ECS650U/ECS789P - SEMI-STRUCTURED DATA AND ADVANCED DATA MODELLING

**CW2: MongoDB Database Development and Performance Tuning**

Group 32

**Assumptions:**

During the development of our system, we made several assumptions. I’ve listed the assumptions we made for each collection below.

**General**

* We created our own custom ids for each collection to facilitate testing relating to the correctness of various queries.
* We deleted the pilot’s collection (mentioned in cw description) and created 1 collection for all AirlineEmployees

**JourneyBooking**

* Additional passengers will be the number of passengers rather than each passengers’ detailed information
* Paying Passenger is the person who made the booking, not included in “addtlPassengers”

**PlaneFlights**

* Departure airport can’t be the same as arrival airport.

**AirlineEmployees**

* Salaries will be recorded as an annual salary in £GBP.
* Employees will have a home address consisting of postcode, street name, house number and city
* Employees will have contact info consisting of an email address and mobile number.
* We assume the airline employs pilots, clerks, maintenance staff and cabin staff.
* Only AirlineEmployees with the position “Pilot” will have a datetime for fitDate.

**Planes**

* serviceLength will be recorded in hours e.g., 5.5hours would be 5hours and 30minutes of service.
* Planes will be serviced every 400-600 hours.
* flyRange will be recorded in kilometres.
* Seat capacity includes flight crew.

**Airports**

* Airport costs will be recorded in £GBP hourly.
* Airport locations will be city names.
* We will calculate the monthly airport cost using the hourly cost, which we will then use in our revenue calculation.

**Diagram:**

**Graphical user interface, application, Word

Description automatically generated**

**Collections:**

JourneyBooking has a one-to-many relationship with PlaneFlights. This means that one journey will have many flights, as it might be a booking with several transitions. The booking passenger will be saved by him/herself and can buy more than one ticked at once which will be saved to “addtlPassengers”. “flightsUsed” is where all the transitions and flights used in the booking will be stored.

**JourneyBooking**

* id int
* timeOfBooking datetime
* payingPassenger varchar
* addtlPassenger int
* flightsUsed [planeFlights\_id]

AirlineEmployees will have a many-to-one relationship with PlaneFlights, this is because there are two employees that will be chosen for PlaneFlights (Pilot and Co-Pilot). Each Employee will have to include their detailed personal information. All employees will have an address and contactInfo which are nested documents. AnnualSalary is found in this collection which will be used to solve for revenue. Although every employee will have fit-to-fly as field, but it will be null unless the employee is in a pilot position.

**AirlineEmployees**

* id int
* fullName varchar
* position varchar
* annualSalary float
* address document
  + postcode varchar
  + streetName varchar
  + houseNo int
  + city varchar
* contactInfo document
  + email varchar
  + mobileNo varchar
* fitDate datetime

Planes have a one-to-one relationship with PlaneFlights, meaning one plane can have one flight. Planes will include the details of the plane, and its maintainability. It includes the serviceLength that will be a float, it increases per use and will be checked before each use to make sure it does not go over the allowed flight hours before it needs servicing, which will be every 400-600 flight hours (standard servicing length). ServiceLength will also be reset to 0 after each service. It has a status field so the employees know where or if the plane is available at all times.

**Planes**

* id int
* make varchar
* model varchar
* flyRange float
* serviceLength float
* status varchar
* seatCapacity int

Airports will have two one-to-many relationships. This is because one airport is connected to many flights as its starting airport and its destination airport. The collection will include the Airport information including the cost of utilizing the airport which is stored per hour, and can be multiplied by the number of hours it is at the airport per month to calculate the revenue.

**Airports**

* id int
* airportName varchar
* airportLocation varchar
* airportCost document
  + parkPrice float
  + refuelPrice float

PlaneFlights is connected to all the tables created, all the relationships can be seen above. The flights will have a planeId that is the primary key of the plane used for transportation, it will have the starting and destination airports which will be the ids of the airport collection. Pilots will be chosen from the AirlineEmployee collection.

**PlaneFlights**

* id int
* planeId int
* airports document
  + startAirport airportId
  + destAirport airportId
* flightPilots document
  + departTime datetime
  + arrivTime datetime
* flightDistance float

**Queries:**