Homework Question 3, Gradescope Q3.3

Document Store Design for Flightapp

Overview

In a document store like AsterixDB, data is stored as JSON-like documents, which allow for nested structures and complex data types. This design supports a flexible and hierarchical data model suitable for Flightapp.

Aggregate Types (Classes)

1. **Flight Document**
    - **Key**: `flight_id`
    - **Structure**:
        - flight_id: unique identifier for the flight
        - carrier: nested object containing:
            - carrier_id: unique identifier for the carrier
            - carrier_name: name of the carrier
        - origin: origin city of the flight
        - destination: destination city of the flight
        - departure_time: departure time of the flight
        - arrival_time: arrival time of the flight
        - capacity: total capacity of the flight
        - reservable_capacity: remaining capacity available for reservations
2. **User Document**
    - **Key**: `user_id`
    - **Structure**:
        - user_id: unique identifier for the user
        - username: username of the user
        - password: hashed password of the user
        - balance: account balance of the user
        - reservations: array of nested objects, each containing:
            - reservation_id: unique identifier for the reservation
            - is_paid: boolean indicating if the reservation is paid
            - reservation_date: date of the reservation
            - flights: array of nested objects, each containing:
                - flight_id: unique identifier for the flight
                - origin: origin city of the flight
                - destination: destination city of the flight
                - departure_time: departure time of the flight
                - arrival_time: arrival time of the flight
3. **Itinerary Document**

    - **Key**: `itinerary_id`
    - **Structure**:
        - itinerary_id: unique identifier for the itinerary
        - origin: origin city of the itinerary
        - destination: destination city of the itinerary
        - is_direct: boolean indicating if the itinerary is direct
        - flights: array of nested objects, each containing:
            - flight_id: unique identifier for the flight
            - departure_time: departure time of the flight
            - arrival_time: arrival time of the flight

Implementation Notes

- **Reservable Capacity Management**: When a reservation is made, the corresponding flight documents are updated to decrement the `reservable_capacity`.
- **User Reservations**: User documents embed reservations, which include detailed flight information to minimize the need for additional queries.
- **Indexing**: Utilize indexes on common query fields such as `origin`, `destination`, and `username` to optimize search and access patterns.

This document-oriented approach leverages the flexibility of document stores to manage hierarchical data structures and efficiently handle the diverse query patterns required by Flightapp.

1. There exists a flight 1234 from Seattle, WA to Los Angeles, CA with exactly 5 unreserved seats
2. Hannah searches for a Seattle->LA itinerary and sees flight 1234
3. She attempts to reserve a seat on flight 1234, but accidentally clicks the button twice

What is your document store's final state?

Choice 1 of 4: the user has two reservations, and the flight's unreserved capacity is consistent with this

Choice 2 of 4: the user has one reservation, and the flight's unreserved capacity is consistent with this

Choice 3 of 4: the user has zero reservations, and the flight's unreserved capacity is consistent with this

Choice 4 of 4: the flight's unreserved capacity is inconsistent with the user's reservations