M149/M116: Database Management Systems
University of Athens
Deprt. of Informatics & Telecommunications

Programming Project I
Today's Date: October 25^{th}, 2020
Due Date: December 4^{th}, 2020

PREAMBLE

In this project, you will design, implement and demonstrate a database solution to manage *"311 Incidents"* data openly published by the city of Chicago, IL. You will also provide access to your database through a web application. Residents of Chicago, may dial *311* to report incidents that affect their every day life in the city. Reports are used by the city to offer recovery, sanitation and municipal services to the public. The database termed "`311-Chicago-Incidents`" or `311CI` will be populated by data available at:
`https://www.kaggle.com/chicago/chicago-311-service-requests.`
Pertinent data of the above data set have to be *stored* in `311CI` and can be used when the system becomes operational.

PROBLEM DESCRIPTION:

Your database should include information about the *311* incidents that have been reported in a *normalized form* and should provide various queries and/or aggregations on incidents recorded. Moreover, the database should hold data related to the registered users of the web application and their actions. Below is a description of the general guidelines of the project. While you are working, keep in mind that these items make up a minimum set of requirements. Hence, you may extend the scope of your work as you see it appropriate and/or interesting.

311 Chicago Incidents data: the city provides its open data so that citizens can be assisted in making decisions about their every day life. Moreover, these data help pinpoint areas that the municipality has to timely expend effort and funds to improve life. `311CI` users are either residents or individuals carrying out data studies for enhancing their wards or neighborhood by designing coordinated civic action(s). In this context, registered users can analyze the data provided either though a set of (canned) queries or via having read-access to the database (for the more sophisticated).

There are *12* sets of information (files in `.csv` format) used to keep track of all types of incidents being reported in Chicago, IL. We are interested in the following *11* incidents reported:

1. Abandoned vehicles: vehicles reportedly open and left unattended on roads, highways and public spaces of the city that have to be attended to. Number of days of observed current status for the vehicle at the time of the report is provided as well.

2. Alley Lights Out: Lights on wooden poles reported as out (one or more). Duplicates might exist for multiple similar reports might have been filed.

3. Garbage Carts: New requests for carts are made to the city when existing one have gone missing and/or are (partially) destroyed. Duplicate requests might exist.

4. Graffiti Removal: List requests for graffiti removal and vandalism recovery services. Duplicates might exist as reported by different parties.

5. Pot Holes Reported: The city oversee 4,000 miles of road surfaces and receives reports for holes created on tarmac due to weather, accidents, and structural damage. Multiple requests may originate from different residents for the same incident.

6. Rodent Baiting: reports for rodents in public areas are reported and the Department of Sanitation has to react through its Ward Sanitation Branches. Duplicate requests might exists.

7. Sanitation Code Complaints: violations including overflowing dumpsters and garbage bin are reported and Depart. of Sanitation has to address the incident report that remains open for multiple parties reporting.

8. Lights All Out: reports the outage of three or more lights in city locations. The Depart. of Transportation oversees 250,000 street lights and has to rectify reported incidents. Duplicate requests may exist and remain open until addressed.

9. Street Light One Out: As above but with only one or two light out.

10. Tree Debris: Piles of branches appearing in public areas, streets and open spaces are reported fro removal. The same goes for broken trees that have to be removed by city trucks.

11. Tree Trims: requests for all tree shortening and trimming are listed. Crisis trimming is performed when hazardous conditions for the public exist. Open-duplicate requests may exist.

Most of the attributes in the above files are self-explanatory. Clearly, 11 tables (one for each `.csv`) will not work in a advantageous manner when it comes to querying the data provided. While developing a *normalized* schema for your database, you should strive to *not discard* any information available from the above data set.

Using `PostgreSQL`,[1] create the schema of your database, and insert *all* pieces of incident information. You have the freedom to define the relations as per your own choice/design. It is imperative that no information is ever deleted even in light of modification(s); obsolete information should be *automatically* staged to table(s) that maintain it for the time to come.

Queries: You should provide SQL queries for each of the following descriptions, considering that all of them are particularly useful to the context of `311CI`:

1. Find the total requests per type that were created within a specified time range and sort them in a descending order.

2. Find the total requests per day for a specific request type and time range.

3. Find the most common service request per zipcode for a specific day.

4. Find the average completion time per service request for a specific date range.

5. Find the most common service request in a specified bounding box (as designated by GPS-coordinates) for a specific day.

6. Find the *top-5* Special Service Areas (SSA) with regards to total number of requests per day for a specific date range (for service requests types that SSA is available: abandoned vehicles, garbage carts, graffiti removal, pot holes reported)

[1] `https://www.postgresql.org/download/`

7. Find the license plates (if any) that have been involved in abandoned vehicle complaints more than once.
8. Find the *second* most common color of vehicles involved in abandoned vehicle complaints.
9. Find the rodent baiting requests where the number of premises baited is less than a specified number.
10. Same as the above (i.e., 9) for premises with garbage.
11. Same as the above (i.e., 10) for premises with rats.
12. Find the police districts that have handled "pot holes" requests with more than one number of potholes on the same day that they also handled "rodent baiting" requests with more than one number of premises baited, for a specific day.

It is particularly important that you make sure that all above queries are executed *efficiently* by adding the necessary *indices.*

Web Application: You should provide access to your database to authorized users through a web application.

The following type of events should be handled:

1. *Registration*: a new `311CI` user has to provide the appropriate information; she/he can pick a login-name and a password. The login name should be checked for uniqueness.
2. *Querying* `311CI`: After registration, a user can start issuing queries and/or updates. In particular, a user may search for all incidents that took place in a specific street, zipcode, or issue any of the queries above. In addition, a user may insert a *new* incident. User actions should be recorded –along with their respective time-stamp– in the database.

IMPLEMENTATION ASPECTS:

You will use `PostgreSQL` as your relational database in this project. In addition, you may use any language/framework you want including `C++, Java, Python, PHP, Ruby on Rails, Django, Spring, Spring-boot, Angular, React`, etc. As a matter of fact, the use of the `https://spring.io/projects/spring-boot` is encouraged.

All the above-mentioned queries to the database could be realized via a Web user interface (UI) that presents users with input-fields (Web forms). The results have to be displayed in some manageable manner (should they are long).

You may work in any environment you wish but at the end you should be able to demonstrate your work (with your notebook or via remote access to a machine).

OVERVIEW OF PROJECT PHASES:

There are two distinct phases in this project that you will need to work on:

◇ Database

In this phase, you are required to sketch (possibly an E/R-diagram) and generate the precise database schema you need based on the problem definition. You must also provide the *SQL* queries and required indices –the latter if deemed necessary.

◇ Web Application

You will use the framework of your choice (such as `Spring-boot`, `Laravel`, `Django`, `RoR`, etc.) to offer required interfaces. You will integrate/extend the Web-based user interface that issues the queries you have developed.

COOPERATION:

You may either work individually or pick *at most one partner* for this project. If you pick a partner you should let us know who this person is. If you choose to work alone you *are not expected* to develop the functionality related to user management (registration, authentication, query logging, etc.).

REPORTING:

The final *typed* project report (brief report) must consist of:

1. A final schema design of the database used along with justification for your choices.
2. SQL code for the queries described in the `Queries` section.
3. The code of your Web Application.
4. Sample snapshots of the interface illustrating the results of the queries.

Finally as mentioned earlier, you will have to demonstrate your work.

SUPPORT:

Panagiotis Liakos (`p.liakos+@-di.`) and George Metaxopoulos (`cs3190002-@-di.` will be escorting this assignment, fill in questions, and carry out the final interviews.