CS4111 Project 2
Xinyue Wu, Xilin Wang
uni: xw2766, xw2767

PostgreSQL account name: xw2767
Password: 7210

Explanation of schema modification

For Project 2, we expanded our database by adding a composite type *event_type* and created a typed table *events* of *event_type* which contains a TEXT attribute called *detail*. We also added a trigger *TotalPartUR* that enforces the total participation constraint on User_Reviews in relationship u_writes_for when there's new insertion to User_Reviews.

For the composite type event_type and typed table events, we added them to our schema so that our project can now model promotional or special events co-hosted by restaurants. These events happen in the real world very often so this expansion of our schema enables our database to better model interactions of restaurants in the real world thus it fits perfectly within the overall project. The event_type composite type consists of three attributes: the event id (eid) which is modeled as an integer, the event name which is a string of variable length and the event detail which is of TEXT type. The typed table *events* is created from event_type with eid as the primary key and name, detail is constrained to be not null. Also, to model the relationship that multiple restaurants can co-host an event, we added the hosts table which contains the restaurant id (rid) and the event id (eid) with rid, eid as foreign keys pointing to table *Restaurants* and *events* respectively and the primary key is set to be (rid,eid) so that it keeps record of (restaurant,event) pairs. These additional types and tables also enable users to run more interesting queries on our database such as finding restaurants that's offering discounts for a promotional event, which we'll cover below in the queries section.

For the *TotalPartUR* trigger, we added it to enforce the total participation constraint on User_Reviews in relationship u_writes_for when there's new insertion to User_reviews. This total participation constraint was in the original E/R diagram from Project 1 part 1 but wasn't implemented in the schema because we haven't learned the usage of trigger at that time. It checks if every newly inserted user review in the User_Reviews table has a (user,restaurant) pair associated with it in u_writes_for. If the review has a (user,restaurant) pair associated with it in u_writes_for, the review is inserted into the User_Reviews table. If not, the inserted review is deleted from the database. We enforce this participation constraint because we don't want to store reviews that's not written for any restaurant nor written by any user in our database. Such reviews can potentially occupy lots of storage since they can contain long text but they don't point to any restaurant or user, which makes them unmeaningful for our application.

An event example:

Suppose user 'Xilin Wang' (me) with uid 'xw2767' is reviewing the restaurant 'Kissaki Sushi' with rid 'ChIJj35dz8xZwokR8d8a6Fnx3vQ' and would like to insert a review into User_Reviews that has urid 'xw2767-3'. Then the trigger is executed after the insertion and checks if the urid 'xw2767-3' is in any tuples in the u_writes_for table.

If the user correctly inserted the review, which is to start a transaction and inside the transaction block, insert the review into User_Reviews and then insert a (uid, urid, rid) tuple into u_writes_for to specify which user wrote this review and which restaurant is this review for and then end the transaction. Then the trigger doesn't modify the database and the review should be successfully inserted.

If the user inserted the review into User_Reviews but forgot to insert the (uid, urid, rid) tuple into u_writes_for, the trigger will delete the review tuple with the new urid from User_Reviews so that the unmeaningful review is not inserted.

SQL code that corresponds to the above example event:

Correct insertion for which the trigger is executed but doesn't modify the database so the review is successfully inserted:

```
BEGIN TRANSACTION;
INSERT INTO User_Reviews
VALUES ('xw2767-3', 5, 'Great sushi spot near Columbia University. The wait staff is friendly and helpful. They are a bit overly techno- you have to scan a QR code to order.... Then the food is awesome.');
INSERT INTO u_writes_for
VALUES ('xw2767', 'xw2767-3', 'ChIJj35dz8xZwokR8d8a6Fnx3vQ', '2020-12-02 18:59:19') ;
END;
```

Incorrect insertion where only the review is inserted but didn't specify which (user,restaurant) pair the review is associated with. Here trigger is executed and the newly inserted review tuple is deleted from User_Reviews table:

```
INSERT INTO User_Reviews
VALUES ('xw2767-3', 5, 'Great sushi spot near Columbia University. The wait staff is friendly and helpful. They are a bit overly techno- you have to scan a QR code to order.... Then the food is awesome.');
```

Queries

1. Find names and average user ratings of restaurants that offer vegetarian meals and participate in either the 'Christmas feast' or 'New Year New Beginning' event.

```
SELECT R.name, AVG(UR.rating)
FROM Restaurant R, events E, hosts H, u_writes_for UW, User_Reviews UR
WHERE
R.rid in (SELECT S.rid
FROM satisfies S
WHERE S.name =  'Vegetarian')
AND R.rid = H.rid
AND R.rid = UW.rid
AND UW.urid = UR.urid
AND E.eid = H.eid
AND (E.name = 'Christmas Feast' OR E.name = 'New Year New Beginning')
GROUP BY R.name;
```

2. Find the names, categories and locations of restaurants that are both offering discounts/coupons for some special events and have dine-in options.

```
SELECT R.name, R.category, L.number, L.street
FROM Restaurant R, Is_at_Locations L, events E, hosts H
WHERE E.eid in (
SELECT E2.eid
FROM events E2
WHERE to_tsvector(detail) @@ to_tsquery('discount | coupon'))
AND R.rid in (
SELECT O.rid
FROM Offers O
WHERE O.type LIKE '%dine-in%')
AND R.rid = H.rid
AND E.eid = H.eid
AND R.rid = L.rid;
```