**CODE NEO4j GROUP 9**

**DFS vs BFS**

CREATE

(nA:Node {name: 'Giám đốc'}),

(nB:Node {name: 'Trưởng phòng 1'}),

(nC:Node {name: 'Trưởng phòng 2'}),

(nD:Node {name: 'Nhân viên 1'}),

(nE:Node {name: 'Nhân viên 2'}),

(nA)-[:REL]->(nB),

(nA)-[:REL]->(nC),

(nB)-[:REL]->(nE),

(nC)-[:REL]->(nD);

MATCH (n) RETURN n LIMIT 25

CALL gds.graph.create('myGraph', 'Node', 'REL')

MATCH (source:Node{name:'Giám đốc'})

CALL gds.alpha.bfs.stream('myGraph', {

startNode: id(source)

})

YIELD path

RETURN path

MATCH (source:Node{name:'Giám đốc'})

CALL gds.alpha.dfs.stream('myGraph', {

startNode: id(source)

})

YIELD path

RETURN path

**BÀI TOÁN PAGERANK**

CREATE CONSTRAINT FOR (p:Player) REQUIRE p.id IS UNIQUE;

CREATE CONSTRAINT FOR (m:Match) REQUIRE m.id IS UNIQUE;

WITH

'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/match\_scores\_2010-2019.csv' AS uri

LOAD CSV WITH HEADERS FROM uri AS row

MERGE (m:Match {id: row.match\_id,

name: row.tourney\_round\_name,

winner: row.winner\_player\_id,

loser: row.loser\_player\_id,

tour\_id: row.tourney\_year\_id,

tour\_name: row.tourney\_name,

year: toInteger(split(row.tourney\_year\_id, "-")[0])});

WITH

'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/match\_scores\_2010-2019.csv' AS uri

LOAD CSV WITH HEADERS FROM uri AS row

MERGE (winner:Player{name: row.winner\_name,

id: row.winner\_player\_id})

MERGE (loser:Player{name: row.loser\_name,

id: row.loser\_player\_id});

MATCH (m:Match), (p:Player)

WHERE m.winner = p.id

MERGE (m)-[w:WINNER]->(p);

MATCH (m:Match), (p:Player)

WHERE m.loser = p.id

MERGE (m)-[:LOSER]->(p);

MATCH p=()-[r:LOSER]->() RETURN p LIMIT 25

MATCH p=()-[r:WINNER]->() RETURN p LIMIT 200

MATCH a=(m:Match)-[:LOSER]->(p:Player)

WHERE p.name="Novak Djokovic"

RETURN a

LIMIT 10

MATCH a=(m:Match)-[]->(p:Player)

WHERE p.name="Dominic Thiem"

AND m.year=2018

RETURN a

MATCH a=(p1)<-[]-()-[]->(p2)

WHERE p1.name="Novak Djokovic" AND p2.name = "Roger Federer"

RETURN a

LIMIT 5

MATCH (p:Player)

WITH p,

size((p)<-[:WINNER]-()) AS wins,

size((p)<-[:LOSER]-()) as defeats

RETURN p.name, wins, defeats,

CASE WHEN wins+defeats = 0 THEN 0

ELSE (wins \* 100.0) / (wins + defeats) END

AS percentageWins

ORDER BY wins DESC

LIMIT 20

MATCH (p1:Player {name: "Roger Federer"}),

(p2:Player {name: "Rafael Nadal"})

RETURN p1.name, p2.name,

size((p1)<-[:WINNER]-()-[:LOSER]->(p2)) AS p1Wins,

size((p1)<-[:LOSER]-()-[:WINNER]->(p2)) AS p2Wins

MATCH (p1)<-[:WINNER]-(match)-[:LOSER]->(p2)

WHERE p1.name IN ["Roger Federer", "Rafael Nadal"]

AND p2.name IN ["Roger Federer", "Rafael Nadal"]

RETURN p2.name AS source, p1.name AS target, count(\*) as weight

LIMIT 10

MATCH (p1)<-[:WINNER]-(Match)-[:LOSER]->(p2)

RETURN p2.name AS source, p1.name AS target, count(\*) as weight;

CALL gds.graph.create.cypher(

'ATP-graph\_2010-2019',

'MATCH (n) RETURN id(n) AS id',

'MATCH (p1)<-[:WINNER]-(Match)-[:LOSER]->(p2) RETURN id(p2) AS source,

id(p1) AS target, count(\*) as weight'

);

CALL gds.pageRank.stream('ATP-graph\_2010-2019', {

maxIterations: 20,

dampingFactor: 0.85,

relationshipWeightProperty: 'weight'

})

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).name AS name, score

ORDER BY score DESC, name ASC;

CALL gds.graph.create.cypher(

'ATP-graph-2018',

'MATCH (n) RETURN id(n) AS id',

'MATCH (p1)<-[:WINNER]-(Match)-[:LOSER]->(p2)

WHERE Match.year = 2018

RETURN id(p2) AS source, id(p1) AS target, count(\*) as weight'

);

CALL gds.pageRank.stream('ATP-graph-2018', {

maxIterations: 20,

dampingFactor: 0.85,

relationshipWeightProperty: 'weight'

})

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).name AS name, score

ORDER BY score DESC, name ASC;

**BÀI TOÁN SHORTEST PATH**

WITH 'https://raw.githubusercontent.com/tranhuonggiang22/essay/main/transport\_node.csv' AS uri

LOAD CSV WITH HEADERS FROM uri AS row

MERGE (p:Place {id:row.place, street:row.street})

MATCH (n:Place) RETURN n LIMIT 25

WITH ‘https://raw.githubusercontent.com/

leslien10/Essay-Dataset/main/transport\_node.csv’ AS url

LOAD CSV WITH HEADERS FROM url AS row

MERGE (p:Place {id:row.place, street:row.street})

WITH

'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/transport\_relationship.csv' AS url

LOAD CSV WITH HEADERS FROM url AS row

MATCH (source:Place {id: row.source})

MATCH (destination:Place {id: row.destination})

MERGE (source)-[:to {cost: toInteger(row.distance), street:row.street}]->(destination);

MATCH (n:Place) RETURN n LIMIT 25

CALL gds.graph.create(

'transport\_graph',

'Place',

{ to: {orientation: "NATURAL"}},

{

relationshipProperties: 'cost'

}

)

MATCH (source:Place {id: "Times City"}), (destination:Place {id: "Vincom Bà Triệu"})

CALL gds.shortestPath.dijkstra.stream(

'transport\_graph',

{

sourceNode: source,

targetNode: destination,

relationshipWeightProperty: 'cost'

})

YIELD nodeIds, costs

WITH [nodeId in nodeIds|gds.util.asNode(nodeId).id] AS nodeName,costs AS costs, size(nodeIds) AS siz

UNWIND range(0,siz-1) AS n

RETURN nodeName[n] AS place, costs[n] AS distance

CALL gds.graph.create(

'gr1',

'Place',

'to',

{

relationshipProperties: 'cost'

}

)

MATCH (source:Place {id: 'Times City'}), (target:Place {id: 'Vincom Bà Triệu'})

CALL gds.shortestPath.yens.stream('gr1', {

sourceNode: source,

targetNode: target,

k: 3,

relationshipWeightProperty: 'cost'

})

YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path

RETURN

index,

gds.util.asNode(sourceNode).id AS source,

gds.util.asNode(targetNode).id AS destination,

totalCost,

[nodeId IN nodeIds | gds.util.asNode(nodeId).id] AS nodeNames,

costs,

nodes(path) as path

ORDER BY index

MATCH (source:Place {id: 'Times City'}), (target:Place {id: 'Vincom Bà Triệu'})

CALL gds.shortestPath.yens.stream('gr1', {

sourceNode: source,

targetNode: target,

k: 3,

relationshipWeightProperty: 'cost'

})

YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path

WHERE index=0

RETURN index, path

ORDER BY index

**BÀI TOÁN KEGG\_DISEASE**

CREATE CONSTRAINT ON (n:disease) ASSERT n.ko IS UNIQUE;

CREATE CONSTRAINT ON (n:drug) ASSERT n.ko IS UNIQUE;

CREATE CONSTRAINT ON (n:pathogen) ASSERT n.ko IS UNIQUE;

LOAD CSV WITH HEADERS FROM ' https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/disease.csv' AS row

MERGE (n:disease {name: row.name, ko: row.ko, description: row.description, disease\_category:row.disease\_category});

LOAD CSV WITH HEADERS FROM ' 'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/drug.csv' AS row

MERGE (n:drug {name: row.name, ko: row.ko});

LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/pathogen.csv' AS row

MERGE (n:pathogen {name: row.name, ko: row.ko, taxonomy: row.taxonomy});

LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/drug\_disease.csv' AS row

MERGE (n1:drug {ko: row.from})

MERGE (n2:disease {ko: row.to})

MERGE (n1)-[r:treats]->(n2);

LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/leslien10/Essay-Dataset/main/pathogen\_disease.csv' AS row

MERGE (n1:pathogen {ko: row.from})

MERGE (n2:disease {ko: row.to})

MERGE (n1)-[r:causes]->(n2);

MATCH p=()-->() RETURN p LIMIT 25

MATCH p=()-[r:causes]->() RETURN p LIMIT 25

MATCH p=()-[r:treats]->() RETURN p LIMIT 25

MATCH (p:pathogen) RETURN COUNT(DISTINCT p);

MATCH (dr:drug) RETURN COUNT(DISTINCT dr);

MATCH (di:disease) RETURN COUNT(DISTINCT di);

MATCH (d:disease)

RETURN d.disease\_category, COUNT(d.disease\_category) as count

ORDER BY count DESC LIMIT 10;

MATCH (p:pathogen)

RETURN split(p.taxonomy, "; ")[0] as domain, COUNT(split(p.taxonomy, "; ")[0]) as count

ORDER BY count DESC;

MATCH (p:pathogen)

WHERE p.taxonomy contains "virus"

RETURN split(p.taxonomy, "; ")[0] + "; " + split(p.taxonomy, "; ")[1] as domain, COUNT(split(p.taxonomy, "; ")[0] + "; " + split(p.taxonomy, "; ")[1]) as count

ORDER BY count DESC;

MATCH (dr:drug) -[]->(di:disease)

WHERE di.disease\_category = "Infectious disease"

RETURN COUNT(DISTINCT(dr));

MATCH (dr:drug) -[r:treats]->(di:disease)

RETURN dr.name as drug, COUNT(r) as count

ORDER BY count DESC LIMIT 10;

MATCH (dr:drug)-[r:treats]->(di:disease)

WITH dr, COUNT(r) AS indicationCount ORDER BY indicationCount DESC LIMIT 10

MATCH p=(dr:drug)-[r:treats]->(di:disease)

RETURN p

MATCH (dr:drug) -[]->(di:disease)

WHERE dr.name="Prednisolone sodium phosphate"

RETURN di.disease\_category, count(di.disease\_category) as count

ORDER BY count DESC LIMIT 10;

MATCH (dr:drug) -[]->(di:disease)

WHERE dr.name="Prednisolone sodium phosphate" AND (di.disease\_category = "Lung disease" or di.disease\_category="Infectious disease")

RETURN di.name, di.disease\_category

ORDER BY di.disease\_category;

MATCH (p:pathogen) -[r:causes]->(di:disease)

RETURN p.name, COUNT(r) as count ORDER BY count DESC LIMIT 10;

MATCH a=(p:pathogen)-[r:causes]->(di:disease)

WHERE p.name="Human papillomavirus type 16"

or p.name="Human papillomavirus type 18"

RETURN a;

MATCH a=(p:pathogen)-[r:causes]->(di:disease)

WHERE di.name="COVID-19"

or di.name="Severe acute respiratory syndrome"

RETURN a;

MATCH a=(p:pathogen)-[r:causes]->(di:disease)<-[t:treats]-(dr:drug)

WHERE di.name="COVID-19"

RETURN a;

MATCH a=(p:pathogen)-[r:causes]->(di:disease)<-[t:treats]-(dr:drug)

WHERE di.name="HIV infection"

RETURN a;

CALL gds.graph.create.cypher(

'disease-graph',

'MATCH (n) RETURN id(n) AS id',

'MATCH (n)--(m) RETURN id(n) AS source, id(m) AS target'

);

CALL gds.louvain.stats('disease-graph')

YIELD communityCount

CALL gds.louvain.stream('disease-graph')

YIELD nodeId, communityId, intermediateCommunityIds

RETURN communityId, COUNT(communityId) as count

ORDER BY count DESC LIMIT 10;

CALL gds.louvain.stream('disease-graph')

YIELD nodeId, communityId, intermediateCommunityIds

WHERE communityId = 3421

RETURN gds.util.asNode(nodeId).name AS name, communityId

CALL gds.pageRank.stream('disease-graph')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).name AS name, score

ORDER BY score DESC, name ASC LIMIT 12;

CALL gds.pageRank.stream('disease-graph')

YIELD nodeId, score

WITH COLLECT(DISTINCT(gds.util.asNode(nodeId))) AS di\_list, score

ORDER BY score DESC

MATCH (dr:drug) -[r:treats]-> (di:disease)

WHERE di in di\_list

RETURN di.name as disease\_name, score AS pagerank\_score, COUNT(DISTINCT(r)) AS drug LIMIT 20;

MATCH (dr:drug) -[r:treats]->(di:disease)

RETURN di.ko, di.name, COUNT(r) as count ORDER BY count DESC LIMIT 10;

MATCH a=(dr:drug)-[t:treats]->(di:disease)

WHERE di.name="High blood pressure"

RETURN a;