Simple Airline Management System (SAMS)

CS 4400: Introduction to Database Systems
Course Project: Fall 2022 Semester

Version History

Version	Date	Notes
0	January 28, 2023	Initial release
1	March 3, 2023	Added data type clarifications and images into the document

Scenario Description & Motivation

The following is a text description of the system you are being tasked to develop. The system requirements – explicit and implicit – are included this document, and they need to be identified and reflected in your Enhanced/Extended Entity-Relationship Diagram (EERD).

We've observed several disruptions to air travel in the United States over the past few weeks. And many of these disruptions have been related – in some significant part – to computing challenges:

The FAA is continuing a thorough review to determine the root cause of the Notice to Air Missions (NOTAM) system outage. Our preliminary work has traced the outage to a damaged database file. At this time, there is no evidence of a cyber attack. The FAA is working diligently to further pinpoint the causes of this issue and take all needed steps to prevent this kind of disruption from happening again.

https://www.faa.gov/newsroom/faa-notam-statement

Lyn Montgomery, the president of Transport Workers Union Local 556, which represents Southwest Airlines flight attendants, said she had spoken Tuesday with Pete Buttigieg, the transportation secretary, to discuss the breakdown at Southwest. She said that Southwest's technology was a major cause of the meltdown and that her union had long pressed the company's leaders to improve it.

https://www.nytimes.com/2022/12/27/business/southwest-flights-canceled-travel.html

As of noon E.T. more 6,988 flights into, within or out of the country had been delayed, and just over 1,100 have been canceled altogether, according to data from the tracking site FlightAware. ... A total of 21,464 flights were scheduled to depart U.S. airports on Wednesday with a carrying capacity of nearly 2.9 million passengers, Reuters reported, citing data from aviation analytics company Cirium.

https://www.npr.org/2023/01/11/1148340708/faa-notam-ground-stop-flight-delay

We have tremendous respect for the members of the FAA, Southwest Airlines, and any other organization that has the responsibility of keeping these systems running as smoothly as possible, which is clearly not as easy task. And you're not going to be asked to solve all these technical challenges during this project. **You will, however, be asked to develop, implement, and test a system to keep track of how airlines use airplanes flown by pilots to move passengers between different airports.** There are two more key sections of the document below:

- Problem Requirements: This section describes the main elements of the "airline management" mini-world, and contains the entities, attributes, and relationships that you must include in your design model.
- Sample Data Elements: This section includes some data values that you can use to help determine if your design model (i.e., EERD) is correct, consistent, comprehensive, and concise as reasonably possible.

Problem Requirements

The primary aim of our system is to track the overall status of passenger-based commercial aircraft across a general region with multiple airports. This means that it will have to manage the data related to various types of entities – namely, the airplanes, airports, and other things relevant to our scenario. The single-most important resource in our scenario is people. We will track people in our system as users who are acting in various capacities. People can be aircraft pilots, or passengers riding on one or more flights, or possibly both. All people must be either pilots or passengers – our system will not keep track of people acting in any other types of roles. All people must have distinct identifiers in this system. People might also have first and last names. First names will be required, and some celebrities might travel using only their first name (e.g., Madonna, Sade, Beyonce).

Attributes that are used to identify entities in our system will normally consist of fifty (50) or fewer alphanumeric characters in some regular pattern/format. This will be the default format for entity-identifying and "entity unique" attributes in our system unless otherwise noted. Dates will be represented in the "yyyy-mm-dd" date format by default unless otherwise noted. Some users might have relatively long first and/or last names, so we must ensure that we can manage first and last names that have one hundred (100) or fewer characters. Our default size for storing "general purpose" descriptive attributes will be one hundred (100) or fewer characters unless otherwise noted. Please note that these "early" data type specifications might be superseded once we've received more sample data from the customer in later phases of the project.

The main "drivers" in our scenario are the airlines. An airline is an organization that helps to manage and/or provide flights for passengers. Some airlines own one or more airplanes that can be used to accomplish these tasks. Some other airlines don't own any planes, but instead operate in a "virtual/shadow" mode: for example, coordinating with other airlines that do own airplanes, and providing funds to invest in the other airline while collecting some of the resulting revenues. Each airline has a unique identifier.

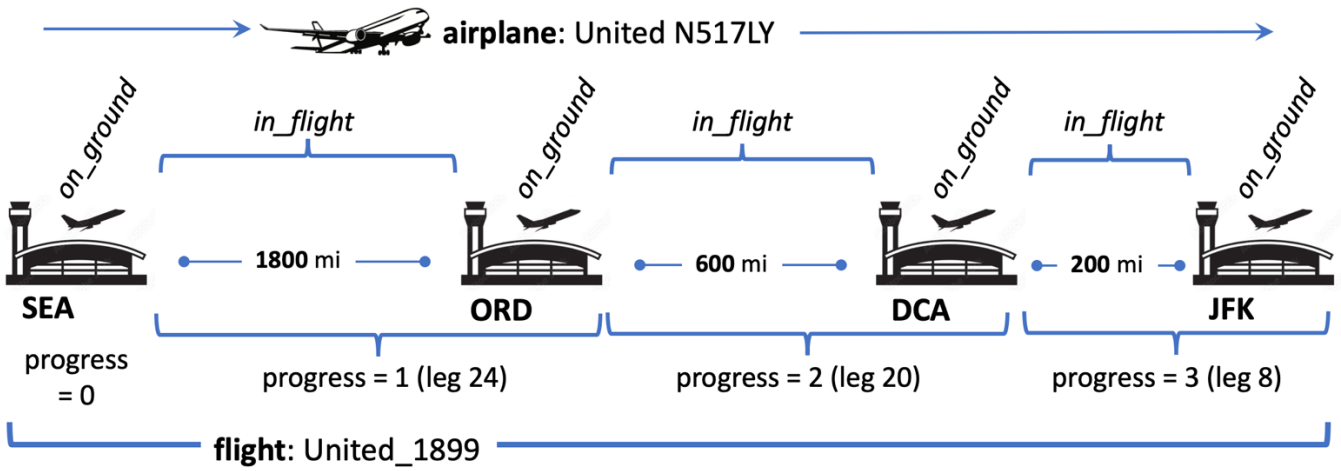
Airplanes are the craft used to transport passengers between airports. Airplanes can be identified uniquely by the combination of the airline that owns them and their tail number. Airplanes come in various types and capabilities, but the main distinction that we will make are between propeller-driven airplanes and jet-driven airplanes. For example, the speed of the airplane is important in determining the time needed to determine the flight time, and jet-driven planes tend to travel faster than propeller-driven planes. Jet-driven planes tend to have more seats (i.e., passenger carrying capacity) than propeller-driven planes. Meanwhile, propeller-driven planes are sometimes more versatile – for example, some have skids that allow them to land on water, which is not true for the jet-driven planes in our system. We also need to track the number of propellers or engines for propeller-driven or jet-driven planes, respectively. Each airplane can be classified as a propeller-driven or jet-driven airplane exclusively. And we're also experimenting with a very small number of new airplanes that are neither propeller- nor jet-driven, with extremely high speeds and a relatively small number of seats for certain special passengers. Regardless of the airplane type, each airplane must have a valid speed and seating capacity.

Airplanes deliver passengers from one "departure" airport to a different "arrival" airport. Airports have a three letter identifier, along with a longer, more human-readable name – for example, "ATL" represents the Hartsfield-Jackson Atlanta International Airport. Each airport is located in some city and state. We need to track the location of each airport to determine which airports can be used to serve the same geographical regions. And we will use the standard two-letter abbreviations for U.S. states and territories.

Airplanes will support flights on behalf of the airlines. Each flight follows a specific route from some starting airport to some ending airport. Some flights are "non-stop", and go simply from starting airport to the ending airport. Other flights might follow a specific sequence of multiple legs, where the airplane lands and then takes off again at various intermediate airports. We must track the distance for each leg of a flight – whether it's non-

stop or multi-stop – so that we can determine the total duration of the flight based on the speed of the supporting airplane.

new As an example of routes and flights, *consider the **eastbound_north_milk_run** route*. This route has three legs and originates in Seattle, Washington. The first leg takes you to Chicago, Illinois; the second leg takes you to Washington, D.C.; and the third leg takes you to New York, New York. The distances for the legs are 1800 miles, 600 miles, and 200 miles, respectively. The legs are also labeled for easy reference and reuse – legs #24, #20 and #8, respectively. When keeping track of flight status, the flight always starts in progress state 0. In this situation, the airplane supporting this flight is "on_ground" at the departing airport (**SEA**) for the first leg. Once the airplane takes off, then it is "in_flight" towards the first leg destination airport (**ORD**), and the progress state advances to 1. When that flight lands, the progress state remains at 1, but the airplane's status reverts to the "on_ground" value. This pattern of landing and takeoff transitions continues until the flight lands at the airport at the end of the route – in this case, the John F. Kennedy Airport (**JFK**) in New York. This is *the route currently flown by **United Flight 1899***, which is *supported by the **airplane United N517LY** (tail number)*.



The pilots in our system are qualified to fly the airplanes. Propeller-driven airplanes must have at least one assigned pilot when they are supporting a flight, while jet-driven airplanes must have a minimum of two pilots. An airplane does not require pilots be assigned if it's not supporting a flight. We will only track that a pilot is part of the team that is flying the plane – we're not concerned with their specific role such as Captain, First Officer, Flight Engineer, etc. Each pilot can have one or more licenses that confirm their flight skills. And a pilot gains experience for each leg of a flight when they are assigned as a member of the flight team. We must also keep track of each pilot's tax identifier to ensure that they are paid appropriately for their services. The tax-identifier will be stored using a "*xxx-xx-xxxx*" format.

When an airplane is assigned to support a specific flight, we must track the status of the airplanes progress along the flight's designated route. More specifically, we must track which leg of the flight is currently being traversed; whether the airplane is on the ground waiting to takeoff, or is in the air waiting to land; and when the next takeoff or landing will occur.

People can also purchase tickets for various flights as passengers or pilots. When a ticket is purchased, it will have an identifier, a specific cost, and a list of one or more seat numbers. The ticket will have some representation for the flight being taken, and the person who purchased the flight. Pilots will sometimes be given "$0"/zero-dollar tickets just for seat tracking purposes. Tickets permit people to board a flight at any airport along the route, but the ticket must explicitly state the airport where that passenger will deplane. More specifically, passengers might board and later deplane a flight at any of the valid airports along that flight's

route, and consistent with the passenger's ticket. We must also track the number of miles earned by each passenger so that we can provide frequent flyer rewards appropriately.

Whether a person is waiting in an airport for their next flight, or seated on an airplane waiting it to reach its destination, each person will always be associated with one of those valid types of locations. We will not track people at other locations, such as traveling to/from the airport from home, staying in local hotels, etc.

Our system must also be able to display information that will help us keep track of the flow of airplanes and passengers throughout the system. Each airplane has a maximum capacity of the number of people that be flown, so the system must be able to determine how many people are on each airplane. Similarly, the system must be able to determine how many people are at each airport. The main procedure for our system is that airplanes will depart from airports and then land at airports in time-sequential order. The database "simulation" state will be configured initially to represent a 24-hour day of travel. As each airplane arrives at the next airport along its flight route, passengers who are due to depart the airplane at that stop will deplane the craft, and passengers at that airport who are scheduled to take that flight will board the aircraft. Finally, the airplane will take off again towards the next airport along its flight route until it reaches its final destination airport.

new As an example of the simulation, consider the normal progression of flights as discussed above. ***For each flight, there is a next_time value that indicates when that flight will "logically change state"*** for our simulation purposes. If an airplane is in flight, then the next_time value indicates when the flight will land at the destination airport of the current leg. If an airplane is on the ground, then the next_time value indicates when the airplane will take off towards the arrival airport of the following leg. So, *the simulation cycle can be executed by* ***finding the "next flight" that will change its state chronologically*** – more specifically, that will land or take off – before the other flights based on having the smallest next_time value. *If the next flight is* ***landing***, then we must allow the flight to land, and allow passengers to disembark if their ticket destination matches the current airport. *If the next flight is* ***taking off***, then we must allow passengers to board the flight, and then allow for the flight to take off (if permitted). And, for the case where *the airplane is already* ***at the final airport*** *in the route*, we must recycle the crew so that they can leave the plane, get some rest, and prepare for their next flight. We must also remove the flight once it has reached the end of its route.

[1] **flight_landing()** & [2] **passengers_disembark()** [5] **passengers_board()** & [6] **flight_takeoff()**

[3] **recycle_crew()** & [4] **retire_flight()** [*if – and only if – this is the final airport on the route*]

Simulation Cycle Steps

Sample Data Elements

After consulting with the customer, **we've provided an even larger and more complete dataset in the denormalized_data spreadsheet**. While using the sample data was recommended but not required during Phase 1, here you must enter all the given data into your database as a Phase 2 requirement. If there are elements of the data that can't be represented in an appropriate attribute, entity, or relationship, then perhaps you need to review your design. Similarly, if there are attributes, entities, relationships, etc. that haven't been used after you've stored all the data, then perhaps your design has unnecessary elements. This exercise doesn't guarantee that your EERD is fully correct, but it does offer some validation that you are on the correct track.