CptS -451 Introduction to Database Systems
Spring 2021

Project Milestone-2

This project is for the CptS451 students majoring in Data Analytics

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,
- ✓ 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) (45%) 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 Canvas).

  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) (10%) 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) (10%) Write half a page paper where you describe your proposed "use case". Include the following in your paper:
   - Provide a brief description of your use case.
   - If you created additional tables for this use case, include the schemas of those tables.
   - 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.

Note: Please refer to Section C in the appendix of the project description document (page 8). In the business page screenshots, the blue bordered table on the upper right is reserved for the output of your use case. However, you are welcome to change the UI design of your application and display your output in a different way.

6) (30%) Start implementing your user interface. You should complete the following features in milestone2. If you would like to **replicate your DB on multiple machines**, you can create a backup of your DB using pg_dump and restore it . See Appendix-B for more information.

- 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:
    i. Retrieve and display all the business categories <u>for the businesses that appear in that zipcode.</u> (e.g. in a QListWidget.)
    ii. 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).
- When user selects one or more categories, the search results will be filtered based on the selected business categories.

(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.)

<u>*Milestone-2 Deliverables - Checklist:*</u>

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 executable files.)
5. Your code that generates (or executes) 'INSERT' statements. (i.e., Task3 code)
6. The paper describing your proposed use case.

Create a zip archive *"<your-team-name>_milestone2.zip*" that includes all the 6 items above. Upload your milestone-2 submission on Canvas 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 in early April.

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.

Appendix B - Extract a PostgreSQL database into a script file

Assume the name of your database is “yelp”.

To dump the yelp database into a SQL-script file, run the following on the terminal (or Windows command line):

```
pg_dump -U postgres yelp > yelpdbbackup.sql
```

Copy the `yelpdbbackup.sqlfile` to the other machine, create the database “yelp”.
To reload such a script into the (freshly created) database named “yelp” :

```
psql -U postgres -d yelp < yelpdbbackup.sql
```

OR

```
psql -U postgres -d yelp -f yelpdbbackup.sql
```

If you want to create a “tar” or directory backup, check the pg_dump documentation at :
https://www.postgresql.org/docs/13/app-pgdump.html
https://www.postgresqltutorial.com/postgresql-backup-database/