Homework Question 5

Graph Database Design for Flightapp

Rowkeys, Column Families, and Columns

In a graph database like Neo4j, data is represented as nodes (entities) and edges (relationships) connecting these nodes. Each node and edge has properties (columns), and they form the backbone of the schema. Here, we describe the key elements in the schema for Flightapp:

Node Types and Properties

1. **User**
    - **Properties (Columns)**:
        - user_id: Unique identifier for the user
        - username: Username of the user
        - password: Hashed password
        - balance: Account balance
2. **Flight**
    - **Properties (Columns)**:
        - flight_id: Unique identifier for the flight
        - carrier_id: Identifier for the carrier
        - origin: Origin city
        - destination: Destination city
        - departure_time: Departure time
        - arrival_time: Arrival time
        - capacity: Total capacity
        - reservable_capacity: Remaining reservable capacity
3. **Carrier**
    - **Properties (Columns)**:
        - carrier_id: Unique identifier for the carrier
        - carrier_name: Name of the carrier
4. **Itinerary**
    - **Properties (Columns)**:
        - itinerary_id: Unique identifier for the itinerary
        - origin: Origin city
        - destination: Destination city
        - is_direct: Boolean indicating if the itinerary is direct
5. **Reservation**
    - **Properties (Columns)**:

- reservation_id: Unique identifier for the reservation
- is_paid: Boolean indicating if the reservation is paid
- reservation_date: Date of the reservation

Edge Types

1. **User RESERVES Reservation**
2. **Reservation CONTAINS Flight**
3. **Itinerary INCLUDES Flight**
4. **Flight OPERATED_BY Carrier**

Commands and Use of Schema

1. **Create User**
   - **Command**: Create a `User` node
   - **Properties Used**: user_id, username, password, balance
   - **Example**: Create a new node with user_id, username, password, and balance.
2. **Search Itineraries**
   - **Command**: Query for `Itinerary` nodes based on origin, destination, and is_direct
   - **Properties Used**: origin, destination, is_direct
   - **Example**: Find all itineraries matching the search criteria.
3. **Reserve Flights**
   - **Command**: Create a `Reservation` node, link it to `User` and `Flight` nodes
   - **Properties Used**: reservation_id, is_paid, reservation_date
   - **Example**: Create a reservation node, link it to the user node, and update the flight node's reservable_capacity.
4. **Pay for Reservation**
   - **Command**: Update `Reservation` node to mark it as paid, decrement user balance
   - **Properties Used**: reservation_id, is_paid, user balance
   - **Example**: Mark the reservation as paid and update the user's balance.
5. **List Reservations**
   - **Command**: Query for `Reservation` nodes linked to a `User`
   - **Properties Used**: user_id
   - **Example**: Retrieve all reservations associated with a user.

Timestamps

- Timestamps are not explicitly mentioned in the provided schema. If needed, they can be added as properties to track creation and modification times for nodes and edges.

This schema leverages the strengths of a graph database to manage interconnected data efficiently, supporting complex queries and relationships inherent in Flightapp's operations.