**Gestão de Aeroporto**

QuAIRy *- Flying with class*

Bases de Dados

Ano Letivo 2018/2019

Turma 6 Grupo 6

Daniel Ferreira Brandão, up201705812, up201705812@fe.up.pt

Henrique José Santos, up201706898, up201706898@fe.up.pt

Pedro Miguel Moás, up201705208, up201705208@fe.up.pt

17 de março de 2019

Índice

[**Índice** 2](#_Toc4953793)

[Tema do trabalho 3](#_Toc4953794)

[Implementação 4](#_Toc4953795)

[Modelo Conceptual 6](#_Toc4953796)

[Esquema relacional e Dependências Funcionais 7](#_Toc4953797)

[Formas Normais 9](#_Toc4953798)

[Restrições 10](#_Toc4953799)

[Person 10](#_Toc4953800)

[Employee 10](#_Toc4953801)

[IsBoss 10](#_Toc4953802)

[Passenger 10](#_Toc4953803)

[Country 10](#_Toc4953804)

[City 11](#_Toc4953805)

[Airport 11](#_Toc4953806)

[Trip 11](#_Toc4953807)

[Departure 11](#_Toc4953808)

[Arrival 12](#_Toc4953809)

[Class 12](#_Toc4953810)

[Ticket 13](#_Toc4953811)

[Luggage 13](#_Toc4953812)

[Airplane 13](#_Toc4953813)

[Airline 13](#_Toc4953814)

[AirplaneModel 14](#_Toc4953815)

[Workplace 14](#_Toc4953816)

[Gate 14](#_Toc4953817)

[Runway 14](#_Toc4953818)

[Desk 14](#_Toc4953819)

[CheckInDesk 15](#_Toc4953820)

[HelpDesk 15](#_Toc4953821)

[HasCheckInDesk 15](#_Toc4953822)

[LuggageBelt 15](#_Toc4953823)

Tema do trabalho

Este projeto tem como objetivo criar uma base de dados que conterá a informação necessária para gerir um aeroporto. Isto inclui os seus funcionários e passageiros, como também informações sobre as viagens que chegam e saem do respetivo aeroporto, para que seja possível geri-las.

Implementação

Começando pela classe **Person**, sobre esta interessa saber informações básicas da pessoa, como SSN, nome, data de nascimento e número de telefone, mas para detalhes mais complexos criaram-se especializações desta, as classes **Employee**, que guarda o seu salário e NIF, e **Passenger**, que apenas contém o seu número de identificação, no caso de haver problemas de verificação.

A parte central é, naturalmente, a classe **Trip**, que guarda a data/hora de partida, data/hora de chegada e duração. A data de chegada poderia ser determinada através dos outros atributos, mas dependeria da diferença de fuso horário entre aeroportos, sendo necessário aceder a várias outras classes sempre que fosse preciso calcular esse atributo. A classe **Trip** é uma generalização de **Departure** e **Arrival**, que ajudarão a separar as viagens para melhor clareza.

Naturalmente, uma **Trip** tem um **Airplane** associado (que por sua vez pode ter várias viagens), cuja classe guarda o seu nome. Um **Airplane** pertencerá a uma **Airline** (que, certamente, poderá ter vários aviões), e está ligado a uma classe **AirplaneModel**, que guarda todos os modelos de aviões conhecidos, para não repetir informação no caso de haver vários com o mesmo modelos. Para além do nome do modelo, é mantido o número de filas no avião, assim como o número de lugares por fila, possivelmente útil para a atribuição de lugares. Por fim, as viagens têm sempre uma ligação com a classe **Airport**, que guarda um código de identificação (sigla) e um nome.

Continuando, também é importante que haja uma gestão das bagagens que passam pelo aeroporto. Assim, uma **Trip** pode estar ligada a mais do que uma **Luggage**. Sobre esta última classe apenas poderá interessar saber o seu peso. Naturalmente, esta classe não corresponde ao objeto físico em si, pois se um passageiro trouxer a mesma mala para dois voos diferentes, certamente que o Aeroporto guardará duas instâncias diferentes. Esta classe tem sempre um passageiro associado, que pode ter outras malas associadas. A classe **Arrival** está ainda sempre associada a um **LuggageBelt**, onde serão despachadas as suas malas, a uma hora de recolha guardada na classe de associação **LuggageDropoff**. Como seria expectável, estes podem estar ligados a várias viagens.

Para unir as classes **Trip** e **Passenger**, criou-se uma classe **Ticket**. Esta contém o lugar do passageiro, assim como informação sobre se ele já fez check-in, se já entrou na zona de embarque e se já embarcou. Estes últimos três atributos apenas são relevantes nas viagens de partida. **Ticket** também se associa com uma classe **Class**, que apenas tem um nome (“Primeira Classe”, “Executiva”, etc.). Deste modo, um bilhete tem um único passageiro e uma única viagem, fazendo então a ligação entre **Trip** e **Passenger**.

Agora, relativamente à classe **Funcionário**, este está ligado a um **Workplace**, que é uma generalização de **Runway, Gate** (guarda booleano “*isBoardingGate*”), **LuggageBelt**, e **Desk**. Esta última é uma generalização (completa e disjunta) de **HelpDesk** e **CheckInDesk**. A classe **Workplace** é uma generalização incompleta e disjunta, pois existem outros tipos de locais de trabalho, mas cujos detalhes não são tão cruciais. No entanto, esses funcionários ficam registados na mesma, por uma questão de fiabilidade e consistência. Sobre as classes **Runway** e **Gate**, ambas podem ter várias viagens associadas, mas uma **Trip** apenas tem uma de cada. Por exemplo, no caso de partida do aeroporto, uma viagem está associada a uma Porta de Embarque. Um **CheckInDesk** está ligado a uma ou mais companhias aéreas, e estas podem ter vários balcões também. Um **Employee** pode ainda ter um chefe, ou ser chefe de vários funcionários.

Por fim, existem ainda classes **Country** e **City**, que são usadas para não repetir informação em muitas classes desnecessariamente. Uma pessoa tem uma nacionalidade, e um aeroporto pertence a uma cidade. Naturalmente que uma cidade pertence a um país, e que um país pode conter uma ou mais cidades.

Modelo Conceptual

O seguinte diagrama, que contém as diversas classes, atributos e associações, também se encontra anexado em formato pdf.

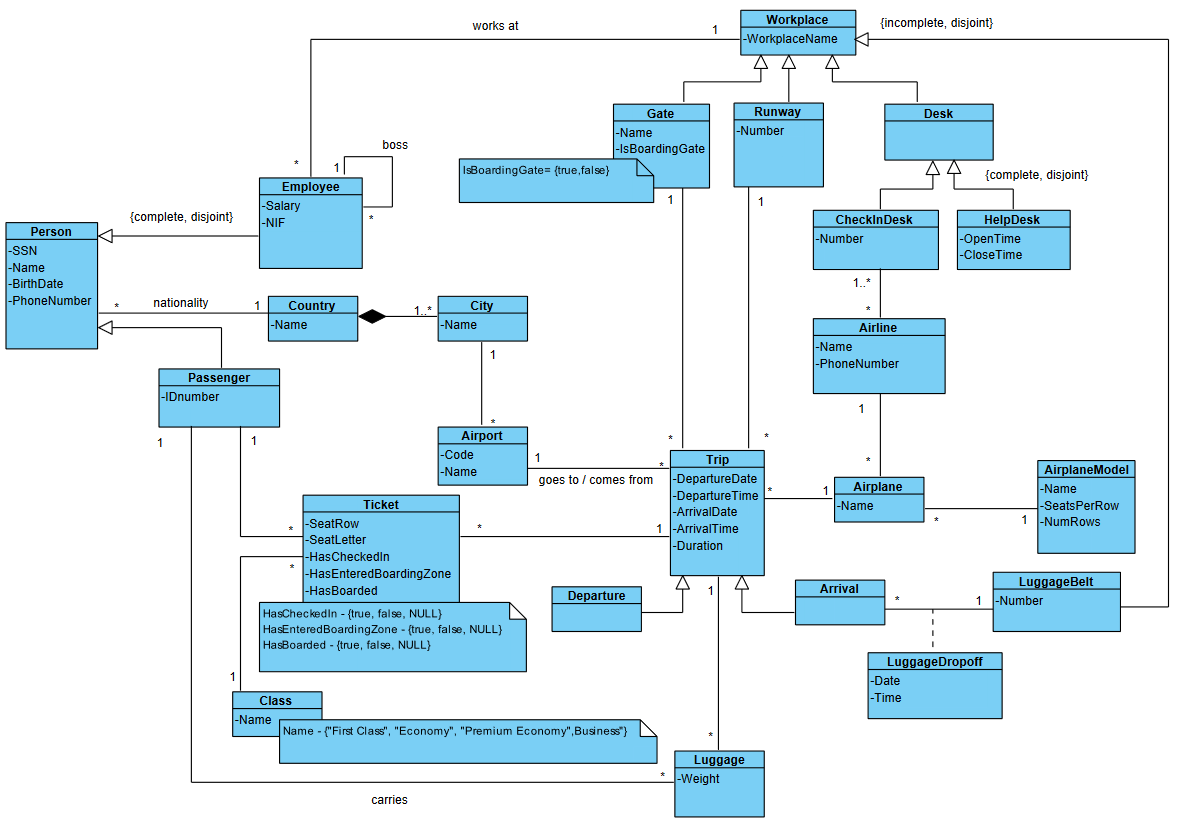


Figura 1 - Diagrama de classes UML

Esquema relacional e Dependências Funcionais

**Person** (PersonID, SSN, PersonName, BirthDate, PhoneNumber, Country->Country)

1. {PersonID} -> {SSN, PersonName, BirthDate, PhoneNumber, Country }
2. {SSN} -> {PersonID}
3. {PhoneNumber} -> {PersonID}

**Employee** (Person->Person, Salary, NIF, Workplace->Workplace)

1. {Person} -> {Salary , NIF , Workplace }
2. {NIF} -> {Person}

**Passenger** (Person->Person, IDnumber)

1. {Person}->{IDnumber}
2. {IDnumber}->{Person}

**IsBoss** (BossID->Employee, BossedID->Employee)

1. {BossedID} -> {BossID}

**Country** (CountryID, CountryName)

1. {CountryID} -> {CountryName}

**City** (CityID, CityName, Country->Country)

1. {CityID} -> {CityName, Country}
2. {CityName, Country} -> {CityID}

**Airport** (AirportCode, City->City, AirportName)

1. {AirportCode}->{City, AirportName}
2. {City, AirportName}->{AirportCode}

**Trip**(TripID)

**Departure** (Trip->Trip, Airport->Airport, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, DurationMinutes, Gate->Gate, Runway->Runway, Airplane->Airplane)

1. {Trip} -> {Airport, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, DurationMinutes, Gate, Runway, Airplane}
2. {DepartureDate, DepartureTime, Runway} -> {Trip}
3. {DepartureDate, DepartureTime, Gate} -> {Trip}
4. {DepartureDate, DepartureTime, Airplane}->{Trip}

**Arrival** (Trip->Trip, Airport->Airport, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, DurationMinutes, Gate->Gate, Runway->Runway, Airplane->Airplane, LuggageBelt->LuggageBelt, DropoffDate, DropoffTime)

1. {Trip}-> {DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, Gate, Runway, Airplane, Airport, LuggageBelt, DropoffDate, DropoffTime}
2. {ArrivalDate, ArrivalTime, Runway} -> {Trip}
3. {DepartureDate, DepartureTime, Gate} -> {Trip}
4. {DepartureDate, DepartureTime, Airplane}->{Trip}

**Class** (ClassID, ClassName)

1. {ClassID} -> {ClassName}
2. {ClassName} -> {ClassID}

**Ticket** (Passenger->Passenger, Trip->Trip, SeatRow, SeatLetter, HasCheckedIn, HasBoarded, HasEnteredBoardingZone, Class->Class)

1. {Passenger, Trip}->{SeatRow, SeatLetter, HasCheckedIn, HasBoarded, HasEnteredBoardingZone, Class}
2. {SeatRow, SeatLetter, Trip}->{Passenger, Trip}

**Luggage** (LuggageID, Weight, Trip->Trip, Passenger->Passenger)

1. {LuggageID}->{Weight, Trip, Passenger}

**Airplane** (AirplaneID, AirplaneName, Airline->Airline, AirplaneModel->AirplaneModel)

1. {AirplaneID}->{AirplaneName, Airline->Airline, AirplaneModel->AirplaneModel}
2. {AirplaneName, Airline->Airline}->{AirplaneID}

**Airline** (AirlineID, AirlineName, PhoneNumber)

1. {AirplaneID} -> {AirlineName, PhoneNumber}
2. {PhoneNumber} -> {AirlineID}
3. {AirlineName} -> {AirlineID}

**AirplaneModel** (ModelID, ModelName, SeatsPerRow, NumRows)

1. {ModelID} -> {ModelName, SeatsPerRow, NumRows}

**Workplace** (WorkplaceID, WorkplaceName)

1. {WorkplaceID} -> {WorkplaceName}
2. {WorkplaceName} -> {WorkplaceID}

**Gate** (Workplace->Workplace, GateName, IsBoardingGate)

1. {Workplace}->{GateName, IsBoardingGate}
2. {GateName} -> {Workplace}

**Runway** (Workplace->Workplace, RunwayNum)

1. {Workplace} -> {RunwayNum}
2. {RunwayNum} -> {Workplace}

**Desk** (Workplace->Workplace)

**CheckInDesk** (Desk->Desk, CheckInNum)

1. {Desk}->{CheckInNum}
2. {CheckInNum}->{Desk}

**HelpDesk** (Desk->Desk, OpenTime, CloseTime)

1. {Desk}->{OpenTime,CloseTime}

**LuggageBelt** (Workplace->Workplace, BeltNum)

1. {Workplace} -> {BeltNum}
2. {BeltNum}->{Workplace}

**HasDesk** (Airline -> Airline, CheckInDesk->CheckInDesk)

Formas Normais

Analisando então as dependências funcionais listadas, podemos ver que, em praticamente todas as relações, existem um ou mais IDs que permite chegar a todos os outros atributos da relação, sendo esses IDs chaves das relações. Assim, para todas as relações com apenas uma dependência funcional listada, conseguimos concluir que estão na forma normal de Boyce-Codd, pois o lado esquerdo contém sempre o ID da relação, que é uma chave (no caso de Airport não guardamos um número mas sim um código único, o que terá o mesmo efeito).

Para as relações que têm mais do que uma dependência funcional, consegue-se observar que os atributos do lado esquerdo permitem inferir os tais IDs, que são chaves das relações, sendo trivial concluir que esses atributos são, por sua vez, chaves da relação.

Relativamente às relações sem dependências funcionais, a chave corresponderá a todos os atributos.

Concluindo, como para todas as dependências funcionais, os atributos do lado esquerdo correspondem a sempre chaves da relação respetiva, o modelo relacional está na Forma Normal de Boyce-Codd, o que implica que esteja também na 3ª Forma Normal, não havendo violações a nenhuma destas Formas Normais.

Restrições

Person

PersonID é a chave primária (PRIMARY KEY)

SSN e PhoneNumber são atributos únicos (UNIQUE)

BirthDate tem um comprimento obrigatório de 10 carateres (CHECK)

Country é uma chave estrangeira (REFERENCES) de CountryID da classe Country

Employee

PersonID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de PersonID da classe Person

Salary é um número positivo superior a 600 (CHECK) euros

WorkplaceID é uma chave estrangeira (REFERENCES) de WorkplaceID da classe Workplace

IsBoss

BossID e BossedID são chaves estrangeiras (REFERENCES) de PersonID da classe Employee

Um employee não pode ser chefe de si (CHECK) mesmo (BossID != BossedID)

Passenger

PersonID é a chave primária (PRIMARY KEY)

IDNumber é um atributo único (UNIQUE)

Country

CountryID é a chave primária (PRIMARY KEY)

City

CityID é a chave primária (PRIMARY KEY)

CountryID é uma chave estrangeira (REFERENCES) de CountryID da classe Country

Quando se apaga/altera um tuplo de Country (ON DELETE/ON UPDATE), essa ação é propagada nos tuplos de City que os referenciam (CASCADE).

Airport

AirportCode é uma chave primária (PRIMARY KEY) de comprimento obrigatório de 3 carateres (CHECK)

CityID é uma chave estrangeira (REFERENCES) de CityID da classe City

Trip

TripID é a chave primária (PRIMARY KEY)

Departure

TripID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de TripID da classe Trip

AirportCode é uma chave estrangeira (REFERENCES) de AirportCode da classe Airport

DepartureTime e ArrivalTime têm comprimento obrigatório de 5 carateres (CHECK)

DepartureDate e ArrivalDate têm comprimento obrigatório de 10 carateres (CHECK)

DurationHours é um número positivo (CHECK)

DurationMinutes é um número entre 0 e 59 (CHECK)

GateID é uma chave estrangeira (REFERENCES) de GateID da classe Gate

RunwayID é uma chave estrangeira (REFERENCES) de RunwayID da classe Runway

AirplaneID é uma chave estrangeira (REFERENCES) de AirplaneID da classe Airplane

Não podem existir dois Departures com a mesma DepartureDate, DepartureTime e usar o mesmo GateID, RunwayID ou AirplaneID (UNIQUE)

Pelo menos um de DurationHours ou DurationMinutes tem de ser positivo (CHECK)

Arrival

TripID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de TripID da classe Trip

AirportCode é uma chave estrangeira (REFERENCES) de AirportCode da classe Airport

DepartureTime e ArrivalTime têm comprimento obrigatório de 5 carateres (CHECK)

DepartureDate e ArrivalDate têm comprimento obrigatório de 10 carateres (CHECK)

DurationHours é um número positivo (CHECK)

DurationMinutes é um número entre 0 e 59 (CHECK)

GateID é uma chave estrangeira (REFERENCES) de GateID da classe Gate

RunwayID é uma chave estrangeira (REFERENCES) de RunwayID da classe Runway

AirplaneID é uma chave estrangeira (REFERENCES) de AirplaneID da classe Airplane

BeltID é uma chave estrangeira (REFERENCES) de WorkplaceID da classe LuggageBelt

Não podem existir dois Departures com a mesma DropoffDate, DropoffTime e usar o mesmo BeltID (UNIQUE)

Não podem existir dois Departures com a mesma DepartureDate, DepartureTime e usar o mesmo GateID, RunwayID ou AirplaneID (UNIQUE)

Pelo menos um de DurationHours ou DurationMinutes tem de ser positivo (CHECK)

Class

ClassID é a chave primária (PRIMARY KEY)

Ticket

SeatRow, SeatLetter e TripID são a chave primária (PRIMARY KEY)

SeatRow é um inteiro superior a 0 (CHECK)

SeatLetter tem comprimento de apenas um caractér (CHECK)

HasCheckedIn, HasEnteredBoardingZone e HasBoarded são booleanos e, como tal, podem assumir os valores 0 (falso), 1 (verdadeiro) ou NULL (caso seja um arrival) (CHECK)

ClassID é uma chave estrangeira (REFERENCES) de ClassID da classe Class

Luggage

LuggageID é a chave primária (PRIMARY KEY)

Weight é superior a 0 (CHECK)

TripID é uma chave estrangeira (REFERENCES) de TripID da classe Trip

PersonID é uma chave estrangeira (REFERENCES) de PersonID da classe Passanger

Airplane

AirplaneID é a chave primária (PRIMARY KEY)

AirlineID é uma chave estrangeira (REFERENCES)de AirlineID da classe Airline

ModelID é uma chave estrangeira (REFERENCES) de ModelID da classe Model

Airline

AirlineID é a chave primária (PRIMARY KEY)

Tanto AirlineName como PhoneNumber são únicos (UNIQUE)

AirplaneModel

ModelID é a chave primária (PRIMARY KEY)

SeatsPerRow e NumRows são inteiros superiores a 0 (CHECK)

Workplace

WorkplaceID é a chave primária (PRIMARY KEY)

WorkplaceName é único (UNIQUE)

Gate

WorkplaceID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Workplace  
GateName é único (UNIQUE)

IsBoardingGate é um booleano e, como tal, pode assumir os valores 0 (falso) e 1 (true) (CHECK)

Runway

WorkplaceID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Workplace

RunwayNum é único (UNIQUE)

Desk

WorkplaceID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Workplace

CheckInDesk

DeskID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Desk

CheckInName é único (UNIQUE)

HelpDesk

DeskID é a chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Desk

OpenTime é uma hora anterior a CloseTime (CHECK)

HasCheckInDesk

AirlineID e DeskID são a chave primária (PRIMARY KEY)

AirlineID é chave estrangeira (REFERENCES) de AirlineID da classe Airline

DeskID é chave estrangeira (REFERENCES) de DeskID da classe CheckInDesk

LuggageBelt

WorkplaceID é chave primária (PRIMARY KEY) e chave estrangeira (REFERENCES) de WorkplaceID da classe Workplace

BeltNum é único (UNIQUE)

**Nota:**  Os atributos classificados como chaves primárias (PRIMARY KEY) ou únicos (UNIQUE) não podem aparecer duas vezes na mesma tabela.