Implementação Data Mart (P01)

DECISION SUPPORT SYSTEMS, 2024-25

João Ribeiro (23795), José Senra (21154), Pedro Pérez (18623)

Introdução

Este projeto tem como objetivo a construção de um Sistema de Apoio à Decisão (SAD) baseado na base de dados AirportDB, que contém informações operacionais sobre voos, passageiros, companhias aéreas, aviões e aeroportos. O objetivo é transformar os dados operacionais em informações analíticas para apoiar a tomada de decisões estratégicas, como a análise de desempenho das companhias aéreas, a movimentação de passageiros, a pontualidade dos voos e a utilização dos aeroportos.
O sistema de apoio à decisão será baseado na modelação dimensional, utilizando conceitos de Data Warehouse, onde serão definidos fatos e dimensões, de forma a permitir análises multidimensionais. O projeto contempla o desenho do modelo dimensional, definição das tabelas de fato e dimensão, bem como o mapeamento das principais métricas e indicadores de desempenho.
Os processos de negócio suportados pela base de dados incluem a gestão de voos, reservas de passageiros, monitoramento das condições meteorológicas que afetam os voos, gestão de aviões e operações de companhias aéreas. O SAD permitirá responder a perguntas como:
Quais são os aeroportos com maior volume de passageiros?
Quais companhias aéreas operam os voos mais pontuais?
Quais as rotas que têm maior demanda ao longo do tempo?

A metodologia adotada incluiu a utilização de uma base de dados fornecida pelo professor Joaquim Silva, com o objetivo de implementar procedimentos ETL (Extract, Transform, Load), documentar o conteúdo da *data mart*, analisar o esquema estrela, desenvolver um modelo de *data warehouse*, e identificar os objetivos dimensionais, bem como os factos e dimensões relevantes. Este projeto ilustra a capacidade das bases de dados em suportar processos de negócio essenciais, diferenciando-se dos DW pela sua habilidade em permitir não apenas a consulta, mas também a modificação de dados.

Data Profiling

Data profiling é uma etapa crítica no processo de preparação de dados, especialmente quando se objetiva a construção de data marts eficazes. Esta fase envolve uma análise detalhada da estrutura, conteúdo e qualidade dos dados armazenados. Identifica-se padrões, anomalias, e a consistência dos dados, permitindo uma compreensão profunda do seu significado e relevância. Através do *data profiling*, pode ser garantida a integridade e a utilidade dos *data marts*, facilitando análises precisas e decisões informadas. Este processo não apenas permite a melhoria da qualidade dos dados, mas também otimiza o desempenho dos sistemas de BI (*Business Intelligence*), contribuindo para uma gestão de dados mais eficiente e confiável.

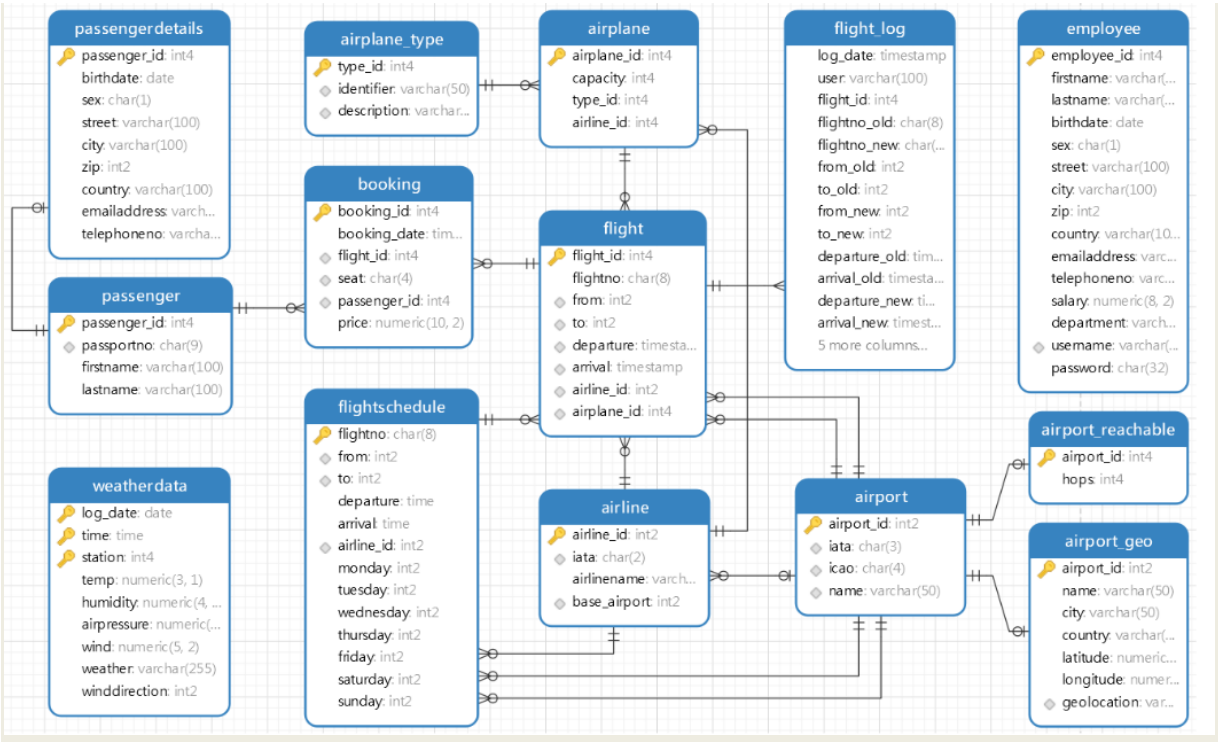
A base de dados apresentada é composta por 14 tabelas e foi desenvolvida para gerir informações relacionadas com voos, companhias aéreas, passageiros, funcionários e aeroportos. Esta estrutura permite o armazenamento eficiente de dados relacionados com reservas, horários de voos, informações detalhadas de aviões e tripulações, bem como o registo de dados meteorológicos e mudanças nos voos. A seguir, é feita uma descrição detalhada de cada uma das tabelas que compõem esta base de dados, permitindo uma compreensão clara dos dados que são armazenados e como estão interligados:

- **passengerdetails**: Esta tabela armazena os detalhes dos passageiros, incluindo o seu identificador único (passenger_id), data de nascimento, sexo, endereço (rua, cidade, código postal, país), e dados de contato, como email e número de telefone.
- **airplane_type**: Armazena os tipos de aviões com um identificador (type_id), uma descrição e um código único para cada tipo de avião.
- **airplane**: Regista informações sobre aviões, como a capacidade (número de lugares), o tipo de avião e a companhia aérea a que pertence.
- **flight_log**: Mantém um registo detalhado das alterações de voo, incluindo a data e hora de log, o utilizador que fez a alteração, e informações sobre alterações de horários e rotas (antes e depois da modificação).
- **employee**: Armazena dados dos funcionários, como o identificador (employee_id), nome, data de nascimento, endereço, salário, departamento, e credenciais de acesso (nome de usuário e senha).
- **passenger**: Tabela que contém os dados básicos dos passageiros, como o identificador (passenger_id), número de passaporte, nome próprio e apelido.
- **booking**: Regista as reservas, incluindo o identificador da reserva, a data, o voo associado, o passageiro, o número de assento e o preço pago pela reserva.

- **flightschedule**: Contém os horários de voos, com informações sobre a rota (de, para), horário de partida, chegada, e a companhia aérea.
- **flight**: Armazena os dados dos voos, incluindo o número do voo, aeroporto de origem e destino, horário de partida e chegada, avião e companhia aérea associada.
- **airline**: Contém informações sobre as companhias aéreas, como o identificador da companhia (airline_id), código IATA e ICAO, nome da companhia e o aeroporto-base.
- **airport**: Armazena informações sobre os aeroportos, incluindo o seu identificador, código IATA e ICAO, nome e localização.
- **weatherdata**: Contém os dados meteorológicos associados aos aeroportos, como a data, hora, estação de registo, temperatura, humidade, pressão do ar, vento e direção do vento.
- **airport_reachable**: Armazena a relação entre aeroportos que podem ser alcançados, com informações sobre o número de escalas necessárias (hops).
- **airport_geo**: Armazena os dados geográficos dos aeroportos, como nome, cidade, país, latitude e longitude.

O esquema completo Entity-Relationship pode ser visto na imagem abaixo:

Figura 1 – Modelo Inicial de Dados



1.1 Análise preliminar de alguns dados

Tabela 1: Summary of AAA database contents

Event / object	Table	Nr. Records
Aeroportos	*airport*	9853
Aviões	*airplane*	5436
Tipos de aviões	*airplane_type*	342
Companhias aéreas	*airline*	113
Voos	*flight*	301945
Programação de voos	*flightschedule*	9633
Reservas de voos	*booking*	3291585
Passageiros	*passenger*	33889
Detalhes dos passageiros	*passengerdetails*	33897
Funcionários	*employee*	1000
Dados Climáticos	*weatherdata*	4626432
Detalhes dos voos	*flight_log*	0
Escalas necessárias	*airport_reachable*	0
Dados geográficos dos aeroportos	*airport_geo*	9854

Modelação dimensional

Tabela 2: Data Warehouse Matrix

DIMENSIONS / FACT TABLES	PassengerDetails	Airplane_Type	Airplane	Passenger	Airline	Airport	Airport_Geo	FlightSchedule
Booking	X			X				
Flight_Log (se tiver dados)			X		X	X		X
Flight		X	X		X	X	X	X
WeatherData						X	X	

Design do modelo de dados dimensional

1.2 Avaliação da granularidade

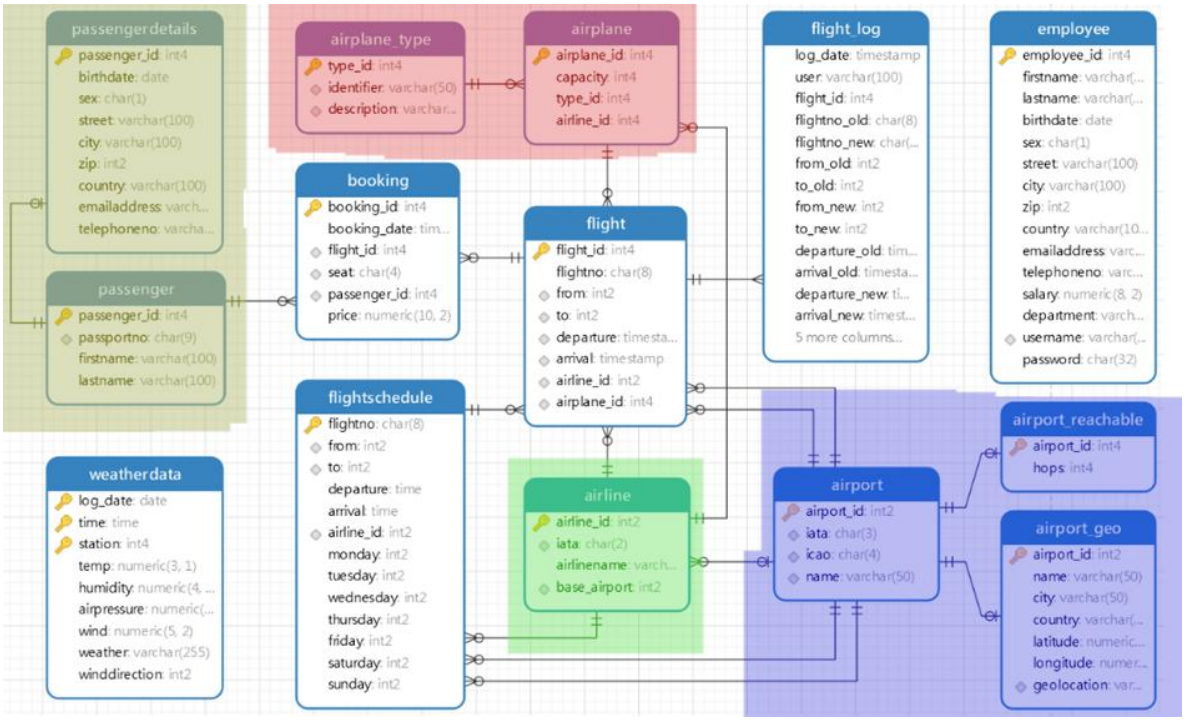
Esta base de dados está bem estruturada e normalizada, graças a uma clara separação entre as diferentes tabelas, como por exemplo as tabelas *"booking"* e *"passenger"*. A normalização permite um nível de detalhe mais elevado em cada tabela, o que confere à base de dados uma granularidade fina. Além disso, a granularidade da informação corresponde a cada linha da tabela *"booking"*, representando individualmente cada reserva efetuada.

1.3 Desnormalização de dados

Para o modelo dimensional, foi necessário desnormalizar, pois essa é uma parte essencial para a criação dos DataMarts. Por desnormalizar, compreende-se o ato de criar redundância dentro de um modelo de dados, a fim de poder ter resultados mais rápidos e precisos, embora custando a melhor organização dos dados. Por esse fim, e para desnormalizar, reduziu-se o número de tabelas e aumentou-se os campos de cada tabela, com o objetivo de criar essa mesma redundância e acesso mais rápido à informação. Decidiu-se assim, juntar os dados seguintes, representados por cores diferentes, em tabelas próprias, agregando-as em partes como é possível ver na figura abaixo.

Será adicionada também a dimensão "date" onde será armazenado todas as datas de funcionamento, esta dimensão estará ligada por uma ligação de muitos para 1 à tabela de factos "WeatherData", armazenando assim a informação meteorológica de especificas datas.

Figura 2 - Desnormalização de dados



Fact_Booking

- booking_id (PK)
- flight_id (FK)
- passenger_id (FK)
- seat
- price

Fact_WeatherData

- log_date (PK)
- time (PK)
- station (PK)
- temp
- humidity
- airpressure

- wind
- weather
- winddirection

Fact_Flight

- flight_id (PK)
- flightno (FK)
- from (FK)
- to (FK)
- airline_id (FK)
- airplane_id (FK)
- desparture_id (FK)
- arrival_id (FK)

DIM_PASSENGER

- passengerID (PK)
- passportno
- firstname
- lastname
- sex
- street
- city
- zip
- country
- emailaddress
- telephoneno
- datatime_id (FK)

DIM_AIRPORT

- airport_id (PK)
- iata
- icao
- name
- city
- country
- latitude
- longitude
- geolocation

DIM_AIRPLANE

- Airplane_id (PK)
- capacity
- identifier
- description
- airline_airline_id

DIM_AIRLINE

- airline_id (PK)
- iata
- airlinename
- base_airport (FK)
- dim_airport_airport_id

DIM_Employee

- employee_id (PK)
- firstname
- lastname
- birthdate
- sex
- street
- city
- zip
- country

- emailaddress
- telephoneno
- salary
- department
- username
- password

DIM_DataTime

- id (PK)
- day
- month
- year
- time

DIM_FlightSchedule

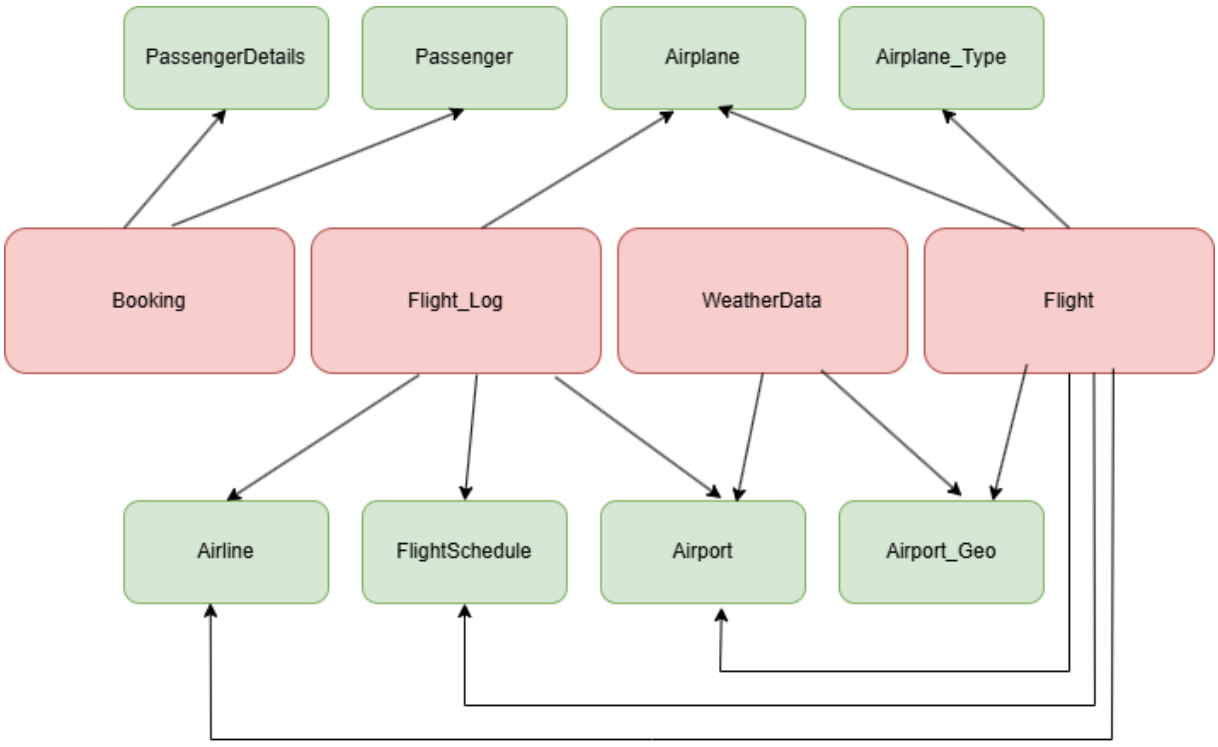
- flightno (PK)
- from (FK)
- to (FK)
- departure
- arrival
- airline_id (FK)
- monday
- tuesday
- wednesday
- thursday
- friday
- saturday
- sunday

1.4 Esquema em estrela

Neste esquema em estrela, estão representadas, tanto as tabelas de facto como as tabelas de dimensão em que, a vermelho estão visíveis as tabelas de facto e a verde as tabelas de dimensão.

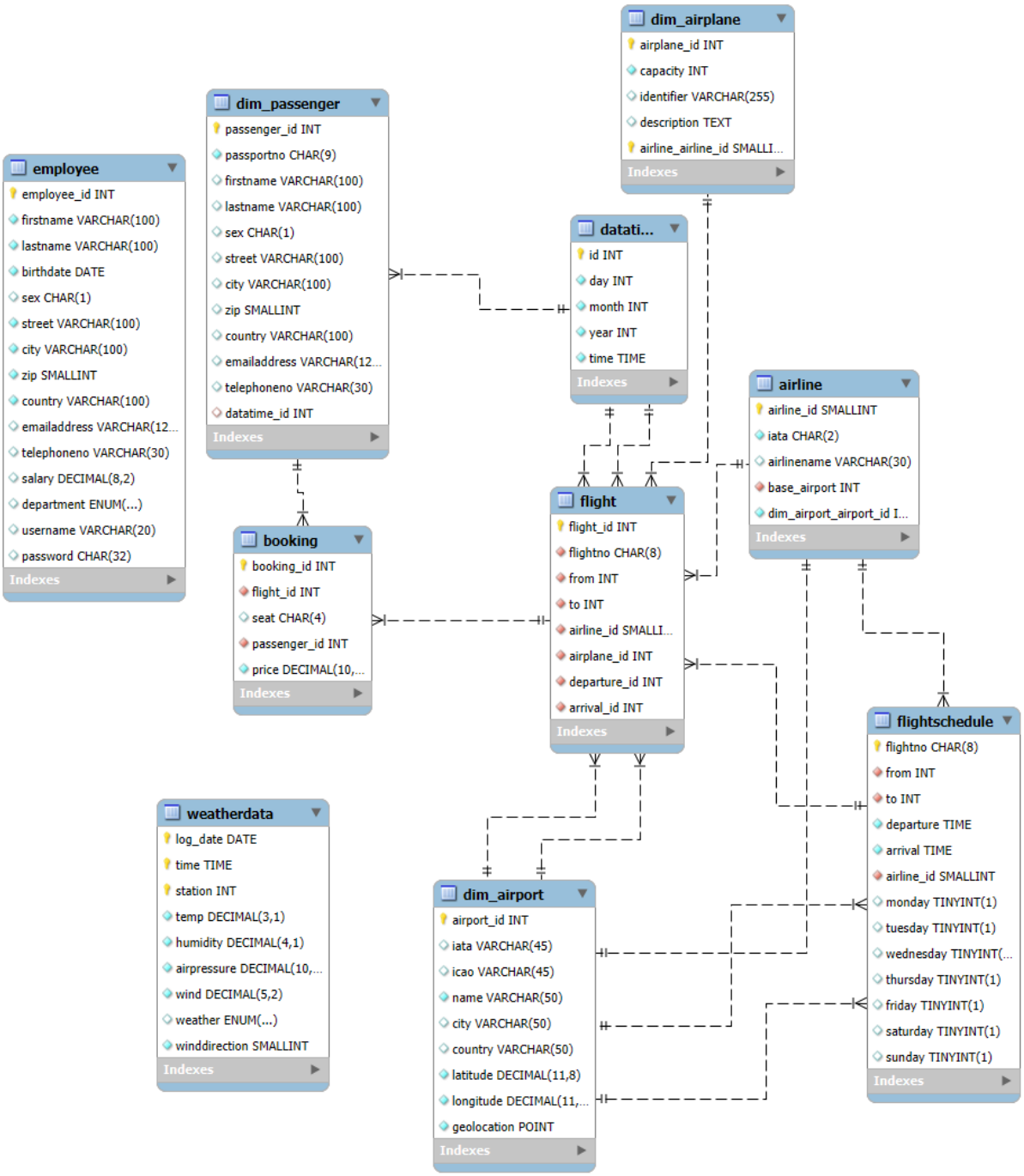
- A tabela de facto “Booking” liga às tabelas de dimensões “PassengerDetails” e “Passenger”.
- A tabela de facto “Flight_Log” liga às tabelas de dimensões “Airplane”, “Airline”, “FlightSchedule” e “Airport”.
- A tabela de facto “WeatherData” liga às tabelas de dimensões “Airport” e “Airport_Geo”.
- A tabela de facto “Flight” liga às tabelas de dimensões “Airplane”, “Airplane_Type”, “Airline”, “Airport”, “Airport_Geo” e “FlightSchedule”.

Figura 3 - Esquema em estrela



1.5 Diagrama ER final do DataMart

Figura 4 - Diagrama ER final



Implementação do Data Mart

1.6 Número de aviões por companhia

Esta implementação tem como objetivo verificar o número de aviões que cada companhia possui. Nesta implementação, requisita os parâmetros airline_id e airlinename da dimensão airline e o atributo airline_id da dimensão dim_airplane. Comparando os atributos airline_id das duas dimensões e quando o mesmo é igual utiliza o contador (number_of_airplanes), estando este associado ao airlinename. Por fim, organiza os dados pelo número de aviões.

```
SELECT
    a.airline_id AS Airline_ID,
    a.airlinename AS Airline,
    COUNT(*) AS Number_of_Airplanes
FROM
    dim_airplane da
JOIN
    airline a ON da.ailene_id = a.airline_id
GROUP BY
    a.airline_id,
    a.airlinename
ORDER BY
    Number_of_Airplanes DESC;
```

Figura 5 - Resultado do numero de aviões por companhia

#	Airline_ID	Airline	Number_of_Airplanes
1	73	Oman Airlines	90
2	45	Iceland Airlines	89
3	94	Syria Airlines	89
4	24	Djibouti Airlines	89
5	81	Romania Airlines	89
6	92	Sudan Airlines	89
7	64	Macau Airlines	88
8	26	Ecuador Airlines	86
9	108	Wake I Airlines	86
10	55	Kenya Airlines	85
11	69	Namibia Airlines	85
12	46	India Airlines	84
13	78	Puerto Rico Airlines	84
14	71	Nicaragua Airlines	84
15	38	Ghana Airlines	83
16	63	Luxembourg Airlines	83
17	12	Brazil Airlines	82
18	68	Myanmar Airlines	82
19	34	Fiji Is Airlines	82
20	91	St Kitts Airlines	82
21	29	Equatorial Guinea Airlines	81
22	104	Uzbekistan Airlines	80
23	18	Croatia Airlines	79
24	66	Micronesia Airlines	78
25	2	Albania Airlines	78

1.7 Número de voos por dia

Esta implementação tem como objetivo verificar o número de voos efetuados em cada dia. Nesta implementação, requisita os parâmetros year, month e day da dimensão datatime e o atributo departure_id da dimensão flight. Comparando os atributos departure_id e datatime_id e caso sejam iguais utilizamos um contador respetivo a esse dia. Por fim, organiza os dados pela data do voo.

```
SELECT
    CONCAT(dt.year, '-', LPAD(dt.month, 2, '0'), '-', LPAD(dt.day, 2, '0')) AS
flight_date,
    COUNT(*) AS number_of_flights
FROM
    flight f
JOIN
    datatime dt ON f.departure_id = dt.id
GROUP BY
    dt.year, dt.month, dt.day
ORDER BY
```

```
    flight_date;
```

Figura 6 - Resultado do numero de voos por dia

#	flight_date	number_of_flights
1	2015-06-01	4925
2	2015-06-02	5085
3	2015-06-03	4947
4	2015-06-04	4894
5	2015-06-05	5017
6	2015-06-06	4957
7	2015-06-07	4986
8	2015-06-08	4925
9	2015-06-09	5085
10	2015-06-10	4947
11	2015-06-11	4894
12	2015-06-12	5017
13	2015-06-13	4957
14	2015-06-14	4986
15	2015-06-15	4925
16	2015-06-16	5085
17	2015-06-17	4947
18	2015-06-18	4894
19	2015-06-19	5017
20	2015-06-20	4957

1.8 Número de voos ao fim de semana por companhia

Esta implementação tem como objetivo verificar o número de voos realizados ao fim de semana por cada companhia. Nesta implementação requisita os parâmetros airline_id e airlinename da dimensão airline e os atributos airline_id, saturday e Sunday da dimensão flightschedule. É verificado se os valores de flightschedule.Saturday ou flightschedule.Sunday é verdadeiro, e caso seja verdadeiro é utilizado o contador respetivo a essa airline. Por fim, organiza os dados pelo número de voos ao fim de semana.

```
SELECT
    a.airline_id,
    a.airlinename,
    COUNT(*) AS weekend_operations
FROM
    flightschedule fs
JOIN
    airline a ON fs.airline_id = a.airline_id
WHERE
    fs.saturday = 1 OR fs.sunday = 1
GROUP BY
```

```
    a.airline_id, a.airlinename
ORDER BY
    weekend_operations DESC;
```

Figura 7 - Resultado do numero de voos ao fim de semana por companhia

#	airline_id	airlinename	weekend_operations
1	66	Micronesia Airlines	85
2	12	Brazil Airlines	83
3	82	Russia Airlines	83
4	69	Namibia Airlines	82
5	99	Tunisia Airlines	81
6	63	Luxembourg Airlines	80
7	28	El Salvador Airlines	79
8	54	Kazakhstan Airlines	79
9	8	Bahamas Airlines	78
10	55	Kenya Airlines	78
11	27	Egypt Airlines	77
12	29	Equatorial Guinea Airlines	77
13	46	India Airlines	77
14	92	Sudan Airlines	77
15	20	Cyprus Airlines	76
16	76	Philippines Airlines	76
17	78	Puerto Rico Airlines	76
18	105	Vanuatu Airlines	76
19	80	Reunion Airlines	75
20	73	Oman Airlines	74

Conclusion

Concluindo, os benefícios da modelação dimensional para as empresas, bem como a segmentação do datawarehouse em data marts, traduzem-se, geralmente, numa melhoria do desempenho operacional. Esta abordagem permite dividir grandes volumes de dados em conjuntos mais específicos, facilitando e acelerando o acesso à informação e o seu uso na tomada de decisões ao nível corporativo. Desde a definição dos processos de negócio, passando pela modelação e pelo desenvolvimento do processo ETL (Extract, Transform, Load), foi possível conceber uma solução de data mart mais eficaz, permitindo uma tomada de decisão mais incisiva.

Este projeto permitiu-nos também assimilar novos conceitos fundamentais no âmbito da inteligência empresarial e da gestão de dados, contribuindo significativamente para o nosso desenvolvimento enquanto profissionais nesta área. Compreendemos a importância de uma arquitetura bem estruturada e de processos consistentes para garantir a fiabilidade da informação e a agilidade na sua análise.

Contudo, é importante reconhecer que o projeto está longe de estar concluído. O data mart, tal como qualquer sistema de apoio à decisão, deve ser encarado como um sistema dinâmico, passível de ser reestruturado ou expandido no futuro, consoante a evolução das necessidades da organização e dos seus objetivos estratégicos.

Assim, este trabalho representa não apenas uma solução técnica, mas também um ponto de partida para melhorias contínuas, sempre com o foco na criação de valor a partir dos dados.

Bibliography

- Material de apoio no Moodle da cadeira de Sistemas de Apoio à decisão disponibilizado pelo professor Joaquim Silva
- https://www.databricks.com/glossary/star-schema
- https://www.youtube.com/watch?v=gRE3E7VUzRU&t=809s
- https://www.youtube.com/watch?v=7HyGM3Iw0Kc

Appendix A – SQL DW Script

Código sql da estrutura da base de dados (MySQL).

Factos Booking

```
DROP TABLE IF EXISTS `booking`;
CREATE TABLE `booking` (
  `booking_id` int NOT NULL AUTO_INCREMENT,
  `flight_id` int NOT NULL,
  `seat` char(4) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `passenger_id` int NOT NULL,
  `price` decimal(10,2) NOT NULL,
  PRIMARY KEY (`booking_id`),
  KEY `flug_idx` (`flight_id`) /*!80000 INVISIBLE */,
  KEY `fk_booking_passenger` (`passenger_id`),
  CONSTRAINT `buchung_ibfk_1` FOREIGN KEY (`flight_id`) REFERENCES `flight` (`flight_id`),
    CONSTRAINT `fk_booking_passenger` FOREIGN KEY (`passenger_id`) REFERENCES
`dim_passenger` (`passenger_id`)
) ENGINE=InnoDB AUTO_INCREMENT=55099781 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='Flughafen DB by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC
BY 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```

Factos WeatherData

```
DROP TABLE IF EXISTS `weatherdata`;
CREATE TABLE `weatherdata` (
  `log_date` date NOT NULL,
  `time` time NOT NULL,
  `station` int NOT NULL,
  `temp` decimal(3,1) NOT NULL,
  `humidity` decimal(4,1) NOT NULL,
  `airpressure` decimal(10,2) NOT NULL,
  `wind` decimal(5,2) NOT NULL,
   `weather` enum('fog-snowfall','snowfall','rain','rain-snowfall','fog-rain','fog-rain-
thunderstorm','thunderstorm','fog','rain-thunderstorm') CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci DEFAULT NULL,
  `winddirection` smallint NOT NULL,
  PRIMARY KEY (`log_date`,`time`,`station`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='Flughafen DB
by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC BY 4.0. To view a
copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```

Factos Flight

```
DROP TABLE IF EXISTS `flight`;
CREATE TABLE `flight` (
  `flight_id` int NOT NULL AUTO_INCREMENT,
  `flightno` char(8) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `from` int NOT NULL,
  `to` int NOT NULL,
  `airline_id` smallint NOT NULL,
  `airplane_id` int NOT NULL,
  `departure_id` int NOT NULL,
  `arrival_id` int NOT NULL,
  `departed_onschedule` tinyint DEFAULT NULL,
  PRIMARY KEY (`flight_id`),
  KEY `fluglinie_idx` (`airline_id`),
  KEY `fk_arrival_datatimeid_idx` (`arrival_id`),
  KEY `fk_departure_datatimeid_idx` (`departure_id`),
  KEY `fk_airplane_id_idx` (`airplane_id`),
  KEY `fk_from_id_idx` (`from`),
  KEY `fk_to_id_idx` (`to`),
  KEY `fk_flightno_id_idx` (`flightno`),
   CONSTRAINT `fk_airplane_id` FOREIGN KEY (`airplane_id`) REFERENCES `dim_airplane`
(`airplane_id`),
   CONSTRAINT `fk_arrival_datatimeid` FOREIGN KEY (`arrival_id`) REFERENCES `datatime`
(`id`),
  CONSTRAINT `fk_departure_datatimeid` FOREIGN KEY (`departure_id`) REFERENCES `datatime`
(`id`),
   CONSTRAINT `fk_flightno_id` FOREIGN KEY (`flightno`) REFERENCES `flightschedule`
(`flightno`),
  CONSTRAINT `fk_from_id` FOREIGN KEY (`from`) REFERENCES `dim_airport` (`airport_id`),
  CONSTRAINT `fk_to_id` FOREIGN KEY (`to`) REFERENCES `dim_airport` (`airport_id`),
  CONSTRAINT `flug_ibfk_3` FOREIGN KEY (`airline_id`) REFERENCES `airline` (`airline_id`)
) ENGINE=InnoDB AUTO_INCREMENT=503856 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='Flughafen DB by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC
BY 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```

Dimensão Passenger

```
DROP TABLE IF EXISTS `dim_passenger`;
CREATE TABLE `dim_passenger` (
  `passenger_id` int NOT NULL,
  `passportno` char(9) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `firstname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `lastname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `sex` char(1) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `street` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `city` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
```

```
  `zip` smallint DEFAULT NULL,
  `country` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `emailaddress` varchar(120) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `telephoneno` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `datatime_id` int DEFAULT NULL,
  PRIMARY KEY (`passenger_id`),
  KEY `id` (`datatime_id`) /*!80000 INVISIBLE */,
  CONSTRAINT `fk_passanger_datatimeid` FOREIGN KEY (`datatime_id`) REFERENCES `datatime`
(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Dimensão Airport

```
DROP TABLE IF EXISTS `dim_airport`;
CREATE TABLE `dim_airport` (
  `airport_id` int NOT NULL,
  `iata` varchar(45) DEFAULT NULL,
  `icao` varchar(45) DEFAULT NULL,
  `name` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `city` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `country` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `latitude` decimal(11,8) NOT NULL,
  `longitude` decimal(11,8) NOT NULL,
  `geolocation` point NOT NULL,
  PRIMARY KEY (`airport_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Dimensão Airplane

```
DROP TABLE IF EXISTS `dim_airplane`;
CREATE TABLE `dim_airplane` (
  `airplane_id` int NOT NULL,
  `capacity` int NOT NULL,
  `ailene_id` int NOT NULL,
  `identifier` varchar(255) DEFAULT NULL,
  `description` text,
  PRIMARY KEY (`airplane_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Dimensão Airline

```
DROP TABLE IF EXISTS `airline`;
CREATE TABLE `airline` (
  `airline_id` smallint NOT NULL AUTO_INCREMENT,
  `iata` char(2) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `airlinename` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `base_airport` int NOT NULL,
  PRIMARY KEY (`airline_id`),
  UNIQUE KEY `iata_unq` (`iata`),
  KEY `fk_baseairport_id_idx` (`base_airport`)
) ENGINE=InnoDB AUTO_INCREMENT=114 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='Flughafen DB by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC
BY 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```

Dimensão Employee

```
DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `employee_id` int NOT NULL AUTO_INCREMENT,
  `firstname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `lastname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `birthdate` date NOT NULL,
  `sex` char(1) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `street` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `city` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `zip` smallint NOT NULL,
  `country` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
   `emailaddress` varchar(120) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT
NULL,
  `telephoneno` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `salary` decimal(8,2) DEFAULT NULL,
      `department`     enum('Marketing','Accounting','Management','Logistics','Airfield')
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `username` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `password` char(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`employee_id`),
  UNIQUE KEY `benutzer_unq` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=1001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='Flughafen DB by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC
BY 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```

Dimensão DataTime

```
DROP TABLE IF EXISTS `datatime`;
CREATE TABLE `datatime` (
  `id` int NOT NULL AUTO_INCREMENT,
  `day` int NOT NULL,
  `month` int NOT NULL,
  `year` int NOT NULL,
  `time` time NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_datetime` (`day`,`month`,`year`,`time`)
) ENGINE=InnoDB AUTO_INCREMENT=103381 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Dimensão FlightSchedule

```
DROP TABLE IF EXISTS `flightschedule`;
CREATE TABLE `flightschedule` (
  `flightno` char(8) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `from` int NOT NULL,
  `to` int NOT NULL,
  `departure` time NOT NULL,
  `arrival` time NOT NULL,
  `airline_id` smallint NOT NULL,
  `monday` tinyint(1) DEFAULT '0',
  `tuesday` tinyint(1) DEFAULT '0',
  `wednesday` tinyint(1) DEFAULT '0',
  `thursday` tinyint(1) DEFAULT '0',
  `friday` tinyint(1) DEFAULT '0',
  `saturday` tinyint(1) DEFAULT '0',
  `sunday` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`flightno`),
  KEY `fk_airline_id_idx` (`airline_id`),
  KEY `fk_scheduleto_id_idx` (`to`),
  KEY `fk_schedulefrom_id_idx` (`from`),
    CONSTRAINT `fk_airline_id` FOREIGN KEY (`airline_id`) REFERENCES `airline`
(`airline_id`),
    CONSTRAINT `fk_schedulefrom_id` FOREIGN KEY (`from`) REFERENCES `dim_airport`
(`airport_id`),
  CONSTRAINT `fk_scheduleto_id` FOREIGN KEY (`to`) REFERENCES `dim_airport` (`airport_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='Flughafen DB
by Stefan Pröll, Eva Zangerle, Wolfgang Gassler is licensed under CC BY 4.0. To view a
copy of this license, visit https://creativecommons.org/licenses/by/4.0';
```