Data Management - Project Outline

by Tinessa Francesco and Franzoso Antonio

This document tries to summarize crucial points we aim to discuss in the Data Management project's discussion.

Project was about the use of a NoSQL tool, using Neo4j to create a graph database from a social media app dataset, comparing it with a relational one.

Load data into the database

Import csv file into Neo4j transforming a relational database into a graph database. To do so we had to:

- Pre-process the csv file (using a python script) in order to match the syntax accepted by neo4j
    - use commas as delimiters for columns
    - in case of multivalue columns (like tags or friends) write every value separated by the pipe characted (e.g "tag1|tag2|tag3")
- Write scripts using Cypher query language to correctly import nodes and relationship from csv file to Neo4j.
    - Nodes are:
        - User (userId, screenName, avatar, followersCount, followingCount, lang)
            - uniqueness constraint on userId
        - Tag (text)
            - uniqueness constraint on text
        - Post (postId)
            - uniqueness constraint on postId
    - Relationships are:
        - FOLLOWS (from User to User)
        - TWEETED (from User to Post, with attribute timestamp)
        - HAS_TAG (from Post to Tag)

> A constraint create an index on such attribute, in order to speed-up queries that tries to MATCH on that attributes.

> Later, depending on the queries that we would like to execute is it possible to create indexes on different attributes.

Query the database

Think about common queries in a social network-domain that could (or could not) benefit from the use of graph database.

For each one of them compare querying time between graph and relational database (need to decide which RDBMS to use).

> Try the queries in different scenarios, changing the number of rows considered in the original dataset, in order to show how querying time changes based on number of records in the database. Show results in tables/plots.
> e.g. 1k rows -> 5k rows -> 10k rows -> 20k rows -> 40k rows (all)

Starting from basic social media features queries to more interesting ones they could be:

1. Get Users based on their screenName.
2. Get Posts containing a specific Tag.
3. Get followers of followers of ... (up to a certain *k*-degree) of a specific User.
4. Find most influencing Users in our social graph based on their follower number.
5. Find most influencing Users in our social graph according to centrality algorithms, in general and based on a specific Tag.
    - Could be interesting to compare this with query #2 to identify potential differences
6. Get Users interested in a specific Tag.
7. Get most trending Tags across all Users.
8. Get most trending Tags across most influencing Users.
9. Based on a Tag, get similar Tags.
    - May need to clusterize Tags into more general topics or subjects - Need to understand if is ML involved
10. Based on a User, suggest him more Users to follow.
    - Still need to understand how this could work, query #3 can help with that
    - **Idea**: clusterize Users into similar ones?
11. Based on a User, suggest him Post that he could be interested in.
    - This could help populate its feed (called Timeline in X, For You in TikTok, etc.)

View the database

Talk about benefits of graph databases with regards to visualization of data.

Specific tools such Neo4j Bloom provide simplified visualization of graph database, allowing non tech people to quickly inspect the data in an interactive way.