CptS -451 Introduction to Database Systems
Spring 2020

Project Milestone-2

This project is for the CptS451 students majoring in Data Analytics

Due Date: Thursday March 26, 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 UI. 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 attributes in the business JSON objects):
     - "numCheckins" : the number of total check-ins to the business.
     - "numTips" : the number of tips provided for the business.
   - ✓ Your user table should include the following attributes (in addition to the attributes in the user JSON objects):
     - "totalLikes" : the total number of likes for the user's tips.
     - "tipCount" : the number of tips that user wrote for various businesses.
     - "lat"/ "long" : latitude/longitude coordinates of the user's location.

   The default value for numCheckins, numTips, totalLikes, and tipCount attributes should be 0.
   The above names are just suggestions.

2) (5%) 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) (42%) Populate your database with the Yelp data.

- 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: Make sure to initialize *"numCheckins"*, *"numTips"*, *"totalLikes"*, and "t*ipCount*" attributes to `0`. In task-5, you will write UPDATE statements where you will calculate and update the values for these attributes.

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

  i. (*Recommended*) 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 is 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 matters. The INSERTs to referenced tables should be run before INSERTs to referencing tables.

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

4) (8%) Calculate and update the "*numCheckins*", "*numTips*", "*totalLikes*", and "t*ipCount*" attributes for each business.

- "*numCheckins*" value for a business should be updated to the count of all check-in counts for that business. Similarly, "*numTips*" should be updated to the number of tips provided for that business. You should query the "Checkins" and "Tips" tables to calculate these values.

  "*totalLikes*" value for a user should be updated to the sum of all likes for the user's tips. And "t*ipCount*" should be updated to the number of tips that the user provided for various businesses. You should query the "Tips" table to calculate these values.

  In grading, points will be deducted if you don't update these values correctly.

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

(**Note**: On some systems, the update statement may take a long time to complete. To speed up the process, you may calculate and store the calculated "*numCheckins*", "*numTips*", "*totalLikes*", and "t*ipCount*" values into temporary tables and then update the business/user tables using the values from these temporary tables.)

5) (24%) Write a 2 page paper where you describe your proposed metrics for classifying businesses as "popular" and "having good service", i.e.,

   - Popular businesses that seem to attract more customers and that have more positive reviews compared to other businesses in the same category and zipcode.
   - Businesses which provide more comprehensive and convenient service to their customers.

In your paper you need to propose and formulate your own metrics for classifying the businesses into these two groups. You should include the following in your paper:

- Provide a brief description about each of your metrics. Your description should identify all data items you will use in your metrics and how you will combine that information to argue whether a business "is popular" or "has good service".
- If you made some pre-processing on the data, extracted some information, and used those information in your metrics, make sure to explain them in your paper.

- Include all queries you used to process and analyze the data. Also include a brief description that summarizes the goal of each query and what information it extracts.
- If you stored the results of your data-analysis in any tables, include the schemas of those tables.

Note that "teams with 2 students" don't need to propose and implement the metric for "services with good service".

6) (20%) Start implementing your user interface. You should complete the following features in milestone2.
   - Retrieve all distinct states that appear in the Yelp business data and list them (e.g. in a QComboBox).
   - 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 QListWidget.)
   - When a city is selected, retrieve the zipcodes that appear in the Yelp business data in the selected city (e.g. in a QListWidget.)
   - When a zipcode is selected, retrieve and display all the business categories for the businesses that appear in that zipcode. (e.g. in a QListWidget.)
   - When a zipcode is selected, retrieve and display the zipcode statistics (#of businesses in the zipcode and top categories in the zipcode.)
   - When the user searches for businesses, all the businesses in the selected zipcode will be displayed (e.g. in a QTableWidget).

   (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. **Should be submitted in .pdf format.** Name this file *"<your-team-name>*_ER_v2.*pdf"*
2. SQL script file containing all CREATE TABLE statements. Name this file *"<your-team-name>*_RELATIONS_v2.sql"
3. SQL script file containing all UPDATE TABLE statements. Name this file *"<your-team-name>*_UPDATE.sql"
4. The source code of your application. (Please zip your source files including the QT user interface (.ui) file.) Exclude the binary files and executables.)
5. The paper explaining your proposed metrics for classifying businesses.

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.

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.