U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Bases de Dados

Gestão de Aeroportos

Grupo 506 – 2MIEIC05

(27 de maio de 2018)

José Rodrigues **up201708806**@fe.up.pt

Luís Borges **up201605859**@fe.up.pt

Sandro Campos **up201605947**@fe.up.pt

Descrição

Este projeto baseia-se na gestão de um aeroporto e das suas partes integrantes principais inerentes ao seu funcionamento.

A base de dados armazena, sobre uma instância de **Aeroporto**, a informação que o representa e identifica, como a *sigla*, o seu *nome* e a *cidade* em que este se encontra.

Na relação com um aeroporto encontram-se presentes os seus **Funcionários**, as **Companhias Aéreas**, e os **Voos** que são operados por estas e que têm ligação com o respetivo aeroporto – isto é, os voos tanto podem descolar como aterrar nele.

Cada **Funcionário** tem um *cargo* e um *salário*, além de ser caraterizado por um *nome*, *NIF*, *data de nascimento*, *nacionalidade*, *morada*, *código postal* e ainda número de *telefone*, associada ao facto de este ser uma **Pessoa**.

Uma pessoa que não faça parte dos funcionários será um **Passageiro** que passa pelo aeroporto, tem um **Bilhete** com a informação sobre o *lugar* que ocupa no avião e o *tipo* de bilhete que possui para a viagem, que pode ser de classe económica, executiva ou primeira classe. Cada viajante tem a possibilidade de trazer consigo **Bagagens**, de *peso* e *dimensões* bem definidos.

Do voo interessa saber qual o seu *número*, as suas *horas de partida e de chegada*, a *lotação* a um determinado momento e a *data* na qual se realiza. Este é manobrado por 2 **Pilotos** (Piloto e Copiloto), entre os aeroportos de origem e destino.

Por fim e retratando a realidade, cada **Aeronave** é única, identificada pela sua *matrícula*, um **Modelo** e uma **Marca** que o produz. De forma a evitar inconsistências na informação, certifica-se que a *lotação* da aeronave terá que obrigatoriamente não exceder a *capacidade* da mesma.

Atributos

Pessoa
- Nome
- *NIF*
- *Data de Nascimento*
- Nacionalidade
- Morada
- Código Postal
- Telefone

Funcionário
- Cargo
- Salário

Passageiro
- (…)

Piloto
- (…)

Bilhete
- Tipo
- Lugar

Voo
- Número do Voo
- Hora de Partida
- Hora de Chegada
- Lotação
- Data

Bagagem
- Peso
- Dimensões

Aeroporto
- Sigla
- Nome
- Cidade

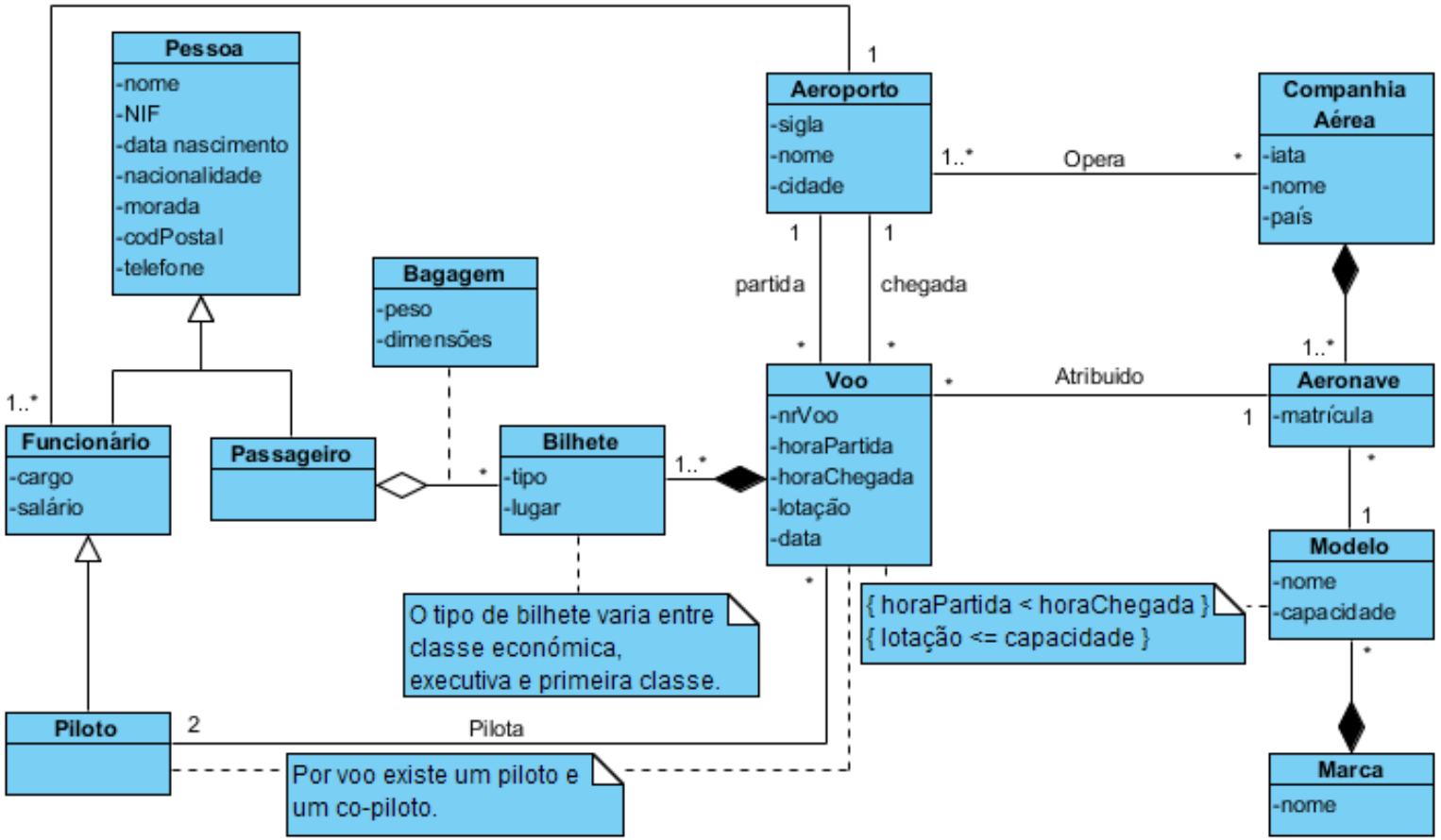
Companhia Aérea
- IATA
- Nome
- País

Aeronave
- Matrícula

Modelo
- Nome
- Capacidade

Marca
- Nome

Diagrama de Classes



Esquema Relacional e Dependências Funcionais

Aeroporto (sigla, nome, cidade)

- sigla -> {nome, cidade}

- {nome, cidade} -> sigla

. Assumindo que a sigla é um identificativo único do aeroporto e que, numa determinada cidade, não podem existir aeroportos com nomes em comum.

CompanhiaAerea (iata, nome, país)

- iata -> {nome, país}

- nome -> {iata, país}

. Considerando que cada companhia tem apenas um determinado país de origem e não existem companhias aéreas com nomes em comum.

Pessoa (nome, nif, dataNascimento, nacionalidade, morada, codPostal, telefone)

- nif -> {nome, dataNascimento, nacionalidade, morada, codPostal, telefone}

- telefone -> nif

. Considerando que cada pessoa, além de um nif, possui também um número de telefone único.

Funcionário (nif->Pessoa, cargo, salário, aeroporto->Aeroporto)

- {nif->Pessoa} -> {cargo, salário, aeroporto->Aeroporto}

Piloto (nif->Pessoa)

Passageiro (nif->Pessoa)

Voo (nrVoo, horaPartida, horaChegada, lotação, data)

- nrVoo -> {horaPartida, horaChegada, lotação, data}

Bilhete (tipo, lugar, nif->Pessoa, nrVoo->Voo)

- {lugar, nrVoo->Voo} -> {tipo, nif->Pessoa}

- {nif->Pessoa, nrVoo->Voo} -> {tipo, lugar}

. Considerando que a cada tipo de bilhete é reservado um conjunto de assentos no avião, separado dos outros restantes tipos.

Bagagem (lugar, nrVoo->Bilhete, peso, dimensões)

- {lugar, nrVoo->Bilhete} -> {peso, dimensões}

Aeronave (matricula, iata->CompanhiaAerea, modelo->Modelo)

- matrícula -> {iata->CompanhiaAerea, modelo->Modelo}

Modelo (nome, capacidade, marca->Marca)

- nome -> {capacidade, marca->Marca}

Marca (nome)

Opera (iata->CompanhiaAerea, aeroporto->Aeroporto)

Partida (nrVoo->Voo, sigla->Aeroporto)

- {nrVoo->Voo} -> {sigla->Aeroporto}

Chegada (nrVoo->Voo, sigla->Aeroporto)

- {nrVoo->Voo} -> {sigla->Aeroporto}

Pilota (nrVoo->Voo, nifPiloto->Pessoa, nifCoPiloto->Pessoa)

- {nrVoo->Voo} -> {nifPiloto->Pessoa, nifCoPiloto->Pessoa}

Atribuido (nrVoo->Voo, matricula->Aeronave)

- {nrVoo->Voo} -> {matricula->Aeronave}

Formas Normais

O modelo relacional apresentado cumpre a **BCNF** (Forma Normal de Boyce Codd) uma vez que todas as dependências funcionais não triviais acima possuem uma (super)chave à sua esquerda.

Acontece que, no entanto, não se encontra na **3NF** devido à transitividade existente entre dependências funcionais presentes na relação *Pessoa*. Observa-se que, ao se assumir um *telefone* único para cada pessoa, é possível obter o seu *nif* que por sua vez nos permite saber todos os outros atributos a ela referente. Este facto viola, por isso, o princípio necessário ao cumprimento desta regra.

Restrições

De forma a reforçar a consistência da base de dados e a assegurar uma maior correção dos dados nela inseridos, implementámos algumas restrições que considerámos relevantes.

Entre elas podemos encontrar restrições do tipo **PRIMARY KEY**, **FOREIGN KEY e REFERENCES**, que permitem declarar chaves primárias e estrangeiras, respetivamente, e **NOT NULL**, **UNIQUE** e **CHECK**, representando restrições de integridade referencial. Destas últimas, a primeira impossibilita que o atributo a ela associado seja nulo, ou seja inexistente, a segunda assegura que o atributo é único na relação, ainda que este não seja chave, e a última permite-nos verificar uma determinada condição entre um ou mais atributos.

A seguir apresentam-se, para cada relação, as suas utilizações.

Relação Pessoa:

. O *nome*, a *dataNascimento* e a *nacionalidade* não podem ser nulos - Restrição **NOT NULL**

. Não pode haver duas pessoas com o mesmo *nif*, nem este pode ser nulo - Restrição **PRIMARY KEY**

. Não pode haver duas pessoas com o mesmo *telefone* - Restrição **UNIQUE**

Relação Funcionário:

. Não pode haver dois funcionários com o mesmo *nif*, nem este pode ser nulo - Restrição **PRIMARY KEY REFERENCES Pessoa**

. O *cargo* e o *salário* não podem ser nulos - Restrição **NOT NULL**

. Um funcionário trabalha num e num só aeroporto – Restrições **REFERENCES Aeroporto** e **NOT NULL**

Relação Passageiro:

. Não pode haver dois passageiros com o mesmo *nif*, nem este pode ser nulo - Restrição **PRIMARY KEY REFERENCES Pessoa**

Relação Bilhete:

. O *tipo* varia entre "1" (primeira classe), "2" (classe executiva) ou "3" (classe económica) e não pode ser nulo - Restrições **CHECK (*tipo* = 1 or *tipo* = 2 or *tipo* = 3)** e **NOT NULL**

. Não pode haver dois voos com o mesmo *nrVoo*, nem este pode ser nulo - Restrição **REFERENCES Voo**

. Não pode haver dois passageiros com o mesmo *nif*, nem este pode ser nulo - Restrições **REFERENCES Passageiro** e **NOT NULL**

. Não pode haver dois bilhetes num mesmo voo com o mesmo *lugar* - Restrição **PRIMARY KEY(lugar, nrVoo)**

Relação Voo:

. Não pode haver dois voos com o mesmo *nrVoo* - Restrição **PRIMARY KEY**

. A *horaPartida* tem de ser anterior à *horaChegada*, e nenhuma destas pode ser nula - Restrições **CHECK (*horaChegada* < *horaPartida*)** e **NOT NULL**

. A *lotacao* tem que ser positiva – Restrições **CHECK(lotacao >= 0)** e **NOT NULL**

. A *data* não pode ser nula – Restição **NOT NULL**

Relação Aeroporto:

. Não pode haver dois aeroportos com a mesma *sigla* - Restrição **PRIMARY KEY**

Relação Companhia Aérea:

. Não pode haver duas companhias aéreas com o mesmo *iata* - Restrição **PRIMARY KEY**

Relação Aeronave:

. Não pode haver duas aeronaves com a mesma *matrícula* - Restrição **PRIMARY KEY**

. O *iata* da companhia a que a aeronave pertence não pode ser nulo – Restrições **REFERENCES CompanhiaAerea** e **NOT NULL**

. O *modelo* não pode ser nulo - Restrições **REFERENCES Modelo** e **NOT NULL**

Relação Modelo:

. Não pode haver dois modelos com o mesmo *nome* - Restrição **PRIMARY KEY**

. A *marca* de um modelo não pode ser nula – Restrições **REFERENCES Marca** e **NOT NULL**

. A *capacidade* tem que ser positiva – Restrições **CHECK(capacidade >= 0)** e **NOT NULL**

Relação Marca:

. Não pode haver duas marcas com o mesmo *nome* - Restrição **PRIMARY KEY**

Relação Piloto:

. Não pode haver dois pilotos com o mesmo *nif*, nem este pode ser nulo - Restrições **REFERENCES Funcionario** e **NOT NULL**

Relação Partida:

. Não pode haver dois voos com o mesmo *nrVoo*, nem este pode ser nulo – Restrição **PRIMARY KEY REFERENCES Voo**

. Não pode haver dois aeroportos com a mesma *sigla*, nem esta pode ser nula – Restrições **REFERENCES Aeroporto** e **NOT NULL**

Relação Chegada:

. Não pode haver dois voos com o mesmo *nrVoo*, nem este pode ser nulo – Restrição **PRIMARY KEY REFERENCES Voo**

. Não pode haver dois aeroportos com a mesma *sigla*, nem esta pode ser nula – Restrições **REFERENCES Aeroporto** e **NOT NULL**

Relação Bagagem:

. O *nrVoo* e o *lugar* não podem ser nulos – Restrição **NOT NULL**

. O peso tem que ser positivo – Restrições **CHECK(peso >= 0)** e **NOT NULL**

. Não pode haver duas bagagens com o mesmo *lugar* e *nrVoo*, ou seja, cada bagagem pertence a um e um só bilhete – Restrições **PRIMARY KEY(lugar, nrVoo)** e **FOREIGN KEY(lugar, nrVoo) REFERENCES Bilhete**

Relação Pilota:

. Não pode haver dois voos com o mesmo *nrVoo*, nem este pode ser nulo – Restrição **PRIMARY KEY REFERENCES Voo**

. Não pode haver dois pilotos com o mesmo *nif*, nem este pode ser nulo - Restrições **REFERENCES Piloto** e **NOT NULL**

Relação Opera:

. Não pode haver duas companhias aéreas com o mesmo *iata*, nem este pode ser nulo - Restrições **REFERENCES CompanhiaAerea** e **NOT NULL**

. Não pode haver dois aeroportos com a mesma *sigla*, nem esta pode ser nula - Restrições **REFERENCES Aeroporto** e **NOT NULL**

Relação Atribuido:

. Não pode haver dois voos com o mesmo *nrVoo*, nem este pode ser nulo – Restrição **PRIMARY KEY REFERENCES Voo**

. Não pode haver duas aeronaves com a mesma *matrícula*, nem esta pode ser nula – Restrições **REFERENCES Aeronave** e **NOT NULL**

Queries

Através do uso de interrogações a uma base de dados podemos obter informação acerca dos dados nela armazenados. Com esse objetivo elaborámos um conjunto de interrogações para a base de dados por nós criada, sendo estas pertinentes para verificarmos o seu bom funcionamento.

1. Aeronaves a operar voos no dia 12/04/2018: permite obter os **nrVoo** e **matrícula** de cada um dos voos a serem operados num determinado dia, neste caso 12/04/2018, a partir de qualquer aeroporto. Com esta interrogação podemos verificar a correspondência que existe entre os atributos nrVoo das relações Voo e Atribuido, recorrendo-se à operação **JOIN** entre as duas tabelas com o uso da primitiva **USING** associada a nrVoo.
2. Voos sem passageiros: permite selecionar os **voos** que ainda não tenham passageiros, isto é, voos cujos bilhetes ainda não tenham sido vendidos. Recorre-se, por isso, ao uso de uma *subquery* à relação **Bilhete**, que nos permita identificar os voos que não estejam nessa mesma relação.
3. Peso médio das bagagens com destino a Madrid: seleciona-se o **peso médio** das bagagens recorrendo à primitiva **AVG**, existente na linguagem SQL. Para isso efetua-se uma interrogação envolvendo as relações **Bagagem** e **Chegada**, tendo em atenção apenas os voos com término no aeroporto de Madrid, ou seja, com a sigla 'MAD'.
4. Nacionalidade predominante na profissão de piloto, e o número de pilotos respetivo: possibilita a obtenção da **nacionalidade** predominante nos pilotos registados na base de dados, e o seu **número**. Para esse efeito utilizou-se a primitiva **MAX**, também ela existente na linguagem SQL, associada a uma *subquery* envolvendo as relações **Funcionário** e **Pessoa**. Observe-se também a necessidade de utilizar **GROUP BY**, de forma a organizar os diversos pilotos com a mesma nacionalidade, agrupando-os.
5. Nomes e nif de todos os passageiros portugueses que partem dos aeroportos JFK e LAX: obtém os **nomes** e **nifs** de todos os passageiros portugueses presentes em voos com partida nos aeroportos JFK e LAX, dos aeroportos internacionais com

mais tráfego dos EUA. De forma a comparar as nacionalidades e as siglas dos aeroportos recorre-se à primitiva **LIKE**.

6. Nome e cargo de funcionários do aeroporto BCN: seleciona, **ordenados** por **cargo**, o **nome** e **cargo** dos funcionários do aeroporto BCN.
7. Nomes e nacionalidades de funcionários estrangeiros: permite a seleção dos **nomes** e **nacionalidades** de todos os funcionários cuja nacionalidade não é a portuguesa. Para isso fez-se uso do operador de conjuntos **EXCEPT** entre as relações resultantes de 2 *queries* distintas. A primeira, à esquerda, selecionando todos os funcionários registados, e a outra selecionando todos aqueles cuja nacionalidade é a portuguesa.
8. Companhias aéreas e o nº de aeronaves, ordenadas por marca: permite selecionar o **nº de aeronaves**, **ordenadas** por **marca**, para cada **companhia aérea**, recorrendo ao uso da primitiva **COUNT** existente na linguagem SQL.
9. Nº e hora de partida de todos os voos do dia 11/04/2018: possibilita a obtenção de uma grelha programática com todas as **partidas** para um determinado dia, neste caso 11/04/2018, de modo semelhante ao observado nos ecrãs dos terminais de embarque dos aeroportos. Cada partida é constituída por um **aeroporto**, uma **companhia aérea**, um **número de voo** e uma **hora de partida**.
10. Nome e morada de funcionários que trabalham na cidade onde habitam: seleciona os **nomes** e **moradas** do conjunto de funcionários que vivem e trabalham na mesma cidade, **ordenados** por **nome**.

Triggers

De forma a garantir uma boa monitorização e manutenção da base de dados, implementaram-se ainda um conjunto de gatilhos que são ativados aquando da ocorrência de alterações na mesma. Estes gatilhos são, por isso, específicos para cada caso e encontram-se associados a operações de **INSERT**, **UPDATE** ou **DELETE** nas relações existentes.

1. VerificaAeronave**:** gatilho para verificação de **aeronave existente** na base de dados, aquando de uma tentativa de **inserção** de uma outra com a mesma matrícula (*BEFORE INSERT*). No caso de a aeronave ser existente o gatilho **ignora** a operação de inserção.
2. ApagaVoo: gatilho para operação de *CASCADE ON DELETE* aquando da remoção de um voo da base de dados (*AFTER DELETE*). Por consequência, todas as referências a esse voo são igualmente apagadas, como é o caso dos seus bilhetes, bagagens, ou da atribuição dos pilotos a esse mesmo voo.
3. VerificaHora: gatilho associado à implementação de uma **restrição** existente na relação **Voo** (a hora de partida de um voo ser obrigatoriamente menor que a hora de chegada desse mesmo voo). Dessa forma, numa tentativa de *UPDATE* ao atributo *horaPartida* de um tuplo dessa relação, é verificado o cumprimento dessa restrição. Caso a restrição seja violada a operação **aborta**, mostrando uma mensagem de erro **adequada**: 'Restrição horaPartida < horaChegada violada!'.