Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia e Gestão de Sistemas de Informação

Assignment 1 - MySQL

Analysis of flights from and to Brazil in the year 2020

Data Bases 2

Ano Letivo de 2023/2024

e11273, Tomaso Stefanizzi

Guimarães, Março de 2024

Index

1 Introduction

This assignment consists in finding a dataset online and executing some queries in order to answer some chosen analytical questions. In the context of this project, I wanted to analyze the flights from and to Brazil in the year 2020. This data is publicly available in the website of Agência Nacional de Aviação Civil (ANAC) [1], the regulatory agency responsible for overseeing and regulating civil aviation activities within the country of Brazil. The airlines must share with ANAC several information in order to ensure safety, compliance with regulations, and protection of the interests of both the industry and passengers. In the following sections I'll provide the data source, the data wrangling procedure, the relational and conceptual model of the dataset, a data dictionary, as well as some analytical questions and answers.

2 MySQL

2.1 Chosen dataset

As I anticipated in the introduction, I chose to work with flights data from and to Brazil. The chosen dataset was found on kaggle (click here to access it). The original data contains the hystorical data of flights from the year 2000 until 2021.
The original dataset was subdivided in six tables:

- **DW_ARPT_DEST**: it contains the destinations of the flights.
- **DW_ARPT_ORIGEM**: it contains the airports of departure for the flights.
- **DW_EMPRESA**: it provides the informations about the airlines.
- **DW_EQPT**: it contains models of the planes used for the flights.
- **DW_TIPO_LINHA**: it describes some informations about the route, like its purpose and if it contains passengers
- **DW_VOOS**: each row represents a single flight. this table contains all the flights from the year 2000 to the year 2021, as well as some informations about them.

2.2 Data Wrangling

In order to work with a relatively concise dataset, I decided to filter the data, by considering only the flights taken in the year 2020. The flights table has been drastically re-scaled, from 5GB of data to approximately 55Mb (still, it's a fair amount of data points, since we have 367'029 rows, hence single flights). As a consequence, I just kept the airports and airlines which where present in this sub-set of the original dataset.

Moreover, the original dataset presented some inconsistencies that needed to be solved before proceeding with the population of the SQL Database. Some examples are duplicated rows for the same airport, or even wrong airport identifiers (both iata and icao identifiers).

Lastly, I decided to merge the tables **DW_ARPT_ORIGEM** and **DW_ARPT_DEST** in a unique table called **ariport**, since it didn't make sense to consider the origin and the destination as different entities, and finally I dropped some of the columns from the tables that where not that interesting or that contained too many null values.

The processed table are now:

- **airport**: as I said, it's a merge between the two tables from destinations and origin
- **company**: cleaned version of `DW_EMPRESA`
- **equipment**: cleaned version of `DW_EQPT`
- **line**: cleaned version of `DW_TIPO_LINHA`
- **flights**: cleaned and filtered version of `DW_VOOS`

2.3 Conceptual and Relational Database Model

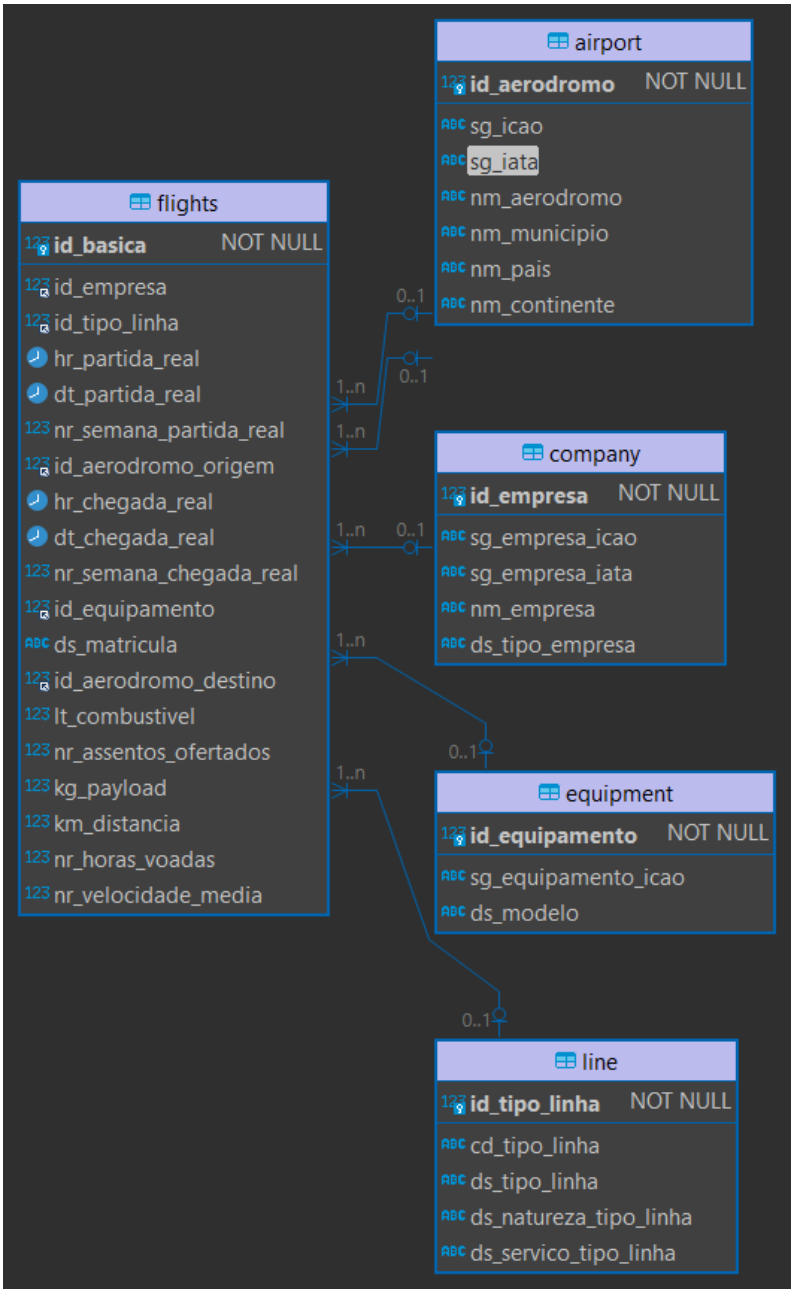

Figura 2.1: ER-Diagram

The conceptual model is the following:

Table `airport`

- `id_aerodromo` (Primary Key)
- `sg_icao`
- `sg_iata`
- `nm_aerodromo`
- `nm_municipio`

- nm_pais
- nm_continente

Table `line`

- id_tipo_linha (Primary Key)
- cd_tipo_linha
- ds_tipo_linha
- ds_natureza_tipo_linha
- ds_servico_tipo_linha

Table `equipment`

- id_equipamento (Primary Key)
- sg_equipamento_icao
- ds_modelo

Table `company`

- id_empresa (Primary Key)
- sg_empresa_icao
- sg_empresa_iata
- nm_empresa
- ds_tipo_empresa

Table `flights`

- id_basica (Primary Key)
- id_aerodromo_origem (References `airport.id_aerodromo`)
- id_aerodromo_destino (References `airport.id_aerodromo`)
- id_empresa (References `company.id_empresa`)
- ds_natureza_etapa (References `line.ds_natureza_tipo_linha`)
- id_equipamento (References `equipment.id_equipamento`)
- hr_partida_real
- dt_partida_real
- nr_semana_partida_real
- hr_chegada_real
- dt_chegada_real
- nr_semana_chegada_real

- ds_matricula
- lt_combustivel
- nr_assentos_ofertados
- kg_payload
- km_distancia
- nr_horas_voadas
- nr_velocidade_media

2.4 Data Dictionary

Table `airport`

Attribute	Data Type	Description
id_aerodromo	Integer	Unique identifier for an airport.
sg_icao	Varchar(4)	ICAO code for the airport.
sg_iata	Varchar(3)	IATA code for the airport.
nm_aerodromo	Varchar(50)	Name of the airport.
nm_municipio	Varchar(50)	Municipality where the airport is located.
nm_pais	Varchar(50)	Country where the airport is situated.
nm_continente	Varchar(30)	Continent where the airport is situated.

Table `line`

Attribute	Data Type	Description
id_tipo_linha	Integer	Unique identifier for a line type.
cd_tipo_linha	Char	Code representing the type of line.
ds_tipo_linha	Varchar(30)	Description of the line type.
ds_natureza_tipo_linha	Varchar(15)	Nature of the line type.
ds_servico_tipo_linha	Varchar(15)	Description of the service referring to the type of line (Passageiro/Cargueiro).

Table `equipment`

Attribute	Data Type	Description
id_equipamento	Integer	Code identifying the aircraft model.
sg_equipamento_icao	Varchar(4)	ICAO designator of the aircraft model (“Type Designator”)
ds_modelo	Varchar(50)	Description of the aircraft model.

Table `company`

Attribute	Data Type	Description
id_empresa	Integer	Unique identifier for a company.
sg_empresa_icao	Varchar(3)	ICAO acronym of the airline. Refers to the designator of the air transport company obtained from the ICAO (International Civil Aviation Organization).
sg_empresa_iata	Varchar(2)	IATA acronym for the airline. Refers to the air transport company designator obtained from IATA (International Air Transport Association).
nm_empresa	Varchar(100)	Airline name.
ds_tipo_empresa	Text	Description of the type of company. Refers to the description of the type of company in relation to the service performed.

Table `flights`

Key	Data Type	Description
id_basica	Integer	Unique identifier for a flight.
id_aerodromo_origem	Integer	foreign key
id_aerodromo_destino	Integer	foreign key
id_empresa	Integer	foreign key
id_tipo_linha	Integer	foreign key
id_equipamento	Integer	foreign key
hr_partida_real	TIME(0)	Actual departure time.
dt_partida_real	Datetime	Actual departure date.
nr_semana_partida_real	Integer	Week number of actual departure.
hr_chegada_real	TIME(0)	Actual arrival time.
dt_chegada_real	Datetime	Actual arrival date.
nr_semana_chegada_real	Integer	Week number of actual arrival.
ds_matricula	Varchar(3)	Aircraft registration code.
lt_combustivel	Integer	Fuel capacity in liters.
nr_assentos_ofertados	Integer	Number of offered seats.
kg_payload	Integer	Payload weight in kilograms.
km_distancia	Integer	Flight distance in kilometers.
nr_horas_voadas	Real	flying hours (for a single flight).
nr_velocidade_media	Real	Average flight speed.

2.5 Analytical Questions

In this section I provide three analytical questions that I find interesting in the context of flights data.

2.5.1 Query 1 - Airplane stats

For each plane model, print some informations as it's avg speed, avg liters consumed etc.

```
SELECT
  e.ds_modelo AS plane_model,
  COUNT(*) AS flights_count,
  AVG(f.nr_velocidade_media) AS avg_speed,
  AVG(f.lt_combustivel) as avg_liters,
  AVG(f.nr_horas_voadas) as avg_hours_per_flight,
  AVG(f.km_distancia)as avg_distance_in_km,
  AVG(f.kg_payload) as avg_payload_in_kg
FROM
  bd2.flights f
JOIN
  bd2.equipment e ON f.id_equipamento = e.id_equipamento
WHERE f.lt_combustivel != 0 and
      f.nr_horas_voadas != 0 and
      f.kg_payload != 0 and
      f.nr_velocidade_media != 0 and
      f.km_distancia != 0
GROUP BY
  e.ds_modelo
ORDER BY
  flights_count DESC
```

plane_model	flights_count	avg_speed	avg_liters	avg_hours_per_flight	avg_distance_in_km	avg_payload_in_kg
AIRBUS A320-100/200	90.084	561,491697	5.540,8644	2,0339746068	1.220,0161	17.251,4186
BOEING 737-800	86.355	544,216963	5.681,546	2,0630639117	1.210,5085	19.800
EMBRAER 195/ERJ-190-200	43.362	468,956514	3.167,1831	1,3261827007	649,5523	11.047,2702
AEROSPATIALE/ALENIA ATR 72 201/202	36.999	294,171877	1.264,1717	1,3666500288	417,8461	6.691,5954
AIRBUS A319	25.125	464,351876	3.701,9059	1,3999867835	695,8428	16.399,8622
BOEING 737-700	22.659	500,879962	4.080,4701	1,6938170158	912,0162	15.947
AIRBUS A321-100/200	17.567	595,518005	7.993,2556	2,3254919468	1.458,5914	26.881,2525
CESSNA 208 CARAVAN	9.839	230,705179	276,1319	1,5356083117	367,7004	647,0647
BOEING 767-300	8.119	670,879665	23.037,8916	4,4885123872	3.215,4756	46.701,2369
EMBRAER E195-E2/ERJ-190-400	7.466	479,375885	2.052,1547	1,404786142	697,9502	13.892,9293
EMBRAER 190/ERJ-190-100	3.123	481,289686	3.353,8236	1,4270786593	708,9705	10.622,0199
AIRBUS A321NEO	2.071	592,598812	5.434,6045	2,1737647842	1.382,9522	22.941,071
AIRBUS A330-200	1.980	731,356672	39.220,8798	5,8818264949	4.403,1263	45.137,8111
BOEING 777-300ER PAX	1.748	772,388221	72.020,5824	8,2364418089	6.582,1276	63.805,6104
BOEING 737-400	1.623	625,354572	8.221,87	3,1004107431	2.012,7184	18.278,5644
BOEING 727-200	1.565	431,750562	5.517,2856	1,1681682524	526,0128	22.402,278
AIRBUS A350-900	1.168	801,594358	70.581,6858	9,4846600856	7.694,4872	52.309
AEROSPATIALE/ALENIA ATR 42-300 / 320	1.032	361,502713	1.014,7878	1,4495962994	534,6366	4.536,4312
AEROSPATIALE/ALENIA ATR 42-500	894	358,218523	1.031,7506	1,3762676454	502,3949	4.697,3848
AIRBUS A330-900NEO	793	761,114943	22.168,5839	7,6357294451	5.897,0757	44.548,6154
BOEING 737-300	535	613,453159	8.581,8748	2,8896883607	1.843,9215	17.037,9421

2.5.2 Query 2 - Airline stats

For each company, display their total distance traveled and total hours of flight

```
SELECT
  c.nm_empresa,
  SUM(f.km_distancia) AS total_distance,
  SUM(f.nr_horas_voadas) as total_hours
```

```
FROM bd2.company c
JOIN bd2.flights f ON c.id_empresa = f.id_empresa
GROUP BY c.nm_empresa
ORDER BY total_distance DESC
```

nm_empresa	total_distance	total_hours
TAM LINHAS AÉREAS S.A.	139.666.212	226.523,4187290089
GOL LINHAS AÉREAS S.A. (EX- VRG LINHAS AÉREAS S.A.)	125.625.460	217.564,8337770038
AZUL LINHAS AÉREAS BRASILEIRAS S/A	116.701.938	219.485,6532709143
ABSA - AEROLINHAS BRASILEIRAS S.A.	10.144.439	14.282,232209
PASSAREDO TRANSPORTES AÉREOS S.A.	3.665.057	11.264,250263
TWO TÁXI AÉREO LTDA.	3.426.548	14.532,916866
MODERN TRANSPORTE AÉREO DE CARGA S.A.	1.843.708	2.950,516526
MAP TRANSPORTES AÉREOS LTDA.	1.155.101	3.338,533464
TOTAL LINHAS AÉREAS S.A.	964.627	2.232,299926
ASTA LINHAS AÉREAS LTDA (EX - AMÉRICA DO SUL LINHAS AÉREAS LTDA.)	210.066	844,633297
CONNECT LINHAS AÉREAS S.A. (ANTIGA CONNECT TÁXI AÉREO LTDA.)	183.634	277,799963
OMNI TÁXI AÉREO S.A.	168.344	451,365951
SIDERAL LINHAS AÉREAS LTDA.	4.174	7,583333

2.5.3 Query 3 - Flight Routes

Find all the routes in the database (hence, for year 2020). Also, count how many times they have been done.

```
SELECT
  a_origem.nm_aerodromo AS origin_airport,
  a_origem.sg_iata as IATA_origin,
  a_destino.nm_aerodromo AS destination_airport,
  a_destino.sg_iata as IATA_destination,
  COUNT(*) AS route_count
FROM
  bd2.flights f
JOIN
  bd2.airport a_origem ON f.id_aerodromo_origem = a_origem.id_aerodromo
JOIN
  bd2.airport a_destino ON f.id_aerodromo_destino = a_destino.id_aerodromo
GROUP BY
  f.id_aerodromo_origem,
  a_origem.nm_aerodromo,
  f.id_aerodromo_destino,
  a_destino.nm_aerodromo
ORDER BY
  route_count DESC
```

2.6 Conclusions

This project consisted in analyzing an existing dataset and perform some queries, in order to respond at some analytical questions.
The dataset was quite large, so I had to perform some filtering in order to reduce the initial dimensions. SQL is an easy way to extract interesting insights from this dataset, and I didn't find too many difficulties in doing so. Perhaps, given the nature of the data, it might be more interesting to perform the queries and visualize some informations (such the routes) with a graph based technology.

origin_airport	IATA_origin	destination_airport	IATA_destination	route_count
CONGONHAS	CGH	SANTOS DUMONT	SDU	6.592
SANTOS DUMONT	SDU	CONGONHAS	CGH	6.569
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	SALGADO FILHO	POA	3.907
SALGADO FILHO	POA	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	3.880
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	GUARARAPES - GILBERTO FREYRE	REC	3.129
GUARARAPES - GILBERTO FREYRE	REC	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	3.049
DEPUTADO LUÍS EDUARDO MAGALHÃES	SSA	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.932
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	DEPUTADO LUÍS EDUARDO MAGALHÃES	SSA	2.929
SANTOS DUMONT	SDU	PRESIDENTE JUSCELINO KUBITSCHEK	BSB	2.703
TANCREDO NEVES	CNF	CONGONHAS	CGH	2.696
PRESIDENTE JUSCELINO KUBITSCHEK	BSB	SANTOS DUMONT	SDU	2.680
TANCREDO NEVES	CNF	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.677
CONGONHAS	CGH	TANCREDO NEVES	CNF	2.677
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	TANCREDO NEVES	CNF	2.666
PRESIDENTE JUSCELINO KUBITSCHEK	BSB	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.615
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	PRESIDENTE JUSCELINO KUBITSCHEK	BSB	2.606

BRIG. EDUARDO GOMES	ZZ9	JUÍNA	JIA	1
JOÃO DURVAL CARNEIRO	N/I	VIRACOPOS	VCP	1
GLAUBER DE ANDRADE ROCHA	VDC	BAHIA - JORGE AMADO	IOS	1
GLAUBER DE ANDRADE ROCHA	VDC	AEROPORTO ESTADUAL DE JUNDIAÍ	QDV	1

3 Cassandra

In this section I will answer the same analytical questions seen in the Section 2, by using a NoSQL, columnar database technology, Cassandra.

3.1 Data preparation

Before importing the data into the Cassandra database, I had to do some modifications on the previous structure of my data. This technology unfortunately doesn't allow to perform the classic join operation, so I had to merge the tables into a single, big one, in SQL:

```sql
SELECT
  f.*,
  c.sg_empresa_icao, c.sg_empresa_iata, c.nm_empresa, c.ds_tipo_empresa,
        l.cd_tipo_linha, l.ds_tipo_linha, l.ds_natureza_tipo_linha,
        l.ds_servico_tipo_linha, e.sg_equipamento_icao, e.ds_modelo,
  airport_destino.sg_icao as icao_airport_destino,
        airport_destino.sg_iata as iata_airport_destino,
  airport_destino.nm_aerodromo as nm_aerodromo_destino,
        airport_destino.nm_municipio as nm_municipio_destino,
  airport_destino.nm_pais as nm_pais_destino,
        airport_destino.nm_continente as nm_continente_destino,
  airport_origem.sg_icao as icao_airport_origem,
        airport_origem.sg_iata as iata_airport_origem,
  airport_origem.nm_aerodromo as nm_aerodromo_origem,
        airport_origem.nm_municipio as nm_municipio_origem,
  airport_origem.nm_pais as nm_pais_origem,
        airport_origem.nm_continente as nm_continente_origem
FROM bd2.flights f
  JOIN bd2.company c
            ON f.id_empresa = c.id_empresa
  JOIN bd2.line l
            ON f.id_tipo_linha = l.id_tipo_linha
  JOIN bd2.equipment e
            ON f.id_equipamento = e.id_equipamento
  JOIN bd2.airport AS airport_origem
            ON f.id_aerodromo_origem = airport_origem.id_aerodromo
  JOIN bd2.airport AS airport_destino
            ON f.id_aerodromo_destino = airport_destino.id_aerodromo
```

After exporting the result of the query into a csv file, I tried to import the data into the Cassandra database with the copy command. Unfortunately, there were many problems, since the csv exported from DBeaver wasn't correctly formatted. As the matter of fact, when trying to export the full table in the JSON format, I noticed that there were many \r characters, that had the same effect of a \n, so moving the cursor down to the next line. In order to find a workaround to this issue, I manually removed all the character in the JSON file with a Python script, and then export it into a csv format using the pandas library.

Figura 3.1: first 3 lines before, last 3 lines after

Finally, in order to avoid any problem with data types, I split the date and hour columns into single integers columns.

3.2 Analytical Questions

3.2.1 Query 1 - Airplane stats

For each plane model, print some informations as it's avg speed, avg liters consumed etc.

```sql
SELECT ds_modelo AS plane_model,
        COUNT(*) AS flights_count,
        AVG(nr_velocidade_media) AS avg_speed,
        AVG(lt_combustivel) AS avg_liters,
        AVG(nr_horas_voadas) AS avg_hours_per_flight,
        AVG(km_distancia) AS avg_distance_in_km,
        AVG(kg_payload) AS avg_payload_in_kg
FROM
  bd2.flights
WHERE
  lt_combustivel != 0
  AND nr_horas_voadas != 0
  AND kg_payload != 0
  AND nr_velocidade_media != 0
  AND km_distancia != 0
GROUP BY
  ds_modelo
ORDER BY
  flights_count DESC;
```

plane_model	flights_count	avg_speed	avg_liters	avg_hours_per_flight	avg_distance_in_km	avg_payload_in_kg
AIRBUS A320-100/200	90.084	561,491697	5.540,8644	2,0339746068	1.220,0161	17.251,4186
BOEING 737-800	86.355	544,216963	5.681,546	2,0630639117	1.210,5085	19.800
EMBRAER 195/ERJ-190-200	43.362	468,956514	3.167,1831	1,3261827007	649,5523	11.047,2702
AEROSPATIALE/ALENIA ATR 72 201/202	36.999	294,171877	1.264,1717	1,3666500288	417,8461	6.691,5954
AIRBUS A319	25.125	464,351876	3.701,9059	1,3999867835	695,8428	16.399,8622
BOEING 737-700	22.659	500,879962	4.080,4701	1,6938170158	912,0162	15.947
AIRBUS A321-100/200	17.567	595,518005	7.993,2556	2,3254919468	1.458,5914	26.881,2525
CESSNA 208 CARAVAN	9.839	230,705179	276,1319	1,5356083117	367,7004	647,0647
BOEING 767-300	8.119	670,879665	23.037,8916	4,4885123872	3.215,4756	46.701,2369
EMBRAER E195-E2/ERJ-190-400	7.466	479,375885	2.052,1547	1,404786142	697,9502	13.892,9293
EMBRAER 190/ERJ-190-100	3.123	481,289686	3.353,8236	1,4270786593	708,9705	10.622,0199
AIRBUS A321NEO	2.071	592,598812	5.434,6045	2,1737647842	1.382,9522	22.941,071
AIRBUS A330-200	1.980	731,356672	39.220,8798	5,8818264949	4.403,1263	45.137,8111
BOEING 777-300ER PAX	1.748	772,388221	72.020,5824	8,2364418089	6.582,1276	63.805,6104
BOEING 737-400	1.623	625,354572	8.221,87	3,1004107431	2.012,7184	18.278,5644
BOEING 727-200	1.565	431,750562	5.517,2856	1,1681682524	526,0128	22.402,278
AIRBUS A350-900	1.168	801,594358	70.581,6858	9,4846600856	7.694,4872	52.309
AEROSPATIALE/ALENIA ATR 42-300 / 320	1.032	361,502713	1.014,7878	1,4495962994	534,6366	4.536,4312
AEROSPATIALE/ALENIA ATR 42-500	894	358,218523	1.031,7506	1,3762676454	502,3949	4.697,3848
AIRBUS A330-900NEO	793	761,114943	22.168,5839	7,6357294451	5.897,0757	44.548,6154
BOEING 737-300	535	613,453159	8.581,8748	2,8896883607	1.843,9215	17.037,9421

3.2.2 Query 2 - Airline stats

For each company, display their total distance traveled and total hours of flight.

```sql
SELECT
  nm_empresa,
  SUM(km_distancia) AS total_distance,
  SUM(nr_horas_voadas) as total_hours
```

```
FROM bd2.flights
GROUP BY nm_empresa
ORDER BY total_distance DESC;
```

nm_empresa	total_distance	total_hours
TAM LINHAS AÉREAS S.A.	139.666.212	226.523,4187290089
GOL LINHAS AÉREAS S.A. (EX- VRG LINHAS AÉREAS S.A.)	125.625.460	217.564,8337770038
AZUL LINHAS AÉREAS BRASILEIRAS S/A	116.701.938	219.485,6532709143
ABSA - AEROLINHAS BRASILEIRAS S.A.	10.144.439	14.282,232209
PASSAREDO TRANSPORTES AÉREOS S.A.	3.665.057	11.264,250263
TWO TÁXI AÉREO LTDA.	3.426.548	14.532,916866
MODERN TRANSPORTE AÉREO DE CARGA S.A.	1.843.708	2.950,516526
MAP TRANSPORTES AÉREOS LTDA.	1.155.101	3.338,533464
TOTAL LINHAS AÉREAS S.A.	964.627	2.232,299926
ASTA LINHAS AÉREAS LTDA (EX - AMÉRICA DO SUL LINHAS AÉREAS LTDA.)	210.066	844,633297
CONNECT LINHAS AÉREAS S.A. (ANTIGA CONNECT TÁXI AÉREO LTDA.)	183.634	277,799963
OMNI TÁXI AÉREO S.A.	168.344	451,365951
SIDERAL LINHAS AÉREAS LTDA.	4.174	7,583333

3.2.3 Query 3 - Flight Routes

Find all the routes in the database (hence, for year 2020). Also, count how many times they have been done.

```
SELECT
  nm_aerodromo_origem AS origin_airport,
  iata_airport_origem AS IATA_origin,
  nm_aerodromo_destino AS destination_airport,
  iata_airport_destino AS IATA_destination,
  COUNT(*) AS route_count
FROM
  bd2.flights
GROUP BY
  id_aerodromo_origem,
  nm_aerodromo_origem,
  iata_airport_origem,
  id_aerodromo_destino,
  nm_aerodromo_destino,
  iata_airport_destino
ORDER BY
  route_count DESC;
```

origin_airport	IATA_origin	destination_airport	IATA_destination	route_count
CONGONHAS	CGH	SANTOS DUMONT	SDU	6.592
SANTOS DUMONT	SDU	CONGONHAS	CGH	6.569
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	SALGADO FILHO	POA	3.907
SALGADO FILHO	POA	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	3.880
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	GUARARAPES - GILBERTO FREYRE	REC	3.129
GUARARAPES - GILBERTO FREYRE	REC	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	3.049
DEPUTADO LUÍS EDUARDO MAGALHÃES	SSA	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.932
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	DEPUTADO LUÍS EDUARDO MAGALHÃES	SSA	2.929
SANTOS DUMONT	SDU	PRESIDENTE JUSCELINO KUBITSCHEK	BSB	2.703
TANCREDO NEVES	CNF	CONGONHAS	CGH	2.696
PRESIDENTE JUSCELINO KUBITSCHEK	BSB	SANTOS DUMONT	SDU	2.680
TANCREDO NEVES	CNF	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.677
CONGONHAS	CGH	TANCREDO NEVES	CNF	2.677
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	TANCREDO NEVES	CNF	2.666
PRESIDENTE JUSCELINO KUBITSCHEK	BSB	GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	2.615
GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO	GRU	PRESIDENTE JUSCELINO KUBITSCHEK	BSB	2.606

BRIG. EDUARDO GOMES	ZZ9	JUÍNA	JIA	1
JOÃO DURVAL CARNEIRO	N/I	VIRACOPOS	VCP	1
GLAUBER DE ANDRADE ROCHA	VDC	BAHIA - JORGE AMADO	IOS	1
GLAUBER DE ANDRADE ROCHA	VDC	AEROPORTO ESTADUAL DE JUNDIAÍ	QDV	1

3.3 Conclusions

The queries performed in CQL (Cassandra Query Language) gave the same output of the ones in SQL, so the whole procedure was correct and there was no data loss. Also, the merged table was the same size of the table flights, as expected. A positive aspect of Cassandra is that its queries are really simple and readable, but columnar databases, including Cassandra, are often optimized for analytical workloads and may not be as well-suited for transactional workloads with complex relationships and dependencies, since they do not support a relational schema with foreign keys and join tables, being more flexible on the data structure.

4 Neo4j

In this new section, I will answer the same analytical questions of Section 2 by using the last NoSQL technology, Neo4j.

4.1 Entities and Properties

In Neo4j, data modeling involves representing entities as nodes and their relationships as edges between nodes. First of all, I decided to design the entities along with their respective properties as follows:

Entity	Properties
Airport	id_aerodromo, sg_icao, sg_iata, nm_aerodromo, nm_municipio, nm_pais, nm_continente
Line	id_tipo_linha, cd_tipo_linha, ds_tipo_linha, ds_natureza_tipo_linha, ds_servico_tipo_linha
Equipment	id_equipamento, sg_equipamento_icao, ds_modelo
Company	id_empresa, sg_empresa_icao, sg_empresa_iata, nm_empresa, ds_tipo_empresa
Flight	id_basica, hr_partida_real, dt_partida_real, nr_semana_partida_real, hr_chegada_real, dt_chegada_real, nr_semana_chegada_real, ds_matricula, lt_combustivel, nr_assentos_ofertados, kg_payload, km_distancia, nr_horas_voadas, nr_velocidade_media

4.2 Database Schema

With graph databases technologies, relationships between entities are captured as edges between nodes. In the context of this dataset, I decided to define five different relationships, which can be seen in the database schema:

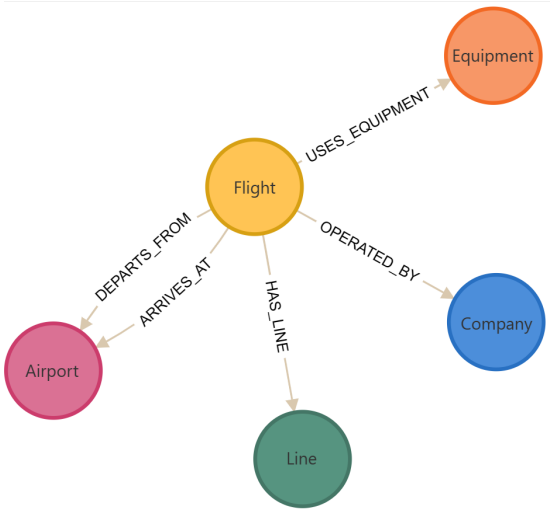

Figura 4.1: Database schema.We can spot both entities and their relationships.

4.3 Analytical Questions

4.3.1 Query 1 - Airplane stats

For each plane model, print some informations as it's avg speed, avg liters consumed etc.

```
MATCH (f:Flight)-[u:USES_EQUIPMENT]->(e:Equipment)
WHERE f.lt_combustivel <> 0 and f.nr_horas_voadas <> 0 and f.kg_payload <>
    0 and f.nr_velocidade_media <> 0 and f.km_distancia <> 0
RETURN e.ds_modelo as airplane_model,
    count(*) as flight_count,
    AVG(f.nr_velocidade_media) AS avg_speed,
    AVG(f.lt_combustivel) as avg_liters,
    AVG(f.nr_horas_voadas) as avg_hours_per_flight,
    AVG(f.km_distancia)as avg_distance_in_km,
    AVG(f.kg_payload) as avg_payload_in_kg
ORDER BY flight_count DESC
```

airplane_model	flight_count	avg_speed	avg_liters	avg_hours_per_flight	avg_distance_in_km	avg_payload_in_kg
"AIRBUS A320-100/200"	90084	5.614.916.973.047.380	5.540.864.359.930.810	2.033.974.606.755.900	12.200.160.516.851.000	17.251.418.575.995.800
"BOEING 737-800"	86355	5.442.169.626.541.450	5.681.545.978.808.450	2.063.063.911.747.990	12.105.085.055.873.900	19800.0
"EMBRAER 195/ERJ-190-200"	43362	4.689.565.144.596.680	3.167.183.132.696.830	13.261.827.006.826.200	649.552.280.798.855	11.047.270.213.551.000
"AEROSPATIALE/ALENIA ATR 72 201/202"	36999	29.417.187.734.803.600	12.641.716.803.156.800	13.666.500.287.575.200	4.178.461.039.487.550	6.691.595.394.470.140
"AIRBUS A319"	25125	46.435.187.582.089.800	3.701.905.910.447.740	1.399.986.783.482.590	6.958.427.860.696.480	1.639.986.224.875.620
"BOEING 737-700"	22659	50.087.996.160.465.900	4.080.470.100.180.930	16.938.170.158.435.800	912.016.152.522.179	15947.0
"AIRBUS A321-100/200"	17567	5.955.180.053.509.420	7.993.255.592.872.970	23.254.919.467.752.100	14.585.913.929.527.000	2.688.125.251.892.760
"CESSNA 208 CARAVAN"	9839	2.307.051.793.881.490	27.613.192.397.601.300	15.356.083.117.186.600	3.677.003.760.544.760	6.470.647.423.518.700
"BOEING 767-300"	8119	6.708.796.654.760.450	23.037.891.612.267.400	4.488.512.387.239.800	32.154.755.511.762.500	4.670.123.685.182.890
"EMBRAER E195-E2/ERJ-190-400"	7466	479.375.885.346.905	2.052.154.701.312.620	1.404.786.141.976.960	6.979.501.741.226.900	13.892.929.279.399.900
"EMBRAER 190/ERJ-190-100"	3123	4.812.896.861.991.680	3.353.823.567.082.940	14.270.786.593.019.500	7.089.705.411.463.310	10.622.019.852.705.700
"AIRBUS A321NEO"	2071	5.925.988.121.680.340	54.346.045.388.701	21.737.647.841.622.300	13.829.521.970.062.700	22.941.070.980.202.800
"AIRBUS A330-200"	1980	7.313.566.717.171.710	3.922.087.979.797.980	5.881.826.494.949.500	440.312.626.262.625	4.513.781.111.111.100
"BOEING 777-300ER PAX"	1748	7.723.882.208.237.990	7.202.058.237.986.260	8.236.441.808.924.470	6.582.127.574.370.710	6.380.561.041.189.920
"BOEING 737-400"	1623	6.253.545.717.806.530	8.221.869.993.838.570	31.004.107.430.683.900	20.127.184.226.740.500	1.827.856.438.693.770
"BOEING 727-200"	1565	4.317.505.623.003.180	5.517.285.623.003.190	11.681.682.523.961.600	5.260.127.795.527.140	22.402.277.955.271.500
"AIRBUS A350-900"	1168	8.015.943.578.767.120	7.058.168.578.767.130	9.484.660.085.616.420	7.694.487.157.534.240	52309.0
"AEROSPATIALE/ALENIA ATR 42-300 / 320"	1032	36.150.270.833.333.300	10.147.877.906.976.700	1.449.596.299.418.600	5.346.366.279.069.770	4.536.431.201.550.390
"AEROSPATIALE/ALENIA ATR 42-500"	894	35.821.852.348.993.200	10.317.505.592.841.100	13.762.676.454.138.700	5.023.948.545.861.290	4.697.384.787.472.040
"AIRBUS A330-900NEO"	793	7.611.149.432.534.680	22.168.583.858.764.200	7.635.729.445.145.020	5.897.075.662.042.880	44.548.615.384.615.400
"BOEING 737-300"	535	6.134.531.588.785.050	8.581.874.766.355.130	28.896.883.607.476.600	18.439.214.953.271.000	17.037.942.056.074.700

4.3.2 Query 2 - Airline stats

For each company, display their total distance traveled and total hours of flight.

```
MATCH (f:Flight)-[d:OPERATED_BY]->(c:Company)
RETURN c.nm_empresa,
    SUM(f.km_distancia) AS total_distance,
    SUM(f.nr_horas_voadas) as total_hours
ORDER BY total_distance DESC
```

c.nm_empresa	total_distance	total_hours
"TAM LINHAS AÉREAS S.A."	139666212	22.652.341.872.900.900
"GOL LINHAS AÉREAS S.A. (EX- VRG LINHAS AÉREAS S.A.)"	125625460	2.175.648.337.770.090
"AZUL LINHAS AÉREAS BRASILEIRAS S/A"	116701938	21.948.565.327.091.000
"ABSA - AEROLINHAS BRASILEIRAS S.A."	10144439	14.282.232.208.999.900
"PASSAREDO TRANSPORTES AÉREOS S.A."	3665057	1.126.425.026.300.000
"TWO TÁXI AÉREO LTDA."	3426548	14.532.916.866.000.100
"MODERN TRANSPORTE AÉREO DE CARGA S.A."	1843708	29.505.165.259.999.900
"MAP TRANSPORTES AÉREOS LTDA."	1155101	33.385.334.639.999.900
"TOTAL LINHAS AÉREAS S.A."	964627	22.322.999.259.999.900
"ASTA LINHAS AÉREAS LTDA (EX - AMÉRICA DO SUL LINHAS AÉREAS LTDA.)	210066	8.446.332.970.000.000
"CONNECT LINHAS AÉREAS S.A. (ANTIGA CONNECT TÁXI AÉREO LTDA.)"	183634	27.779.996.300.000.000
"OMNI TÁXI AÉREO S.A."	168344	4.513.659.509.999.990
"SIDERAL LINHAS AÉREAS LTDA."	4174	7.583.333

4.3.3 Query 3 - Flight Routes

Find all the routes in the database (hence, for year 2020). Also, count how many times they have been done.

```
MATCH (a_origem:Airport)<-[d:DEPARTS_FROM]-(f: Flight)-[a:ARRIVES_AT]->(
    a_destino:Airport)
RETURN a_origem.nm_aerodromo AS origin_airport,
    a_origem.sg_iata as IATA_origin,
    a_destino.nm_aerodromo AS destination_airport,
    a_destino.sg_iata as IATA_destination,
    COUNT(*) AS route_count
ORDER BY route_count DESC
```

origin_airport	IATA_origin	destination_airport	IATA_destinatior	route_count
"CONGONHAS"	"CGH"	"SANTOS DUMONT"	"SDU"	6592
"SANTOS DUMONT"	"SDU"	"CONGONHAS"	"CGH"	6569
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"SALGADO FILHO"	"POA"	3907
"SALGADO FILHO"	"POA"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	3880
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"GUARARAPES - GILBERTO FREYRE"	"REC"	3129
"GUARARAPES - GILBERTO FREYRE"	"REC"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	3049
"DEPUTADO LUÍS EDUARDO MAGALHÃES"	"SSA"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	2932
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"DEPUTADO LUÍS EDUARDO MAGALHÃES"	"SSA"	2929
"SANTOS DUMONT"	"SDU"	"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	2703
"TANCREDO NEVES"	"CNF"	"CONGONHAS"	"CGH"	2696
"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	"SANTOS DUMONT"	"SDU"	2680
"CONGONHAS"	"CGH"	"TANCREDO NEVES"	"CNF"	2677
"TANCREDO NEVES"	"CNF"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	2677
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"TANCREDO NEVES"	"CNF"	2666
"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	2615
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	2606
"CONGONHAS"	"CGH"	"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	2551
"AFONSO PENA"	"CWB"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	2550
"PRESIDENTE JUSCELINO KUBITSCHEK"	"BSB"	"CONGONHAS"	"CGH"	2548
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"AFONSO PENA"	"CWB"	2534
"SANTOS DUMONT"	"SDU"	"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	2407
"GUARULHOS - GOVERNADOR ANDRÉ FRANCO MONTORO"	"GRU"	"SANTOS DUMONT"	"SDU"	2371
...				
"LEITE LOPES"	"RAO"	"MINISTRO VICTOR KONDER"	"NVT"	1
"GOVERNADOR JOSÉ RICHA"	"LDB"	"MINISTRO VICTOR KONDER"	"NVT"	1
"AEROPORTO ESTADUAL DE JUNDIAÍ"	"QDV"	"MINISTRO VICTOR KONDER"	"NVT"	1
"MAESTRO WILSON FONSECA"	"STM"	"TROMBETAS"	"TMT"	1
"JOÃO CORREA DA ROCHA"	"MAB"	"TROMBETAS"	"TMT"	1
"MONTE ALEGRE"	"MTE"	"ORIXIMINÁ"	"ORX"	1

4.4 Conclusions

The conclusions drawn from the Neo4j queries were consistent with those from MySQL and Cassandra. One significant advantage of Neo4j is its ability to efficiently handle complex relationships and traverse graph structures, making it particularly suitable for scenarios where data relationships are the main focus of the problems to solve. Also, I think that the querying process is one of the most intuitive ones, making it very simple to accomplish. Furthermore, the nature of this database alignes with the graph structures particularly well. The only drawback I found when dealing with this technology was the data insertion process, which took a lot of time to execute. Overall, it's my favourite technology so far.

5 Comparative Analysis

All the three technologies seen in this project have their own use cases and profound differences. In order to make a comparison between them, I'll structure this section in 5 sub-section, namely **use-cases**, **performances**, **querying language**, **flexibility** and **adoption**.

5.1 Use-cases

Depending on the situation, one should use a technology over another.

MySQL is the most famous yet popular database technology, and it's used in relatively small projects, and when the data is well structured in tables. Moreover, its data should be relational and structured, in order to respect the **ACID** properties.

Cassandra, instead, is a Columnar database technology, and it's meant to be used in **distributed** (non-local) scenarios where there are very big amounts of data, in order to perform data analytics, so we will have to make a huge amount of small write operations (such as tweets, posts etc.). Looking at the **CAP Theorem** (see [2]), Cassandra focuses on Availability and Partition Tolerance. However, it also keeps an eye on consistency with Tunable consistency.

Neo4j is used when the focus is the **relationships**. Relational databases are not good in managing relationships (relation = table, not relationships) because they have big infrastructure to support them and a big operation cost (join). Consider that its purpose was meant to be a transactional DB, not specifically for analytics. It's efficient to individuate efficiently nodes, not so efficient in whole-graph analysis.

5.2 Performances

MySQL's performance is influenced by factors such as the size of the dataset, the complexity of queries, and the efficiency of indexing. While it is suitable for small to medium-sized projects, its performance can degrade with large datasets, especially during complex join operations. While MySQL supports joins for querying data across multiple tables, complex relationships and nested structures can be challenging to represent and query efficiently.

Cassandra implements data compression, which not only saves storage costs but also contributes to faster data retrieval due to reduced I/O operations. Moreover, during read and write operations, only the relevant columns are transmitted, leading to more efficient data transfer and reduced network congestion. Also, columnar databases exhibit improved cache locality as they store data belonging to the same column contiguously.
Cassandra of course has also its drawbacks: retrieving specific rows of data may require seeking across multiple disk locations, especially if the data is distributed across different columns. This can result in higher latency for certain types of queries. Moreover, when new data is added to the database, especially in scenarios where columns are frequently updated or inserted, the process may be less efficient compared to row-oriented databases.

Neo4j is known for its ability to efficiently handle complex relationships between data points. It excels in traversing relationships between data points, making it incredibly fast at tasks like pathfinding and graph analytics. While Neo4j performs well with smaller to medium-sized graphs, scaling it for extremely large datasets can be challenging.

5.3 Querying Language

The querying languages used in this project are SQL, CQL and Cypher. All of them are declarative, so we just have to declare what we want to find, and the system automatically uses its algorithms to fulfil the query.

SQL and CQL are almost the same language, but CQL doesn't support a lot of the classical operations (such as joins). This is because on Columnar databases, you don't query on the column values but only query by key. In order to use very well Cassandra, one should start from queries, and then design the data model (schema on read approach).

Cypher, on the other hand, is very different from the traditional approaches. Infact, it's querying language consist in "drawing"the entities and relation: we have to define a "shape" inside our graph. The result is a set of shapes that corresponds to the queried one. This approach is very intuitive if you have a strong base in graph theory.

5.4 Flexibility

When it comes to flexibility, we have different things to be considered. Neo4j is schema free: data does not have to adhere to any convention. Everything is stored as instances. Maximum flexibility implies maximum care to keep everything "clean". For schema flexibility, Neo4j and Cassandra offer more flexible schema designs compared to MySQL, as they allow for dynamic schema evolution and support for varied data structures. Neo4j's graph-based model excels in representing complex relationships, while Cassandra's columnar model accommodates wide rows and sparse data. As data modeling is concerned, Neo4j's graph model is highly flexible for representing interconnected data, making it ideal for scenarios where relationships are a primary concern. MySQL and Cassandra offer more traditional tabular and columnar models, respectively, providing flexibility in different data modeling paradigms.

5.5 Adoption

MySQL is one of the most popular relational database management systems globally, known for its ease of use, reliability, and strong open-source community. MySQL has been adopted by a lot of organizations, ranging from small businesses to large enterprises, across various industries. It is commonly used for web applications, content management systems, e-commerce platforms, business applications, and data analytics.

Cassandra was built by Facebook in 2007, by combining 2 different technologies: Google Bigtable (2006), a columnar database and Amazon Dynamo (2007), a key-value database. Cassandra'storage is column-based, access is key-based. Cassandra has gained adoption primarily in industries that require highly scalable and fault-tolerant distributed databases, such as web applications, IoT (Internet of Things), real-time analytics, and financial services. It is particularly well-suited for applications with large volumes of data and high write throughput requirements.

Neo4j was developed by Neo Technologies in the early 2000s. It is developed entirely in Java, and it is open-source too. It is widely used in social networking, recommendation systems, fraud detection, network and IT operations, life sciences, and knowledge graphs, and it has a very active community.

Bibliography

[1] Agência Nacional de Aviação Civil (ANAC). https://www.anac.gov.br/. Accessed: March 2024.

[2] Eric A Brewer. Towards robust distributed system. 2000, pp. 7–10. URL: https://dl.acm.org/doi/10.1145/343477.343502.