Name: Han Hoang
Unix ID: dbs042
PSID: 2088807

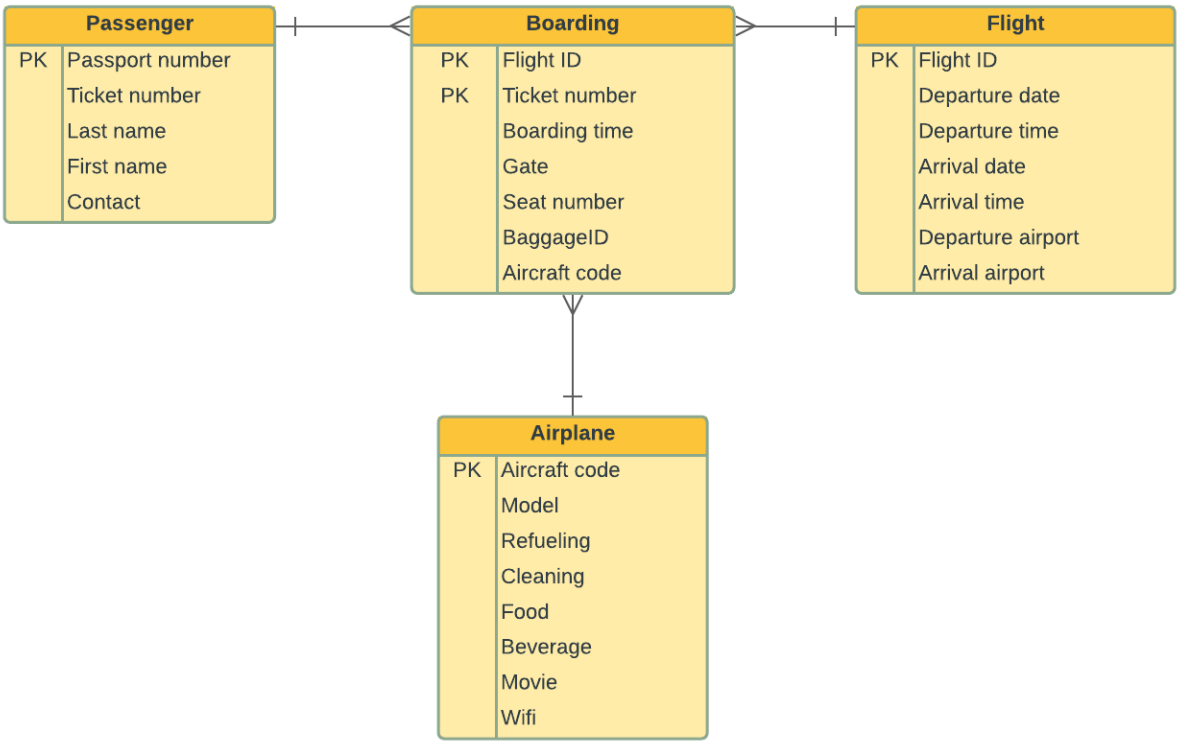
AIRLINE SUMMARY REPORT

Table of contents

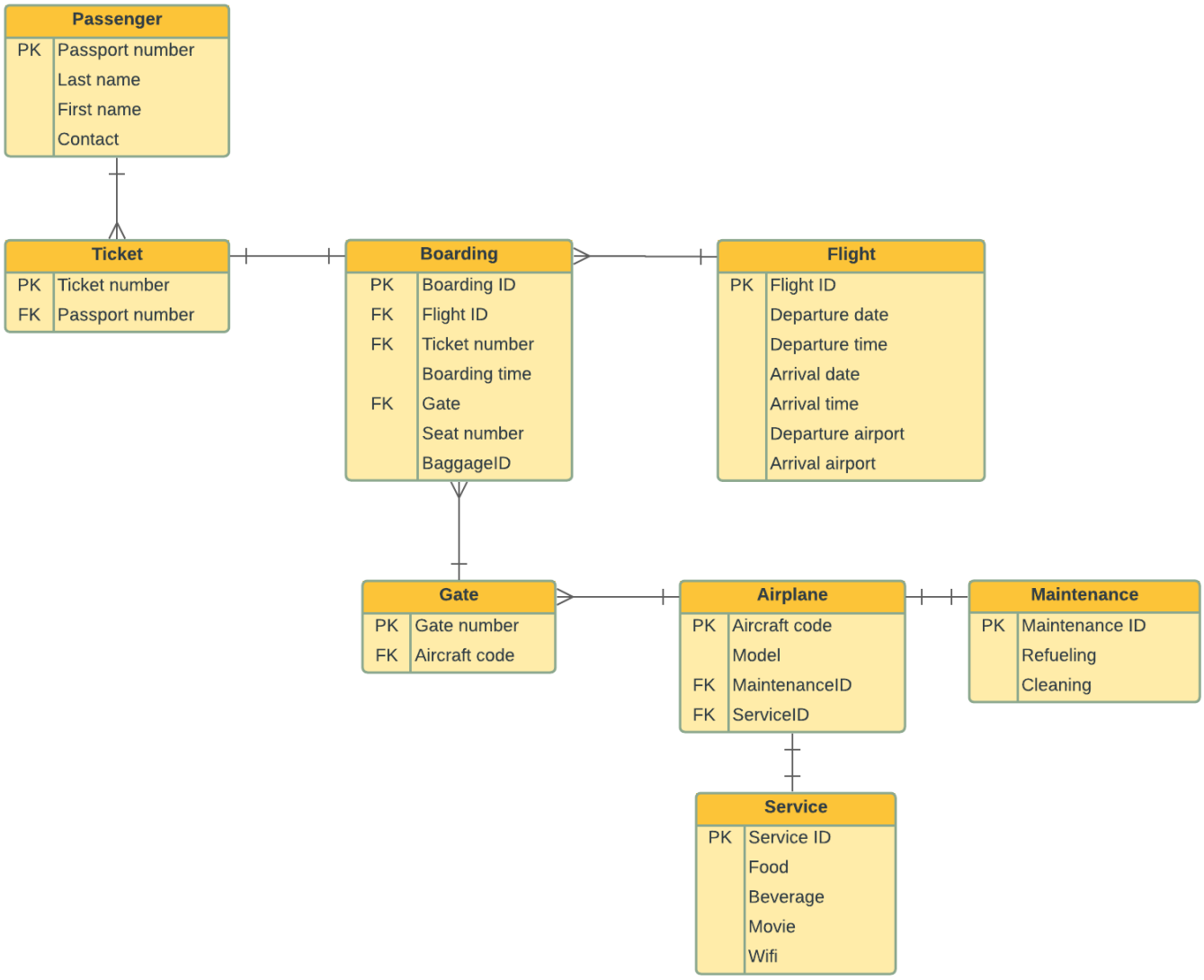
ER DIAGRAM

Initial Phase

In this diagram, there are 4 main tables: Passenger, Boarding, Flight and Airplane. Here, the Boarding table works as a bridge table with a composite key of Flight ID and Ticket number to avoid a many-to-many relationship between Passenger table and Flight table. Besides that, all four tables are in First Normal Form with atomic attributes and no repeating groups. However, except for the Flight table, the other 3 tables all need further normalization.



Final Phase


Passenger
PK Passport number
Last name
First name
Contact
Ticket
PK Ticket number
FK Passport number
Boarding
PK Boarding ID
FK Flight ID
FK Ticket number
Boarding time
FK Gate
Seat number
BaggageID
Flight
PK Flight ID
Departure date
Departure time
Arrival date
Arrival time
Departure airport
Arrival airport
Gate
PK Gate number
FK Aircraft code
Airplane
PK Aircraft code
Model
FK MaintenanceID
FK ServiceID
Maintenance
PK Maintenance ID
Refueling
Cleaning
Service
PK Service ID
Food
Beverage
Movie
Wifi

Normalization

- **Passenger:** The initial Passenger will cause modifying anomalies since for every new ticket a passenger purchased, all four attributes Passport number, Last name, First name and Contact are repeating. Therefore, the Passenger table is broken down into tables Passenger with attributes (Passport number, Last name, First name, Contact) and Ticket with attributes (Ticket number, Passport number).
- **Boarding:** The initial Boarding table passes 2NF but violates 3NF because the aircraft code (non-prime key) is dependent on the gate (non-prime key). Therefore, the Aircraft code attribute is removed from the Boarding table and added to the Gate table with attributes (Gate number, Aircraft code) so the Boarding table in the final ER diagram can pass 3NF.
- **Airplane:** The Airplane table also needs normalizing to avoid modifying anomalies, especially the deletion anomaly. For example, if you want to delete an attribute in a row, this will cause loss of other attributes. Therefore, the initial table is broken down into 3 normalized tables: Airplane (Aircraft code, Model, Maintenance ID, Service ID), Maintenance (Maintenance ID, Refuelling, Cleaning) and Service (Service ID, Food, Beverage, Movie, Wifi)

ER cardinality explanation

- A passenger can buy many tickets
- A ticket can be bought by only one passenger
- A ticket can have only one boarding information
- A flight can have many boarding passes
- Each boarding occurs at one gate
- One gate can have many boarding
- A gate can have only one airplane parks there
- An airplane can go to many gates
- Each airplane has one maintenance information and one service information

SQL SUMMARY

1. Database

With the ER diagram, we can easily map all the entities from the diagram to the tables in the database using CREATE and fill the data using INSERT. Notice that, when creating tables, there must be constraints for primary key, foreign keys corresponding to the ER diagram and reasonable data types for each attribute.

Example: Creating table Flight

```
CREATE TABLE flight (
    flight_id character(5) NOT NULL,
    departure_date DATE,
    departure_time TIME,
    arrival_date DATE,
    arrival_time TIME,
    departure_airport character(3),
    arrival_airport character(3),
    CONSTRAINT flights_check CHECK (((arrival_date > departure_date) OR ((arrival_date =
departure_date) AND (arrival_time > departure_time)))),
    PRIMARY KEY (flight_id)
);
```

In this table, the columns created are flight_id (primary key), departure_date, departure_time, arrival_date, arrival_time, departure_airport, arrival_airport. Besides the constraint for primary key, I also added the constraint flights_check for checking departure/arrival date and time and this will be used to validate input in the web application later.

2. Queries

These queries are SELECT statements. They do not affect tables but only retrieve data from the tables when users are making requests to the database such as view or search.
Main queries:

- View all rows within a table
  “SELECT * FROM table_name”
- Search for some data by an ID input by user
  “SELECT * FROM table_name WHERE id = $1”, [user_input]

3. Transaction

Transactions are “packages” containing one or more queries that could manipulate the database. Which commonly are *INSERT*, *UPDATE* and *DELETE* statements. Each time a user inserts, updates or deletes any data, transactions begin with *BEGIN TRANSACTION*, packaging all the data manipulation language. If no error occurs, the database follows with *COMMIT TRANSACTION* and new data is implemented.

- Add new information
  BEGIN TRANSACTION
  INSERT INTO table_name (column_1, column_2, column_3, ...)
  VALUES ($1,$2,$3,...)", [value_1, value_2, value_3...]
  COMMIT TRANSACTION

- Edit existed information
  BEGIN TRANSACTION
  “UPDATE table_name
  SET column_1 = $1, column_2 = $2, column_3 = $3..
  WHERE id = $4", [value_1, value_2, value_3,..., user_input]
  COMMIT TRANSACTION

- Delete information
  BEGIN TRANSACTION
  "DELETE FROM table_name WHERE id = $1", [user_input]
  COMMIT TRANSACTION

DEMO VIDEO

Link: https://youtu.be/9MxbZRGW0Js

WEB APPLICATION OVERVIEW

The Flight Management System web application has 3 tabs: Flight, Boarding and Airplane.

- In the Flight tab, the web application allows users to view, create, update any information about flights and delete flight information that is not referenced by other tables.
- In the Boarding tab, users can do the same operations with the boarding information as in the Flight tab such as view, create, update and delete. Besides that, for each boarding ID, you can also view gate-aircraft information (which aircraft code is at which gate number) in this page.
- Finally in the Airplane tab, in addition to viewing, creating and updating the airplane information, another feature in this tab is viewing maintenance and service information of the airplanes.

In terms of input validation, when creating the database, constraints for primary key, foreign keys and data types constraints for each attribute were also specified. Therefore, data to enter the input fields are restricted to follow the constraints. Whenever a user enters an invalid input that does not match the data type or primary key constraints, the server will catch an error and render a pop-up alert in the same page telling the user that the input is missing/invalid and resetting the form for the user to re-enter data.