U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2L.EIC012 – Bases de Dados

Sistema de Controlo de um Aeroporto

Novembro de 2023

Grupo 106

Trabalho Realizado Por:

Hugo Alexandre Almeida Barbosa	**up202205774**@fe.up.pt
João Pedro Nogueira da Hora Santos	**up202205794**@fe.up.pt
Rodrigo Albergaria Coelho e Silva	**up202205188**@fe.up.pt

ÍNDICE

Introdução

Para esta segunda entrega do trabalho no âmbito da Unidade Curricular de Bases de Dados, é-nos proposta a conversão do modelo conceptual definido no trabalho anterior para esquema relacional, assim como a criação de uma base de dados em SQL do esquema e o carregamento de dados para a mesma.

Na primeira fase, vamos apresentar pequenas correções e melhorias no modelo em UML a partir do feedback obtido na entrega anterior do trabalho.

Seguidamente, procedemos à sua conversão para o esquema relacional e identificamos as dependências funcionais de cada uma das entidades consideradas. A partir destas dependências funcionais devemos verificar e corrigir as entidades de modo a que todos se apresentem na Forma Normal Boyce-Codd ou na Terceira Forma Normal.

Depois das correções no esquema da base de dados, procedemos à sua conversão para documento de código SQL e à criação de dados para introduzir na mesma.

Em cada uma destas fases vamos recorrendo a ferramentas de Inteligência Artificial de modo a melhorar a nossa solução para cada tarefa (ou pelo menos ter outra abordagem), explicando o input colocado e o output obtido por este, ou, as diferenças entre o output e a nossa própria resposta.

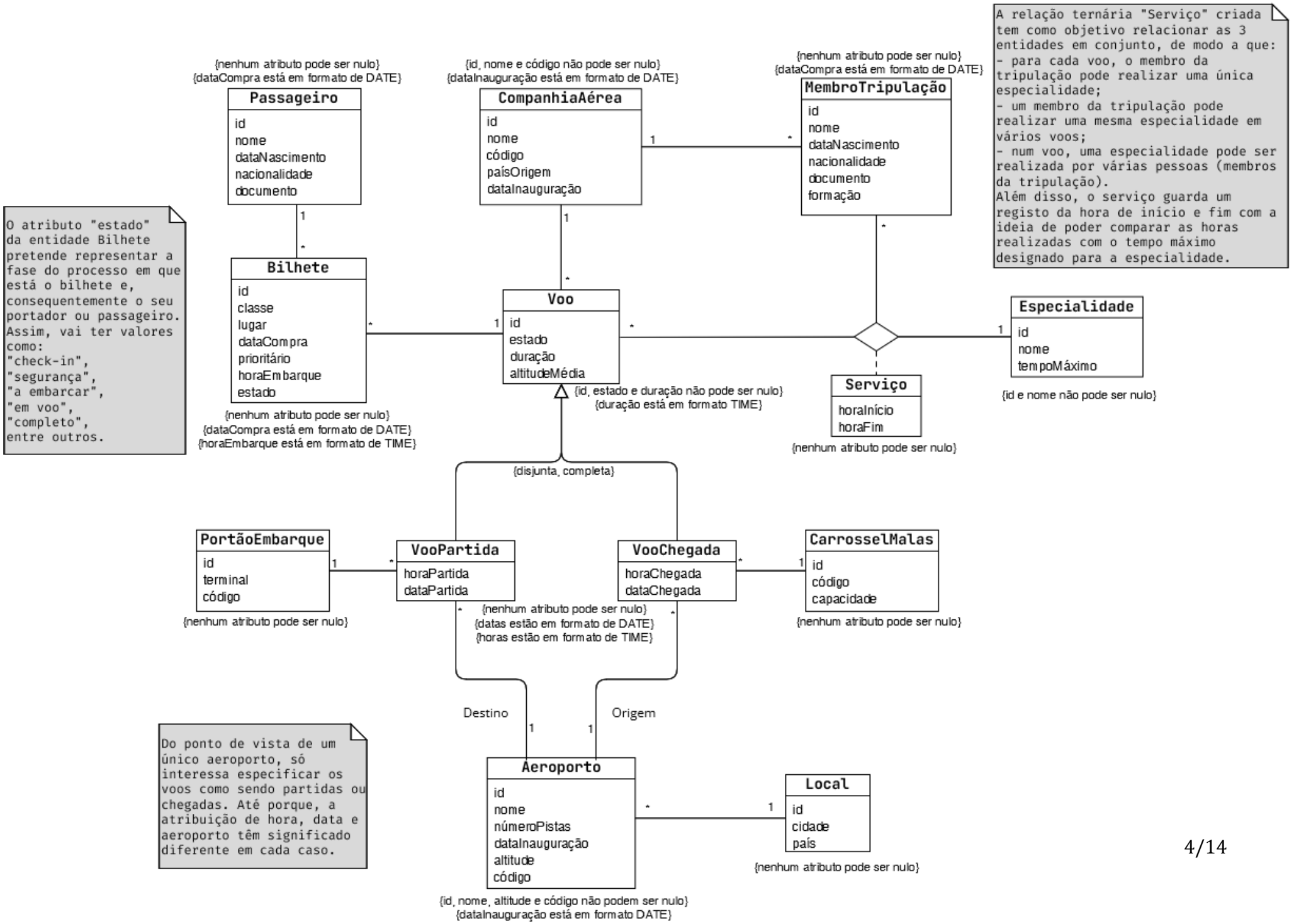
No final, devemos apresentar a solução final do esquema relacional e os ficheiros de SQL, de acordo com as nossas propostas iniciais e os ajustes feitos pela IA, avaliando criticamente este último em termos de vantagens e limitações.

Melhorias Pós-Avaliação

Relativamente à nossa proposta anterior, foram apontados 3 problemas a corrigir (cujas mudanças estão refletidas no esquema seguinte):

1. **Associação entre Membro Tripulação e Companhia Aérea**
   Embora fosse algo que tivéssemos considerado na proposta anterior, esta associação many-to-one não estava presente no diagrama entregue, por isso, incluímo-la agora.
2. **Necessidade de mais restrições aos dados**
   Perante este feedback, acrescentamos restrições à maioria das entidades, pelo menos quanto à possibilidade de valores nulos e valores em formatos de DATA e TEMPO.
3. **Classe Pessoa e subclasses Passageiro e Membro Tripulação**
   Quanto a estas entidades, para simplificar a relação e a criação dos atributos de Ids, decidimos remover a classe Pessoa e "isolar" cada uma das outras classes, tendo estas agora todos os atributos.

DIAGRAMA DO MODELO



Esquema Relacional

No que toca à passagem do modelo em UML anterior para esquema relacional, não surgiram dúvidas e tentamos manter um certo padrão durante a conversão. Nomeadamente, decidimos que todas as associações muitos-para-um (many-to-one) seriam mapeadas com recurso a uma chave estrangeira (foreign key); todas as foreign keys referem-se aos Ids (chaves primárias) das entidades associadas e têm uma abreviatura dessa entidade no nome.

Além disso, no que toca à Generalização de Voo em Partida e em Chegada recorremos ao estilo E/R lecionado nas aulas, ou seja, existe a entidade principal (Voo) e em cada uma das relações-filho (VooPartida e VooChegada) existe uma foreign key para o atributo id da relação-pai. No final, embora as relações filho tenham os seus atributos específicos e relações associadas, têm também chave estrangeira para a chave primária de Voo.

Quanto à Associação Ternária foi criada com recurso a uma nova relação (Serviço). Esta tem chaves estrangeiras para todas as relações associadas (Voo, Especialidade e Membro Tripulação) e a chave primária é constituída pelos ids das relações que têm 'muitos' (*) perto (a chave primária é constituída por duas chaves estrangeiras, a de Voo e a de MembroTripulação).

Passageiro(id, nome, dataNascimento, nacionalidade, documento)

CompanhiaAérea(id, nome, código, paísOrigem, dataInauguração)

Bilhete(id, classe, lugar, dataCompra, prioritário, horaEmbarque, estado, pid→Passageiro, vid→Voo)

Voo(id, estado, duração, altitudeMédia, cid→CcompanhiaAérea)

MembroTripulação(id, nome, dataNascimento, nacionalidade, documento, formação, cid→CompanhiaAérea)

Especialidade(id, nome, tempoMáximo)

Serviço(vid→Voo, tid→MembroTripulação, eid→Especialidade, horaInício, horaFim)

VooPartida(id, id→Voo, horaPartida, dataPartida, peid→PortaoEmbarque, aid→Aeroporto)

PortãoEmbarque(id, terminal, código)

VooChegada(id, id→Voo, horaChegada, dataChegada, cmid→CarrosselMalas, aid→Aeroporto)

CarrosselMalas(id, código, capacidade)

Aeroporto(id, nome, código, númeroPistas, dataInauguração, altitude, lid→Local)

Local(id, cidade, país)

Dependências Funcionais e Formas Normais

Passando agora às dependências funcionais, estas são bastante diretas de obter a partir do passo anterior, devido à forma como o trabalho foi estruturado. Uma vez que o trabalho teve uma fase inicial de planeamento e criação do diagrama em UML, as várias entidades já se encontram isoladas e separadas, de modo a que para todas elas, as relações funcionais relevantes são o Id determinar os restantes dos seus atributos:

Passageiro:	id → nome, dataNascimento, nacionalidade, documento
CompanhiaAérea:	id → nome, código, paísOrigem, dataInauguração
Bilhete:	id → classe, lugar, dataCompra, prioritário, horaEmbarque, estado
Voo:	id → estado, duração, altitudeMédia
MembroTripulação:	id → nome, dataNascimento, nacionalidade, documento, formação
Especialidade:	id → nome, tempoMáximo
VooPartida:	id → horaPartida, dataPartida
PortãoEmbarque:	id → terminal, código
VooChegada:	id → horaChegada, dataChegada
CarrosselMalas:	id → código, capacidade
Aeroporto:	id → nome, código, númeroPistas, dataInauguração, altitude
Local:	id → cidade, país

Para uma relação estar na Forma Normal Boyce-Codd, é necessário que para todas as dependências funcionais não triviais, o lado esquerdo seja uma superchave (inclui a chave primária) da relação. Para a Terceira Forma Normal, consideramos todos os casos anteriores e ainda acrescentamos aqueles em que os atributos do lado direito são parte de uma chave.

Assim sendo, temos que para uma relação estar numa ou em ambas as formas basta garantir que: para todas as suas dependências funcionais os atributos do lado esquerdo são uma chave ou superchave.

Como para todas as relações já vimos que as dependências funcionais partem das chaves primárias, concluímos que todas as entidades estão pelo menos numa das formas normais pedidas, não necessitando de correções ou decomposições das relações/entidades no esquema relacional.

Mais uma vez, dada a forma como este projeto foi criado e desenvolvido desde o princípio, seria de esperar que todas as relações tivessem dependências funcionais muito simples e dependentes dos atributos que correspondem à sua chave primária. Por essa mesma razão, já era expectável que não existissem infrações quanto à representação na Forma Normal Boyce-Codd e/ou na Terceira Forma Normal.

Criação da Base de Dados em SQL

Tendo em conta que o diagrama UML já está corrigido e já foi convertido para o modelo relacional, a tarefa de criar o ficheiro create1.sql está bastante facilitada.

Para cada relação considerada é criada uma tabela com esse nome a partir do comando "CREATE TABLE <nome da relação>(<atributos>);", sendo que os atributos devem estar separados por vírgulas e cada um deles deve estar associado a um tipo. Embora o SQLite não faça verificações quanto ao tipo de dados inserido em cada atributo, achamos um bom princípio especificá-los no ficheiro de criação da base de dados.

No nosso caso, os tipos usados foram VARCHAR(255) para os atributos de texto, INTEGER para os valores numéricos, DATE para datas (no formato 'YYYY-MM-DD') e TIME para horas (no formato 'HH:MM').

Em todas as declarações de tabelas, incluímos também as primary keys (chaves primárias) da relação através das palavras 'PRIMARY KEY' colocada à frente do atributo (nos casos em que a chave primária era constituída por apenas um atributo) e pela expressão 'PRIMARY KEY(<atributo1>, <atributo2>)' nos casos em que a chave primária era composta (constituída por mais do que um atributo).

Além das chaves primárias, incluímos também chaves estrangeiras para representar as associações entre relações usando os comandos '<atributo> REFERENCES <tabela>(<atributo>) ON UPDATE CASCADE', para que quando os valores fossem alterados serem atualizados em todas as tabelas associadas.

Após isso, incluímos também restrições para praticamente todos os atributos das tabelas, na maior parte dos casos através de 'NOT NULL'. Para alguns valores que devem estar sempre dentro de uma gama específica, acrescentamos 'CONSTRAINT <nome da restrição> CHECK (<atributo> in (<valores possíveis>))'. Exemplos disto são os atributos de estado e classe dentro da tabela Bilhete.

Finalmente, para garantir que não há problemas a criar a base de dados, no início do ficheiro são colocados comandos para apagar os dados das tabelas se estas existirem: 'DROP TABLE <nome da tabela> IF EXISTS;'.

Antes de concluir esta parte, vale a pena salientar dois aspetos. O primeiro é que algumas das ideias apresentadas aqui não constam do ficheiro creates1.sql pois na primeira versão do ficheiro estas de facto não existiam e foram acrescentadas (por nós e não por Inteligência Artificial) no ficheiro create2.sql.

O segundo é que nas tabelas de VooPartida e VooChegada foram usados dois atributos, um com tipo de DATE e outro TIME, sendo que estes podiam também surgir num mesmo atributo do formato DATETIME. Tendo em conta a forma como projetamos o diagrama UML, achamos mais correto continuar com os dois atributos.

Carregamento de dados

Quanto à inserção de dados na tabela, utilizamos vários comandos de 'INSERT INTO TABLE <nome da tabela> VALUES (<valores>);' no ficheiro populate1.sql.

Para as tabelas principais procuramos colocar, regra geral, entre 8-12 tuplos de valores e ao invés de usarmos dados totalmente fictícios, procuramos introduzir dados que fizessem algum sentido e pudessem ser, no nosso entender, realistas.

Tentamos também que os valores tivessem associações lógicas e se cruzassem para tornar os casos mais próximos da realidade e testarem melhor a base de dados, uma vez que a inteligência artificial tende a falhar na geração de dados que se referenciam entre si, assunto mais detalhado na análise crítica.

Para tornar o trabalho mais interessante e ser possível testá-lo com interrogações mais criativas e diferentes, consideramos estas restrições na nossa versão inicial da inserção de dados:

- Pelo menos 2 Locais com mais do que um Aeroporto associado
- Pelo menos 1 MembroTripulação que realiza Serviço em mais que uma Especialidade
- Pelo menos 1 MembroTripulação que também viaje como Passageiro (dados semelhantes, entidades diferentes)
- Pelo menos 2 Passageiros que tenham mais que um Bilhete
- Pelo menos 2 Voos (Partidas e Chegadas) para os mesmos Aeroportos
- Pelo menos 2 Voos (Partidas e Chegadas) da mesma CompanhiaAérea

Descrição do Modelo de Inteligência Artificial

No que diz respeito à integração de inteligência artificial neste projeto decidimos recorrer ao mesmo modelo usado na primeira fase do projeto, mas agora, apenas a partir de uma mesma plataforma. Dado que este já foi abordado em detalhe no relatório anterior, a descrição do modelo apresentado será mais sucinta.

O modelo de inteligência artificial que usamos foi o ChatGPT, um *chatbot* desenvolvido pela OpenAI (um dos principais laboratórios responsáveis pelos avanços da IA). Este é baseado na arquitetura GPT-3.5 (*Generative Pre-trained Transformer*), um modelo de linguagem (*Large Language Model*) que recorre a aprendizagem profunda (*Deep Learning*) para gerar respostas e conteúdo, baseado nos dados que lhe são fornecidos

A principal função destes modelos é então receber questões e textos humanos (*prompts*) sobre qualquer tópico e tentar responder, o mais corretamente possível, através da geração de explicações, recomendações e conteúdo em geral. No nosso projeto, este modelo foi o utilizado para obter uma possível solução a cada etapa, através de prompts semelhantes às especificadas na próxima secção.

Descrição das Questões Colocadas

Numa tentativa de melhorar as contribuições da inteligência artificial face à primeira entrega deste projeto, optamos por colocar questões ao modelo de 2 formas diferentes.

Inicialmente, procuramos manter uma conversa fluida. Para começar apresentamos a descrição do modelo anterior e fomos pedindo ajustes para o refinar até ficar igual ao que tínhamos projetado. A partir daí, colocamos perguntas consecutivas sobre os vários pontos que tínhamos de abordar:

1. Para a base de dados indicada, converte o modelo para um esquema relacional.
2. Lista as dependências funcionais para cada uma das relações.
3. Para cada relação, indica, se existirem as violações da Forma Normal Boyce-Codd/Terceira Forma Normal
4. Com base no esquema relacional, gera um ficheiro, em linguagem SQL, onde cries todas as relações. Inclui antes de cada tabela o comando DROP TABLE IF EXISTS.
5. Gera um ficheiro em SQL para popular a base de dados, considerando o ficheiro de criação das tabelas anterior

Depois, fomos intercalando esta conversa com perguntas específicas sobre as diferenças relativas às nossas respostas iniciais. Pedimos também para gerar uma resposta para cada passo baseado na nossa resposta ao modelo relacional ou no ficheiro de criação da base de dados e experimentamos expandir o ficheiro de inserções através de IA com base nos tuplos que já tínhamos criado.

Análise das Respostas Obtidas

Modelo Relacional e Dependências Funcionais

Começando pelo modelo relacional, foram necessárias várias tentativas até o output ter o formato desejado. Na resposta inicial, o modelo relacional veio apresentado por tópicos numa linguagem semelhante a SQL, pelo que pedimos para alterar a resposta para o esquema R1(atr1, atr2, atr3→R2).

Esta sugestão foi a primeira melhoria e funcionou, porém, o modelo não apresentava agora as chaves primárias para cada relação, logo, pedimos para representar as chaves primárias no formato _atr_.

Mais uma vez, esta sugestão aproximou-se do nosso resultado, o modelo, de facto, identificou todas as chaves primárias desta forma, porém também devolveu todas as chaves estrangeiras neste formato.

Este problema, um pouco mais grave no nosso caso devido às relações com chaves estrangeiras como chaves primárias, foi corrigido posteriormente após nova insistência.

No final, o resultado desta etapa foi bastante próximo do nosso, embora tivesse sido necessária alguma intervenção e insistência até lá chegar.

Passando às dependências funcionais, tal como já mencionei anteriormente, devido à forma como foi planeado o trabalho, estas não são difíceis de identificar e avaliar. Assim sendo, tal como nós identificamos as dependências funcionais de cada relação como dependentes apenas do id (caso geral) e, nalguns casos, nem sequer existirem dependências funcionais de atributos relevantes para identificar, o modelo de IA chegou à mesma conclusão.

Por causa disto, a resposta foi praticamente igual à nossa tirando a relação Serviço. Neste caso mais complexo, o modelo identificou todas as chaves estrangeiras como parte esquerda da DF, algo que, no nosso caso, não consideramos o mais acertado, tendo em conta a justificação para a decomposição da relação ternária já apresentada.

Finalmente, depois da definição das dependências funcionais, pedimos para avaliar as mesmas quanto à forma normal de Boyce-Codd e à terceira forma normal. Embora fossem questões colocadas em separado, o output foi avaliado conjuntamente, concluindo-se também que todas elas estavam em pelo menos uma destas (e a maioria em ambas).

Criação da Base de Dados

Na criação do ficheiro SQL, colocamos duas questões (uma para cada abordagem):

1) Criar um ficheiro SQL que modelasse a base de dados que já estava a ser trabalhada, incluindo os DROP TABLES;

2) Criar um ficheiro SQL que modelasse a base de dados dado o esquema relacional já obtido, incluindo os DROP TABLES;

Comparando as duas respostas com o nosso ficheiro, os 3 apresentam uma estrutura praticamente igual no que toca ao código e aos comandos, porém reparamos que a IA não colocou tantas restrições de NOT NULL ou CHECK como nós.

Este fenómeno é facilmente explicado uma vez que não tinham sido especificadas que restrições incluir para cada atributo, pelo que o modelo colocou estas restrições nos atributos principais para aquilo que sabia ser correto (nunca colocou 'CHECK's de atributos em enumerações por exemplo).

Além disto, a IA demonstrou ter algumas falhas a gerar respostas ou processar questões mais longas pois não apresentou alguns dos atributos e não criou algumas das chaves estrangeiras esperadas, mais notoriamente as chaves estrangeiras que relacionam VooPartida e VooChegada com Voo, essenciais para o projeto.

Este tipo de críticas vai um pouco de encontro ao que já tínhamos observado na fase anterior, questões maiores ou que necessitem de uma resposta maior aumentam a hipótese de surgirem alguns erros que precisam de ser analisados.

Como chamada de atenção, houve um aspeto que nos surpreendeu pela positiva já que, no geral, o modelo conseguiu modelar corretamente o esquema e, quando lhe pedimos para avaliar o nosso código para a criação do esquema em SQL, este apontou um erro semântico (que entretanto corrigimos) de um atributo que estava colocado na tabela errada e ainda enumerou vários aspetos que pareciam estar corretos mas aos quais devíamos prestar especial atenção no código (chaves estrangeiras, restrições colocadas, enumerações, tipos de variáveis escolhidos...).

Carregamento de Dados

Por fim, quanto à inserção de dados na tabela, começamos por tentar povoar a base de dados a partir de mensagens anteriores ou dadas apenas as tabelas, mas rapidamente percebemos que não era o melhor método pois a inteligência artificial recorreu, na maioria dos casos, a valores genéricos como "Passageiro1" ou "Aeroporto2".

Assim, decidimos que a melhor solução era fornecer o código das tabelas já povoadas e pedir para expandir as inserções com novos valores. Embora os novos dados não se encontrassem tão bem relacionados como os criados por nós manualmente (dado que esses foram escolhidos e analisado com mais detalhe), acabaram por apresentar um estilo e formato mais próximo dos nossos e foram incluídos no ficheiro final, em conjunto com os do ficheiro anterior.

Em tabelas, como CompanhiaAérea, Aeroporto, Passageiro, Local, entre outras, em que os dados a inserir são mais objetivos e não existem tantas relações, o modelo fez um trabalho excelente e gerou uma boa quantidade de tuplos num curto espaço de tempo. Já em tabelas como Voo, Bilhete ou Serviço em que existem mais referências e relações entre os dados, o modelo teve mais dificuldade e, por isso, alguns dos dados no ficheiro populate2.sql podem não fazer tanta lógica do ponto de vista do mundo real.

Ainda assim, dada a clara dificuldade associada a criar inserções para bases de dados mais complexas com várias associações e relações entre os dados, consideramos que esta foi uma das fases do trabalho todo em que a Inteligência Artificial mais brilhou por conseguir gerar quantidades enormes de dados, no formato desejado, de forma rápida e com um resultado proveitoso.

Visão Geral

Fazendo uma avaliação geral da utilização do modelo de inteligência artificial nesta segunda fase do projeto, facilmente vemos que o seu contributo foi muito superior ao anterior.

Na nossa opinião, o facto da natureza desta segunda entrega não ser tão abstrata como a de criação de um problema e uma forma de o modelar, mas sim mais exata e apoiada num modelo preciso e já definido facilitou a integração das sugestões do modelo de inteligência artificial nas respostas finais.

Na criação do modelo relacional e melhoria do mesmo com eliminação das dependências funcionais, a inteligência artificial gerou praticamente as mesmas respostas que nós, o que serve de validação e suporte às nossas decisões. Devido às semelhanças e ao facto de ser uma tarefa mais metódica, consideramos a nossa solução de modelo final como sendo igual à inicial.

No que toca à escrita do modelo em SQL e criação de dados, foi onde o modelo mais brilhou devido a serem tarefas mais objetivas.

Como já salientado, uma vez que a nossa versão inicial do ficheiro create.sql sofreu alterações feitas por nós posteriores ao uso do modelo, vamos incluir todas as mudanças em conjunto no ficheiro create2.sql.

No ficheiro de inserção de dados, , vamos manter os valores apresentados inicialmente no ficheiro populate1.sql pois são uma boa base de dados para realizar alguns testes e interrogações. No ficheiro populates2.sql, vamos acrescentar dados para a maioria das tabelas gerados pelo modelo de inteligência artificial (com intervenção mínima da nossa parte) permitindo comparar os dois e verificar a principal crítica já apontada.

Esquema Relacional Final

Passageiro(id, nome, dataNascimento, nacionalidade, documento)

CompanhiaAérea(id, nome, código, paísOrigem, dataInauguração)

Bilhete(id, classe, lugar, dataCompra, prioritário, horaEmbarque, estado, pid→Passageiro, vid→Voo)

Voo(id, estado, duração, altitudeMédia, cid→CcompanhiaAérea)

MembroTripulação(id, nome, dataNascimento, nacionalidade, documento, formação, cid→CompanhiaAérea)

Especialidade(id, nome, tempoMáximo)

Serviço(vid→Voo, tid→MembroTripulação, eid→Especialidade, horaInício, horaFim)

VooPartida(id, id→Voo, horaPartida, dataPartida, peid→PortaoEmbarque, aid→Aeroporto)

PortãoEmbarque(id, terminal, código)

VooChegada(id, id→Voo, horaChegada, dataChegada, cmid→CarrosselMalas, aid→Aeroporto)

CarrosselMalas(id, código, capacidade)

Aeroporto(id, nome, código, númeroPistas, dataInauguração, altitude, lid→Local)

Local(id, cidade, país)

Conclusão

Em conclusão, o modelo relacional da solução final (apresentado na página anterior), mantém-se o mesmo da solução inicial e os ficheiros de criação e povoamento da base de dados enviados juntamente com o relatório apresentam alguma evolução e melhorias introduzidas pelo modelo de inteligência artificial.

Tal como no trabalho anterior, ainda antes de integrar as ferramentas de inteligência artificial, tivemos o cuidado de criar e analisar as nossas soluções o melhor possível. Como já mencionado, este trabalho foi mais objetivo, daí as nossas respostas iniciais sofrerem apenas adições e poucas ou nenhumas alterações provenientes do modelo de IA a que recorremos.

Em contraste com a primeira entrega, consideramos muito mais útil a utilização de IA, especialmente ao nível da geração de código em SQL para as últimas fases. Os modelos de Inteligência Artificial têm revelado um aproveitamento enorme no que toca à explicação e criação de código em várias linguagens de programação e tudo indica que linguagens declarativas como SQL são algumas delas. Embora ainda seja recomendada a verificação e análise do output gerado por parte de alguém entendido no assunto, não é expectável serem necessárias tantas mudanças que justifiquem o não uso desta ferramenta.

De forma mais vincada do que no relatório anterior, consideramos ser pertinente a realização deste projeto, dado que nos permite ter mais contacto direto com as aplicações reais desta unidade curricular e criar uma base de dados realmente funcional e utilizável a partir de um modelo que nós mesmo desenvolvemos, algo claramente importante para o nosso futuro. A integração da inteligência artificial é ainda mais relevante nesta fase para compreendermos o seu poder como ferramenta de auxílio, não devendo ignorar a existência de limitações e cuidados a ter quando trabalhamos com ela.

Participação dos Membros e Divisão das Tarefas

De forma semelhante à fase anterior, podemos dizer que todos estivemos diretamente envolvidos no desenvolvimento do produto final. A correção inicial do modelo relativa ao feedback foi feita pelo Rodrigo, assim como a conversão para o modelo relacional e a identificação e decomposição de dependências funcionais.

A criação do ficheiro inicial de criação da base de dados em SQL e a revisão das respostas em SQL do modelo de IA foram feitas pelo João. A integração com a inteligência artificial foi realizada pelo Hugo, assim como a exploração e transmissão das principais diferenças que surgiam, por vezes, nalgumas das respostas obtidas.

Por fim, a criação do ficheiro inicial de inserção de dados, a integração das sugestões de IA sobre o código SQL e a elaboração do relatório para entrega foi realizada pelo Rodrigo. Todas as fases desta entrega foram ainda sendo trabalhadas e aperfeiçoadas por todos os membros do grupo.

Extra: Exemplos de Interrogações

Como extra, decidimos incluir algumas interrogações à base de dados em formato de SQL, ainda não treinamos esta parte nas aulas, pelo que podem existir otimizações para as mesmas, estas são apenas exemplos:

Voos de Chegada de Paris registados

SELECT aeroporto.nome as aeroporto, aeroporto.codigo as codigoA, companhiaAerea.nome as companhiaAerea, companhiaAerea.codigo as codigoC, voo.duracao, horaChegada, dataChegada
FROM aeroporto, local, companhiaAerea, voo, vooChegada
WHERE aeroporto.id=aeroportoid AND lid=local.id AND companhiaid=companhiaAerea.id AND voo.id=vooChegada.vooId AND local.nome='Paris';

Registo dos Serviços realizados

SELECT membroTripulacao.nome, nacionalidade, dataNascimento, especialidade.nome, horaInicio, horaFim, duração, estado
FROM membroTripulacao, especialidade, serviço, voo
WHERE vooid=voo.id AND tripulacaoid=membrotripulacao.id AND especialidadeid=especialidade.id;

Passageiros a passar a Segurança e Portão para onde devem ir

SELECT passageiro.nome, nacionalidade, portaoembarque.terminal, portaoembarque.codigo, classe, companhiaAerea.nome
FROM passageiro, bilhete, voo, vooPartida, portaoEmbarque, companhiaAerea
WHERE passageiroid=passageiro.id AND bilhete.vooid=voo.id AND vooPartida.vooid=voo.id AND portaoid=portaoembarque.id AND companhiaid=companhiaaerea.id AND bilhete.estado='Segurança';

Referências Utilizadas

https://online.visual-paradigm.com

https://pt.wikipedia.org/wiki/ChatGPT

https://pt.wikipedia.org/wiki/GPT-3

https://emvlab.org/mrz/

https://en.wikipedia.org/wiki/List_of_airports_by_IATA_airport_code

https://en.wikipedia.org/wiki/List_of_airline_codes

https://www.fakenamegenerator.com

https://chat.openai.com