CS 3810 - Principles of Database Systems (Fall 2019)

Activity 20

Neo4J: Clustering

In this activity you will create a social graph using Neo4J and apply a community detection algorithm to identify groups of individuals that are highly interconnected in the graph. First download the CSV dataset karate_club_network.txt with the social graph of students in a karate club. Each line of the file describes the "knows" connection between two nodes. When you are done downloading the file, create a Neo4J project + database with the name `karate-club`, start the database and open Neo4J Browser. Place the CSV dataset in Neo4J's import folder then run the following commands:

```
LOAD CSV FROM 'file:///karate_club_network.txt' as line FIELDTERMINATOR '\t'
MERGE (a:Person {id: line[0]})
MERGE (b:Person {id: line[1]})
MERGE (a)-[:knows]->(b);
```

One way to detect communities in a graph is to look for partitions that have high modularity scores. Modularity is a scale value between -1 and 1 that measures *the density of edges inside communities to edges outside communities*. In other words, modularity is higher when *the number of edges that fall within the partition is significantly higher than what would be expected if edges were distributed at random*.

The Louvain algorithm for community detection is based on the modularity concept. It uses a greedy optimization approach in two steps: first it looks for small communities by optimizing modularity locally; the algorithm then repeats itself aggregating small communities into larger ones by looking at each small community as a single node.

Use the following code to run the Louvain algorithm to detect communities in the karate club graph. Then export the results into a CSV file named `communities.csv`.

```
CALL algo.louvain.stream('Person', 'knows')
YIELD nodeId, community
RETURN algo.asNode(nodeId).id AS id, community
ORDER BY community;
```

Using the `communities.csv` file, apply different labels to each community (for example, `comm0`, `comm1`, etc.) so you can display them later with different colors on Neo4J Browser. Your final result should look like the following: