Flights Delay Look Up Application

Dong Chen, Weiqing Huang

Introduction

In this project, we are using the 2015 Flight Delays and Cancellations datasets, and we are making an application to search, view, and operate the information about flights.

We chose this dataset because this dataset includes enough information that we can work with and apply functions to that. We hope the data could contain multiple data types including text and numbers. In addition to that, we hope that the data is able to be analyzed on, which means that there are relationships between some attributes, as well as some units of data.

We implemented this application because we want to provide users a platform that can look up delayed flights easily and clearly. Not only presenting the data in text format, we are also implementing some visualizations of the data, to provide users a better experience.

Final Implementation

Description of Use Cases

In this Delayed Flights Search application, we implemented all the functions on the web. Once you start the server, a webpage will pump up and begin our application.
The functions are listed in the drop-down box in the sidebar. There are 7 functions:

1. **View Airline Delay Information**
   In this function, users could look up the delayed flight count and average delayed time of each airline. This will be presented as a barplot, as well as a table.
2. **View Airport Delay Information**
   In this function, users could look up the delayed flight count and average delayed time of the airports that are both as an origin airport and a destination airport. For each airport, the departure delay and arrival delay is calculated separately, and scatter plots against the delayed flight count and average delayed time are presented. In addition to the plots, a table is also presented at the bottom.
3. **Find Flights Information**
   In this function, users can specify an origin airport and a destination airport, and a schedule information of all the flights that flies from the origin airport to the destination airport will be shown up here.
4. **Find Flights Delay Information**

In this function, users can find out delayed information by entering the airline and flight number. For example, if you enter “UA544”, the delayed information of UA544 will be shown.

5. **Insert A New Flight Information**
   In this function, users can insert a new flight information by specifying the airline, flight number, origin airport, destination airport, date, scheduled departure time and scheduled arrival time of the flight. After inserting the flight departure time and flight arrival time, the scheduled flight time would be calculated automatically. The updated scheduled flight time would be shown on the webpage if the user inserted the information correctly. If the flight information already existed, an error message will pump up and the user may not insert this information.
6. **Update Delayed Information**
   In this function, users can update a flight’s delayed information. That is, a delayed flight might have multiple reasons of delay. In this database, it specifies 5 reasons, and for each reason it can store time of delay due to that reason. For some flights, it has a total arrival delay time, but does not include the specific delay time due to different reasons. Therefore, users can update such information by type in the delay time due to different reasons. There is a limit to this update, which is the sum of all the delayed time due to different reasons must be equal to the total arrival delay time. Once the update is successful, the updated information of the flight as well as the historical changes of this flight will be shown upon the page. Otherwise, the total arrival time will appear so that you would know how to insert the correct data.
7. **Delete An Existed Flight**
   In this function, users can delete a flight information by specifying the airline, flight number, origin airport, destination airport, and the date of the flight. If the flight does not exist, it will report an error. If the flight is deleted successfully, the deleted information will be shown on the page as a backup.

Final Database Design

Description of the Data

We get the data from the kaggle: https://www.kaggle.com/usdot/flight-delays. This dataset describes the information about on-time, delayed, cancelled and diverted information about flights in 2015. We use the three main tables from this data and we used normalization techniques to divide these tables separately.

Here are the three tables originally in the dataset. We mainly normalize the database on the flights table.

flights = (YEAR, MONTH, DAY, DAY_OF_WEEK, AIRLINE, FLIGHT_NUMBER, TAIL_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, SCHEDULED_DEPARTURE, DEPARTURE_TIME, DEPARTURE_DELAY, TAXI_OUT, WHEELS_OFF, SCHEDULED_TIME, ELAPSED_TIME, AIR_TIME, DISTANCE, WHEELS_ON, TAXI_IN, SCHEDULED_ARRIVAL, ARRIVAL_TIME, ARRIVAL_DELAY, DIVERTED, CANCELLED, CANCELLATION_REASON, AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY)

airlines = (IATA_CODE, AIRLINE)
airports = (IATA_CODE, AIRPORT, CITY, STATE, COUNTRY, LATITUDE, LONGITUDE)

Set the Candidate Key and Primary Key

First, we can easily find that here are 5819079 rows in this dataset and there are 5819079 distinct rows in MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT column. The origin airport and destination airport are also included because there exists the same airline and flight number flying from different origin airports or to different destination airports. Also, we can see that there are 5817746 distinct rows without these columns. Thus, the candidate key and primary key is and only **MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT**.

Find the Dependency on the Dataset

Take a rough look, we get schedule information, real situation information, time related to this information, diverted information, cancel information and delay information at first and we want to find potential dependency and attribute closure. After digging into the dataset, we find that:

MONTH, DAY -> DAY_OF_WEEK

AIRLINE, FLIGHT_NUMBER -> TAIL_NUMBER:

The dependencies below represent times and the time gap between these times:

The time point that the aircraft's wheels leave the ground – Departure time = The time duration elapsed between departure from the origin airport gate and wheels off

DEPARTURE_TIME, WHEELS_OFF -> TAXI_OUT

The scheduled arrival time – The scheduled departure time = Scheduled time:

SCHEDULED_DEPARTURE,SCHEDULED_ARRIVAL -> SCHEDULED_TIME

The time point that the aircraft's wheels touch the ground - The time point that the aircraft's wheels leave the ground = Air Time:

WHEELS_OFF, WHEELS_ON -> AIR_TIME

The gate arrival at the destination airport - The time point that the aircraft's wheels leave the ground = Taxi In Time:

WHEELS_ON, ARRIVAL_TIME -> TAXI_IN

TAXI_OUT, +AIR_TIME + TAXI_IN = ELAPSED_TIME:

TAXI_OUT, AIR_TIME, TAXI_IN -> ELAPSED_TIME

Departure Time - Planned Departure Time = Departure Delay:

SCHEDULED_DEPARTURE, DEPARTURE_TIME -> DEPARTURE_DELAY

Arrival Time - Planned Arrival Time = Arrival Delay:

SCHEDULED_ARRIVAL, ARRIVAL_TIME -> ARRIVAL_DELAY

SCHEDULED_ARRIVAL, ARRIVAL_TIME , AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY -> ARRIVAL_DELAY

Normalization

Then, we can normalize the dataset based on the information from the reference website and the dependency we find. The dataset doesn't contain any repeating columns and satisfy 1NF. To make it in 2NF, every non-prime attribute of the relation is dependent on the whole of every candidate key. We know the only candidate key is (**MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT**). Thus, we need to split **DAY_OF_WEEK** and **TAIL_NUMBER** above to satisfy 2NF. To make it in 3NF, there must not be any dependency among Non-key attributes (other than Primary Key). We are going to split other columns shown above.

In addition, in order to make the small tables more understandable, I will split more tables according to the category.

At last, we get the following tables. They satisfy 3NF and are lossless.

schedule_info (YEAR, MONTH, DAY, DAY_OF_WEEK, AIRLINE, FLIGHT_NUMBER, TAIL_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, SCHEDULED_DEPARTURE, DISTANCE, SCHEDULED_ARRIVAL)
date_info (MONTH, DAY, DAY_OF_WEEK)
flight_info (AIRLINE, FLIGHT_NUMBER, TAIL_NUMBER)
scheduled_time_info(SCHEDULED_DEPARTURE,SCHEDULED_ARRIVAL,SCHEDULED_TIME)
real_info (MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, DEPARTURE_TIME, WHEELS_OFF, WHEELS_ON, ARRIVAL_TIME)
taxi_out_interval (DEPARTURE_TIME, WHEELS_OFF, TAXI_OUT)
taxi_in_interval (WHEELS_ON, ARRIVAL_TIME, TAXI_IN)
air_time_interval (WHEELS_OFF, WHEELS_ON, AIR_TIME)
elapsed_time_interval (TAXI_OUT, AIR_TIME, TAXI_IN, ELAPSED_TIME)
departure_delay (SCHEDULED_DEPARTURE, DEPARTURE_TIME, DEPARTURE_DELAY)
arrival_delay (SCHEDULED_ARRIVAL, ARRIVAL_TIME, ARRIVAL_DELAY)
diverted_info (MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, DIVERTED)
cancelled_info (MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, CANCELLED, CANCELLATION_REASON)
delay_info (MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, SCHEDULED_ARRIVAL, ARRIVAL_TIME, AIR_SYSTEM_DELAY,SECURITY_DELAY,AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY)

In the entire mega table, there are some problems with some data. A time bug occurs in 09-09-2015 with WHEELS_OFF = '00:01:00', WHEELS_ON= '00:19:00' and AIR_TIME=78. Therefore, about 1% data does not completely satisfy the rules we discussed above. In order to keep the tables originally, we select all columns as primary keys in **scheduled_time_info, taxi_out_interval, taxi_in_interval, air_time_interval, elapsed_time_interval, department_delay, arrival_delay**. In theory, the primary key should be on the left side of the dependency, but this might lead to some data loss. This is the result of a tradeoff.

Database Design

- The **schedule_info** represents the original schedule of the flights including the scheduled time, aircraft number, flight distance and other information.
- The **date_info** is to make the table meet 3NF, indicating the date and day of week. The **flight_info** represents the information of the aircraft.
- The **scheduled_time_info** represents the air time in the flight plan.
- The **real_info** represents the actual flight information of the flight, mainly in the time dimension. The **taxi_out_interval, taxi_in_interval, air_time_interval, elapsed_time_interval, departure_delay, arrival_delay** represent time and time interval.
- The **cancelled_info** indicates whether to cancel.
- The **delay_info** shows the detailed reasons for the aircraft delay. These tables can describe the characteristics of flights in detail with lossless and standardized data type.

The final UML graph is in the appendix.

Description of Data Used for Testing

For data used for testing, we selected all the information in the **airports** and **airlines** tables, and the first 200 rows in the **flights** table. So for the test data in the flights table, it contains 200 rows of information, and we use statistical software to get it from flights.csv. Our demonstration of implemented use cases is also based on these data. In order to run our test cases in the walkthrough successfully, the data files in sample_data folder should be loaded into the database.

Summary of implemented use cases

Generally, the use cases can help users view, search, insert, update, delete the data and connect to the database lively. **View Airline Delay Information** and **View Airport Delay Information** can read the summarized data and view their related plots. **Find Flights Information** and **Find Flights Delay Information** can search for information that users want. **Insert new flight Information** can insert data to the database directly. **Update Delayed Information** can update the delayed information about a delayed flight. **Delete An Existed Flight** can delete a flight information.

Illustration of Functionality

1. **View Airline Delay Information**
   Option on the sidebar: View Airline Delay Information
   Input: N/A
   Output: Bar plot & DataFrame

This function shows a view created in the database side. The view is created by counting the number of delayed flights based on the airline companies as well as the average delayed time (in minutes). In the frontend, a query is written to call the view and present the table in a DataFrame format, as well as a barplot of this table. The bar plot has X axis of airline names and Y axis of number of delayed flights. The color mapped to the column is based on the average delayed time.

2. **View Airport Delay Information**
   Option on the side bar: View Airport Delay Information
   Input: N/A
   Output: 2 Scatter plots & DataFrame
   This function shows a view created in the database side. The view is created by counting the number of delayed flights and calculating average delayed time of each airport. For each airport, the departure delay and arrival delay is calculated separately, and rejoined together according to the airport. In the frontend, a query is written to call the view and present the table in a DataFrame format, as well as 2 scatter plots against the delayed flight count and average delayed time for both departure delayed information and arrival delayed information.

3. **Find Flights Information**
   Option on the side bar: Find Flights Information
   Input: IATA Code of airports. E.g.
   Please select origin airport: SFO
   Please select destination airport: LAX
   Output: A table that contains the flight flies from the specified origin airport to the specified destination airport.

	AIRLINE	FLIGHT_NUMBER	SCHEDULED_DEPARTURE	SCHEDULED_ARRIVAL	SCHEDULED_TIME
0	UA	1224	6:00:00	7:28:00	88

   This function calls a stored procedure in the database side. The stored procedure requires two inputs of the origin airport and the destination airport, and it returns a table that contains the flight that flies from the origin airport to the destination airport. In the frontend side, the procedure is called and the result table is shown up here.

4. **Find Flights Delay Information**
   Option on the side bar: Find Flights Delay Information
   Input: Airline + Flight Number
   Flight Number (e.g. UA544): UA544
   Output: A sentence that shows the airline, flight number, departure delay time (in minutes) and arrival delay time (in minutes).

   Flight UA544, DEPART_DELAY: 22, ARRIVAL_DELAY: 24

   This function calls a stored procedure in the database side. The stored procedure requires two inputs of the airline and the flight number. It returns a dataframe including

the airline, flight number, departure delay time and arrival delay time. The frontend side parsed this dataframe into a sentence as it shows up there. If the flight is not included in our database, an error message will show. If the flight number does not show in a correct format, an error message will show.

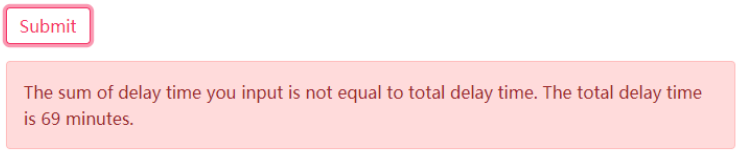
5. **Insert A New Flight Information**
   Option on the side bar: Insert A New Flight Information
   Input: AIRLINE, FLIGHT_NUMBER, YEAR, MONTH, DAY, ORIGIN_AIRPORT, DESTINATION_AIRPORT, SCHEDULED_DEPARTURE, DISTANCE, SCHEDULED_ARRIVAL
   Example Input: UA, 289, LAX, JFK, 2015, 1, 2, 00:50:00, 1416, 3:15:00
   This information should be pre-entered in the textbox.
   Then hit "Submit".
   Output: A success or error message depends on your input. If success, a table with airline, flight number and the calculated scheduled time will be shown.

	AIRLINE	FLIGHT_NUMBER	SCHEDULED_TIME
0	UA	289	145

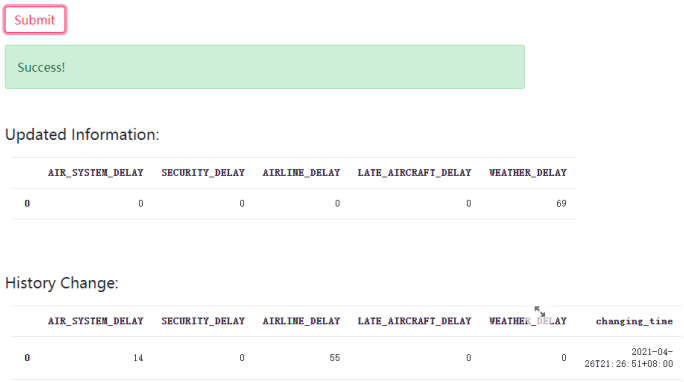
   This function calls a stored procedure in the database side. The procedure requires an input of the airline, flight number, origin airport, destination airport, date, scheduled departure time and scheduled arrival time of the flight. The procedure will check if the flight information already existed or not. If it is, the insert will not be successful. There is also an INSERT trigger connected to this procedure. After inserting the flight departure time and flight arrival time, the scheduled flight time in minutes would be calculated automatically by minus the departure time from the arrival time, as well as inserting this information to the schedule time table. In the frontend side, a "Success" message will be raised and the updated scheduled flight time would be shown on the webpage if the user inserted the information correctly. If the flight information already existed, an error message will be raised and the insert will not take place.

6. **Update Delayed Information**
   Option on the side bar: Update Delayed Information
   Input: MONTH, DAY, AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY
   Example Input: AA, 2299, JFK, MIA, 2015, 1, 1, 0, 0, 0, 0, 69
   This information should be pre-entered in the textbox. The last 5 input can be adjusted as long as their sum is 69 (just for this flight). If the sum is not 69, an error message will pump up.
   Then hit "Submit".

Output: A success or error message depends on your input. If success, a table with airline, flight number and the calculated scheduled time will be shown.
Error



Success



This function calls a stored procedure in the database side. The procedure requires an input of the flight information, as well as the time of 5 delayed reasons. For each delayed fight, the sum of all the delayed time due to different reasons must be equal to the total arrival delay time. If it is not, the update will not be successful. There is also an UPDATE trigger related to this procedure. For every update, the trigger to record the change and store it in a backup table. In the frontend side, it will show the historical changes according to this flight, as well as the updated delay information once the update is successful. If not, an error message will pump up.

7. **Delete An Existed Flight**
   Option on the side bar: Update Delayed Information
   Input: AIRLINE, FLIGHT_NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, YEAR, MONTH, DAY
   Example Input: UA, 1224, SFO, LAX, 2015, 1, 1
   This information should be pre-entered in the textbox.
   Then hit “Submit”.
   Output: A success or error message depends on your input. If success, a table of the deleted information will be shown as a backup.
   Error:

Success:

	original_YEAR	original_MONTH	original_DAY	original_AIRLINE	original_FLIGHT_NUMBER	original_ORIGIN_AIRPORT
0	2015	1	1	UA	1224	SFO

This function calls a stored procedure in the database side. It requires an input of the airline, flight number, origin airport, destination airport, and the date of the flight. There is also a DELETE trigger connected to this procedure. Once a delete takes place, it will store the deleted information into a backup table. In the frontend side, if the flight does not exist, it will report an error. If it deleted the flight successfully, the deleted information will be shown on the page as a backup.

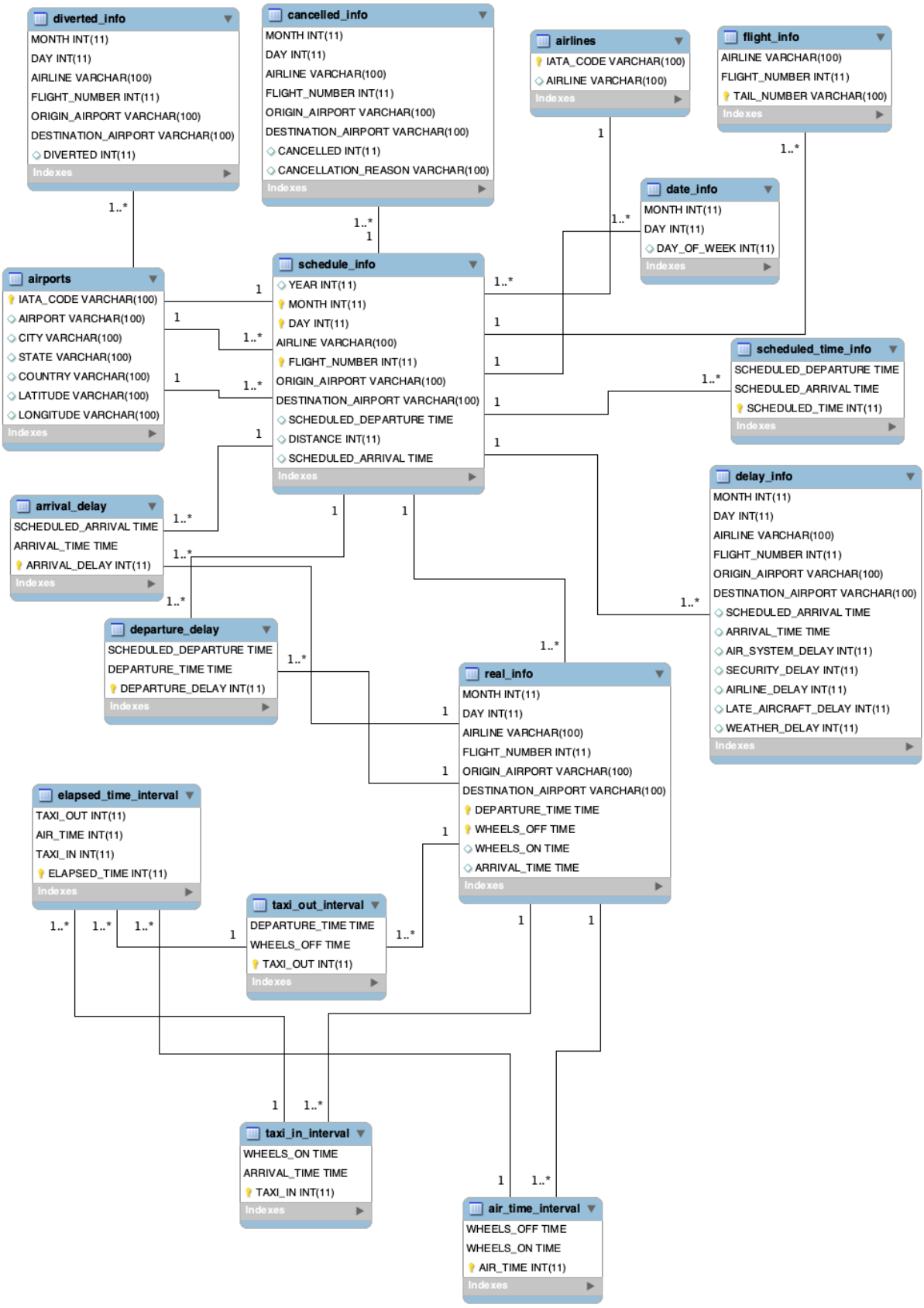
Summary Discussion

We have completed the development of this data set, and the functions we want to implement are available.

I think the main difficulty is that the amount of data is too large. There are a total of 5,819,079 rows and 600MB of data, so it is difficult to run a query. The data is very closely related, we build many small tables showing the time interval. In the end we have more than a dozen tables. If we want to change one table, we may have to change many tables. It is also difficult to join these tables back. In addition, the data set itself has some errors, such as the time interval is not equal to the difference between the two times. This also brings a lot of challenges for us to regulate data types and import data. We want to retain the original data as much as possible, so we made some changes to the database design after considering the tradeoff between the efficient design and the complete data.

We split our work half and half generally. We selected the data set, determined the research direction, and wrote the report together. For individuals, Dong is responsible for the setup and normalize of the database. Weiqing is responsible for the establishment of the front and the interactive interface. For 7 functions and their use cases, we implemented the related stored procedures, triggers and views together. We think this is reasonable and is a fair split because of our undergraduate background and skills, and we help each other’s work when someone is left behind.

Appendix: UML Graph


diverted_info
MONTH INT(11)
DAY INT(11)
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
ORIGIN_AIRPORT VARCHAR(100)
DESTINATION_AIRPORT VARCHAR(100)
DIVERTED INT(11)
Indexes
cancelled_info
MONTH INT(11)
DAY INT(11)
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
ORIGIN_AIRPORT VARCHAR(100)
DESTINATION_AIRPORT VARCHAR(100)
CANCELLED INT(11)
CANCELLATION_REASON VARCHAR(100)
Indexes
airlines
IATA_CODE VARCHAR(100)
AIRLINE VARCHAR(100)
Indexes
flight_info
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
TAIL_NUMBER VARCHAR(100)
Indexes
date_info
MONTH INT(11)
DAY INT(11)
DAY_OF_WEEK INT(11)
Indexes
airports
IATA_CODE VARCHAR(100)
AIRPORT VARCHAR(100)
CITY VARCHAR(100)
STATE VARCHAR(100)
COUNTRY VARCHAR(100)
LATITUDE VARCHAR(100)
LONGITUDE VARCHAR(100)
Indexes
schedule_info
YEAR INT(11)
MONTH INT(11)
DAY INT(11)
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
ORIGIN_AIRPORT VARCHAR(100)
DESTINATION_AIRPORT VARCHAR(100)
SCHEDULED_DEPARTURE TIME
DISTANCE INT(11)
SCHEDULED_ARRIVAL TIME
Indexes
scheduled_time_info
SCHEDULED_DEPARTURE TIME
SCHEDULED_ARRIVAL TIME
SCHEDULED_TIME INT(11)
Indexes
arrival_delay
SCHEDULED_ARRIVAL TIME
ARRIVAL_TIME TIME
ARRIVAL_DELAY INT(11)
Indexes
delay_info
MONTH INT(11)
DAY INT(11)
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
ORIGIN_AIRPORT VARCHAR(100)
DESTINATION_AIRPORT VARCHAR(100)
SCHEDULED_ARRIVAL TIME
ARRIVAL_TIME TIME
AIR_SYSTEM_DELAY INT(11)
SECURITY_DELAY INT(11)
AIRLINE_DELAY INT(11)
LATE_AIRCRAFT_DELAY INT(11)
WEATHER_DELAY INT(11)
Indexes
departure_delay
SCHEDULED_DEPARTURE TIME
DEPARTURE_TIME TIME
DEPARTURE_DELAY INT(11)
Indexes
real_info
MONTH INT(11)
DAY INT(11)
AIRLINE VARCHAR(100)
FLIGHT_NUMBER INT(11)
ORIGIN_AIRPORT VARCHAR(100)
DESTINATION_AIRPORT VARCHAR(100)
DEPARTURE_TIME TIME
WHEELS_OFF TIME
WHEELS_ON TIME
ARRIVAL_TIME TIME
Indexes
elapsed_time_interval
TAXI_OUT INT(11)
AIR_TIME INT(11)
TAXI_IN INT(11)
ELAPSED_TIME INT(11)
Indexes
taxi_out_interval
DEPARTURE_TIME TIME
WHEELS_OFF TIME
TAXI_OUT INT(11)
Indexes
taxi_in_interval
WHEELS_ON TIME
ARRIVAL_TIME TIME
TAXI_IN INT(11)
Indexes
air_time_interval
WHEELS_OFF TIME
WHEELS_ON TIME
AIR_TIME INT(11)
Indexes