UNIVERSIDADE LUSÓFONA
de Humanidades e Tecnologias
Humani nihil alienum

Base de Dados para a Gestão de Aeroportos Nacionais

- Base de Dados I -

Eng. Informática D11 - DSAS

Elaborado Por:

2110 6069 Tadeu Bento

Lisboa, Sexta-feira, 22 de Março de 2013

Índice

Introdução

No âmbito da disciplina de Bases de Dados foi proposta a elaboração de um base de dados relacional, com aplicação numa situação real ou fictícia.

O objetivo principal deste trabalho é a consolidação de conhecimentos já abordados nas aulas. Além disso, este trabalho tem como objetivos complementares, estimular a autonomia dos alunos, a aprendizagem de metodologias de trabalho de projeto, estimular a expressividade, a criatividade e o espírito crítico.

Tema Escolhido

O tema escolhido escolhido foi a Gestão do Trafego Aéreo Nacional. O caso de estudo na qual se baseia a elaboração da base de dados é o seguinte:

> *Os colaboradores dos aeroportos de Portugal necessitam de um sistema de informação para a gestão do espaço aéreo, no qual deverá ser possível registar todos os aeroportos, aviões, companhias aéreas, rotas de voo, voos e bilhetes atribuídos.*
>
> *O sistema deverá ser capaz de prever situações impossíveis, por exemplo um avião transportar mais passageiros e/ou carga do que a sua capacidade para descolar, um avião ser atribuído a um voo para o qual não tem capacidade etc...*
>
> *Em termos de consulta deverá ser possível:*
>
> *1) Obter uma lista dos próximos voos a aterrar/descolar num aeroporto ou em todos;*
> *2) Obter lista de voos a decorrer;*
> *3) Obter lista de voos a decorrer por cada companhia aérea;*
> *4) Consultar todos os passageiros num determinado voo.*

Desenho e Planificação da Base de Dados

Para dar resposta às necessidades dos aeroportos concluí-se que são necessárias as seguintes tabelas com os objectivos descritos:

- **airline**: Armazena a informação relativa ás companhias aéreas que vão operar;
- **airplane**: Armazena os tipos de aviões disponíveis bem como as suas características;
- **airport**: Armazena todos os aeroportos;
- **constant**: Armazena constantes do sistema utilizadas para cálculos intermédio;
- **flight**: Armazena os voos bem como referências para outras tabelas;
- **route**: Armazena todas as rotas possíveis a tomar pelos voos, bem como características das mesmas;
- **ticket**: Armazena os bilhetes atribuindo-os aos voos.

Modelo Entidade Relação Simplificado

Agora que descrita a importância de cada tabela da base de dados, segue-se o modelo entidade relação simplificado:

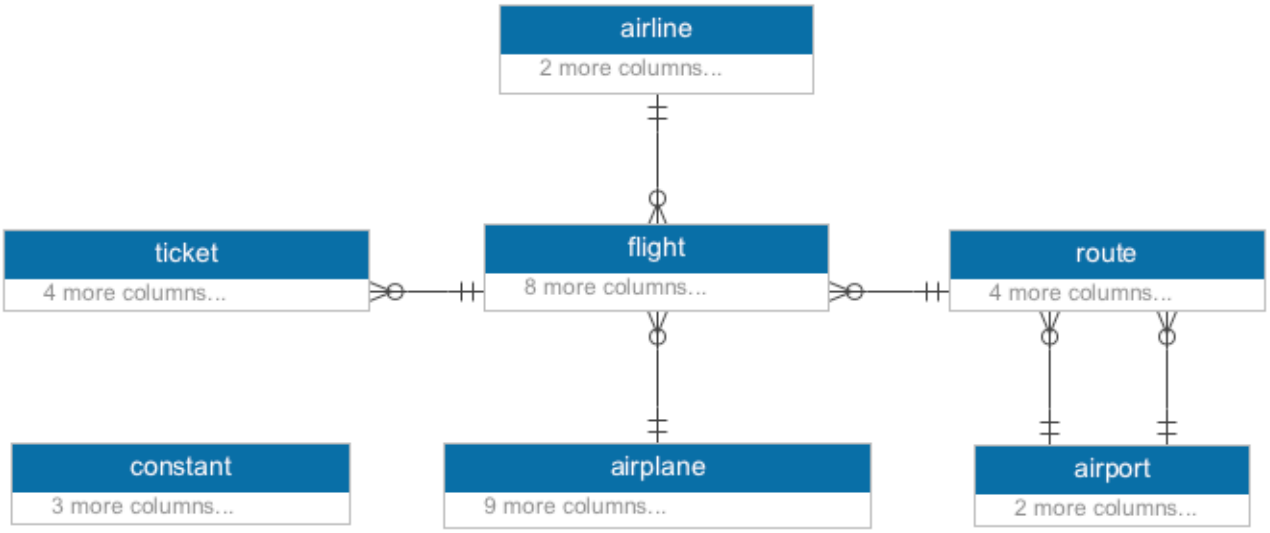

Fig 1. — Modelo entidade relação simplificado.

Descrição dos Campos

Cada tabela apresentada anteriormente, no modelo entidade relação simplificado, tem diversos campos, estes guardam informação específica da tabela. De seguida a descrição dos campos e a sua importância individual:

Tabela airline:

id_airline: Identificador único da companhia aérea (número inteiro);
name: Nome comercial da companha aérea (string).

Tabela airplane:

id_airplane: Identificador único do tipo de avião (número inteiro);
name: Nome comercial do avião (string);
passenger_capacity: Capacidade máxima de passageiros (número inteiro);
mtow: Peso máximo suportado para a descolagem (número inteiro);
oew: Peso máximo operacional vazio (número inteiro);
cruising_speed: Velocidade média do avião (número inteiro);
max_range: Alcance máximo do avião em MTOW (número inteiro);
max_fuel: Capacidade máxima de combustível em MTOW (número inteiro);
nmi_consumption: Consumo médio estimado por milha náutica (número decimal).

Tabela airport:

id_airport: Identificador internacional do aeroporto (3 caracteres);
name: Nome comercial do aeroporto (string);

Tabela constant:

id_constant: Identificador único da constante (número inteiro);
name: Nome único da constante (string);
value: Valor da constante (número decimal).

Tabela flight:

id_flight: Identificador único do voo (número inteiro);
id_airline: Identificador único da companhia aérea (número inteiro);
id_airplane: Identificador único do tipo de avião (número inteiro);
id_route: Identificador único da rota (número inteiro);
arrow: Identificador do sentido de voo aplicado á rota (varia entre original e inverso);

time_departure: Data de partida do voo;

time_offset: Desvio de tempo acrescentado ao voo, utilizado para criar atrasos de partida/*chegada*;

post_weight: Peso da carga de correio adicionada ao voo.

Tabela route:

id_route: Identificador único da rota (número inteiro);

id_origin: Identificador internacional do aeroporto de origem (3 caracteres);

id_destination: Identificador internacional do aeroporto de destino (3 caracteres);

distance: Distância percorrida pela rota (número inteiro).

Tabela ticket:

id_ticket: Identificador único do bilhete (número inteiro);

id_flight: Identificador único do voo (número inteiro);

id_passport: Identificador único do passaporte do passageiro (string);

luggage_units: Número de unidades de bagagem (número inteiro);

check_in: Define se o passageiro já fez ou não checkin (varia entre 0 e 1).

Modelo Entidade Relação Completo

Atentando às tabelas descritas e aos tipos de dados esperados em cada uma, foi então criado o seguinte modelo entidade relação completo:

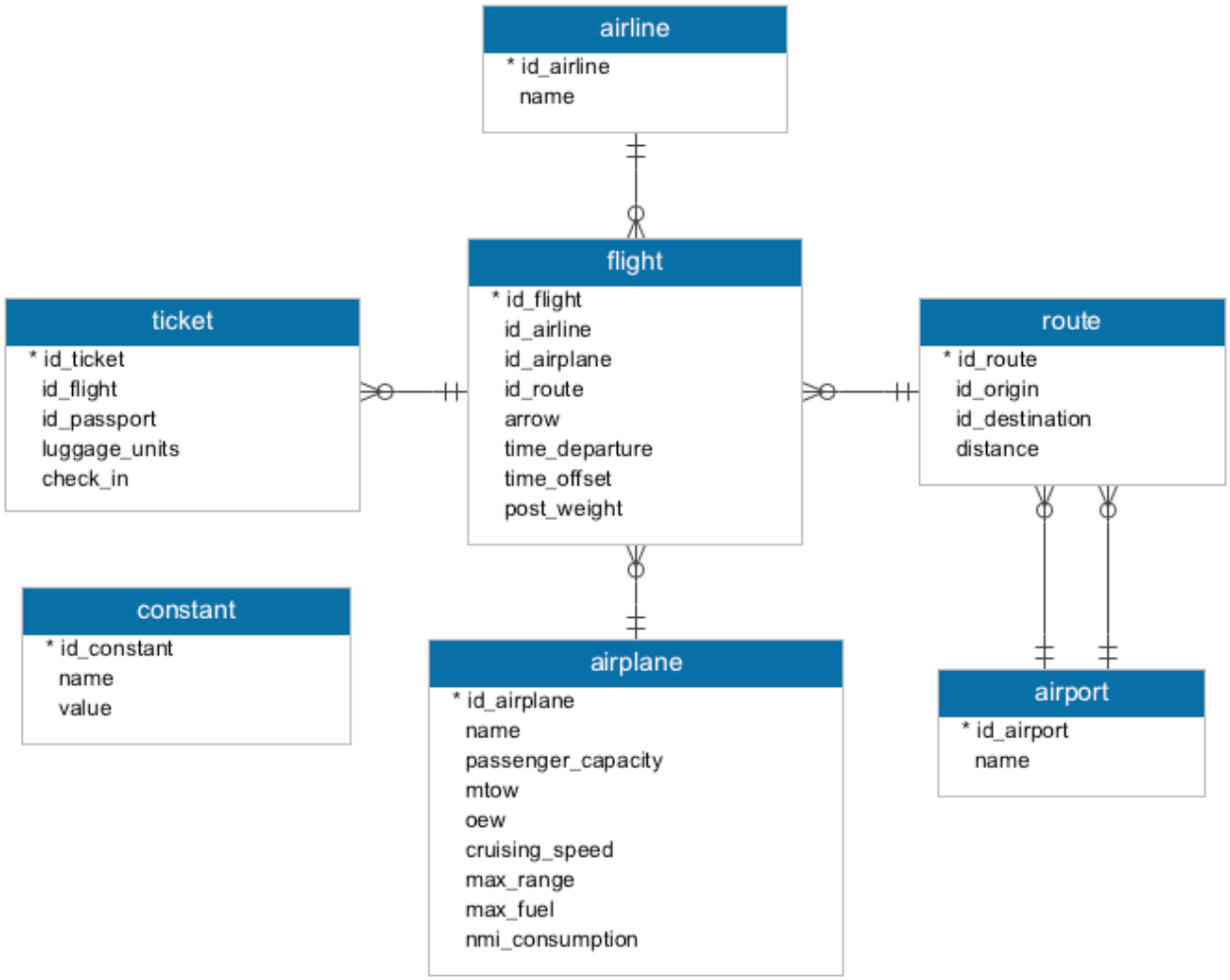

Fig 2. — Modelo entidade relação completo.

Definição dos Tipos de Dados

Tendo em conta os valores expectáveis em cada campo da base de dados, foram definidos os seguintes tipos de dados do MySQL (alguns apresentam constrangimentos):

Tabela airline:

id_airline: Identificador único da companhia aérea (número inteiro);
name: Nome comercial da companha aérea (string).

Tabela airplane:

id_airplane: INT UNSIGNED PRIMARY KEY AUTO_INCREMENT) NOT NULL;
name: VARCHAR(15)) NOT NULL UNIQUE;
passenger_capacity: INT(3) UNSIGNED) NOT NULL;
mtow: INT(6) UNSIGNED) NOT NULL;
oew: INT(6) UNSIGNED) NOT NULL;
cruising_speed: INT(4) UNSIGNED) NOT NULL;
max_range: INT(4) UNSIGNED) NOT NULL;
max_fuel: INT(6) UNSIGNED) NOT NULL;
nmi_consumption: FLOAT(10,2)) NOT NULL.

Tabela airport:

id_airport: VARCHAR(3) PRIMARY KEY NOT NULL;
name: VARCHAR(50) NOT NULL UNIQUE.

Tabela constant:

id_constant: INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL;
name: VARCHAR(15) NOT NULL UNIQUE;
value: FLOAT NOT NULL.

Tabela flight:

id_flight: INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL;
id_airline: INT UNSIGNED NOT NULL;
id_airplane: INT UNSIGNED NOT NULL;
id_route: INT UNSIGNED NOT NULL;
arrow: INT(1) NOT NULL;
time_departure: DATETIME NOT NULL;

time_offset: INT;
post_weight: INT UNSIGNED NOT NULL.

Tabela route:

id_route: INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL;
id_origin: VARCHAR(3) NOT NULL;
id_destination: VARCHAR(3) NOT NULL;
distance: FLOAT NOT NULL.

Tabela ticket:

id_ticket: INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
id_flight: INT UNSIGNED NOT NULL
id_passport: VARCHAR(9) NOT NULL,
luggage_units: INT(100) UNSIGNED NOT NULL,
check_in: INT(1) UNSIGNED NOT NULL,

Considerações para o entendimento do funcionamento da base de dados:

1) Todos os campos com nomes iguais em diversas tabelas devem ser relacionados como chaves estrangeiras, tal como representado no modelo completo entidade relação;
2) Os identificadores únicos de cada tabela foram definidos como sendo números inteiros apenas positivos uma vez que o sistema *AUTO_INCREMENT* inicia a contagem no zero. Maximizamos assim o número de entradas nas tabelas sem penalizar o espaço ocupado/ eficiência da base de dados.
3) O campo *time_offset* da tabela *flight* é um valor inteiro positivo ou negativo que representa o desvio horário do voo em relação ao esperado;
4) Não existem nenhum campo que represente a data de chegada porque isto pode ser calculado a partir da velocidade média do avião (*cruising_speed)*, data de partida (*time_departure)*, distância da rota (distance) e do desvio horário (*time_offset).*
5) As rotas não contemplam um sentido específico para o voo, sendo este definido por meio do campo *arrow* na tabela *flight*.

Verificações de Dados Introduzidos

Atentando aos requisitos levantados que especificam que esta terá se ser capaz de evitar a introdução de dados incorrectos, foram implementadas diversas verificações de dados ao longo da criação das tabelas.

> ***ATENÇÃO**: As verificações são implementadas recorrendo ao comando standard do SQL CHECK(); infelizmente este comando é ignorado por todo os motores do MySQL[1]. Para tirar partido desta funcionalidade será necessário usar um SGBD como Microsoft SQL Server ou Oracle.*

Tabela flight:

1) No campo id_route é necessário que o sistema se certifique a a rota escolhida pode ser tomada pelo avião atribuído por questões de distância:

   Distância da rota < Alcance máximo do avião;

Esta foi implementada pelo seguinte código na tabela:

```
id_route INT UNSIGNED NOT NULL
            CHECK ( (SELECT distance FROM route WHERE route.id_route =
flight.id_route) < (
                    (SELECT max_range FROM airplane WHERE airplane.id_airplane =
flight.id_airplane)
            )),
```

2) No campo *arrow* é necessário verificar se o sentido da rota é válida, variando esta entre 0 (original definida pela tabela *route*) ou 1 (inversa). Isto foi implementado pelo seguinte código:

```
arrow INT(1) NOT NULL DEFAULT 0 CHECK(arrow in (0,1)),
```

[1] http://dev.mysql.com/doc/refman/5.1/en/create-table.html

3) Para terminar, nesta tabela é necessário verificar se carga de correio incluída associada ao peso dos passageiros, sua bagagem e peso do combustível necessário para a rota não excedem o peso máximo de descolagem. Foi idealizada a seguinte verificação:

*MTOW > OEW + MaxPass * (AvgPass + DefLuggUnits * LuggUnitsMulti) + PostWeigh + JETLiterWeigh * RouteDistance * AvgConsump*

Onde:

MTOW: Peso máximo de descolagem. Disponível na tabela *airplane* por avião;
OEW: Peso operacional vazio. Disponível na tabela *airplane* por avião;
MaxPass: Número máximo de passageiros. Disponível na tabela *airplane* por avião;
AvgPass: Peso médio de um passageiro. Disponível na tabela *constant*;
DefLuggUnits: Número padrão de unidades de bagagem. Disponível na tabela *constant*;
LuggUnitsMulti: Multiplicador das unidades de bagagem. Disponível na tabela *constant*;
PostWeigh: Peso do correio associado ao voo. Disponível no campo *post_weight;*
JETLiterWeigh: Peso de cada litro de JET. Disponível na tabela *constant*;
RouteDistance: Distância da rota. Disponível na tabela *route* por rota;
AvgConsump: Consumo médio. Disponível na tabela *airplane* por avião.

Nota: Todas as unidades foram rectificadas em concordância para permitir os cálculos.

Esta verificação foi implementada pelo seguinte código na tabela:

```
post_weight INT UNSIGNED NOT NULL DEFAULT 0
            CHECK( (SELECT mtow FROM airplane WHERE airplane.id_airplane =
flight.id_airplane) > (
                  (SELECT oew FROM airplane WHERE airplane.id_airplane =
flight.id_airplane) +
                  (SELECT passenger_capacity FROM airplane WHERE
airplane.id_airplane = flight.id_airplane) *
                        (
                              (SELECT value FROM constant WHERE name = 'pas-
senger_weigh') + # Peso de cada passageiro
                              (SELECT value FROM constant WHERE name = 'de-
f_luggage_uni') * # Número de unidades de bagagem padrão
                              (SELECT value FROM constant WHERE name = 'lugga-
ge_unit_mu') # Multiplicador das unidades de bagagem
                        )
```

```
                    +
                    post_weight +
              (SELECT value FROM constant WHERE name = 'jet_liter_weigh') *
              (SELECT route FROM constant WHERE route.id_route = flight.id_route)
*
                    (SELECT nmi_consumption FROM airplane WHERE
airplane.id_airplane = flight.id_airplane)
              )),
```

Tabela ticket:

1) No campo id_flight é importante verificar se é ou não possível adicionar mais um bilhete ao voo referenciado pelo campo. Para tal foi idealizada a seguinte verificação:

   id_flight <= Capacidade de Passageiros do Avião

Esta verificação foi implementada pelo seguinte código na tabela:

```
id_flight INT UNSIGNED NOT NULL
              CHECK ( id_flight <=
                    (SELECT passenger_capacity FROM airplane WHERE
airplane.id_airplane =
                          (SELECT id_airplane FROM flight WHERE flight.id_flight
= ticket.id_flight ) )
              ),
```

Para além destas verificações/constrangimentos criados, na base de dados ainda são impostos outros pelo motor da mesma. Em todas as chaves estrangeiras os tipos de dados e valores introduzidos têm sempre de corresponder aos campos referenciados.

Utilização da Base de Dados

Após criada a base de dados, é necessário escrever as queries responsáveis por permitir ao sistema de informação introduzir e consultar informação disponível na base de dados.

A introdução de dados é realizada através do comando SQL INSERT(), e não será explicada aqui uma vez que é de simples compreensão e os exemplos incluídos no Anexo II já demonstram todas as operações necessárias.

No caso da consulta de informação já é necessário definir queries e/ou views na base de dados para o efeito. Entre as mais úteis para o caso em estudo temos:

1) Consulta de todos os voos:

```
SELECT flight.id_flight,
CASE flight.arrow
      WHEN '0' THEN (SELECT CONCAT_WS(',',route.id_origin, route.id_destination)
FROM route WHERE route.id_route = flight.id_route)
      WHEN '1' THEN (SELECT CONCAT_WS(',',route.id_destination, route.id_origin)
FROM route WHERE route.id_route = flight.id_route)
    END AS machine_route,
CASE flight.arrow
      WHEN '0' THEN (SELECT airport.name FROM airport WHERE airport.id_airport =
(SELECT route.id_origin FROM route WHERE route.id_route = flight.id_route))
      WHEN '1' THEN (SELECT airport.name FROM airport WHERE airport.id_airport =
(SELECT route.id_destination FROM route WHERE route.id_route = flight.id_route))
    END AS human_route_origin,
CASE flight.arrow
      WHEN '0' THEN (SELECT airport.name FROM airport WHERE airport.id_airport =
(SELECT route.id_destination FROM route WHERE route.id_route = flight.id_route))
      WHEN '1' THEN (SELECT airport.name FROM airport WHERE airport.id_airport =
(SELECT route.id_origin FROM route WHERE route.id_route = flight.id_route))
    END AS human_route_destination,
time_departure,
(
(SELECT flight.time_departure) + INTERVAL flight.time_offset MINUTE +
```

```
INTERVAL ( (SELECT route.distance FROM route WHERE route.id_route =
flight.id_route) / (SELECT airplane.cruising_speed FROM airplane WHERE
airplane.id_airplane = flight.id_airplane)
) HOUR
) AS expected_arrival,
airline.name AS airline FROM flight
LEFT JOIN airline ON flight.id_airline = airline.id_airline
```

Esta query retorna ao um conjunto de dados semelhante ao seguinte exemplo:

id_flight	machine_route	human_route_origin	human_route_destination	time_departure	expected_arrival	airline
1	PXO,LIS	Porto Santo, Portugal	Lisbon, Portugal	2013-03-24 20:50:00	2013-03-24 21:50:00	TAP Portugal
2	OTA,PXO	OTA, Portugal	Porto Santo, Portugal	2013-03-26 08:10:00	2013-03-26 09:10:00	SATA Internacional
3	FNC,FAO	Funchal, Portugal	Faro, Portugal	2013-03-27 12:00:00	2013-03-27 13:00:00	Hi Fly
4	FLW,OTA	Flores Island, Portugal	OTA, Portugal	2013-03-20 16:00:00	2013-03-20 18:00:00	Hi Fly

Fig 3. — Consulta de todos os voos.

Como se observa a extensa query cria diversos campos úteis para um aeroporto como o nome da origem e destino do voo em formato humanamente legível pronto a ser utilizado em painéis de informação, a data de chegada estimada para o voo, o nome da companhia a operar o voo.

As rotas retornadas pela query já incluem a correcção do sentido do voo que na base de dados é implementada a partir do campo *arrow* na tabela *flight*.

Para simplificar o processo de consulta deverá ser criada uma View para esta query, como definido no **Anexo III - Views, Consulta de voos**.

A partir da View *vFlightList* poderão então ser feitas queries que restrinjam por exemplo os voos por data e hora ou estado. Exemplo:

```
SELECT * FROM vflightlist
WHERE expected_arrival > (SELECT NOW()) AND
id_flight BETWEEN ((SELECT MAX(id_flight) FROM vflightlist)-4) AND (SELECT
MAX(id_flight) FROM vflightlist) AND
machine_route LIKE '%PXO' ORDER BY expected_arrival DESC
```

Que nos retorna os últimos quatro voos por aterrar (com data de chegada superior à data actual) no aeroporto de Porto Santo identificado como PXO:

id_flight	machine_route	human_route_origin	human_route_destination	time_departure	expected_arrival	airline
2	OTA,PXO	OTA, Portugal	Porto Santo, Portugal	2013-03-26 08:10:00	2013-03-26 09:10:00	SATA Internacional

Fig 4. — Consulta de todos os voos por aterrar em Porto Santo.

De forma bastante semelhante podemos também obter a lista de aviões a descolar brevemente de um dado aeroporto modificando apenas a localização do operador "%":

```
SELECT * FROM vflightlist
WHERE expected_arrival > (SELECT NOW()) AND
id_flight BETWEEN ((SELECT MAX(id_flight) FROM vflightlist)-4) AND (SELECT
MAX(id_flight) FROM vflightlist) AND
machine_route LIKE 'OTA%'
```

Obtendo neste caso os voos a descolar do aeroporto da OTA:

id_flight	machine_route	human_route_origin	human_route_destination	time_departure	expected_arrival	airline
2	OTA,PXO	OTA, Portugal	Porto Santo, Portugal	2013-03-26 08:10:00	2013-03-26 09:10:00	SATA Internacional

Fig 5. — Consulta de todos os voos por descolar na OTA.

2) Consulta aviões no ar/voos a decorrer:

Podemos recorrer à última View crida para lista todos os voos a decorrer no momento, através da seguinte query:

```
SELECT * FROM vflightlist
WHERE time_departure < (SELECT NOW()) AND
expected_arrival > (SELECT NOW());
```

Obtendo um resultado semelhante a:

id_flight	machine_route	human_route_origin	human_route_destination	time_departure	expected_arrival	airline
5	FLW,OTA	Flores Island, Portugal	OTA, Portugal	2013-03-25 20:00:00	2013-03-25 22:00:00	Hi Fly

Fig 6. — Consulta de todos os voos a decorrer no momento.

3) Consulta de passageiros num voo:

Para além de estar query já ter sido realizada por intermédio de outra, é importante a qualquer aeroporto conseguir saber a qualquer momento os passageiros de um determinado voo (id 3 neste caso), para tal temos a seguinte query planeada:

```
SELECT id_ticket, id_passport, check_in FROM ticket WHERE id_flight = 3
```

Isto irá ter um retorno relativamente simples uma vez que a única informação que temos dos passageiros é o número do passaporte:

id_ticket	id_passport	check_in
6	239032499	0
7	630498904	1
8	111908498	0
9	902220912	0

Fig 7. — Consulta de passageiros por voo.

A query também nos retorna se o passageiro em questão já fez ou não checkin.

4) Consulta de todas as companhias aéreas com voos a decorrer:

É importante saber quantos voos cada companhia aérea têm a decorrer num dado momento, para tal remos a seguinte query:

```
SELECT COUNT(*) AS flight_cnt, airline FROM vflightlist
WHERE time_departure < (SELECT NOW()) AND
expected_arrival > (SELECT NOW())
GROUP BY airline;
```

Isto resulta no seguinte:

flight_cnt	airline
1	Hi Fly
2	Orbest

Fig 8. — Consulta de todas as companhia aéreas com voos a decorrer.

Na coluna da direita podemos encontrar uma contagem de voos, e na da esquerda o nome da empresa.

Terminam-se assim os pedidos básicos e fundamentais que podem ser feitos à base de dados.

Conclusão

Com este projecto foi possível aprofundar conhecimentos no desenho, normalização, implementação e utilização de bases de dados.

O tema escolhido era bastante abrangente o que permitiu criar mais tabelas do que inicialmente previsto e uma base de dados fortemente baseada em relações onde existe muito pouca redundância.

O principal desafio deste trabalho não esteve apenas focado na área técnica do desenho e normalização de bases de dados e consulta de informação, uma vez que para desenhar uma base de dados minimamente coerente com a realidade e útil na situação proposta era necessário ter um conhecimento considerável do negócio.

Devido à abrangência e complexidade do tema foram de despedidas mais horas a estudar o funcionamento no negócio do que propriamente a implementar a solução, o que por um lado reflecte um projecto real ideal, mas por outro e no contexto do trabalho pode não ter sido assim tão ideal.

Um condicionalismo inesperado que surgiu durante a realização do trabalho foi a falta de suporte ao comando CHECK() nos motores do MySQL o que impossibilitou o teste do mesmo.

De qualquer modo acabo este trabalho considerando que foi positivo, por ter experimentado algumas queries mais avançadas de SQL e também pelo resultado da investigação realizada sobre o negócio.

Bibliografia

[1] Resumo de Unidades para Aviação: http://ivaous.org/academy/index.php/pilot/atp/boeing-737
[2] Airbus A380 OEW:
http://www.airliners.net/aviation-forums/general_aviation/print.main?id=1442319
[3] Dados de Aeroportos e Rotas: http://openflights.org/data.html
[4] Conversão de JET em Litros para Kg:
http://www.airliners.net/aviation-forums/tech_ops/read.main/200513/
[5] Unidades para Aviação: http://www.pnggas.com/pdf/mobil/aviation_guide_tables.pdf
[6] Família Airbus A320: http://en.wikipedia.org/wiki/Airbus_A320_family
[7] Família Airbus A330: http://en.wikipedia.org/wiki/Airbus_A330
[8] Família Airbus A380: http://en.wikipedia.org/wiki/Airbus_A380
[9] Componentes do Takeoff:
http://en.wikipedia.org/wiki/File:Takeoff_Weight_Components_(Malshayef_06-05-2010).jpg

ANEXO I — Código SQL para Gerar a Base de Dados

```
DROP DATABASE IF EXISTS pt_airpace;
CREATE DATABASE pt_airpace DEFAULT CHARACTER SET UTF8 DEFAULT COLLATE utf8_general_ci;
USE pt_airpace;

CREATE TABLE constant(
	id_constant INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
	name VARCHAR(15) NOT NULL UNIQUE,
	value FLOAT NOT NULL
);

CREATE TABLE airport(
	id_airport VARCHAR(3) PRIMARY KEY NOT NULL,
	name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE airplane(
	id_airplane INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
	name VARCHAR(15) NOT NULL UNIQUE,
	passenger_capacity INT(3) UNSIGNED NOT NULL,
	mtow INT(6) UNSIGNED NOT NULL,
	oew INT(6) UNSIGNED NOT NULL,
	cruising_speed INT(4) UNSIGNED NOT NULL, # Milhas nauticas por hora
	max_range INT(4) UNSIGNED NOT NULL,
	max_fuel INT(6) UNSIGNED NOT NULL,
	nmi_consumption FLOAT(10,2) NOT NULL
);

CREATE TABLE airline(
	id_airline INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
	name VARCHAR(50) UNIQUE
);

CREATE TABLE route(
	id_route INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
	id_origin VARCHAR(3) NOT NULL,
	id_destination VARCHAR(3) NOT NULL,
```

```
	distance FLOAT NOT NULL, # Milhas Nauticas
	FOREIGN KEY (id_origin) REFERENCES airport(id_airport),
	FOREIGN KEY (id_destination) REFERENCES airport(id_airport)
);

CREATE TABLE flight(
	id_flight INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
	id_airline INT UNSIGNED NOT NULL,
	id_airplane INT UNSIGNED NOT NULL,
	id_route INT UNSIGNED NOT NULL
		# Verificar se o avião escolhido pode fazer esta rota! ( Distância da rota
< Alcance máximo do avião )
		CHECK ( (SELECT distance FROM route WHERE route.id_route =
flight.id_route) < (
			(SELECT max_range FROM airplane WHERE airplane.id_airplane =
flight.id_airplane)
		)),
	arrow INT(1) NOT NULL DEFAULT 0 CHECK(arrow in (0,1)),
	time_departure DATETIME NOT NULL,
	time_offset INT DEFAULT 0, # Permite criar atrasos na chegada
	post_weight INT UNSIGNED NOT NULL DEFAULT 0
		CHECK( (SELECT mtow FROM airplane WHERE airplane.id_airplane =
flight.id_airplane) > (
			# OEW
			(SELECT oew FROM airplane WHERE airplane.id_airplane =
flight.id_airplane) + # Peso operacional vazio
			# Peso das Pessoas + Bagagem
			(SELECT passenger_capacity FROM airplane WHERE airplane.id_airplane
= flight.id_airplane) * # Número de passageiros
				(
					(SELECT value FROM constant WHERE name = 'passen-
ger_weigh') + # Peso de cada passageiro
					(SELECT value FROM constant WHERE name = 'def_lugga-
ge_uni') * # Número de unidades de bagagem padrão
					(SELECT value FROM constant WHERE name = 'luggage_u-
nit_mu') # Multiplicador das unidades de bagagem
				)
			+
			# Peso do correio
			post_weight +
			# Peso global do combustível * (milhas da rota * consumo por milha
do avião)
```

```
                  (SELECT value FROM constant WHERE name = 'jet_liter_weigh') *
                  (SELECT route FROM constant WHERE route.id_route = flight.id_route)
*
                  (SELECT nmi_consumption FROM airplane WHERE airplane.id_airplane =
flight.id_airplane)
            )),
      FOREIGN KEY (id_route) REFERENCES route(id_route),
      FOREIGN KEY (id_airline) REFERENCES airline(id_airline),
      FOREIGN KEY (id_airplane) REFERENCES airplane(id_airplane)
);

CREATE TABLE ticket(
      id_ticket INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
      id_flight INT UNSIGNED NOT NULL
            # Verificar se se vôo tem este lugar disponível
            CHECK ( id_flight <=
                  (SELECT passenger_capacity FROM airplane WHERE airplane.id_airplane
=
                        (SELECT id_airplane FROM flight WHERE flight.id_flight =
ticket.id_flight ) )
            ),
      id_passport VARCHAR(9) NOT NULL,
      luggage_units INT(100) UNSIGNED NOT NULL DEFAULT 2,
      check_in INT(1) UNSIGNED NOT NULL DEFAULT 0,
      FOREIGN KEY (id_flight) REFERENCES flight(id_flight)
);
```

ANEXO II — Dados de Exemplo

```
INSERT INTO constant
	(name, value)
VALUES
	('passenger_weight', '80'),
	('jet_liter_weight', '0.79'),
	('luggage_unit_mu', '10'),
	('def_luggage_uni', '2'),
	('airplane_stocki', '0.6')
;

INSERT INTO airport
	(name, id_airport)
VALUES
	('Corvo Island, Portugal', 'CVU'),
	('Faro, Portugal', 'FAO'),
	('Flores Island, Portugal', 'FLW'),
	('Funchal, Portugal', 'FNC'),
	('Graciosa Island, Portugal ', 'GRW'),
	('Horta, Portugal', 'HOR'),
	('Lisbon, Portugal', 'LIS'),
	('OTA, Portugal', 'OTA'),
	('Pico Island, Portugal', 'PIX'),
	('Ponta Delgada, Portugal', 'PDL'),
	('Porto Santo, Portugal', 'PXO'),
	('Porto, Portugal', 'OPO'),
	('Santa Maria, Portugal', 'SMA'),
	('Terceira Island, Portugal', 'TER')
;

INSERT INTO airplane
	(name, passenger_capacity, mtow, oew, cruising_speed, max_range, max_fuel,
nmi_consumption)
VALUES
	('Airbus A318', 132, 68000, 39500, 444, 3100, 24210, 7.81),
	('Airbus A319', 156, 75500, 40800, 444, 3700, 24210, 6.54),
	('Airbus A320', 180, 78000, 42600, 444, 3300, 24210, 7.34),
```

```
	('Airbus A321', 220, 93500, 48500, 444, 3220, 24050, 7.47),
	('Airbus A330', 440, 235000, 124500, 470, 5850, 97530, 16.67),
	('Airbus A380', 853, 560000, 280713, 510, 8300, 320000, 38.55)
;

INSERT INTO airline
	(name)
VALUES
	('TAP Portugal'),
	('SATA Internacional'),
	('Orbest'),
	('Luzair'),
	('EuroAtlantic Airways'),
	('Hi Fly'),
	('White')
;

INSERT INTO route
	(id_origin, id_destination, distance)
VALUES
	('LIS', 'PXO', 487.856),
	('OTA', 'PXO', 509.049),
	('LIS', 'OPO', 148.142),
	('OTA', 'FLW', 1031.780),
	('OTA', 'PIX', 904.933),
	('LIS', 'FAO', 135.359),
	('FAO', 'FNC', 514.703)
;

INSERT INTO flight
	(id_route, arrow, id_airline, time_departure, time_offset, id_airplane,
post_weight)
VALUES
	(1, 1, 1, "2013-03-24 20:50:00", 0, 3, 500),
	(2, 0, 2, "2013-03-26 08:10:00", 0, 5, 1000),
	(7, 1, 6, "2013-03-27 12:00:00", 0, 1, 300),
	(4, 1, 6, "2013-03-20 16:00:00", 0, 1, 300),
	(4, 1, 6, "2013-03-25 20:00:00", 0, 1, 300),
	(4, 1, 3, "2013-03-25 20:30:00", 0, 1, 300),
	(7, 1, 3, "2013-03-25 20:35:00", 0, 1, 300)
;
```

```
INSERT INTO ticket
       (id_flight, id_passport, luggage_units)
VALUES
       (1, "129384928", 2),
       (1, "564543219", 3),
       (1, "324239048", 2),
       (1, "112332323", 2),
       (1, "290384029", 5),
       (3, "239032499", 2),
       (3, "630498904", 8),
       (3, "111908498", 2),
       (3, "902220912", 1)
;
```

ANEXO III — Views

Consulta de Voos:

```
CREATE VIEW vFlightList ( id_flight, machine_route, human_route_origin, human_rou-
te_destination, time_departure, expected_arrival, airline ) AS
SELECT flight.id_flight,
CASE flight.arrow
       WHEN '0' THEN (SELECT CONCAT_WS(',',route.id_origin, route.id_destination) FROM
route WHERE route.id_route = flight.id_route)
       WHEN '1' THEN (SELECT CONCAT_WS(',',route.id_destination, route.id_origin) FROM
route WHERE route.id_route = flight.id_route)
    END AS machine_route,
CASE flight.arrow
       WHEN '0' THEN (SELECT airport.name FROM airport WHERE airport.id_airport = (SE-
LECT route.id_origin FROM route WHERE route.id_route = flight.id_route))
       WHEN '1' THEN (SELECT airport.name FROM airport WHERE airport.id_airport = (SE-
LECT route.id_destination FROM route WHERE route.id_route = flight.id_route))
    END AS human_route_origin,
CASE flight.arrow
       WHEN '0' THEN (SELECT airport.name FROM airport WHERE airport.id_airport = (SE-
LECT route.id_destination FROM route WHERE route.id_route = flight.id_route))
       WHEN '1' THEN (SELECT airport.name FROM airport WHERE airport.id_airport = (SE-
LECT route.id_origin FROM route WHERE route.id_route = flight.id_route))
    END AS human_route_destination,
time_departure,
(
(SELECT flight.time_departure) + INTERVAL flight.time_offset MINUTE +
INTERVAL ( (SELECT route.distance FROM route WHERE route.id_route = flight.id_route) /
(SELECT airplane.cruising_speed FROM airplane WHERE airplane.id_airplane =
flight.id_airplane)
) HOUR
) AS expected_arrival,
airline.name AS airline FROM flight
LEFT JOIN airline ON flight.id_airline = airline.id_airline
```