Universidad de las Américas Puebla

LIS4102 Cloud Computing and Big Data

José Antonio Solís Martínez

ID: 162442

Use Case 1: Polyglot Persistence

23/02/2022

Introduction

The purpose of this use case was to design a polyglot database to handle different types of data that originate from social media, as this database will be used in an application that integrates all the data from social media in just one place.
The structure of this document is very similar to the one given as the problem statement, but some specially important parts were highlighted by text in **bold**, *italic* or ***both*** to distinguish it. First we have the general idea of the application, from which I then identified the three databases that will be a part of the polyglot database, incluiding the logic behind the decision and which cloud service could be used to deploy it.
Then there's the To Do section, which outlines certain characteristics of the app, and for each characteristic, an explanation of how to tackle it is added as a sub-bullet.
After that comes the To Hand In section, in which specific deliverables are specified, and under each specification the corresponding deliverable is added. For some as a diagram and for some as text.
Finally, a small conclusion of the use case and some general reflections on the work that was done.

The application idea

Integrated Social Data aims to develop an application for integrating **personal contacts information and posts** from **several social networks** belonging to a user to **provide a global view** of such information and **provide querying functions** that can help a user have different views of the posts. For instance, organize the posts according to the **geographical region of the authors**, or **according to a period**, **count the posts submitted by a specific contact or contact group**; retrieve the **posts including images published by a particular contact at a given date**.

First database: Neo4j (Graph NoSQL)

Problem statement: The user will also have a global view of her contacts' network: connections can be related among each other by the relation "**is contact**", but they can also be connected according to a shared property inherent to their profile or expressed explicitly by the user **(i.e., family, colleagues, acquaintances, classmates).**

Why this database: The connections between users are better represented as a graph, and this graph database allows for such representations. also, Neo4j allows us to specify the type of relationship (just a contact, family, colleague, etc.)

Cloud service: Neo4j AuraDB (https://neo4j.com/cloud/aura/)

Second database: PostgreSQL (Relational SQL)

Problem statement: The objective is to design and specify the application MyNet that will enable the integration of social networks **information of users**

Why this database: The information of users is homogeneous; all users have names, last names, emails, genders, passwords, etc. A relational database can work for this purpose without any issue.

Cloud service: Google Cloud's CloudSQL (https://cloud.google.com/sql/docs/postgres)

Third database: MongoDB (Document NoSQL)

Problem statement: and the **storage of some relevant posts or contents**. These data can be **organized according to a topic, a date,** and the content will maintain information about **authorship and provenance** to share on the web.

Why this database: Social media posts are often in JSON format. MongoDB works on JSON documents and allows querying, tagging, sharding, etc., helping in organizing the data according to a *topic* or *date*. Authorship and provenance can also be added to the document without any issues.

Cloud service: MongoDB Atlas (https://cloud.mongodb.com/)

To Do

Design and specify a cloud-aware polyglot database solution based on services. We rely on your ability for imagining an original app that would use this solution and **provide an integrated global view of the social network contacts of a person**. Attention! The person wants to see an integrated set of contacts, but she/he also wants to **know whether it is a private, furtive or professional contact**.

(So know if the contact is from LinkedIn, Tinder, Facebook, etc.)

The solution must be simple, but it must have the following characteristics:

1. Deal with constructing a polyglot database, with an ETL process that can populate it by interacting with existing data services (e.g., LinkedIn, Facebook, Twitter, Deezer, Tinder, Tik Tok, Snapchat, ….)
    - ETL process: MyNet will allow users to log into the desired social media, this gives access to the posts, contacts and their respective posts. The posts are obtained in JSON format and loaded into the MongoDB database. The contact list is read and added to the Neo4j database by default related to the user with the relationship of "is contact", unless there is another type of relationship specified. Finally, the user id of said social media is added to the tuple in the PostgreSQL database of the user; that way there is a way to relate the contacts to a specific user and a specific social media, aiding in knowing if they are private, furtive or professional contacts.
2. Deployment of the polyglot database on several clouds
    - MongoDB to Atlas, Postgres to CloudSQL, Neo4j to Aura
3. Export **services** that can give access to it **for manipulating and querying data**.
    - Since the user would be logging into their desired social media using their credentials, we would have access to all the information the user themselves have available. As such, the information can be obtained via REST API calls to the social media and fed to the ETL process.
4. Define a global interface to explore (query) the polyglot database. How can global queries work on the different stores composing your polyglot store? Include the operations to be processed such that subqueries on the stores of your solution can use:
    - Selection, projection style queries
    - Aggregation and group by style queries
    - Join or correlation style queries
5. Design and specify a consolidation strategy **when data are updated on the polyglot database**. What are the BASE guarantees that your solution ensures? How do the internal processes of the system ensure them?
    - Remember that BASE stands for Basic Availability, Soft state, and Eventual consistency. The Basic Availability is inherited by the reliability of the databases used in the solution, and further ensured by deploying the polyglot database to several clouds. The Soft state and Eventual consistency are mainly handled by the ETL process. Given that the design for now doesn't have sharding or isn't distributed beyond the cloud where it is deployed, state and consistency errors shouldn't be an issue.

To Hand In

- The ***polyglot database***
    - characterizing the **data using *UML***, (Diagram 1)
    - **justifying the choice of the models used for storing the data**, (Explained in the Application Idea section)
    - **the ETL process (UML activity or sequence diagram) used for feeding it** (Diagram 2)
    - and the **functional architecture**. (Diagram 1)

Diagram 1:

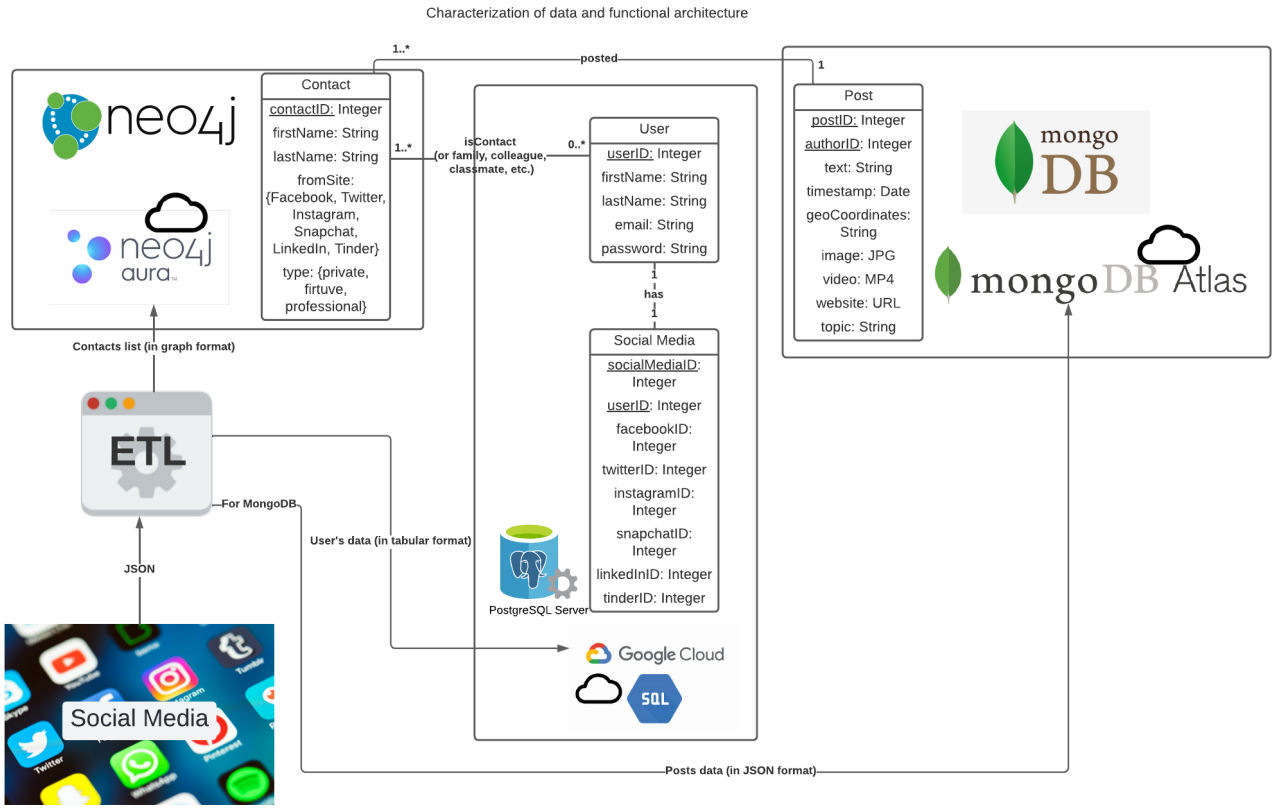
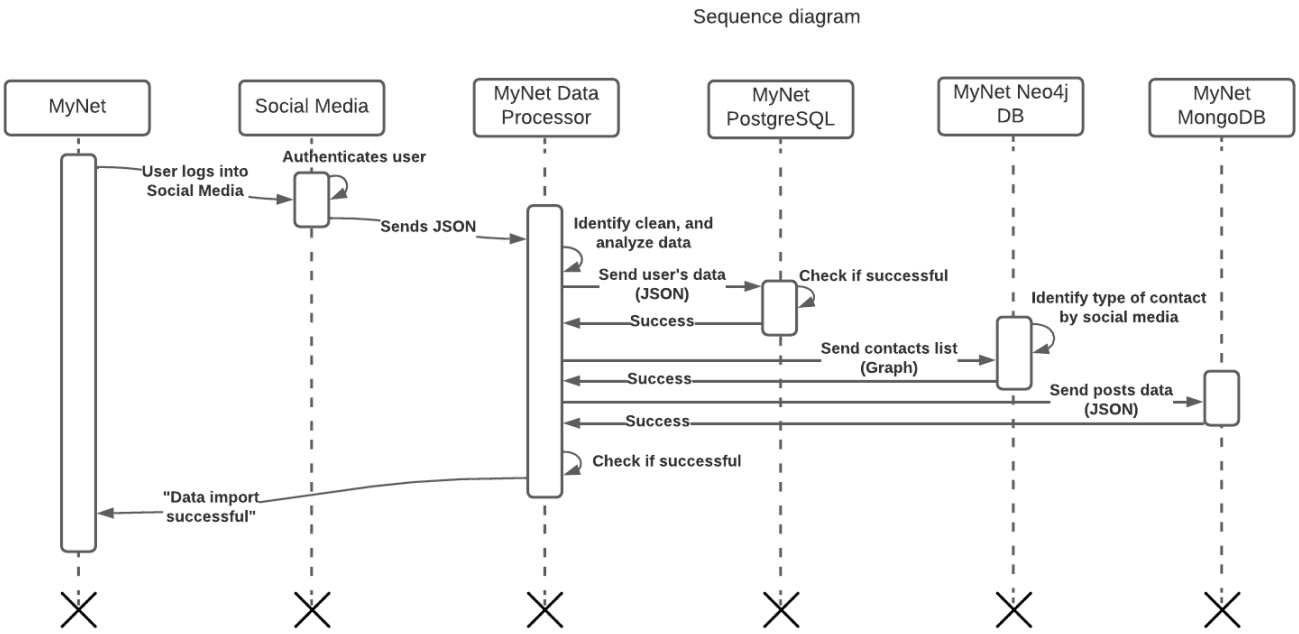


Diagram 2:



- Explain which **services** can be specified (**API and functional logic**) **for accessing the polyglot database** and how they can be **deployed on one or several clouds** (**functional architecture** as studied).
    - Logging into social media account (MyNet)

- Importing data from social media to MyNet (MyNet)
- Browse contacts (Neo4j)
- See posts from linked social media (MyNet + MongoDB)
- Filter posts by (MongoDB):
    - Location of the author
    - Date and/or time posted
    - Media content
    - Author
    - Topic
- Count posts made by a specific contact (MongoDB + Neo4j)
- Check the relationship between contacts and users (Neo4j)
- Edit user information

There are more possible services or APIs, but I just added the ones that I found to be most important and the ones the problem statement specified as more relevant. All of the services listed above are, at least in my view of my design, feasible to implement.

- **Describe the architecture of the application** that can use your polyglot solution.

    - I think the architecture that best fits the MyNet design would be a client-server application, where a server (or servers) have all the logic for accessing the polyglot database and manipulating the data, and the clients have the front-end of the application that graphically allows to see and query information; which is interpreted as API calls to the services explained above, and the server returns the result to the client, which displays it nicely.

Conclusion

As a conclusion, I can say that designing polyglot databases is very time consuming and requieres knowledge on the specifics of each DBMS to use them for their correct purpose and at their full potential. That being said, the example of processing data from social media really shows why polyglot databases are so important; it's much more time consuming to transform data so it fits in just one database, so storing it in it's nearly raw form is much more efficient.