José Mariano Portilla Landa 163449 23/02/2022

Polyglot Data management on the Cloud

Social networks

The social networks that are considered for this use case are: Facebook, Twitter, Instagram and Reddit. These are some of the most popular ones and have properties that are interesting to combine. Below are some properties of each one, they are reduced in order to simplify the use case. We can see that most of them follow the same principles having users, posts, reposting, and relationships between users.

Facebook:

- Users
- Posts: author, images, text, reactions, comments
- Relationships between users (friends, family, etc)
- Group belonging (communities)
- Repost

Twitter:

- Users
- Followers
- Retweets
- Posts: author, Text, images, Like, Comments

Instagram:

- Users
- Posts: author, images, text, comments, likes
- Followers

Reddit:

- Users
- Posts: author, images, text, comments
- Followers
- Communities

Schemas

To create the schemas first we thought about a general one (Fig. 1), this schema follows the properties of each social network in a generalized way, meaning that we do not care if in Twitter a friend is called a follower or if in Reddit a group is a community, each class has its own string id and attributes related to their purpose, some of them have association classes and their correspondent relationship to other classes.

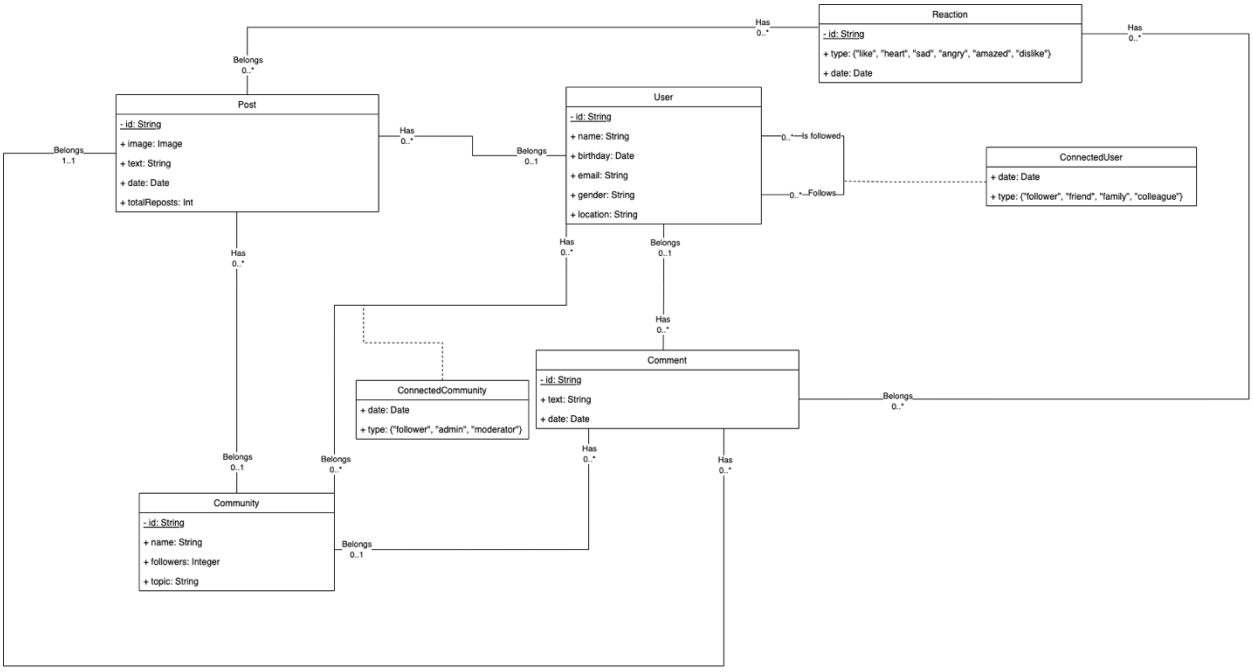

Figure 1: UML Schema of general data model.

Fig. 2 shows us the schema that was done for the document base, the rules that were followed in order to produce it were based on the ones that we had on the relational model, however they were adjusted and some of them were not taken into account. If we were constructing a relational database, then we would create a new table for the relationships that contain 0..1 if there is not a 1..1 relationship, however because in NoSQL we do not really care for null values, we just treated a 0..1 relationship as a 1..1 relationship and include the primary key in the correspondent collection (attribute id from Community is inserted in collection Comment). This was really the only rule that we follow, besides making each class a collection of course. Furthermore we thought that relationships with 0..* in both sides were naturally fit for a graph schema, so we omitted them from the document schema.

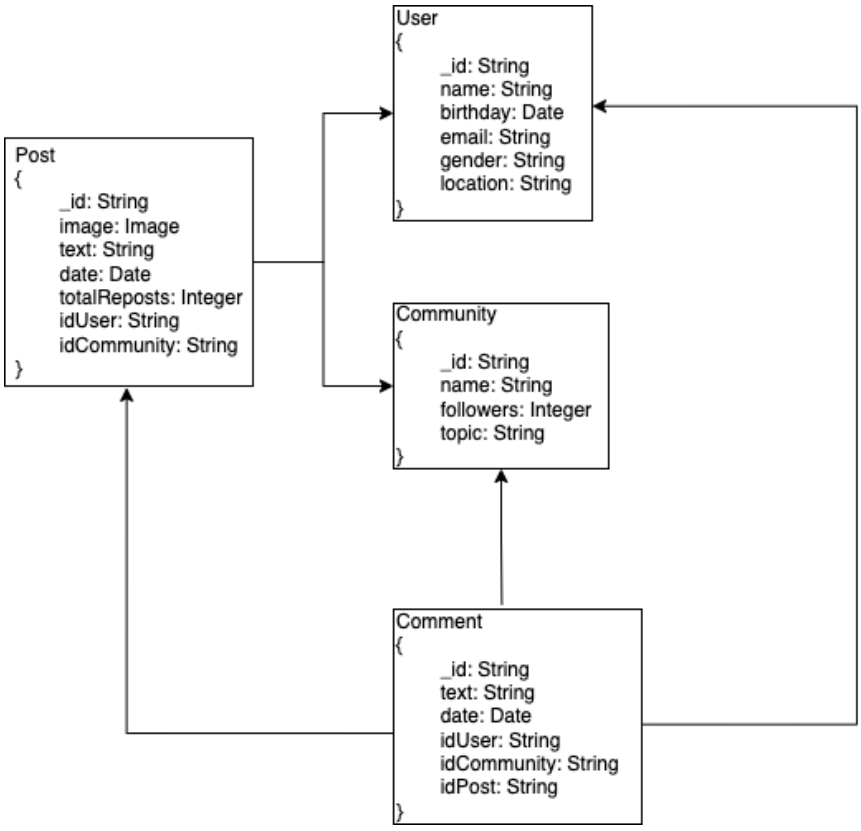

Figure 2: Document schema of the data stored in document database.

As said before, in Fig. 3 we declared the relationships and classes that were missing. A user can follow either a community or another user, the type of the relationship dictates the role that the user takes, this is to address the different types that we can have in different social networks. Furthermore, a reaction can be in a comment or in a post, with the type of the reaction as an attribute of the reaction node itself (to address the different reactions in different social media). An important observation is that we are only specifying the essential fields of data in order to save resources and avoid having two types of databases with the same information, we just need the id to identify each node and the attributes that were not considered for the document schema, any other data can be looked up in the document database.

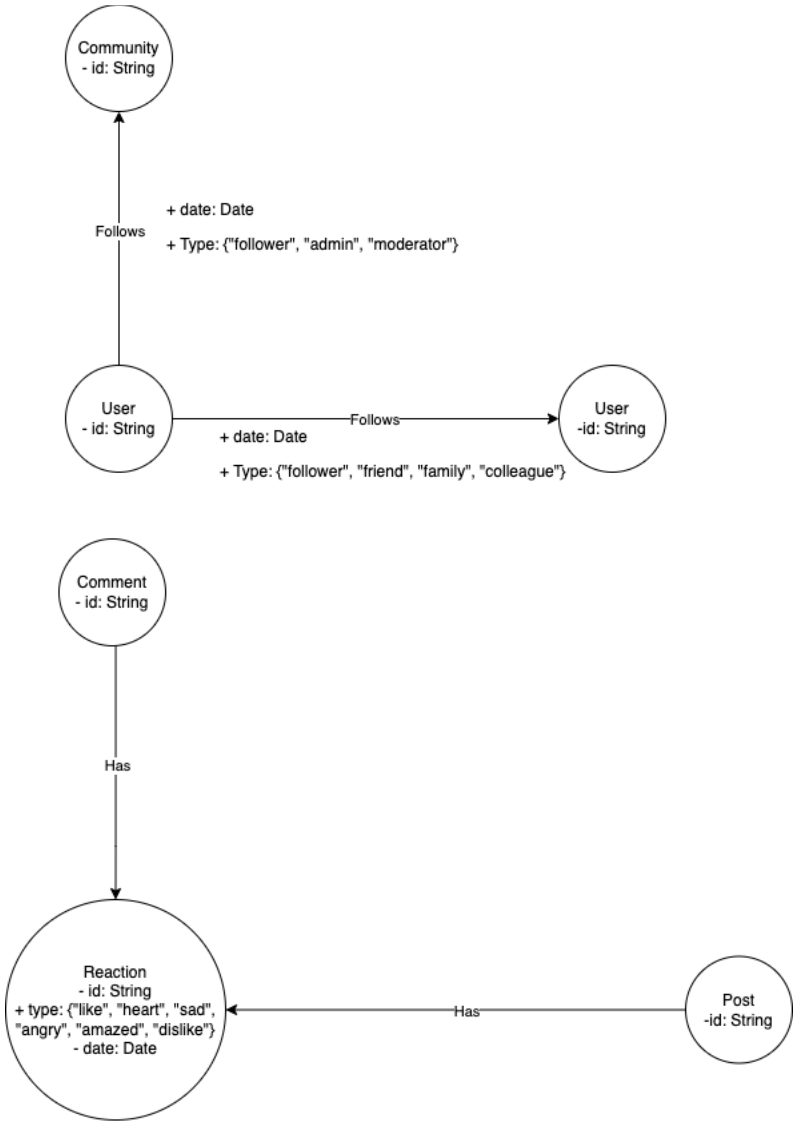

Figure 3: Graph schema of data stored in graph database.

Queries

For the document base the following queries are proposed.

- Information about the user: name, birthday, email, gender, location,
- Number of posts published by a user
- Total number of comments of a user
- Total number of reposts from all the posts that the user has
- Information about the community: name, followers, topic
- Number of posts published by a community

- Total number of reposts from all the posts that the community has
- Total number of comments of a community

For the graph base the queries that are proposed are the following.

- Number of friends/followers of users
- List of friends/followers of users sort by alphabetical order or date in which they established a relationship
- Number of communities followed by a user
- List of communities followed by a user sort by alphabetical order or date in which the user joined
- Most used reaction in comments
- Most used reaction in posts

For both of them we propose the following queries.

- View of user profile with general information as well as the users or the communities that they follow
- View of community profile with general information as well as information about the users that follow them
- View of information about individual comments (including reactions) sort by date and with a reference to the post they belong to
- View of information about each published post according to date, including reactions
- View of information about posts of other users filtered by date or type of relationship
- View of information about posts of communities filtered by date

Guarantees

The reason why we choose non-relational databases was for the following reasons.

- The schema is prone to change
- Friends or followers are well suited for graph databases because we may have queries that traverse multiple levels through graph data such as friends of friends queries
- Data variety and possibly huge amounts of data
- We do not require all the safety that ACID guarantees provide to us, we can do fine with BASE because we value availability over consistency

- Following the CAP theorem, we would sacrifice consistency to get availability and partition tolerance
- Even though we can be fine without ACID it is important to note that most graph databases use an ACID consistency model

Architecture

In Fig. 4 we can see the general architecture for the solution, while the user is interacting with the application, the backend recollects data from users via the API's that these platforms have and makes CRUD operations in order to reflect the information that we have in these social networks. Replicas of data are periodically managed in order to ensure availability, queries are requested and processed by our application whenever the user changes the view.

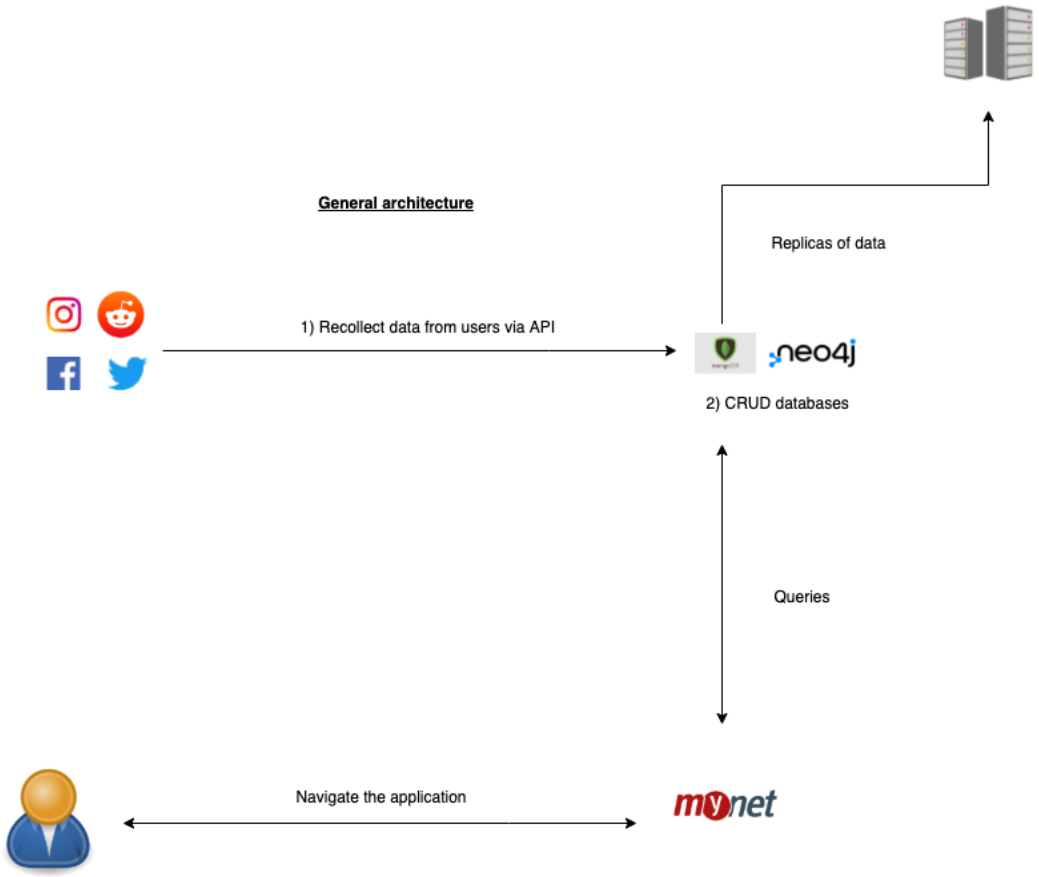

Figure 4: General architecture of application

Fig. 5 introduces the ETL diagram when requesting data from Facebook, Twitter, Reddit and Instagram. The extraction process is via API and then the backend handles where the data is going to go, either MongoDB (document database) or Neo4j (graph database), databases to process

queries are selected depending on the nature of the data that the query requires. Finally, the data is loaded into MyNet application.

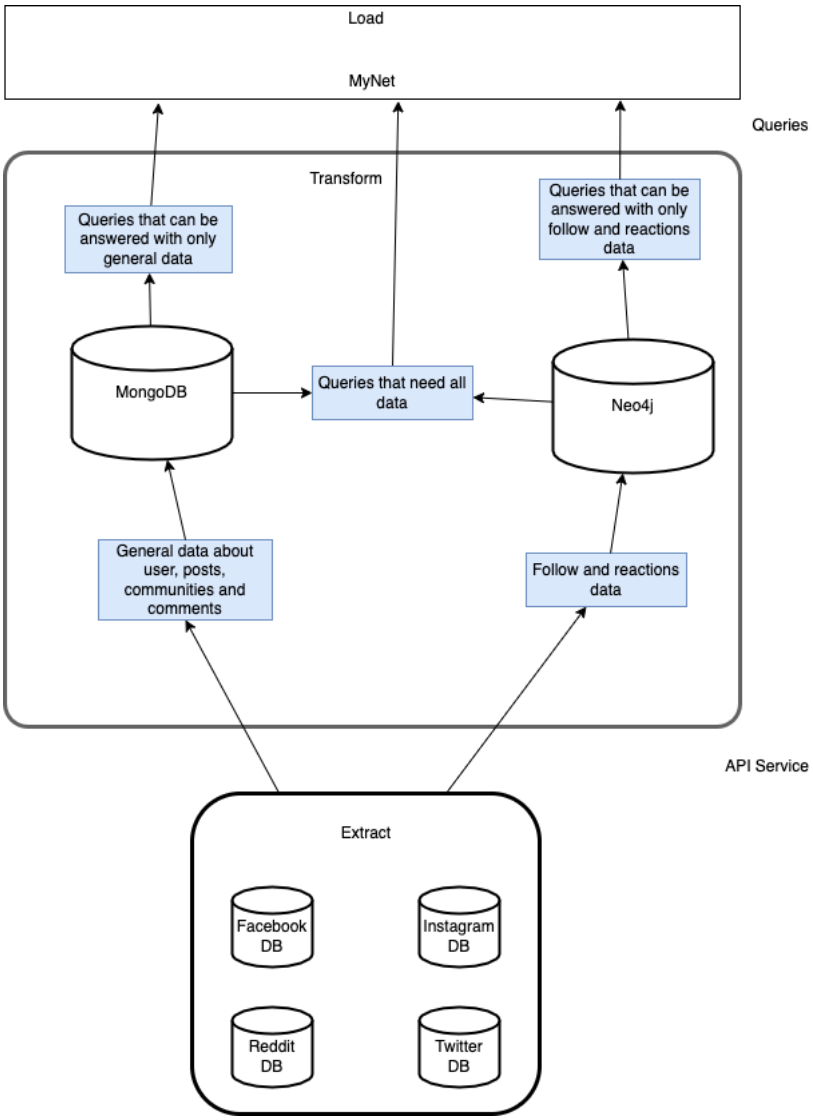

Figure 5: ETL process used for feeding application.