INITIALIZATION

In this project i used the graph db g,ven by our professor and followed the instructions to get connected to corresponding graph db.

I use neo4j community edition.

I copied default.graphdb file which is given by professor into Document\Neo4j folder.

Then i start neo4j and connected it via <http://localhost:7474/browser/>.

OBSERVE EXISTING DB

I investigated which edges and nodes exists first.

Here i observed 2 type of nodes which are Employee and Takdir nodes.

And two main relation between these two nodes which are TAKDIR\_ALDI and TAKDIR\_ETTI.

I realized that there is no direct relation between two Employees.

COLLABORATIVE FILTERING

When i look to graph i want to see who appreciated to whom in another words which two employee have a relation such that they both have an edge to same Takdir object one of these is TAKDIR\_ETTI other is TAKDIR\_ALDI.

In order to see this i applied collaborative filtering by using neo4j cypher queryies.

**MATCH**

**(e:Employee)-[:TAKDIR\_ALDI]->(t:Takdir)<-[:TAKDIR\_ETTI]-(e2:Employee)**

**return e.name,e2.name limit 10**

|  |
| --- |
| Takdir Edildi Takdir Etti |
| CE1241 Ür40 |
| HÜ1179 Ür40 |
| İL1270 Ür40 |
| ŞE1170 Ür40 |
| MU1174 Ür40 |
| CA1172 Ür40 |
| CA1172 Ür40 |
| Hİ1225 Ür40 |
| YO1149 Ür40 |
| YO1149 AL1164 |

To decrease list of the size i put the limit 10.

Moreover this collaborative filtering point of view removes the Takdir object with degree one. This means if takdir comes from someone it goes another person and if a takdir received by someone it must come from another person.

FIND WHO IS ADMIRED MOST

To find who received TAKDIR\_ALDI relation from a Takdir object i wroted to cypher query below:

**MATCH**

**(e:Employee) - [:TAKDIR\_ALDI] -> (Takdir)**

**return e.name,count(Takdir) as degree**

**order by degree desc**

Result of this query actually gives a degree centrality only difference it considers TAKDIR\_ALDI relationships.



Result may show us who is most successful and takes admirations from other Emloyees.

WHO ADMIRE MOST

In society people who admire are generally people that experienced and very successful in his or her topic. So, this info can be usefull too.

I wrote query below for this :

**MATCH**

**(e:Employee) - [:TAKDIR\_ALDI] -> (Takdir)**

**return e.name,count(Takdir) as degree**

**order by degree desc**



NEW RELATION 🡪 T\_ROLE

In the beginning like i say i wanted to see who admired whom and i wonder that why these Employee nodes donot have relationship except for holding data of Takdir objects content.

So that, i decided to create new relationship,new type of edge, between Employee nodes. It is named T\_ROLE. This T\_ROLE means an employee admired another employee. I created this edge type by writing query below:

**MATCH**

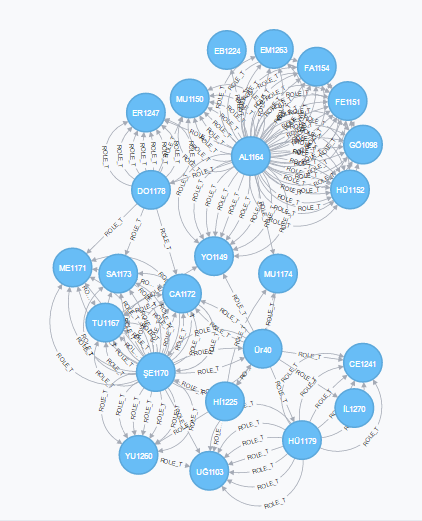
**(e:Employee)-[:TAKDIR\_ALDI]->(Takdir)<-[:TAKDIR\_ETTI]-(e2:Employee)**

**create (e2)-[r:ROLE\_T]->(e) RETURN r**

And i run the query below to observe relation between Employees in this view i dont see any Takdir nodes. Which clearifies view of the observer and see relation of Employees much easier.

**MATCH (e:Employee)-[:ROLE\_T]->(e2:Employee)return \* limit 100**

And result of this query is this:



As it is seen above clusters,bridges and hubs are much clearly visible in comparison to initial situation of graph db.

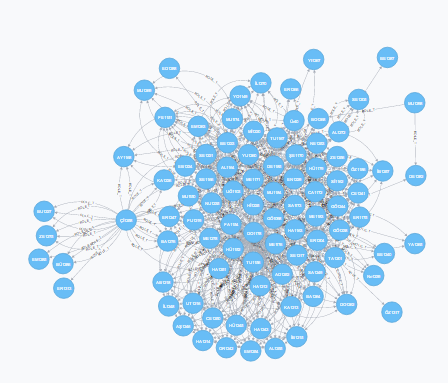
FIND ALL SHORTEST PATHS BY GIVING MIN AND MAX PATH LENGTH

In next steps i will calculate betweenness centrality which considers how important a node in shortest paths. So i consider that it would be good to calculate all possible shortest paths. I wrote the cypher query below:

**MATCH p=allShortestPaths((e:Employee)-[:ROLE\_T\*0..3]->(e2:Employee))**

**return p**

As a result of a this query i have all shortest paths of all of Employee nodes with the length starting from 0 to 3 by considering only ROLE\_T edges.



All paths i mentioned is shown above.

CHECK WHETHER AN EMPLOYEE ADMIRED BY WHO IS ADMIRED BY HIMSELF OR HERSELF

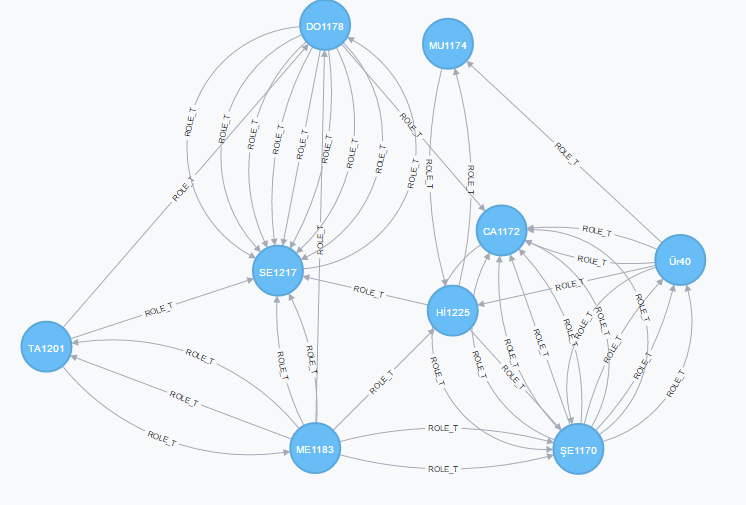
There are some cases such as it can be fake admirations or it can be really good coworkers. To check this i wrote the query below:

**MATCH (e:Employee)-[:ROLE\_T]->(e2:Employee)**

**where (e2)-[:ROLE\_T]->(e)**

**return \***

And i found that yeah there are some nodes like i consider that it would be.



After observing these nodes to understand why they admire each other i need deeper observations such as talking with them etc.

Till now , i created a new relationship type ROLE\_T and by using this new relationship i am able to see and whole graph easier.

I wondered some cases and calculated these cases.

Now it is time to use some well-known metrics.

I take <https://neo4j.com/blog/using-neo4j-hr-analytics/> website as reference.

I adopted algorithms explained in this webpage according to my node and edge type.

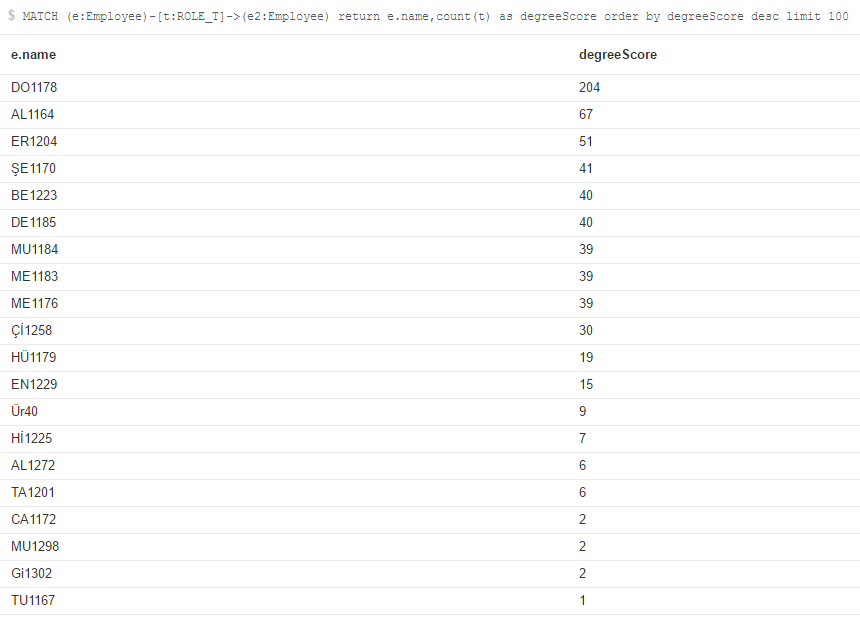
CALCULATE DEGREE CENTRALITY OF NODES

I calculated degree centrality with cyper query given below:

MATCH (e:Employee)-[t:ROLE\_T]->(e2:Employee) return e.name,count(t) as degreeScore order by degreeScore desc limit 100

Actually it is seen that after putting direct relationship between Employee nodes these calculations are simpler.

Result of this query is :



Here above DO1178 has highest centrality in terms of degree. I can say that this Employee has important role. In my new relationship i combined both TAKDIR\_ETTI and TAKDIR\_ALDI edges. So this is more general result than i calculated at the beginning.

CALCULATING BETWEENNESS CENTRALITY

I used reference website to calculate this centrality and runned the query below:

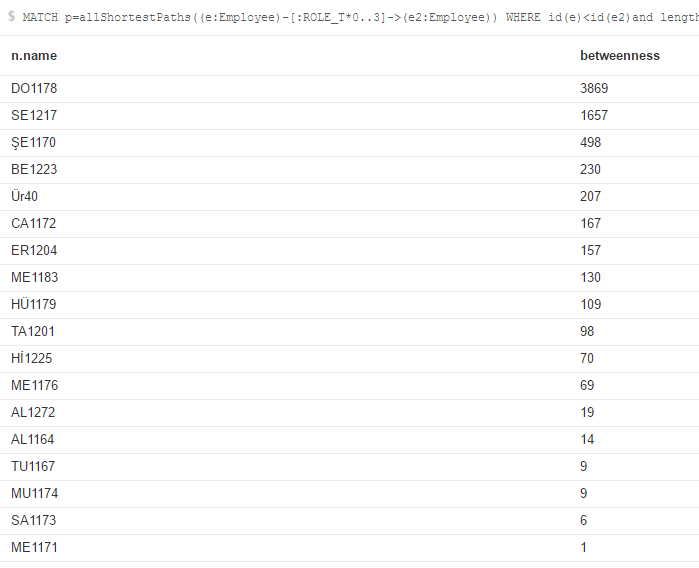
**MATCH p=allShortestPaths((e:Employee)-[:ROLE\_T\*0..3]->(e2:Employee)) WHERE id(e)<id(e2)and length(p)>1**

**unwind nodes(p)[1..-1]as n**

**return n.name,count(\*) as betweenness**

**order by betweenness desc**

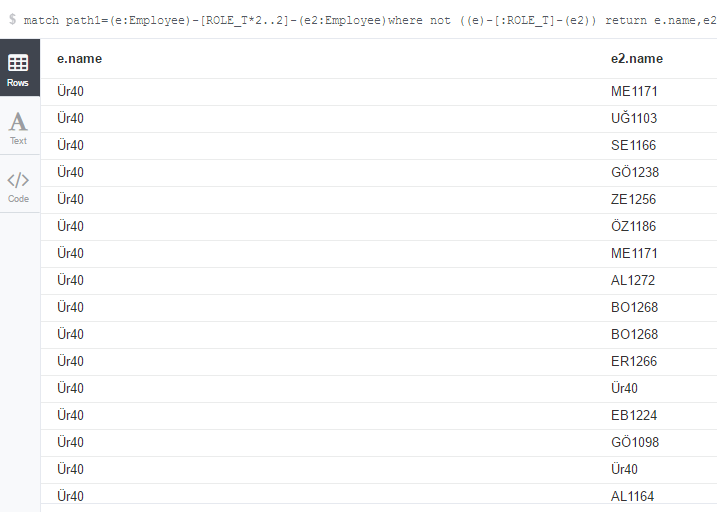
As a result of this query again most central node is DO1178. This means double check actually. Here is result:



FIND MISSING TRIADIC CLOSURES

**match path1=(e:Employee)-[ROLE\_T\*2..2]-(e2:Employee)where not ((e)-[:ROLE\_T]-(e2)) return e.name,e2.name limit 50**

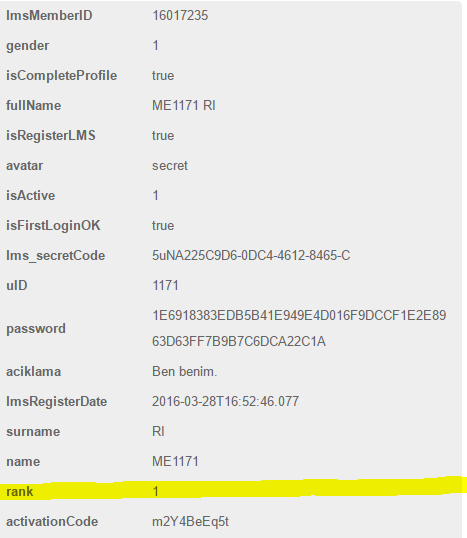
This give me pairs that recommended not directly, recommendation of recommended people exists. So that e recommended e2 indirectly some e3 is recommended by e and this e3 recommends e2 meaning that e supports e2 indirectly.



ADD RANK ATTRIBUTE TO NODES

To calculate pagerank for nodes i need to store that value in nodes. I created this attribute and started with initial value zero.

**MATCH (e:Employee) set e+= {rank: 0}**



CALCULATE PAGERANK OF NODES

Now i have rank attribute in each node. By using that probability i will implement the code given by the reference website.

**UNWIND range(1,2) AS ROUND**

**MATCH (e:Employee)**

**where rand()<0.1**

**MATCH (e:Employee)-[:ROLE\_T\*..10]->(e2:Employee)**

**set e2.rank=coalesce(e2.rank,0)+1;**

After this statement ranks are calculated.



Observe calculated ranks :

**match(e:Employee) return e.name,e.rank order by e.rank desc**

