Trabalho prático I - Flight Radar

Informática - 2º ano
Bases de dados II

Daniel Duarte 220000942
Tiago Tomás 220001585

Índice

Índice de figuras

Apresentação do tema do trabalho – Flight Radar

O tema deste trabalho assemelha-se a um radar que acompanha a trajetória de aviões durante os seus respetivos voos, sendo essa a principal parte de visualização de dados, para além destes elementos de visualização de dados, também estão presentes outros, como grelhas, possibilitando a restante leitura dos dados presentes em cada tabela da base de dados, BD, como também formulários para realizar operações de criação, atualização e exclusão de registos, comumente referidos pela sigla, CRUD.

Toda a interação com o sistema é feita através de um navegador web, sendo o trabalho desenvolvido em JavaScript Node.js. A base de dados trata-se de uma do tipo relacional, MySQL.

Como o intuito do trabalho era utilizar o maior número de comportamentos presentes no MySQL o trabalho nasceu com uma natureza muito modular, sendo possível realizar qualquer operação CRUD em qualquer tabela.

Trabalho realizado

O início da realização do trabalho deu-se pela idealização da base de dados com o diagrama entidade relacionamento.

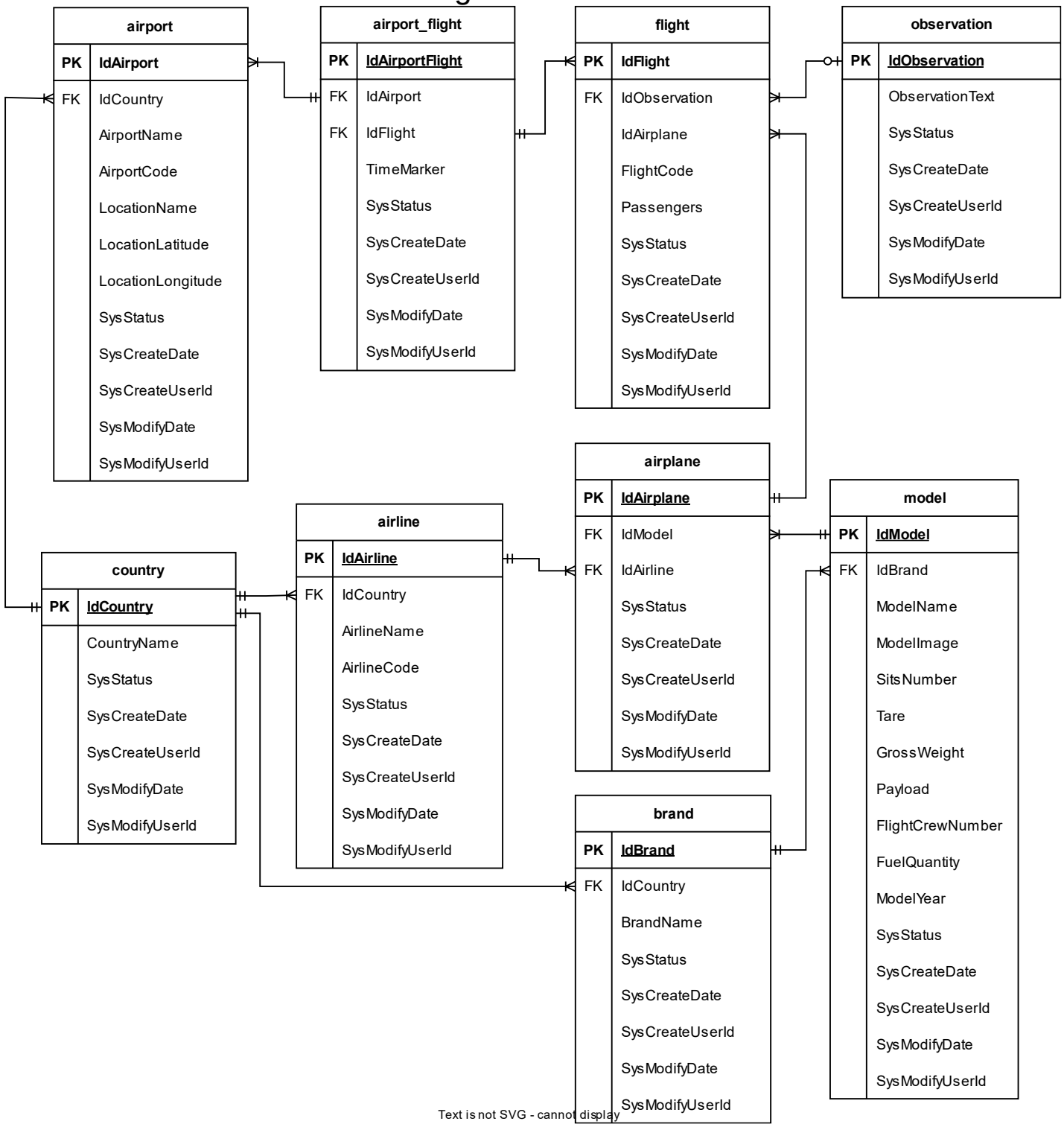

Figura 1 - Diagrama ER Final

Na figura acima encontra-se a versão final desse diagrama, fruto de alterações cumulativas da evolução do conceito do trabalho.

O conceito originou na entidade *flight* – voo, sendo ela a principal, esta como pretende representar um voo, conta com um avião, representado por uma chave estrangeira, um código de voo, único, pois cada voo requer um identificador próprio, pois diferentes companhias aéreas podem fornecer voos com origens e destinos iguais, este código replica a norma usada na vida real por elas. O voo terá também o número de passageiros e uma observação, representada pela chave estrangeira, esta é opcional, e é usada para relatar algo que se deu durante um voo, como por exemplo a falhas de um motor do avião, em termos de BD ao se adicionar um voo, um *trigger* é executado com 25% de probabilidade de dar uma observação a ele mesmo.

Para as entidades que se relacionam com o *flight* é possível observar a *observation* - observação, pensada para ser uma lista de observações já existente, sob forma de texto para o qual a própria BD define para os voos.

A tabela associativa *airport_flight* nasce fruto de uma relação de muitos para muitos entre a *flight* e a *airport* – aeroporto, representa um momento em que um voo esteve num aeroporto, com uma estampilha de tempo, pode indicar a partida, caso seja o marco de tempo mais antigo de um voo, como a chegada, se for a estampilha mais recente, os valores intermédios tratam-se de escalas, sabe-se em que aeroporto esses movimentos ocorreram pois a tabela para além de contemplar o *IdFlight* – IdVoo como chave estrangeira, conta também com o *IdAirport* – IdAeroporto.

A entidade *airport* – aeroporto, representa um determinado aeroporto de um país daí a referência sob forma de chave estrangeira, *IdCountry* – IdPaís à tabela *country* – país. O aeroporto também tem um nome e código, ambos únicos, uma localização e as coordenadas.

Tanto um aeroporto, como uma companhia aérea - *airline* estão ligados a um país – *country,* que se trata de uma entidade apenas com nome.

A *airline* é a detentora dos aviões, possui um nome e um código, ambos únicos.

A entidade *airplane* – avião representa os aviões que executam os voos, os aviões possuem um modelo.

O modelo – *model* trata de todas as características de um avião como nome do modelo, número de passageiro, peso, entre outras. O modelo tem uma marca a si associada.

Brand – marca é a entidade responsável a designar uma marca a cada modelo de avião, ela para além de possuir um nome, tem ligação com a entidade país.

Em todas as entidades é possível observar um conjunto de atributos comuns, estes são de controlo, servem para observar quando uma tabela foi criada e por que utilizador, de forma semelhante, o mesmo se dá para quando é modificada, nestes atributos de controlo é também possível se observar a maneira como se remove registos da base de dados, que na realidade são sempre escondidos e nunca excluídos, graças ao atributo “sys_status” e “IsDelete” onde fica explicito se um registo está ativo ou não.

Após a conclusão do diagrama, deu-se início ao desenvolvimento da BD com um *script* de inicialização, responsável pela criação de toda a estrutura e a inserção de dados por omissão.

Ao se ter uma base de dados, o foco foi em programá-la, grande parte da lógica desenvolvida em BD pode ser resumida da seguinte forma: as operações CRUD utilizam dois *procedures* para cada entidade, um para inserção, atualização e exclusão de dados e outro para busca filtrada e ordenada dos mesmos. A restante lógica envolve comportamentos auxiliares como busca de informação especifica, de partidas e chegadas de um voo num aeroporto e em que momento, envio de dados específicos para partes do *website*, como evidenciado pelos aviões presentes no mapa e *triggers*, usados para ocultarem tabelas dependentes de outras que foram removidas, entre outros comportamentos.

Feita a base de dados, partiu-se para a sua integração com uma estrutura web, na qual se utilizou o Express e o mysql2 para a ligação com a BD, ambas dependências do gestor de pacotes do Node.js. Todo o código foi feito com uma arquitetura *Model-View-Controller* em mente, cada entidade da BD corresponde a um modelo em código e essa comunicação é feita através dos repositórios à qual chamam os procedimentos da BD. Nos controladores é feita toda a lógica de processamento de pedidos e respostas. Os serviços servem para auxílio à programação, tal como os objetos de transferência de dados que manipulam, DTOs. Por fim a vista é responsável por todos os elementos legíveis, as páginas web.

Objetivos alcançados

Com o desenvolvimento do trabalho foi possível criar-se uma base de dados que possibilite operações CRUD, graças à sua lógica, para além de conseguir alimentar toda uma estrutura web, não só com operações simples de CRUD, como também outros comportamentos, executados por *triggers*, prevenção de erros com uso de transações e a possibilidade de leitura dos mesmos graças a *handlers*. Isto tudo utilizado outros requisitos necessários para execução do trabalho, como *joins*, subconsultas e cursores.

Na página principal é possível observa um mapa populado por aviões.

Figura 2 - Mapa de radar de voos

Cada avião corresponde a um voo com as suas informações.

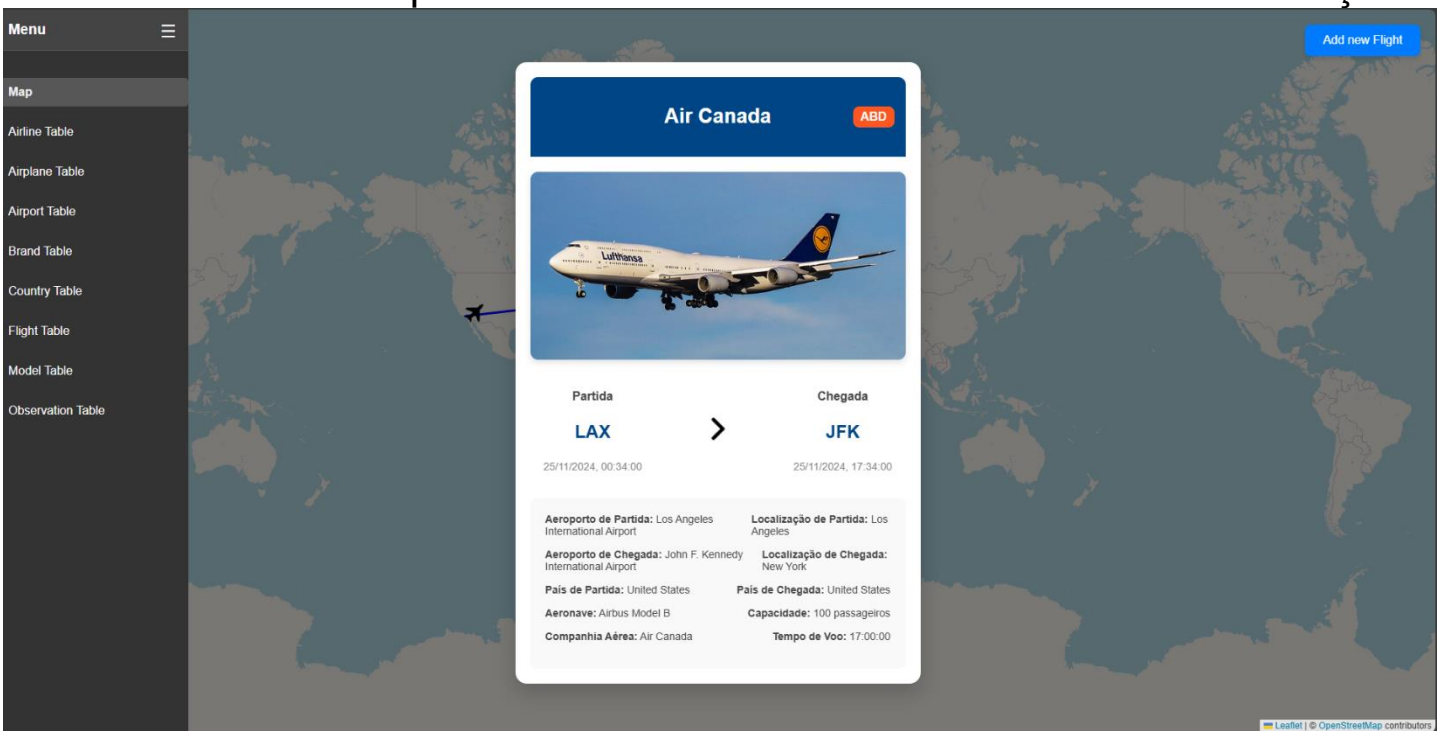

Figura 3 - Informação de um voo

Que podem ser adicionados com através de um formulário ao se premir num respetivo botão no canto superior direito do ecrã.

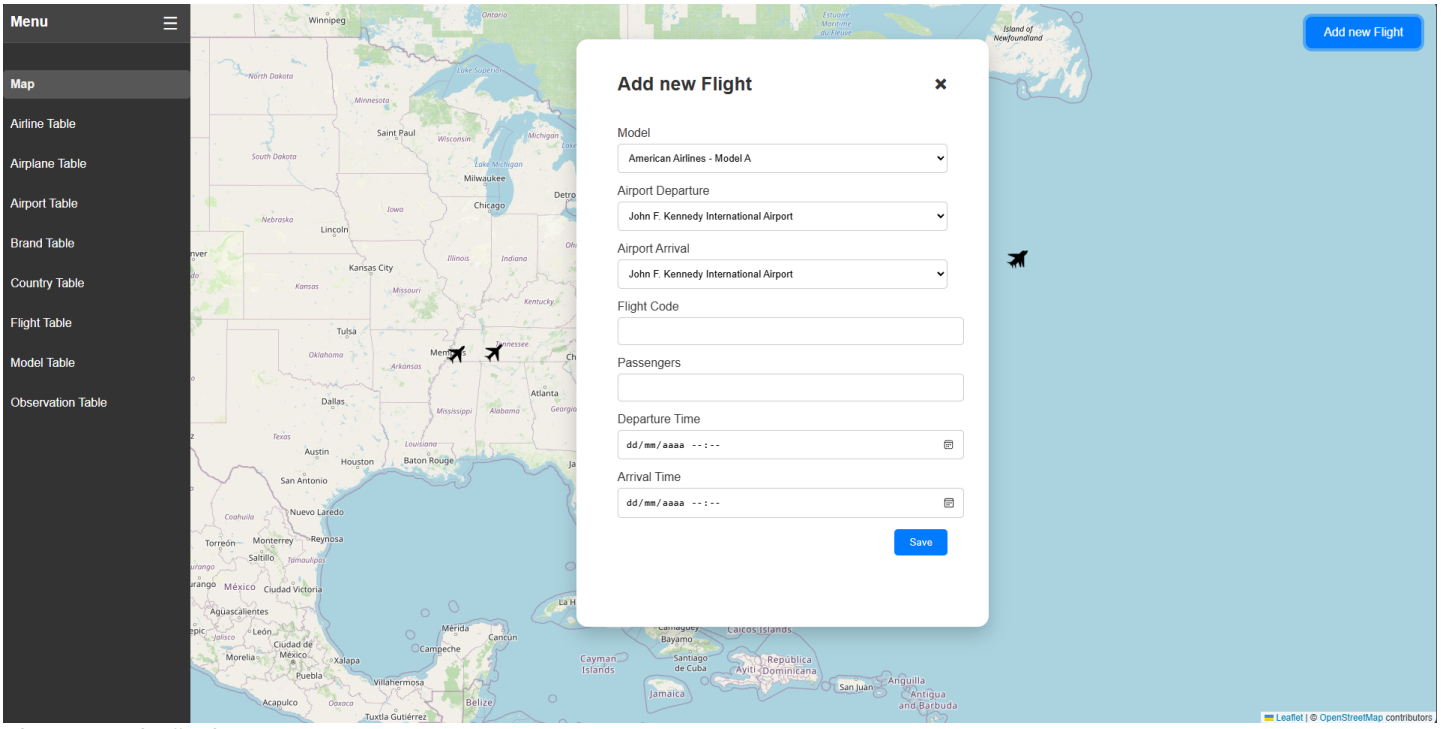

Figura 4 - Criação de um voo

Para além deste tipo dados também é possível interagir com as restantes tabelas da BD através do menu lateral.

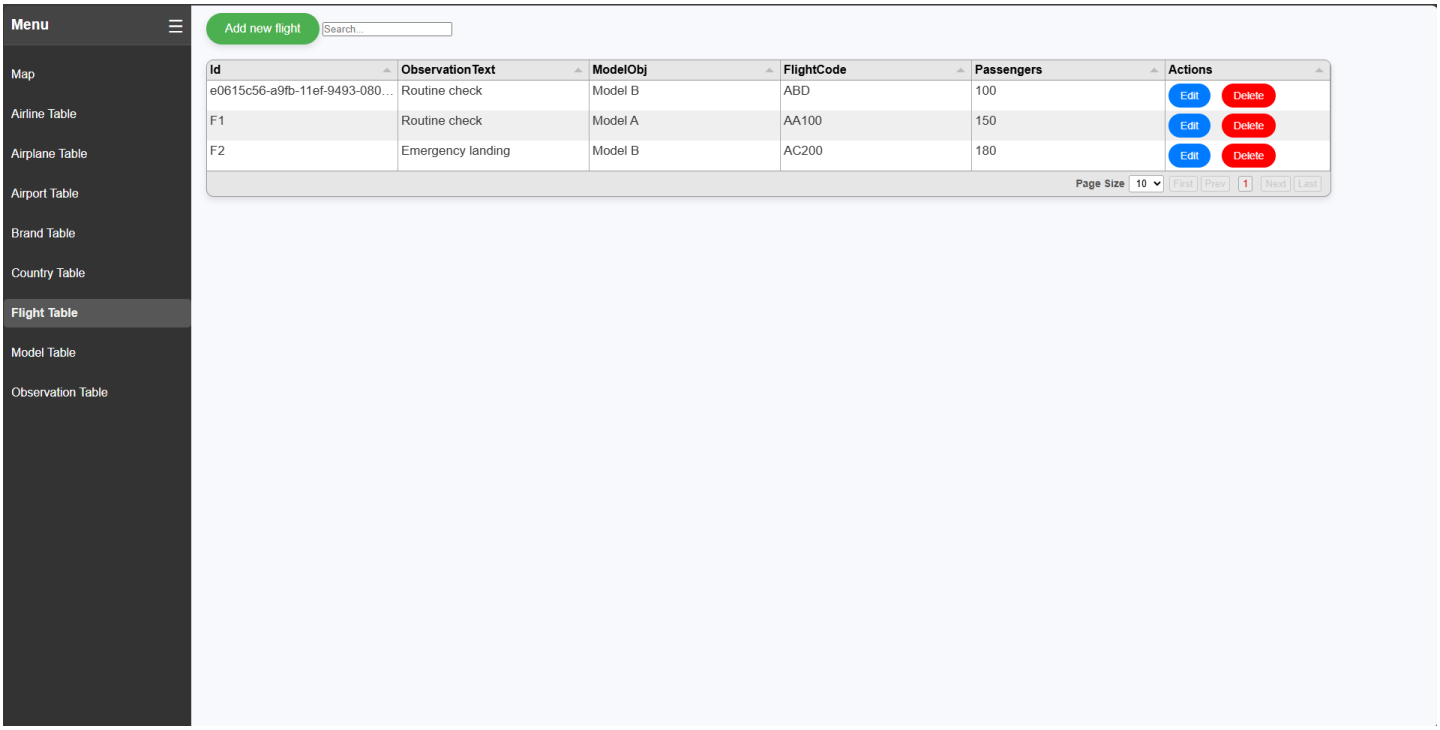

Figura 5 - Grelha de voos com outras informações e operações CRUD

Dificuldades

A maior dificuldade foi a definição da base de dados, com o diagrama entidade relacionamento, especialmente, em como seriam estabelecidas as escalas dos voos, afetando o relacionamento das tabelas da BD, que sofreram várias alterações já numa fase bastante avançada do trabalho, o que resultou em grandes perdas de tempo.

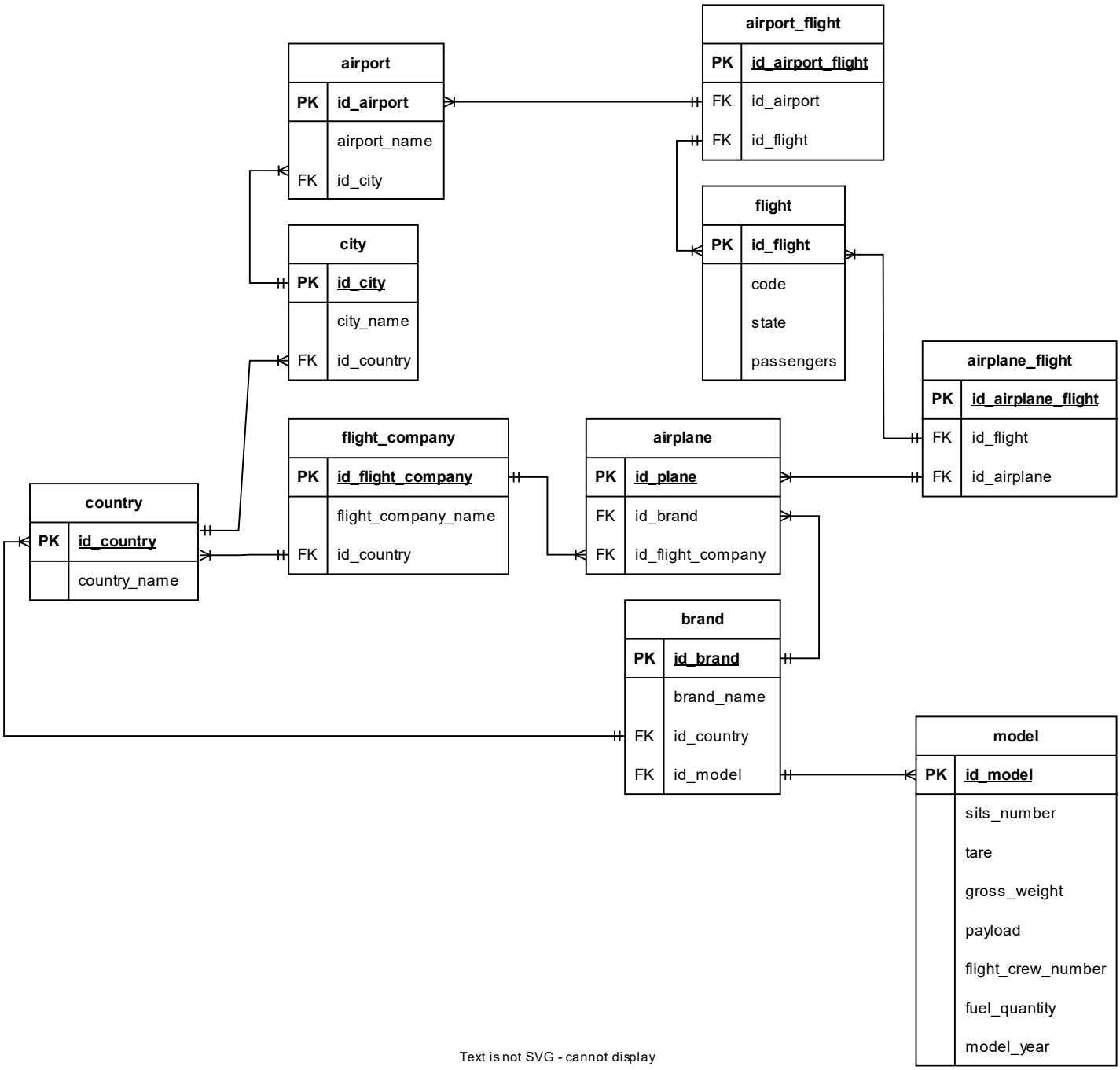

Figura 6 - Versão 1 Diagrama ER

Esta trata-se da primeira versão do diagrama, com uma tabela associativa a mais, para contemplar a troca de aviões num mesmo voo e

ainda com grande parte dos parâmetros em falta, o problema foi mesmo começar-se o desenvolvimento da BD com o diagrama incompleto, pensou-

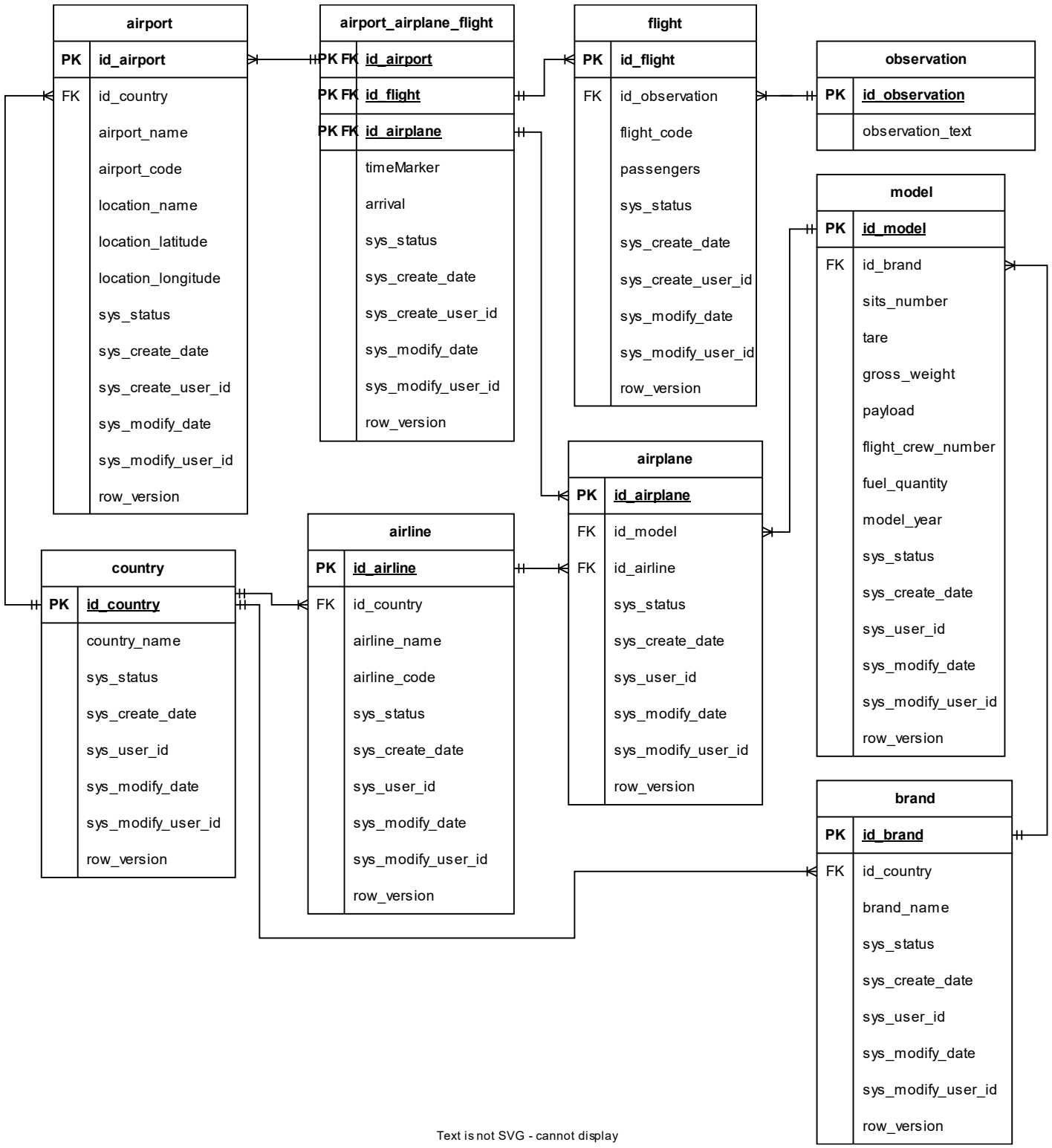

Figura 7 - Versão 2 Diagrama ER

se que se iria poupar tempo, sendo que se deu o completo oposto.

O diagrama acima trata-se da segunda versão feita, mais próxima de que viria a ser a versão final, porém com alguns atributos ainda por remover e outros por adicionar.

Outra dificuldade deu-se ao tentar integrar toda a matéria dada em aula no trabalho, dado que, se pretendia criar comportamentos funcionais e uteis em vez de se criarem só para cumprir com o requisito de eles existirem.

A dificuldade que nos acompanhou até à conclusão do trabalho foi encontrar o equilíbrio entre o que programar na base de dados e o que se fazer em código. É facto que dado o trabalho ser realizado para a avaliação na unidade curricular de Bases de Dados II, obriga grande parte da lógica ser feita na BD, porém a questão está em que ponto é que se torna viável deixar de ter apenas operações genéricas na BD que podem ser usadas em qualquer *Front-End* e criar comportamentos que só se aplicam para um *Front-End* específico. Um exemplo é como os aviões, mostrados no mapa, e a sua informação são processados, decidimos carregar os dados diretamente da BD sem qualquer tipo de formatação em código, os dados são específicos e já filtrados tanto para a posição dos aviões como para o painel que exibe a informação relevante de um voo ao se selecionar um avião. Podia-se aplicar o mesmo conceito para o resto do trabalho e evitar-se-ia o uso de DTOs, porém alargaria o tempo de desenvolvimento em pontos cruciais em que as regras do projeto ainda não estavam exatamente definidas, como a estrutura de cada página e que informação era apresentada nelas. Ao se fazer pedidos à BD, diminuiria as possibilidades em que se podia trabalhar com a estrutura das páginas web, não porque se tornaria impossível, mas porque ficava inviável devido aos elevados custos temporais e à impossibilidade de se ir criando conforme se fosse idealizando.

Conclusão

Para concluir, este trabalho permitiu trabalhar com metodologias já conhecidas e já familiares, porém a um nível diferente, programando-se na BD em vez de se usar uma linguagem orientada a objetos. Além de se aplicar novos conceitos de bases de dados relacionais dados em aula, como *triggers,* funções, procedimentos, cursores e toda a sintaxe necessária para a escrita de lógica numa base de dados relacional.