CptS -451 Introduction to Database Systems
Spring 2019
Everett

Project Milestone-2

Due Date: Thursday March 21, 11:59pm

Summary:

In this milestone you will:

- ✓ design the database schema for your application and provide the ER diagram for your database design,
- ✓ translate your entity relationship model into relations and produce DDL SQL statements for creating the corresponding tables in a relational DBMS,
- ✓ populate your database with the Yelp data and get to practice generating INSERT statements and running those to insert data into your DB,
- ✓ write triggers to enforce additional constraints,
- ✓ start developing your application GUI . In Milestone3 you will develop the full final application with all required features.

Milestone Description:

You need to complete the following in milestone-2:

1) Revise the database schema that you created in milestone-1. Make sure that your database schema is complete (i.e., stores all data necessary for the application) and appropriate (i.e., all queries/data retrievals on/from the database can be run efficiently and effectively).

   Additional notes about the database schema:

   - ✓ Your business table should include the following attributes (in addition to the business attributes provided in the business JSON objects):
     - "numCheckins" which stores the number of total check-ins for each business.
     - "reviewrating" which stores the avg star rating for each business.

2) (4%) Translate your revised ER model into relations and produce DDL SQL (CREATE TABLE) statements for creating the corresponding tables in a relational DBMS. Note the constraints, including primary key constraints, foreign key constraints, not NULL constraints, etc. needed for the relational schema to capture and enforce the semantics of your ER design. Write your CREATE TABLE statements to a file named "<*your-team-name*>_RELATIONS_v2.sql".

3) (46%) Populate your database with the Yelp data.

   a. Generate INSERT statements for your tables and run those to insert data onto your DB. You will use your JSON parsing code from Milestone-1 and use the data you extracted from JSON objects to generate the INSERT statements for your tables.

      Note: In your business table, initialize "*numCheckins*" attribute to `0`, and "*reviewrating*" attribute to `0.0` for all businesses (i.e., assign "*numCheckins*" to 0 in the INSERT statements). In task-5, you will write an UPDATE statement where you calculate the sum of all check-ins and avg star rating for each business and update the value of "*numCheckins*" and "*reviewrating*" attributes in the business table.

b. You may populate your DB with data in 2 different ways:

i. You may embed your INSERT statement inside your JSON parsing code and execute them one by one as you generate them. (Sample Python code for connecting to PostgreSQL database and executing SQL statements will be available on Blackboard).

ii. Alternatively, you may write the INSERT statements to a SQL script file and then run this (large) script file. (You will find some information about how to generate and run SQL scripts in Appendix-A of this document).

Please note that due to foreign key constraints, the order of tables you insert data to matters. The INSERTs to referenced tables should be run before INSERTs to referencing tables.

Please do not create any INDEXES for your tables until you insert all the data. Indexes can slow down the data insertion.

4) After you insert all your data, retrieve the number of tuples in each of your tables and copy/paste that information into a text file. (Simply run the query "`SELECT COUNT(*) FROM tablename`" for each table in your database.) Copy the results to *"<your-team-name>*_TableSizes.txt file.

5) (6%) Calculate and update the "*numCheckins*", "*reviewcount*", and "*reviewrating*" attributes for each business.

a. "*numCheckins*" value for a business should be updated to the sum of all check-in counts for that business. Similarly, "*reviewcount*" should be updated to the number of reviews provided for that business (Note that you will overwrite the values extracted from the JSON data). "*reviewrating*" is the average of the review star ratings provided for each business. You should query the review table to calculate the number of reviews and avg review rating for each business. Similarly, you should query the check-in table to calculate the total number of check-ins. In grading, points will be deducted if you don't update these values.

b. Write your UPDATE statements to a file named "*"<your-team-name>*_UPDATE.sql".

(**Note**: Running the update statement for the *reviewcount* and *reviewrating* may take a long time. To speed up the process, you may calculate and store the calculated *reviewcount*, *reviewrating*, and *numCheckins* into a temporary table and then update the business table using the values from this temporary table.)

6) (18%) Create triggers to enforce the following constraints in your database:

a. Whenever a new review is provided for a business, the "*reviewcount*" and "*reviewrating*" values for that business should be automatically updated. (Note that the updated "*reviewrating*" can be obtained by summing all the review star ratings for the business (including the rating for the new review) and dividing the sum by the number of reviews)

b. Similarly, when a customer checks-in a business, the "*numCheckins*" value for that business should be automatically updated.

Test your triggers with INSERT/UPDATE statements, i.e.,

- Add a review for a business (insert to review table) and make sure that the number of reviews in the business table is updated.
- Check-in to a business make sure that the checkin table is updated (i.e., the checkin count for the corresponding time is incremented by 1). In addition, make sure that *numCheckins* attribute in the business table is also updated.

Write your TRIGGER statements and **test statements** (INSERT, UPDATE statements) to a file named "*"<your-team-name>*_TRIGGER.sql".

7) (26%) Start implementing your user interface. You should complete the following features in milestone2.
    a. Retrieve all distinct states that appear in the Yelp business data and list them (e.g. in a combo-box).
    b. When user selects a state, retrieve and display the cities that appear in the Yelp business data in the selected state (e.g. in a list a listbox.)
    c. When a city is selected, retrieve the zipcodes that appear in the Yelp business data in the selected city (e.g. in a list a listbox.)
    d. When a zipcode is selected, retrieve all the business categories for the businesses that appear in that zipcode. (e.g. in a list a listbox.)
    e. When the user searches for businesses, all the businesses in the selected zipcode will be displayed (e.g. in a datagrid).
    f. If the user selects any categories, the search results will be filtered based on the selected business categories. The search results should include the businesses that belong to all selected categories.
    g. User selects a business and displays the reviews of a selected business.
    h. User adds a new review for a selected business (i.e., enters a review text and chooses a star rating value for the selected business; the review text and star rating should be inserted to the review table and a unique *review_id* should be generated for the new review. The trigger you implemented in task 6(a) should update the "*reviewcount*" and "*reviewrating*" values for that business automatically)

(Note: The mentioned interface components are only suggested ways to display the data. As long as it is functional and easy to use, any design is acceptable.)

Milestone-2 Deliverables - Checklist:

1. The revised E-R diagram for your database design. To create your ER diagram, you can use an ER modeling tool that you get from the web (ERDraw), or your favorite drawing tool (e.g., Visio, Word, PowerPoint). **Should be submitted in .pdf format.** Name this file *"<your-team-name>*_ER.*pdf*"
2. SQL script file containing all CREATE TABLE statements. Name this file *"<your-team-name>*_RELATIONS_v2.sql"
3. Text file containing all table sizes. Name this file *"<your-team-name>*_ TableSizes.txt"
4. SQL script file containing all UPDATE TABLE statements. Name this file *"<your-team-name>*_UPDATE.sql"
5. SQL script file containing all TRIGGER statements and the test statements. Name this file *"<your-team-name>*_TRIGGER.sql"

Create a zip archive *"<your-team-name>_milestone2.zip*" that includes all the 5 items above. Upload your milestone-2 submission on Blackboard until the deadline. One submission per team is sufficient. Either of the team members can submit it.
You will demonstrate your Milestone-2 to the instructor and the TA after spring break.

References:
1. Yelp Dataset Challenge, http://www.yelp.com/dataset_challenge/
2. Samples for users of the Yelp Academic Database, https://github.com/Yelp/dataset-examples
3. Yelp Challenge, University of Washington Student Paper 1 http://courses.cs.washington.edu/courses/cse544/13sp/final-projects/p08-fants.pdf
4. Yelp Challenge, University of Washington Student Paper 2, http://courses.cs.washington.edu/courses/cse544/13sp/final-projects/p10-michelmj.pdf

Appendix A – How to create and run a SQL script file in PostgreSQL

Simply open a text editor and write all of your queries, separating them with empty lines. Make sure that each query is terminated by a ';'.
As an example, suppose that you have the following two queries, and your database name is yelpDB:
Q1:

```
select * from  reviewTable;
```

Q2:

```
select name from businessTable
where  state>'AZ';
```

Your script file should then look like as follows:

```
select * from reviewTable;
select name from businessTable
where  star>3;
```

You should save this file with a ".sql" extension.

Running The Script

In the command line, run the following :

```
psql -d yelpdb -U postgres
```

(on Windows: run cmd to open command line window)
(if `psql` is not recognized, you need to add the PostgreSQL installation path to the PATH environment variable. Alternatively, you may browse to the installation directory of PostgreSQL and then run the above command).

- You have to supply a database name to connect to. The above statement assumes your database name is "yelpdb".
- If you would be running postgreSQL with another username (other than `postgres`), replace `postgres` with that username. You will be asked to enter your password for the username you specify.

Assuming that you have saved the script file in the folder c:\myfolder, run the following in command line:

```
yelpdb=#> \i ./myscript.sql
```

(update the path of the file if your script file is not in the current directory.)
(The "yelpdb=#" here is the command prompt. Yours will look different depending on your database name.)

The above command will execute all the queries in the `myscript.sql` file **.**

Check http://www.postgresqlforbeginners.com/2010/11/interacting-with-postgresql-psql.html for a brief tutorial about interacting with PostgreSQL in the command line.