CSE7053 [Social Network Analysis] Course Term Project Report
by
Hande KARA and Mesut ORMANLI
Department of Pure and Applied Sciences
Marmara University

What is the purpose of this project?

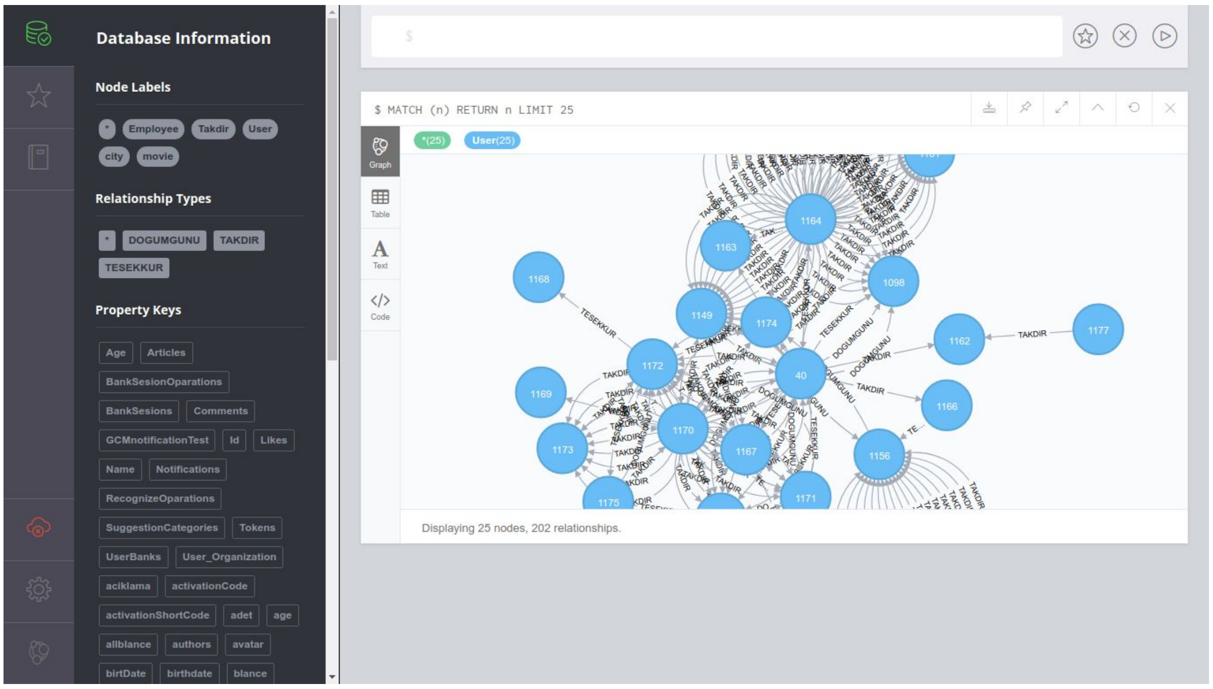
In this project, we decided to generate a report for Human Resources department, could be used for enhancing performance evaluation processes' efficiency; and observing misleading effects of factionalism between departments, and emotional closeness between individual employees.

What is our dataset?

The given dataset for this project (in Neo4J graph database format), there are structures represent employees (represented as nodes with some attributes like uID, departmanID, managerID, ustdepartmanID, etc.), and three relations:

- **TAKDIR** (appreciation),
- **TESEKKUR** (thank) and
- **DOGUMGUNU** (birthday)

could be seen below:



```
$ MATCH (n) RETURN n LIMIT 25
```

Displaying 25 nodes, 202 relationships.

Which metrics we used?

For this purpose, we used TAKDIR relation as a endorsement indicator for performance evaluation, DOGUMGUNU relation as a indicator of emotional closeness, and departmanID and ustdepartmanID as a metric for evaluate TAKDIR relations to remove misleading effect of possible factionalism between departments. In addition to this, centrality metrics for TAKDIR relation (betweenness and closeness centralities) also used for evaluate TAKDIR relations.

Which additional softwares we used?

To calculate centrality metrics, we used “algo” package from “Efficient Graph Algorithms For Neo4J”. We also used a RDBMS (SQLite) for data manipulation and result generation, because this is easier for us, as software engineers used relational databases intensively in their professional lives.

What we have done?

- Firstly, we executed some Cypher queries in Neo4J, to calculate centrality metrics and to discriminate relevant and irrelevant data of this dataset, for our purpose, and filter data we will use.
- Later, we exported results of these queries to .csv files, then imported to SQLite as separate tables, and manipulated data and generated results from that manipulated data, in SQLite.
- Finally, we exported results to Excel, and created a chart for data visualization.

How we have done?

Step 1: To export nodes (employees) and their non-spare attributes, we executed a Cypher query as follows:

```
MATCH (n)
RETURN  n.uID AS uID,
n.centrality AS centrality,
n.ustdepartmanName AS ustdepartmanName,
n.name AS name,
n.departmanName AS departmanName,
n.isManager AS isManager,
n.departmanID AS departmanID,
n.managerID AS managerID,
n.ustdepartmanID AS ustdepartmanID,
n.isActive AS isActive
ORDER BY n.uID;
```

Step 2: To export relations we will use in analysis, we executed a Cypher queries as follows:

```
MATCH (n)-[r:TAKDIR]->(m)
RETURN  n.uID AS takdir_eden,
m.uID AS takdir_edilen;

MATCH (n)-[r:DOGUMGUNU]->(m)
RETURN  n.uID AS dogumgunu_kutlayan,
m.uID AS dogumgunu_kutlanan;
```

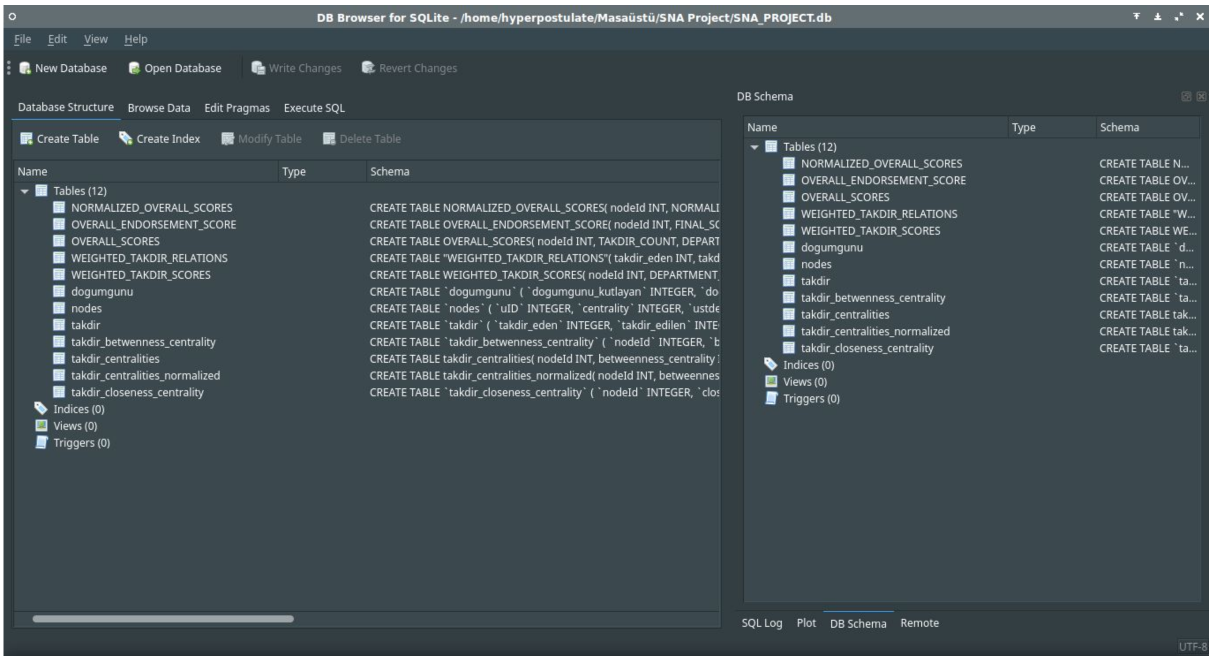
Note that, we didn't export TESEKKUR relation, because this relation is irrelevant for our analysis.

Step 3: To calculate and export centrality metrics, we called functions of "algo" package as follows:

```
CALL algo.betweenness.stream('*', 'TAKDIR', {direction:'in',concurrency:1})
YIELD nodeId, centrality
RETURN  nodeId,
centrality AS betweenness_centrality
ORDER BY  nodeId;
```

```
CALL algo.closeness.stream('*', 'TAKDIR', {direction:'in',concurrency:1})
YIELD nodeId, centrality
RETURN  nodeId,  centrality AS closeness_centrality
ORDER BY  nodeId;
```

Step 4: After these exports, we imported .csv files to SQLite as separate tables, could be seen below:



Name	Type	Schema
Tables (12)		
NORMALIZED_OVERALL_SCORES		CREATE TABLE NORMALIZED_OVERALL_SCORES(nodeId INT, NORMALI
OVERALL_ENDORSEMENT_SCORE		CREATE TABLE OVERALL_ENDORSEMENT_SCORE(nodeId INT, FINAL_SC
OVERALL_SCORES		CREATE TABLE OVERALL_SCORES(nodeId INT, TAKDIR_COUNT, DEPART
WEIGHTED_TAKDIR_RELATIONS		CREATE TABLE "WEIGHTED_TAKDIR_RELATIONS"(takdir_eden INT, takd
WEIGHTED_TAKDIR_SCORES		CREATE TABLE WEIGHTED_TAKDIR_SCORES(nodeId INT, DEPARTMENT,
dogumgunu		CREATE TABLE `dogumgunu` (`dogumgunu_kutlayan` INTEGER, `do
nodes		CREATE TABLE `nodes` (`uID` INTEGER, `centrality` INTEGER, `ustde
takdir		CREATE TABLE `takdir` (`takdir_eden` INTEGER, `takdir_edilen` INTE
takdir_betwenness_centrality		CREATE TABLE `takdir_betwenness_centrality` (`nodeId` INTEGER, `b
takdir_centralities		CREATE TABLE takdir_centralities(nodeId INT, betweenness_centrality
takdir_centralities_normalized		CREATE TABLE takdir_centralities_normalized(nodeId INT, betweennes
takdir_closeness_centrality		CREATE TABLE `takdir_closeness_centrality` (`nodeId` INTEGER, `clos
Indices (0)		
Views (0)		
Triggers (0)		

Name	Type	Schema
Tables (12)		
NORMALIZED_OVERALL_SCORES		CREATE TABLE N...
OVERALL_ENDORSEMENT_SCORE		CREATE TABLE OV...
OVERALL_SCORES		CREATE TABLE OV...
WEIGHTED_TAKDIR_RELATIONS		CREATE TABLE "W...
WEIGHTED_TAKDIR_SCORES		CREATE TABLE WE...
dogumgunu		CREATE TABLE `d...
nodes		CREATE TABLE `n...
takdir		CREATE TABLE `ta...
takdir_betwenness_centrality		CREATE TABLE `ta...
takdir_centralities		CREATE TABLE tak...
takdir_centralities_normalized		CREATE TABLE tak...
takdir_closeness_centrality		CREATE TABLE `ta...
Indices (0)		
Views (0)		
Triggers (0)		

Step 5: To aggregate centralities of nodes for TAKDIR relation, we executed a SQL query as follows:

```
CREATE TABLE takdir_centralities AS
SELECT tb.nodeId,
tb.betweenness_centrality,
tc.closeness_centrality
FROM  takdir_betwenness_centrality tb,
takdir_closeness_centrality tc
where tb.nodeId = tc.nodeId;
```

Step 6: To normalize centralities of nodes for TAKDIR relation, between 1 for maximum and 0 for minimum, we executed a SQL query as follows:

```
CREATE TABLE takdir_centralities_normalized AS
SELECT nodeId,
betweenness_centrality /(SELECT MAX(betweenness_centrality) FROM takdir_centralities ) AS
betweenness_centrality_normalized,
closeness_centrality /(SELECT MAX(closeness_centrality) FROM takdir_centralities ) AS
closeness_centrality_normalized
FROM  takdir_centralities;
```

Now, we have normalised centralities. We will use these later.

We must weight the relations (given endorsments) to observe effects of possible factionalism and emotional closeness.

To achieve this, firstly we decided to weight relations to eleminate effect of possible factionalism such that:

- If appreciative and appreciated employees are members of same department, evaluate this relation with **1**.
- If appreciative and appreciated employees are not members of same department, but members of same super-department, evaluate this relation with **2**.
- If appreciative and appreciated employees are not members of even same super-department, weight this relation with **3**.

Secondly, we decided to weight relations to eliminate effect of emotional closeness such that:

- If appreciative and appreciated employees both celebrated birthdays each other, evaluate this relation with **1**.
- If appreciative employee celebrated birthday of appreciated employee, but appreciated employee didn't celebrate birthday of appreciative employee, evaluate this relation with **2**.
- If appreciative and appreciated employees both didn't celebrated birthdays each other, weight this relation with **3**.

Step 7: To weight TAKDIR relations using department property of nodes and DOGUMGUNU relation between them, we executed a SQL query as follows:

```
CREATE TABLE WEIGHTED_TAKDIR_RELATIONS AS
SELECT wt.takdir_eden, wt.takdir_edilen,
wt.DEPARTMENT_WEIGHTED_TAKDIR_SCORE AS DEPARTMENT_WEIGHTED_TAKDIR_SCORE,
wt.DOGUMGUNU_WEIGHTED_TAKDIR_SCORE AS DOGUMGUNU_WEIGHTED_TAKDIR_SCORE,
(DEPARTMENT_WEIGHTED_TAKDIR_SCORE * DOGUMGUNU_WEIGHTED_TAKDIR_SCORE) AS
OVERALL_WEIGHTED_SCORE
FROM (SELECT t.takdir_eden,  t.takdir_edilen,
CASE WHEN ((SELECT ustdepartmanID FROM nodes where uID = t.takdir_eden) == (SELECT ustdepartmanID
FROM nodes where uID = t.takdir_edilen) AND (SELECT departmanID FROM nodes where uID =
t.takdir_eden) == (SELECT departmanID FROM nodes where uID = t.takdir_edilen)) THEN 1
WHEN ((SELECT ustdepartmanID FROM nodes where uID = t.takdir_eden) == (SELECT ustdepartmanID FROM
nodes where uID = t.takdir_edilen) AND (SELECT departmanID FROM nodes where uID = t.takdir_eden) <>
(SELECT departmanID FROM nodes where uID = t.takdir_edilen)) THEN 2 ELSE 3 END
DEPARTMENT_WEIGHTED_TAKDIR_SCORE,
CASE WHEN ((t.takdir_edilen in (SELECT d.dogumgunu_kutlayan FROM dogumgunu d where
d.dogumgunu_kutlanan = t.takdir_eden)) AND (t.takdir_eden in (SELECT d.dogumgunu_kutlayan FROM
dogumgunu d where d.dogumgunu_kutlanan = t.takdir_edilen))) THEN 1
WHEN ((t.takdir_edilen in (SELECT d.dogumgunu_kutlayan FROM dogumgunu d where
d.dogumgunu_kutlanan = t.takdir_eden)) AND (t.takdir_eden not in (SELECT d.dogumgunu_kutlayan FROM
dogumgunu d where d.dogumgunu_kutlanan = t.takdir_edilen)))
THEN 2 ELSE 3 END DOGUMGUNU_WEIGHTED_TAKDIR_SCORE
FROM takdir t) wt;
```

Step 8: To calculate weighted takdir scores for each employee, we executed a SQL query as follows:

```
CREATE TABLE WEIGHTED_TAKDIR_SCORES AS
SELECT  takdir_edilen AS nodeId,
sum(DEPARTMENT_WEIGHTED_TAKDIR_SCORE) AS DEPARTMENT_WEIGHTED_SCORE,
sum(DOGUMGUNU_WEIGHTED_TAKDIR_SCORE) AS DOGUMGUNU_WEIGHTED_SCORE,
sum(overall_weighted_score) AS OVERALL_WEIGHTED_SCORE
FROM  WEIGHTED_TAKDIR_RELATIONS
GROUP BY  takdir_edilen
ORDER BY  OVERALL_WEIGHTED_SCORE desc;
```

Step 9: To calculate overall endorsement score from weighted takdir scores, with centrality metrics calculated in Neo4J, we executed a SQL query as follows:

```
CREATE TABLE OVERALL_ENDORSEMENT_SCORE AS
SELECT s.uID AS nodeId,
(s.OVERALL_SCORE * (ifnull(cent.betweenness_centrality_normalized,0) + 1) *
(ifnull(cent.closeness_centrality_normalized,0) + 1)) AS FINAL_SCORE
FROM  (SELECT n.uID, ifnull(te.OVERALL_SCORE, 0) OVERALL_SCORE FROM nodes n
LEFT JOIN (SELECT nodeId, OVERALL_WEIGHTED_SCORE AS OVERALL_SCORE FROM
WEIGHTED_TAKDIR_SCORES) te
ON n.uID = te.nodeId ) s  LEFT JOIN  takdir_centralities_normalized cent
on s.uID = cent.nodeId;
```

Step 10: To aggregate all scores these we calculated, we executed **a** SQL query as follows:

```
CREATE TABLE OVERALL_SCORES AS
SELECT A.nodeId AS nodeId,
a.TAKDIR_COUNT AS TAKDIR_COUNT,
b.DEPARTMENT_WEIGHTED_SCORE,
b.DOGUMGUNU_WEIGHTED_SCORE,
b.OVERALL_WEIGHTED_SCORE,
c.FINAL_SCORE AS FINAL_SCORE_WITH_CENTRALITY
FROM (SELECT n.uID AS nodeId, ifnull(tc.TAKDIR_COUNT, 0) AS TAKDIR_COUNT FROM nodes n left join
(SELECT t.takdir_edilen AS nodeId, count(t.takdir_eden) AS TAKDIR_COUNT FROM takdir t GROUP BY
nodeId) tc on n.uID = tc.NodeId) A,
(SELECT * FROM WEIGHTED_TAKDIR_SCORES) B,
(SELECT * FROM OVERALL_ENDORSEMENT_SCORE) C
where  a.nodeId = b.nodeId and b.nodeId = c.nodeId;
```

Step 11: To normalize all scores for comparison, between 100 for maximum score and 0 for minimum score, we executed a SQL query as follows:

```
CREATE TABLE NORMALIZED_OVERALL_SCORES AS
SELECT nodeId, ((TAKDIR_COUNT * 100) / (SELECT MAX(TAKDIR_COUNT) FROM overall_scores)) AS
NORMALISED_DEGREE_SCORE,
((DEPARTMENT_WEIGHTED_SCORE * 100) / (SELECT MAX(DEPARTMENT_WEIGHTED_SCORE) FROM
overall_scores))  AS NORMALISED_DEPARTMENT_WEIGHTED_SCORE,
((DOGUMGUNU_WEIGHTED_SCORE * 100) / (SELECT MAX(DOGUMGUNU_WEIGHTED_SCORE) FROM
overall_scores))  AS NORMALISED_DOGUMGUNU_WEIGHTED_SCORE,
((OVERALL_WEIGHTED_SCORE * 100) / (SELECT MAX(OVERALL_WEIGHTED_SCORE) FROM overall_scores))
AS NORMALISED_OVERALL_WEIGHTED_SCORE,
((FINAL_SCORE_WITH_CENTRALITY * 100) / (SELECT MAX(FINAL_SCORE_WITH_CENTRALITY) FROM
overall_scores))  AS NORMALISED_FINAL_SCORE_WITH_CENTRALITY FROM  overall_scores;
```

Step 12: Finally, we exported NORMALIZED_OVERALL_SCORES table to Excel, to represent and visualise results with a chart.

Results and chart could be seen in Appendix, which is assumed as the report that will be used by human resources department.

Conclusion

With this project, we **generated a report** with performance evaluation metrics (scores) of employees,

- using raw endorsement data,
- after elimination effects of possible factionalism,
- after elimination effects of emotional closeness,
- after elimination both of them, and finally
- after elimination both of them and reclamation with centrality metrics.

Difference between score with raw endorsement data and others shows us each negative effect.

We also **generated a chart** to visualize this report, to compare employees and their different scores easily.

We believe this **weighted scoring approach** for performance evaluation is a far better approach than scoring with only raw endorsement, for companies' Human Resources departments.