A Study of Database Models for Social Network Analysis

Mariana D. A. Salgueiro[1], Sérgio Lifschitz[1], Edward Hermann Haeusler[1], Verônica dos Santos[1], Alexandre A. P. Heine[1]

[1]Departamento de Informática – (PUC-Rio) - Rio de Janeiro - RJ

{msalgueiro, sergio, hermann, vdsantos, aheine}@inf.puc-rio.br

Abstract. *This paper discusses conceptual and logical data models for social media datasets and applications. On the one hand, we focus on the data representation requirements from the available APIs of some social network systems. On the other hand, we consider those application requirements for information manipulation. We propose a conceptual meta-model and one possible instantiation. We also give preliminary practical results considering relational and graph database systems.*

1. Introduction

There are currently many research works that deal with social network analysis (SNA), extracting data and relationships from the connected information usually expressed by social media applications (e.g. Facebook). Most (if not all) works focus on the available data from the social media APIs, querying and evaluating CSV, TSV or JSON files or structures. In the presence of very large datasets, we expect to work with actual database management systems (DBMS), which are suitable for dealing with large volumes of data.

In this paper, we propose a conceptual meta-model to represent and manage Multimedia Social Networks (MSN). We first instantiate this meta-model for Twitter and obtain its Logical Model in a *Top-Down* process. Next, we analyze the Twitter dataset to identify data requirements gaps between the logical model and its data source. Then we derive the physical model and load the dataset using both a relational and graph DBMS. Finally, we show some experiments with SQL queries and SNA using SPARQL extensions.

Our contributions in this work are: (1) a conceptual meta model that describes at a high level any MSN, (2) a bi-directional data modeling process that deals with both the instantiation of the application model of a specific MSN and the mapping of the discrepancies between this model and the dataset provided by the platform, and (3) the characterization of an SNA task implemented in DBMSs with different data models.

2. Context, Motivation and Related Works

SNA applications extract social datasets using MSN interface. For example, eTC (ePOCS Twitter Crawler) is an application that collects data using Twitter API [Heine et al. 2021]. Users can specify extraction criteria, and the system schedules the request. However, handling large files of collected data on personal computers presents low performance.

In [Stieglitz et al. 2018], the authors attest that it can be argued that social network data share many characteristics of *Big Data*. When data takes up so much physical space that it does not fit in memory, we have a volume problem. Since some datasets collected by users can be significant in volume and complexity, a DBMS could store it after data

collection as suggested in [Angles et al. 2013], which allows to generate basic statistics and analyze the data through queries, without storing all the data on memory at the same time. Considering this issue, the first step we took to build a data collection and analysis pipeline for any MSN was the database modeling.

An application data model specifies how entities and their relationships are represented and operated [Davoudian et al. 2018]. Such models can be created using different abstraction levels (Conceptual, Logical, and Physical) through two approaches: forward (*Top-Down*) and reverse (*Bottom-Up*). In this work we navigated in both directions to deal with data requirements from users and data available from MSN platforms.

In order to incorporate SNA functionalities target to any MSN using large data volumes, we searched for MSNs data models that can accomplish both user requirements and data available through their APIs. We found [Reinhardt et al. 2010] as a general network model for social networks, but it does not cover conceptual layer and current entities attributes. In [Bouraga et al. 2016] the authors proposed an unified Conceptual Model to allow user profile portability between MSNs. We analyzed their model and the separation between implicit and explicit informations doesn't attend our users requirement for SNA tasks. A conceptual model is essential to achieve an effective communication with our users, from different knowledge areas, so we decided to model our own.

After that, with the purpose of analyzing tweets' datasets using a graph representation of our model, we identified an ontology for MSNs called SIOC[1] (Semantically-Interlinked Online Communities) Core Ontology, which represents posts, user accounts, and other entities used on MSNs; and a public corpus of tweets using RDF, called TweetsKB [Fafalios et al. 2018], which makes possible integration with other public knowledge bases. Our approach on graphs was to explore relationships between tweets by using database predicates which extends SPARQL to facilitate the use of SNA, while SIOC focus on the ontology RDF/S schema and TweetsKB uses SNA without traversing the predicates which relate tweets by reference.

3. Database Modeling

In [Carr and Hayes 2015], the authors define *Social Media* as persistent channels of masspersonal communication facilitating perceptions of interactions among users, deriving value primarily from user-generated content. From that perspective, we can highlight two main basic concepts: users and user-generated content. Our meta-model is, therefore, composed of these two central concepts represented by the entities *User* and *Post*, as shown in Figure 1. *Posts* are content shared by users and can be texts, images, videos, etc. Other users can usually like, comment or reshare the post. The relation *Reacts_to* represents likes, whereas the *References* represents comments or reshare actions. This meta-model contains constructs that can be mapped in elements of a specific MSN platform to generate its application data models.

Twitter is a micro-blogging service that allows its users to post the so-called tweets that can have a maximum of 280 characters and can contain images, videos, gifs, hashtags, etc. The basic actions on Twitter are: following a user, tweeting, liking a tweet, replying to a tweet, quoting a tweet and retweeting a tweet. Instantiating this MSN

[1] `https://www.w3.org/Submission/sioc-spec/`

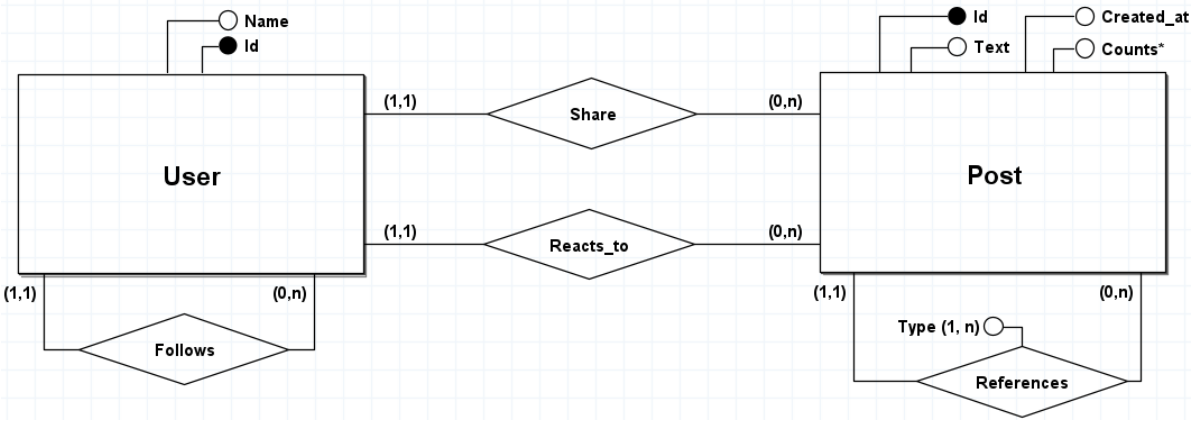

Figure 1. MSN Meta-Model

corresponds to the representation of these actions as relationships: following a user = *User-Follows-User*, tweeting = *User-Share-Post*, liking a tweet = *User-Reacts_to-Post*, replying/quoting/retweeting a tweet = *Post-References-Post*.

Following the conventional rules in mapping conceptual models to relational logical models, we generated three tables *User*, *Post* and *Type*. We want to understand what questions we can answer with this model and what are the ones we cannot. Within the ones we cannot, we aim to divide them into two categories: (i) when data is not available from the APIs, even though we were able to model it, so there is no way for us to answer them and (ii) there are queries in the relational model that are too complex, so maybe we can answer them, but it is not the best way in doing it.

As an example of a gap between the Logical model and Twitter's data source, we created the terminology *direct retweet* and *indirect retweet*. A direct retweet is when a user directly retweets a tweet that shows on their timeline. An indirect retweet is when a user retweets a retweet; so the user retweets a tweet because someone else retweeted it and that's why that tweet is showing on their timeline. On Twitter's API response, indirect retweets count as direct retweets. So if we have user 1, user 2 and user 3: user 1 tweeted a tweet, user 2 retweeted user 1's tweet and user 3 retweeted the retweet made by user 2 (see Figure 2), the response shows that user 3 directly retweeted user 1 (see Table 1).

Figure 2. Retweets notifications from Twitter UI

From the Logical model, it was implemented the Physical model on PostgreSQL. To take advantage from the idea of tweets self-referencing each other, a RDF/S schema

Table 1. Retweets extracted from Twitter API

Users	1	2	3
text	test	RT @TwitterDev: test	RT @referenced: test
tweet_id	**1541580**	154824577	15318912
referenced_tweets_type	null	retweeted	retweeted
referenced_tweets_id	null	**1541580**	**1541580**
author_id	12744554	3150497	1270577
author_name	Twitter Dev	mari	maxteralex

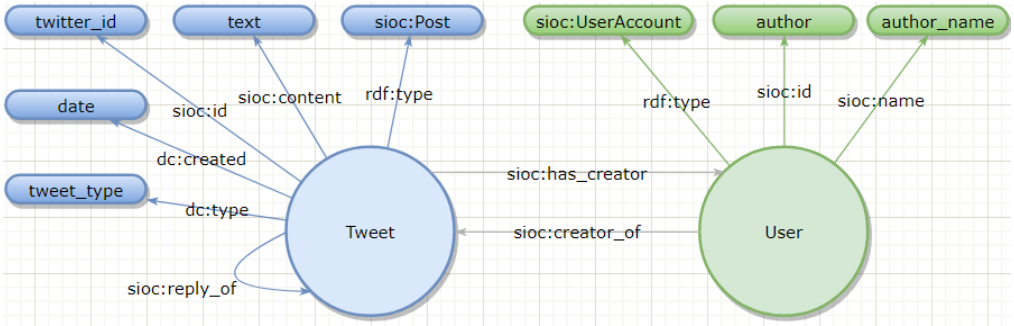

Figure 3. Twitter RDF/S Schema Proposal

was created based on the conceptual schema use case for Twitter and the SIOC Core Ontology as shown in Figure 3. The RDF/S schema represents each attribute in a triple format (subject, predicate, object) and the tweet's reference to another tweet as the predicate *sioc:reply_of*. From this schema, it was decided to use AllegroGraph, a graph DBMS, since it implements SNA predicates to use with SPARQL.

4. Preliminary Results of an SNA Task

To implement both the relational and graph representations, an example dataset was collected using the eTC historic search tool, which collects tweets from a specific date range according to a search theme specified by the user, and was preprocessed using Python, removing some columns and non-compatible characters using Pandas[2].

On relational DBMSs, it is possible to create queries to identify some major concepts of SNA centrality of groups and its main actors. The concept of centrality measures allows researchers to infer how information flows on a social network by calculating how connected a certain group or actor is to the network in comparison to the total number of connections [Borgatti 2005]. In Frame 1 a SNA query that calculates centrality measure using SQL is presented. In this query, it is calculated the actor centrality for a specific analysis, based on the total number of retweets of this person's tweets and the total amount of retweets of the whole analysis.

For the graph implementation, the example dataset was converted into RDF triples on RDF+XML file format, using Rdflib[3] package and loaded to the DBMS using AllegroGraph API. On AllegroGraph, SNA, geolocation and temporal predicates are called

[2]https://pandas.pydata.org/
[3]https://rdflib.readthedocs.io/en/stable/

```
WITH main_actors as
    (SELECT username,
        SUM(retweets) as sum_retweets
    FROM tweet, user
    WHERE user.id = tweet.fk_user_id
    AND analysis_id = X
    GROUP BY username
    ORDER BY sum_retweets),
  total_retweets (total) as
    (SELECT SUM(retweets)
    FROM tweet
    WHERE analysis_id = X)
SELECT username,
   sum_retweets/total as actor_centrality
FROM main_actors, total_retweets
```

```
SELECT DISTINCT ?actor ?centrality {
{ SELECT ?group
  {?tweet rdf:type sioc:Post .
    ?group sna:egoGroup
        (ex:treeTweets ?tweet 10) .
    ?group sna:size ?size . }
  ORDER BY DESC(?size)
  LIMIT 10 }
 (?actor ?centrality)
    sna:actorDegreeCentrality
    (ex:treeTweets ?group) . }
ORDER BY DESC(?centrality)
```

Frame 1. SNA Relational Query (to the left) and SNA Graph Query (to the right)

SPARQL Magic Properties[4]. To prepare queries with the purpose of analyzing the relationship between posts of a theme or discussion on a social network, first, it is necessary to implement a generator. That is, a predicate or a SPARQL query that describes which predicates are traversed from an input to find direct neighbors, subjects or predicates, as an output. Therefore, these generators allow to answer questions related to SNA on a specified dataset, for example, to find the structure of a network, its most sizeable groups, and their most influential actors on a certain information flow.

From the example tweet dataset transformed according to the RDF/S schema, it was created a generator which uses the "sioc:reply_of" predicate to find tweets which refer to the input tweet. Then, in order to identify the 10 most sizeable groups in this dataset's network and who are their most influential actors, it was used the SPARQL (graph) query in Frame 1[5] with the predicates: "sna:egoGroup" to group the tweets, "sna:size" to find the size of these groups, and "sna:actorDegreeCentrality" to know the centrality of each actor related to how many other tweets referenced another one, and then discover who are the main actors according to the result.

Finally, as displayed in Figure 4, the result of this query was exhibited on Gruff, an AllegroGraph extension which generates graphical representations for a query and its result. As it is possible to observe, in a group, there are tweets at the center of a discussion, the most influential actor (in yellow), and some tweets which reference this influential tweet and are referenced by others, the intermediate actors (in red). In this representation we use tweets as nodes and the relation between tweets as edges to represent a group of 127 tweets using an hierarchical tree. Conversely, it is possible to use the tweet's creator names to identify who leads a discussion and who takes the information forward.

[4]https://franz.com/agraph/support/documentation/current/magic-properties.html

[5]Prefixes declaration were omitted due to space limitations

Figure 4. SNA Graph Result on Gruff

5. Conclusions

We have proposed in this paper a conceptual meta-model for describing MSNs. A straightforward instantiation for Twitter also enabled both a relational and graph logical data schemas, used to study SNA queries using SQL and SPARQL. As future works, we plan to generate instantiations for other MSNs such as Facebook and Tiktok to test the potential of the proposed conceptual meta-model and evaluate multi-model or polyglot modeling to further understand SNA querying on top of actual database systems.

References

Angles, R., Prat-Pérez, A., Dominguez-Sal, D., and Larriba-Pey, J. L. (2013). Benchmarking database systems for social network applications. In *First International Workshop on Graph Data Management Experiences and Systems*, page 15.

Borgatti, S. P. (2005). Centrality and network flow. *Social Networks*, 27(1):55–71.

Bouraga, S., Jureta, I., and Faulkner, S. (2016). Towards data portability between online social networks, a conceptual model of the portable user profile. *Int. J. Virtual Communities Soc. Netw.*, 8(3):37–54.

Carr, C. T. and Hayes, R. A. (2015). Social media: Defining, developing, and divining. *Atlantic Journal of Communication*, 23(1):46–65.

Davoudian, A., Chen, L., and Liu, M. (2018). A survey on nosql stores. *ACM Computing Surveys (CSUR)*, 51(2):1–43.

Fafalios, P., Iosifidis, V., Ntoutsi, E., and Dietze, S. (2018). Tweetskb: A public and large-scale rdf corpus of annotated tweets. In *European Semantic Web Conference*, pages 177–190. Springer.

Reinhardt, W., Varlemann, T., Moi, M., and Wilke, A. (2010). Modeling, obtaining and storing data from social media tools with artefact-actor-networks. In *LWA 2010 - Lernen, Wissen & Adaptivität, Workshop Proceedings*, pages 323–330.

Stieglitz, S., Mirbabaie, M., Ross, B., and Neuberger, C. (2018). Social media analytics – challenges in topic discovery, data collection, and data preparation. *International Journal of Information Management*, 39:156–168.