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

Programming Project I
Today's Date: October 17^{th}, 2023
Due Date: November 28^{st}, 2023

PREAMBLE

In this project, you will design, implement and demonstrate a database solution to manage *server logs*. You will also provide access to your database through a web application. A server–log is a file that consists of a list of activities a machine has performed and recorded. For example, a web–server log comprises a list of page requests.

The database you will create, termed `LogDB`, will be populated with data already collected and available at:
`https://uoa2-my.sharepoint.com/:u:/g/personal/grad0990_o365_uoa_gr/ERDcKZaokndKrLzjibb4v2AB71mSiqH8NM12ots9PyPp5Q?e=6RxGBp`.
Pertinent data of the above data set have to be *stored* in `LogDB` and can be used when the system becomes operational.

PROBLEM DESCRIPTION:

Your database should include information about the server logs in a *normalized form* and should provide various queries and/or aggregations on logs 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.

Server Log Data: Different services run by the OS often maintain a history of their activities scattered in a number of files and formats. A typical web server will add an entry to its access–log for every request it services, featuring among else the IP address of the client, the request line, and a time stamp.

`LogDB` will allow for its users to perform powerful analysis on various server logs of different services and formats. 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 *3* sets of log files in `.txt` format:

1. access.log: This is a web–server log that features:
   - IP address of the client (remote host),
   - Remote name of the User performing the request. In the majority of the applications, this is confidential information and is hidden or not available.
   - User ID of the person requesting the document as determined by HTTP authentication. If the document is not password protected, this entry will be '-',
   - Time stamp,
   - HTTP method (GET, POST, etc),

- Resource requested,
- HTTP response status (200, 400, etc),
- Response size,
- Referer,
- User agent string.

2. HDFS_DataXceiver.log:
   - Time stamp,
   - Block ID,
   - Source IP,
   - Destination IP,
   - (Optional) Size,
   - Type (receiving, received, served, etc).
3. HDFS_FS_Namesystem.log:
   - Time stamp,
   - Block ID(s),
   - Source IP,
   - Destination IP(s),
   - Type (replicate, remove).

Clearly, one table for each type of log 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 data set provided.

Using `PostgreSQL`,[1] create the schema of your database, and *insert all pieces* of log information. You have the freedom to define the relations as per your own choice/design.

User Data: should include the name of every registered user, his/her login name, password, address, email, and time-stamped queries issued to the database through the web application.

SQL Queries: You should compose a SQL query to serve the following user needs, as they are particularly useful to the context of `LogDB`:

1. Find the total logs per type that were created within a specified time range and sort them in a descending order. Please note that individual files may log actions of more than one type.
2. Find the total logs per day for a specific action type and time range.
3. Find the most common log per source IP for a specific day.
4. Find the *top-5* Block IDs with regards to total number of actions per day for a specific date range (for types that Block ID is available)
5. Find the referers (if any) that have led to more than one resources.

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

6. Find the 2^{nd}–most–common resource requested.
7. Find the access log (all fields) where the size is less than a specified number.
8. Find the blocks that have been replicated the same day that they have also been served.
9. Find the blocks that have been replicated the same day and hour that they have also been served.
10. Find access logs that specified a particular version of Firefox as their browser.
11. Find IPs that have issued a particular HTTP method on a particular time range.
12. Find IPs that have issued two particular HTTP methods on a particular time range.
13. Find IPs that have issued any *four* distinct HTTP methods on a particular time range.

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

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

The following type of events should be handled:

A. *Registering with* `LogDB`: a new `LogDB` 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.

B. *Querying* `LogDB`: After registration, a user can start issuing queries. In particular, a user may search for all logs associated with a specific IP (source or destination) and issue any of SQL queries listed (1–13).

C. *Updating* `LogDB`: A user may also issue updates as she should be able to insert a *new* log of any type.

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 parameterized *SQL* stored-functions. The Web user interface (UI) will present users with input-fields (Web forms) and the data are expected to be passed on to the *SQL* stored functions. The results have to also 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 implement the stored-functions and views—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 consumes the stored-functions you created.

COOPERATION:

You may either work individually or pick *at most one partner* for this project. If you work with a partner you are *both expected to* ***contribute equally***. You should articulate the division of labor in your report **and** provide sufficient explanation while demostrating your work.

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. A parameterized stored-function for each of the requirements presented in the `Stored Functions` section.
3. The code of your Web Application, preferably through a link to a `git` repository.
4. Sample snapshots of the interface.

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

SUPPORT:

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