BASE DE DADOS COMPANHIA AÉREA

2LEIC09 - Grupo 901
3° Entrega
(25 de janeiro de 2022)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Grupo de Trabalho:
Diogo Babo **up202004950**
João Matos **up202006280**
João Oliveira **up202004407**

Índice

Contexto da Base de Dados

Uma companhia aérea pretende guardar numa base de dados o histórico relativo à informação dos seus aviões, voos, passageiros e funcionários

Sobre o avião sabe-se o tipo de aeronave, uma matrícula única, a periodicidade da manutenção, o número de lugares, o peso do avião e o peso máximo transportável. Quanto aos voos é possível saber os horários e datas previstas e efetivas, e o número de passageiros. Cada voo segue uma rota predefinida com pelo menos aeroportos de partida e chegada, podendo haver escalas. Os aeroportos são identificados pelo seu nome, não existindo dois aeroportos diferentes com o mesmo nome.

A companhia dispõe também de pilotos (pelo menos um por voo, no máximo 3 contando com co-pilotos), tripulação dos aviões e funcionários de limpeza. Sobre cada trabalhador sabe-se o nome, o salário, o género e a sua data de nascimento. Os funcionários de limpeza têm um período de trabalho noturno ou diurno, os tripulantes de cabine desempenham têm diferentes cargos ("Flight Attendants", "Senior Flight Attendants", "Pursers" ou "Onboard chefs") e sobre os pilotos conhece-se o seu número de horas de voo.

Para poderem voar, os passageiros cujo nome, género e data de nascimento são conhecidos, terão de ter adquirido um bilhete e podem escolher levar bagagem.

Deverá ser possível obter uma variedade de informação tal como tempos de atraso, bilhetes vendidos por voo, rotas mais frequentadas, aeroportos mais visitados, intervalo de preços de viagem preferidos pelos passageiros ,etc..

Diagrama UML (Antigo)

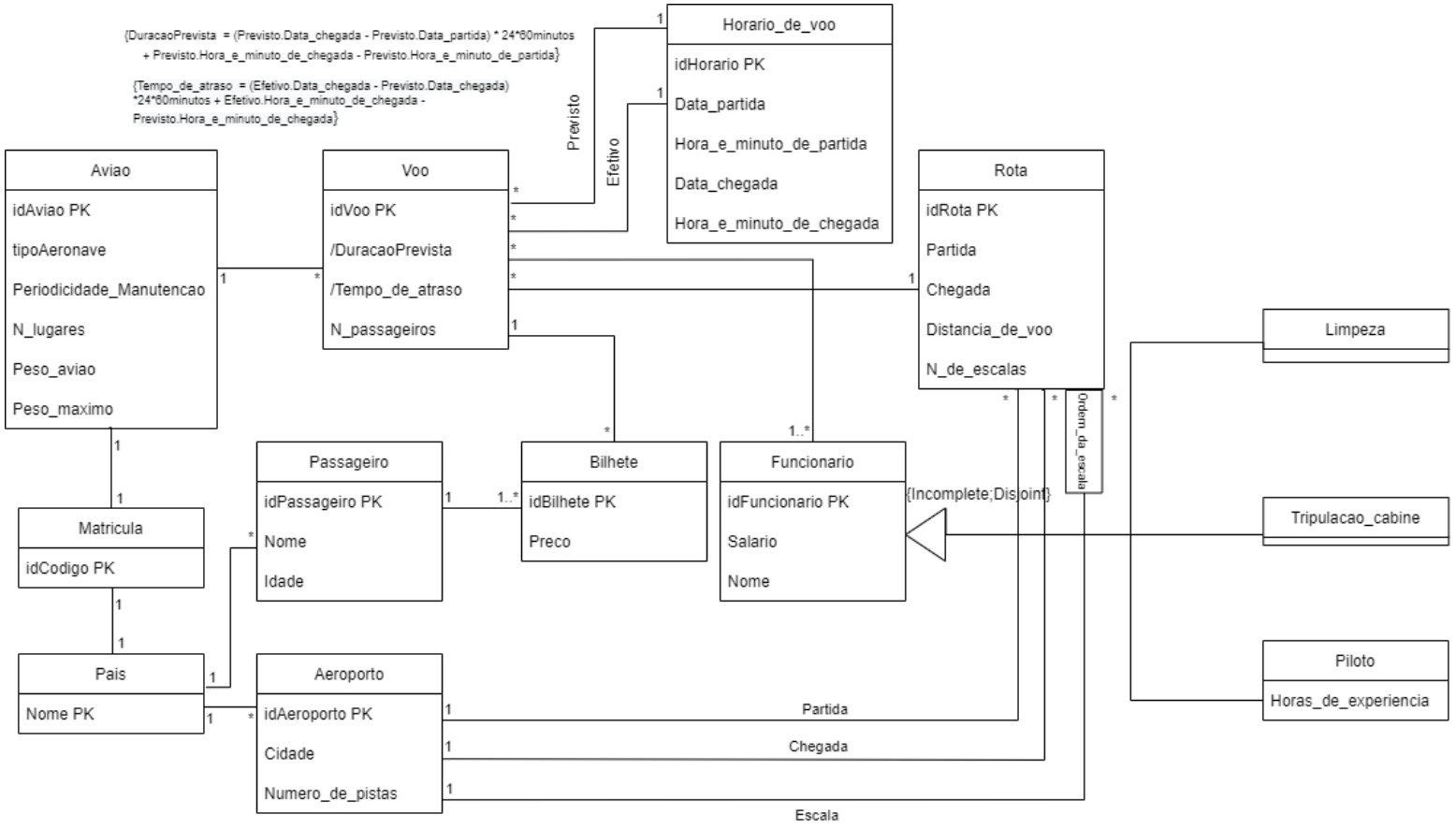
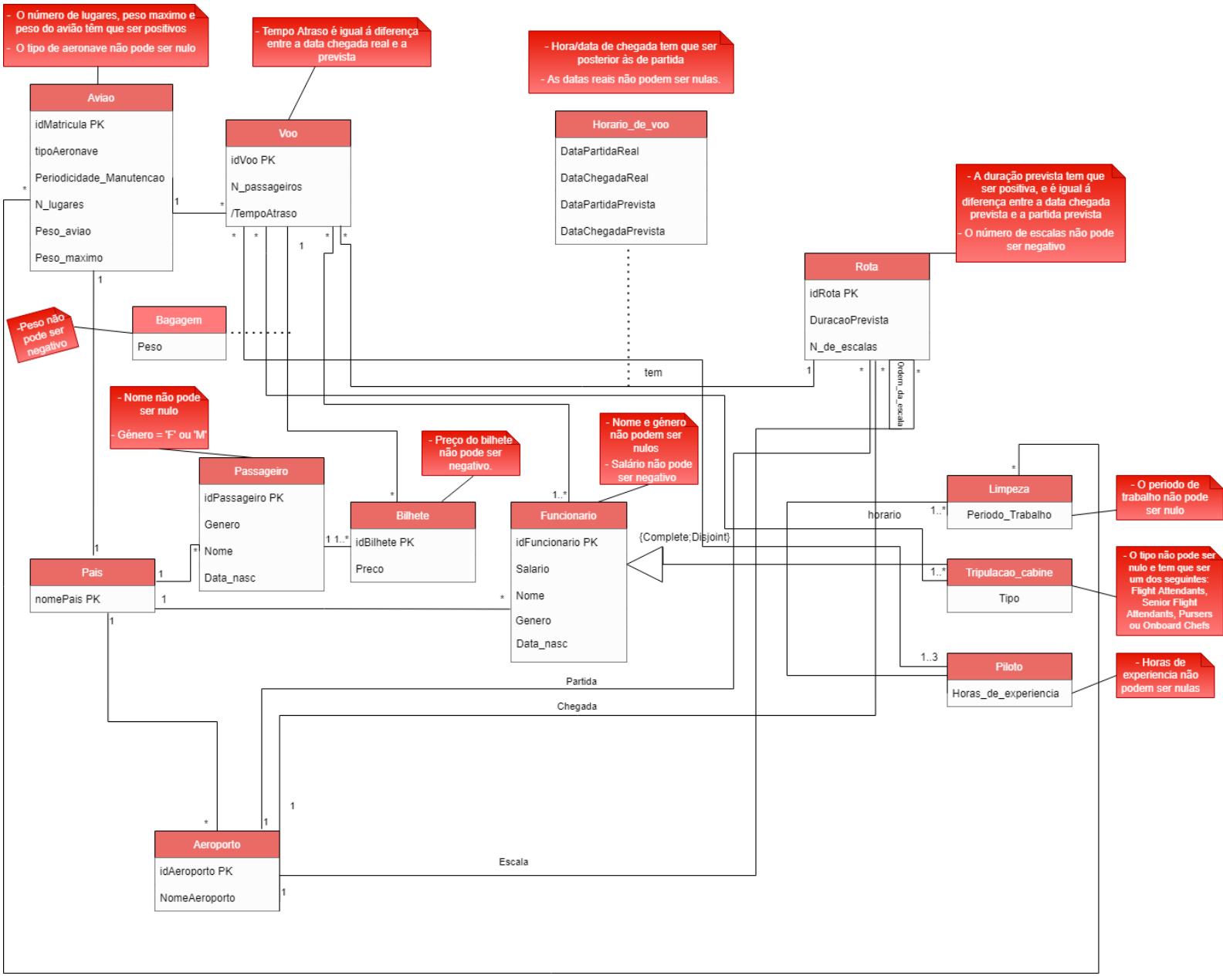


Diagrama UML (Revisto)



Esquema Relacional

- Aviao(idMatricula, tipoAeronave, PeriodicidadeManutencao, N_Lugares, PesoAviao. PesoMaximo, NomePais -> Pais)
- Pais(NomePais)
- Voo(idVoo, TempoAtraso, N_passageiros, idAviao -> Aviao)
- Rota(idRota, DuracaoPrevista, N_Escalas)
- Passageiro(idPassageiro, Nome, Data_nasc, Genero, NomePais -> Pais)
- Bilhete(idBilhete, Preco, idPassageiro -> Passageiro, idVoo -> Voo)
- Aeroporto(idAeroporto, NomeAeroporto, NomePais -> Pais)
- Horario_de_voo(idVoo -> Voo, idRota -> Rota, DataPartidaReal, DataChegadaReal, DataPartidaPrevista, DataChegadaPrevista)
- FuncionarioPiloto(idFuncionario, Nome, Salario, Genero, Data_nasc,NomePais -> Pais, Horas_de_experiencia)
- FuncionarioLimpeza(idFuncionario, Nome, Salario, Genero, Data_nasc, NomePais -> Pais, Periodo_de_trabalho)
- FuncionarioCabine(idFuncionario, Nome, Salario, Genero, Data_nasc, NomePais -> Pais, Tipo)
- Limpa(idFuncionario -> FuncionarioLimpeza, idMatricula -> Aviao)
- TripulantedoVoo(idFuncionario -> FuncionarioCabine, idVoo -> Voo)
- Pilota(idFuncionario -> FuncionarioPiloto, idVoo -> Voo)
- Partida(idRota -> Rota, idAeroporto -> Aeroporto)
- Chegada(idRota -> Rota, idAeroporto -> Aeroporto)
- Escala(idRota -> Rota, idAeroporto -> Aeroporto, Ordem_da_escala)
  {Rota, Ordem_da_escala} UK
- Bagagem(idVoo -> Voo, idBilhete -> Bilhete, Peso)

Análise de Dependências Funcionais

- Aviao(idMatricula, tipoAeronave, PeriodicidadeManutencao, N_Lugares, PesoAviao. PesoMaximo, NomePais -> Pais)

idMatricula -> TipoAeronave, PeriodicidadeManutencao, N_Lugares, PesoAviao. PesoMaximo, NomePais
TipoAeronave -> PeriodicidadeManutencao, N_Lugares, PesoAviao, PesoMaximo
BCNF: não
3NF: não
Decompondo a relação em várias, de modo a cumprir pelo menos uma das anteriores formas normais, obtemos as seguintes relações:
Aviao(idMatricula, NomePais, tipoAeronave -> TipoAviao)
TipoAviao(tipoAeronave, Periodicidade_Manutencao, N_Lugares, PesoAviao, Peso_maximo)

BCNF: sim
3NF: sim

- **Pais(NomePais)**
  3NF: sim
  BCNF: sim
  Nome é chave e a FD NomePais -> NomePais é trivial

- **Voo(idVoo, TempoAtraso, N_passageiros, idMatricula -> Aviao)**

  idVoo -> TempoAtraso, N_passageiros, idMatricula
  BCNF: sim
  3NF: sim

- **Rota(idRota,N_escalas, DuracaoPrevista)**

  idRota -> N_escalas, DuracaoPrevista
  BCNF: sim
  3NF: sim

- **Passageiro(idPassageiro, Nome, Data_nasc, Genero, NomePais -> Pais)**

  idPassageiro -> Nome, Data_nasc, Idade, Genero, NomePais
  BCNF: sim
  3NF: sim

- **Bilhete(idBilhete, Preco, idPassageiro -> Passageiro, idVoo -> Voo)**

  idBilhete -> Preco, idPassageiro, idVoo
  idPassageiro, idVoo -> idBilhete, Preco
  {idPassageiro, idBilhete} UK
  BCNF: sim
  3NF: sim

- Aeroporto(idAeroporto, NomeAeroporto, NomePais -> Pais)

  idAeroporto -> NomeAeroporto,NomePais
  NomeAeroporto -> NomePais
  BCNF: não
  3NF: não

  Decompondo a relação em várias, de modo a cumprir pelo menos uma das anteriores formas normais, obtemos as seguintes relações:

  Aeroporto(idAeroporto, NomeAeroporto -> LocalAeroporto)
  LocalAeroporto (NomeAeroporto, NomePais)

  BCNF: sim
  3NF: sim

- **Horario_de_voo(idVoo -> Voo, idRota -> Rota, DataPartidaReal,DataChegadaReal,DataPartidaPrevista,DataChegadaPrevista)**

  idVoo->idRota,DataPartidaReal,DataChegadaReal,DataPartidaPrevista,DataChegadaPrevista
  BCNF: sim
  3NF: sim

- **FuncionarioPiloto(idFuncionario, Nome, Salario, Genero, Data_nasc, NomePais -> Pais, HorasExperiencia)**

  idFuncionario -> Nome, Salario, Genero, Data_nasc, Nome, Horas_de_experiencia
  BCNF: sim
  3NF: sim

- **FuncionarioLimpeza(idFuncionario, Nome, Salario, Genero, Data_nasc, NomePais -> Pais,Periodo_de_trabalho)**

  idFuncionario -> Nome, Salario, Genero, Data_nasc, NomePais, Periodo_de_trabalho
  BCNF: sim
  3NF: sim

- **FuncionarioCabine(idFuncionario, Nome, Salario, Genero, Data_nasc, NomePais -> Pais, Tipo)**

  idFuncionario -> Nome, Salario, Genero, Data_nasc, NomePais,Tipo
  BCNF: sim
  3NF: sim

- **Limpa(idFuncionario -> FuncionarioLimpeza, idMatricula -> Aviao)**
  Não tem FD's não triviais
  BCNF: sim
  3NF: sim

- **TripulantedoVoo(idFuncionario -> FuncionarioCabine, idVoo -> Voo)**
  Não tem FD's não triviais
  BCNF: sim
  3NF: sim

- **Pilota(idFuncionario -> FuncionarioPiloto, idVoo -> Voo)**
  Não tem FD's não triviais
  BCNF: sim
  3NF: sim

- **Partida(idRota -> Rota, idAeroporto -> Aeroporto)**

  idRota -> idAeroporto
  BCNF: sim
  3NF: sim

- **Chegada(idRota -> Rota, idAeroporto -> Aeroporto)**

  idRota -> idAeroporto
  BCNF: sim
  3NF: sim

- **Escala(idRota -> Rota, idAeroporto -> Aeroporto, Ordem_da_escala)**
  {Rota, Ordem_da_escala} UK

  idRota, idAeroporto -> Ordem_da_escala
  idRota, Ordem_da_escala -> idAeroporto
  BCNF: sim
  3NF: sim

- **Bagagem(idVoo -> Voo, idBilhete -> Bilhete, Peso)**
  idBilhete -> idVoo, Peso
  BCNF: sim
  3NF: sim

Nos casos em que foi necessário decompor a relação inicial, tanto pelo algoritmo que permite obter relações que cumpram a terceira forma normal ou a forma normal de Boyce-Codd, obtiveram-se as mesmas decomposições, não existindo exemplos de relações que cumpram a 3NF e não cumpram a BCNF. Tal deve-se ao conjunto de dependências funcionais consideradas que são poucas e simples.

Restrições

- Todos os atributos que sejam uma **‘PRIMARY KEY’**, têm que ser únicos, ou seja, não pode haver duas instâncias da mesma classe com a mesma PK.
- Os atributos com a restrição **‘NOT NULL’**, implica que uma instância dessa tabela tem que ter obrigatoriamente esses atributos associados a um valor.

Aviao

- idMatricula **PRIMARY KEY**
- NomePais, TipoAeronave **- NOT NULL**
- O tipo de aeronave (**FOREIGN KEY**) deve corresponder a um tipo de aeronave de uma instância do TipoAviao
- O nomePais (**FOREIGN KEY**) deve corresponder a um nomePais de uma instância de País

TipoAviao

- TipoAeronave **PRIMARY KEY**
- O peso do avião tem que ser positivo
- O peso máximo do avião tem que ser positivo
- O número de lugares no avião tem que ser positivo
- O peso máximo do avião tem que ser maior que o peso do avião

Voo

- idVoo **PRIMARY KEY**
- idMatricula **- NOT NULL**
- O ID da matrícula (**FOREIGN KEY**) deve corresponder a um ID matricula de uma instância de Avião

Rota

- idRota **PRIMARY KEY**
- O número de escalas não pode ser negativo
- A duração prevista do voo tem que ser positiva

Passageiro

- idPassageiro **PRIMARY KEY**
- Nome, NomePais **- NOT NULL**
- O género de qualquer passageiro deve ser: ‘F’ para feminino ou ‘M’ para masculino

- O nomePais (**FOREIGN KEY**) deve corresponder a um nomePais de uma instância de País

Pais

- NomePais **PRIMARY KEY NOT NULL**

Bilhete

- idBilhete PRIMARY KEY
- idVoo, idPassageiro **- NOT NULL**
- O preço do bilhete não pode ser negativo
- O ID do passageiro deve corresponder a um ID passageiro de uma instância da tabela Passageiro
- O ID do voo (**FOREIGN KEY**) deve corresponder a um ID do Voo de uma instância de Voo
- Para cada par (idVoo,idPassageiro) existe apenas um bilhete

Aeroporto

- idAeroporto **PRIMARY KEY**
- NomeAeroporto **- NOT NULL**
- O NomeAeroporto (**FOREIGN KEY**) deve corresponder a NomeAeroporto de uma instância de LocalAeroporto

LocalAeroporto

- NomeAeroporto **PRIMARY KEY**
- NomePais **- NOT NULL**
- O nomePais (**FOREIGN KEY**) deve corresponder a um nomePais de uma instância de País

Horario_de_voo

- idVoo **PRIMARY KEY**
- idRota, DataPartidaReal, DataChegadaReal **- NOT NULL**
- O idRota (**FOREIGN KEY**) deve corresponder a um idRota de uma instância de Rota
- O idVoo (**FOREIGN KEY**) deve corresponder a um idVoo de uma instância de Voo
- As datas de chegadas têm que ser posteriores às de partida

Bagagem

- idBilhete **PRIMARY KEY**
- idVoo **- NOT NULL**
- O idBilhete (**FOREIGN KEY**) deve corresponder a um idBilhete de uma instância de Bilhete

- O idVoo (**FOREIGN KEY**) deve corresponder a um idVoo de uma instância de Voo
- O peso tem que ser positivo

Partida/Chegada

- idRota **PRIMARY KEY**
- idAeroporto **- NOT NULL**
- O idRota (**FOREIGN KEY**) deve corresponder a um idRota de uma instância de Rota
- O idAeroporto (**FOREIGN KEY**) deve corresponder a um idAeroporto de uma instância de Aeroporto

Escala

- (idRota,idAeroporto) **PRIMARY KEY**
- idRota,idAeroporto,Ordem_da_escala **- NOT NULL**
- O idRota (**FOREIGN KEY**) deve corresponder a um idRota de uma instância de Rota
- O idAeroporto (**FOREIGN KEY**) deve corresponder a um idAeroporto de uma instância de Aeroporto
- Para cada par (idRota,Ordem_da_escala) existe apenas uma escala

Funcionarios (de uma forma geral)

- idFuncionario **PRIMARY KEY**
- Nome, NomePais **- NOT NULL**
- O género de qualquer funcionário deve ser: ‘F’ para feminino ou ‘M’ para masculino
- O nomePais (**FOREIGN KEY**) deve corresponder a um nomePais de uma instância de País
- Salário não pode ser negativo

FuncionarioPiloto

- HorasExperiencia **- NOT NULL**
- As horas de experiência não podem ser negativas

FuncionarioCabine

- Tipo **- NOT NULL**
- O tipo de qualquer funcionário cabine deve ser: "Flight Attendants" ou "Senior Flight Attendants" ou "Pursers" ou "Onboard chefs"

FuncionarioLimpeza

- Periodo_de_trabalho **- NOT NULL**
- O periodo de trabalho deve ser: “diurno” ou “noturno”

Limpa

- idFuncionario, idMatricula **- NOT NULL**
- (idFuncionario,idMatricula) **PRIMARY KEY**
- O idFuncionario (**FOREIGN KEY**) deve corresponder a um idFuncionario de uma instância de FuncionarioLimpeza
- O idMatricula (**FOREIGN KEY**) deve corresponder a um idMatricula de uma instância de Aviao

TripulantedoVoo

- idFuncionario, idVoo **- NOT NULL**
- (idFuncionario,idVoo) **PRIMARY KEY**
- O idFuncionario (**FOREIGN KEY**) deve corresponder a um idFuncionario de uma instância de FuncionarioCabine
- O idVoo (**FOREIGN KEY**) deve corresponder a um idVoo de uma instância de Voo

Pilota

- idFuncionario, idVoo **- NOT NULL**
- (idFuncionario,idVoo) **PRIMARY KEY**
- O idFuncionario (**FOREIGN KEY**) deve corresponder a um idFuncionario de uma instância de FuncionarioPiloto
- O idVoo (**FOREIGN KEY**) deve corresponder a um idVoo de uma instância de Voo

Lista de Interrogações

Em algumas destas interrogações são usadas **views**, para facilitar o desenvolvimento da interrogação em si, tanto como a sua legibilidade. Além disso, tentamos dar preferência à eficiência minimizando o uso de sub-queries.

- **1)** Listagem de todos os funcionários do sexo feminino,da sua função e do seu salário em ordem decrescente.
- **2)** Listagem do número de funcionários masculinos, do seu nome e salário sendo que este é superior à média dos salários dos funcionários do sexo feminino.
- **3)** Listagem do tempo de atraso mínimo e máximo para os voos, assim como os respectivos aeroportos de partida/chegada. Ordenado em ordem crescente de tempo de atraso
- **4)** Listagem de todos os passageiros (nome, nacionalidade e género), assim como os dados dos respectivos voos, datas e aeroportos (partida e chegada).
- **5)** Listagem de todos os voos com data de partida entre 2021/11/30 e 2021/12/10, mas também das respectivas datas e países de partida e chegada.
- **6)** Listagem de todos os voos em que o aeroporto de Lisboa se encontra como escala, ou aeroporto de destino.
- **7)** Listagem do número de funcionários e da respectiva soma dos salários para cada voo, assim como a média do número de funcionários e de salários para cada voo.
- **8)** Listagem dos voos, aeroportos de partida/chegada e dos aviões responsáveis por estes, cujo peso máximo dos destes é superior a 20000.
- **9)** Listagem dos aeroportos e do número de vezes que estes são visitados em rotas, ordenados de forma decrescente por este.
- **10)** Listagem do peso total de bagagem por cada voo.

Lista de Triggers

Apesar de apenas serem necessários 3 **triggers**, ao desenvolvermos alguns triggers foram necessários criar outros uma vez que não era possível fazer tudo num único trigger (visto que os triggers têm o mesmo fim, estes devem ser interpretados como um único). Por exemplo, no desenvolvimento do trigger **1** para verificar as idades dos funcionários foi preciso criar diferentes triggers para os diferentes tipos de funcionários, assim como para as diferentes ações (inserir valores e dar update a estes). O mesmo acontece com o trigger **2** e com o trigger **3** (sendo que neste além de inserir valores e dar update, também permite a remoção destes).

- **1)** Quando é adicionado um funcionário verifica se este tem idade mínima de 18 anos, caso isto não se verifique lança um erro e impede a operação. Também impede que seja atualizada a data de nascimento se esta não respeitar a condição imposta em cima.
- **2)** Quando é inserido um horário de voo, o tempo de atraso para o respectivo voo é atualizado. Quando se dá update a um horário de voo, o novo tempo de atraso é calculado com base nos novos dados e por fim quando este é apagado, o tempo de atraso desse mesmo voo passa a ser **NULO**.
- **3)** Verifica ao inserir um bilhete se o avião já possui a capacidade lotada para um determinado voo, caso isto se verifique lança um erro e impede que seja adicionado um passageiro a esse voo. Caso contrário, adiciona o passageiro ao voo, e incrementa o número de lugares ocupados.