Dimensional modelling and ETL data processing

Yago Estévez Figueiras[1],
Andrea Real Blanco[2],
Francisco Manuel Vázquez Fernández[3]

The objective of this lab practice is to design and implement a dimensional model to support the analysis and prediction of flight delays. In the implementation, the data is organized using a star-schema structure, and an ETL pipeline is used to extract, transform, and load the source data into the data warehouse.

Contents

[1] yago.estevez.figueiras@rai.usc.es
[2] andrea.real@rai.usc.es
[3] franciscomanuel.vazquez.fernandez@rai.usc.gal

1 Software and Requirements

To reproduce the pipeline and populate the data warehouse, the following software and steps are required:

1. For the relational database, install MySQL and MySQL Workbench.
2. Run the provided `create_database.sql` script to generate the model.
3. Install these Python packages (recommended Python version 3.11.9):

   ```
   pip install pandas numpy holidays sqlalchemy pymysql
   ```

4. Finally, run the `pipeline.py` script to execute the ETL process and populate the MySQL database.

2 Data Analysis and Modelling

2.1 Analytical Objectives

For our dimensional model, we decided to focus on the analysis of departure delays since, in practice, an arrival delay is often just a consequence of a departure delay. Specifically, to develop our model we consider the following questions that it will answer.

1. Which were the airlines with the most delayed flights?
2. Which were the origin airports with the most delayed flights?
3. How do holidays affect the delay?
4. Which month/day has the most delays?
5. Which hours have the most delays?

2.2 Dimensional Model Diagram

With the objectives clear, we developed the following model using a star-type schema.

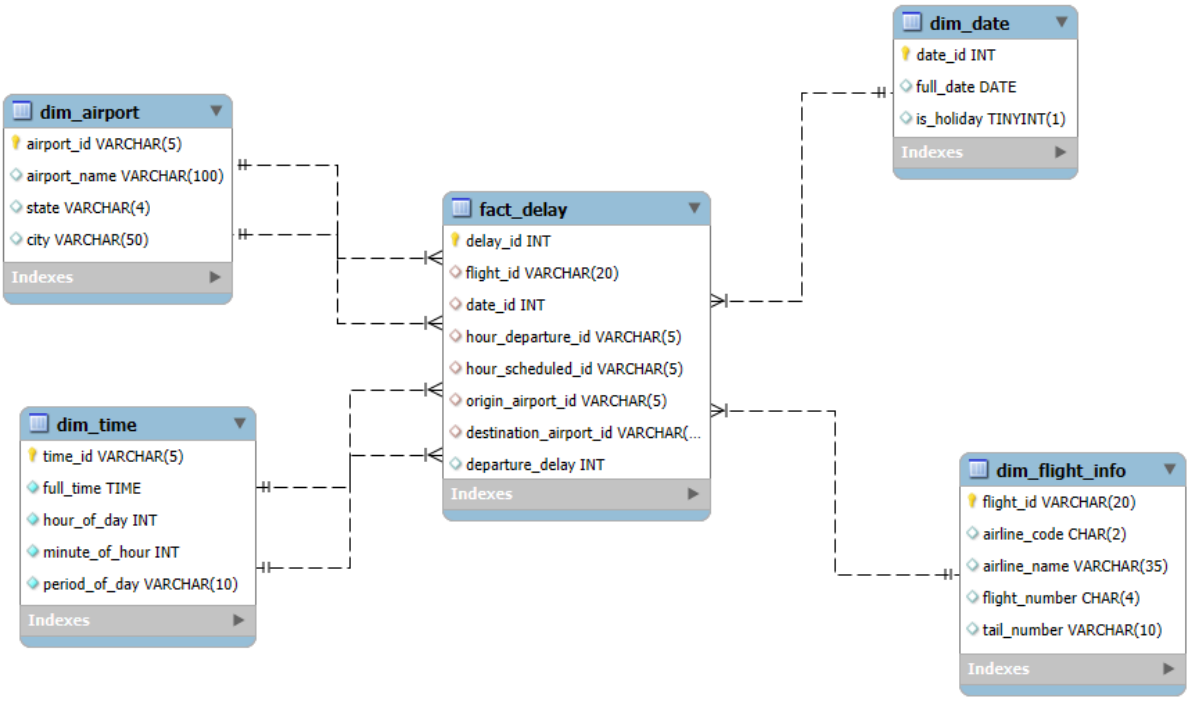

Figure 1: Dimensional star schema for the flight delays data warehouse.

A star schema is a common type of data modelling used in data warehouses, designed for fast querying. It is composed of the fact table and the dimension tables. The fact table contains the quantitative data that can be used for quick analysis, average, counts, sums, etc. The dimension tables surround the fact table giving the schema the star-like shape and have the descriptive attributes in order to provide context to the measures. This attributes are used as filters in our queries to the database. We shall describe our schema as it follows:

- **Fact table:**
  As our fact table we propose 'fact_delay'. It contains a primary key (PK) 'delay_id', an auto increment int in order to get updated with every file loaded. This table also contains the foreing keys (FKs) to establish relationships with the other dimension tables, as it will be explained later. Also the quantity needed for our queries to the database which is the 'departure_delay', an int that measures the delay in a certain flight.
- **Dimension tables:**
  With the intention of filtering the data in our model we defined some dimension tables, as it follows:
  - 'dim_date'
    This table contains the information about dates and if that date was a holiday, the PK 'date_id' in this case is an int generated with the date info as it follows YYYYMMDD (i.e. 20150311 would be the 11th of March from 2015). The table also contains 'full_date' as a date type and 'is_holiday' as a boolean.
  - 'dim_time'
    This table contains information about the time (hour and minute). As our PK 'time_id', we used a varchar(5) with the following format hh:mm (ie. 23:30), the table also include 'full_time' a time-type data, 'hour_of_day' and 'minute_of_hour' as int and 'period_of_day', a varchar(10) which has the information about if that time was morning, afternoon, evening or night.
  - 'dim_flight_info'
    The PK 'flight_id' is a varchar(20) consisting of the airline code plus the flight number plus the tail number, in order to identify each flight individually. This table also contains 'airline_code' as a varchar(2) which is the IATA code for the airline, 'airline_name' as a varchar(35), 'flight_number' as a char(4) and 'tail_number' as a varchar(4)
  - 'dim_airport'
    The PK 'airport_id' is a varchar(5) containing the IATA code of each airport, 'airport_name' is a varchar(100), 'state' is a varchar(4) containing the codes for each US state and 'city' is a varchar(50) with the name of each city that has an airport.
- **Relationships:**
  The FKs used in order to make our relations between the fact table 'fact_delay' and the corresponding dimension:
  - 'flight_id' is the FK related to 'dim_flight_info' 'flight_id'.
  - 'date_id' is the FK related to 'dim_date' 'date_id'.
  - 'hour_departure_id' is the FK related to 'dim_time' 'time_id'.
  - 'hour_scheduled_id' is the FK related to 'dim_time' 'time_id'.
  - 'origin_airport_id' is the FK related to 'dim_airport' 'airport_id'.
  - 'destination_airport_id' is the FK related to 'dim_airport' 'airport_id'.

2.3 Queries to our model:

In order to verify our model we made some queries to the database, those are found in the 'queries.sql' file.

3 Data Pipeline

For populating the data warehouse, we use an ETL pipeline, which is a process wherein data is extracted from various input sources, transformed to ensure consistency and quality, and loaded into the database [1]. This process ensures that the data is clean, structured, and ready for analytical queries, facilitating OLAP (Online Analytical Processing) operations that support the analysis of flight delays.

3.1 Extraction

This extraction step involves reading raw data from the input sources and loading it for processing. In this case, the source data consists of multiple CSV files (`flights.csv`, `airlines.csv`, and `airports.csv`) from the following Kaggle dataset: Predicting flight delays (Tutorial).

The files are loaded into Python using the `pandas` library as DataFrames, which allows for convenient inspection and preliminary cleaning. During this step, key columns are identified, and missing or null values are noted for further handling.

3.2 Transformation

In the transformation step, the raw data is cleaned, standardized, and enriched to prepare it for loading into the data warehouse. The main operations carried out during the process can be classified into the following groups:

1. **Column removal**. Based on the analytical objectives of our model, we focus on analyzing *departure delays* by destination, date, or time. Consequently, columns such as `ARRIVAL_DELAY`, `TAXI_OUT`, or `SECURITY_DELAY` are not required and therefore dropped, along with `COUNTRY`, as all the flights in the dataset take place in the United States.
2. **Data standardization and null handling**. Missing values in `TAIL_NUMBER` are filled with a "000000" placeholder, `FLIGHT_NUMBER` values are converted into fixed-length strings, and `DEPARTURE_DELAY` values are cast from floats to integers, with nulls and negative delays set to 0. Cancelled flights are removed, as they fall outside the scope of our analysis and are the source of all the nulls present in the `SCHEDULED_DEPARTURE` and `DEPARTURE_DELAY` columns. Additionally, some values of `DESTINATION_AIRPORT` and `ORIGIN_AIRPORT` contain numerical entries instead of valid IATA codes; these rows are also removed, as they are not useful for our analytical objectives.
3. **Date and time formatting**. The `YEAR`, `MONTH`, and `DAY` columns are combined into `FLIGHT_DATE` with a `%Y-%m-%d` format suitable for date structures. From this, a `DATE_ID` is derived, which consists of a `%Y%m%d` integer. Departure times, both scheduled and actual, are formatted as `%H:%M` strings. These are later used as foreign keys in the fact table, linking to the time dimension by formatting them as strings we avoid potential issues with database TIME objects.
4. **Derived columns and identifiers**. A unique `FLIGHT_ID` is created by combining the `AIRLINE`, `FLIGHT_NUMBER`, and `TAIL_NUMBER` columns. A boolean `IS_HOLIDAY` column is derived by obtaining all the holidays for a given year in the US through the `holidays` package and comparing them against `FLIGHT_DATE`. In addition, a separate time dimension table is created to classify flights by time-of-day period (morning, afternoon, evening, and night).
5. **Renaming**. Columns are renamed for consistency with those of the data warehouse, e.g. `FLIGHT_DATE` to `full_date`.

3.3 Load

In the final step of the pipeline, the transformed data is loaded into a relational database (in our case, MySQL) that constitutes the data warehouse. The schema follows a star design, where

the central fact table records delay events and is connected to several dimension tables that provide descriptive context (see Figure 1).

Dimensions:

- `dim_date` (`date_id`, `full_date`, `is_holiday`)
- `dim_time`(`time_id`, `full_time`, `hour_of_day`, `minute_of_hour`, `period_of_day`)
- `dim_airport` (`airport_id`, `airport_name`, `state`, `city`)
- `dim_flight_info` (`flight_id`, `airline_code`, `airline_name`, `flight_number`, `tail_number`)

Fact table:

- `fact_delay`, which records each flight's scheduled and actual departure times, origin/ destination airports, and departure delay.

Each dimension has its own primary key (`*_id`), which is referenced as a foreign key in the fact table. In `fact_delay`, for example, `date_id` links to `dim_date`, while `hour_scheduled_id` and `hour_departure_id` both reference `time_id` in `dim_time`. Similarly, `origin_airport_id` and `destination_airport_id` reference `airport_id` in `dim_airport`, and `flight_id` links to `dim_flight_info`.

The data is inserted from Python into the warehouse through the `.to_sql()` method provided by `pandas`, with connection handled by `SQLAlchemy`, which allows specifying column data types and appending records into the tables. An example snippet is shown in Listing 1.

```
df_date[['date_id', 'full_date', 'is_holiday']] \
    .to_sql(
    'dim_date',
    con=engine,
    if_exists='append',
    index=False,
    dtype={
        'date_id': Integer(),
        'full_date': Date(),
        'is_holiday': Boolean()
    }
)
```

Listing 1: Example of loading a dimension table into MySQL using pandas and SQLAlchemy.

This process is repeated for each dimension and finally for the `fact_delay` table, completing the ETL pipeline.

Bibliography

[1] "What Is an ETL Pipeline?." Accessed: Sep. 25, 2025. [Online]. Available: https://www.geeksforgeeks.org/software-testing/what-is-an-etl-pipeline/