COMSW4111 Project1 Part3

Jinyang Cai(JC5513) Jiapeng Xu(JX2427)
UNI used on the database server: JC5513

1 ER Diagram

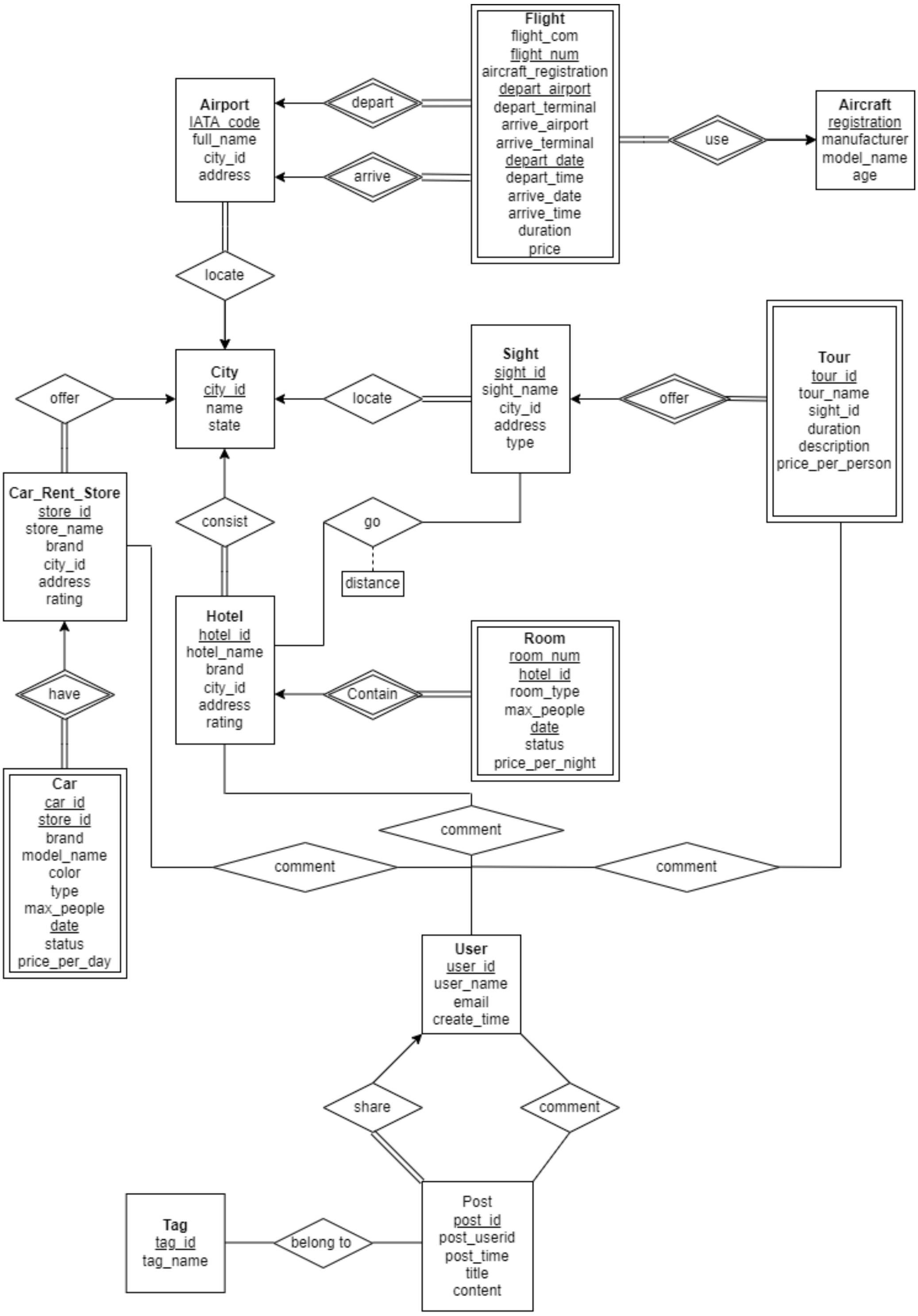

Figure 1: ER Diagram

2 SQL Schema

```
CREATE TABLE Aircraft (
  registration text UNIQUE NOT NULL,
  manufacturer text,
  model_name text,
  age int,
  PRIMARY KEY (registration)
);

CREATE TABLE City (
  city_id SERIAL PRIMARY KEY,
  name text,
  state text
);

CREATE TABLE Airport (
  IATA_code text PRIMARY KEY,
  full_name text,
  city_id int,
  address text,
  FOREIGN KEY (city_id)
  REFERENCES City (city_id)
);

CREATE TABLE Flight (
  flight_com text,
  flight_num text,
  aircraft_registration text,
  depart_airport text,
  depart_terminal text,
  arrive_airport text,
  arrive_terminal text,
  depart_date date,
  depart_time time,
  arrive_date date,
  arrive_time time,
  duration text,
  price float,
  PRIMARY KEY (flight_num,
               depart_airport,
               depart_date),
  FOREIGN KEY (aircraft_registration)
  REFERENCES Aircrafts (registration)
  ON DELETE CASCADE,
  FOREIGN KEY (depart_airport)
  REFERENCES Airport (IATA_code)
  ON DELETE CASCADE,
  FOREIGN KEY (arrive_airport)
  REFERENCES Airport (IATA_code)
  ON DELETE CASCADE
);

CREATE TABLE Car_Rent_Store (
  store_id SERIAL PRIMARY KEY,
  store_name text,
  brand text,
  city_id int REFERENCES City (city_id),
  address text
  rating float
);
```

```
CREATE TABLE Car (
  car_id int,
  store_id int,
  brand text,
  model_name text,
  color text,
  type text,
  max_people int,
  date date,
  status text,
  price_per_day float,
  CHECK (status IN (’Available’, ’Not Available’)),
  PRIMARY KEY (car_id, store_id, date),
  FOREIGN KEY (store_id)
  REFERENCES Car_Rent_Store (store_id)
  ON DELETE CASCADE
);

CREATE TABLE Hotel (
  hotel_id serial PRIMARY KEY,
  hotel_name text,
  brand text,
  city_id int,
  address text,
  rating float,
  FOREIGN KEY (city_id) REFERENCES City (city_id)
);

CREATE TABLE Room (
  room_num int,
  hotel_id int,
  room_type text,
  max_people int,
  date date,
  status text,
  price_per_night float,
  CHECK (status IN (’Avaiable’, ’Not Avaiable’)),
  PRIMARY KEY (room_num, hotel_id, date),
  FOREIGN KEY (hotel_id) REFERENCES Hotel (hotel_id)
  ON DELETE CASCADE
);

CREATE TABLE Sight (
  sight_id SERIAL PRIMARY KEY,
  sight_name text,
  city_id int,
  address text,
  type text,
  FOREIGN KEY (city_id) REFERENCES City (city_id)
);

CREATE TABLE HS_distance (
  hotel_id int,
  sight_id int,
  distance float,
  FOREIGN KEY (hotel_id) REFERENCES Hotel (hotel_id),
  FOREIGN KEY (sight_id) REFERENCES Sight (sight_id),
  PRIMARY KEY (hotel_id, sight_id)
);
```

```
CREATE TABLE User_info (
  user_id SERIAL PRIMARY KEY,
  user_name text UNIQUE,
  email text UNIQUE,
  create_time timestamp
);

CREATE TABLE User_comment_CarRentStore (
  comment_id SERIAL PRIMARY KEY,
  comment_userid int
  REFERENCES User_info (user_id),
  store_id int
  REFERENCES Car_Rent_Store (store_id),
  comment_time timestamp,
  content text,
  rec_or_not text,
  CHECK(rec_or_not IN (’Positive’, ’Negative’))
);

CREATE TABLE Tour (
  tour_id SERIAL PRIMARY KEY,
  tour_name text,
  sight_id int REFERENCES Sight (sight_id),
  duration text,
  description text,
  price_per_person float
);

CREATE TABLE User_comment_Tour (
  comment_id SERIAL PRIMARY KEY,
  comment_userid int
  REFERENCES User_info (user_id),
  tour_id int REFERENCES Tour (tour_id),
  comment_time timestamp,
  content text,
  rec_or_not text,
  CHECK(rec_or_not in (’Positive’, ’Negative’))
);
```

```
CREATE TABLE User_comment_Hotel (
  comment_id SERIAL PRIMARY KEY,
  comment_userid int REFERENCES User_info (user_id),
  hotel_id int REFERENCES Hotel (hotel_id),
  comment_time timestamp,
  content text,
  rec_or_not text,
  CHECK (rec_or_not in (’Positive’, ’Negative’))
);

CREATE TABLE Post (
  post_id SERIAL PRIMARY KEY,
  post_userid int REFERENCES User_info (user_id),
  post_time timestamp,
  title text,
  content text
);

CREATE TABLE User_comment_Post (
  comment_id SERIAL PRIMARY KEY,
  comment_userid int REFERENCES User_info (user_id),
  post_id int REFERENCES Post (post_id),
  comment_time timestamp,
  content text
);

CREATE TABLE Tag (
  tag_id SERIAL PRIMARY KEY,
  tag_name text
);

CREATE TABLE Post_Tag (
  post_id int REFERENCES Post (post_id),
  tag_id int REFERENCES Tag (tag_id),
  PRIMARY KEY (post_id, tag_id)
)
```

3 Interesting Queries

1. Description:
   Mark decides to rent a car in Miami. He would like to make choice not only based on the price of the car, but also based on customers’ former experience in that store. He is trying to select the car-rent store with the most number of positive recommendation.

   ```
   SELECT b.*, a.num_pos FROM
   (SELECT store_id, count(comment_id) as num_pos FROM User_comment_CarRentStore
   WHERE store_id IN
   (SELECT store_id FROM Car_Rent_Store WHERE
   city_id IN
   (SELECT city_id FROM City where name=’Miami’))
   AND rec_or_not = ’Positive’
   GROUP BY store_id) a
   JOIN
   (SELECT crs.store_id, crs.store_name, crs.address FROM City c
   JOIN Car_Rent_Store crs
   ON c.city_id = crs.city_id
   WHERE c.name=’Miami’) b
   ON a.store_id = b.store_id
   ORDER BY num_pos DESC
   ```

2. Description:
   Lily plans to visit San Francisco during the spring break. She doesn't know much about there. She tries to find local tours in San Francisco.

```
SELECT * FROM Tour
WHERE sight_id IN (
  SELECT sight_id FROM Sight s
  JOIN City c ON s.city_id = c.city_id
  WHERE c.name = 'San Francisco'
)
```

3. Description:
   Jack wants to go to Miami this spring break. However, he knows nothing about what things to do, and which places to stay. Therefore, he is going to search for the posts and comments that contain the tag 'Miami' and was posted later than Jan 1, 2022, so that he can make a plan for traveling to Miami.

```
SELECT P.title, P.content, C.content AS comment
FROM Post_Tag AS PT, Post AS P, Tag AS T, User_comment_Post AS C
WHERE PT.post_id = P.post_id
 AND PT.tag_id = T.tag_id
 AND P.post_id = C.post_id
 AND T.tag_name = 'Miami'
 AND P.post_time > '2022-01-01 00:00:00';
```

4 Description of Extensions

The first part of our application mainly provides users with information about hotels, flights, and car-rent stores. Users are able to make their choice by searching objective information from our application. In the second part, firstly we add an entity 'Tour', which provides convenient tours for those users who are willing to join a local tour to explore sights. What's more, we decide to add a community to our application: users can share or comment on what they have already experienced. Such a community will also help users make their decision and bring them a happy spring break.
At first, we create a User_info table that contains information about users. Users are able to create their accounts on our website and post or comment on some information. Secondly, we allow users to comment on what they have already experienced. They can leave their comments on hotels, tours, or car-rent stores that they experienced and tell other users whether or not they would recommend the hotel, tour, or car-rent store. This will help potential customers know if they are going to have a good experience. Other users' recommendations help them make decisions. For this, we create three tables: User_comment_CarRentStore, User_comment_Tour and User_comment_Hotel. These tables contain users' account information, the time of the comment, the hotel, tour, or car-rent store they comment on, the contents of the comment, and whether the user will recommend it or not. In our design, if some users delete their accounts, their comments will still be available.
Besides, we create a Post entity: users are able to ask questions related to their trip or share what they met during their last trip, which is really useful for others' trip plans. Users can also comment on others' posts or answer the questions. We create a User_comment_Post table which contains the user's account, comment time, the post commented, and the content. What's more, we also try to group different posts into different tags, which help users search for the post they wanted. The Tag table contains each tag's name. Same as above, if the user deletes account, posts, and comments will also be available.