Assumptions

Relationship between user and user

Based on reality and how people use social media, we assume that a person can follow multiple people, and each person can be followed by multiple people. This creates a many to many relationship.

Relationship between city and vacation spots

A vacation spot has one CityID because unlike a city, which represents a broad geographic area, a vacation spot could be a specific attraction, hotel, landmark, or event venue. Therefore, to allow for more specificity in searches we made it an entity. There are many vacation spots within one city, creating a many to one relationship.

Relationship between user and vacation spots

A vacation spot can be favorited by multiple users. Many users will be visiting the same vacation spot, and more popular spots will have more reviews. This is a many to many relationship since a vacation spot can have multiple users favorite it and a user can favorite multiple vacation spots.

Relationship between user and review

A vacation spot can have multiple reviews. Again, lots of users will be reviewing the same location. The vacation spot to review is a many to many relationship in which each vacation spot can have multiple reviews, and each review can have multiple vacation spots. We are assuming users can create multiple reviews and a review can only be created by one user. This user to review relationships is then one to many.

Relationship between reviews and images

We assume that each image can only correspond to one review because they are only present in that review. A review has a one-to-many relationship in that one review could contain multiple images, but each image can only be mapped to one review.

Relationship between vacation spot and reviews

We are assuming that people who review can visit multiple vacation spots in the same review. This creates a many-to-one relationship between vacation spots and reviews

Users

Users is an entity because it is a distinct object that has its own properties such as a username and password. There are complex relations between users and many other entities that cannot be modelled if user was an attribute.

Reviews

Reviews is an entity because it is a distinct object that has its own properties. It also is related to another entity, and therefore cannot be an attribute

Vacation Spots

A vacation spot represents a specific attraction, hotel, landmark, or event venue. It provides deeper information than a city and has its own attributes. Therefore, it cannot be modeled as an attribute.

City

A city has distinct attributes that describe it such as its population and country. It does not rely on anything else to give it value, meaning it cannot be an attribute

Images

Images have complex relationships with individual reviews that cannot be modeled if it were an attribute. Furthermore, multiple images can be related to a review ensuring that it cannot be an attribute.

Normalization

Our schema adheres to the standards for BCNF normalization: the left side of every non-trivial functional dependency (FD) in a table is a superkey.

User Accounts

- username → password, profilePictureURL, profileDescription, gender, country, age

Reviews

- reviewId → username, reviewText

Images

- imageURL → reviewId

Vacation Spots

- vacationSpotName → cityId

City

- cityId → cityName, longitude, latitude, population, language, country, province, avgTmp, avgMealPrice, avgTicketPrice
- cityName → cityId, longitude, latitude, population, language, country, province, avgTmp, avgMealPrice, avgTicketPrice

Follows

- There are no non-trivial functional dependencies

Favourite_Sports

- There are no non-trivial functional dependencies

Vacation_Spot_Reviews

- There are no non-trivial functional dependencies

Relational Schema

CITY(
  cityId: INT [PK],
  cityName: VARCHAR(50),
  longitude: DECIMAL,
  latitude: DECIMAL,
  population: INT,
  language: VARCHAR(50),
  country: VARCHAR(50),
  province: VARCHAR(50),
  avgTmp: DECIMAL,
  avgMealPrice: DECIMAL,
  avgTicketPrice: DECIMAL
)

VACATION_SPOT(
  vacationSpotName: VARCHAR(50) [PK],
  city: [FK to CITY.cityId]
)

USER_ACCOUNTS(
  username: VARCHAR(50) [PK],
  password: VARCHAR(100),
  profilePictureURL: VARCHAR(255),
  profileDescription: VARCHAR(255),
  gender: VARCHAR(10),
  country: VARCHAR(50),
  age: INT
)

REVIEWS(
  reviewId: INT [PK],
  username: INT [FK to USER_ACCOUNTS.username],
  reviewText: VARCHAR(2000)
)

IMAGES(
  imageURL: VARCHAR(255) [PK],
  reviewId: INT [FK to REVIEWS.reviewId]
)

Many-to-many self-relationship (user follows user)
FOLLOWS(
  followerId: INT [FK to USER_ACCOUNTS.username],
  followedId: INT [FK to USER_ACCOUNTS.username]
)

Many-to-many relationship (user favorites a vacation spot)
FAVORITE_SPOTS(
  username: VARCHAR(50) [FK to USER_ACCOUNTS.username],
  vacationSpotId: INT [FK to VACATION_SPOT.vacationSpotId]
)

Many-to-many relationship (reviews and vacation spots)
VACATION_SPOT_REVIEWS (
  reviewId: INT [FK to REVIEWS.reviewId]
  vacationSpotId: INT [FK to VACATION_SPOT.vacationSpotId],
)