



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2016/2017

LEI EXPRESS

Reservas de viagens comboios nacionais e internacionais

Adriana Guedes, Diogo Soares, José Bastos, Ricardo Certo

Novembro, 2016

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Reservas de viagens comboios nacionais e internacionais

Adriana Guedes, Diogo Soares, José Bastos, Ricardo Certo

novembro, 2016

<</opcional Dedicatória>>

Resumo

O presente relatório foi efetuado no âmbito da Unidade Curricular de Base de Dados. O tema que será abordado é reservas de viagens em comboios nacionais e internacionais e, ao longo do relatório, será descrito o processo de elaboração da nossa base de dados, desde o modelo concetual até à implementação do modelo físico.

Neste relatório aplica-se a metodologia indicada na bibliografia, para a modelação da base de dados. Será realizada uma análise de requisitos onde identificamos as entidades que achamos mais essenciais para este trabalho.

Ao nível do modelo concetual serão descritas as entidades e os relacionamentos identificando os respetivos atributos. Além disso, é determinado o domínio de valores possíveis para os atributos e as chaves candidatas, identificando a chave primária e as alternativas se estas existirem.

A passagem do modelo concetual para modelo lógico aprofunda o estudo dos elementos descritos anteriormente, nomeadamente os relacionamentos e os atributos.

No modelo físico, apresenta-se a conversão dos dois modelos anteriores num esquema, com apresentação de código em SQL para a criação da base de dados

O principal objetivo da elaboração deste trabalho é realizar um sistema de base de dados, que seja capaz de gerir a informação acerca de um sistema de reserva de bilhetes numa rede ferroviária, fornecendo informações essenciais que constituem uma viagem.

Área de Aplicação: Análise de requisitos, desenho e arquitetura de Sistemas de Base de Dados

Palavras-Chave: Requisitos, Relacionamentos, SQL, MySQL Workbench, modelo físico, modelo lógico, esquema conceptual, companhia ferroviária, Base de dados , bilhete , sistema de reservas, cliente , viagens.

Índice

1

1. Introdução.....	9
1.1 Contextualização	9
1.2 Apresentação do Caso de Estudo	9
1.3 Motivação e Objectivos	10
1.4 Estrutura do Relatório	11

2. Requisitos.....	12
--------------------	----

3

3. Modulação Concetual.....	14
3.1 Verificação das chaves candidatas e chave primária	14
3.1.1 Bilhete.....	14
3.1.2 Comboio	14
3.1.3 Percurso	14
3.1.4 Cliente	15
3.1.5 Estação.....	15
3.2 Identificação dos tipos de entidade	15
3.3 Identificação dos tipos de relacionamento	16
3.4 Identificação dos atributos.....	18
3.4.1 Cliente	19
3.4.2 Bilhete.....	19
3.4.3 Comboio	20
3.4.4 Percurso	20
3.4.5 Estação.....	21
3.5 Domínio das entidades.....	21
3.5.1 Cliente	21
3.5.2 Bilhete.....	21
3.5.3 Comboio	22
3.5.4 Percurso	22
3.5.5 Estação.....	23
3.6 Modelo Concetual	23

4

4. Modulação lógica de dados.....	24
4.1 Passagem do modelo de dados concetual para um seu correspondente modelo de dados lógico	24

4.1.1 Entidades Fortes	24
4.1.2 Relacionamentos 1:N	24
4.1.3 Relacionamentos 1:1	24
4.1.4 Relacionamentos N:M	25
4.1.5 Relacionamentos Complexos.....	25
4.1.6 Atributos multivalor	25
4.2 Validação do modelo lógico através da normalização	25
4.2.1 1ª Forma Normal	26
4.2.2 2ª Forma Normal	28
4.2.3 3ª Forma Normal	29
4.3 Validação do modelo com as transações do utilizador – elaboração do correspondente mapa de transações.....	29
4.4 Elaboração e validação do esquema lógico da base de dados	30
4.5 Regras de integridade	30
4.6 Esquema Lógico.....	33
4.7 Definição do tamanho inicial da base de dados e análise do seu crescimento futuro	34
4.8 Revisão do modelo lógico final com os futuros utilizadores do sistema de base de dados ..	35
5	
5. Modelação Física	36
5.1 Tradução do Modelo Lógico para um SGBD	36
5.2 Retrato da representação física	40
5.2.1 Cliente	40
5.2.2 Bilhete.....	41
5.2.3 Percurso	41
5.2.4 Estação.....	42
5.2.5 Cidade	42
5.2.6 País	42
5.2.7 Comboio	43
5.2.8 Lugares.....	43
5.3 Implementação do Esquema Físico - Povoação.....	43
5.3.1 Tabela País	43
5.3.2 Tabela Cidades	44
5.3.3 Tabela Estação	44
5.3.4 Tabela Comboio	44
5.3.5 Tabela Cliente	45
5.3.6 Tabela Lugares	46
5.3.7 Tabela Percurso	46
5.3.8 Tabela Bilhete	46
5.4 Análise de transações	47
5.4 Estimativa dos Requisitos do Espaço em Disco	47

5.5 Definição das Vistas dos Utilizadores	53
5.6 Definir as Regras de acesso	53
5.7 Controlo de Concorrência	53
5.8 Apresentação de Queries.....	53

I. Índice de Figuras

FIGURA 1 - RELACIONAMENTO ENTRE AS ENTIDADES BILHETE E PERCURSO	18
FIGURA 2 - RELACIONAMENTO ENTRE AS ENTIDADES CLIENTE E BILHETE.....	19
FIGURA 3 - RELACIONAMENTO ENTRE AS ENTIDADES PERCURSO E COMBOIO	19
FIGURA 4 - RELACIONAMENTOS ENTRE AS ENTIDADES ESTAÇÃO E PERCURSO.....	19
FIGURA 5 - MODELO CONCETUAL	25
FIGURA 6 - EXEMPLOS DE ATRIBUTOS COMPOSTOS NO MODELO CONCETUAL E LÓGICO	28
FIGURA 7 - EXEMPLO DE ATRIBUTOS MULTIVALOR.....	29
FIGURA 8 - TABELAS PAÍS E CIDADE	29
FIGURA 9 – MAPA DE TRANSAÇÕES	31
FIGURA 10 - ESQUEMA LÓGICO.....	35
FIGURA 11 - CÓDIGO RELATIVO À TABELA CLIENTE	42
FIGURA 12 - CÓDIGO RELATIVO À CRIAÇÃO DA TABELA BILHETE	43
FIGURA 13 - CÓDIGO RELATIVO À CRIAÇÃO DA TABELA PERCURSO	43
FIGURA 14 - GERAÇÃO DA TABELA DE ESTAÇÕES DA NOSSA BASE DE DADOS	44
FIGURA 15 - CÓDIGO DE CRIAÇÃO DA TABELA CIDADE.....	44
FIGURA 16 – CÓDIGO DE GERAÇÃO DA TABELA PAÍS.....	44
FIGURA 17 - GERAÇÃO DA TABELA COMBOIO.....	45
FIGURA 18 - CÓDIGO DE CRIAÇÃO DA TABELA LUGARES.....	45
FIGURA 19 - POVOAMENTO DA TABELA PAIS E RESPETIVA TABELA	45
FIGURA 20 - POVOAMENTO DA TABELA CIDADE E RESPETIVA TABELA.....	46
FIGURA 21 - POVOAMENTO DA TABELA ESTAÇÃO E RESPETIVA TABELA.....	46
FIGURA 22 - POVOAMENTO DA TABELA COMBOIO E RESPETIVA TABELA.....	46
FIGURA 23 - POVOAMENTO DA TABELA CLIENTE	46
FIGURA 24 - TABELA CLIENTE POVOADA.....	47
FIGURA 25 - POVOAMENTO DA TABELA LUGARES E RESPETIVA TABELA.....	47
FIGURA 26 - POVOAMENTO DA TABELA PERCURSO E RESPETIVA TABELA	48
FIGURA 27 - POVOAMENTO DA TABELA BILHETE E RESPETIVA TABELA.....	49
FIGURA 28 - TRANSAÇÃO DE UM NOVO LUGAR NUM NOVO COMBOIO	49
FIGURA 29 - TRIGGER	50
30 ERRO! MARCADOR NÃO DEFINIDO.	
FIGURA 31 - GRÁFICO DE ESPAÇO OCUPADO PELO NÚMERO DE CLIENTES	51

FIGURA 32 - GRÁFICO DO ESPAÇO OCUPADO PELO Nº DE BILHETES	52
FIGURA 33 - GRÁFICO DO ESPAÇO OCUPADO PELO NÚMERO DE PERCURSOS	53
FIGURA 34 - GRÁFICO DO ESPAÇO OCUPADO PELO NÚMERO DE ESTAÇÃO.....	54
FIGURA 35 - GRÁFICO DO ESPAÇO OCUPADO PELO Nº DE COMBOIOS	55
FIGURA 36 - GRÁFICO DO ESPAÇO OCUPADO POR TODOS OS ATRIBUTOS.....	55
FIGURA 37 - VIEW 1	56
FIGURA 38 - VIEW 2	56
FIGURA 39 - VIEW 3	57
FIGURA 40 - VIEW 4	57
FIGURA 41 - VIEW 5	57
FIGURA 42 - ADMINISTRADOR.....	58
FIGURA 43 - GESTOR DE EMPRESA	58
FIGURA 44 - CLIENTE	59
FIGURA 45 - QUERIE 1	60
FIGURA 46 - RESULTADO DA QUERIE 1	60
FIGURA 47 - QUERIE 2	60
FIGURA 48 - RESULTADO QUERIE 2.....	61
FIGURA 49 - QUERIE 3	61
FIGURA 50 - RESULTADO QUERIE 3.....	61
FIGURA 51 - QUERIE 4	61
FIGURA 52 - RESULTADO QUERIE 4.....	62
FIGURA 53 - QUERIE 5	62
FIGURA 54 - RESULTADO QUERIE 5.....	62
FIGURA 55 - QUERIE 6	62
FIGURA 56 - RESPOSTA QUERIE 6.....	63

II. Índice de Tabelas

TABELA 1 - TABELA DE ENTIDADES.....	18
TABELA 2 - TABELA DOS RELACIONAMENTOS	20
TABELA 3 - TABELA DOS ATRIBUTOS	20
TABELA 4 - TABELA DAS MULTIPLICIDADES DOS RELACIONAMENTOS	33
TABELA 5 - TAMANHO OCUPADO PELOS ATRIBUTOS DA TABELA CLIENTE	50
TABELA 6 - ESTIMATIVA DE UM CRESCIMENTO FUTURO E DO ESPAÇO OCUPADO PELO NÚMERO DE CLIENTES.....	51
TABELA 7 - TAMANHO OCUPADO PELOS ATRIBUTOS DA TABELA BILHETE	51
TABELA 8 - ESTIMATIVA DE UM CRESCIMENTO FUTURO E DO ESPAÇO OCUPADO PELO Nº DE BILHETES	52
TABELA 9 - TAMANHO OCUPADO PELOS ATRIBUTOS DA TABELA PERCURSO.....	52

TABELA 10 - ESTIMATIVA DE UM CRESCIMENTO FUTURO E DO ESPAÇO OCUPADO PELO NÚMERO DE PERCURSOS.....	53
TABELA 11 - TAMANHO OCUPADO PELOS ATRIBUTOS DA TABELA ESTAÇÃO	53
TABELA 12 - ESTIMATIVA DE UM CRESCIMENTO FUTURO E DO ESPAÇO OCUPADO PELO NÚMERO DE ESTAÇÕES	54
TABELA 13 - TAMANHO OCUPADO PELOS ATRIBUTOS DA TABELA COMBOIO	54
TABELA 14 - ESTIMATIVA DE UM CRESCIMENTO FUTURO E DO ESPAÇO OCUPADO PELO Nº DE COMBOIOS	55

III. Índice de Ilustrações

ILUSTRAÇÃO 1 - DEPENDÊNCIAS DA CHAVE PRIMÁRIA DA ENTIDADE CLIENTE.....	30
ILUSTRAÇÃO 2 - DEPENDÊNCIAS DA CHAVE PRIMÁRIA DA ENTIDADE BILHETE	30

1. Introdução

<<Este primeiro capítulo deverá ter obrigatoriamente as subseções abaixo apresentadas.>>

1.1 Contextualização

Uma companhia ferroviária é uma empresa responsável por prestar serviços de deslocação via terra, entre dois pontos geográficos, através de uma vasta rede de linhas férreas que atendam a interesses económicos e sociais. Dentro de uma companhia existe um sistema próprio de controlo de reservas de viagens que permita uma organizada venda de bilhetes.

O início do transporte ferroviário data do século VI. Os primeiros vestígios da existência de uma linha férrea remontam à Grécia Antiga, por volta do século 6 a.C., servindo, na altura, para o transporte de barcos.

Em 1804 foi criada a primeira ferrovia a operar com uma locomotiva a vapor, mas só em 1825 é que foi criada a primeira companhia ferroviária, operando locomotivas a vapor, a Stockton and Darlington Railway.

O aparecimento do transporte ferroviário esteve estritamente relacionado com a Revolução Industrial, ao longo dos séculos XVIII e XIX. Este meio de transporte emergiu na Europa, mais precisamente na Inglaterra, no século XIX. As locomotivas eram movidas a vapor. Após o surgimento deste inovador transporte, rapidamente a sua tecnologia se alastrou para outros pontos do mundo.

Em Portugal, as primeiras realizações práticas no meio ferroviário, ocorreram nas décadas de 1850 e 1860. A Companhia Central Peninsular dos Caminhos de Ferro de Portugal, mais conhecida como Companhia Peninsular, foi a primeira empresa ferroviária portuguesa, que construiu o troço entre Lisboa e o Carregado da Linha do Norte.

Em 2014 nasceu uma nova empresa ferroviária, Lei Express, pensada e criada por um grupo de jovens empresários dinâmicos e ambiciosos com o objetivo de inovar o transporte dos passageiros por um preço mais acessível do que o mercado oferecia até à data. Para isto ser possível optou-se por comboios com menor capacidade mas que conseguem fazer grandes percursos, tornando assim o sistema mais acessível e económico para os clientes.

O objetivo desta nova empresa é crescer a nível internacional, melhorando os seus recursos de forma a tentar conquistar o seu lugar entre as grandes empresas do sector, e tornar-se numa das maiores potências da área em questão.

1.2 Apresentação do Caso de Estudo

Na realização deste trabalho iremos focar-nos apenas no sistema de reservas de uma rede ferroviária, mais concretamente no seu sistema de reserva de bilhetes efetuada online.

Como forma de exemplo para ajudar a entender este processo podemos dar o exemplo da CP, que é a maior companhia ferroviária em Portugal até ao momento, que sabemos que possui inúmeros comboios de diferentes tipos, como por exemplo de longo curso, internacionais e regionais, podendo ter carruagens de várias classes desde executiva a económica dependendo do tipo do comboio. A CP tem de ter uma grande capacidade de controlo em todas as suas viagens de forma a que nada falhe.

Com base neste conhecimento que possuímos vamos implementar uma pequena base de dados correspondente a uma companhia ferroviária de uma dimensão considerável que nos permitirá ter acesso a informação relativa às reservas efetuadas, sendo que para isso criamos uma lista de requisitos que a nossa base de dados terá que obedecer.

A nossa companhia é composta por 5 comboios. Todos os comboios possuem a mesma capacidade assim como a mesma estrutura, 15 lugares divididos por 3 carruagens (5 lugares em cada carruagem). Na primeira carruagem todos lugares estão destinados a bilhetes do tipo executivos. Na segunda e terceira carruagem 3 dos 5 lugares são para ser atribuídos a bilhetes do tipo turístico e os restantes 2 reservados para as pessoas com mobilidade reduzida. Também existem, ao todo, 5 linhas na nossa companhia e 5 destinos distintos: Porto, Lisboa, Madrid, Paris e Londres, perfazendo assim 10 percursos distintos, como iremos explicar posteriormente. Cada um desses percursos acontece uma ou duas vezes por dia e os horários são os mesmos todos os dias. No total existem 16 viagens por dia.

1.3 Motivação e Objectivos

Desde o início que a intenção foi criar um sistema que regista todos os bilhetes reservados na nossa companhia ferroviária. Considerando a contextualização apresentada, sabemos que o sistema de reservas é um sistema de elevado grau de complexidade de gerir. Existem aspetos bastante importantes a ter em conta tais como

ter a informação completa e atualizada e sempre disponível para uma gestão de recursos mais eficiente e que seja lucrativa para a empresa.

Uma base de dados bem construída é fundamental para o correto funcionamento do sistema de reservas de uma companhia ferroviária.

Na estruturação do problema e elaboração da lista de requisitos deparamo-nos com a possibilidade de existir uma grande variedade de entidades assim como de relacionamentos. Tendo em conta que o objetivo é ter um sistema simples de base de dados tiveram que ser tomadas decisões tendo sempre em conta que o sistema teria que possuir o máximo de informação útil que permitisse uma melhor organização.

O grande objetivo desta base de dados é modelar um protótipo de um sistema que permita a organização correta de toda a informação relativa à reserva de bilhetes de uma companhia ferroviária. Para isso é necessário analisar todas as possibilidades à disposição do utilizador para efetuar a reserva e fazer o tratamento desses dados.

A viabilidade presente na elaboração desta base de dados vai depender da possibilidade da sua implementação física a nível de terminais de acesso e também da qualidade da informação na atualização de dados.

A inserção dos registos acerca da informação pedida vai ser feita numa plataforma online onde o cliente tem de efetuar o login e ter um saldo superior ao preço da reserva que quer efetuar antes de os dados da reserva serem registados na nossa base de dados.

1.4 Estrutura do Relatório

Depois de termos apresentado a contextualização, o caso de estudo e a motivação por detrás deste, as fases seguintes do projeto são a análise e construção do Modelo Conceptual da base de dados, a passagem para o esquema Lógico correspondente e a posterior implementação do Modelo Físico, o que será tudo detalhado ao pormenor ao longo deste relatório.

Primeiramente, é realizado um estudo aprofundado do Modelo Conceptual da base de dados do sistema de reservas de bilhetes. É apresentado uma lista das chaves candidatas, onde é apresentada a chave primária assim como possíveis chaves alternativas. Posteriormente, para cada entidade é apresentada informação sobre a sua descrição e ocorrência, nos relacionamentos falar sobre a respetiva multiplicidade entre as entidades envolvidas. Em relação aos atributos, identifica-se o tipo de dados, seu

tamanho, e também se estes podem ser null, multivalorados e/ou derivados. No final desta secção apresenta-se a representação gráfica do modelo concetual.

De seguida, vamos fazer a passagem do Modelo Concetual para o Modelo Lógico. Vamos começar com a especificação das entidades e identificar os diferentes tipos de relacionamentos entre elas, analisamos também a multiplicidade de cada entidade e definimos quais iam ser as nossas chaves estrangeiras em cada tabela. Posto isto, efetuamos a normalização das tabelas aplicando as três regras de normalização. De seguida, identificamos todas as transações presentes no nosso modelo através de um desenvolvimento de um Mapa de Transações, que de certa forma vai servir para nos mostrar a sequência das ações necessárias para realizarmos uma transação válida.

Apresentamos o nosso Modelo Lógico utilizando o MySQL Workbench, que mostra todos os relacionamentos e o tipo de atributos que foram descritos anteriormente, ou seja fizemos a distinção entre os atributos que são chaves primárias e estrangeiras e os de carácter obrigatório. Tivemos que verificar a integridade deste modelo, identificando assim toda a informação sobre os seus atributos se são ou não de preenchimento obrigatório ou não e quais o seu domínio de valores. Estudarmos também a multiplicidade dos relacionamentos através da contextualização das chaves primárias e estrangeiras. De seguida, fomos justificar a integridade referencial entre as entidades, ou seja, apresenta-mos todas as regras que irão manter a consistência da informação através das tabelas que partilham uma chave estrangeira (FK). Por último, realizamos um pequeno estudo sobre a viabilidade do crescimento futuro da nossa base de dados.

Finalmente, apresentamos o Modelo Físico da nossa base de dados e descrevemos assim todo o processo que o originou. Realizamos uma descrição detalhada de todos os atributos e relacionamentos pois este modelo segue uma semântica da linguagem SQL, onde podemos identificar essa semântica quando implementamos as tabelas, onde vão ser feitas as inserções de dados. O modelo foi traduzido numa linguagem física, que tem características de tamanho que foram apresentadas numa pequena simulação do crescimento da nossa base de dados e espaço ocupado por ela ao longo do tempo. Defini-mos também as restrições de acesso que cada utilizador da base de dados vai ter, desde o administrador ao simples cliente.

Por fim, temos as conclusões que chegamos com a realização deste trabalho, os anexos e as referências bibliográficas.

2. Análise do Caso em Estudo

2.1 Análise de Requisitos

Para podermos fazer uma base de dados de um sistema de reservas de comboios primeiro necessitamos de saber como funciona uma companhia ferroviária e quais os elementos necessários para a correta construção base de dados.

No que diz respeito à companhia em si, tivemos de nos assegurar que o seu funcionamento corria sem problemas, isto é, tivemos de verificar que um comboio não tem dois percursos simultâneos, que na mesma linha não se encontram dois comboios em sentidos opostos ou até que não é atribuído um percurso Lisboa-Madrid a um comboio que à hora da viagem se encontra no Porto.

Para conseguirmos ter uma base de dados bem estruturada certas entidades têm que obedecer a certos requisitos.

- **Bilhete**

Em cada bilhete é necessário saber para que percurso foi reservado, a data do dia da viagem reservada, a indicação do lugar associado aquele bilhete, o preço do bilhete reservado e o tipo da viagem (se nacional o internacional) e também a classe desta (executiva, turística ou deficientes). Assim é possível controlar quantos bilhetes foram reservados para um certo dia, pois temos registo do dia da viagem, é possível consultar a que preço foi vendido um certo bilhete, pois haver alterações no preço dos percursos e assim fica sempre registado a que preço foi vendido um bilhete, quais os bilhetes são do tipo nacional ou internacional e também saber a classe de um bilhete.

- **Percurso**

O percurso tem que ter a informação de hora de partida e chegada, preço do percurso, estação de partida e estação de chegada e tem que estar associado a um comboio, o que irá realizar a viagem.

Seguem-se algumas *queries* que achamos importante que a nossa base de dados consiga responder

1) Quantas viagens foram reservadas para um determinado percurso num determinado dia?

Para saber quantas viagens foram reservadas num determinado percursos num determinado dia, o sistema recorre à data de cada bilhete e devolve todos os bilhetes com a data desse dia. Depois, intercalando com o percurso em questão, o sistema devolve todos os bilhetes reservados do determinado percurso.

2) Quais os lugares desocupados para um certo percurso num dado dia?

O sistema através do percurso y chega ao comboio que o percorre. Depois recorrendo à listagem dos lugares do comboio e aos lugares reservados (lugares que se encontram nos bilhetes reservados nesse dia para esse percurso nesse comboio) devolve a lista de lugares desocupados.

3) Quais os possíveis percursos a realizar num dado dia num intervalo de horas dadas?

Podemos pedir ao sistema que nos mostre todos os percursos realizados entre esse intervalo de horas.

4) Qual o histórico de reservas de um dado cliente?

Recorrendo ao id_cliente na tabela do bilhete, conseguimos que o sistema devolva todos os bilhetes que esse cliente reservou.

5) Quais os lugares desocupados no comboio x no percurso y para deficientes?

O sistema através do percurso y chega ao comboio que o percorre. Depois recorrendo à listagem dos lugares do comboio e aos lugares reservados (lugares que se encontram nos bilhetes reservados para esse percurso nesse comboio) devolve a lista de lugares de deficientes, especificando que só pretendemos estes, desocupados.

3. Modulação Concetual

3.1 Verificação das chaves candidatas e chave primária

A chave primária de uma entidade deve ser aquela cujo valor não pode repetir na mesma tabela, sendo usado assim como um índice de referência, um atributo usado para identificar. Por essas razões a chave primária nunca pode ter valor nulo.

3.1.1 Bilhete

Chaves candidatas: id_Bilhete

Chave primária: id_Bilhete

Para esta entidade a escolha da chave primária escolhida foi o id_Bilhete pois esta tem que ser algo único que identifique o bilhete em questão.

3.1.2 Comboio

Chaves candidatas: id_Comboio

Chave primária: id_Comboio

Para esta entidade a chave primária é o id_comboio, pois é a chave que reúne condições necessárias para ser uma chave candidata. Para cada comboio existe um “código” associado.

3.1.3 Percurso

Chaves candidatas: id_percurso

Chave primária: id_percurso

Sendo a única chave candidata e capaz de representar a entidade Percurso, a chave primária escolhida foi id_percurso, um número de identificação único para representar o percurso em questão.

3.1.4 Cliente

Chaves candidatas -> id_cliente e e-mail

Chave primária: id_cliente

Optámos por escolher a chave id_cliente como primária pois é apenas um número de fácil identificação e como outras as possibilidades eram mais extensas decidimos criar um id para cada bilhete.

3.1.5 Estação

Chaves candidatas: id_estação e cidade

Chave primária: id_estação

Depois de uma análise à nossa base de dados decidimos não optar pela escolha da cidade como chave primária, pois poderia resultar em complicações. Hipoteticamente, num futuro da empresa, poderá haver mais que uma estação numa cidade, dificultando assim a identificação da entidade. Decidimos criar, portanto, um atributo, id_estação, especificamente com o objetivo de identificar a estação.

3.2 Identificação dos tipos de entidade

Após efetuarmos a análise de requisitos, foi-nos possível concluir que as entidades enumeradas em baixo na tabela são essenciais para a realização deste projeto:

Entidade	Descrição	Aliases	Ocorrências
Bilhete	Termo que identifica todos os bilhetes presentes no sistema	Reserva	Cada bilhete tem um percurso e um comboio associado de uma dada viagem
Comboio	Termo que identifica todos os comboios presentes no sistema	Transporte	Cada comboio tem bilhetes associados e pode realizar vários percursos

Percurso	Termo que identifica todos os percursos presentes no sistema	Viagem	Cada percurso tem bilhetes, um comboio e duas estações associado
Cliente	Termo que identifica todos os clientes presentes no sistema	Utilizador	Cada cliente pode efetuar reservas de um dado percurso
Estação	Termo que identifica todas as estações presentes no sistema	Paragem	Cada estação tem um percurso associado

Tabela 1 - Tabela de entidades

3.3 Identificação dos tipos de relacionamento

Nesta fase, depois de já termos identificado as entidades, vamos identificar os relacionamentos existentes entre elas.

A entidade bilhete vai ser uma entidade crucial no nosso modelo, pois esta relaciona-se com todas as outras.

Existe um relacionamento entre a entidade Bilhete e a entidade Percurso, pois um percurso tem a si associado vários bilhetes, logo a cardinalidade deste relacionamento é 1:N.

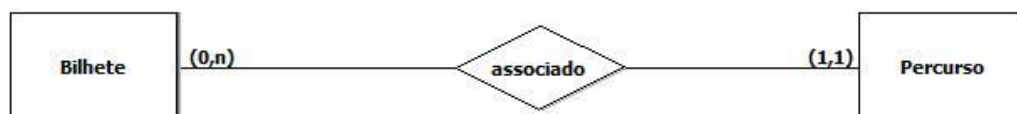


Figura 1 - Relacionamento entre as entidades Bilhete e Percurso

A entidade Bilhete também se relaciona com a entidade Cliente, ou seja o Cliente para efetuar uma viagem precisa de reservar um ou mais bilhetes, o que resulta numa cardinalidade de 1:N.

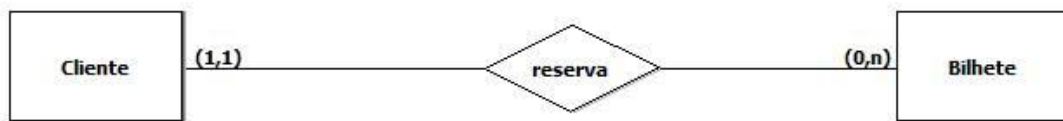


Figura 2 - Relacionamento entre as entidades Cliente e Bilhete

A entidade Comboio relaciona-se com a entidade Percurso, pois alguns comboios vão realizar vários percursos. Daí resulta uma cardinalidade de 1:N.



Figura 3 - Relacionamento entre as entidades Percurso e Comboio

A entidade Percurso vai-se relacionar com a estação de uma forma dupla, no sentido em que cada percurso vai possuir duas estações, uma é a estação de origem e a outra é a de destino. A cardinalidade do relacionamento entre a Estação e o Percurso vai ser 1:1.

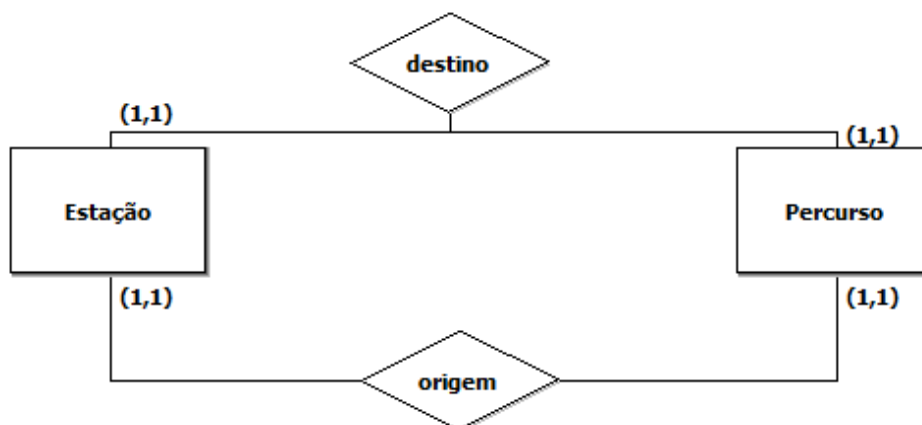


Figura 4 - Relacionamentos entre as entidades Estação e Percurso

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Cliente	1	Reserva	N	Bilhete
Bilhete	N	Associado	1	Percurso
Percurso	N	Associado	1	Comboio
Percurso	1	Origem	1	Estação
Estação	1	Destino	1	Percurso

Tabela 2 - Tabela dos relacionamentos

3.4 Identificação dos atributos

Entidade	Atributos	Data type & length	Null	Multivalor	Derivado	VP
Cliente	id_cliente (PK) e-mail password nome saldo	INT VARCHAR(90) VARCHAR(45) VARCHAR(90) FLOAT	não não não não não	não não não não não	não não não não não	-
Comboio	id_comboio (PK) capacidade númeroLugar tipo carruagem	INT INT INT INT INT	não não não não não	não não sim sim sim	não não não não não	-
Bilhete	id_bilhete (PK) carruagem númeroLugar tipo viagem data preçoBilhete	INT INT INT INT CHAR DATE FLOAT	não não não não não não não	não sim sim sim não não não	não não não não não não não	-
Percurso	id_percurso (PK) linha hora partida hora chegada preçoPercurso	INT INT TIME(0) TIME(0) INT	não não não não não	não não não não não	não não não não não	-
Estação	id_estação (PK) país cidade	INT VARCHAR(45) VARCHAR(45)	Não não não	não não não	não não não	-

Tabela 3 - Tabela dos atributos

De seguida temos uma breve descrição dos atributos presentes em cada entidade.

3.4.1 Cliente

- **id_cliente** - número que serve para identificar um cliente.
- **saldo** - valor que representa o saldo que um cliente tem na conta.
- **email** - o email vai servir de *username* para o cliente poder aceder à sua conta.
- **password** - serve para o cliente entrar na sua conta conjuntamente com o seu email.
- **nome** - nome completo do cliente.

O cliente para aceder à sua conta com o objetivo de efetuar a reserva de uma viagem tem que efetuar o login na aplicação usando o seu email e a password. Na sua conta o cliente vai ter um saldo, saldo este que resulta de um carregamento prévio da conta por parte do utilizador, que mais tarde irá servir para efetuar a reserva de um bilhete, ou seja só vai poder efetuar a reserva se tiver saldo suficiente na conta.

Por exemplo, o cliente 122 representa o senhor Pedro Rocha que se inscreveu na aplicação com o email “pedrorocha136@gmail.com” e com a password “marine”. Neste momento o saldo do senhor Rocha é de 32.16€.

3.4.2 Bilhete

- **id_Bilhete** - número que serve para identificar os bilhetes.
- **carruagem** - número que serve para identificar a carruagem onde está a reservar o bilhete.’
- **data** - é a data do dia para o qual o cliente quer efetuar a reserva.
- **preçoBilhete** - É um número que não é inteiro, que nos dá o preço de uma dada viagem em função do tipo de viagem escolhida. Este preço é calculado pelo programa que suporta a base de dados tendo em conta o preço de percurso e do tipo do lugar reservado. Por exemplo, se um percurso tem o preço de 18.00€ o preço do bilhete vai variar consoante o tipo do lugar. Se o tipo do bilhete for executivo (tipo 1) o preço do bilhete será de $18.00 * 1.50 = 27.00€$.
- **numeroLugar** - número que vai identificar o lugar do cliente.
- **Tipo** - Significa o tipo de viagem que o cliente pode escolher para efetuar a viagem. O cliente possui três opções de escolha para efetuar a viagem, que são: turístico,

deficientes e executivo. Cada tipo tem um rácio associado, ao nível da aplicação, que depois é utilizado para calcular o preço do bilhete.

→ **Viagem** - caracter que serve para identificar se a viagem é nacional ou internacional.

O bilhete reservado por um cliente irá possuir uma data do dia para o qual a viagem foi reservada, a classe da viagem, o tipo da viagem (nacional ou internacional), o identificador do comboio, uma carruagem, um número que vai identificar o sítio onde o cliente se vai sentar numa dada carruagem e o preço correspondente ao preço do bilhete.

Por exemplo, o bilhete 381 é um bilhete para uma viagem nacional, reservado para o dia 12 de Maio de 2015 do tipo executivo. O portador desse bilhete tem o lugar 3 da primeira carruagem do comboio 2 reservado. O bilhete teve um custo de 27,00€.

3.4.3 Comboio

- **id_comboio** - número que serve para identificar um comboio.
- **lugares** - lista de lugares de um comboio.
- **carruagem** - carruagem onde se encontra o lugar.
- **numeroLugar** - número do lugar de uma carruagem de um comboio.
- **tipo** - tipo do lugar (executivo, turístico ou deficiente).
- **capacidade** - capacidade máxima de passageiros num comboio.

Cada comboio tem um número fixo de lugares, representado pela capacidade e um id que vai servir para o identificar.

Por exemplo, o comboio 1 tem capacidade de 15 lugares. Estes estão organizados por tipo, carruagem e lugar.

3.4.4 Percurso

- **id_percurso** - número que serve para identificar um Percurso.
- **linha** - número que identifica uma dada linha onde um determinado comboio vai circular.
- **horaPartida** - cada percurso tem associado uma hora de partida.
- **horaChegada** - cada percurso tem associado uma hora de chegada.
- **preçoPercurso** - cada percurso tem associado a si um dado preço.

Cada percurso tem uma hora de partida e chegada. Tem também a indicação da linha pela qual vai circular e um preçoPercurso.

Por exemplo, o percurso 01 representa o percurso efetuado na linha 4 com saída às 08:45h e com chegada às 12:30h, tem o preço de 23€.

3.4.5 Estação

- ➔ **id_estacao** - número que serve para identificar cada estação na base de dados.
- ➔ **país** - identifica o país onde a estação está localizada.
- ➔ **cidade** - Identifica a cidade onde a estação está localizada.

Cada estação tem o seu id de identificação, a cidade e o país correspondente à sua localização.

Por exemplo, a estação 02 situa-se em Lisboa, Portugal.

3.5 Domínio das entidades

3.5.1 Cliente

- ➔ **id_cliente** - número de 3 dígitos que identifica cada cliente na base de dados. Ex: '122'
- ➔ **saldo** - número que pode não ser um inteiro. Ex: '32,16'.
- ➔ **email** - combinação de caracteres até 90. Ex: 'pedrorocha136@gmail.com'
- ➔ **password** - combinação de caracteres até 45. Ex: 'caneta55'
- ➔ **nome** - combinação de caracteres até 90, que representa o nome do cliente. Ex: 'Pedro Rocha'

3.5.2 Bilhete

- ➔ **id_Bilhete** - número de 6 dígitos que serve para identificar um bilhete na base de dados, O maior número de dígitos do id_bilhete comparado ao id_cliente deve-se ao facto de um cliente poder reservar vários bilhetes. Ex: '381841'

- **carruagem** - número inteiro que varia entre 1 e 3 que serve para identificar a carruagem onde está a reservar o bilhete. Cada comboio tem 3 carruagens. Ex: '1'
- **data** - atributo do tipo data. Ex: '2015-5-12'
- **preçoBilhete** - É um número que não é inteiro, que nos dá o preço de uma dada viagem em função do percurso e do tipo do lugar reservado. Ex: '18,50'
- **numeroLugar** - número inteiro de 2 dígitos que identifica o lugar do cliente. Ex: '3'
- **Tipo** (Executivo/Turístico/Deficientes) - número inteiro de 1 dígito que varia entre 1 e 3 que serve para identificar o tipo do bilhete. Ex: '1'
- **Viagem** - caracter ('N' ou 'I') que identifica a natureza da viagem (nacional ou internacional). Ex: 'N'

3.5.3 Comboio

- **id_comboio** -. É representado por um dígito entre 1 a 5. Ex: '1'
- **lugares** - Lista de lugares de um comboio. Sendo que cada lugar vai ser um conjunto de caracteres. Ex: ['1 1 01', '1 1 02', ..., '3 3 14', '3 3 15'] (classe, carruagem, lugar)
- **carruagem** - Número de 1 a 3 que identifica a carruagem do comboio.
- **numeroLugar** - Número de 01 a 15 que identifica o número de lugar no comboio.
- **tipo** - Número de 1 a 3 que identifica o tipo do lugar\viagem (executivo, turístico ou deficiente)
- **capacidade** - inteiro de 2 dígitos. Ex: '15'.

3.5.4 Percurso

- **id_percurso** - número de 2 dígitos que serve para identificar um Percurso. Ex: '01'
- **linha** - número inteiro de 1 dígito (não existem mais de 9 linhas) que identifica uma dada linha onde um determinado comboio vai circular. Ex: '4' para um percurso que seja efetuado pela linha 4.
- **horaPartida** - atributo do tipo tempo. Ex: '08:45'
- **horaChegada** - atributo do tipo tempo. Ex: '12:30'
- **preçoPercurso** - É um número que não é inteiro, que nos informa o preço de uma dada viagem, Ex: '2.38'

3.5.5 Estação

- **id_estação** - número de 2 dígitos. Ex: '02'
- **país** - Conjunto de caracteres que vão identificar um país . Ex: 'Portugal'
- **cidade** - Conjunto de caracteres que vão identificar uma cidade . Ex 'Lisboa'

3.6 Modelo Concetual

Dados todos os requisitos que consideramos em acima, vamos apresentar agora, o esquema concetual, tendo sido alvo de várias propostas de resolução ao longo da realização do trabalho e naturalmente de mudanças, com o intuito de ter o melhor modelo possível. Como é normal as considerações e o acompanhamento das várias etapas da metodologia foram essenciais para a elaboração do mesmo, pois se não tivéssemos isso em conta não seria possível apresentar o modelo numa forma simples e perceptível.

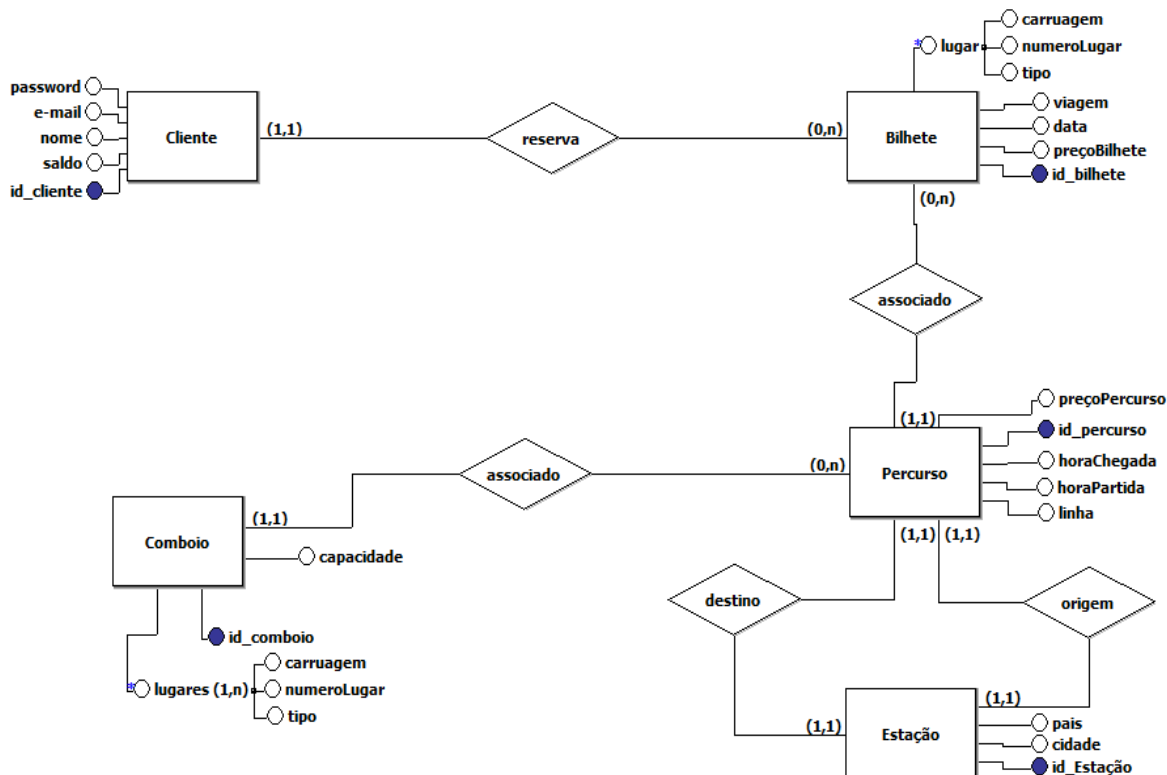


Figura 5 - Modelo concetual

4. Modulação lógica de dados

4.1 Passagem do modelo de dados concetual para um seu correspondente modelo de dados lógico

4.1.1 Entidades Fortes

Uma entidade forte é uma entidade que possui uma chave primária que a identifique unicamente, não dependendo assim de outras chaves.

Foi criada uma relação com os atributos simples e foram introduzidos no relacionamento quanto aos compostos consideramos apenas os atributos simples correspondentes. Todos os nomes das relações começam com letra maiúscula e aos atributos foram removidos os espaços e a sua respetiva acentuação.

→ Clientes

Chave-primária: id_Cliente

→ Comboio

Chave-primária: id_Comboio

4.1.2 Relacionamentos 1:N

Em cada relacionamento 1:N, foi criada uma cópia da chave primária da relação-pai e esta foi colocada na relação-filho (sendo que a relação-filho é a relação de menor cardinalidade) na forma de chave estrangeira (FK). Esta chave tem sempre nome igual à da relação-pai. No nosso modelo temos vários exemplos deste relacionamento, como por exemplo um Cliente reserva Bilhete.

4.1.3 Relacionamentos 1:1

Cada uma das duas entidades envolvidas referenciam obrigatoriamente e exclusivamente uma unidade da outra, podendo-se introduzir a chave primária da primeira entidade como chave estrangeira da segunda. Por exemplo, um determinado percurso tem origem em uma estação e destino em uma estação também.

4.1.4 Relacionamentos N:M

Não temos nenhum relacionamento deste tipo presente no modelo. Em cada relacionamento N:M é feita através da criação de dois novos relacionamentos 1:N, que contém as chaves primárias das entidades iniciais como chaves estrangeiras e também poderá possuir atributos que definem este novo relacionamento.

4.1.5 Relacionamentos Complexos

Um relacionamento complexo é relacionamento que possui um grau igual ou superior a 3, ou seja, quando 3 ou mais entidades intervêm. No modelo relacional, estes casos podem ser desfeitos encontrando uma entidade intermédia, sendo assim as entidades ficam relacionadas com relacionamentos de grau 2. Neste modelo não possuímos nenhum relacionamento destes.

4.1.6 Atributos multivalor

Um atributo multivalor é um atributo que pode tomar um ou mais valores para cada entidade.

No nosso modelo temos um atributo multivalor que vai dar origem a uma nova relação, cujo nome é o nome do atributo. Neste tipo de relações vamos ter uma chave primária composta, colocando assim uma cópia da chave primária da relação à qual pertencem os atributos, funcionando assim como uma chave estrangeira, e ainda acrescentamos um atributo identificador. O atributo multivalor presente no modelo é Lugares do comboio.

4.2 Validação do modelo lógico através da normalização

Ao passarmos do modelo concetual para o modelo lógico existem, por vezes, algumas anomalias que podem tornar a base de dados inconsistente. Devido a isso, vamos ter que validar o modelo lógico através da normalização.

Para o nosso caso de estudo vamos considerar algumas das mais importantes normas para a validação do modelo fundamentando sempre as decisões efetuadas.

4.2.1 1ª Forma Normal

A primeira forma normal é válida quando todos os atributos que forem atômicos, isto é, estarem numa forma em que não é possível decompô-los mais. Ao fazermos a interseção entre cada linha e cada coluna só vai ter apenas um valor.

Num primeiro momento, analisamos as tabelas do modelo à procura de grupos repetidos, que existiriam nas tabelas se elas não fossem normalizadas. Assim identificamos os atributos multivalor e os compostos.

i. Atributos compostos

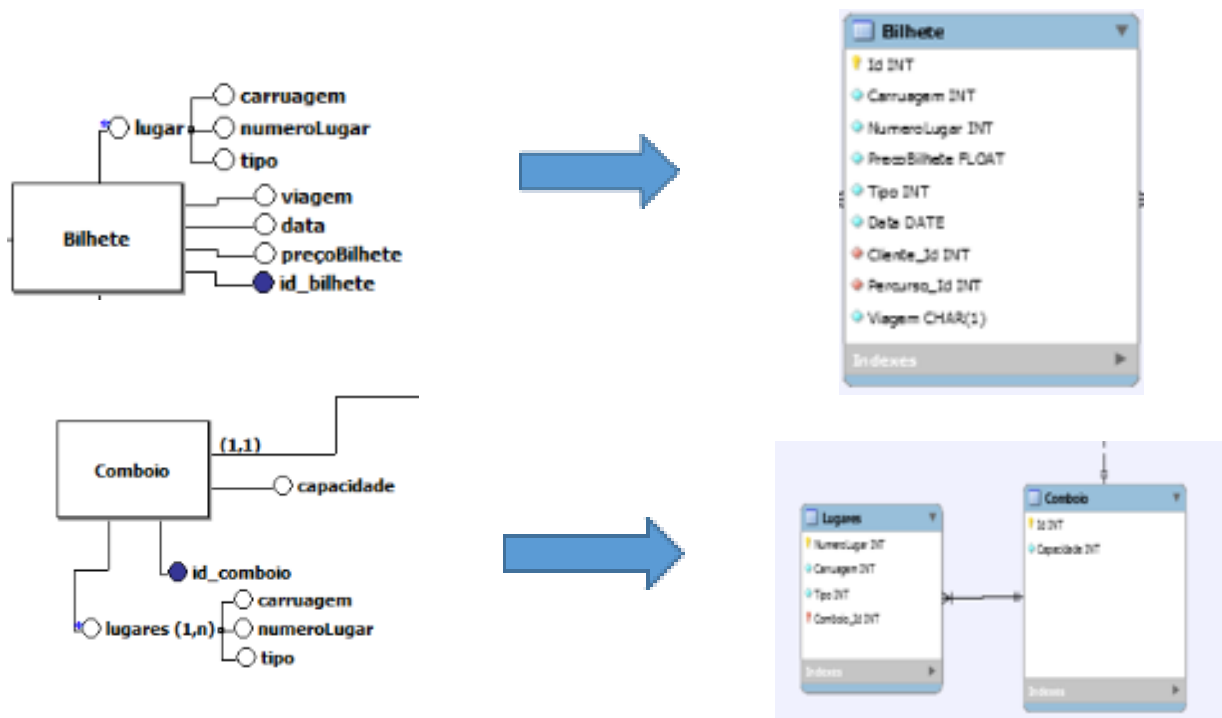


Figura 6 - Exemplos de atributos compostos no modelo conceitual e lógico

ii. Atributos multivalor

No nosso modelo conceitual possuímos apenas um atributo Multi Valor que é Lugares (Comboio). Ao efetuarmos a transição para o modelo lógico, os atributos vão dar origem a uma tabela adicional e que vai ter obrigatoriamente pertencer à primeira forma normal.

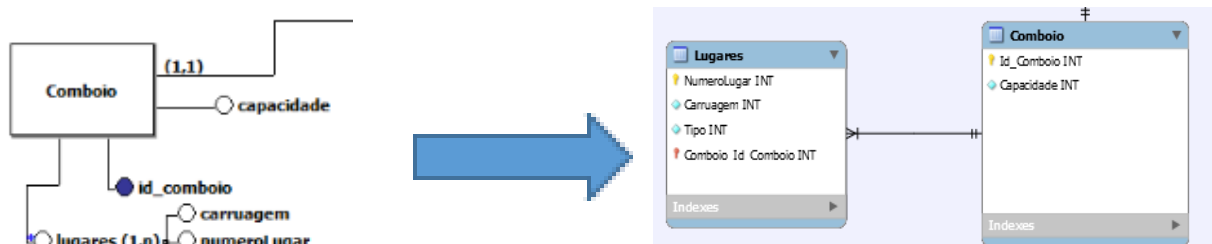


Figura 7 - Exemplo de atributos multivalor

iii. Explicação da criação das tabelas Países e Cidades

Criou-se tabelas para as cidades e para os países de forma a melhor gerir as estações nestes locais. Ou seja, se quisermos ver as estações que a empresa tem em funcionamento em determinado país ou mesmo numa cidade, torna-se assim muito mais fácil.

Outra das razões é a de evitar redundâncias no sistema. Imaginemos que a empresa tinha uma estação em Vila Nova de Famalicão, a referência a V. N. de Famalicão, Vila Nova de Famalicão ou simplesmente Famalicão seria exatamente à mesma cidade mas o nosso sistema iria considerar como cidades diferentes. Ao criarmos tabelas para as cidades evitamos este tipo de situações.

Relacionamento 1:1 Estação-Cidade:

Apenas existe uma estação por cidade. Caso o cenário se altere é preciso atualizar a base de dados de forma a alterar o relacionamento de 1:1 para N:1.

Relacionamento N:1 Cidade-País:

Como várias cidades diferentes podem fazer parte do mesmo país, a relação da cidade-país é de N:1.

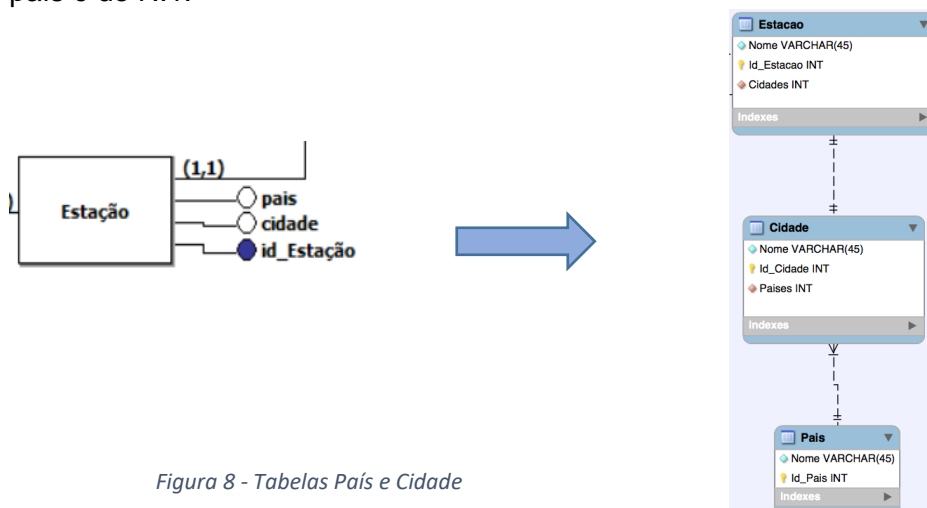


Figura 8 - Tabelas País e Cidade

4.2.2 2ª Forma Normal

A segunda forma normal tem como objetivo principal eliminar redundâncias e analisar dependências parciais. Neste momento, o modelo já verifica a primeira Forma Normal e cada atributo que não pertença à chave primária vai depender totalmente desta. O nosso modelo já se encontra na segunda forma normal pois não possuímos nenhuma dependência parcial. De seguida temos um exemplo de uma dependência total, 'password', 'nome', 'email' e 'saldo' dependem da chave primária 'id_cliente'.

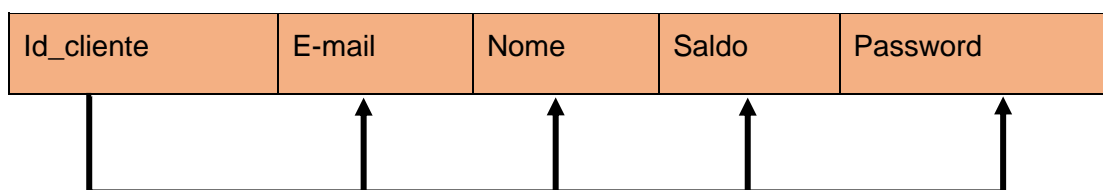


Ilustração 1 - Dependências da chave primária da entidade Cliente

4.2.3 3ª Forma Normal

O objetivo da terceira forma é análise das dependências transitivas. Por exemplo, observamos a tabela Bilhete existe a capacidade que vai corresponder a cada classe e sendo assim não existe necessidade de existir um atributo Lugar pois podemos calculá-lo somando os três atributos.

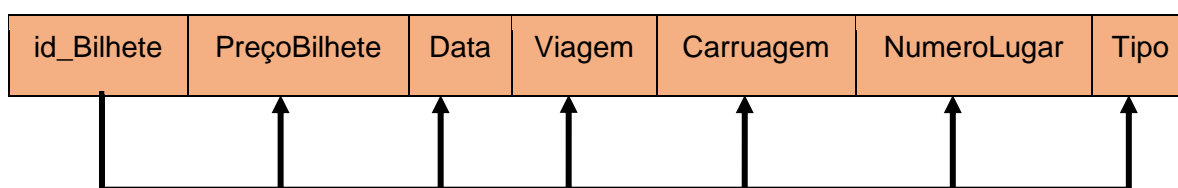


Ilustração 2 - Dependências da chave primária da entidade Bilhete

A terceira forma normal é aquela que geralmente nos indica o fim do processo de normalização, mas em alguns casos específicos ainda transporta alguns problemas. Mas neste caso em específico não vamos continuar com o processo de normalização. Esta incerteza de continuidade do processo de normalização ou não remete-nos para um dilema, pois por um lado pretendemos sistemas flexíveis, sem problemas de redundância e por outro lado exige-se sistemas de alto desempenho. Para resolvermos este dilema temos de chegar a um consenso de modo a construir um esquema

equilibrado que nunca ponha em risco a integridade da base de dados, mas ao mesmo tempo tenho um desempenho aceitável. Só com o equilíbrio entre essas duas características é que vamos obter uma base de dados pronta a ser utilizada.

4.3 Validação do modelo com as transações do utilizador – elaboração do correspondente mapa de transações

Ao nível do Modelo Lógico, fizemos uma validação das transações do utilizador através de mapas de transações. Desta forma, é-nos possível:

➔ Adicionar/remover Lugar

Esta transação é uma adição de um lugar no comboio, que é feita na tabela Lugares. Ao fazermos esta alteração, vamos alterar a tabela comboio, mais especificamente a capacidade do comboio, que vai incrementar.

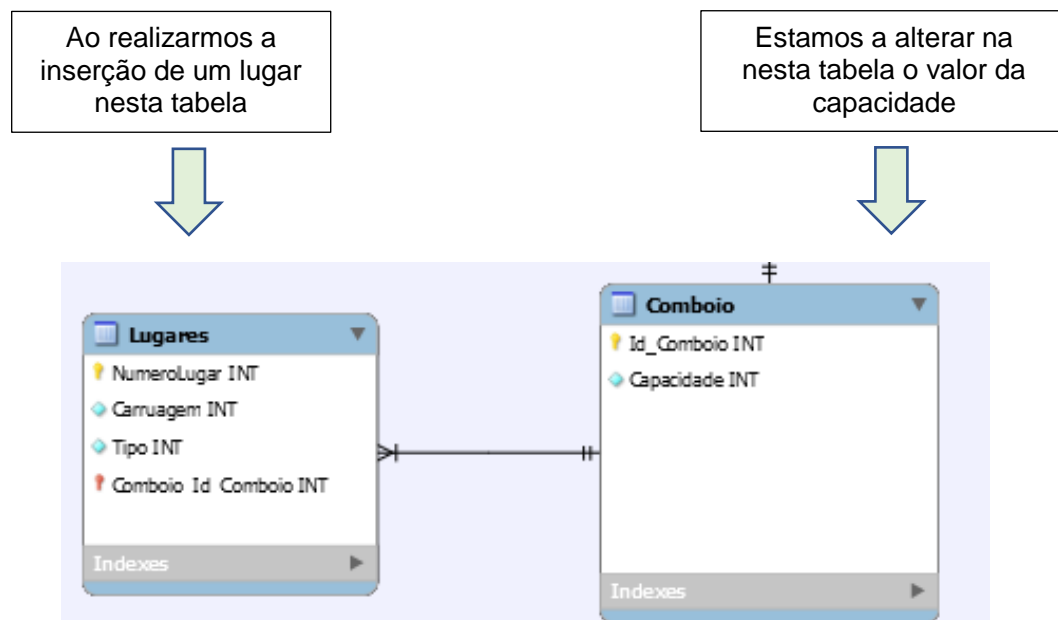


Figura 9 – Mapa de transações

4.4 Elaboração e validação do esquema lógico da base de dados

Com base na metodologia específica apresentada no livro de base de dados recomendado e tendo em conta o esquema lógico, este parâmetro vem complementar a informação que já foi descrita no capítulo “Informação e domínios dos atributos”.

4.5 Regras de integridade

As regras de integridade vão garantir uma base de dados completa e consistente. Mas para isso é preciso respeitar as seguintes regras:

➔ Necessidade de valores

Alguns dos atributos têm obrigatoriamente, que possuir um valor não nulo, como por exemplo as chaves primárias. Ficou decidido que a maior parte dos atributos tinha de possuir um valor válido, para a base de dados possuir um melhor controlo de todos os seus dados. De seguida vamos dar exemplos de atributos que não são chaves primárias mas são essenciais:

- DataPartida/ DataChegada: atributo da entidade Percurso que indica as horas de partida e de chegada de uma viagem.
- Nome: atributo da entidade Cliente que serve para identificar quem vai efetuar uma viagem, logo não pode ser nulo.
- Preço: atributo da entidade Bilhete que vai definir o preço de um bilhete logo não pode ser nulo.

➔ Restrições do domínio do atributo

Todos os atributos têm de ser associados a um intervalo de valores que podem tomar. Podemos verificar isso No anexo número 2.

➔ Multiplicidade

A base de dados vai ser composta por diversos relacionamentos cuja multiplicidade deve ser respeitada ao passarmos do modelo conceitual para o modelo lógico. Por vezes também são criados novos relacionamentos. Esses novos relacionamentos costumam surgir por possuímos atributos de multivalor e/ou possuímos relacionamentos N:M. Como no modelo só temos atributos de multivalor, vamos ter de criar um novo relacionamento com uma cardinalidade 1:N com a tabela que foi criada. De seguida vamos apresentar a multiplicidade das novas tabelas.

Entidade	Multiplicidade	Multiplicidade	Entidade
Comboio	N	1	Comboio
Estação	1	1	Cidade
Cidades	N	1	País

Tabela 4 - Tabela das multiplicidades dos relacionamentos

➔ Integridade da Entidade

Esta regra afirma que nenhuma chave primária pode assumir valores nulos. Podemos verificar isso anexo no anexo número 2.

➔ Integridade Referencial

Sabemos que o valor de uma chave estrangeira numa relação “filho” tem de existir sempre na chave primária da relação “pai”. Ou seja, sempre que estamos a fazer uma alteração na relação “pai”, essa alteração vai-se refletir nas chaves estrangeiras a que está associada.

No caso da entidade Percurso, é indispensável que a chave estrangeira “id_comboio” exista, pois não é possível realizarmos um dado percurso ferroviário sem um comboio para o efetuar.

Para o segundo problema de integridade referencial, usando as entidades Cliente e Bilhete, é essencial considerar os seguintes 5 casos:

- **Inserir um registo na relação filho**

Vai verificar se a chave estrangeira na relação filho têm algum tipo de correspondência com o valor da chave primária do seu pai, ou seja, para a integridade referencial se verificar temos de assegurar que a chave estrangeira, Cliente, do novo elemento da entidade bilhete está definido.

- **Eliminar um registo da relação filho**

Ao efetuarmos uma remoção na relação filho não vai afetar a integridade em qualquer aspeto, ou seja se removermos um Bilhete a entidade Cliente não é afetada nem a integridade referencial é posta em causa.

- **Atualizar o registo da chave estrangeira na relação filho**

Ao atualizarmos o registo vamos ter de verificar que o atributo Cliente é a chave estrangeira na tabela Bilhete e também tem de ser um valor existente (valor não nulo) da tabela Cliente.

- **Inserir um registo na relação pai**

Ao realizarmos uma inserção na relação pai em nada vai interferir com a integridade referencial. Mas se quisermos manter a informação atualizada devemos também atualizar a relação filho. Por exemplo, ao inserimos um cliente não vai afetar a integridade referencial, pois esse cliente vai ser um cliente sem nenhuma reserva de bilhete efetuada.

- **Remover registo na relação pai**

Ao realizarmos uma remoção de um registo na relação pai a integridade vai ser colocada em causa, pois pode passar a haver um registo na relação filho que não possua uma entidade pai. Por exemplo ao efetuarmos a remoção de um dado Cliente e se existir um registo na entidade Bilhete que faça referência a esse Cliente a integridade referencial não vai ser garantida. (Para prevenirmos este tipo de situações adotamos uma estratégia NO ACTION que vai servir para prevenirmos que estes tipos de casos aconteçam.)

4.6 Esquema Lógico

De acordo com o nosso modelo concetual vamos apresentar o nosso modelo lógico. Neste modelo vão surgir as tabelas que vão corresponder às diversas entidades assim como dos vários relacionamentos. De seguida apresentamos um modelo representativo dos vários tipos de relacionamentos.

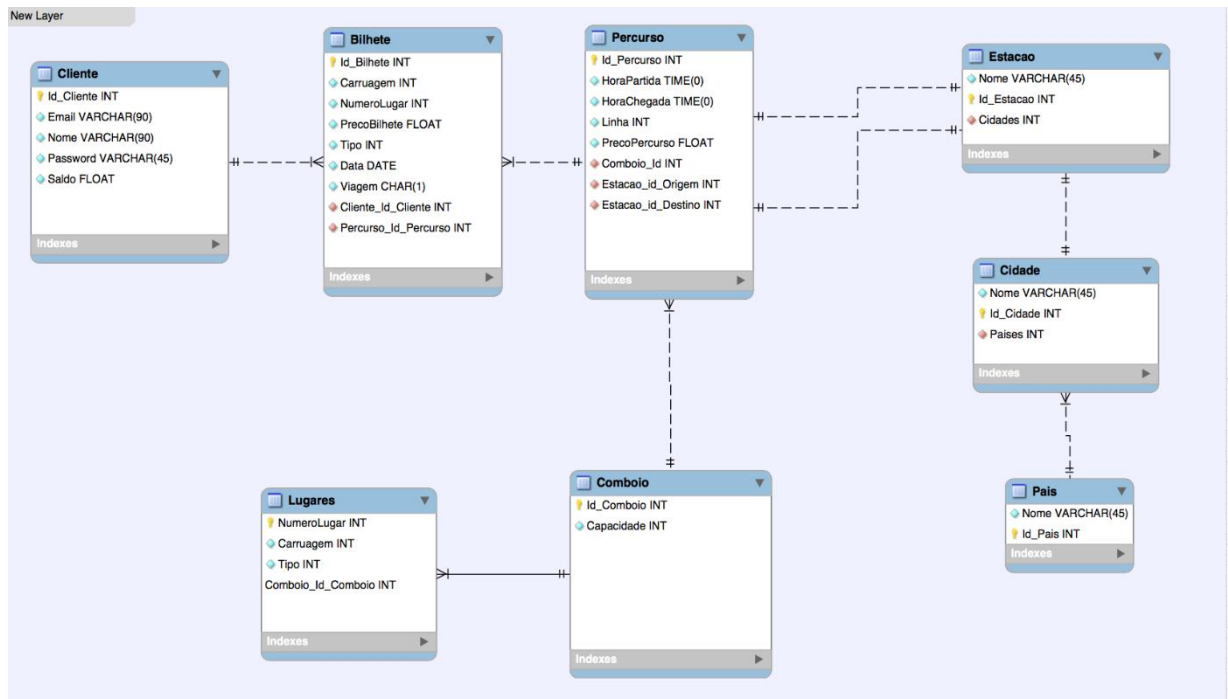


Figura 10 - Esquema lógico

4.7 Definição do tamanho inicial da base de dados e análise do seu crescimento futuro

O tamanho inicial da nossa Base de Dados vai ser o seguinte:

- Nº de Comboios: 5
- Nº de Percursos: 16
- Nº de Clientes: 13
- Nº de Lugares de um Comboio: 15
- Nº de Classes: 3
- Nº de Lugares Executivos: 5
- Nº de Lugares Turísticos: 6
- Nº de Lugares Deficientes: 4
- Nº de Carruagens: 3

- Composição da 1ª Carruagem: 5 Executivos
- Composição da 2ª e 3ª Carruagem: 3 Turísticos + 2 deficientes

O crescimento da nossa empresa vai depender da longevidade que a nossa base de dados tiver, ou seja vai depender da capacidade adaptativa a novos requisitos que sejam propostos. Por exemplo, se tivermos uma base de dados que só suporta os requisitos para a qual foi projetada mais tarde pode-se tornar um sistema obsoleto ou bastante dispendioso por causa das diversas atualizações a efetuar na base de dados para suportar todos os requisitos. Dentro do contexto em estudo, seria fácil uma implementação de novas funcionalidades como por exemplo, novas categorias de comboios (urbano, regionais, alfas...)

O modelo apresentado neste trabalho está um pouco limitado ao sistema de reservas de bilhetes de companhias ferroviárias. Para além dos requisitos pedidos ainda podíamos acrescentar mais para fazer da nossa base de dados mais útil. A gestão de reservas de bilhetes não tem que ficar limitada apenas aos requisitos pedidos. Poderia-se aumentar certos atributos nas entidades Estação e Comboio por exemplo, como certas funções específicas para cada comboio como: acesso a Internet, presença de WC, bar ou outro lugar destinado a venda de comida e bebida, etc... Ao nível do Cliente, poderíamos implementar um passe que servia para descontar no preço do bilhete ou então acrescentar um atributo que servia para saber se esse determinado cliente usufruía de outro tipo de descontos como: desconto de sénior, desconto de criança, desconto de grupo, desconto de certos eventos especiais ou desconto de outras promoções relativas à empresa.

Certamente que estas ideias não foram acrescentadas à base de dados por concluirmos que não queríamos uma base de dados demasiado "pesada" e que não ia ser de grande utilidade no momento inicial, pois não ia ser algo obrigatório de estar presente na implementação da base de dados, mas para acrescentar num futuro, com vista à melhoria da empresa, sem nunca retirar aquilo que consideramos fundamental para o bom funcionamento da mesma.

4.8 Revisão do modelo lógico final com os futuros utilizadores do sistema de base de dados

Para dar como terminada esta fase, o modelo lógico deve ser visto na perspetiva do utilizador. Este processo tem um papel muito importante pois o utilizador tem que ter

a capacidade de reconhecer, que o modelo lógico idealizado é uma representação real dos requisitos da empresa que está a ser modelada.

Por isso é que tivemos de avaliar toda a documentação associada ao modelo lógico desde as entidades até aos relacionamentos entre elas passando pelos atributos. Ao efetuarmos a revisão do modelo foi-nos possível identificar todas as tabelas importantes no modelo e os seus diversos relacionamentos. Ao revermos o nosso dicionário de relacionamentos (tabela de relacionamentos) conseguimos justificar que os relacionamentos utilizados são aqueles que melhor se identificam com a realidade do problema. De seguida, validamos os atributos onde o utilizador consegue verificar toda a informação de uma entidade e ter uma melhor noção dos domínios de cada atributo.

Por último, fizemos a revisão geral do nosso modelo, ou seja, fomos verificar se o nosso modelo respondia a certas perguntas do nosso problema de forma a testarmos as capacidades do mesmo. Sendo assim, o nosso modelo foi aprovado pelo cliente pois ele é visto como uma solução que responde a todas as necessidades pretendidas.

5. Modelação Física

Nesta parte do relatório efetuamos a tradução do modelo lógico anteriormente implementado para o modelo físico a ser implementado no SGBD.

Para construirmos a Base de Dados proposta utilizamos como Sistema de Base de Dados o MySQL Workbench da Oracle. Este sistema foi escolhido devido a ser o sistema usado nas aulas práticas na Unidade Curricular, o que de certa forma veio facilitar a nossa implementação da Base de Dados.

5.1 Tradução do Modelo Lógico para um SGBD

O processo de modelação de um esquema físico de uma base de dados envolve a tradução dos relacionamentos, definidos previamente no Modelo Lógico, num código que passa a ser implementado no SGBD. Este processo vai ser dividido em dois subprocessos, em que o primeiro é responsável por tratar da informação de todos os relacionamentos e todos os atributos, que foi gerado durante a criação do Modelo Lógico, juntamente com a informação do tratamento de requisitos do Modelo Concetual. O segundo vai usar toda a informação recolhida para reproduzir o Modelo Físico dos relacionamentos base e restrições gerais.

Como a base de dados não possui restrições gerais nem atributos derivados vai ser descrita apenas por relacionamentos base. Relacionamentos base consistem nos seguintes parâmetros: Nome do relacionamento, chave primária e por vezes também pode possuir uma chave estrangeira, conjunto de restrições de integridade referencial para cada chave estrangeira identificada. Cada atributo vai possuir os seguintes parâmetros: um domínio, um valor por defeito, se o atributo é derivado ou não, e se for como é calculado e por último vamos verificar se um atributo suporta ou não valores nulos.

➔ Descrição do relacionamento Cliente e dos seus atributos

Domínio id_Cliente: Inteiro.

Domínio Email: Sequência de caracteres com tamanho máximo igual a 90.

Domínio Nome: Sequência de caracteres com tamanho máximo igual a 90.

Domínio Password: Sequência de caracteres com tamanho máximo igual a 45.

Domínio Saldo: Float.

Cliente (

Id_Cliente NOT NULL,

Email NOT NULL,
Nome NOT NULL,
Password NOT NULL,
Saldo NOT NULL,
PRIMARY KEY (Id_Cliente));
)

➔ **Descrição do relacionamento Bilhete e dos seus atributos**

Domínio Id_Bilhete: Inteiro
Domínio Carruagem: Inteiro
Domínio NúmeroLugar: Inteiro
Domínio PreçoBilhete: Float
Domínio Tipo: Inteiro
Domínio Data: Data, Formato: 'AAAA-MM-DD'
Domínio Viagem: Char
Domínio Cliente_Id_Cliente: Inteiro
Domínio Percurso_Id_Percurso: Inteiro

Bilhete (

Id_Bilhete NOT NULL,
Carruagem NOT NULL,
NúmeroLugar NOT NULL,
PreçoBilhete NOT NULL,
Tipo NOT NULL,
Data NOT NULL,
Viagem NOT NULL,
Cliente_Id_Cliente: NOT NULL,
Percurso_Id_Percurso: NOT NULL,

PRIMARY KEY (Id_Bilhete),

FOREIGN KEY (Cliente_Id_Cliente) **REFERENCES** Cliente (Id_Cliente),

FOREIGN KEY (Percurso_Id_Percurso) **REFERENCES** Percurso

(Id_Percurso));

➔ **Descrição do relacionamento Percurso e dos seus atributos**

Domínio Id_Percurso : Inteiro
Domínio HoraPartida: Data, Formato: 'AAAA-MM-DD'
Domínio HoraChegada: Data, Formato: 'AAAA-MM-DD'

Domínio Linha: Inteiro

Domínio precoPercurso: Float

Domínio Comboio_Id: Inteiro

Domínio Estacao_id_Origem: Inteiro

Domínio Estacao_id_Destino: Inteiro

Percurso (

Id_Percurso NOT NULL,

HoraPartida NOT NULL,

HoraChegada NOT NULL,

Linha NOT NULL,

PrecoPercurso NOT NULL,

Comboio_Id NOT NULL,

Estacao_id_Origem NOT NULL,

Estacao_id_Destino NOT NULL,

PRIMARY KEY (Id_Percurso),

FOREIGN KEY (Estacao_Id_Origem) **REFERENCES** Estacao (Id_Estacao),

FOREIGN KEY (Estacao_Id_Destino) **REFERENCES** Estacao (Id_Destino));

➔ **Descrição do relacionamento Estação e dos seus atributos**

Domínio Id_Estacao: Inteiro

Domínio Cidades: Inteiro

Domínio Nome: Data, Formato: 'AAAA-MM-DD'

Estacao (

Id_Estacao: NOT NULL,

Cidades: NOT NULL,

Nome: NOT NULL,

PRIMARY KEY (Id_Estacao),

FOREIGN KEY (Cidades) **REFERENCES** Cidades (Id_Cidade));

➔ **Descrição do relacionamento Cidades e dos seus atributos**

Domínio Id_Cidade: Inteiro

Domínio Nome: Data, Formato: 'AAAA-MM-DD'

Domínio Países: Inteiro

Cidade (

Id_Cidade: NOT NULL,
Nome: NOT NULL,
Países: NOT NULL,
PRIMARY KEY (Id_Cidade),
FOREIGN KEY (Países) **REFERENCES** Pais (Id_Pais));

→ **Descrição do relacionamento Países e dos seus atributos**

Domínio Nome: Sequência de caracteres com tamanho máximo igual a 45.

Domínio Id_Pais: Inteiro

Países (

Nome NOT NULL,
Id_Pais NOT NULL,
PRIMARY KEY (Id_Pais));

→ **Descrição do relacionamento Comboio e dos seus atributos**

Domínio Id_Comboio: Inteiro

Domínio Capacidade: Inteiro

Comboio (

Id_Comboio NOT NULL,
Capacidade NOT NULL,
PRIMARY KEY (Id_Comboio));

→ **Descrição do relacionamento Lugares e dos seus atributos**

NumeroLugar: Inteiro

Carruagem: Inteiro

Tipo: Inteiro

Comboio_Id_Comboio: Inteiro

Lugares (

NumeroLugar NOT NULL
Carruagem NOT NULL
Tipo NOT NULL
Comboio_Id_Comboio NOT NULL
PRIMARY KEY (NumeroLugar,Comboio_Id_Comboio),

FOREIGN KEY (Comboio_Id_Comboio) REFERENCES Comboio (Id_Comboio));

5.2 Retrato da representação física

Para conseguirmos gerar o esquema físico utilizamos a ferramenta *Forward Enginner* do programa *MySQL Workbench*. Esta funcionalidade tem a capacidade de criar código SQL para a criação das tabelas.

De seguida vamos apresentar o código equivalente às tabelas:

- ➔ Cliente
- ➔ Bilhete
- ➔ Percurso
- ➔ Estação
- ➔ Cidade
- ➔ Pais
- ➔ Comboio
- ➔ Lugares

De seguida vamos apresentar o código equivalente às tabelas:

5.2.1 Cliente

```
DROP TABLE IF EXISTS `leiExpress`.`Cliente` ;  
  
CREATE TABLE IF NOT EXISTS `leiExpress`.`Cliente` (  
  `Id_Cliente` INT NOT NULL,  
  `Email` VARCHAR(90) NOT NULL,  
  `Nome` VARCHAR(90) NOT NULL,  
  `Password` VARCHAR(45) NOT NULL,  
  `Saldo` FLOAT NOT NULL,  
  PRIMARY KEY (`Id_Cliente`))  
ENGINE = InnoDB;
```

Figura 11 - Código relativo à tabela Cliente

5.2.2 Bilhete

```
DROP TABLE IF EXISTS `leiExpress`.`Bilhete` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Bilhete` (
  `Id_Bilhete` INT NOT NULL,
  `Carruagem` INT NOT NULL,
  `NumeroLugar` INT NOT NULL,
  `PrecoBilhete` FLOAT NOT NULL,
  `Tipo` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Viagem` CHAR(1) NOT NULL,
  `Cliente_Id_Cliente` INT NOT NULL,
  `Percurso_Id_Percurso` INT NOT NULL,
  PRIMARY KEY (`Id_Bilhete`),
  INDEX `fk_Bilhete_Cliente1_idx` (`Cliente_Id_Cliente` ASC),
  INDEX `fk_Bilhete_Percurso1_idx` (`Percurso_Id_Percurso` ASC),
  CONSTRAINT `fk_Bilhete_Cliente1`
    FOREIGN KEY (`Cliente_Id_Cliente`)
      REFERENCES `leiExpress`.`Cliente` (`Id_Cliente`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bilhete_Percurso1`
    FOREIGN KEY (`Percurso_Id_Percurso`)
      REFERENCES `leiExpress`.`Percurso` (`Id_Percurso`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 12 - Código relativo à criação da tabela Bilhete

5.2.3 Percurso

```
DROP TABLE IF EXISTS `leiExpress`.`Percurso` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Percurso` (
  `Id_Percurso` INT NOT NULL,
  `HoraPartida` TIME(0) NOT NULL,
  `HoraChegada` TIME(0) NOT NULL,
  `Linha` INT NOT NULL,
  `PrecoPercurso` FLOAT NOT NULL,
  `Comboio_Id` INT NOT NULL,
  `Estacao_id_Origem` INT NOT NULL,
  `Estacao_id_Destino` INT NOT NULL,
  PRIMARY KEY (`Id_Percurso`),
  INDEX `fk_Percurso_Comboio1_idx` (`Comboio_Id` ASC),
  INDEX `fk_Percurso_Estacao1_idx` (`Estacao_id_Origem` ASC),
  INDEX `fk_Percurso_Estacao2_idx` (`Estacao_id_Destino` ASC),
  CONSTRAINT `fk_Percurso_Comboio1`
    FOREIGN KEY (`Comboio_Id`)
      REFERENCES `leiExpress`.`Comboio` (`Id_Comboio`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Percurso_Estacao1`
    FOREIGN KEY (`Estacao_id_Origem`)
      REFERENCES `leiExpress`.`Estacao` (`Id_Estacao`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Percurso_Estacao2`
    FOREIGN KEY (`Estacao_id_Destino`)
      REFERENCES `leiExpress`.`Estacao` (`Id_Estacao`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 13 - Código relativo à criação da tabela Percurso

5.2.4 Estação

```
DROP TABLE IF EXISTS `leiExpress`.`Estacao` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Estacao` (
  `Nome` VARCHAR(45) NOT NULL,
  `Id_Estacao` INT NOT NULL,
  `Cidades` INT NOT NULL,
  PRIMARY KEY (`Id_Estacao`),
  INDEX `fk_Estação_Cidades1_idx` (`Cidades` ASC),
  CONSTRAINT `fk_Estação_Cidades1`
    FOREIGN KEY (`Cidades`)
      REFERENCES `leiExpress`.`Cidade` (`Id_Cidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 14 - Geração da tabela de estações da nossa Base de Dados

5.2.5 Cidade

```
DROP TABLE IF EXISTS `leiExpress`.`Cidade` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Cidade` (
  `Nome` VARCHAR(45) NOT NULL,
  `Id_Cidade` INT NOT NULL,
  `Países` INT NOT NULL,
  PRIMARY KEY (`Id_Cidade`),
  INDEX `fk_Cidades_Paises1_idx` (`Países` ASC),
  CONSTRAINT `fk_Cidades_Paises1`
    FOREIGN KEY (`Países`)
      REFERENCES `leiExpress`.`Pais` (`Id_Pais`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 15 - Código de criação da tabela Cidade

5.2.6 País

```
DROP TABLE IF EXISTS `leiExpress`.`Pais` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Pais` (
  `Nome` VARCHAR(45) NOT NULL,
  `Id_Pais` INT NOT NULL,
  PRIMARY KEY (`Id_Pais`))
ENGINE = InnoDB;
```

Figura 16 – Código de geração da tabela País

5.2.7 Comboio

```
DROP TABLE IF EXISTS `leiExpress`.`Comboio` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Comboio` (
  `Id_Comboio` INT NOT NULL,
  `Capacidade` INT NOT NULL,
  PRIMARY KEY (`Id_Comboio`))
ENGINE = InnoDB;
```

Figura 17 - Geração da tabela Comboio

5.2.8 Lugares

```
DROP TABLE IF EXISTS `leiExpress`.`Lugares` ;

CREATE TABLE IF NOT EXISTS `leiExpress`.`Lugares` (
  `NumeroLugar` INT NOT NULL,
  `Carruagem` INT NOT NULL,
  `Tipo` INT NOT NULL,
  `Comboio_Id_Comboio` INT NOT NULL,
  PRIMARY KEY (`NumeroLugar`, `Comboio_Id_Comboio`),
  INDEX `fk_Lugares_Comboio1_idx` (`Comboio_Id_Comboio` ASC),
  CONSTRAINT `fk_Lugares_Comboio1`
    FOREIGN KEY (`Comboio_Id_Comboio`)
      REFERENCES `leiExpress`.`Comboio` (`Id_Comboio`)
    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;
```

Figura 18 - Código de criação da tabela Lugares

5.3 Implementação do Esquema Físico - Povoação

5.3.1 Tabela Pais

```
INSERT INTO Pais(nome,Id_Pais)
VALUES
  ('Portugal',1),
  ('Espanha',2),
  ('França',3),
  ('Inglaterra',4);
```

nome	id_Pais
Portugal	1
Espanha	2
França	3
Inglaterra	4

Figura 19 - Povoamento da tabela Pais e respetiva tabela

5.3.2 Tabela Cidade

```
INSERT INTO Cidade(nome,Id_Cidade,Países)
VALUES
('Porto',1,1),
('Lisboa',2,1),
('Madrid',3,2),
('Paris',4,3),
('Londres',5,4);
```

nome	Id_Cidade	Países
Porto	1	1
Lisboa	2	1
Madrid	3	2
Paris	4	3
Londres	5	4

Figura 20 - Povoamento da tabela Cidade e respetiva tabela

5.3.3 Tabela Estação

```
INSERT INTO Estação(nome,id_Estacao,Cidades)
VALUES
('Estação LX',1,1),
('Estação das Antas',2,2),
('Estación Blanca',3,3),
('Gare Royal',4,4),
('Great Central',5,5);
```

nome	id_Estacao	Cidades
Estação LX	1	1
Estação das Antas	2	2
Estación Blanca	3	3
Gare Royal	4	4
Great Central	5	5

Figura 21 - Povoamento da tabela Estação e respetiva tabela

5.3.4 Tabela Comboio

```
INSERT INTO Comboio(Id_Comboio,Capacidade)
VALUES
(1,15),
(2,15),
(3,15),
(4,15),
(5,15);
```

Id_Comboio	Capacidade
1	15
2	15
3	15
4	15
5	15

Figura 22 - Povoamento da tabela Comboio e respetiva tabela

5.3.5 Tabela Cliente

```
INSERT INTO Cliente(Id_Cliente,Email,Nome>Password,Saldo)
VALUES
(1,'oquemimportaetersaude@gmail.com','Renato Portões','pussycat',170.50),
(2,'saralexx@gmail.com','Sara Pereira','lindinha123',87.00),
(3,'josefasjoga@live.com','Josefas Dionisia','tacaca5478',500.00),
(4,'iwishyouamerrychristmas@gmail.com','Márcio Faria','whoelse',729.43),
(5,'omelettedufromage@gmail.com','Dexter Ferreira','laboratorio',144.00),
(6,'barcasorridente@gmail.com','Raúl Cunha','doisPontosSegredo',81.00),
(7,'snoopydog@gmail.com','Rodolfo Bacelar','canil231',62.69),
(8,'gigapreto@gmail.com','Rasputin Fonseca','pauliteiro5',155.55),
(9,'galatushka@gmail.com','Shazod Yusupov','naoqueriaserokenny',47.00),
(10,'thisismyhorse@gmail.com','João Amílcar','filipa71',213.79),
(11,'unicorndinossaur45@gmail.com','João Paulo','yodameister',145.00),
(12,'voluntarinahahah@gmail.com','António Valente','nascementodouniverso',627.00)
```

Figura 23 - Povoamento da tabela Cliente

Id_Cliente	Email	Nome	Password	Saldo
1	oqueimportaetersaude@gmail.com	Renato Portões	pussycat	170.5
2	saralex@gmail.com	Sara Pereira	lindinha123	87
3	josefasjoga@live.com	Josefas Dionísia	tacaca5478	500
4	iwishyouamerrychristmas@gmail.com	Márcio Faria	whoelse	729.43
5	omelettedufromage@gmail.com	Dexter Ferreira	laboratorio	144
6	barcasorridente@gmail.com	Raúl Cunha	doispontossegredo	81
7	snoopydog@gmail.com	Rodolfo Bacelar	canil231	62.69
8	gigapreto@gmail.com	Rasputin Fonseca	pauliteiro5	155.55
9	galatushka@gmail.com	Shazod Yusupov	naoqueriaserokenny	47
10	thisismyhorse@gmail.com	João Amílcar	filipa71	213.79
11	unicorndinossaur45@gmail.com	João Paulo	yodameister	145
12	voluntarioahahah@gmail.com	António Valente	pensamentodouniverso	627
13	dijeioito@gmail.com	Carlos Ribeiro	parademeimitar	0

Figura 24 - Tabela cliente povoada

5.3.6 Tabela Lugares

INSERT INTO Lugares (N
VALUES

(1,1,1,1),
(2,1,1,1),
(3,1,1,1),
(4,1,1,1),
(5,1,1,1),
(6,2,2,1),
(7,2,2,1),
(8,2,2,1),
(9,2,2,1),
(10,2,3,1),
(11,3,2,1),
(12,3,2,1),
(13,3,2,1),
(14,3,2,1),
(15,3,3,1),
(1,1,1,2),
(2,1,1,2),
(3,1,1,2),
(4,1,1,2),
(5,1,1,2),
(6,2,2,2),
(7,2,2,2),
(8,2,2,2),
(9,2,2,2),
(10,2,3,2),
(11,3,2,2),
(12,3,2,2),
(13,3,2,2),
(14,3,2,2),
(15,3,3,2),
(1,1,1,3),
(2,1,1,3),
(3,1,1,3),
(4,1,1,3),
(5,1,1,3),
(6,2,2,3),
(7,2,2,3),
(8,2,2,3),
(9,2,2,3),
(10,2,3,3),
(11,3,2,3),
(12,3,2,3),
(13,3,2,3),
(14,3,2,3),
(15,3,3,3),
(1,1,1,4),
(2,1,1,4),
(3,1,1,4),
(4,1,1,4),
(5,1,1,4),
(6,2,2,4),
(7,2,2,4),
(8,2,2,4),
(9,2,2,4),
(10,2,3,4),
(11,3,2,4),
(12,3,2,4),
(13,3,2,4),
(14,3,2,4),
(15,3,3,4),
(1,1,1,5),
(2,1,1,5),
(3,1,1,5),
(4,1,1,5),
(5,1,1,5),
(6,2,2,5),
(7,2,2,5),
(8,2,2,5),
(9,2,2,5),
(10,2,3,5),
(11,3,2,5),
(12,3,2,5),
(13,3,2,5),
(14,3,2,5),
(15,3,3,5);

NumeroL...	Carruagem	Tipo	Comboio_Id_Comboio	8	2	2	3
1	1	1	1	8	2	2	4
1	1	1	2	8	2	2	5
1	1	1	3	9	2	2	1
1	1	1	4	9	2	2	2
1	1	1	5	9	2	2	3
2	1	1	1	9	2	2	4
2	1	1	2	9	2	2	5
2	1	1	3	10	2	3	1
2	1	1	4	10	2	3	2
2	1	1	5	10	2	3	3
3	1	1	1	10	2	3	4
3	1	1	2	10	2	3	5
3	1	1	3	11	3	2	1
3	1	1	4	11	3	2	2
3	1	1	5	11	3	2	3
4	1	1	1	11	3	2	4
4	1	1	2	11	3	2	5
4	1	1	3	12	3	2	1
4	1	1	4	12	3	2	2
4	1	1	5	12	3	2	3
5	1	1	1	12	3	2	4
5	1	1	2	12	3	2	5
5	1	1	3	13	3	2	1
5	1	1	4	13	3	2	2
5	1	1	5	13	3	2	3
6	2	2	1	13	3	2	4
6	2	2	2	13	3	2	5
6	2	2	3	14	3	2	1
6	2	2	4	14	3	2	2
6	2	2	5	14	3	2	3
7	2	2	1	14	3	2	4
7	2	2	2	14	3	2	5
7	2	2	3	15	3	3	1
7	2	2	4	15	3	3	2
7	2	2	5	15	3	3	3
8	2	2	1	15	3	3	4
8	2	2	2	15	3	3	5

Figura 25 - Povoamento da tabela Lugares e respetiva tabela

5.3.7 Tabela Percurso

```
INSERT INTO Percurso(Id_Percurso,HoraPartida,HoraChegada,Linha,precoPercurso,Comboio_Id,Estação_id_Origem,Estação_id_Destino)
VALUES
(1,'20:00','22:00',2,20,1,5,4),
(2,'11:00','14:00',2,20,2,5,4),
(3,'17:00','19:00',2,20,1,4,5),
(4,'5:00','7:00',2,20,2,4,5),
(5,'9:00','12:00',3,25,3,2,1),
(6,'17:00','20:00',3,25,3,2,1),
(7,'13:00','16:00',3,25,3,1,2),
(8,'21:00','00:00',3,25,3,1,2),
(9,'10:00','15:00',4,50,4,2,3),
(10,'22:00','02:00',4,50,4,2,3),
(11,'16:00','21:00',4,50,4,3,2),
(12,'03:00','08:00',4,50,4,3,2),
(13,'06:00','12:00',5,60,5,1,3),
(14,'13:00','19:00',5,60,5,3,1),
(15,'04:00','16:00',1,80,1,3,4),
(16,'23:00','11:00',1,80,1,4,3);
```

Id_Percurso	HoraPartida	HoraChegada	Linha	precoPercurso	Comboio_Id	Estação_id_Origem	Estação_id_Destino
1	20:00:00	22:00:00	2	20	1	5	4
2	11:00:00	14:00:00	2	20	2	5	4
3	17:00:00	19:00:00	2	20	1	4	5
4	05:00:00	07:00:00	2	20	2	4	5
5	09:00:00	12:00:00	3	25	3	2	1
6	17:00:00	20:00:00	3	25	3	2	1
7	13:00:00	16:00:00	3	25	3	1	2
8	21:00:00	00:00:00	3	25	3	1	2
9	10:00:00	15:00:00	4	50	4	2	3
10	22:00:00	02:00:00	4	50	4	2	3
11	16:00:00	21:00:00	4	50	4	3	2
12	03:00:00	08:00:00	4	50	4	3	2
13	06:00:00	12:00:00	5	60	5	1	3
14	13:00:00	19:00:00	5	60	5	3	1
15	04:00:00	16:00:00	1	80	1	3	4
16	23:00:00	11:00:00	1	80	1	4	3

Figura 26 - Povoamento da tabela Percurso e respetiva tabela

5.3.8 Tabela Bilhete

```
INSERT INTO Bilhete(Id_Bilhete,Carruagem,NumeroLugar,PrecoBilhete,Tipo,Data,Viagem,Cliente_Id_Cliente,Percurso_Id_Percurso)
VALUES
(1,1,1,37,5,1,'2016-12-15','N',1,6),
(2,1,2,37,5,1,'2016-12-15','N',3,6),
(3,2,6,25,2,'2016-12-15','N',2,6),
(4,3,15,25,3,'2016-12-15','N',4,6),
(5,1,1,37,5,1,'2016-12-15','N',2,6),
(6,2,7,25,2,'2016-12-25','N',3,6),
(7,2,6,25,2,'2016-12-25','N',1,6),
(8,2,10,25,3,'2016-12-25','N',4,6),
(9,1,1,30,1,'2016-12-10','I',5,3),
(10,2,10,20,3,'2016-12-10','I',10,3),
(11,3,8,20,2,'2016-12-10','I',12,3),
(12,2,7,20,2,'2016-12-10','I',6,3),
(13,1,2,30,1,'2016-12-11','I',1,3),
(14,2,6,20,2,'2016-12-11','I',11,3),
(15,2,5,20,2,'2016-12-11','I',13,3),
(16,3,11,20,2,'2016-12-11','I',3,3),
(17,1,1,75,1,'2016-12-9','I',1,9),
(18,1,2,75,1,'2016-12-9','I',2,9),
(19,1,3,75,1,'2016-12-9','I',3,9),
(20,1,4,75,1,'2016-12-9','I',4,9),
(21,1,5,75,1,'2016-12-9','I',11,9),
(22,2,9,50,2,'2016-12-9','I',6,9),
(23,2,10,50,3,'2016-12-9','I',7,9),
(24,3,14,50,3,'2016-12-9','I',8,9),
(25,3,15,50,3,'2016-12-9','I',8,9),
(26,2,7,20,2,'2017-01-1','I',4,1),
(27,3,11,60,2,'2017-01-1','I',12,14),
(28,1,1,120,1,'2017-01-12','I',10,16);
```


Id_Bilhete	Carruagem	NumeroLugar	PrecoBilhete	Tipo	Data	Viagem	Cliente_Id_Cliente	Percurso_Id_Percurso
1	1	1	37.5	1	2016-12-15	N	1	6
2	1	2	37.5	1	2016-12-15	N	3	6
3	2	6	25	2	2016-12-15	N	2	6
4	3	15	25	3	2016-12-15	N	4	6
5	1	1	37.5	1	2016-12-15	N	2	6
6	2	7	25	2	2016-12-25	N	3	6
7	2	6	25	2	2016-12-25	N	1	6
8	2	10	25	3	2016-12-25	N	4	6
9	1	1	30	1	2016-12-10	I	5	3
10	2	10	20	3	2016-12-10	I	10	3
11	3	8	20	2	2016-12-10	I	12	3
12	2	7	20	2	2016-12-10	I	6	3
13	1	2	30	1	2016-12-11	I	1	3
14	2	6	20	2	2016-12-11	I	11	3
15	2	5	20	2	2016-12-11	I	13	3
16	3	11	20	2	2016-12-11	I	3	3
17	1	1	75	1	2016-12-09	I	1	9
18	1	2	75	1	2016-12-09	I	2	9
19	1	3	75	1	2016-12-09	I	3	9
20	1	4	75	1	2016-12-09	I	4	9
21	1	5	75	1	2016-12-09	I	11	9
22	2	9	50	2	2016-12-09	I	6	9
23	2	10	50	3	2016-12-09	I	7	9
24	3	14	50	3	2016-12-09	I	8	9
25	3	15	50	3	2016-12-09	I	8	9
26	2	7	20	2	2017-01-01	I	4	1
27	3	11	60	2	2017-01-01	I	12	14
28	1	1	120	1	2017-01-12	I	10	16

Figura 27 - Povoamento da tabela Bilhete e respetiva tabela

5.4 Análise de transações

```

USE `leiExpress`

DELIMITER $$

CREATE PROCEDURE inserirLugar (
    IN Id_Comb    INT,
    IN Cap        INT,
    IN NrLugar    INT,
    IN Carr       INT,
    IN Tip        INT)
BEGIN
    DECLARE Erro  BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
    START TRANSACTION;

    INSERT INTO Comboio
        (Id_Comboio,Capacidade)
    VALUES
        (Id_Comb,Cap);

    INSERT INTO Lugares
        (NumeroLugar,Carruagem,Tipo,Comboio_Id_Comboio)
    VALUES
        (NrLugar, Carr, Tip, Id_Comb);

    IF erro
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$

```

Figura 28 - Transação de um novo lugar num novo comboio

Esta transação tem o objetivo de adicionar um lugar a um novo comboio, se realizarmos a transação com sucesso realizamos COMMIT dessa transação, se ocorrer algum erro vamos fazer ROLLBACK dessa transação e vamos voltar ao estado anterior da Base de Dados antes da realização dessa transação.

Ao realizar esta transação tivemos a necessidade de implementar um trigger chamado atualizaCapacidade, que na realização da transação Novo Lugar no Comboio incrementa uma unidade á capacidade total do comboio em que o lugar foi inserido.

```
DELIMITER $$

CREATE TRIGGER atualizaQuantidade
AFTER INSERT ON Lugares
FOR EACH ROW
BEGIN
    UPDATE Comboio
    SET Capacidade = Capacidade+1
    WHERE Comboio.Id_Comboio = NEW.Comboio_Id_Comboio;
END $$
```

Figura 29 - Trigger

5.5 Estimativa dos Requisitos do Espaço em Disco

O tamanho da base de dados é um fator importante num base de dados. A informação contida numa base de dados ocupa espaço físico, em memória, por isso é necessário o calculo de uma estimativa realista do tamanho ocupado pela base de dados, quer para a situação atual como para expansões futuras.

Para isso vamos começar por determinar o espaço ocupado por cada atributo calculando depois também umas previsões futuras para o seu crescimento e com as respetivas representações gráficas.

Cliente		
Atributos	Domínio	Tamanho (bytes)
Id_cliente	INTEGER	4
Password	VARCHAR	45
Email	VARCHAR	90
Nome	VARCHAR	90
Saldo	FLOAT	5

Tabela 5 - Tamanho ocupado pelos atributos da tabela Cliente

Espaço estimado (1 registo) = 4 + 45 + 90 + 90 +5 = 234 bytes

Espaço estimado (cardinalidade atual = 15) = 234 * 15 = 3510 bytes

Estimando-se um crescimento da empresa pressupõe se também um crescimento em relação ao número de clientes. Cada registo de cliente inserido ocupa 234 bytes.

Ano	Nº Clientes	Espaço Ocupado (bytes)
2016	90	21060
2017	102	23868
2018	110	25740
2019	125	29250
2020	145	33930

Tabela 6 - Estimativa de um crescimento futuro e do espaço ocupado pelo número de clientes



Figura 30 - Gráfico de espaço ocupado pelo número de Clientes

Bilhete		
Atributos	Domínio	Tamanho (bytes)
Carruagem	INTEGER	3
NumeroLugar	INTEGER	2
Viagem	CHAR	1
Data	DATE	3
preçoBilhete	FLOAT	4
Id_bilhete	INTEGER	6
Tipo	INTEGER	1

Tabela 7 - Tamanho ocupado pelos atributos da tabela Bilhete

Espaço estimado (1 registo) = 3 + 2 + 1 + 3 + 4 + 6 = 15 bytes

Espaço estimado (cardinalidade atual = 20) = 14 * 20 = 300 bytes

Estimando-se um crescimento do número de clientes pressupõe se também um crescimento em relação ao número de bilhetes. Cada registo de bilhete inserido ocupa 14 bytes.

Ano	Nº Bilhetes	Espaço Ocupado (bytes)
2016	133	1955
2017	248	3720
2018	336	5040
2019	418	6270
2020	495	7425

Tabela 8 - Estimativa de um crescimento futuro e do espaço ocupado pelo nº de bilhetes

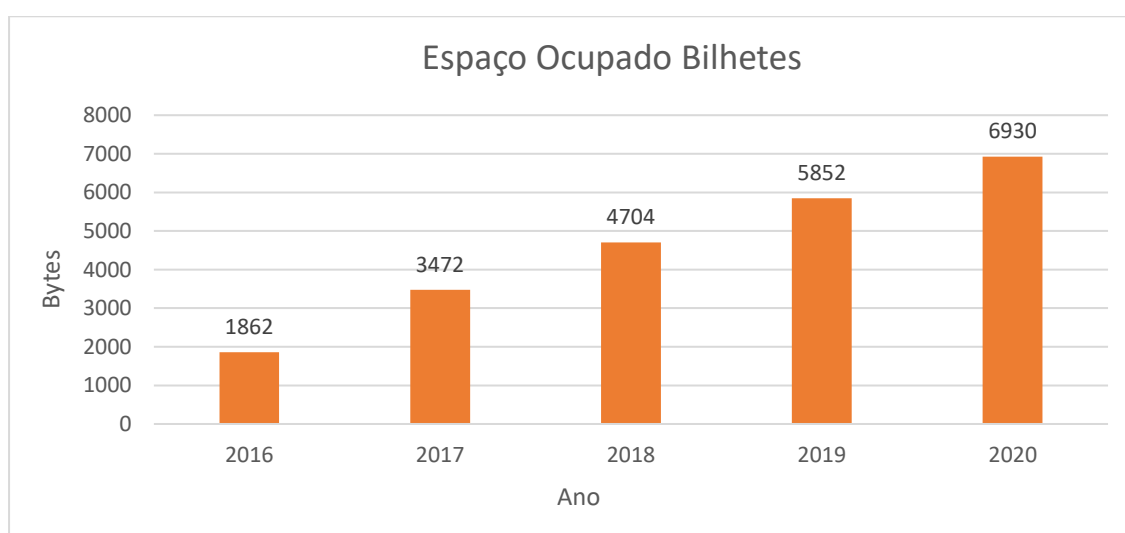


Figura 31 - Gráfico do espaço ocupado pelo nº de bilhetes

Percurso		
Atributos	Domínio	Tamanho (bytes)
preçoPercurso	FLOAT	4
Id_percurso	INTEGER	2
HoraChegada	TIME(0)	3
HoraPartida	TIME(0)	3
Linha	INTEGER	1

Tabela 9 - Tamanho ocupado pelos atributos da tabela Percurso

Espaço estimado (1 registo) = $4 + 2 + 3 + 3 + 1 = 13$ bytes

Espaço estimado (cardinalidade atual = 10) = $13 * 10 = 130$ bytes

Estimando-se um crescimento da empresa pressupõe se também um crescimento em relação ao número de percursos. Cada registo de um percurso inserido ocupa 13 bytes.

Ano	Nº Percursos	Espaço Ocupado (bytes)
2016	10	130
2017	10	130
2018	12	156
2019	13	169
2020	13	169

Tabela 10 - Estimativa de um crescimento futuro e do espaço ocupado pelo número de percursos

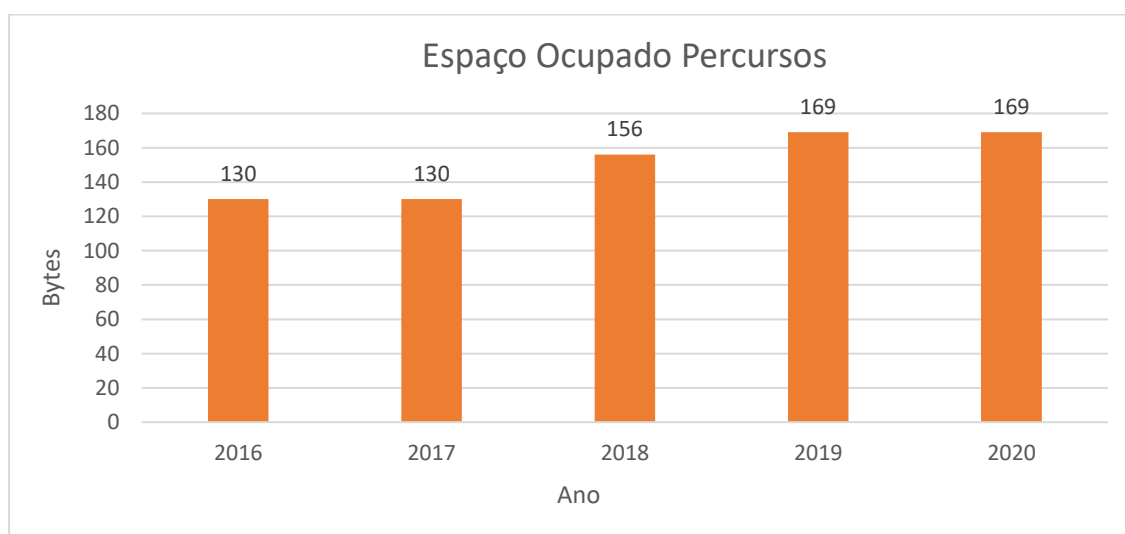


Figura 32 - Gráfico do espaço ocupado pelo número de percursos

Estação		
Atributos	Domínio	Tamanho (bytes)
País	VARCHAR	45
Cidade	VARCHAR	45
Id_Estação	INTEGER	2

Tabela 11 - Tamanho ocupado pelos atributos da tabela Estação

Espaço estimado (1 registo) = $45 + 45 + 2 = 92$ bytes

Espaço estimado (cardinalidade atual = 5) = $92 * 5 = 460$ bytes

Estimando-se um crescimento da empresa pressupõe se também um crescimento em relação ao número de estações. Cada registo de estação inserido ocupa 92 bytes.

Ano	Nº Estação	Espaço Ocupado (bytes)
2016	5	460
2017	5	460
2018	6	555
2019	6	555
2020	6	555

Tabela 12 - Estimativa de um crescimento futuro e do espaço ocupado pelo número de estações

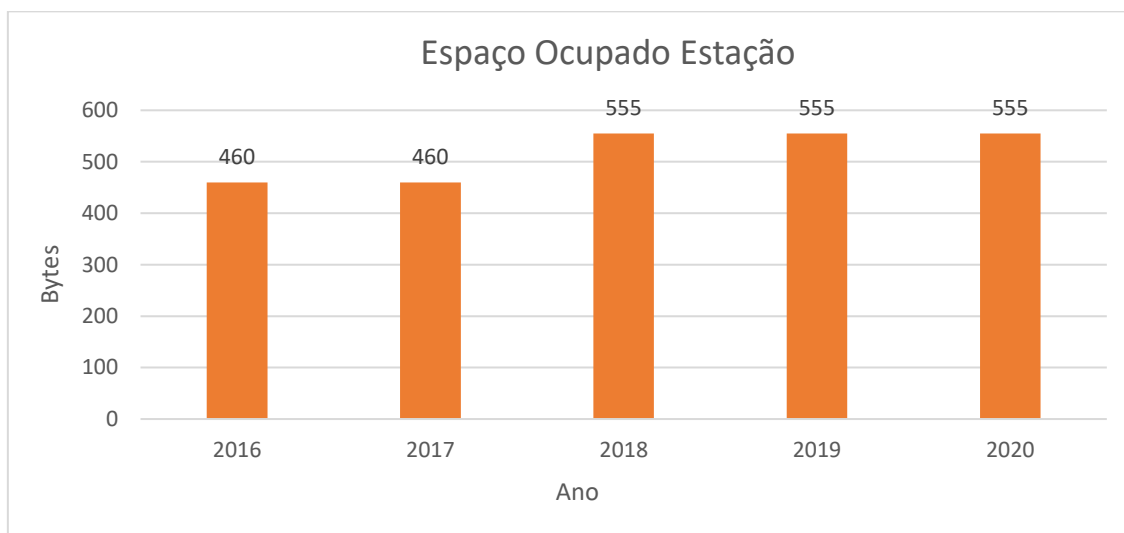


Figura 33 - Gráfico do espaço ocupado pelo número de estação

Comboio		
Atributos	Domínio	Tamanho (bytes)
Capacidade	INTEGER	2
Id_comboio	INTEGER	1
Carruagem	INTEGER	1
NúmeroLugar	INTEGER	2
Tipo	INTEGER	1

Tabela 13 - Tamanho ocupado pelos atributos da tabela Comboio

Espaço estimado (1 registo) = $2 + 1 + 1 + 2 + 1 = 7$ bytes

Espaço estimado (cardinalidade atual = 5) = $7 * 5 = 35$ bytes

Estimando-se um crescimento da empresa e do número de percursos pressupõe-se também um crescimento em relação ao número de comboios. Cada registo de um comboio inserido ocupa 7 bytes.

Ano	Nº Comboios	Espaço Ocupado (bytes)
2016	5	35
2017	5	35
2018	6	42
2019	6	42
2020	6	42

Tabela 14 - Estimativa de um crescimento futuro e do espaço ocupado pelo nº de comboios

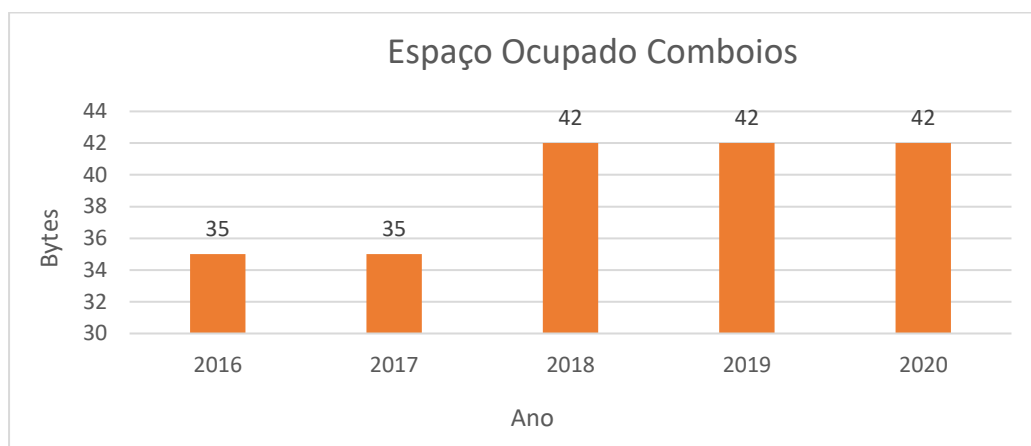


Figura 34 - Gráfico do espaço ocupado pelo nº de comboios

Atualmente necessitamos de 4415 bytes de espaço de disco para armazenar a base de dados presente.

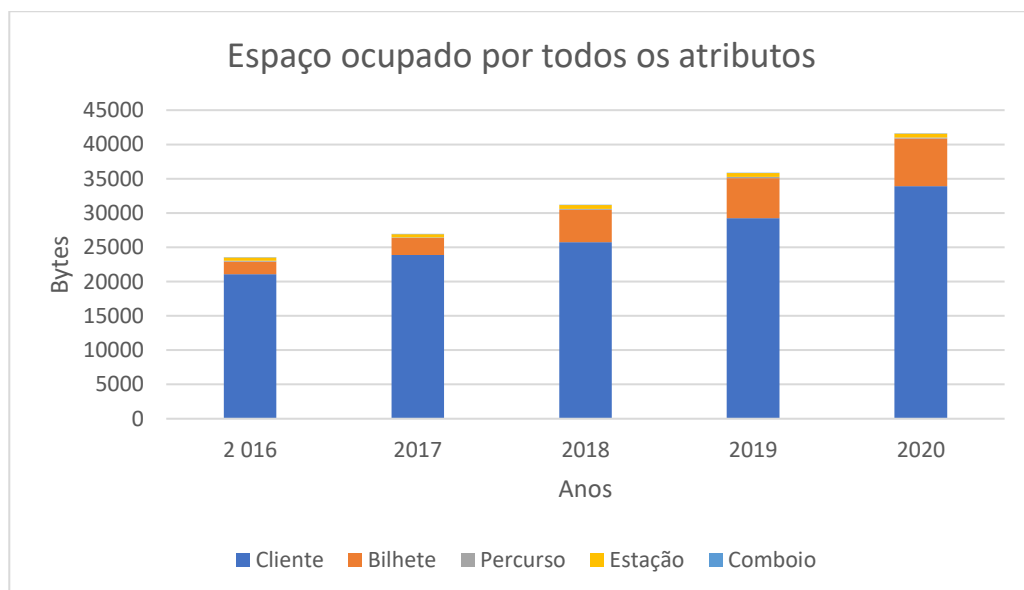


Figura 35 - Gráfico do espaço ocupado por todos os atributos

5.6 Definição das Vistas dos Utilizadores

Uma Vista de Utilizador é um mecanismo presente na linguagem SQL, que obtém o resultado de um comando de pesquisa pré-definido. Uma Vista não vai fazer parte do esquema físico da Base de Dados, mas no entanto vai criar uma tabela virtual com toda a informação da pesquisa que foi realizada. Sempre que desejarmos aceder a uma Vista esta vai ser atualizada em cada invocação consoante a atualização correspondente da informação na Base de Dados.

As vistas oferecem algumas vantagens em relação às consultas diretas às tabelas na Base de Dados, pois esta vai mostrar apenas um conjunto que é previamente definido de informação presente em uma ou mais tabelas. Elas por vezes são utilizadas de forma a limitarem o acesso a informação sem termos que definir regras de acesso para esse tal efeito. Uma vista é capaz de agregar diversas colunas de várias tabelas, podendo fazer cálculos sobre várias colunas podendo-as apresentar juntamente com os resultados sob a forma de uma única tabela, que vai possuir um nome da Vista associado.

Nas figuras que se seguem vamos identificar alguns exemplos de Vista de Utilizadores no nosso Modelo Físico da Base de Dados.

View (1) – Vista que mostra o Nome do Cliente ordenado associado ao Identificador do Cliente.

```
CREATE VIEW VistaDosClientes AS
SELECT Id_Cliente, Nome FROM Cliente
ORDER BY Nome
```

Figura 36 - View 1

View (2) – Vista que mostra todos os Bilhetes possuídos por todos os clientes.

```
CREATE VIEW VistaDosBilhetesDosClientes AS
SELECT Id_Bilhete, Carruagem, NumeroLugar, PrecoBilhete, Data, Viagem, Cliente_Id_Cliente FROM Bilhete
ORDER BY Cliente_Id_Cliente ASC
```

Figura 37 - View 2

View (3) – Vista que mostra os Percursos conjuntamente com os identificadores da Estação destino e da Origem, ordenados ascendentemente pela Hora de Partida.

```
CREATE VIEW VistaPercursos AS
SELECT Id_Percurso AS 'Identificador do Percurso',
       HoraPartida AS 'Hora de Partida',
       HoraChegada AS 'Hora de Chegada',
       Linha AS 'Linha',
       PrecoPercurso AS 'Preço do Percurso',
       Comboio_ID AS 'Identificador do Comboio',
       Estacao_id_Origem AS 'Estação Origem',
       Estacao_id_Destino AS 'Estação Destino'
FROM PERCURSO
INNER JOIN Estacao AS EO on EO.Id_Estacao = Estacao_id_Origem
INNER JOIN Estacao AS EP on EP.Id_Estacao = Estacao_id_Destino
ORDER BY HoraPartida ASC
```

Figura 38 - View 3

View (4) – Vista que seleciona os três percursos mais caros e respetiva informação ordenados pelo preço do percurso decendentemente.

```
CREATE VIEW Top3PercursosMaisCaros AS
SELECT Id_Percurso AS 'Identificador do Percurso',
       PrecoPercurso AS 'Preço do Percurso',
       EO.nome AS 'Estação de Origem',
       EP.nome AS 'Estação de Destino',
       Estacao_id_Origem AS 'Identificador da Estação Origem',
       Estacao_id_Destino AS 'Identificador da Estação Destino'
FROM Percurso
INNER JOIN Estacao AS EO on EO.Id_Estacao = Estacao_id_Origem
INNER JOIN Estacao AS EP on EP.Id_Estacao = Estacao_id_Destino
ORDER BY PrecoPercurso DESC
LIMIT 3;
```

Figura 39 - View 4

View (5) – Vista que mostra os todos os bilhetes e respetiva informação, em que a viagem fosse nacional e que fossem adquiridos durante o ano de 2016.

```
CREATE VIEW MostrarBilhetesNacionaisAntesDoFim2016 AS
SELECT Id_Bilhete AS 'Identificador do Bilhete',
       Viagem AS 'Tipo Viagem',
       Data AS 'Data da Viagem',
       Cliente_Id_Cliente AS 'Identificador do Cliente'
FROM Bilhete AS B
INNER JOIN Cliente AS C on C.Id_Cliente = B.Cliente_Id_Cliente
WHERE (Viagem = 'N') AND B.Data > DATE_SUB('2016-12-31', INTERVAL 1 YEAR)
ORDER BY B.Data DESC;
```

Figura 40 - View 5

5.7 Definir as Regras de acesso

Um dos pontos fulcrais no desenvolvimento de um Sistema de Gestão de Base de Dados é a restrição das vistas de cada utilizador e a implementação das suas regras de acesso. Assumimos que a nossa base de dados tem os seguintes tipos de utilizadores:

Administrador – Tem acesso a toda a Base de Dados e pode executar qualquer ação que quiser.

```
CREATE USER 'admin'@'localhost'  
    identified by 'adminpassword';  
  
GRANT ALL ON *.* TO 'admin'@'localhost';
```

Figura 41 - Administrador

Gestor da empresa – Pode visualizar toda a base de dados. Não pode criar e eliminar tabelas. Pode inserir e atualizar os bilhetes e os clientes.

```
CREATE USER 'gestor'@'localhost'  
    identified by 'gestorpassword';  
  
REVOKE DROP, CREATE, DELETE  
    ON *.*  
    FROM 'gestor'@'localhost';
```

Figura 42 - Gestor de empresa

Cliente – Tem apenas acesso a visualizar as viagens, isto é, os percursos disponíveis e toda a informação relacionada com isso (horários, preços, origem e destino...)

```
CREATE USER 'cliente'@'localhost'
    identified BY 'clientepassword';

GRANT SELECT ON leiExpress.Percurso TO 'cliente'@'localhost';

GRANT SELECT ON leiExpress.Bilhete TO 'cliente'@'localhost';

REVOKE DROP, CREATE, DELETE, UPDATE, INSERT
ON *.*
FROM 'cliente'@'localhost';
```

Figura 43 - Cliente

5.8 Controlo de Concorrência

O controlo de Concorrência é um método usado para garantir que as transações sejam executadas e sigam as regras ACID. Estas regras servem para evitar as seguintes anomalias: acesso a dados inconsistentes, perdas de atualizações e de consistência. Os SGBDs devem ser capazes de garantir que nenhuma ação de transação completadas com sucesso seja perdida. Uma transação é uma unidade que preserva a consistência. Para possuímos controlo de concorrência na nossa base de dados precisamos de a escalonar e fazer bloqueios para que sejam evitados erros de atualizações perdidas e de leitura suja (ou seja, ler um valor que não era suposto ter lido ou leu-o de maneira errada). Estes erros ocorrem devido ao facto de serem efetuadas muitas operações ao mesmo tempo na base de dados. Para solucionarmos esse problema consideramos um método de controlo que tem de garantir que em todas as execuções concorrentes das nossas transações tem que esperar que cada transação termine e cada transação tem de ser executada sem interferência das outras, e sem que anomalias de sincronização ocorram.

Podemos concluir que para possuímos um acesso concorrente á base de dados e mantermos a consistência da mesma, temos de fazer uso da nossa solução enunciada a cima.

5.9 Apresentação de Queries

- 1) Mostrar clientes e respetivos saldos na base de dados com saldo superior a uma certa quantidade, ordenados por ordem decrescente de saldo.

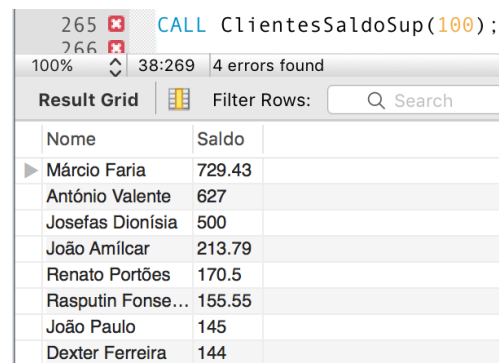
```

DELIMITER $$
CREATE PROCEDURE ClientesSaldoSup
    (IN Quantidade FLOAT)
BEGIN
    SELECT Nome, Saldo from Cliente
    WHERE Saldo > Quantidade
    ORDER BY Saldo DESC;
END $$

```

Figura 44 - Querie 1

Responde-se a esta *querie* criando um *procedure* que recebe como *input* uma quantidade e selecciona o nome e saldo da tabela Cliente e devolve aqueles que tem saldo superior à quantidade dada, ordenando-a por ordem decrescente de saldo.



265 266 CALL ClientesSaldoSup(100);

100% 38:269 4 errors found

Result Grid Filter Rows: Search

Nome	Saldo
▶ Márcio Faria	729.43
António Valente	627
Josefas Dionísia	500
João Amílcar	213.79
Renato Portões	170.5
Rasputin Fonse...	155.55
João Paulo	145
Dexter Ferreira	144

Figura 45 - Resultado da querie 1

2) Quais os 3 clientes com mais bilhetes comprados e quantos bilhetes são?

```

SELECT Nome, Count(*) num FROM Cliente c
INNER JOIN Bilhete b ON c.Id_Cliente=b.Cliente_Id_Cliente
GROUP BY Nome
ORDER BY num DESC
LIMIT 3;

```

Figura 46 - Querie 2

Selecciona-se o nome da tabela Cliente e fazendo uma junção com a tabela do Bilhete, juntando os nomes e ordenando pela contagem de ordem decrescente, com limite de 3, chega-se à resposta.

Nome	num
Renato Portões	4
Josefas Dionísia	4
Márcio Faria	4

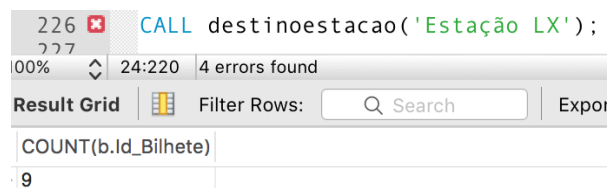
Figura 47 - Resultado querie 2

3) Quantas bilhetes é que tem destino a uma certa estação?

```
DELIMITER $$
CREATE PROCEDURE destinoestacao
  (IN Nomeestacao VARCHAR(45))
BEGIN
  SELECT COUNT(b.Id_Bilhete) FROM Bilhete b
  INNER JOIN Percurso p ON p.Id_Percurso=b.Percurso_Id_Percurso
  INNER JOIN Estacao e ON e.id_Estacao=p.Estacao_id_Destino
  WHERE e.nome=Nomeestacao;
END$$
```

Figura 48 - Querie 3

Cria-se um *procedure* que recebe o nome da estação. Há uma contagem dos bilhetes e 2 *inner join* uma com a tabela Percurso e outra com a Estação, onde se verifica qual dos nomes da tabela Estação é igual à estação dada.



The screenshot shows a database client window with the following content:

- Execution bar: 226, 777, CALL destinoestacao('Estação LX');
- Status bar: 00%, 24:220, 4 errors found
- Buttons: Result Grid, Filter Rows, Search, Export
- Result Grid:

COUNT(b.Id_Bilhete)
9

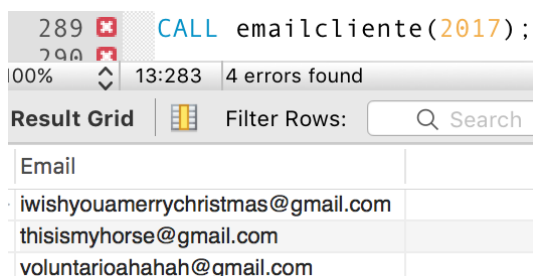
Figura 49 - Resultado querie 3

4) Quais os emails dos clientes que tem bilhetes comprados num certo ano?

```
DELIMITER $$
CREATE PROCEDURE emailcliente
  (IN Ano Int)
BEGIN
  SELECT c.Email from Cliente c
  INNER JOIN Bilhete b ON c.Id_Cliente=b.Cliente_Id_Cliente
  WHERE year(b.data) = Ano
  GROUP BY c.Email;
END $$
```

Figura 50 - Querie 4

Cria-se um *procedure* que recebe o ano pretendido. Selecciona-se os *emails* através da junção de tabelas de Cliente e Bilhete onde o ano é o *input* fornecido, e agrupa-se por *email*.



The screenshot shows a database interface with a command bar containing the text: `CALL emailcliente(2017);`. Below the command bar, a status bar indicates '100%' completion, a duration of '13:283', and '4 errors found'. A 'Result Grid' tab is active, displaying a table with the following data:

Email
iwishyouamerrychristmas@gmail.com
thisismyhorse@gmail.com
voluntarioahahah@gmail.com

Figura 51 - Resultado querie 4

5) Qual o tipo de bilhete menos comprado?

```
SELECT Tipo, Count(*) num FROM Bilhete b
GROUP BY Tipo
ORDER BY num ASC
LIMIT 1;
```

Figura 52 - Querie 5

Seleccionando o tipo e sua contagem da tabela Bilhete, agrupando pelo tipo e ordenado pela contagem por ordem crescente, impondo um limite de 1, obtém-se o tipo menos comprado.

Tipo	num
3	6

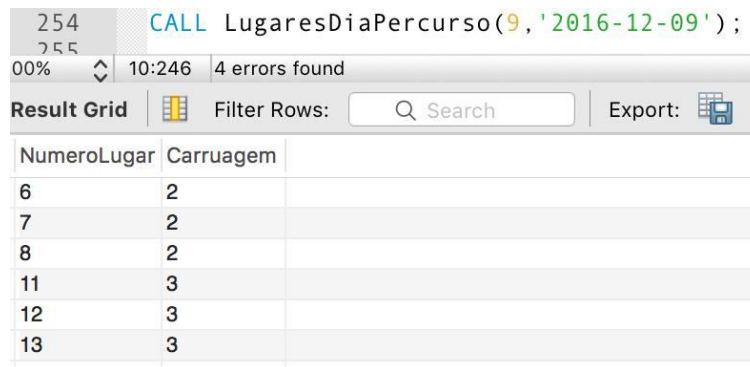
Figura 53 - Resultado querie 5

6) Lugares e respetiva carruagem ocupados num certo dia num certo percurso.

```
DELIMITER $$
CREATE PROCEDURE LugaresDiaPercurso
(IN Viagem INT, IN Dia DATE)
BEGIN
SELECT Lugares.NumeroLugar, Lugares.Carruagem FROM Percurso
INNER JOIN Lugares
ON Lugares.Comboio_Id_Comboio = Percurso.Comboio_Id
INNER JOIN Comboio
ON Comboio.Id_Comboio = Lugares.Comboio_Id_Comboio
WHERE Percurso.Id_Percurso = Viagem
AND Lugares.NumeroLugar NOT IN (
SELECT b.NumeroLugar FROM Bilhete b
WHERE b.data = Dia and b.Percurso_Id_Percurso = Viagem);
END$$
```

Figura 54 - Querie 6

Criando-se um procedimento que recebe uma certa viagem e uma data, e fazendo uma junção das tabelas Percurso, Lugares e Comboio onde o id do percurso é igual à viagem fornecida e os lugares não pertencem aos bilhetes reservados na determinada data e viagem.



254 CALL LugaresDiaPercurso(9, '2016-12-09');
255
00% 10:246 4 errors found

Result Grid Filter Rows: Search Export:

NumeroLugar	Carruagem
6	2
7	2
8	2
11	3
12	3
13	3

Figura 55 - Resposta querie 6

5. Conclusões e Trabalho Futuro

Ao implementarmos a base de dados proposta fizemos uma abordagem com diversos cuidados para resolvermos o problema em questão, onde foram atingidos todos os principais requisitos. Acreditamos que esta solução poderá vir a ser uma ferramenta útil na gestão de reservas de bilhetes para os diferentes tipos de viagens que uma companhia ferroviária pode oferecer aos seus clientes.

Durante a realização deste projeto foi-nos possível consolidar os conteúdos e competências relacionados com o processo de criação e desenvolvimento de um SGBD. Numa fase inicial deste projeto foi proposto a implementação de um sistema de Base de Dados para reserva de bilhetes para viagens nacionais e internacionais. Seguidamente, foi necessário realizar uma análise e levantamento de requisitos necessários para a implementação desta.

Após o levantamento de requisitos foi possível identificar que a base de dados necessitava de entidades fundamentais, tal como Bilhete, sendo esta a principal pois o objetivo deste projeto é a reserva de bilhetes. Partindo desta entidade outras se desenvolveram por consequência, tais como, Comboio, Percurso, Estação e Cliente. Sentimos alguma dificuldade na colocação dos atributos, pois muitas das vezes ficávamos em dúvida acerca de qual a entidade mais acertada para possuir um determinado atributo.

Após a estruturação do nosso projeto, avançamos para a construção do modelo concetual que nos dá uma pré visualização sobre o funcionamento de um sistema de reserva de bilhetes. Seguidamente, na passagem para o modelo lógico desenvolveu-se a transformação do modelo conceptual para modelo lógico. Foi utilizada como ferramenta o MySQL Workbench, procedendo de seguida ao processo de normalização até à 3ª Forma Normal e várias regras de integridade assegurando que a base de dados se encontrava consistente e sem redundância.

Por último, implementamos o modelo físico passando assim à fase de criar relações base e o respetivo povoamento. Nesta fase foi realizado o cálculo da estimativa de espaço de disco que iria ser necessária, incluindo projeções futuras ilustrada por gráficos temporais.

Depois de passarmos por estes processos envolventes da criação de uma base de dados de raiz, desde a análise de requisitos até à criação do respetivo modelo físico, conseguimos desenvolver diversas capacidades na elaboração de base de dados. O processo de análise de requisitos permitiu-nos desenvolver a nossa capacidade de abstração no sentido de encontrar-mos a melhor solução do problema. Ao elaborarmos

o modelo concetual permitiu-nos alinhar a nossa solução para que ele cumprisse todos os requisitos pedidos desde a definição das entidades, passando pelo seus atributos até ao relacionamento entre eles. Com a elaboração do modelo lógico aprofundamos todos os conceitos do modelo anterior e aproximamo-nos de um modelo de programação orientada a SGBD. Por fim, ao realizarmos a conversão para o modelo físico obtivemos um maior conhecimento da linguagem SQL.

Na análise efetuada ao nosso trabalho chegamos á conclusão que podíamos melhorar alguns aspetos, tais como: o facto de o preço ser calculado ao nível da nossa base de dados e não ao nível da aplicação, para facilitar as inserções nas tabelas.

Em suma, podemos concluir que a modelação de base de dados é uma tecnologia importante no tratamento de informação, sendo o modelo apresentado no relatório um pequeno exemplo de uma situação real.

Referências

[01] Thomas Connolly, Carolyn Begg 2005. DATABASE SYSTEMS A Practical Approach to Design, Implementation, and Management, Harlow, Addison-Wesley.

[2] Feliz Gouveia, 2014. Fundamentos de Bases de Dados.

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
FK	<i>Foreign Key</i>
DW	Data Warehouse
OLTP	<i>On-Line Analytical Processing</i>

Anexos

I. Anexo 1

Criou-se tabelas para as cidades e para os países de forma a melhor gerir as estações nestes locais. Ou seja, se quisermos ver as estações que a empresa tem em funcionamento em determinado país ou mesmo numa cidade, torna-se assim muito mais fácil.

Outra das razões é a de evitar redundâncias no sistema. Imaginemos que a empresa tinha uma estação em Vila Nova de Famalicão, a referência a V. N. de Famalicão, Vila Nova de Famalicão ou simplesmente Famalicão seria exatamente à mesma cidade mas o nosso sistema iria considerar como cidades diferentes. Ao criarmos tabelas para as cidades evitamos esta situação.

Relacionamento 1:1 Estação-Cidade:

- Apenas existe uma estação por cidade. Caso o cenário se altere é preciso atualizar a base de dados de forma a alterar o relacionamento de 1:1 para n:1.

Relacionamento n:1 Cidade-País

- Como várias cidades diferentes podem fazer parte do mesmo país, a relação da cidade-país é de n:1.

II. Anexo 2

Entidade	Atributos	Descrição	Domínio	NULL	MV	Tamanho
Cliente	Password	Password do cliente	VARCHAR	Não	Não	45
	Email	Email do cliente	VARCHAR	Não	Não	90
	Nome	Nome do cliente	VARCHAR	Não	Não	90
	Saldo	Saldo da conta do cliente	FLOAT	Não	Não	5
	Id_cliente (PK)	Número identificador do cliente	INTEGER	Não	Não	3
Bilhete	Carruagem	Número da carruagem	INTEGER	Não	Não	3
	NumeroLugar	Número do lugar do comboio	INTEGER	Não	Não	2
	Tipo	Tipo do lugar	INTEGER	Não	Não	1
	Viagem	Tipo da viagem	CHAR	Não	Não	1
	Data	Data da viagem	DATE	Não	Não	3

	preçoBilhete	Preço do bilhete	FLOAT	Não	Não	5
	Id_bilhete (PK)	Número identificador do bilhete	INTEGER	Não	Não	6
Percurso	preçoPercurso	Preço do percurso	FLOAT	Não	Não	5
	Id_percurso (PK)	Número identificador do percurso	INTEGER	Não	Não	2
	horaChegada	Hora prevista da chegada do comboio	DATE	Não	Não	
	HoraPartida	Hora prevista da partida do comboio	DATE	Não	Não	
	Linha	Linha do percurso	INTEGER	Não	Não	1
Estação	País	País onde se encontra localizada a estação	VARCHAR	Não	Não	45
	Cidade	Cidade onde se encontra localizada a estação	VARCHAR	Não	Não	45
	Id_Estação (PK)	Número identificador da estação	INT	Não	Não	2
Comboio	Capacidade	Capacidade do comboio	INT	Não	Não	2
	Id_Comboio (PK)	Número identificador do comboio	INT	Não	Não	1
	Carruagem	Carruagem do comboio	INT	Não	Sim	1
	númeroLugar	Número do lugar do comboio	INT	Não	Sim	2
	Tipo	Tipo do lugar	INT	Não	Sim	1