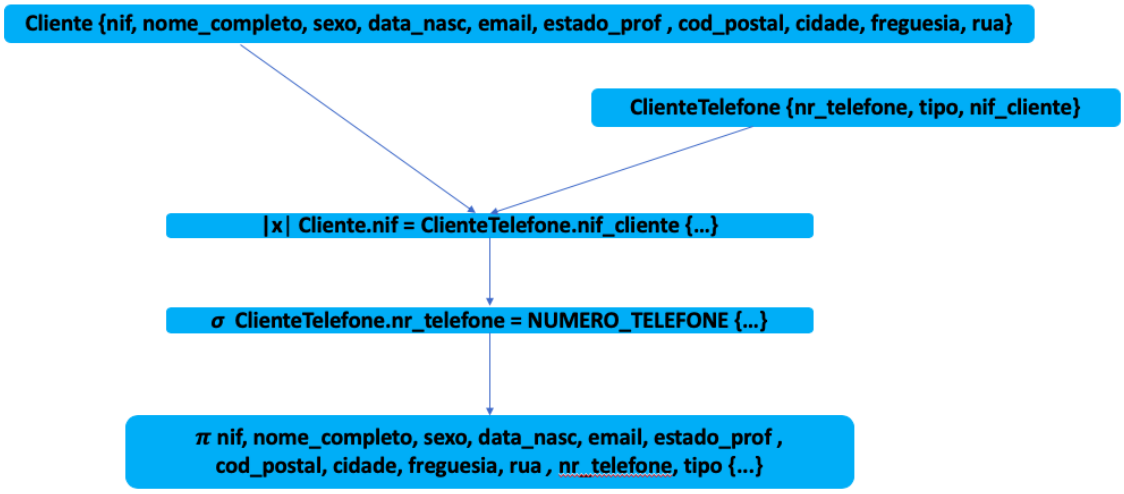

Figura 14 - Query 12.

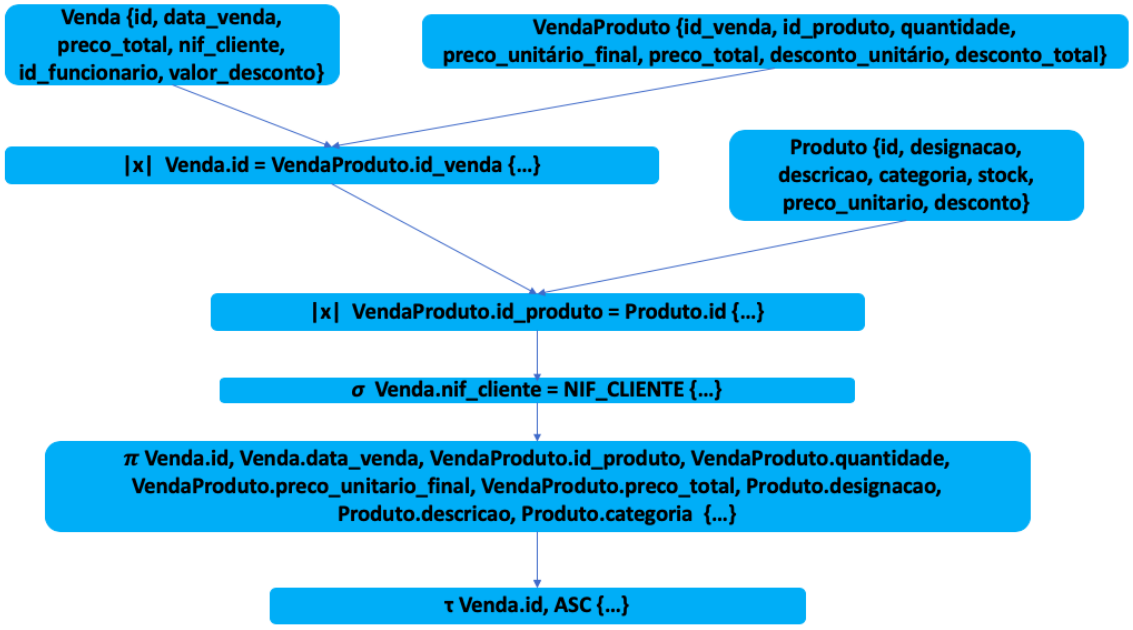

Figura 15 – Query 13.

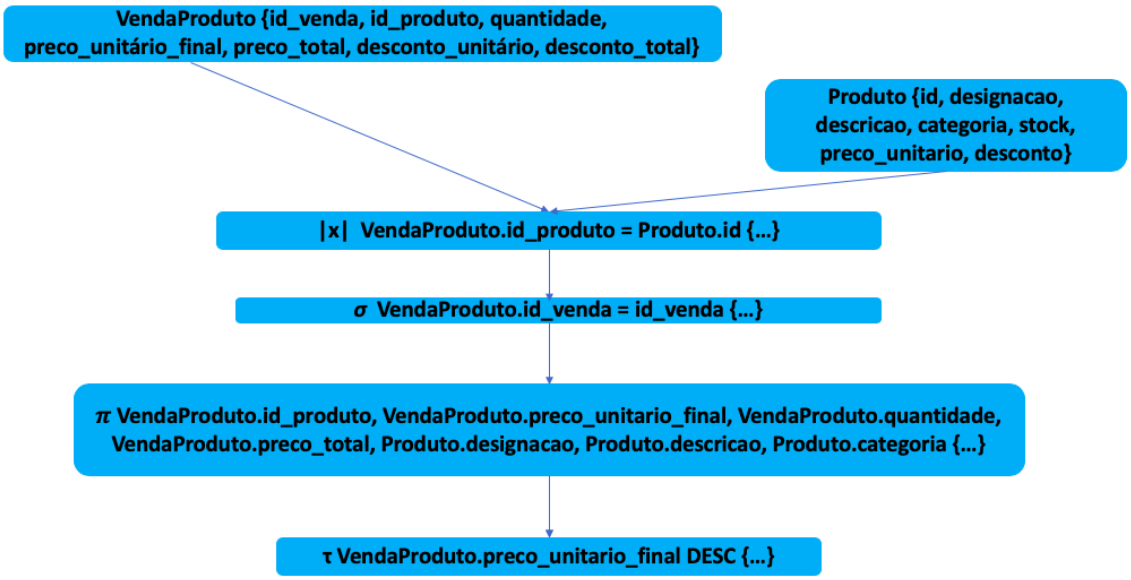

Figura 16 – Query 14.

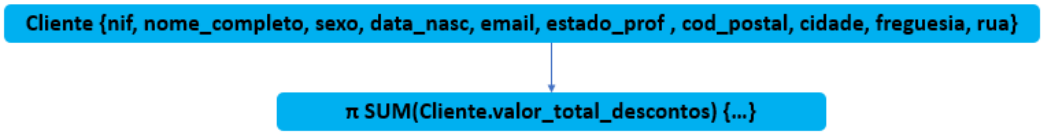

Figura 17 – Query 15.

4.5. Validação do modelo com as transações estabelecidas

De modo a validar uma venda, decidimos implementar duas transações capazes de assegurar a consistência de dados de todas as tabelas associadas. Esta consistência é garantida pela imposição de atualização de todas as tabelas relativas à transação. Visto que as transações em SQL permitem também garantir alterações atómicas, nada mais é alterado ao mesmo tempo.

Como uma venda pode ser realizada através de uma prestação de serviço ou de venda de produto(s), elaboramos uma transação para cada caso de atualização.

A ordem estabelecida garante que os dados são atualizados devidamente.

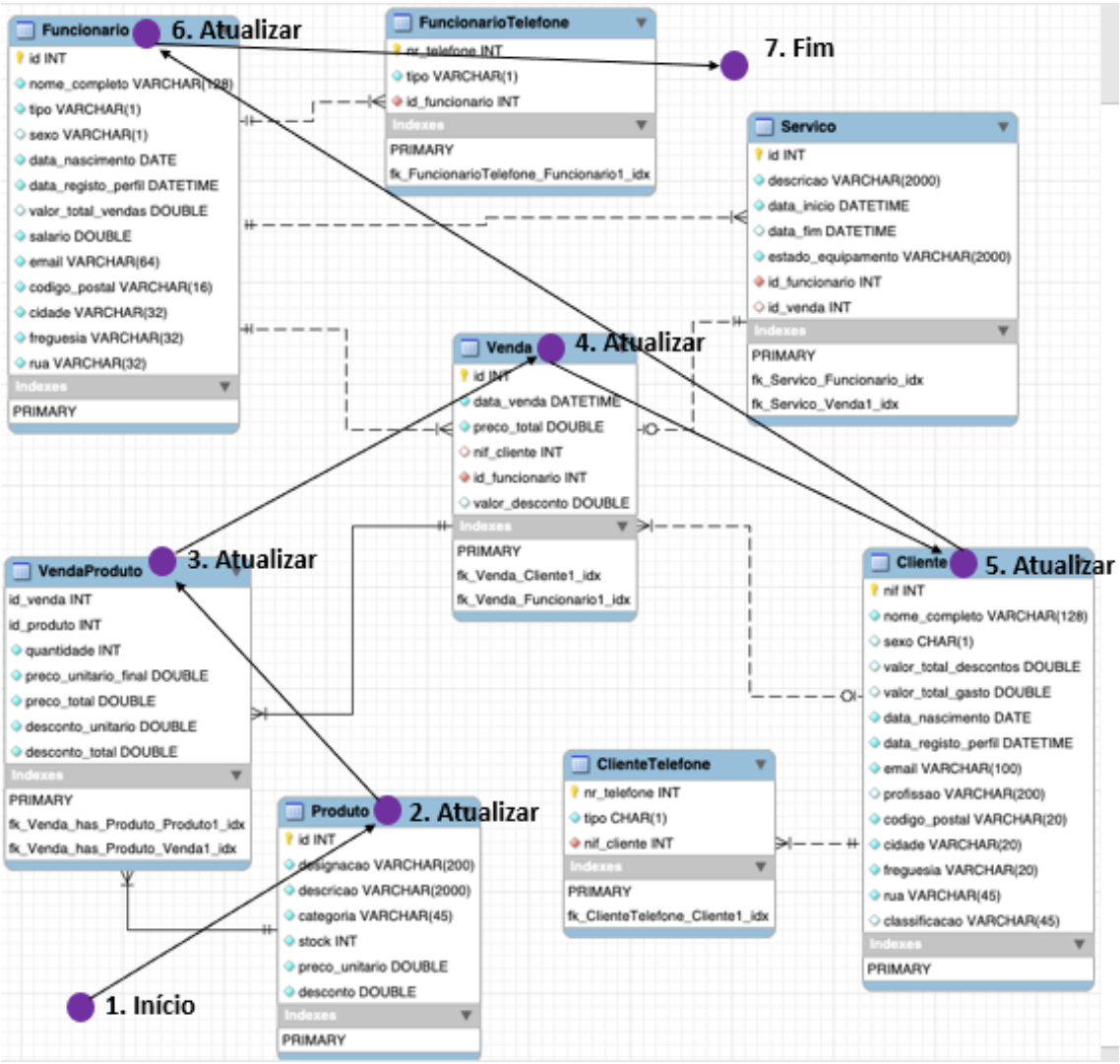

Figura 18 - Mapa de transação - Validar uma venda relativa a produto(s).

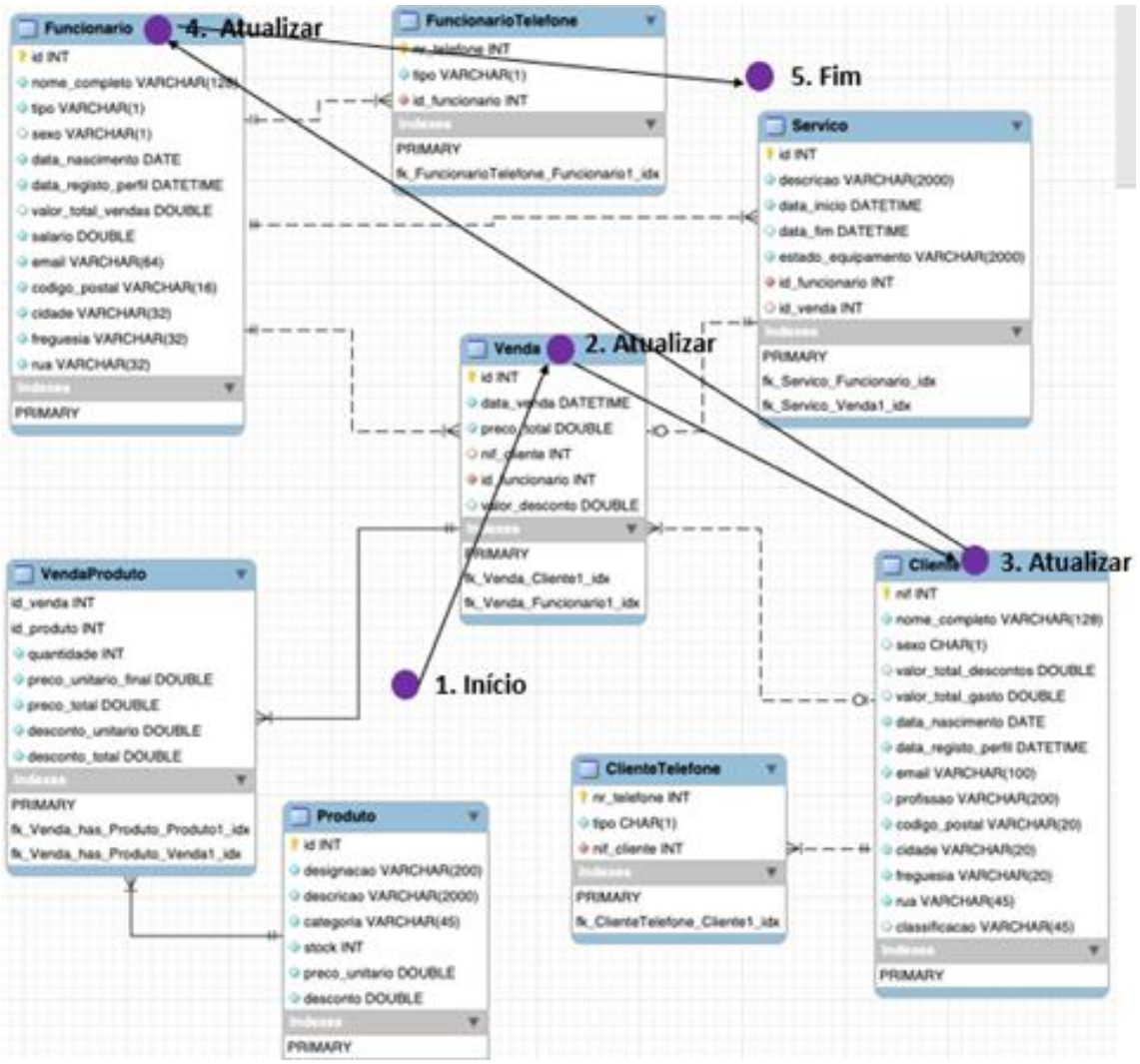

Figura 19 - Mapa de transação - Validar uma venda relativa a serviço

4.6. Reavaliação do modelo lógico (se necessário)

Visto que os principais objetivos do nosso SBD são criar ficha de clientes e permitir efetuar registos de vendas, consideramos que apresentamos, de forma eficiente, o que a empresa nos solicitou.

4.7. Revisão do modelo lógico com o utilizador

Após a validação do modelo lógico, de interrogações e transações, apresentamos a nossa proposta à empresa, com vista sua aprovação. Após uma breve exposição e debate relativamente à constituição do modelo lógico, chegou-se ao consenso entre o que apresentamos e o que a firma deseja. Tendo em conta a aprovação, partimos,

finalmente, para a última fase do projeto, isto é, à implementação física do sistema de Base de Dados.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

Para a realização da implementação física, escolhemos utilizar o sistema *mySQL,* visto que, para além de ter sido o único aprendido em contexto de sala de aula, apresenta inúmeras vantagens, nomeadamente o fato de ter um alto desempenho, de ser "*cross platform*", isto é, ser um sistema compatível com sistemas operativos distintos (Linux, Windows, OSX), bem como por ser um sistema seguro. É de referir também que o sistema em si apresenta todas as funcionalidades necessárias à sua implementação.

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

```sql
-- -----------------------------------------------------
-- Schema BracaTECH
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `BracaTECH` DEFAULT CHARACTER SET utf8 ;
USE `BracaTECH` ;

-- -----------------------------------------------------
-- Table `BracaTECH`.`Funcionario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`Funcionario` (
  `id` INT NOT NULL,
  `nome_completo` VARCHAR(128) NOT NULL,
  `tipo` VARCHAR(1) NOT NULL,
  `sexo` VARCHAR(1) NULL,
  `data_nascimento` DATE NOT NULL,
  `data_registo_perfil` DATETIME NOT NULL,
  `valor_total_vendas` DOUBLE NULL,
  `salario` DOUBLE NOT NULL,
  `email` VARCHAR(64) NOT NULL,
  `codigo_postal` VARCHAR(16) NOT NULL,
  `cidade` VARCHAR(32) NOT NULL,
  `freguesia` VARCHAR(32) NOT NULL,
  `rua` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BracaTECH`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`Cliente` (
  `nif` INT NOT NULL,
  `nome_completo` VARCHAR(128) NOT NULL,
  `sexo` CHAR(1) NULL,
  `valor_total_descontos` DOUBLE NULL,
  `valor_total_gasto` DOUBLE NULL,
  `data_nascimento` DATE NOT NULL,
  `data_registo_perfil` DATETIME NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `profissao` VARCHAR(200) NULL,
  `codigo_postal` VARCHAR(20) NOT NULL,
  `cidade` VARCHAR(20) NOT NULL,
  `freguesia` VARCHAR(20) NOT NULL,
  `rua` VARCHAR(45) NOT NULL,
  `classificacao` VARCHAR(45) NULL,
  PRIMARY KEY (`nif`))
ENGINE = InnoDB;
```

Figura 20 – Criação da tabela Funcionario e Cliente.

```sql
-- -----------------------------------------------------
-- Table `BracaTECH`.`Produto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`Produto` (
  `id` INT NOT NULL,
  `designacao` VARCHAR(200) NOT NULL,
  `descricao` VARCHAR(2000) NOT NULL,
  `categoria` VARCHAR(45) NOT NULL,
  `stock` INT NOT NULL,
  `preco_unitario` DOUBLE NOT NULL,
  `desconto` DOUBLE NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BracaTECH`.`VendaProduto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`VendaProduto` (
  `id_venda` INT NOT NULL,
  `id_produto` INT NOT NULL,
  `quantidade` INT NOT NULL,
  `preco_unitario_final` DOUBLE NOT NULL,
  `preco_total` DOUBLE NOT NULL,
  `desconto_unitario` DOUBLE NOT NULL,
  `desconto_total` DOUBLE NOT NULL,
  PRIMARY KEY (`id_venda`, `id_produto`),
  INDEX `fk_Venda_has_Produto_Produto1_idx` (`id_produto` ASC) VISIBLE,
  INDEX `fk_Venda_has_Produto_Venda1_idx` (`id_venda` ASC) VISIBLE,
  CONSTRAINT `fk_Venda_has_Produto_Venda1`
    FOREIGN KEY (`id_venda`)
    REFERENCES `BracaTECH`.`Venda` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Venda_has_Produto_Produto1`
    FOREIGN KEY (`id_produto`)
    REFERENCES `BracaTECH`.`Produto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BracaTECH`.`ClienteTelefone`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`ClienteTelefone` (
  `nr_telefone` INT NOT NULL,
  `tipo` CHAR(1) NOT NULL,
  `nif_cliente` INT NOT NULL,
  PRIMARY KEY (`nr_telefone`),
  INDEX `fk_ClienteTelefone_Cliente1_idx` (`nif_cliente` ASC) VISIBLE,
  CONSTRAINT `fk_ClienteTelefone_Cliente1`
    FOREIGN KEY (`nif_cliente`)
    REFERENCES `BracaTECH`.`Cliente` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 21 – Criação da tabela Produto, VendaProduto e ClienteTelefone.

```sql
-- -----------------------------------------------------
-- Table `BracaTECH`.`Venda`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`Venda` (
  `id` INT NOT NULL,
  `data_venda` DATETIME NOT NULL,
  `preco_total` DOUBLE NOT NULL,
  `nif_cliente` INT NULL,
  `id_funcionario` INT NOT NULL,
  `valor_desconto` DOUBLE NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Venda_Cliente1_idx` (`nif_cliente` ASC) VISIBLE,
  INDEX `fk_Venda_Funcionario1_idx` (`id_funcionario` ASC) VISIBLE,
  CONSTRAINT `fk_Venda_Cliente1`
    FOREIGN KEY (`nif_cliente`)
    REFERENCES `BracaTECH`.`Cliente` (`nif`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Venda_Funcionario1`
    FOREIGN KEY (`id_funcionario`)
    REFERENCES `BracaTECH`.`Funcionario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BracaTECH`.`Servico`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`Servico` (
  `id` INT NOT NULL,
  `descricao` VARCHAR(2000) NOT NULL,
  `data_inicio` DATETIME NOT NULL,
  `data_fim` DATETIME NULL,
  `estado_equipamento` VARCHAR(2000) NOT NULL,
  `id_funcionario` INT NOT NULL,
  `id_venda` INT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Servico_Funcionario_idx` (`id_funcionario` ASC) VISIBLE,
  INDEX `fk_Servico_Venda1_idx` (`id_venda` ASC) VISIBLE,
  CONSTRAINT `fk_Servico_Funcionario`
    FOREIGN KEY (`id_funcionario`)
    REFERENCES `BracaTECH`.`Funcionario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Servico_Venda1`
    FOREIGN KEY (`id_venda`)
    REFERENCES `BracaTECH`.`Venda` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 22 – Criação da tabela Venda e Servico.

```sql
-- -----------------------------------------------------
-- Table `BracaTECH`.`FuncionarioTelefone`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BracaTECH`.`FuncionarioTelefone` (
  `nr_telefone` INT NOT NULL,
  `tipo` VARCHAR(1) NOT NULL,
  `id_funcionario` INT NOT NULL,
  PRIMARY KEY (`nr_telefone`),
  INDEX `fk_FuncionarioTelefone_Funcionario1_idx` (`id_funcionario` ASC) VISIBLE,
  CONSTRAINT `fk_FuncionarioTelefone_Funcionario1`
    FOREIGN KEY (`id_funcionario`)
    REFERENCES `BracaTECH`.`Funcionario` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 23 – Criação da tabela FuncionarioTelefone.

5.3. Tradução das interrogações do utilizador para SQL

```sql
-- #6 QUANTOS SERVIÇOS FORAM FEITOS POR UM FUNCIONÁRIO EM DETERMINADO ANO
DELIMITER $$
CREATE PROCEDURE nr_servicos_feitos_por_cada_funcionario(IN ano VARCHAR(5))
BEGIN
    SELECT  Servico.id_funcionario, Funcionario.nome_completo as Nome, COUNT(Servico.id_funcionario) AS Servicos
        FROM Servico
            INNER JOIN Funcionario
                ON Servico.id_funcionario = Funcionario.id
                    WHERE YEAR(Servico.data_fim) = ano
                        GROUP BY Servico.id_funcionario;
END $$
DELIMITER ;
```

Figura 24 – Query 6 - nr_servicos_feitos_por_cada_funcionario

```sql
-- #9 CALCULA A CLASSIFICAÇÃO GERAL DOS CLIENTES
DELIMITER $$
CREATE PROCEDURE classificacao_geral ()
BEGIN
    SELECT AVG(Cliente.classificacao) AS classificacao_dos_clientes
        FROM Cliente
            WHERE Cliente.classificacao IS NOT NULL OR Cliente.classificacao != 0;
END $$
DELIMITER ;
```

Figura 25 – Query 9 - classicacao_geral

```sql
-- #13 - INFORMAÇÃO TODOS OS PRODUTOS COMPRADOS PELO CLIENTE
DELIMITER $$
CREATE PROCEDURE produtos_comprados_cliente(IN nif INT)
BEGIN
    SELECT Venda.id AS id_venda, Venda.data_venda, VendaProduto.id_produto,
           VendaProduto.quantidade,VendaProduto.preco_unitario_final,
           VendaProduto.preco_total, Produto.designacao, Produto.descricao,Produto.categoria
        FROM Venda
            INNER JOIN VendaProduto ON Venda.id = VendaProduto.id_venda
                INNER JOIN Produto ON VendaProduto.id_produto = Produto.id
                    WHERE Venda.nif_cliente = nif
                        ORDER BY Venda.id ASC;
END $$
DELIMITER ;
```

Figura 26 – Query 13 – produtos_comprados_cliente.

5.4. Tradução das transações estabelecidas para SQL

```sql
-- VALIDA A VENDA DE UM PRODUTO, DADO O ID DA VENDA.
DELIMITER $$
CREATE PROCEDURE valida_venda_produto (IN venda_id INT)
BEGIN

    DECLARE var_valor_total_da_venda DOUBLE;
    DECLARE var_valor_total_decontos DOUBLE;

    DECLARE exit handler for sqlexception
    BEGIN
        SELECT 'ERROR: todo o processo foi anulado! ';
        ROLLBACK;
    END;

    DECLARE exit handler for sqlwarning
    BEGIN
        SELECT 'WARNING: todo o processo foi anulado!';
        ROLLBACK;
    END;

    SET AUTOCOMMIT = 0;

    START TRANSACTION;

        IF (SELECT Venda.preco_total FROM Venda WHERE Venda.id = venda_id) = 0 THEN

            UPDATE Produto, VendaProduto
                SET Produto.stock = Produto.stock - VendaProduto.quantidade
            WHERE VendaProduto.id_venda = venda_id and VendaProduto.id_produto = Produto.id;

            UPDATE Produto, VendaProduto -- ATUALIZAMOS O VALOR PREÇO UNITÁRIO FINAL, NA RELAÇÃO VENDAPRODUTO
                SET VendaProduto.preco_unitario_final = Produto.preco_unitario - (Produto.preco_unitario * Produto.desconto)
            WHERE VendaProduto.id_venda = venda_id and VendaProduto.id_produto = Produto.id;

             UPDATE Produto, VendaProduto -- ATUALIZAMOS O VALOR DO DESCONTO UNITÁRIO, NA RELAÇÃO VENDAPRODUTO
                SET VendaProduto.desconto_unitario = Produto.preco_unitario - VendaProduto.preco_unitario_final
            WHERE VendaProduto.id_venda = venda_id and VendaProduto.id_produto = Produto.id;

            UPDATE VendaProduto -- ATUALIZAMOS O PREÇO TOTAL, NA RELAÇÃO VENDAPRODUTO
                SET VendaProduto.preco_total = VendaProduto.preco_unitario_final * VendaProduto.quantidade
            WHERE VendaProduto.id_venda = venda_id;
```

```
        UPDATE VendaProduto     -- ATUALIZAMOS O VALOR TOTAL DO DESCONTO, NA RELAÇÃO VENDAPRODUTO
            SET VendaProduto.desconto_total = VendaProduto.desconto_unitario * VendaProduto.quantidade
        WHERE VendaProduto.id_venda = venda_id;

        SELECT SUM(VendaProduto.preco_total) INTO var_valor_total_da_venda FROM VendaProduto WHERE VendaProduto.id_venda = venda_id;
        SELECT SUM(VendaProduto.desconto_total) INTO var_valor_total_decontos FROM VendaProduto WHERE VendaProduto.id_venda = venda_id;

        UPDATE Venda, Cliente, Funcionario  -- ATUALIZAÇÃO DOS ATRIBUTOS DERIVADOS DA VENDA, CLIENTE, FUNCIONÁRIO
            SET Venda.preco_total = var_valor_total_da_venda,
                Venda.valor_desconto = var_valor_total_decontos,
                Cliente.valor_total_gasto = Cliente.valor_total_gasto  + var_valor_total_da_venda,
                Cliente.valor_total_descontos = var_valor_total_decontos,
                Funcionario.valor_total_vendas = Funcionario.valor_total_vendas + var_valor_total_da_venda
        WHERE Venda.id = venda_id and Cliente.nif = Venda.nif_cliente and Funcionario.id = Venda.id_funcionario;

    END IF;

COMMIT;

END $$
DELIMITER ;
```

Figura 27 – Transação valida_venda_produto.

```
-- VALIDA A VENDA DE UM SERVIÇO DADO O ID DA VENDA E O PREÇO DO SERVIÇO.
DELIMITER $$
CREATE PROCEDURE valida_venda_servico(IN venda_id INT, IN preco_servico DOUBLE)
BEGIN

    DECLARE exit handler for sqlexception
    BEGIN
        SELECT 'ERROR: todo o processo foi anulado!';
        ROLLBACK;
    END;

    DECLARE exit handler for sqlwarning
    BEGIN
        SELECT 'WARNING: todo o processo foi anulado!';
        ROLLBACK;
    END;

    SET AUTOCOMMIT = 0;

    START TRANSACTION;

    -- ATUALIZA A DATA DE FIM DO SERVIÇO, CASO O PROGRAMADOR SE ESQUEÇA DE CHAMAR O PROCEDURE finaliza_servico.
    IF(SELECT Servico.data_fim FROM Servico WHERE Servico.id_venda = venda_id) IS NULL THEN
        UPDATE Servico, Venda
            SET Servico.data_fim = Venda.data_venda
                WHERE Servico.id_venda = venda_id and Venda.id = venda_id;
    END IF;

    -- ESTE IF GARANTE QUE NÃO SE VALIDA A MESMA VENDA MAIS DO QUE UMA VEZ.
    IF (SELECT Venda.preco_total FROM Venda WHERE Venda.id = venda_id) = 0 THEN
        UPDATE Venda
            SET Venda.preco_total = preco_servico
                WHERE Venda.id = venda_id;

        UPDATE Venda,Cliente
            SET Cliente.valor_total_gasto = Cliente.valor_total_gasto + Venda.preco_total
                WHERE Cliente.nif = Venda.nif_cliente and Venda.id = venda_id;

        UPDATE Venda,Funcionario
            SET Funcionario.valor_total_vendas = Funcionario.valor_total_vendas + Venda.preco_total
                WHERE Funcionario.id = Venda.id_funcionario and Venda.id = venda_id;
    END IF;
    COMMIT;
END $$
DELIMITER $$
```

Figura 28 – Transação valida_venda_servico

5.5. Escolha, definição e caracterização de índices em SQL

Para realização da implementação física não achamos relevante a utilização de índices neste projeto, uma vez que este em si é de cariz académico, pelo que não apresenta uma dimensão suficiente para a criação de índices extra.

5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Os tipos de dados usados na base de dados BracaTECH, são representados na seguinte tabela, bem como a respetiva ocupação em bytes.

Tipos de Dados	**Tamanho (bytes)**
INT	4
DOUBLE	8
CHAR	1
VARCHAR(N)	N+1
DATE	3
DATETIME	8

A estimativa do espaço utilizado por cada tabela em bytes, é apresentado na tabela seguinte. As aproximações abaixo, são o limite máximo de espaço ocupado por cada tabela, face aos seus atributos.

Deste modo, é importante realçar que, existem tabelas, como a Produto, Servico, que contém o atributo descricao com a possibilidade máxima de 2000 caracteres, mas o SQL apenas vai guardar os caracteres que forem ocupados, portanto a equipa decidiu que, é favorável permitir este limite de 2000 caracteres, visto que a descrição do produto é significativa para a sua perceção e fica ao critério do utilizador ocupar o espaço que achar necessário até ao limite máximo.

Tipos de Dados	Tamanho (bytes)
Cliente	409
Funcionario	345
ClienteTelefone	10
FuncionarioTelefone	10
Venda	36
Produto	2272
VendaProduto	44
Servico	4030

De seguida apresentamos alguns cenários da composição da base de dados, de forma a verificar o espaço ocupado por esta quantidade de dados.

Cenário 1:

20 Clientes (com um número de telefone), 2 Funcionarios(com um número de telefone), 1000 Vendas, 500 Produtos, 970 VendaProduto, 30 Servico.

Tipos de Dados	Tamanho (bytes)
Cliente	409 x 20 = 8180
Funcionario	345 x 2 = 690
ClienteTelefone	10 x 20 = 200
FuncionarioTelefone	10 x 2 = 20
Venda	36 x 1000 = 36000
Produto	2272 x 500 = 1,136,000
VendaProduto	44 x 1000 = 44,000
Servico	4030 x 30 = 120,900
Total	1,345,990 = 1,28364 MB

Cenário 2:

1000 Clientes(com um número de telefone), 20 Funcionarios(com um número de telefone), 10000 Vendas, 700 Produtos, 9900 VendasProduto, 100 Servico.

Tipos de Dados	**Tamanho (bytes)**
Cliente	409 x 1000 = 409,000
Funcionario	345 x 20 = 6,900
ClienteTelefone	10 x 1000 = 10,000
FuncionarioTelefone	10 x 20 = 200
Venda	36 x 10000 = 360,000
Produto	2272 x 700 = 1,590,400
VendaProduto	44 x 9900 = 435,600
Servico	4030 x 100 = 403,000
Total	3,217,100 = 3,06807 MB

Assim, podemos concluir que a nível de memória a base de dados, tem um crescimento maior, quando o número das tabelas Produto e Servico aumentam, pois são as que ocupam mais espaço de memória por tabela. No entanto, tal como já foi referido anteriormente, apesar do campo descricao, presente nas tabelas referidas ter um grande limite de caracteres, não significa que irá ser usado o limite em todas as instâncias, portanto não é preocupante, e do mesmo modo.

5.7. Definição e caracterização das vistas de utilização em SQL

```sql
-- Mostra a informação dos contactos do Cliente
CREATE VIEW Cliente_Contactos AS
    SELECT Cliente.nif, Cliente.nome_completo AS nome, email, ClienteTelefone.nr_telefone
        FROM Cliente
            INNER JOIN ClienteTelefone
                ON Cliente.nif = ClienteTelefone.nif_cliente;

-- Mostra a informação dos contactos do Funcionario.
CREATE VIEW Funcionario_Contactos AS
    SELECT Funcionario.id, Funcionario.nome_completo AS nome, email, FuncionarioTelefone.nr_telefone
        FROM Funcionario
            INNER JOIN FuncionarioTelefone
                ON Funcionario.id = FuncionarioTelefone.id_funcionario;
```

Figura 29 – As vistas Cliente_Contactos e Funcionario_Contactos

```sql
-- Informação visivel do funcionário sobre o cliente
-- ou outros departamentos
CREATE VIEW Funcionario_Cliente AS
    SELECT nif, nome_completo as nome, data_nascimento, email, profissao
        FROM Cliente;

-- Informação apenas da morada do cliente
CREATE VIEW Cliente_Morada AS
    SELECT nif, nome_completo AS nome, cidade, freguesia, rua, codigo_postal
        FROM Cliente;
```

Figura 30 – As vistas Funcionario_Cliente e Cliente_Morada

5.8. Definição e caracterização dos mecanismos de segurança em SQL

```sql
USE BracaTECH;

DROP USER IF EXISTS 'admin'@'localhost';
DROP USER IF EXISTS 'funcionario'@'localhost';

CREATE USER 'admin'@'localhost'
IDENTIFIED BY 'passwordAdmin';

CREATE USER 'funcionario'@'localhost'
IDENTIFIED BY 'passwordFuncionario'
WITH MAX_QUERIES_PER_HOUR 1000
MAX_UPDATES_PER_HOUR 500
PASSWORD HISTORY 20;

GRANT ALL PRIVILEGES ON * TO 'admin'@'localhost';

-- apenas pode atualizar os dados do Cliente
GRANT UPDATE ON Cliente TO 'funcionario'@'localhost';
GRANT SELECT ON * TO 'funcionario'@'localhost';

-- Mostrar os previlégios
SHOW GRANTS FOR  'admin'@'localhost';
SHOW GRANTS FOR 'funcionario'@'localhost';

SELECT mysql.user.USER FROM mysql.user;

FLUSH PRIVILEGES;
```

Figura 31 – Criação de utilizadores e privilégios dos mesmo.

5.9. Revisão do sistema implementado com o utilizador

Após posterior implementação do sistema no MySQL e apresentação à empresa, esta demonstrou agrado e satisfação com o SBD apresentado, considerando-o funcional.

6. Conclusões e Trabalho Futuro

Ao longo do desenvolvimento do nosso Sistema de Base de Dados, surgiram alguns entraves no desenvolvimento das modelações, uma vez que estas necessitaram de diversas revisões até à sua validação final.

Não registamos dificuldade no levantamento e na análise dos requisitos exigidos pela empresa. Contudo, no que concerne à conservação destes ao longo da realização do projeto, observamos uma certa dificuldade, principalmente durante a elaboração do modelo lógico que será abordado mais adiante.

Relativamente à elaboração da modelação conceptual, começamos por ponderar a existência de uma entidade Contacto. Após debates internos e recorrência externa ao docente, verificamos que a mesma não seria necessária, uma vez que poderíamos ter um atributo multi-valorado denominado telefones. Para além disso, como foi referido na apresentação da abordagem de modelação conceptual realizada, a entidade Produto foi inicialmente representada individualmente, o que implicou uma certa (re)conceptualização desta entidade, quando nos apercebemos de que dois produtos da mesma gama teriam informações em comum.

Reconhecemos algumas falhas na realização do diagrama E-R, como se pode reparar na falta do uso da generalização ou detalhe das entidades. Estas poderiam ser corrigidas, numa próxima fase. Por exemplo, no caso das entidades “Funcionario” e “Cliente”, estas possuem atributos repetidos que poderiam ser evitados criando uma entidade geral.

No que diz respeito à modelação lógica, inicialmente utilizamos o brModelo para gerar automaticamente o modelo lógico. No entanto, este programa não o apresentava, na nossa perspetiva, de forma correta. Mesmo utilizando o MySQL para a elaboração do modelo lógico, apesar de o considerarmos mais fidedigno, tivemos de elaborar alterações ao que foi apresentado pelo mesmo. Além disso, no desenvolvimento desta fase, apercebemo-nos de que o atributo preço no serviço não conferia consistência, visto que a venda em si é que tem um custo, o que implicou alteração no modelo conceptual e lógico.

Por outro lado, apesar dos desafios que tivemos de superar, há que destacar a facilidade e rapidez na realização de ambos os modelos, que não tiveram de ser reajustados totalmente. Há que sublinhar, no modelo lógico, o cumprimento de todas as fórmulas até à 3FN, sem necessidade de um reajustamento generalizado. Isto reflete o excelente levantamento e análise dos requisitos feito.

Quanto à implementação física, no desenvolvimento inicial das transações, surgiu inconsistência nas atualizações, uma vez que houve engano na ordem estabelecida das mesmas.

Em suma, após a abordagem dos vários pontos positivos e negativos, podemos concluir que os objetivos propostos no início deste projeto foram concluídos com sucesso.

7. Anexos

8. Anexo do subcapítulo 5.3

```
-- #1 - top N freguesias
DELIMITER $$
CREATE PROCEDURE top_freguesias(IN nr INT)
BEGIN
  SELECT Cliente.freguesia, COUNT(Cliente.freguesia) AS Numero
    FROM Cliente
        GROUP BY Cliente.freguesia
            ORDER BY Numero DESC
                LIMIT nr;
END $$
DELIMITER ;
```

Figura 32 Query 1 – top_freguesias

```
-- #2 TOP N CLIENTES QUE MAIS GASTARAM
DELIMITER $$
CREATE PROCEDURE top_clientes_que_mais_gastaram(IN nr INT)
BEGIN
   SELECT Cliente.nif, Cliente.nome_completo, Cliente.valor_total_gasto AS ValorGasto
       FROM Cliente
           ORDER BY ValorGasto DESC
               LIMIT nr;
END $$
DELIMITER ;
```

Figura 33 – Query 2 - top_clientes_que_mais_gostaram

```
-- #3 N PRODUTOS COM MENOS STOCK
DELIMITER $$
CREATE PROCEDURE n_produtos_menos_stock(IN nr INT)
BEGIN
   SELECT Produto.id, Produto.designacao, Produto.stock
       FROM Produto
           ORDER BY Produto.stock ASC
               LIMIT nr;
END $$
DELIMITER ;
```

Figura 34 – Query 3 - n_produtos_menos_stock

```
-- #4 N PRODUTOS MAIS COMPRADOS
DELIMITER $$
CREATE PROCEDURE n_produtos_mais_comprados(IN n INT)
BEGIN
    SELECT VendaProduto.id_produto, Produto.designacao, SUM(VendaProduto.quantidade) AS VEZES
        FROM VendaProduto
            INNER JOIN Produto
                ON VendaProduto.id_produto = Produto.id
                    GROUP BY VendaProduto.id_produto
                        ORDER BY VEZES DESC
                            LIMIT n;
END $$
DELIMITER ;
```

Figura 35 – Query 4 - n_produtos_mais_comprados

```
-- #5 QUANTAS VENDAS FORAM FEITAS NUM ANO
DELIMITER $$
CREATE PROCEDURE nr_vendas_feitas_em(IN ano VARCHAR(5))
BEGIN
    SELECT  COUNT(Venda.id) AS Vendas
        FROM Venda
            WHERE YEAR (Venda.data_venda) = ano;
END $$
DELIMITER ;
```

Figura 36 – Query 5 – nr_vendas_feitas_em

```
-- #7 QUANTAS VENDAS FORAM FEITAS POR CADA FUNCIONARIO NUM DETERMINADO ANO.
DELIMITER $$
CREATE PROCEDURE nr_vendas_feitas_por_cada_funcionario(IN ano VARCHAR(5))
BEGIN
    SELECT Venda.id_funcionario, Funcionario.nome_completo as Nome ,COUNT(Venda.id_funcionario) AS Vendas
        FROM Venda
            INNER JOIN Funcionario
                ON Venda.id_funcionario = Funcionario.id
                    WHERE YEAR(Venda.data_venda) = ano
                        GROUP BY Venda.id_funcionario;
END $$
DELIMITER ;
```

Figura 37 – Query 7 – nr_vendas_feitos_por_cada_funcionario

```
-- #8 MOSTRA OS N CLIENTES COM MAIS VALORES EM DESCONTOS
DELIMITER $$
CREATE PROCEDURE top_clientes_com_mais_descontos(IN limite INT)
BEGIN
    SELECT Cliente.nif, Cliente.nome_completo as Nome, Cliente.valor_total_descontos
        FROM Cliente
            ORDER BY Cliente.valor_total_descontos DESC
                LIMIT limite;
END $$
DELIMITER ;
```

Figura 39 – Query 8 – nr_servicos_feitos_por_cada_funcionario

```
-- #10 MOSTRA AS VENDAS DO DIA
DELIMITER $$
CREATE PROCEDURE vendas_do_dia()
BEGIN
        SELECT * FROM Venda
            WHERE DATE(Venda.data_venda) = DATE(NOW());
END $$
DELIMITER ;
```

Figura 40 – Query 10 – vendas_do_dia

```
-- #11 VALOR TOTAL EM SALARIOS NA LOJA BRACATECH
DELIMITER $$
CREATE PROCEDURE valor_em_salarios()
BEGIN
    SELECT SUM(Funcionario.salario) as Valor_total_em_salarios
        FROM Funcionario;
END $$
DELIMITER ;
```

Figura 41 – Query 11 – valor_em_salarios

```
-- #12 PROCURAR UM CLIENTE PELO SEU NÚMERO DE TELEFONE
DELIMITER $$
CREATE PROCEDURE procurar_por_telefone_cliente(IN nr INT)
BEGIN
    SELECT nif, nome_completo, sexo, data_nasc, email, estado_prof ,
        cod_postal, cidade, freguesia, rua , nr_telefone, tipo
        FROM Cliente
            INNER JOIN ClienteTelefone ON Cliente.nif = ClienteTelefone.nif_cliente
                WHERE ClienteTelefone.nr_telefone = nr;
END $$
DELIMITER ;
```

Figura 42 – Query 12 – procurar_por_telefone_cliente

```
-- #14 DETALHAR INFORMAÇÃO DOS PRODUTOS DA VENDA
DELIMITER $$
CREATE PROCEDURE mostra_info_produtos_venda(IN id_venda INT)
BEGIN
    SELECT VendaProduto.id_produto, VendaProduto.preco_unitario_final,
    VendaProduto.quantidade, VendaProduto.preco_total, Produto.designacao, Produto.descricao, Produto.categoria
        FROM VendaProduto
            INNER JOIN Produto
                ON VendaProduto.id_produto = Produto.id
                    WHERE VendaProduto.id_venda = id_venda
                        ORDER BY VendaProduto.preco_unitario_final DESC;
END $$
DELIMITER ;
```

Figura 43 – Query 14 – mostrar_info_produtos_venda

```
-- #15 QUAL O VALOR EM DESCONTOS ADQUIRIDO PELOS CLIENTES
SELECT SUM(Cliente.valor_total_descontos)
    FROM Cliente;
```

Figura 44 – Query 15 – valor_total_descontos_cliente

9. Referências

Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4ª Edição, 2004. ISBN-10: 0321210255. ISBN-13: 978-0321210258

10. Lista de Siglas e Acrónimos

SBD Sistema de Base de Dados