Project

This project asks you to study the data on Twitter communications of state legislators and presidential candidates around the 2016 presidential election. The data is described from two different backend DBMS: MySQL and Neo4j. You are given tweets of these Twitter user accounts (handlers) in two different formats: 1) csv files and 2) Neo4j graph database. The tasks are as follows.

- Design an ER diagram from the given database requirement.
- Design relational schemas to store the data from the csv files and import the data from the csv files into MySQL Server.
- Write SQL queries for the questions specified in this project description.
- Write Cipher queries for the questions specified in this project description.
- Investigate the query performance between MySQL and Neo4j.

Database Requirement

- A tweet has the following properties: id, retweet count (the number of retweets of this tweet), retweeted (whether this tweet has been retweeted), tweet text, created at (timestamp---number of milliseconds since 1/1/1970---in which the tweet was posted), which is converted into the corresponding day, month, and year. The id is unique among all tweets. A tweet must have the user who posted it. If the posting user is deleted, all tweets posted by the user must also be deleted. A retweet has the same tweet text but has a different id from the original tweet. The attribute retweeted is not needed in the database.
- A tweet may have zero or more hashtag in it. Each hashtag has a unique name and must be used in at least one tweet. A tweet has zero or more URLs. Each URL has a unique URL address and must appear in at least one tweet. If a hashtag is deleted, all tweets that have the deleted hashtag must also be deleted. If a URL is deleted, all the tweets that have the deleted URL must also be deleted.
- A tweet may mention zero or more user accounts. A user account can be mentioned in zero or more tweets.
- A user account has the following properties: name, screen name, followers (indicating the number of followers), following (indicating the number of people this user follows), sub_category, category, location, and the state the user lives. A user lives in at most one state. The sub_category indicates the party to which the user belongs. The values of this attribute are “GOP”, “Democrat”, “na”, or null. The values of the category attribute are senate_group, presidential_candidate, reporter, Senator, General, or null. The name property can be null, but the screen name is unique among all users and cannot be null. Presidential candidate accounts are not associated with any state. Each state can appear as either full name or abbreviation. For example, the state of Florida appears as “Florida” or “FL”. Furthermore, “na” denotes the user account without an associated state. Check the data from the csv files to understand the kind of data to be maintained. The updateStates.sql is provided to make the state name consistent.

Neo4j Graph Database

The above database requirement was modeled using a property graph data model. Figure 1 shows the conceptual schema of the graph database. Each rectangle represents a group of nodes with similar properties. We use Neo4j labels to implement the grouping. The node labels are State, User, Century, Year,

Month, Day, Tweet, Url, and Hashtag. In the diagram, User (Posted) and User (Mentioned) both have the same User label. The edge labels are FROM, MENTIONED, POSTED, URL_USED, TAGGED, HAS_TWEET, HAS_YEAR, HAS_MONTH, and HAS_DAY. Edges in this database have no attributes.

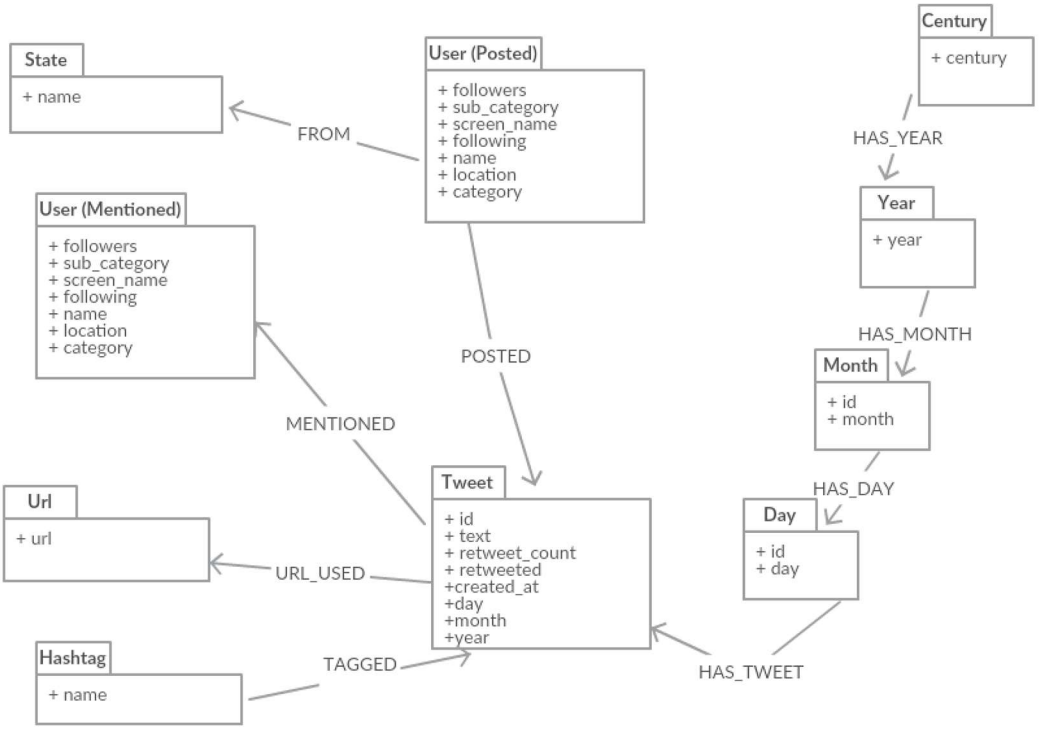

Figure 1: Schema of the graph database of tweets of state legislatures and presidential candidates for the 2016 presidential election.

The design of this data model follows the basic guidelines for graph database design. Nodes represent entities and a directed edge represents a relationship between two nodes. Nodes also represent properties of a tweet with possible multiple values such as hashtags and URLs. Tweet nodes have properties: id, retweet_count (the number of retweets of this tweet), retweeted (whether this tweet has been retweeted), tweet text, created_at (of type long representing the timestamp---the number of milliseconds since 1/1/1970---in which the tweet was posted), day, month, and year. User nodes represent user accounts and have the following properties: name, screen_name, followers (indicating the number of followers), following (indicating the number of people this user follows), sub_category, category, and location. The sub_category indicates the party to which the user belongs: 'GOP', 'Democrat', 'na', or null. The category values are senate_group, presidential_candidate, reporter, Senator, General, or null. The name property can have an empty string as a value.

State nodes have the name property containing the name of the state. There is one "na" State node for the user without state information such as presidential candidates. Hashtag nodes have the tag name property. Url nodes have the url property that can also be an empty string.

This model also uses a time tree created to support time range queries (e.g., list tweets posted during some months). The century node has the property century with an integer value of 21 to represent the current century. Each year node has edges only to the month nodes of that year. Each month node has edges only to the day nodes of that month only. In other words, the day nodes are not shared across

multiple months. A day node has edges to tweets posted on that particular day of that particular month, year, and century.

Table 1: Functionalities of the web application

ID	Description
Q1	List k=*5* most retweeted tweets in a given month = 1 and a given year = 2016; show the retweet count, the tweet text, the posting user's screen name, the posting user's category, the posting user's sub-category in descending order of the retweet count values **Rationale:** This query finds *k* most influential tweets in a given time frame and the users who posted them.
Q2	Find *k=5* hashtags that appeared in the most number of states in a given year = 2016; list the total number of states the hashtag appeared, the list of the distinct states it appeared (FL is the same as Florida*), and the hashtag itself in descending order of the number of states the hashtag appeared. **Rationale:** This query finds *k* hashtags that are used across the most number of states, which could indicate a certain agenda (e.g., education, healthcare) that is widely discussed. **Hint:** Use group_concat() function to create a list *This requires making sure that each state is represented using one name, either FL or Florida, you can use the provided updateStates.sql file to update the states name
Q3	Find k=12 users who used at least one of the hashtags in a given list of hashtags = [HappyNewYear,NewYear,NewYears,NewYearsDay] in their tweets. Show the user's screen name and the state the user lives in descending order of the number of this user's followers. **Rationale:** This is to find k users with similar interests.
Q4	Find top *k=5* most followed users in a given party. Show the user's screen name, the user's party, and the number of followers in descending order of the number of followers. Show your results with subcategory to be 'GOP' and 'Democrat' **Rationale**: This query finds the most influential users measured by the number of followers
Q5	Find the list of distinct hashtags that appeared in one of the states in a given list = [Ohio, Alaska, Alabama] in a given month = 1 of a given year = 2016; show the list of the hashtags and the names of the states in which they appeared. **Rationale:** This is to find common interests among the users in the states of interest.
Q6	Find *k=5* tweets (with the given hashtag = Ohio) posted by republican (GOP) or democrat members of a given state = Ohio in a given month = 1 of a given year = 2016. Show the tweet text, the hashtag, the screen name of the posting user, and the users' party **Rationale:** This query explores the context in which the hashtag was used
Q7	Find users in a given sub-category = 'GOP' along with the list of URLs used in the user's tweets in a given month = 1 of a given year = 2016. Show the user's screen name, the state the user lives, and the list of URLs **Rationale:** This query finds URLs shared by a party.
Q8	Find *k=5* users who were mentioned the most in tweets of users of a given party = 'GOP' in a given month = 1 of a given year = 2016. Show the user's screen name, user's state, and the list of the screen name of the user(s) who mentioned this user in descending order of the number of tweets mentioning this user.
Q9	Find *k=5* most used hashtags with the count of tweets it appeared posted by a given sub-category = 'GOP' of users in a list of months = [1, 2, 3] of a given year = 2016. Show the hashtag name and the count in descending order of the count.

Steps you may follow to start the project

1. Install neo4j community server and refer to Neo4jInstallation file for the steps of installation
2. Load the provided graph database tweet.zip into neo4j by first unzip tweet.zip and then copy the graph.db to the folder neo4j-community-3.5.17/data/database
3. Design your ER diagram base on the database requirement mentioned above. Save it in ERdiagram.pdf.
4. Check the provided dataCSV file to have an overview of the data used for MySQL.

5. Write queries to create tables based on your ER diagram and the data type and format in dataCSV. Save your create table queries in ProjectDDL.sql.
6. Load data from dataCSV into your created tables. HelpForProject provides an example of how to load data from csv files into mysql workbench, the forimport.sql file is used in the example. Save your insert table queries in ProjectInsert.sql.
7. After you have data loaded successfully into tables. The updateStates.sql file is provided to change all states to full name and make it consistent.
8. Write queries for questions in the table above with SQL and Cipher. Save them in Queries.sql and Queries.cipher correspondingly. The outputs file provides the correct output of all queries in table 1 for you as a reference to check the correctness of your query. Q5output and Q7output are for Q5 and Q7 which are not included in the outputs file.
9. Fill table2 with the server run time for your queries in SQL and Cipher. Save your result in Performance.pdf.

Tasks and deliverables

1. Design an ER diagram to model the database requirement. Use only the ER notations we studied in class.

   Submission
   - ERdiagram.pdf that includes your ER diagram design

2. Design a relational database to store the data from the given csv files. The design needs to minimize unnecessary redundancy and to ensure that all constraints are enforced. Import the data from the csv files into MySQL Server 8.0.

   Submission
   - ProjectDDL.sql that consists of SQL DDL statements that create the schemas and integrity constraints.
   - ProjectInsert.sql to insert the data from the given .csv files into MySQL Server. Consult this page for bulk loading of data (https://dev.mysql.com/doc/refman/8.0/en/load-data.html).

3. Implement all functionalities in Table 1 in SQL against the database you design and Cypher queries against the given Neo4j database.

   Submission
   - Queries.sql that has all the queries against your designed schema
   - Queries.cipher that has all the Cypher queries against the given Neo4j database
   - Performance.pdf that has Table 2 filled up

Table 2: Average server execution time reported by MySQL server vs Neo4j server over 5 runs for each query

Query ID	Average server execution time reported by MySQL server over 5 runs for each query (ms)	Average server execution time reported by Neo4j server over 5 runs for each query (ms)
Q1		
Q2		
Q3		
Q4		

Q5		
Q6		
Q7		
Q8		
Q9		