CSDS341 Project - Airline Querying System - Final Report

Luke Zhang (rxz330), Jiamu Zhang (jxz1217), Quynh Nguyen (qtn2)

2022-07-23

Contents

Introduction

In recent decades, the growing demand for leisure and business travel has led to the prosperity of the airline market. An increasing number of people have been choosing to take flights to travel domestically or internationally. Therefore, an organized and comprehensive database that stores the airline system is critical for both travelers and crew to obtain plenty of simultaneous information.

Although there do exist several flight databases or applications for commercial airlines, it is rare to find comprehensive information - including weather at the departure airport and destination, aircraft type, the total flight hours of pilots, and the number of luggage allowed - in just one database. This information offers travelers a chance to be better prepared for traveling.

Since our airline querying system contains a relatively extensive data sets, the crew members who choose to use our database are able to access the basic information about the travelers who will be on their flight and provides updates about the airline information.

Database High-Level Design

Assumptions

In order to simulate the real-world situation of a complex flight system, we need to design our database based on the following assumptions. These assumptions can also be used as references for all data constraints.

1. Assume that there have and only have two types of users of the airline querying system: travelers and crew.

2. Assume that plane ticket information is stored in the database system and each ticket is only valid for one traveler. However, a traveler may own zero or more plane tickets.

3. Assume that each ticket contains a specific seat location for exactly one flight. However, a flight may have multiple tickets being sold to travelers.

4. Assume that a crew member can be either an air attendant or a pilot. Therefore, a crew member can serve zero or more flights. Additionally, a flight must be served by at least one crew member. It does have a slight chance that a small propeller airplane only needs one crew member (i.e. the pilot).

5. Assume that a flight is operated by exactly one aeroplane. For example, the aircraft with registration number B-6075 is operating a specific flight (flight number: CA862) from Beijing(PEK) to Geneva(GVA). However, it is likely that one aeroplane can fly multiple flights. Notice that the registration number is unique for each aeroplane.

6. Assume that an aeroplane can only belong to one company. Additionally, a airline company can have multiple planes.

7. Assume that each airline company must have at least one airport as its hub, a place where the headquarter of the company locates and where the aeroplanes get maintenance and repaired. However, some large airports can provide services for multiple airline companies. For example, Los Angeles International Airport (LAX) is a hub for both United and Delta Airlines, and Delta Airlines has another hub : Detroit Metropolitan Airport (DTW).

8. Assume that each flight can have multiple schedules, and a schedule can be mapped to multiple flights. It is common for most domestic flights to have the same flight flying the same route on two successive days. There is also a slight chance that two flights have the exact same schedule.

9. Assume that each flight only departures from exactly one airport and only arrives at exactly one airport. However, an airport can have many flights.

Entity-Relationship Model

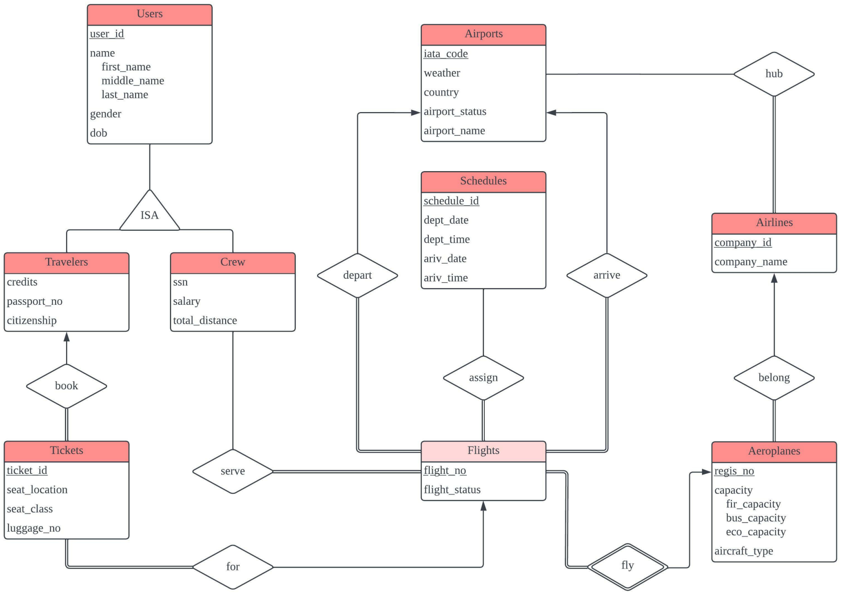

Figure 1: CSDS341 Airline Database ER-Diagram

Data Description

Our database consists of seven entities, eight relationships, and one identifying relationship. Notice that some tables and relationships are merged together to reduce redundancy.

`Travelers` and `Crew` are considered as users of our airline user system. `Travelers` contains each travelers' information such as `credits`, `citizenship`, etc. `Crew`, sharing part of the attributes with `Travelers`, also has its unique attributes such as `ssn` and `salary`. Note that each user should be either a Traveler or a Crew. A user cannot be both or none, which means that if a `user_id` has already been stored in `Travelers` table, it will not be present in the `Crew` Table.

Relational Schema (SQL Language)

Strong Entities

Table of Travelers:

```
CREATE TABLE IF NOT EXISTS Travelers (
    user_id INT NOT NULL,
    first_name VARCHAR(50),
    middle_name VARCHAR(50),
    last_name VARCHAR(50),
    gender CHAR(1),
    dob DATE,
    credits INT DEFAULT 0,
    passport_no VARCHAR(20),
    citizenship VARCHAR(30),
    PRIMARY KEY (user_id),
    CHECK (gender IN ('M' , 'F', 'U')));
```

Table of Crew:

```
CREATE TABLE IF NOT EXISTS Crew (
    user_id INT NOT NULL,
    first_name VARCHAR(50),
    middle_name VARCHAR(50),
    last_name VARCHAR(50),
    gender CHAR(1),
    dob DATE,
    ssn INT NOT NULL,
    salary DOUBLE,
    total_distance INT,
    PRIMARY KEY (user_id),
    CHECK (gender IN ('M' , 'F', 'U')));
```

Table of Airports:

```
CREATE TABLE IF NOT EXISTS Airports (
    iata_code CHAR(3) NOT NULL,
    airport_name VARCHAR(100),
    country CHAR(2),
    weather VARCHAR(30),
    airport_status VARCHAR(30),
    PRIMARY KEY (iata_code),
    CHECK (weather IN ('Sunny' , 'Mostly Sunny', 'Partly Cloudy',
        'Cloudy', 'Rainy', 'Heavy Rainy', 'Foggy',
        'Snowy', 'Heavy Snowy', 'Frost')),
    CHECK (airport_status IN ('Free', 'Normal',
        'Busy', 'Small-Scale Delay', 'Large-Scale Delay')));
```

Table of Airlines:

```
CREATE TABLE IF NOT EXISTS Airlines (
    company_id INT NOT NULL,
    company_name VARCHAR(50),
    PRIMARY KEY (company_id));
```

Table of Schedules:

```
CREATE TABLE IF NOT EXISTS Schedules (
    schedule_id INT NOT NULL,
    dept_date DATE,
    dept_time TIME,
    ariv_date DATE,
    ariv_time TIME,
    PRIMARY KEY (schedule_id));
```

Weak Entity

Table of Flights_ariv_dept:

```
CREATE TABLE IF NOT EXISTS Flights_ariv_dept (
    regis_no VARCHAR(10) NOT NULL,
    flight_no VARCHAR(7) NOT NULL,
    flight_status VARCHAR(10),
    dept_iata_code CHAR(3) NOT NULL,
    ariv_iata_code CHAR(3) NOT NULL,
    PRIMARY KEY (regis_no, flight_no),
    FOREIGN KEY (regis_no)
        REFERENCES Aeroplanes_belong(regis_no),
    FOREIGN KEY (dept_iata_code)
        REFERENCES Airports(iata_code),
    FOREIGN KEY (dept_iata_code)
        REFERENCES Airports(iata_code),
    CHECK (flight_status IN (’On-Time’ , ’Delay’, ’Cancel’)));
```

Relationships

Table of Aeroplanes_belong:

```
CREATE TABLE IF NOT EXISTS Aeroplanes_belong (
    regis_no VARCHAR(10) NOT NULL,
    fir_capacity INT,
    bus_capacity INT,
    eco_capacity INT,
    aircraft_type VARCHAR(50),
    company_id INT NOT NULL,
    PRIMARY KEY (regis_no),
    FOREIGN KEY (company_id)
        REFERENCES Airlines(company_id));
```

Table of Tickets_book_for:

```
CREATE TABLE IF NOT EXISTS Tickets_book_for (
    ticket_id INT NOT NULL,
    seat_location CHAR(4),
    seat_class CHAR(1),
    luggage_no INT,
    regis_no VARCHAR(10) NOT NULL,
    flight_no VARCHAR(7) NOT NULL,
    traveler_id INT NOT NULL,
    PRIMARY KEY (ticket_id),
    FOREIGN KEY (traveler_id)
        REFERENCES Travelers(user_id),
    FOREIGN KEY (regis_no, flight_no)
        REFERENCES Flights_ariv_dept(regis_no, flight_no),
    CHECK (seat_class IN (’F’ , ’B’, ’E’)));
```

Table of serve:

```
CREATE TABLE IF NOT EXISTS serve (
    crew_id INT NOT NULL,
    regis_no VARCHAR(10) NOT NULL,
    flight_no VARCHAR(7) NOT NULL,
    PRIMARY KEY (crew_id , regis_no , flight_no),
    FOREIGN KEY (crew_id)
        REFERENCES Crew(user_id),
    FOREIGN KEY (regis_no, flight_no)
        REFERENCES Flights_ariv_dept(regis_no, flight_no));
```

Table of assign:

```
CREATE TABLE IF NOT EXISTS assign (
    regis_no VARCHAR(10) NOT NULL,
    flight_no VARCHAR(7) NOT NULL,
    schedule_id INT NOT NULL,
    PRIMARY KEY (regis_no , flight_no , schedule_id),
    FOREIGN KEY (regis_no, flight_no)
        REFERENCES Flights_ariv_dept(regis_no, flight_no),
    FOREIGN KEY (schedule_id)
        REFERENCES Schedules(schedule_id));
```

Table of hub:

```
CREATE TABLE IF NOT EXISTS hub (
    company_id INT NOT NULL,
    iata_code CHAR(3) NOT NULL,
    PRIMARY KEY (company_id , iata_code),
    FOREIGN KEY (company_id)
        REFERENCES Airlines(company_id),
    FOREIGN KEY (iata_code)
        REFERENCES Airports(iata_code));
```

Note: In order to prevent redundancy for table `Tickets_book_for`, since both the book and for relationships are one-to-many relationships, they are merged with the entity on many side which is `Tickets`. This table includes three different foreign keys, such as `traveler_id` from the `book` relationship; `regis_no` and `flight_no` from `for` relationship. Additionally, `Tickets` involves in both `for` and `book` relationship as total participation, while `Travellers` (in `book` relationship) and `Flights` (in `for` relationship) are partial relationship.

Primary Key Constraints

1. The primary key attribute of the entities `Travelers` and `Crew*` are both `user_id` which is distinct for each one of the travelers, so each user can be uniquely identified by his/her `user_id`.

2. The entity `Airport` stores the information of airports and has a primary key attribute `iata_code` which stands for the unique three-character International Air Transport Association Code for each airport.

3. The primary key attribute of the `Airlines*` is `company_id` which stores the information of airline companiesis and is unique for each airline.

4. The primary key attribute of the `Schedules` is `schedule_id` which is unique for each schedule.

5. The primary key attribute of the Aeroplanes_belong is regis_no.
6. The primary key attribute of the Flights_ariv_dept is regis_no and flight_no.
7. The primary key attribute of the Tickets_book_for is ticket_id.
8. The primary key attribute of the serve is crew_id, regin_no and flight_no.
9. The primary key attribute of the assign is regis_no, flight_no and schedule_id.
10. The primary key attribute of the hub* is company_id and iata_code.

Overlap Constraint

A User can be either Travelers or Crew, but they share the same primary key attribute user_id, so when a Travelers or a Crew is created, they have their unique user_id.

Foreign key constraints (please also check the schema for more details):

1. The Tickets_book_for schema has three foreign-key constraints: (1)traveler_id references the primary key of the book relationship which is Travelers.user_id. (2)regin_no references the primary key of the for relationship which is Aeroplanes_belong.regis_no. (3)flight_no reference the primary key of the for relationship which is Flights_ariv_dept.flight_no.
2. The Aeroplanes_belong schema has one foreign-key constraint: attribute company_id references the primary key of the Airlines which is Airlines.company_id.
3. The Flights_ariv_dept schema has three foreign-key constraints: (1)regis_no references the primary key of the Aeroplanes_belong relationship which is Aeroplanes_belong.regis_no. (2)dept_iata_code references the primary key of the Airports entity which is Airports.iata_code. (3)ariv_iata_code reference the primary key of the Airports entity which is Airports.iata_code.
4. The serve schema has three foreign-key constraints: (1)crew_id references the primary key of the Crew entity which is Crew.user_id. (2)regis_no references the primary key of the Aeroplanes_belong relationship which is Aeroplanes_belong.regis_no. (3)flight_no reference the primary key of the Flights_ariv_dept entity which is Flights_ariv_dept.flight_no.
5. The assign schema has three foreign-key constraints: (1)‘chedule_ireferences the primary key of theSchedulesentity which isSchedules.schedule_id. (2)regis_noreferences the primary key of theAeroplanes_belongrelationship which isAeroplanes_belong.regis_no. (3)flight_noreference the primary key of theFlights_ariv_deptrelationship which isFlights_ariv_dept.fligh
6. The hub schema has three foreign-key constraints: (1)company_id references the primary key of the Airlines entity which is Airlines.company_id. (2)iata_code references the primary key of the Airports entity which is Aeroplanes_belong.regis_no.

Mapping cardinalities (please also check the ER-Diagram for more details):

1. book in Tickets_book_foris a many-to-one relationship set from Tickets to Travelers, meaning that a travelers can book multiple tickets for flights.
2. for in Tickets_book_for is a many-to-one relationship set from Tickets to Flights, meaning that a flight can have many tickets.
3. serve is a many-to-many relationship set from Crew to Flights, meaning that several crews can work on one flights, and one crew can work on several flights.

4. depart and arrive in Flights_ariv_dept are many-to-one relationship set from Flights to Airports, meaning that several flights can depart from the same airports.
5. assign is a many-to-one relationship set from Flights to Schedules, meaning that several different flights can have the same schedule of event.
6. fly is a many-to-one relationship set from Flights to Aeroplanes, meaning that several different flights can correspond to one planes.
7. belong is a many-to-one relationship set from Aeroplanes to Airlines, meaning that several planes can have the same airline.
8. hub is a many-to-many relationship set from Airlines to Airports, meaning that airline companies can have several of their hub airports, and airports can be the hub airport of several different airlines companies.

Participation Constraints (please also check the ER-Diagram and Schema for more details):

1. Tickets must have a total participation in the book and for relationship set, because every travelers must have at least one ticket, and every tickets must at least for one flight.
2. Flights is a weak entity and totally participates in identifying relationship set fly with Aeroplanes, since a flight must at least corresponds to one plane.
3. Flights must have a total participation in the depart, arrive and assign relationship set, because an airport cannot have no flight for departure or arrival, and the existence of a schedule must have at least on flights relates to.
4. Flights must also a total participation in serve, because there should be at least a crew member to take care of a flight.
5. Aeroplanes must have a total participation in the belong relationship set, since each one of the planes should relate to their airlines company.
6. Airlines must have a total participation in the hub relationship set, meaning that there should be at least one airline company that the airport is the hub of its planes.

Functional Dependencies

Example Queries

Implementation

Roles of members

What we learned from this project

GUI Interface