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

Unidade Curricular de Bases de Dados

Ano Letivo de 2016/2017

Comboio Académico

Bruno Rafael Lamas Corredoura Dantas, A74207

Daniel Camelo Rodrigues, A75655

Hugo Alves Carvalho, A74219

Luís Miguel da Cunha Lima, A74260

Novembro, 2016

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

Comboio Académico

Bruno Rafael Lamas Corredoura Dantas, A74207
Daniel Camelo Rodrigues, A75655
Hugo Alves Carvalho, A74219
Luís Miguel da Cunha Lima, A74260

Novembro, 2016

Resumo

O projeto apresentado neste relatório consiste na implementação de um sistema de base de dados (SBD) referente ao processo de reservas de viagens em comboios nacionais e internacionais.

Numa primeira fase começamos por analisar melhor o problema para compreendermos o que estaria em causa.

De seguida, levantamos os requisitos fundamentais para o sistema, identificando assim as principais entidades, os seus atributos e os respetivos relacionamentos entre estas.

Após a realização do modelo conceptual, passamos à realização do modelo lógico, representando as tabelas correspondentes a cada entidade. O esquema foi validado através da normalização e pelas transações com o utilizador.

Seguidamente passamos à realização e implementação do modelo físico, através do SGBD MySQL. Apresentamos o desenho da representação física e analisamos as principais transações existentes.

Após esta análise, realizamos uma estimativa do espaço necessário em disco para a fase inicial da base de dados. Após concluída esta etapa, efetuamos o povoamento da BD.

Por último, definimos as restrições e as vistas dos utilizadores, bem como as regras de acesso à BD.

Área de Aplicação: Desenho e arquitetura de um Sistema de Base de Dados para reservas em comboios nacionais e internacionais

Palavras-Chave: Estudante, Funcionário, Bilhete, Viagem, Comboio, Reserva, Lugar, Entidade, Atributo, Relacionamento, Base de dados.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Existe um número considerável de estudantes universitários que para se deslocar até à cidade onde estudam, recorrem ao uso de transportes públicos, seja pelo facto de não ter uma viatura própria ou até mesmo razões económicas. Alguns destes alunos realizam as suas viagens apenas no início e no final da semana, uma vez que têm residência na cidade onde estudam. Por outro lado, existe igualmente um elevado número de estudantes que não tem esta possibilidade e como tal realizam diariamente estas viagens através de transportes públicos.

No entanto, existem vários casos em que os horários destes transportes não são compatíveis com o horário escolar dos alunos, fazendo com que estes tenham que esperar várias horas e cheguem tarde às suas casas. Por outro lado, o facto destes transportes serem frequentados não só por estudantes, por exemplo trabalhadores no início e fim do seu horário de trabalho, pode levar a situações de indisponibilidade de bilhetes, por lotação completa, estando o estudante impedido de reservar a sua viagem. Considerando estes problemas que afetam o dia-a-dia dos estudantes universitários, surgiu o “Comboio Académico”.

A empresa “Comboio Académico” dedica-se ao transporte exclusivo de estudantes universitários entre algumas das principais cidades universitárias das regiões Norte de Portugal e Espanha. Estas viagens têm um horário adaptado aos estudantes, uma vez que se realizam entre as 7h e 20h. Em todas estas estações ferroviárias existem comboios que operam em ambas as direções (exceto na Corunha e no Porto, uma vez que são as cidades iniciais e finais do trajeto). Para além desta adaptação aos horários escolares, a empresa vende as suas viagens a um preço mais acessível do que os habituais transportes públicos.

1.2. Apresentação do Caso de Estudo

O “Comboio Académico” dedica-se ao transporte exclusivo de estudantes universitários através de comboios intercidades.

Para isso, a empresa tem em cada uma das estações ferroviárias onde os seus comboios efetuam paragens, um funcionário a quem os estudantes se devem dirigir para efetuarem as suas reservas. Estes precisam de apresentar o seu cartão de cidadão e cartão de

estudante para que possa ser validada a sua identificação. Em seguida, o funcionário mostra a lista de lugares disponíveis para a viagem e o estudante deve escolher o lugar onde pretende viajar.

Os estudantes podem reservar o número de viagens que quiserem, no entanto não podem comprar mais do que um bilhete para cada viagem no mesmo dia. O preço de cada viagem mantém-se igual para cada viagem diária, variando apenas consoante o local de partida e o local de destino.

Após efetuar o pagamento, o estudante recebe um bilhete por cada viagem que reservou.

Depois de efetuar a reserva, o estudante apresenta o seu bilhete juntamente com o cartão de estudante num sensor presente na entrada do comboio, validando a sua reserva e sendo assim autorizado a viajar.

1.3. Motivação

Ao existir um comboio destinado exclusivamente aos estudantes, que diariamente realiza diferentes percursos interligando várias estações ferroviárias de cidades universitárias, faz com que haja um elevado número de reservas de viagens de comboio disponíveis, realizadas por estes clientes (estudantes).

Neste sentido, surge a necessidade de guardar numa base de dados a informação relativa aos comboios da empresa e às viagens que estes efetuam, bem como sobre os seus utilizadores e funcionários. Para além disso, é fundamental guardar também a informação de todas as reservas efetuadas.

Outra das motivações para a realização deste projeto é também de extrema importância e prende-se com a necessidade de organizar toda esta informação de forma a existir uma consulta e análise dos dados mais eficientes.

1.4. Objetivos

O objetivo inicial passa por implementar um SBD forte que possibilite um ambiente apropriado e eficiente, capaz de armazenar toda a informação pertencente aos diferentes estudantes, comboios e também das suas reservas.

É importante construir uma base de dados compreensível, ou seja, que exista uma estrutura clara que leve à fácil, flexível e rápida consulta e atualização dos dados.

A solução implementada deve ser segura, garantindo a integridade, disponibilidade e a confidencialidade da informação.

Com a implementação deste SBD estaremos capazes de obedecer a todos os objetivos aqui mencionados.

1.5. Justificação da Viabilidade do Projeto

Para que o SBD cumpra os requisitos referidos nos pontos 1.3 e 1.4, decidimos optar pelo Sistema de Base de Dados Relacional (SBDR). A escolha baseia-se num grande número de vantagens, das quais se destacam:

Simplicidade - O SBDR estrutura os dados de forma a evitar a complexidade. A estrutura das tabelas é organizada de forma intuitiva. Os dados são organizados naturalmente dentro do modelo, simplificando o desenvolvimento e uso do SBDR.

Facilidade na apresentação dos dados - Outra vantagem é a possibilidade de consultar qualquer tabela no SBDR, combina-la com outra e usar funções de junção para incluir dados relevantes (por exemplo, contidos noutras tabelas) nos resultados. Os resultados podem ser filtrados com base no conteúdo de qualquer coluna e em qualquer número de colunas, permitindo apresentar facilmente resultados significativos. Além disso, podem-se escolher as colunas a incluir nos resultados, de modo a serem exibidos apenas os dados relevantes.

Integridade dos dados - A integridade dos dados é uma característica essencial do SBDR. O tipo dos dados e a sua verificação garantem que os dados sejam aceitáveis. A integridade referencial entre tabelas (chaves estrangeiras) impede que os registos se tornem incompletos e ajuda a garantir a precisão e a consistência dos dados.

Flexibilidade - O SBDR é extensível e fornece uma estrutura flexível para atender às mudanças de requisitos e quantidades crescentes de dados. Permite que alterações numa estrutura de base de dados sejam implementadas facilmente sem afetar os dados ou as restantes bases de dados. É possível adicionar, remover e modificar tabelas e colunas.

Normalização - A normalização garante que o projeto do SBDR esteja livre de anomalias/redundâncias que possam afetar a integridade e a precisão dos dados. A normalização fornece-nos a confiança que o SBDR é robusto e confiável.

1.6. Estrutura do Relatório

Numa parte inicial do relatório vão ser explicados todos os detalhes relacionados com a análise de requisitos que permitiu ter uma melhor compreensão do problema, a fim de realizar o projeto pretendido.

Nos capítulos seguintes, procede-se à apresentação dos modelos conceptual, lógico e físico do SBD, bem como as suas respetivas validações.

Por último é também apresentado um povoamento inicial para o SBD, bem como as respetivas restrições do sistema e diferentes vistas de utilizadores.

2. Análise de Requisitos

A análise de requisitos, embora seja difícil de concretizar com precisão é uma das fases mais importantes para a implementação do SBD. Para obtermos um bom levantamento de requisitos e assim ter uma melhor perceção do problema em questão, recolhemos durante vários dias informações sobre o processo de reservas de viagens nesta empresa.

Ao analisarmos a informação recolhida, consideramos como fundamentais os seguintes requisitos:

- O principal interveniente do sistema é o estudante universitário, uma vez que é este quem faz as reservas e realiza as viagens. A informação que melhor identifica cada estudante é o seu nome, género, nº de cartão de cidadão e o nome da universidade onde estuda.
- A viagem é também uma das principais entidades do sistema. Cada viagem é realizada todos os dias e por isso é caracterizada pelo seu id, preço que a viagem custa no momento atual, local de partida, local de chegada, hora de partida e hora de chegada. É importante referir que a hora de uma viagem nunca é alterada, ou seja, se a hora de uma viagem com o mesmo local de partida e local de chegada alterar, é considerada como uma nova viagem.
- É fundamental que todos os comboios da empresa também estejam identificados, uma vez que são estes quem realizam as viagens. Cada comboio tem um número associado (id), e é também caracterizado pelos seus lugares, uma vez que existem comboios com diferente capacidade. Por sua vez, estes lugares são identificados pelo seu id, número de banco e número de carruagem.
- O funcionário é quem regista as reservas das viagens, logo também precisa de estar identificado no sistema. É caracterizado pelo seu nome e número.
- O bilhete é também uma entidade essencial para o sistema, uma vez que é obrigatória a sua apresentação para que o estudante possa realizar as suas viagens. Assim, deve ser identificado pelo seu id, pela data em que o bilhete é emitido e pela data da viagem. Por outro lado, este precisa de ser identificado pelo lugar que o estudante escolheu no momento da reserva. O preço de uma viagem pode ser alterado a qualquer momento, por isso é também importante considerar no bilhete o preço que a viagem custou no momento da reserva.

3. Modelo Conceptual

3.1. Identificar os tipos de entidades

De forma a conseguirmos identificar as diferentes entidades que o sistema engloba, foi necessário determinar quais os objetos que se enquadram nesta definição. Para isso, após a leitura dos requisitos podemos chegar à conclusão que as entidades fundamentais são: o estudante, a viagem, o comboio, o funcionário e o bilhete.

De seguida é apresentada uma tabela com a descrição de cada uma destas entidades, bem como os seus sinónimos e as suas ocorrências.

Nome da Entidade	Descrição	Sinónimos	Ocorrências
Estudante	Cliente que pretende efetuar uma reserva para uma viagem de comboio.	Cliente, aluno.	Pode reservar o número de viagens que pretender.
Viagem	Percurso feito por um comboio entre duas cidades.	Percurso, trajeto	A mesma viagem pode ser reservada por vários estudantes.
Comboio	Meio de transporte utilizado pela empresa nas suas viagens.	Meio de transporte.	Um comboio realiza várias viagens.
Bilhete	Comprovativo que a reserva foi concluída.	Passe, comprovativo, ingresso.	É utilizado, pelo estudante, na entrada do comboio, de modo a autorizar a sua viagem.
Funcionário	Empregado que atende o cliente.	Empregado, trabalhador.	Regista várias reservas, emitindo, para cada uma, o bilhete correspondente.

Tabela 1 – Tabela das Entidades

3.2. Identificar os tipos de relacionamentos

Após identificadas as entidades do nosso SBD é necessário detetar todos os relacionamentos existentes entre as mesmas. A leitura da análise de requisitos permite identificar estes relacionamentos que as entidades estabelecem entre sim, bem como a sua respetiva cardinalidade.

Desta forma, apresentamos na seguinte tabela todos os relacionamentos entre entidades que conseguimos identificar.

Nome da Entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da Entidade
Estudante	1	Compra	N	Bilhete
Funcionário	1	Regista	N	Bilhete
Bilhete	N	Associado	1	Viagem
Comboio	1	Realiza	N	Viagem

Tabela 2 – Tabela dos Relacionamentos

3.3. Identificar e associar atributos com os tipos de entidades

De seguida, será abordada a associação entre a informação que conseguimos reter sobre as diferentes entidades. Esta informação foi recolhida de acordo com a sua importância para caracterizar essa entidade e também com a relevância que esta pode ter na BD. Serão em seguida apresentados os atributos pertencentes a cada entidade.

3.3.1. Atributos simples/compostos

Depois de levantados os requisitos e identificados os atributos que iriam estar presentes no nosso modelo conceptual, chegou-se à conclusão que o lugar é um atributo composto, uma vez que é caracterizado pelo seu id, número da cadeira e carruagem.

3.3.2. Atributos derivados

Após o levantamento de requisitos e identificação dos atributos, chegou-se à conclusão que não existem atributos derivados no nosso sistema.

3.3.3. Atributos multivalor

Da mesma análise referida nos pontos anteriores, é possível concluir que o atributo lugar, já anteriormente referido como atributo composto, é também um atributo multivalor, uma vez que existem vários lugares para cada comboio.

3.3.4. Associação entre atributos e entidades

De seguida, é apresentada uma tabela com informação sobre a associação dos atributos com as suas respetivas entidades. Nesta tabela, está presente uma breve descrição sobre cada atributo, o seu tipo e tamanho, a sua nulidade, se é multivalor, se é derivado e se é composto.

Nome da Entidade	Atributos	Descrição	Tipo e Tamanho	Nulo	M.V	Derivado	Composto
Estudante	Número_C.C.	Número do Cartão de Cidadão do cliente	Valor inteiro positivo	Não	Não	Não	Não
	Nome	Identifica o cliente	45 Caracteres variáveis	Não	Não	Não	Não
	Género	Género do cliente	1 Caracter (M/F)	Não	Não	Não	Não
	Nome_Universidade	Nome da Universidade onde o cliente estuda	45 Caracteres variáveis	Não	Não	Não	Não
Viagem	idViagem	Número que identifica a viagem	Valor inteiro positivo	Não	Não	Não	Não
	Local_Partida	Local de partida da viagem	45 Caracteres variáveis	Não	Não	Não	Não
	Local_Chegada	Local de chegada da viagem	45 Caracteres variáveis	Não	Não	Não	Não
	Hora_Partida	Hora de partida da viagem	Hora	Não	Não	Não	Não
	Hora_Chegada	Hora de chegada da viagem	Hora	Não	Não	Não	Não
	Preço_Atual	Preço atual da viagem	Valor decimal positivo	Não	Não	Não	Não

Comboio	idComboio	Número que identifica o comboio	Valor inteiro positivo	Não	Não	Não	Não
	Observações	Observações relativas ao comboio	Texto	Sim	Não	Não	Não
	Lugar						
	idLugar	Número que identifica lugar	Valor inteiro positivo	Não	Não	Não	Não
	Numero _Cadeira	Número da cadeira	Valor inteiro positivo	Não	Não	Não	Não
	Carruagem	Letra que identifica a carruagem	1 caracter	Não	Não	Não	Não
Funcionário	idFuncionario	Número que identifica o funcionário	Valor inteiro positivo	Não	Não	Não	Não
	Nome	Nome do funcionário	45 Caracteres variáveis	Não	Não	Não	Não
Bilhete	idBilhete	Número que identifica o bilhete	Valor inteiro positivo	Não	Não	Não	Não
	Data_viagem	Data da viagem reservada	Data	Não	Não	Não	Não
	Data_emissão	Data em que é emitido o bilhete	Data	Não	Não	Não	Não
	Preço	Preço da viagem	Valor decimal positivo	Não	Não	Não	Não
	Lugar	Número do lugar associado ao bilhete reservado	Valor inteiro positivo	Não	Não	Não	Não

Tabela 3 – Associação de Atributos e Entidades

3.4. Determinar domínio dos atributos

Seguidamente, são descritos os domínios dos atributos referentes às diferentes entidades. O domínio consiste num conjunto de valores que pertencem a um determinado tipo, e que pode ser atribuído a cada atributo.

Entidade: Estudante

- Numero_cc: Número do cartão de cidadão do estudante. É um número inteiro positivo;
- Nome: Primeiro e último nome do estudante. É uma *string* com 45 caracteres variáveis;
- Género: Atributo correspondente ao sexo do estudante. É um único caracter, que pode tomar o valor 'M' (masculino) e 'F' (feminino).
- Nome_universidade: Nome da Universidade frequentada pelo aluno. É uma *string* com 45 caracteres variáveis;

Entidade: Viagem

- idViagem: Número que identifica a viagem. É representado por um número inteiro positivo;
- Local_partida: Local de partida da viagem. É representado por uma string e tem 45 caracteres variáveis;
- Local_chegada: Local de chegada da viagem. É uma string com 45 caracteres variáveis;
- Hora_partida: Hora de partida da viagem. Corresponde à hora, minuto e segundo marcados para o início da viagem;
- Hora_chegada: Hora de partida da viagem. Corresponde à hora, minuto e segundo marcados para o fim da viagem;
- Preço_atual: Preço atual da viagem, representado por um valor decimal positivo;

Entidade: Comboio

- idComboio: Número que identifica o comboio. É representado por um número inteiro positivo;
- Observações: Texto correspondente a observações relativas ao comboio.
- Lugar: Identificador dos lugares do comboio, caracterizado por:
    - IdLugar: Número inteiro positivo que identifica um lugar;
    - Numero_Cadeira: Número inteiro positivo que identifica o número da cadeira;
    - Carruagem: representado por um único caracter entre A-Z que identifica cada carruagem;

Entidade: Funcionário

- idFuncionario: Número que identifica o funcionário. É um número inteiro positivo;
- Nome: Nome do funcionário. É uma *string* com 45 caracteres variáveis;

Entidade: Bilhete

- idBilhete: Número que identifica o bilhete. É representado por um número inteiro positivo;
- Data_viagem: Data em que se realiza a viagem. Corresponde ao ano, mês e dia da viagem;
- Data_emissão: Data em que é emitido bilhete, representado pelo ano, mês e dia;
- Preço: Preço da viagem, representado por um valor decimal positivo;
- lugar: número que identifica o lugar associado ao bilhete reservado. É representado por um número inteiro positivo.

3.5. Determinar chaves primárias, candidatas e alternativas

De modo a identificar exclusivamente cada ocorrência das entidades e de maneira a garantir a integridade dos dados, seguiu-se à determinação das respetivas chaves primárias.

No caso da entidade Estudante, o atributo que melhor satisfaz o requisito de chave primária é o **numero_cc** pois trata-se de um identificador numérico irrepetível que garante que a chave não se repete para diferentes ocorrências da entidade. Ainda referente a esta entidade, o nome de estudante também poderia ser candidato a chave primária, porém, apesar de ser um aspeto de reconhecimento de um estudante, esta propriedade não cumpre os requisitos de uma chave primária, já que nada impede que haja mais do que uma ocorrência desta mesma entidade com o mesmo nome. Os restantes atributos da entidade Estudante também não satisfazem este requisito.

Para a entidade Funcionário selecionamos o atributo **id** como chave primária da entidade por ser um atributo numérico bastante simples, que identifica unicamente cada ocorrência da entidade Funcionário. O nome do funcionário, tal como acontece no caso do Estudante, também não pode ser considerado como chave primária.

Para a entidade bilhete, o atributo que melhor se encaixa no sentido de chave primária é o **idBilhete**. O atributo idLugar também poderia, à partida, ser um atributo considerado para chave primária, uma vez que somos levados a pensar que o número do lugar não se repete, mas para diferentes datas de viagem podem existir números repetidos do numero de lugar já que os mesmos comboios com os mesmos lugares efetuam viagens em datas diferentes. Por esse motivo e porque os restantes atributos não satisfazerem os requisitos de chave primária, determinou-se que o **idBilhete** seria a chave primária da entidade.

Para o caso da Viagem o atributo que melhor se encaixa nos requisitos de uma chave primária é o **idViagem**, por ser um atributo bastante simples e que identifica unicamente cada

viagem. Os atributos local_partida e local_chegada poderiam também ser candidatos a chave primária, porém pode existir mais do que uma viagem com o mesmo local de partida e local de chegada, ou seja, não identifica unicamente cada viagem.

Para a entidade Comboio, o atributo que identificamos como chave primária é o **idComboio,** uma vez que cada comboio tem um número associado, que o identifica inequivocamente. Como já referido anteriormente neste relatório, a entidade Comboio contém um atributo composto e multivalor, e como tal precisa também de ser identificado por uma chave primária. Ao analisarmos os atributos do lugar consideramos que o numero_cadeira poderia ser candidato a chave primária, no entanto, existem várias cadeiras com o mesmo número, para diferentes carruagens. Por isso, foi considerado que o **idLugar,** que corresponde a um número inteiro positivo, é o atributo que melhor identifica cada lugar, garantindo a sua unicidade.

3.6. Desenho do diagrama ER

Apresentamos, de seguida, o desenho do diagrama E-R (Entidade-Relacionamento) de forma a representar conceptualmente as relações entre as entidades da base de dados.

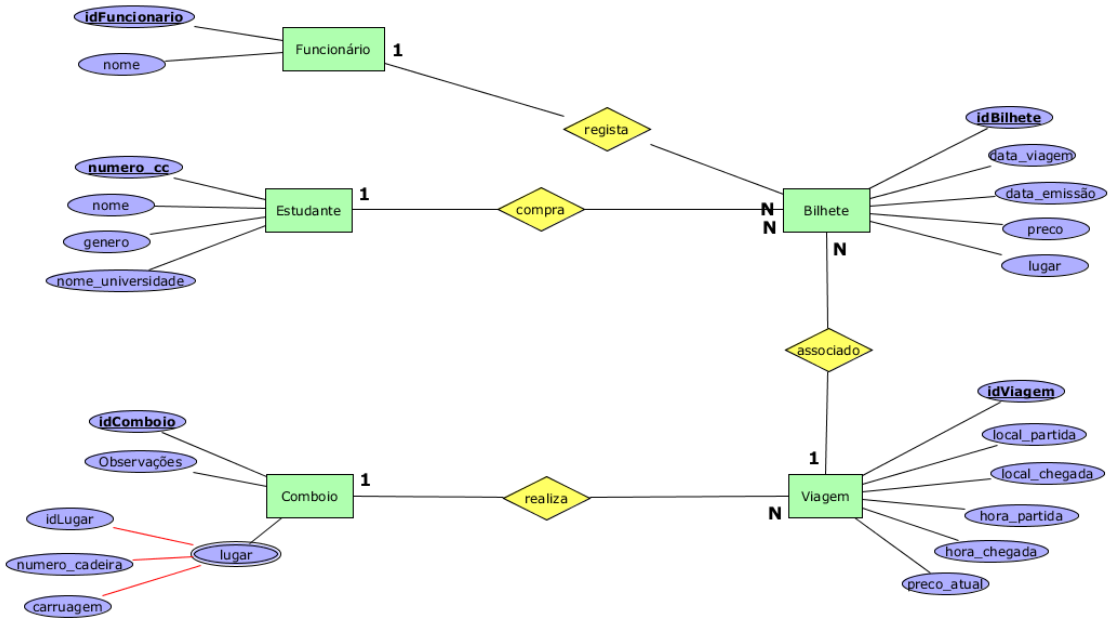

Figura 1 – Desenho do Diagrama ER

3.7. Verificação de redundâncias no modelo

Para a verificação de redundâncias no modelo foram seguidos os seguintes passos:

- Voltar a examinar todos os relacionamentos 1:1;

Este passo tem o objetivo de proteger o modelo dos casos em que identificamos duas entidades que representem o mesmo objeto. Ou seja, quando temos duas entidades que são sinónimos, devemos converge-las apenas numa. Após rever o modelo conceptual verificamos que não possuímos nenhum relacionamento 1:1.

- Remover relacionamentos com redundância;

Quando a mesma informação pode ser obtida por mais do que um relacionamento dizemos que existe redundância.

Através da simples visualização do nosso modelo conceptual verificamos que não existem redundâncias nos relacionamentos.

- Considerar a dimensão do tempo nos relacionamentos;

O significado do relacionamento entre duas entidades pode variar ao longo do tempo. Por esse motivo, examinamos os vários relacionamentos e não conseguimos encontrar situações que pudessem comprometer a redundância do modelo em função do tempo.

3.8. Validar modelo conceptual contra transações de utilizador

Neste momento esta definido um modelo conceptual que representa os requisitos impostos pelo Comboio Académico. Assim, justifica-se o controlo do modelo de forma a garantir que este é capaz de dar respostas às interrogações exigidas pelo utilizador. Se este modelo não conseguir responder a algumas das interrogações observadas, então significa que houve algum erro na modelação e alguns passos da modelação conceptual terão de ser repetidos e analisados novamente. Assim, as perguntas que o grupo entendeu serem as mais importantes são:

Quantos bilhetes foram reservados num determinado dia?

Para saber quantos bilhetes foram reservados num determinado dia apenas precisamos da entidade bilhete, a partir da qual adquirimos toda a informação importante para a obtenção do fim pedido.

Qual o número de reservas que determinado Estudante fez entre duas datas?

Esta pergunta é importante, pois percebe-se de que forma o cliente esta satisfeito com os serviços. Para sabermos quantas reservas foram efetuadas por um dado cliente, temos de ver o número do cliente na tabela bilhete e somar as várias reservas nas respetivas datas.

Qual a hora marcada de uma determinada viagem?

Para saber qual a hora de uma determinada viagem, basta aceder à entidade Viagem e indicar o respetivo número, conseguindo assim saber qual a sua hora.

Qual o número do comboio que faz uma determinada viagem?

Para sabermos o número do comboio que efetua uma determinada viagem basta percorrer a tabela viagem que contém um atributo.

Conseguimos aceder aos lugares que ainda não foram reservados para a viagem X no dia Y?

Sim. Através do dia e do número da viagem, conseguimos saber qual o comboio que realiza a viagem X, ou seja, temos acesso a todos os lugares desse comboio. Para saber os lugares que ainda não foram reservados, basta excluir da lista de todos os lugares desse comboio, aqueles que já estão associados a um bilhete dessa viagem nesse dia.

Podemos consultar os horários de uma viagem com local inicial X e local final Y?

Sim. Uma vez que cada viagem é realizada todos os dias numa determinada hora, é possível consultar a hora de todas as viagens existentes com local inicial X e local final Y.

3.9. Revisão do modelo com o utilizador

Após a realização do modelo conceptual, este foi revisto pelo utilizador. Nenhum problema foi detetado e por isso o modelo de dados foi aceite.

4. Modelo Lógico

4.1. Derivar relações para o modelo lógico de dados

4.1.1. Entidades fortes

As entidades presentes no nosso modelo conceptual são: Estudante, Funcionário, Bilhete, Viagem e Comboio. Todas estas entidades são entidades fortes e, como tal, cada uma destas vai corresponder à criação de uma tabela com os atributos dessa entidade.

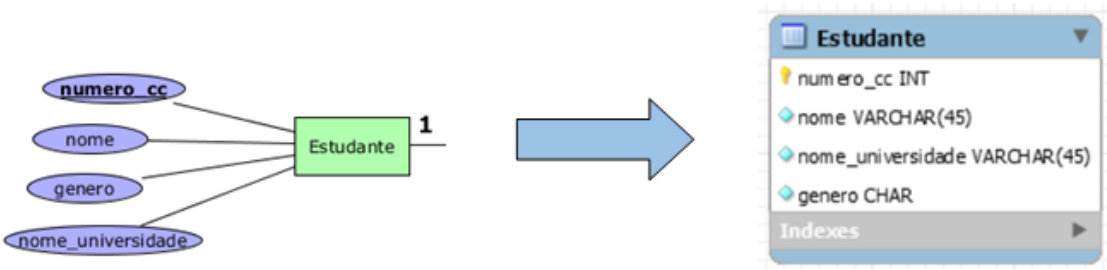

Figura 2 – Conversão do modelo conceptual para modelo lógico (entidade Estudante)

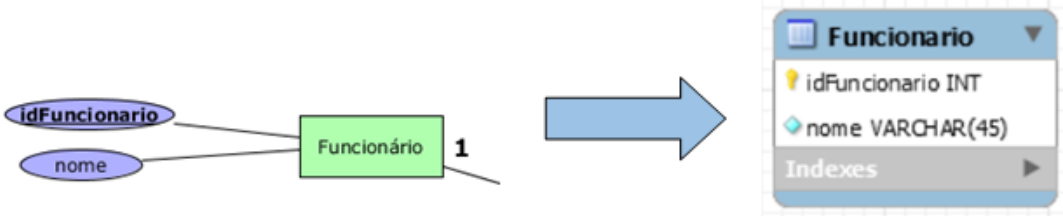

Figura 3 – Conversão do modelo conceptual para modelo lógico (entidade Funcionário)

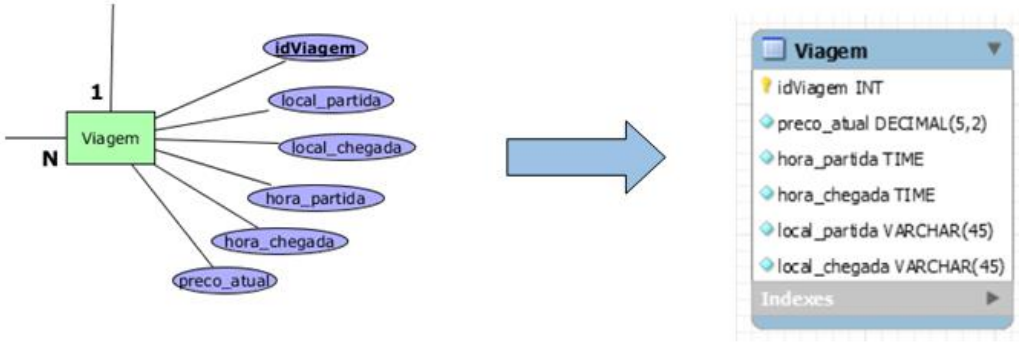

Figura 4 – Conversão do modelo conceptual para modelo lógico (entidade Viagem)

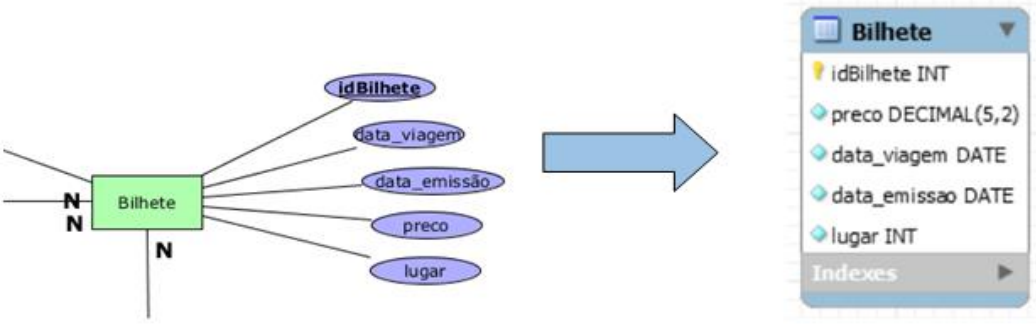

Figura 5 – Conversão do modelo conceptual para modelo lógico (entidade Bilhete)

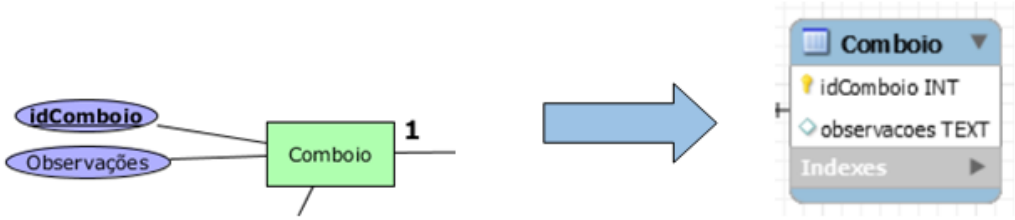

Figura 6 – Conversão do modelo conceptual para modelo lógico (entidade Comboio)

4.1.2. Entidades fracas

No nosso modelo conceptual não há nenhuma ocorrência de entidades fracas.

4.1.3. Relacionamentos binários 1:N

No nosso modelo conceptual todos os relacionamentos são de 1:N. Para cada um destes relacionamentos, a entidade que se encontra do lado de maior cardinalidade (N) vai possuir como chave estrangeira uma referência à entidade com menor cardinalidade (1).

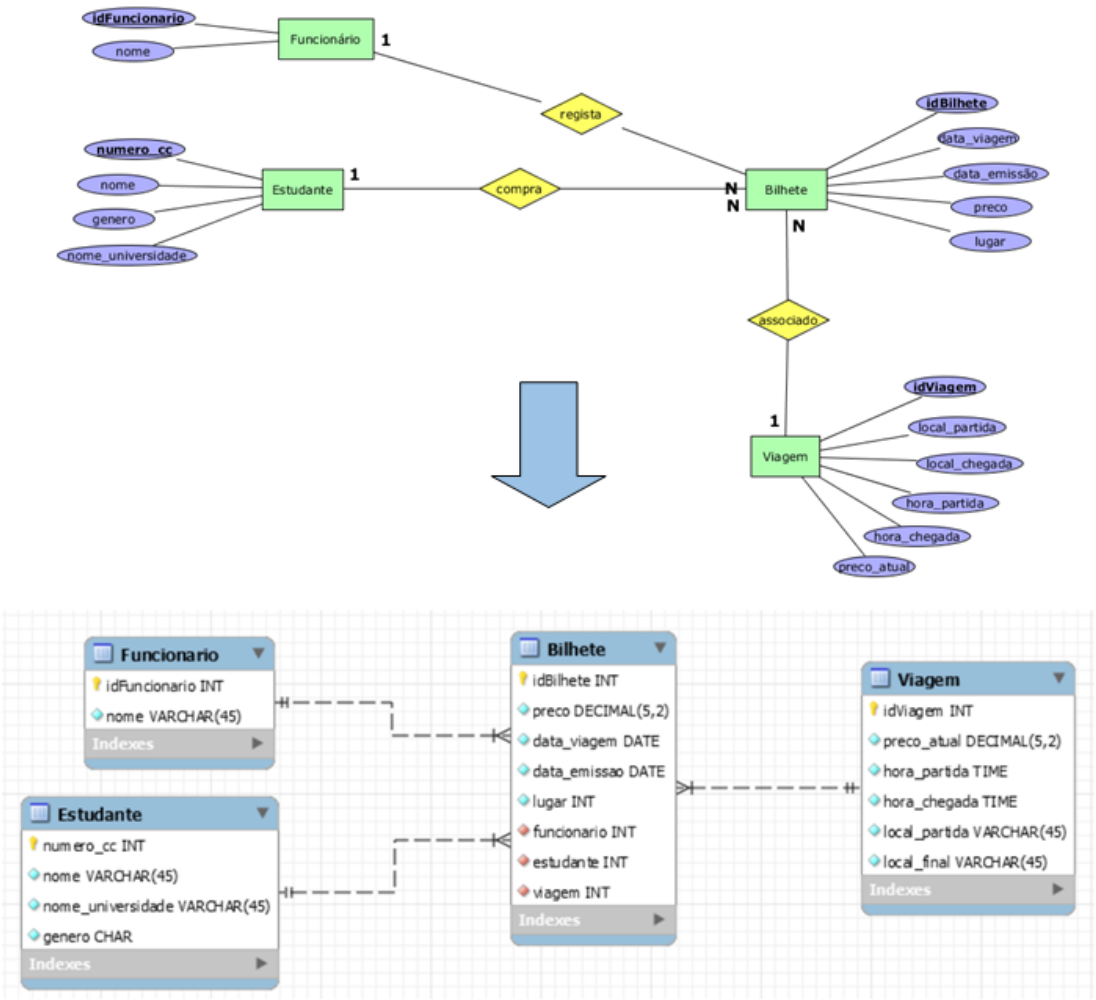

Figura 7 – Conversão do modelo conceptual para o modelo lógico (relacionamentos 1:N com a entidade Bilhete)

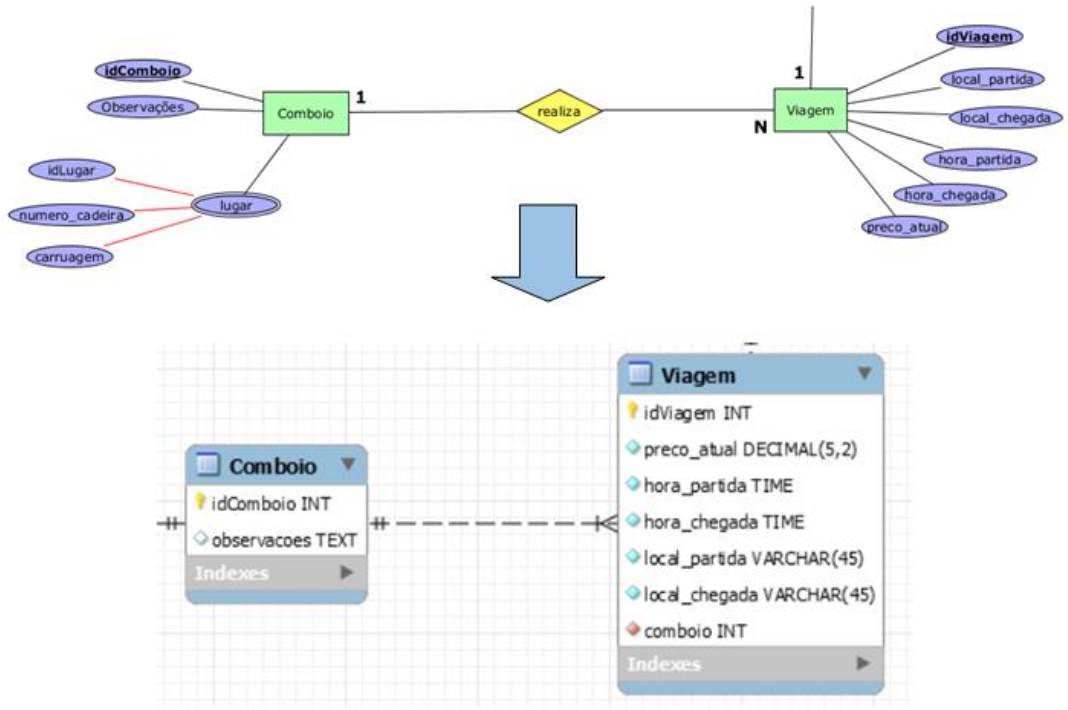

Figura 8 - Conversão do modelo conceptual para o modelo lógico (relacionamento 1:N entre Comboio e Viagem)

4.1.4. Relacionamentos 1:1

No nosso modelo conceptual não existem relacionamentos 1:1.

4.1.5. Relacionamentos recursivos 1:1

No nosso modelo conceptual não existem relacionamentos recursivos 1:1.

4.1.6. Relacionamentos superclasse/subclasse

No nosso modelo conceptual não existem relacionamentos superclasse/subclasse.

4.1.7. Relacionamentos N:M

No nosso modelo conceptual não existem relacionamento N:M.

4.1.8. Relacionamentos Complexos

No nosso modelo conceptual não existem relacionamentos complexos.

4.1.9. Atributos multivalor

No nosso modelo conceptual existe um atributo multivalor, o lugar. Como tal, para este atributo é criada uma nova tabela. Nesta tabela, cujo nome é o mesmo do atributo, é colocada uma cópia da chave primária da entidade à qual pertence o atributo, que também faz parte da chave primária.

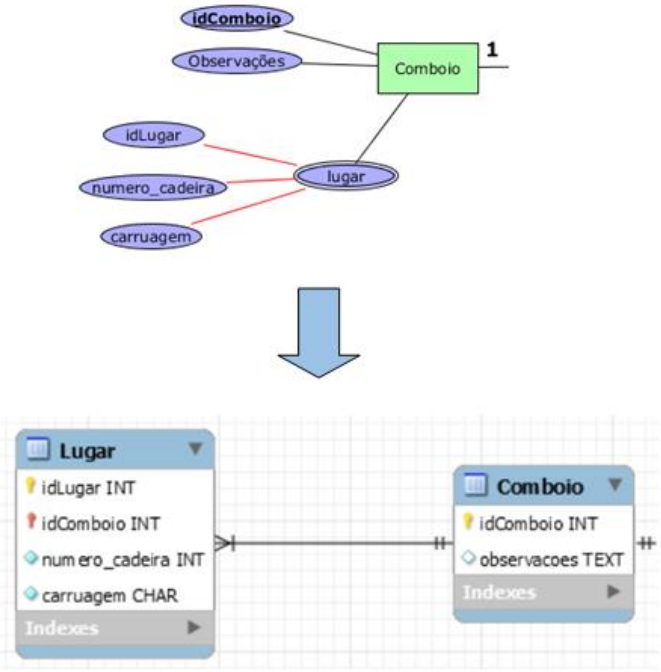

Figura 9 - Conversão do modelo conceptual para o modelo lógico (atributo multivalor Lugar)

4.2. Validar relações usando normalização

Após analisarmos as dependências funcionais de cada relação verificamos que as tabelas respeitam as três primeiras regras de normalização, e como tal, podemos concluir que se encontram normalizadas até à terceira forma normal da normalização.

4.3. Validar relações com as transações do utilizador

Neste ponto decidimos implementar três das mais importantes transações, que no contexto desta base de dados, são suportadas pelo modelo lógico através da realização do mapa de transações.

A primeira transação definida torna possível a inserção de um novo Estudante que poderá efetuar reservas e usufruir do comboio Académico.

A segunda transação permite adicionar um novo funcionário à empresa, que realiza o processo de registo das reservas.

Para finalizar, a terceira transação corresponde à inserção de um novo bilhete, este que é o requisito para poder realizar as viagens para os destinos pretendidos.

- **Inserir novo Estudante**

Esta primeira transação de inserção de um novo estudante começa por inserir um estudante na tabela Estudante incluindo o seu número de C.C, nome, nome da universidade em que estuda e o seu género.

- **Inserir novo Funcionário**

Esta transação permite o registo de um novo funcionário executando uma inserção na tabela Funcionário. Esta inserção tem que incluir o número do funcionário, bem como o seu nome.

- **Inserir novo Bilhete**

Esta transação é a ação mais importante do sistema pois sem ele o Comboio Académico não prestava serviços. Para iniciar, começamos por inserir o novo bilhete na tabela bilhete incluindo o seu preço, data da viagem, data de emissão como o número da viagem que pretende viajar. De seguida, é necessário introduzir as informações relativas ao estudante que está a efetuar a reserva, do funcionário que está a registá-la, assim como da viagem correspondente.

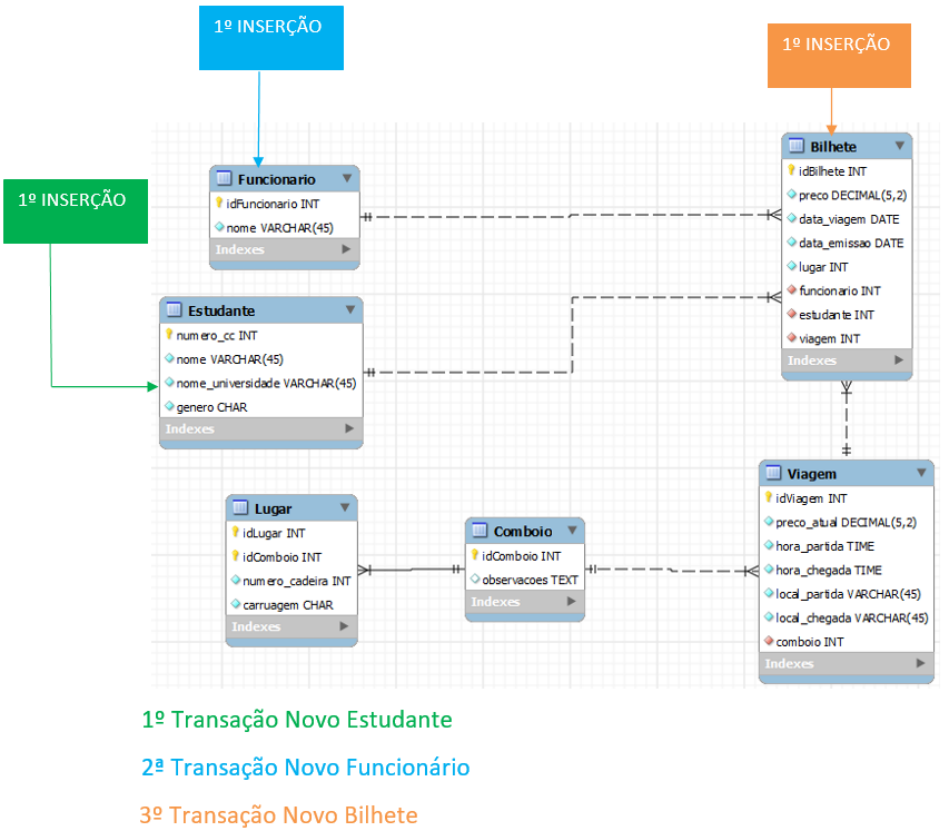

Figura 10 – Transações com o utilizador

4.4. Elaboração e validação do esquema lógico

Após a elaboração do modelo lógico através da passagem efetuada no ponto 4.1 e das validações presentes nos pontos 4.2 e 4.3, conclui-se que o esquema lógico se encontra corretamente elaborado.

De seguida, apresentamos uma imagem do esquema lógico elaborado.

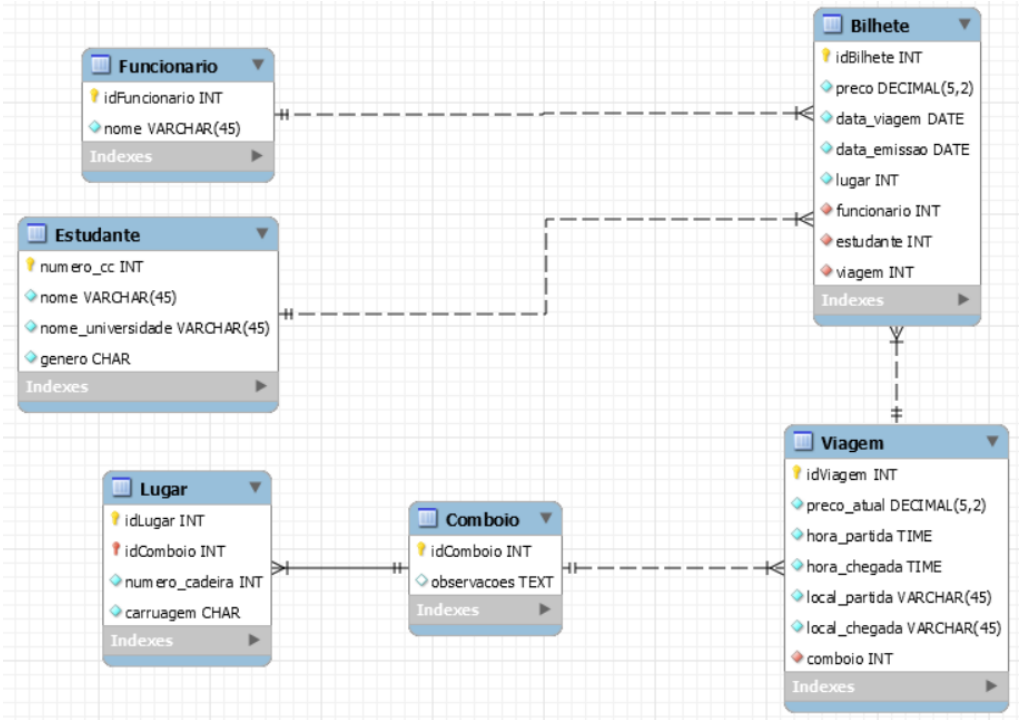

Figura 11 – Modelo lógico

4.5. Revisão do modelo lógico com o utilizador

Após a elaboração do modelo lógico, é importante rever o modelo com o utilizador, com o intuito de assegurar que não existe nenhuma falha que possa causar problemas no futuro.

Posto isto, e após efetuarmos esta revisão, conclui-se que o problema em questão estava representado corretamente no modelo lógico construído.

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

Tendo em conta os requisitos, consideramos que o tamanho inicial da base de dados terá:

- 4 funcionários;
- 14 estudantes;

- 4 comboios:
    - 2 comboios com duas carruagens de 3 lugares cada;
    - 2 comboios com uma única carruagem com 8 lugares;
- 10 viagens
    - 2 viagens realizadas pelo comboio 1;
    - 2 viagens realizadas pelo comboio 2;
    - 4 viagens realizadas pelo comboio 3;
    - 2 viagens realizadas pelo comboio 4;
- 21 bilhetes
    - 3 bilhetes para a viagem 1;
    - 1 bilhete para a viagem 2;
    - 2 bilhetes para a viagem 3;
    - 6 bilhetes para a viagem 4;
    - Nenhum bilhete para a viagem 5;
    - 1 bilhete para a viagem 6;
    - 2 bilhetes para a viagem 7;
    - 2 bilhetes para a viagem 8;
    - 1 bilhete para a viagem 9;
    - 3 bilhetes para a viagem 10;

A base de dados pode ser aumentada no futuro, uma vez que não encontramos nenhum problema que possa ocorrer quando existe um aumento de dados na BD.

5. Modelo Físico

5.1. Tradução do modelo lógico para o SGBD e implementação

Nesta parte do trabalho, traduzimos o modelo lógico construído para a sua implementação física, através do SGDB MySQL.

5.1.1. Relações base

Nesta primeira fase da implementação do modelo físico, foram criadas todas as tabelas que constituem o modelo lógico de dados.

Assim, para cada relação, apresenta-se o domínio dos seus atributos e a sua estrutura:

- **Estudante**

Domínio:

numero_cc	número inteiro positivo não nulo
nome	sequência de caracteres variável não nula
nome_universidade	sequência de caracteres variável não nula
genero	caracter não nulo

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Estudante` (
  `numero_cc` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `nome_universidade` VARCHAR(45) NOT NULL,
  `genero` CHAR NOT NULL,
  PRIMARY KEY (`numero_cc`))
ENGINE = InnoDB;
```

Figura 12 – MySQL criação da tabela Estudante

- **Funcionário**

Domínio:

idFuncionario	número inteiro positivo não nulo
nome	sequência de caracteres variável não nula

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Funcionario` (
  `idFuncionario` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idFuncionario`))
ENGINE = InnoDB;
```

Figura 13 – MySQL criação da tabela Funcionário

- **Comboio**

Domínio:

idComboio	número inteiro positivo não nulo
observacoes	texto

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Comboio` (
  `idComboio` INT NOT NULL,
  `observacoes` TEXT NULL,
  PRIMARY KEY (`idComboio`))
ENGINE = InnoDB;
```

Figura 14 – MySQL criação da tabela Comboio

- **Lugar**

Domínio:

idLugar	número inteiro positivo não nulo
IdComboio	número inteiro positivo não nulo
numero_cadeira	número inteiro positivo não nulo
carruagem	caracter não nulo

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Lugar` (
  `idLugar` INT NOT NULL,
  `idComboio` INT NOT NULL,
  `numero_cadeira` INT NOT NULL,
  `carruagem` CHAR NOT NULL,
  PRIMARY KEY (`idLugar`, `idComboio`),
  INDEX `FK_comboio_idx` (`idComboio` ASC),
  CONSTRAINT `FK_comboioLugares`
    FOREIGN KEY (`idComboio`)
    REFERENCES `ComboioAcademico`.`Comboio` (`idComboio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 15 – MySQL criação da tabela Lugar

- **Viagem**

Domínio:

idViagem	número inteiro positivo não nulo
preco_atual	número decimal positivo não nulo
hora_partida	variável do tipo tempo não nula
hora_chegada	variável do tipo tempo não nula
local_partida	sequência de caracteres não nula
local_chegada	sequência de caracteres não nula
comboio	número inteiro positivo não nulo

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Viagem` (
  `idViagem` INT NOT NULL,
  `preco_atual` DECIMAL(5,2) NOT NULL,
  `hora_partida` TIME NOT NULL,
  `hora_chegada` TIME NOT NULL,
  `local_partida` VARCHAR(45) NOT NULL,
  `local_chegada` VARCHAR(45) NOT NULL,
  `comboio` INT NOT NULL,
  PRIMARY KEY (`idViagem`),
  INDEX `FK_comboio_idx` (`comboio` ASC),
  CONSTRAINT `FK_viagemComboio`
    FOREIGN KEY (`comboio`)
    REFERENCES `ComboioAcademico`.`Comboio` (`idComboio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 16 – MySQL criação da tabela Viagem

- **Bilhete**

Domínio:

idBilhete	número inteiro positivo não nulo
preco	número decimal positivo não nulo
data_viagem	variável do tipo data não nula
data_emissao	variável do tipo data não nula
lugar	número inteiro positivo não nulo
funcionario	número inteiro positivo não nulo
estudante	número inteiro positivo não nulo
viagem	número inteiro positivo não nulo

```
CREATE TABLE IF NOT EXISTS `ComboioAcademico`.`Bilhete` (
  `idBilhete` INT NOT NULL,
  `preco` DECIMAL(5,2) NOT NULL,
  `data_viagem` DATE NOT NULL,
  `data_emissao` DATE NOT NULL,
  `lugar` INT NOT NULL,
  `funcionario` INT NOT NULL,
  `estudante` INT NOT NULL,
  `viagem` INT NOT NULL,
  PRIMARY KEY (`idBilhete`),
  INDEX `FK_Funcionario_idx` (`funcionario` ASC),
  INDEX `FK_Estudante_idx` (`estudante` ASC),
  INDEX `FK_Viagem_idx` (`viagem` ASC),
  CONSTRAINT `FK_Funcionario`
    FOREIGN KEY (`funcionario`)
    REFERENCES `ComboioAcademico`.`Funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_Estudante`
    FOREIGN KEY (`estudante`)
    REFERENCES `ComboioAcademico`.`Estudante` (`numero_cc`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_Viagem`
    FOREIGN KEY (`viagem`)
    REFERENCES `ComboioAcademico`.`Viagem` (`idViagem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 17 – MySQL criação da tabela Bilhete

5.1.2. Representação dos atributos derivados

No nosso modelo já anteriormente apresentado neste relatório, não existem atributos derivados.

5.1.3. Povoamento da Base de Dados

- **Estudante**

```
INSERT INTO estudante VALUES (11111111, 'Gonçalo Lopes', 'Universidade do Minho', 'M');
INSERT INTO estudante VALUES (11221511, 'Luis Carvalho', 'Universidade do Porto', 'M');
INSERT INTO estudante VALUES (15661111, 'Rita Lopes', 'Universidade de Vigo', 'F');
INSERT INTO estudante VALUES (11111661, 'Bruno Rodrigues', 'Universidade do Minho', 'M');
INSERT INTO estudante VALUES (11112221, 'Ana Cunha', 'Instituto Politecnico V. Castelo', 'F');
INSERT INTO estudante VALUES (11166171, 'Rui Pereira', 'Universidade do Porto', 'M');
INSERT INTO estudante VALUES (11999991, 'Filipa Pinto', 'Universidade de Santiago Compostela', 'F');
INSERT INTO estudante VALUES (33311111, 'Leticia Sousa', 'Universidade do Minho', 'F');
INSERT INTO estudante VALUES (14441111, 'Miguel Oliveira', 'Universidade de Vigo', 'M');
INSERT INTO estudante VALUES (11115581, 'Andreia Cunha', 'Universidade do Minho', 'F');
INSERT INTO estudante VALUES (11131321, 'Sónia Pereira', 'Universidade do Porto', 'F');
INSERT INTO estudante VALUES (15551111, 'Marta Lopes', 'Instituto Politecnico V. Castelo', 'F');
INSERT INTO estudante VALUES (34453215, 'Ricardo Guedes', 'Instituto Politecnico Cavado e Ave', 'M');
INSERT INTO estudante VALUES (34555219, 'João Matos', 'Instituto Politecnico Cavado e Ave', 'M');
```

Figura 18 – Povoamento Estudante

- **Funcionário**

```
INSERT INTO funcionario VALUES (1, 'João Sousa');
INSERT INTO funcionario VALUES (2, 'Vitor Silva');
INSERT INTO funcionario VALUES (3, 'Inês Alves');
INSERT INTO funcionario VALUES (4, 'Sara Lima');
```

Figura 19 – Povoamento Funcionário

- **Comboio**

```
INSERT INTO comboio VALUES (1, 'Apenas trabalha de manhã');
INSERT INTO comboio VALUES (2, 'Realiza percursos entre Braga-Porto');
INSERT INTO comboio VALUES (3, NULL);
INSERT INTO comboio VALUES (4, 'Percursos em Espanha');
```

Figura 20 – Povoamento Comboio

- **Lugar**

```
-- inserir lugares comboio 1 (1 única carruagem com 8 lugares)
INSERT INTO lugar VALUES (1, 1, 1, 'A');
INSERT INTO lugar VALUES (2, 1, 2, 'A');
INSERT INTO lugar VALUES (3, 1, 3, 'A');
INSERT INTO lugar VALUES (4, 1, 4, 'A');
INSERT INTO lugar VALUES (5, 1, 5, 'A');
INSERT INTO lugar VALUES (6, 1, 6, 'A');
INSERT INTO lugar VALUES (7, 1, 7, 'A');
INSERT INTO lugar VALUES (8, 1, 8, 'A');

-- inserir lugares comboio 2 (2 carruagens, 3 lugares cada)
INSERT INTO lugar VALUES (1, 2, 1, 'A');
INSERT INTO lugar VALUES (2, 2, 2, 'A');
INSERT INTO lugar VALUES (3, 2, 3, 'A');
INSERT INTO lugar VALUES (4, 2, 1, 'B');
INSERT INTO lugar VALUES (5, 2, 2, 'B');
INSERT INTO lugar VALUES (6, 2, 3, 'B');

-- inserir lugares comboio 3 (1 única carruagem com 8 lugares)
INSERT INTO lugar VALUES (1, 3, 1, 'A');
INSERT INTO lugar VALUES (2, 3, 2, 'A');
INSERT INTO lugar VALUES (3, 3, 3, 'A');
INSERT INTO lugar VALUES (4, 3, 4, 'A');
INSERT INTO lugar VALUES (5, 3, 5, 'A');
INSERT INTO lugar VALUES (6, 3, 6, 'A');
INSERT INTO lugar VALUES (7, 3, 7, 'A');
INSERT INTO lugar VALUES (8, 3, 8, 'A');

-- inserir lugares comboio 4 (2 carruagens, 3 lugares cada)
INSERT INTO lugar VALUES (1, 4, 1, 'A');
INSERT INTO lugar VALUES (2, 4, 2, 'A');
INSERT INTO lugar VALUES (3, 4, 3, 'A');
INSERT INTO lugar VALUES (4, 4, 1, 'B');
INSERT INTO lugar VALUES (5, 4, 2, 'B');
INSERT INTO lugar VALUES (6, 4, 3, 'B');
```

Figura 21 – Povoamento Lugar

- **Viagem**

```
-- comboio 1
INSERT INTO viagem VALUES (1, '2.50', '09:30:00', '10:15:00', 'Viana do Castelo', 'Barcelos', 1);
INSERT INTO viagem VALUES (2, '2.50', '11:00:00', '11:45:00', 'Barcelos', 'Viana do Castelo', 1);
-- comboio 2
INSERT INTO viagem VALUES (3, '2.00', '10:00:00', '11:00:00', 'Braga', 'Porto', 2);
INSERT INTO viagem VALUES (4, '2.00', '11:00:00', '11:45:00', 'Porto', 'Braga', 2);
-- comboio 3
INSERT INTO viagem VALUES (5, '2.00', '12:00:00', '13:00:00', 'Braga', 'Porto', 3);
INSERT INTO viagem VALUES (6, '2.80', '15:00:00', '16:00:00', 'Porto', 'Barcelos', 3);
INSERT INTO viagem VALUES (7, '2.30', '17:00:00', '17:50:00', 'Barcelos', 'Viana do Castelo', 3);
INSERT INTO viagem VALUES (8, '4.30', '19:00:00', '20:30:00', 'Viana do Castelo', 'Braga', 3);
-- comboio 4
INSERT INTO viagem VALUES (9, '3.30', '12:00:00', '13:00:00', 'Santiago de Compostela', 'Corunha', 4);
INSERT INTO viagem VALUES (10, '3.30', '16:00:00', '17:00:00', 'Corunha', 'Santiago de Compostela', 4);
```

Figura 22 – Povoamento Viagem

- **Bilhete**

```
-- inserir reservas
-- viagem 1 (duas reservas para o dia 2016-12-20 e uma para o dia 2016-12-21)
INSERT INTO bilhete VALUES (1, '2.50', '20161220', '20161123', 2, 1, 11999991, 1);
INSERT INTO bilhete VALUES (2, '2.50', '20161220', '20161124', 3, 2, 14441111, 1);
INSERT INTO bilhete VALUES (3, '2.50', '20161221', '20161124', 2, 3, 11999991, 1);
-- viagem 2 (uma reserva para o dia 2016-12-19)
INSERT INTO bilhete VALUES (4, '2.50', '20161219', '20161125', 1, 4, 11111111, 2);
-- viagem 3 (duas reservas para o dia 2016-12-20)
INSERT INTO bilhete VALUES (5, '2.00', '20161220', '20161125', 5, 2, 11221511, 3);
INSERT INTO bilhete VALUES (6, '2.00', '20161220', '20161125', 3, 2, 34453215, 3);
-- viagem 4 (viagem lotada para o dia 2016-12-21)
INSERT INTO bilhete VALUES (7, '2.00', '20161221', '20161127', 1, 1, 11111111 , 4);
INSERT INTO bilhete VALUES (8, '2.00', '20161221', '20161127', 2, 1, 11221511, 4);
INSERT INTO bilhete VALUES (9, '2.00', '20161221', '20161127', 3, 1, 15661111, 4);
INSERT INTO bilhete VALUES (10, '2.00', '20161221', '20161127', 4, 1, 11111661 , 4);
INSERT INTO bilhete VALUES (11, '2.00', '20161221', '20161127', 5, 1, 11112221, 4);
INSERT INTO bilhete VALUES (12, '2.00', '20161221', '20161127', 6, 1, 11166171, 4);
-- viagem 5 (sem reservas)
-- viagem 6 (uma reserva para o dia 2016-12-21)
INSERT INTO bilhete VALUES (13, '2.80', '20161221', '20161127', 2, 3, 11999991, 6);
-- viagem 7 (uma reserva para o dia 2016-12-22 e uma reserva para o dia 2016-12-23)
INSERT INTO bilhete VALUES (14, '2.30', '20161222', '20161126', 1, 2, 33311111, 7);
INSERT INTO bilhete VALUES (15, '2.30', '20161223', '20161126', 3, 2, 14441111, 7);
-- viagem 8 (duas reservas para o dia 2016-12-21)
INSERT INTO bilhete VALUES (16, '4.30', '20161221', '20161127', 4, 2, 11115581, 8);
INSERT INTO bilhete VALUES (17, '4.30', '20161221', '20161127', 5, 2, 14441111, 8);
-- viagem 9 (uma reserva para o dia 2016-12-23)
INSERT INTO bilhete VALUES (18, '3.30', '20161223', '20161126', 1, 2, 11131321, 9);
-- viagem 10 (tres reservas para o dia 2016-12-23)
INSERT INTO bilhete VALUES (19, '3.30', '20161223', '20161130', 1, 1, 15551111, 10);
INSERT INTO bilhete VALUES (20, '3.30', '20161223', '20161130', 2, 1, 34453215, 10);
INSERT INTO bilhete VALUES (21, '3.30', '20161223', '20161130', 3, 1, 34555219, 10);
```

Figura 23 – Povoamento Bilhete

5.1.4. Restrições

As restrições que foram detetas para o problema, são as seguintes:

- Um estudante não pode comprar mais do que um bilhete para a mesma viagem, no mesmo dia.

```
DELIMITER //
CREATE TRIGGER estudanteBilhete
BEFORE INSERT ON Bilhete
FOR EACH ROW
BEGIN

DECLARE msg VARCHAR (200);

IF (new.estudante IN
    (SELECT estudante FROM bilhete where
            estudante = new.estudante
            and viagem = new.viagem
            and data_viagem = new.data_viagem))
THEN
    SET msg = 'O estudante já reservou a viagem para a data indicicada';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;

END IF;
END //
```

Figura 24 – *Trigger* estudanteBilhete

Para verificar a correta implementação deste *trigger*, inserimos um bilhete para uma viagem que o estudante já tinha reservado, na mesma data:

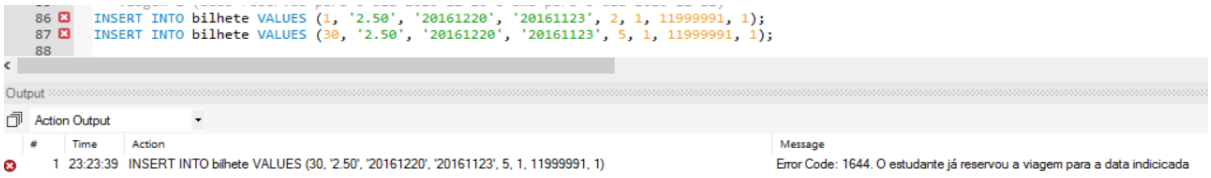

Figura 25 – Comprovativo do correto funcionamento do *trigger* estudanteBilhete

- Um bilhete novo não pode ser associado a um lugar já ocupado para essa viagem, nesse mesmo dia.

```
DELIMITER //
CREATE TRIGGER lugarOcupado
BEFORE INSERT ON Bilhete
FOR EACH ROW
BEGIN

DECLARE msg VARCHAR (200);

IF (new.lugar IN
    (SELECT lugar FROM bilhete where
            data_viagem = new.data_viagem
            and viagem = new.viagem))
THEN
    SET msg = 'Lugar já ocupado';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;

END IF;
END //
```

Figura 26 – *Trigger* lugarOcupado

Tal como para o *trigger* anterior, apresentamos de seguida o comprovativo do seu correto funcionamento:

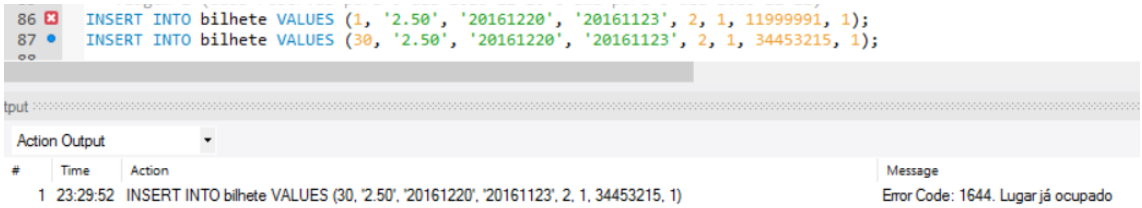

Figura 27 – Comprovativo do correto funcionamento do *trigger* lugarOcupado

5.2. Análise de transações

Neste ponto procede-se à análise das transações anteriormente indicadas no 4.3 onde as principais transações deste projeto são:

- Inserir novo Estudante;
- Inserir novo Funcionário;
- Inserir novo Bilhete;

Apesar destas transações constituírem processos simples, previmos que as tabelas Estudante, Bilhete e Funcionário serão as que terão um maior número de acessos.

Na tabela seguinte pode ser vista a relação das transações com as operações sobre as tabelas a que estas acedem.

I – Insert / R – Read / D – Delete / U – Update

	Inserir novo Estudante				Inserir novo Funcionário				Inserir novo Bilhete			
	I	R	D	U	I	R	D	U	I	R	D	U
Estudante	X									X		
Funcionário					X					X		
Viagem										X		
Bilhete									X			

Tabela 4 – Relações das transações com as operações sobre tabelas

- **Inserir estudante**

```
CREATE PROCEDURE `inserirEstudante`(IN nCC INT,
                                    IN nomeEst VARCHAR(45),
                                    IN nomeUni VARCHAR(45),
                                    IN genero CHAR(1))
BEGIN
DECLARE erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
START TRANSACTION;

INSERT INTO estudante VALUES (nCC, nomeEst, nomeUni, gen);

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

Figura 28- Transação inserir novo estudante

- **Inserir funcionário**

```
CREATE PROCEDURE `inserirFuncionario`(IN nr INT,
                                      IN nomeFun VARCHAR(45))
BEGIN
DECLARE erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
START TRANSACTION;

INSERT INTO funcionario VALUES (nr, nomeFun);

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

Figura 29 – Transação inserir novo funcionário

- **Inserir bilhete**

```
CREATE PROCEDURE `inserirBilhete`(IN nr INT,
                                  IN preco DECIMAL(5,2),
                                  IN dataV DATE,
                                  IN dataE DATE,
                                  IN lugar INT,
                                  IN func INT,
                                  IN est INT,
                                  IN viagem INT)
BEGIN
DECLARE erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
START TRANSACTION;

INSERT INTO bilhete VALUES (nr, preco, dataV, dataE, lugar, func, est, viagem);

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

Figura 30 – Transação inserir novo bilhete

5.3. Estimativa dos requisitos do espaço em disco com base no SGBD escolhido

Neste ponto é feita uma estimativa do espaço necessário em disco para a fase inicial da base de dados.

Em primeiro lugar, calculamos o espaço necessário para cada registo de cada tabela, tendo em conta o tipo de dados de cada atributo. De seguida, com base neste valor total calculado para cada registo de cada tabela, multiplicamos pelo número de dados que foram inseridos inicialmente na base de dados.

Estudante:

Atributos	Tipo	Tamanho (*Bytes*)
numero_cc	INT	4
nome	VARCHAR(45)	45
nome_universidade	VARCHAR(45)	45
genero	CHAR	1

Tabela 5 – Tamanhos dos atributos de Estudante

Para cada registo da tabela Estudante, são ocupados 4 + 45 +45 +1 = 95 *bytes.* Como temos cardinalidade 14 nesta tabela, dá um total de 14 * 95 = 1330 *bytes.*

Funcionário:

Atributos	Tipo	Tamanho *(Bytes)*
idFuncionário	INT	4
nome	VARCHAR(45)	45

Tabela 6 – Tamanho dos atributos de Funcionário

Para cada registo da tabela Funcionário, são ocupados 4 + 45 = 49 *bytes.* Como temos cardinalidade 4 nesta tabela, dá um total de 4 * 49 = 196 *bytes.*

Viagem:

Atributos	Tipo	Tamanho *(Bytes)*
idViagem	INT	4
preço_atual	DECIMAL	5
hora_partida	TIME	5
hora_chegada	TIME	5
local_partida	VARCHAR(45)	45
local_chegada	VARCHAR(45)	45
comboio	INT	4

Tabela 7 – Tamanho dos atributos da Viagem

Para cada registo da tabela Viagem, são ocupados 4 + 5 + 5 + 5 + 45 + 45 + 4 = 113 *bytes.* Como temos cardinalidade 10 nesta tabela, dá um total de 10 * 113 = 1130 *bytes.*

Comboio:

Atributos	Tipo	Tamanho (*Bytes*)
idComboio	INT	4
observaçoes	TEXT	42

Tabela 8 – Tamanho dos atributos de Comboio

Considerado que para o tipo de dados TEXT, um valor estimado pode ser de 40 caracteres, para cada registo da tabela Comboio, são ocupados 4 + 42 = 46 *bytes*. Como temos cardinalidade 4 nesta tabela, dá um total de 4 * 46 = 184 *bytes.*

Lugar:

Atributos	**Tipo**	**Tamanho (*Bytes)***
idLugar	INT	4
IdComboio	INT	4
numero_cadeira	INT	4
carruagem	CHAR	1

Tabela 9 – Tamanho dos atributos de Lugar

Para cada registo da tabela Lugar, são ocupados 4 + 4 + 4 + 1 = 13 *bytes.* Como temos cardinalidade 28 nesta tabela, dá um total de 28 * 13 = 364 *bytes.*

Bilhete:

Atributos	**Tipo**	**Tamanho *(Bytes)***
idBilhete	INT	4
preco	DECIMAL	5
data_viagem	DATE	3
data_emissao	DATE	3
lugar	INT	4
funcionario	INT	4
estudante	INT	4
viagem	INT	4

Tabela 10 – Tamanho dos atributos de Bilhete

Para cada registo da tabela Funcionário, são ocupados 4 + 5 + 3 + 3 + 4 + 4 + 4 + 4 = 31 bytes. Como temos cardinalidade 21 nesta tabela, dá um total de 21 * 31 = 651 *bytes*.

Para calcular o espaço em disco ocupado pelo nosso SBD é preciso somar o tamanho ocupado pelas várias tabelas. Fazendo 1330 + 196 + 1130 + 184 + 364 + 651 = 3855 *bytes*, o espaço necessário para o povoamento inicial é de aproximadamente 3.855 *KBytes*.

5.4. Definição das vistas dos utilizadores e regras de acesso

Neste tópico são apresentadas as regras de acesso de cada perfil de utilização da base de dados.

Como tal, existem dois utilizadores que interagem com a base de dados, sendo que estes têm permissões de acesso diferentes à BD.

- Funcionário – Utilizador que regista as reservas no sistema
- Administrador – responsável por gerir todos os conteúdos da BD

```
-- Criacao de users
-- Administrador
CREATE USER 'admin'@'localhost';
    SET PASSWORD FOR 'admin'@'localhost' = 'admin123';

-- Funcionarios
CREATE USER 'func1'@'localhost';
    SET PASSWORD FOR 'func1'@'localhost' = 'func1';

CREATE USER 'func2'@'localhost';
    SET PASSWORD FOR 'func2'@'localhost' = 'func2';

CREATE USER 'func3'@'localhost';
    SET PASSWORD FOR 'func3'@'localhost' = 'func3';

CREATE USER 'func4'@'localhost';
    SET PASSWORD FOR 'func4'@'localhost' = 'func4';
```

Figura 31 – Criação de *User's*

De seguida, implementamos as diferentes permissões para cada utilizador:

- Funcionário – Cada funcionário pode consultar todas as tabelas, no entanto apenas pode consultar, inserir, remover, e atualizar dados das tabelas Estudante e Bilhete.
- Administrador – Pode consultar, inserir, remover e atualizar os dados de qualquer tabela.

```
-- Permissoes
-- Administrador
GRANT SELECT, INSERT, UPDATE, DELETE ON comboioacademico.* TO 'admin'@'localhost';
```

Figura 32 – Permissões do Administrador

```
GRANT SELECT, INSERT, DELETE, UPDATE ON comboioacademico.bilhete TO 'func1'@'localhost',
                                                                    'func2'@'localhost',
                                                                    'func3'@'localhost',
                                                                    'func4'@'localhost';
```

Figura 33 – Permissões do Funcionário em Bilhete

```
GRANT SELECT, INSERT, DELETE, UPDATE ON comboioacademico.estudante TO 'func1'@'localhost',
                                                                      'func2'@'localhost',
                                                                      'func3'@'localhost',
                                                                      'func4'@'localhost';
```

Figura 34 - Permissões do Funcionário em Estudante

```
GRANT SELECT ON comboioacademico.viagem TO 'func1'@'localhost',
                                           'func2'@'localhost',
                                           'func3'@'localhost',
                                           'func4'@'localhost';
REVOKE INSERT, DELETE, UPDATE ON comboioacademico.viagem FROM 'func1'@'localhost',
                                                              'func2'@'localhost',
                                                              'func3'@'localhost',
                                                              'func4'@'localhost';
```

Figura 35 - Permissões do Funcionário em Viagem

```
GRANT SELECT ON comboioacademico.comboio TO 'func1'@'localhost',
                                            'func2'@'localhost',
                                            'func3'@'localhost',
                                            'func4'@'localhost';
REVOKE INSERT, DELETE, UPDATE ON comboioacademico.comboio FROM 'func1'@'localhost',
                                                               'func2'@'localhost',
                                                               'func3'@'localhost',
                                                               'func4'@'localhost';
```

Figura 36 - Permissões do Funcionário em Comboio

```
GRANT SELECT ON comboioacademico.lugar TO 'func1'@'localhost',
                                          'func2'@'localhost',
                                          'func3'@'localhost',
                                          'func4'@'localhost';
REVOKE INSERT, DELETE, UPDATE ON comboioacademico.lugar FROM 'func1'@'localhost',
                                                             'func2'@'localhost',
                                                             'func3'@'localhost',
                                                             'func4'@'localhost';
```

Figura 37 - Permissões do Funcionário em Lugar

```
GRANT SELECT ON comboioacademico.funcionario TO 'func1'@'localhost',
                                                'func2'@'localhost',
                                                'func3'@'localhost',
                                                'func4'@'localhost';
REVOKE INSERT, DELETE, UPDATE ON comboioacademico.funcionario FROM 'func1'@'localhost',
                                                                   'func2'@'localhost',
                                                                   'func3'@'localhost',
                                                                   'func4'@'localhost';
```

Figura 38 - Permissões do Funcionário em Funcionário

6. Conclusões e Trabalho Futuro

Terminada a realização do projeto, conseguimos concluir que é crucial fazer uma boa análise de requisitos que o sistema deve respeitar, uma vez que esta vai influenciar a estruturação e implementação do sistema de base de dados.

Após o levantamento e análise de requisitos, identificamos as diferentes entidades fundamentais, os seus atributos e os relacionamentos existentes entre estas. Foi importante também realizar uma boa interpretação dos atributos, uma vez que foi necessário identificar um atributo como atributo composto e multivalor. De seguida, determinamos os domínios dos atributos e identificamos as chaves primárias, candidatas e alternativas correspondentes a cada entidade. Verificamos também se existiam redundâncias no modelo, à qual conseguimos concluir que não existia qualquer tipo de problema. Posto isto, conseguimos elaborar o modelo conceptual para o nosso trabalho, que foi validado com as transações de utilizador. Nesta fase, tivemos algumas dificuldades que foram esclarecidas em conjunto com o professor.

Uma vez validado o modelo conceptual prosseguimos para implementação do modelo lógico. Nesta etapa foi fundamental perceber como tínhamos que implementar as diferentes entidades, os seus relacionamentos, bem como os atributos. As maiores preocupações do grupo no processo de passagem do modelo conceptual para o modelo lógico, estão relacionados com a integridade dos dados e a normalização uma vez que são estas que garantem a consistência dos dados e a ausência de redundância. Já depois de implementado o modelo, este foi validado com as transações de utilizador. Foi também definido um tamanho inicial para a base de dados, assim como uma análise do crescimento futuro.

Finalmente, traduzimos o modelo lógico para o modelo físico, onde tivemos bastante preocupação com as relações base, assim como com as restrições fundamentais para o sistema. Em seguida, analisamos e implementamos as diferentes transações de utilizador já identificadas anteriormente no modelo lógico. Foi importante também fazer uma estimativa do espaço em disco que o sistema ocupa, para compreender se existia algum problema de tamanho excessivo. Por último, definimos os diferentes perfis de utilizadores, assim como as diferentes regras de acesso que cada um tem.

A realização deste projeto permitiu-nos consolidar os conhecimentos adquiridos na unidade curricular, bem como identificar algumas lacunas que temos que corrigir futuramente.

Referências

[1] Thomas Connolly, Carolyn Begg 2005. *Database Systems: A Practical Approach To Design, Implementation, and Management*, 4ª edição, Adisson Wesley.

Lista de Siglas e Acrónimos

BD Base de Dados

ER Entidade - Relacionamento

SBD Sistema de Base de Dados

SGBD Sistema Gestor de Base de Dados

SBDR Sistema de Base de Dados Relacional

MV Multivalor