Online Restaurant Review Platform - Yeahp

COMS W 4111 Project 1 Part 1 - Jace Yang (uni: jy3174) & Binghong Yu (uni: by2325)

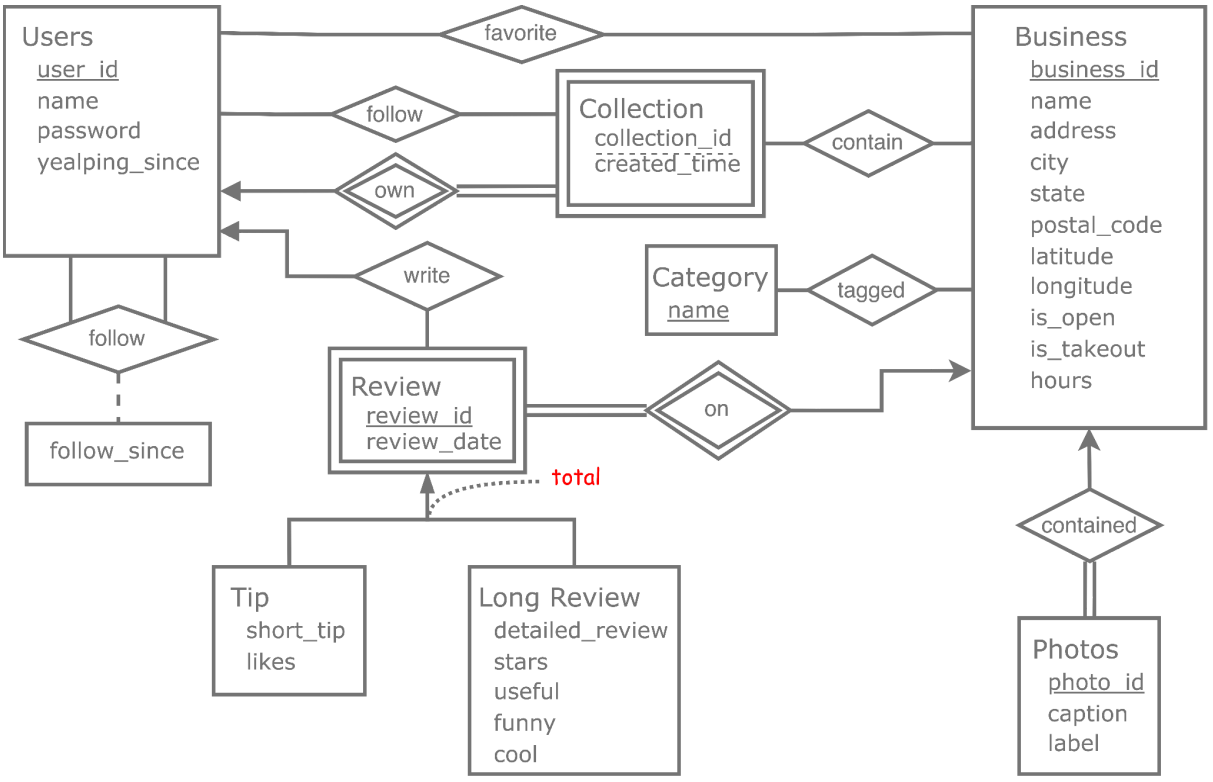
Description

Yeahp is our upcoming online restaurant review platform that functions similarly to Yelp in that users search for eateries. To populate our database, we will use the official Yelp Dataset. **The interesting part**: since our dataset is real data, while browsing on our website, users can trust the reviews and ratings of those small businesses and local restaurants. **The challenging part** is the implementation of interactions between users such as following each other and adding restaurants into favorites, both in front-end and back-end. **Some examples of entities and their relationships and restraints**: (1) Users(attribute: name, id, etc.) could post either Long Reviews(detailed_review, stars, etc.) or Tip(short_tip, likes) on Business(name, address, opening hour, etc.). (2) Reviews are owned by the Businesses, thus will be removed once Businesses deleted their accounts (but not once Users deleted accounts). (3) Users could submit several Reviews, but each Review has only one author (User). (4) Users could follow multiple Collections. Also, Users could own multiple Collections. However, if a user deleted his/her account, the Collection owned by that User would also be removed. **Interaction details between entities include**: (1) users could create an account, log in, and follow other users. (2) Users might find restaurants by entering the city or zip code first to filter down the location. (3) Users could then search for restaurants by star rating, opening time, and distance. In addition, users could also use the name of a restaurant to find it. (4) After locating a certain restaurant, users can leave detailed evaluations and tips, and read reviews about the restaurant. Users in review modules could also follow other users. (5) Users could save the restaurant to their collections or directly favorite that restaurant. In addition, users could also follow other's collections.

Contingency Plan

Disable several complex functions such as follow, tip, collections, and favorite.

E/R diagram for the application



SQL schema

```
!pip3 install sqlalchemy # ORM for databases
!pip3 install ipython-sql # SQL magic function
```

In [1]:

```
%load_ext sql
```

In [2]:

```
%sql postgresql://jy3174:JaceYJH@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/proj1part
2
```

```
/usr/local/lib/python3.7/dist-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2
wheel package will be renamed from release 2.8; in order to keep installing from binary pl
ease use "pip install psycopg2-binary" instead. For details see: <http://initd.org/psycopg
/docs/install.html#binary-install-from-pypi>.
  """)
```

Out[2]:

```
'Connected: jy3174@proj1part2'
```

Translating Entities

In [3]:

```
%%sql
DROP TABLE IF EXISTS Users, User_follow_user, Collection_of_User, Review_of_Business, Cate
gory, Business CASCADE;
DROP TYPE IF EXISTS Opening_hours CASCADE;

CREATE TABLE Users(
    user_id int PRIMARY KEY,
    name text,
    password int,
    yealping_since date
);

CREATE TABlE Collection_of_User(
    collection_id int,
    created_time date,
    user_id int,
    PRIMARY KEY(user_id, collection_id),
    FOREIGN KEY(user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

CREATE TYPE Opening_hours AS (
    Monday text,
    Tuesday text,
    Wednesday text,
    Thursday text,
    Friday text,
    Saturday text,
    Sunday text
);

CREATE TABlE Business(
    business_id int PRIMARY KEY,
    name text,
    address text,
    city text,
    state text,
```

```
    postal_code text,
    latitude numeric(4),
    longitude numeric(4),
    is_open boolean,
    is_takeout boolean,
    hours Opening_hours
);


CREATE TABlE Review_of_Business(
    review_id int PRIMARY KEY,
    review_date date,
    business_id int NOT NULL,
    -- Attributes of Tip
    short_tip text,
    likes int,
    -- Attributes of Long Review
    detailed_review text,
    stars int,
    useful int,
    funny int,
    cool int,

    CHECK (stars >= 0 AND stars <= 5),
    CHECK (
           ((short_tip IS NULL AND likes IS NULL)
            OR
            (detailed_review IS NULL AND stars IS NULL AND useful IS NULL AND funny IS NU
LL AND cool IS NULL))
           AND
           ((short_tip IS NOT NULL)
            OR
            (detailed_review IS NOT NULL))
         ),
    CHECK (length(detailed_review) >= 100 OR detailed_review is NULL),

    FOREIGN KEY(business_id) REFERENCES Business(business_id) ON DELETE CASCADE
);

CREATE TABlE Category(
    name varchar(255) PRIMARY KEY
);
```

```
 * postgresql://jy3174:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/proj1part2
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[3]:

```
[]
```

Translating Relations

In [4]:

```
%%sql
DROP TABLE IF EXISTS Users_favorite_Business, Users_follow_Collection, Collection_contain_
Business,
                     Users_write_Review, Collection_contain_Business, Photo_contained_Bus
iness,
                     Users_follow_Users, business_tagged_category CASCADE;

CREATE TABLE Users_favorite_Business(
  user_id int REFERENCES Users(user_id),
  business_id int REFERENCES Business(business_id),
```

```
  PRIMARY KEY(user_id, business_id)
);

CREATE TABLE Users_follow_Collection(
  fan_user_id int REFERENCES Users(user_id),
  followee_user_id int,
  collection_id int,
  PRIMARY KEY(fan_user_id, followee_user_id, collection_id),
  FOREIGN KEY(followee_user_id, collection_id) REFERENCES Collection_of_User(user_id, coll
ection_id)
);

CREATE TABLE Collection_contain_Business(
  collection_owner_id int,
  collection_id int,
  business_id int REFERENCES Business(business_id),
  PRIMARY KEY(collection_owner_id, collection_id, business_id),
  FOREIGN KEY(collection_owner_id, collection_id) REFERENCES Collection_of_User(user_id, c
ollection_id)
);


CREATE TABLE Users_write_Review(
  user_id int NOT NULL REFERENCES Users(user_id),
  review_id int REFERENCES Review_of_Business(review_id),
  PRIMARY KEY(review_id)
);

CREATE TABLE Business_tagged_Category(
  business_id int REFERENCES Business,
  name text REFERENCES Category,
  PRIMARY KEY(business_id, name)
);

CREATE TABLE Photo_contained_Business(
  photo_id int PRIMARY KEY,
  business_id int NOT NULL,
  caption text,
  label text,
  FOREIGN KEY(business_id) REFERENCES Business
    ON DELETE CASCADE
);

CREATE TABLE Users_follow_Users(
    follwee_user_id int REFERENCES Users(user_id),
    fan_user_id int REFERENCES Users(user_id),
    follow_since date,
    PRIMARY KEY (follwee_user_id, fan_user_id)
);
```

```
 * postgresql://jy3174:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/proj1part2
Done.
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[4]:

```
[]
```

Testing case

In [5]:

```
%%sql

-- Check Review_of_Business (the most complicated one) is properly functioning.
```

```
DELETE FROM Business where business_id = 2 OR business_id = 3;
DELETE FROM Review_of_Business where review_id = 1 OR review_id = 2;

INSERT INTO Business(business_id, name) VALUES (2, 'Mc'), (3, 'KFC');
INSERT INTO Review_of_Business(review_id, business_id, short_tip, likes) VALUES (1, 2, 'I
LIKE IT', 5);
INSERT INTO Review_of_Business(review_id, business_id, detailed_review, useful, funny) VAL
UES (2, 3, 'Great place to hang out after work: the prices are decent, and the ambience is
fun. It''s a bit loud, but very lively. The staff is friendly, and the food is good. They
have a good selection of drinks.', 5, 0);
SELECT * FROM Review_of_Business
```

```
 * postgresql://jy3174:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/proj1part2
0 rows affected.
0 rows affected.
2 rows affected.
1 rows affected.
1 rows affected.
2 rows affected.
```

Out[5]:

review_id	review_date	business_id	short_tip	likes	detailed_review	stars	useful	funny	cool
1	None	2	I LIKE IT	5	None	None	None	None	None
2	None	3	None	None	Great place to hang out after work: the prices are decent, and the ambience is fun. It's a bit loud, but very lively. The staff is friendly, and the food is good. They have a good selection of drinks.	None	5	0	None