UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática
PÓLO II - Pinhal de Marrocos
3030-290 Coimbra - Portugal
Tel. 239 790000 Fax. 239 701266

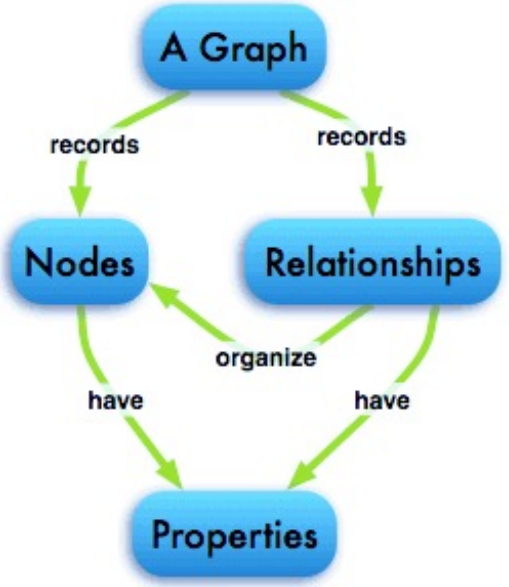
Ficha 3 – Neo4j

(Pedro Furtado)

Preliminary info

For reference only

Graph Databases



The property graph contains **connected entities (the *nodes*)**
can hold any number of attributes (key-value-pairs).

Nodes can be tagged with **labels representing their different roles**

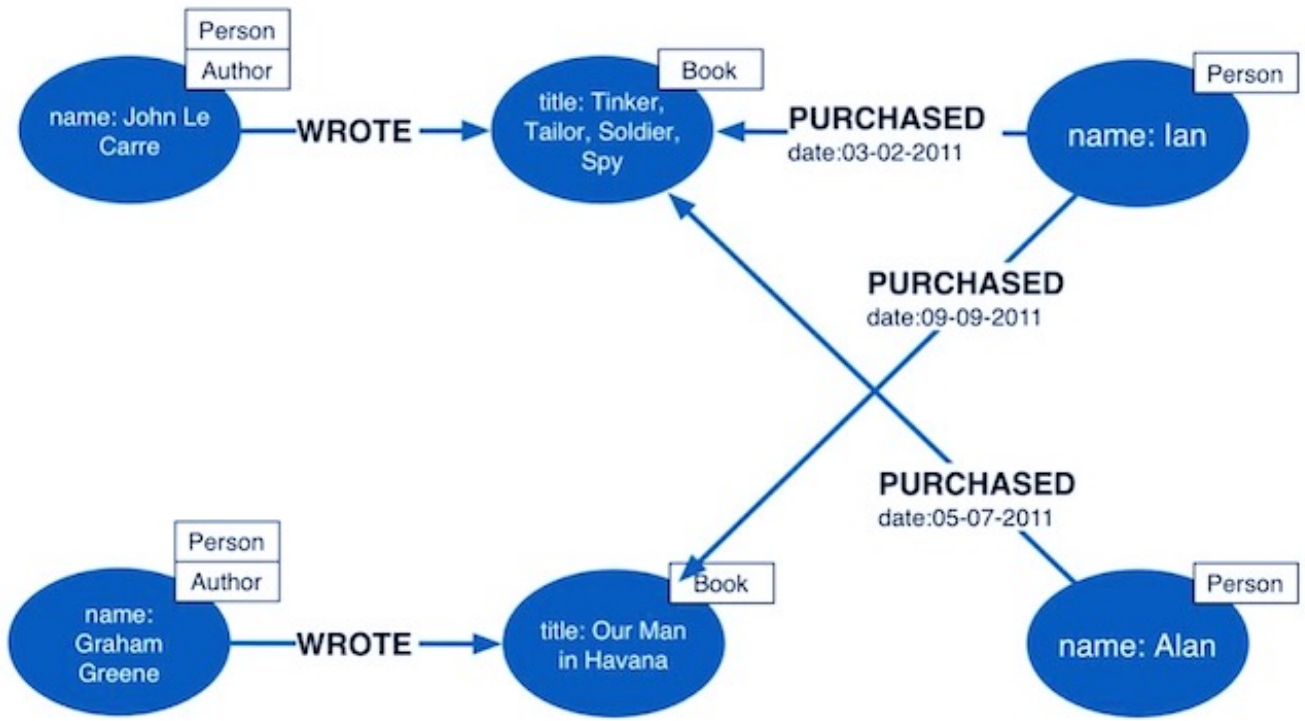
***Relationships* provide directed, named semantically relevant connecti**
node-entities.

A relationship always has a **direction, a type, a *start node*, and an *end node*.**

Like nodes, relationships **can have any properties**.
quantitative, such as weights, costs, distances, ratings, time intervals, or strengths.

Two nodes can share any number or type of relationships

Labeled Property Graph Data Model



Neo4j

Neo4j is an open-source NoSQL graph database implemented in Java and Scala.

The source code and issue tracking are available on GitHub, with support readily available on Stack Overflow and the Neo4j Google group.

Neo4j is used today by hundreds of thousands of companies and organizations in almost all industries.

Use cases include matchmaking, network management, software analytics, scientific research, routing, organizational and project management, recommendations, social networks, and more.

Neo4j implements the Property Graph Model, full database characteristics including ACID transaction compliance, cluster support, and runtime failover, making it suitable to use graph data in production scenarios.

Neo4j Server

You can download Neo4j from http://neo4j.com/download and install it as a server on all operating systems.

By default, the Neo4j Server is bundled with an interactive, web-based database interface bound to http://localhost:7474.

The simplest way of getting started is to use Neo4j's database browser to execute your graph queries (written in Cypher, our graph query language) in a workbench-like fashion.

Results are presented as either intuitive graph visualizations or as easy-to-read, exportable tables.

Cypher

Cypher is a declarative graph query language that allows for expressive and efficient querying and updating of the graph store.

Cypher is a relatively simple but still very powerful language.

Very complicated database queries can easily be expressed through Cypher.

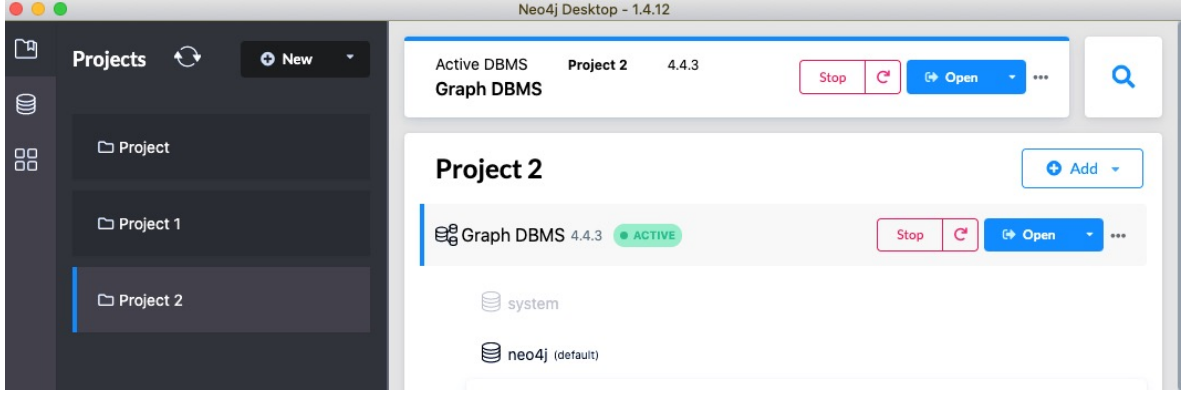
This allows you to focus on your domain instead of getting lost in database access.”

Instalação e corrida do neo4j

Para instalar, fazer download da aplicação.

Depois criar uma nova base de dados e corre-la, seguindo as instruções dos slides seguinte...

Create new Project



2. Create and start a Database
Click the “New Database” or “New graph” button.

This will turn into two blue buttons, one labeled “Local” and one labeled “Remote.” Click “Local.”

Click the blue button labeled “Create” that appears.

The “Create” button will be replaced by a “Start” button. Click it.

3. Open the Neo4j Browser
Once the database has started, click the “Manage” button.

Click on the “Open Browser” button that will appear in the Database Management area.

The browser will open in a new window, and prompt you to enter the password for the database, which is “neo4j.” You will be asked to change this password after you connnect, and then you enter the browser environment as seen in the screen below.

Lab work

Start here...

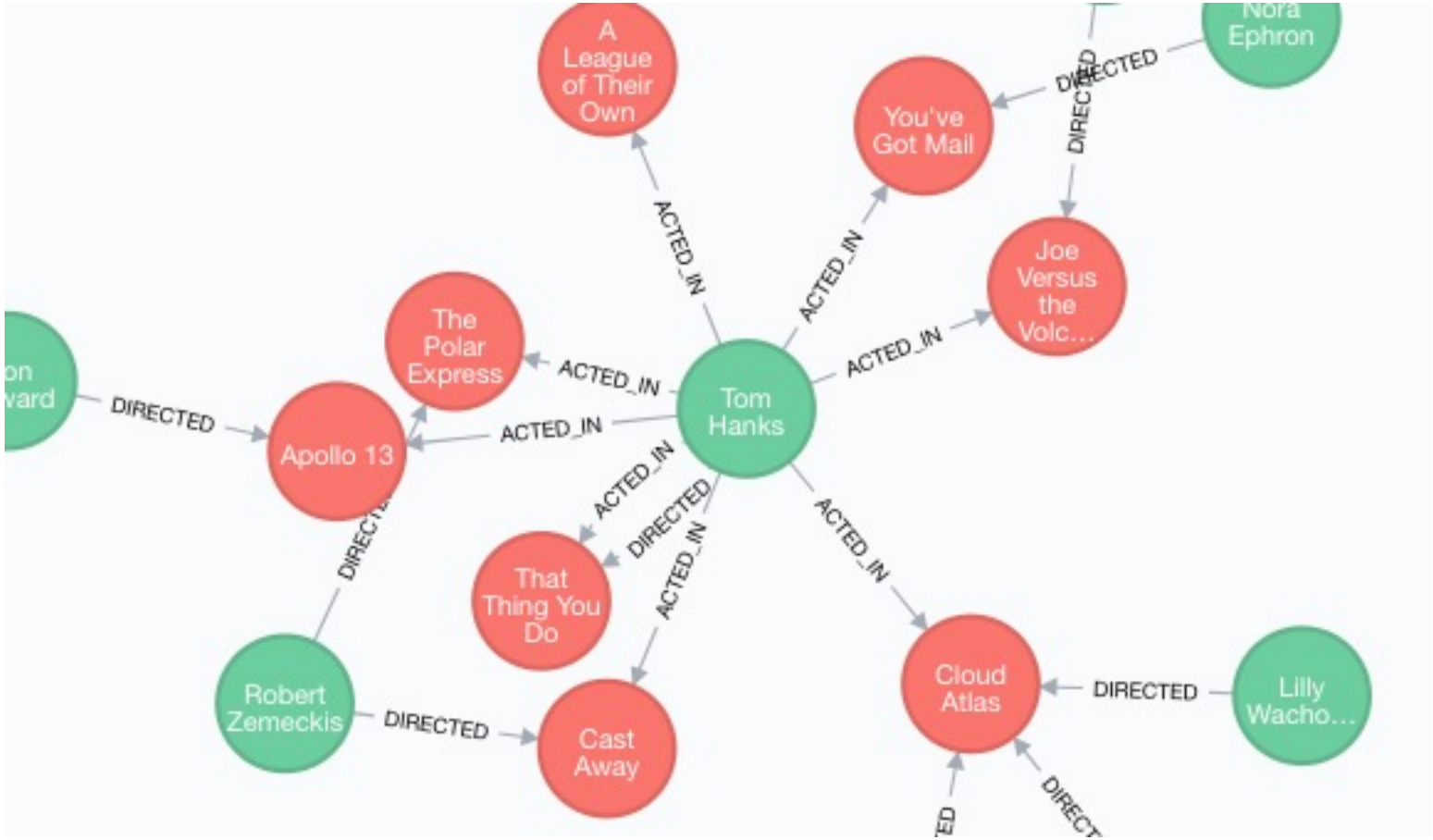
P0(a): o que cria o próximo código?

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})

CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})

CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})

CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})

CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})

CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})

CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})

CREATE (JoelS:Person {name:'Joel Silver', born:1952})
...
```

P0(b): o que cria o próximo código?

```
....
CREATE
        (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
        (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
        (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
        (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
        (LillyW)-[:DIRECTED]->(TheMatrix),
        (LanaW)-[:DIRECTED]->(TheMatrix),
        (JoelS)-[:PRODUCED]->(TheMatrix)
...
```

P3: Crie o grafo de filmes, abrindo o ficheiro de código cypher dado e correndo os comandos



Read Query Structure

```
[MATCH WHERE]
[OPTIONAL MATCH WHERE]
[WITH [ORDER BY] [SKIP] [LIMIT]]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

MATCH

```
MATCH (n:Person)-[:KNOWS]->(m:Person)
WHERE n.name = 'Alice'
```

Node patterns can contain labels and properties.

```
MATCH (n)-->(m)
```

Any pattern can be used in `MATCH`.

```
MATCH (n {name: 'Alice'})-->(m)
```

Patterns with node properties.

```
MATCH p = (n)-->(m)
```

Assign a path to `p`.

```
OPTIONAL MATCH (n)-[r]->(m)
```

Optional pattern: `nulls` will be used for missing parts.

WHERE

```
WHERE n.property <> $value
```

Use a predicate to filter. Note that `WHERE` is always part of a `MATCH`, `OPTIONAL MATCH`, `WITH` or `START` clause. Putting it after a different clause in a query will alter what it does.

RETURN

```
RETURN *
```
Return the value of all variables.

```
RETURN n AS columnName
```
Use alias for result column name.

```
RETURN DISTINCT n
```
Return unique rows.

```
ORDER BY n.property
```
Sort the result.

```
ORDER BY n.property DESC
```
Sort the result in descending order.

```
SKIP $skipNumber
```
Skip a number of results.

```
LIMIT $limitNumber
```
Limit the number of results.

```
SKIP $skipNumber LIMIT $limitNumber
```
Skip results at the top and limit the number of results.

```
RETURN count(*)
```
The number of matching rows. See Aggregating Functions for more.

Questions A: (Replace X and Y so it works)

P1: Find “Tom Hanks”

MATCH (X {name: ”Y"}) RETURN tom

a) Show the result as graph node

b) Show also the result as a table and as JSON

Questions A: (cont.)

P2: Find the movies titled “Cloud Atlas”

MATCH (X {title: "Cloud Atlas"}) RETURN Y

P3: Find 10 persons...

a) MATCH (X:Y) RETURN people.name LIMIT Z

b) Now modify to show graph nodes

P4: Find movies from the nineties

a) MATCH (X:Y) WHERE Z AND T RETURN nineties.title

b) Now modify to show graph nodes

P5: List all Tom Hanks movies

MATCH (tom:Z {name: ”X"})-[:Y]->(tomHanksMovies) RETURN tom,tomHanksMovies

P6: Who directed Cloud Atlas?

MATCH (cloudAtlas {title: ”X"})<-[:Y]-(directors) RETURN directors.name

P7: Tom Hanks' co-actors

MATCH (tom:Person {name:”X"})-[:Y]->(m)<-[:Z]-(coActors) RETURN coActors.name

P8: How people are related to "Cloud Atlas"?

MATCH (people:Person)-[relatedTo]-(:Movie {title: ”X"}) RETURN people.name,
Type(relatedTo), relatedTo

P9, P10: Run the following and tell us what each does?

```
MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
RETURN DISTINCT hollywood
```

```
MATCH p=shortestPath(  (bacon:Person {name:"Kevin Bacon"})-[*]-
(meg:Person {name:"Meg Ryan"}))
RETURN p
```

P11. Find all actors whose name starts by 'T’ or ‘Ro’.

a) **Return as graph nodes**
b) **Return name and birthdate (born)**

P12. Create the query to see who worked in movie "You've Got Mail”

P13. Show movies that include “Matrix” in the name

P14. Return a list of movies and their cast

Transport graph

Figure 4-2 shows the target graph that we want to construct.

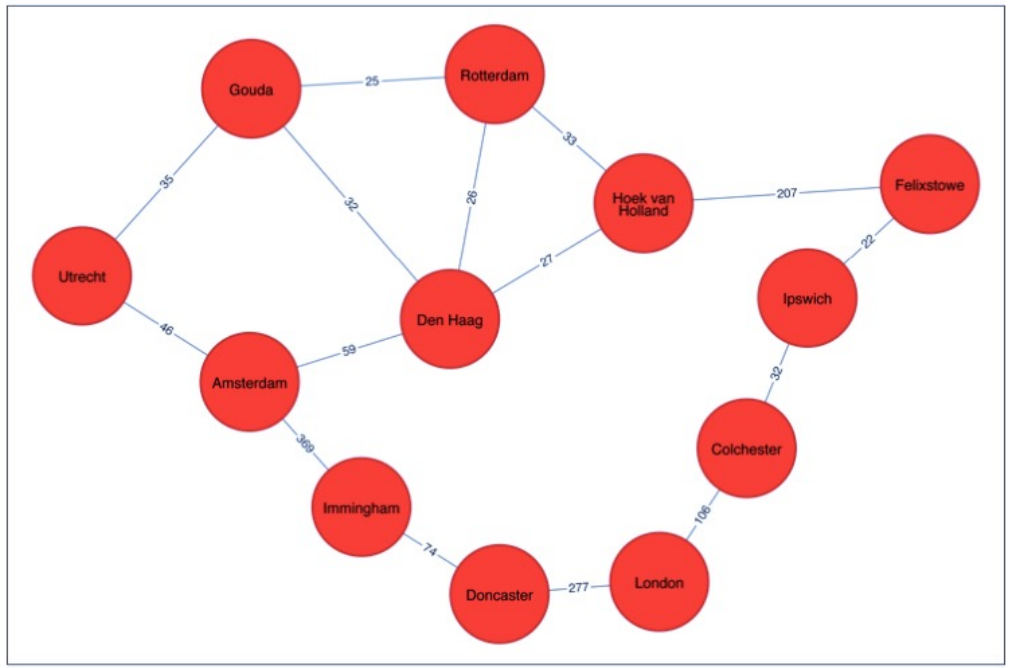


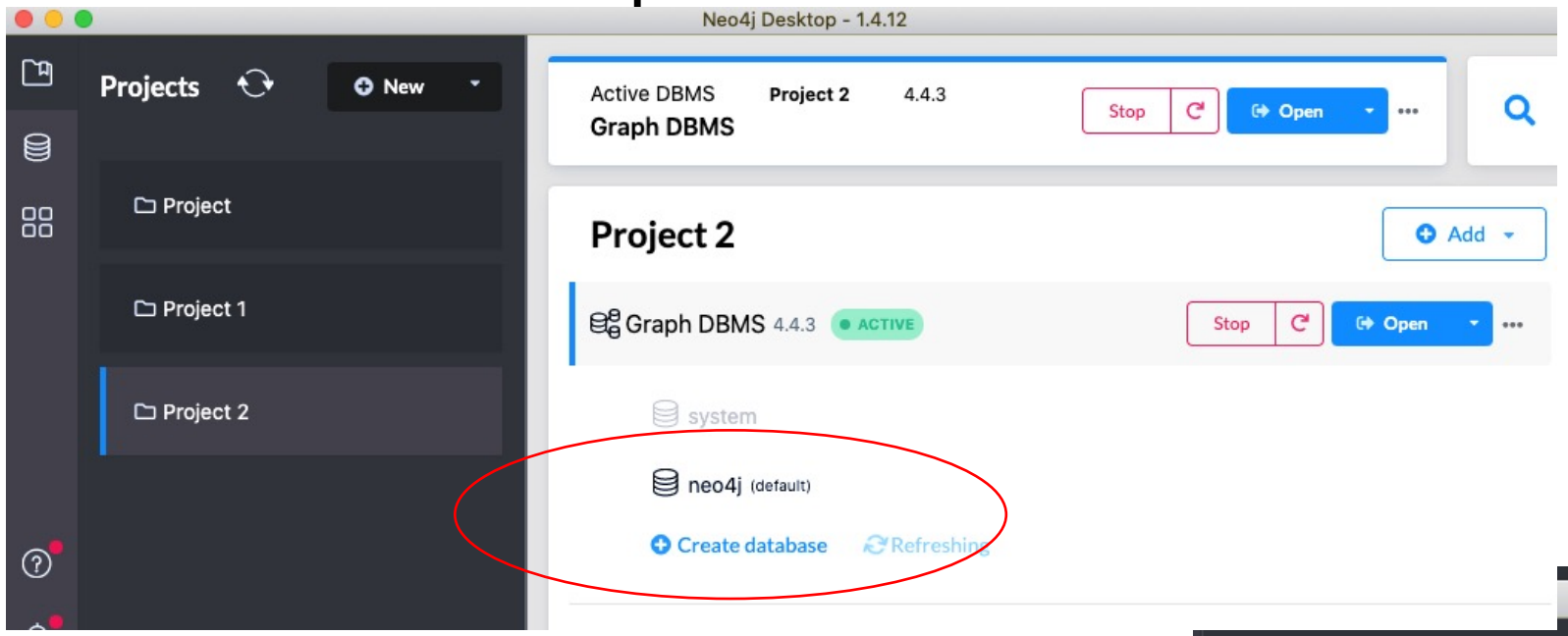
Table 4-2. transport-nodes.csv

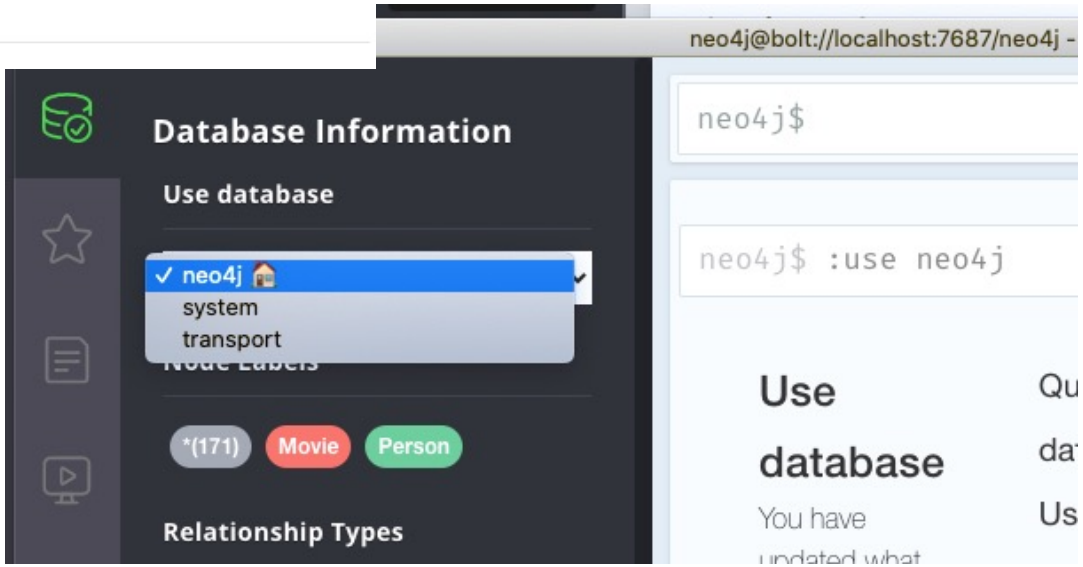
id	latitude	longitude	population
Amsterdam	52.379189	4.899431	821752
Utrecht	52.092876	5.104480	334176
Den Haag	52.078663	4.288788	514861
Immingham	53.61239	-0.22219	9642
Doncaster	53.52285	-1.13116	302400
Hoek van Holland	51.9775	4.13333	9382
Felixstowe	51.96375	1.3511	23689
Ipswich	52.05917	1.15545	133384
Colchester	51.88921	0.90421	104390
London	51.509865	-0.118092	8787892
Rotterdam	51.9225	4.47917	623652
Gouda	52.01667	4.70833	70939

Table 4-3. transport-relationships.csv

src	dst	relationship	cost
Amsterdam	Utrecht	EROAD	46
Amsterdam	Den Haag	EROAD	59
Den Haag	Rotterdam	EROAD	26
Amsterdam	Immingham	EROAD	369
Immingham	Doncaster	EROAD	74
Doncaster	London	EROAD	277
Hoek van Holland	Den Haag	EROAD	27
Felixstowe	Hoek van Holland	EROAD	207
Ipswich	Felixstowe	EROAD	22
Colchester	Ipswich	EROAD	32

Create transport database … Use it…





Or, on cypher:
:use transport

Now load nodes as relationships ...

- Try to load the nodes using url and the relationships using a file in order to learn how it is done in both cases...

id,latitude,longitude,population
"Amsterdam",52.379189,4.899431,821752
"Utrecht",52.092876,5.104480,334176
"Den Haag",52.078663,4.288788,514861
"Immingham",53.61239,-0.22219,9642
"Doncaster",53.52285,-1.13116,302400
"Hoek van Holland",51.9775,4.13333,9382
"Felixstowe",51.96375,1.3511,23689
"Ipswich",52.05917,1.15545,133384
"Colchester",51.88921,0.90421,104390
"London",51.509865,-0.118092,8787892
"Rotterdam",51.9225,4.47917,623652
"Gouda",52.01667,4.70833,70939

src,dst,relationship,cost
"Amsterdam","Utrecht","EROAD",46
"Amsterdam","Den Haag","EROAD",59
"Den Haag","Rotterdam","EROAD",26
"Amsterdam","Immingham","EROAD",369
"Immingham","Doncaster","EROAD",74
"Doncaster","London","EROAD",277
"Hoek van Holland","Den Haag","EROAD",27
"Felixstowe","Hoek van Holland","EROAD",207
"Ipswich","Felixstowe","EROAD",22
"Colchester","Ipswich","EROAD",32
"London","Colchester","EROAD",106
"Gouda","Rotterdam","EROAD",25
"Gouda","Utrecht","EROAD",35
"Den Haag","Gouda","EROAD",32
"Hoek van Holland","Rotterdam","EROAD",33

How to load from https

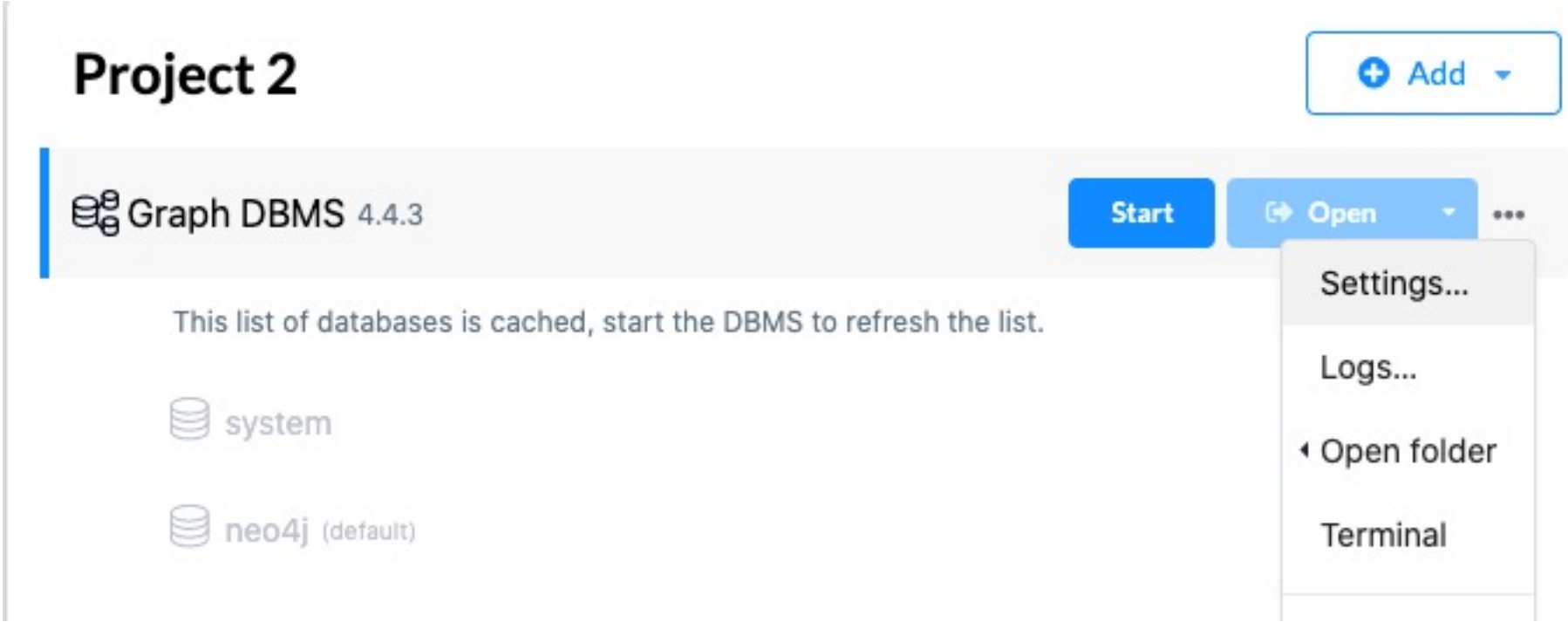
Load transport nodes from http

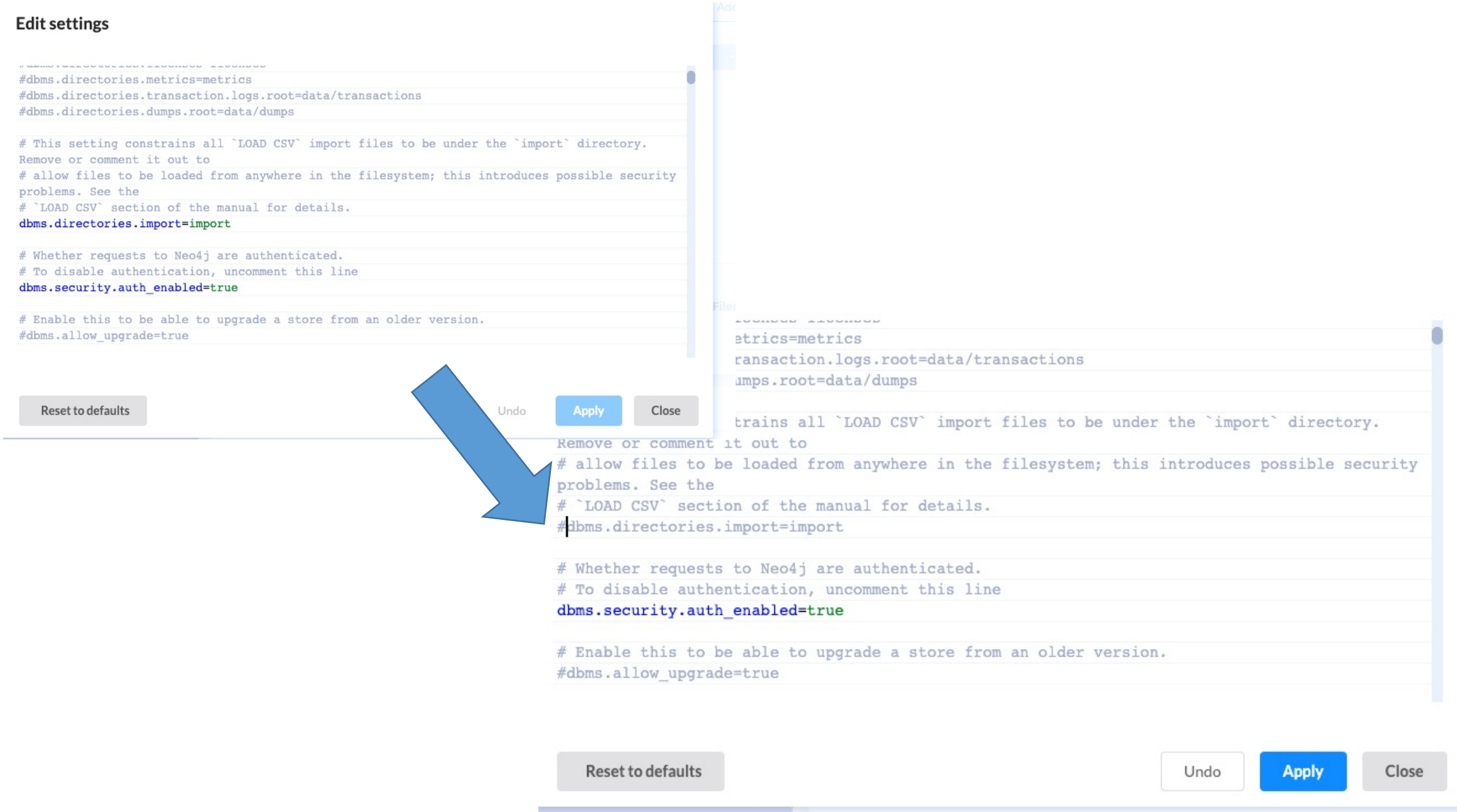
```
WITH 'https://github.com/neo4j-graph-analytics/book/raw/master/data/' AS base
WITH base + 'transport-nodes.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (place:Place {id:row.id})
SET place.latitude = toFloat(row.latitude),
        place.longitude = toFloat(row.longitude),
        place.population = toInteger(row.population);
```

How to load from file

(do not load from both https and file to avoid duplicates

allow import from anywhere (reset to previous after imports, for security)




Edit settings
#dbms.directories.metrics=metrics
#dbms.directories.transaction.logs.root=data/transactions
#dbms.directories.dumps.root=data/dumps
This setting constrains all `LOAD CSV` import files to be under the `import` directory.
Remove or comment it out to
allow files to be loaded from anywhere in the filesystem; this introduces possible security
problems. See the
# `LOAD CSV` section of the manual for details.
dbms.directories.import=import
# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
dbms.security.auth_enabled=true
# Enable this to be able to upgrade a store from an older version.
#dbms.allow_upgrade=true
Reset to defaults
Undo
Apply
Close

Load transport nodes from file

```
WITH
'file:///Users/pedro/Documents/Aulas/SGD/2022SGD/pratica/aula11_12_NOSQLgraphNeo4JApache/neo4Jgraphs/neo4Jgraphalgs/0636920233145-master/data/' AS base
AS base WITH base + 'transport-nodes.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (place:Place {id:row.id})
SET place.latitude = toFloat(row.latitude),
        place.longitude = toFloat(row.longitude),
        place.population = toInteger(row.population);
```

For copy-paste:

```
WITH
'file:///Users/pedro/Documents/Aulas/SGD/2022SGD/pratica/aula11_12_NOSQLgraphNeo4JApache/neo4Jgraphs/neo4Jgraphalgs/0636920233145-master/data/' AS base
WITH base + 'transport-nodes.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (place:Place {id:row.id})
SET place.latitude = toFloat(row.latitude),
        place.longitude = toFloat(row.longitude),
        place.population = toInteger(row.population);
```

Load transport relationships from http or file

```
WITH 'https://github.com/neo4j-graph-analytics/book/raw/master/data/'
AS base WITH base + 'transport-relationships.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MATCH (origin:Place {id: row.src})
MATCH (destination:Place {id: row.dst})
MERGE (origin)-[:EROAD {distance: toInteger(row.cost)}]->(destination);
```

```
WITH
'file:///Users/pedro/Documents/Aulas/SGD/2022SGD/pratica/aula11_12_NOSQLgraphNeo4JApache/neo4Jg
raphs/neo4Jgraphalgs/0636920233145-master/data/' AS base
WITH base + 'transport-relationships.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MATCH (origin:Place {id: row.src})
MATCH (destination:Place {id: row.dst})
MERGE (origin)-[:EROAD {distance: toInteger(row.cost)}]->(destination);
```

Show everything

P1. Why does the following show everything?

```
match (n)-[r]->(m) RETURN n,r,m
```

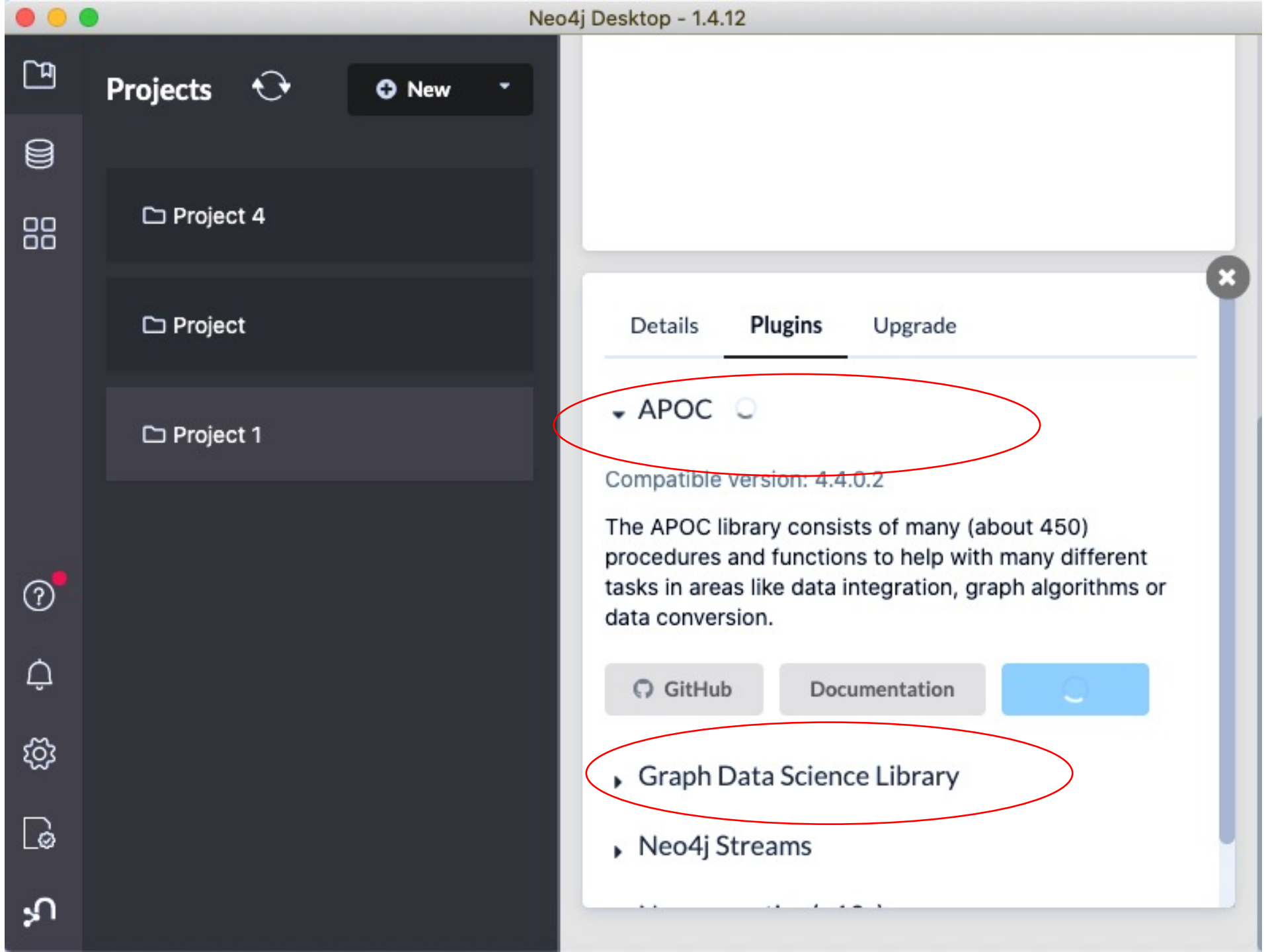
P2. Run it

P3. Bssed on the gaph you see, what is the distance between London and Ipswish?

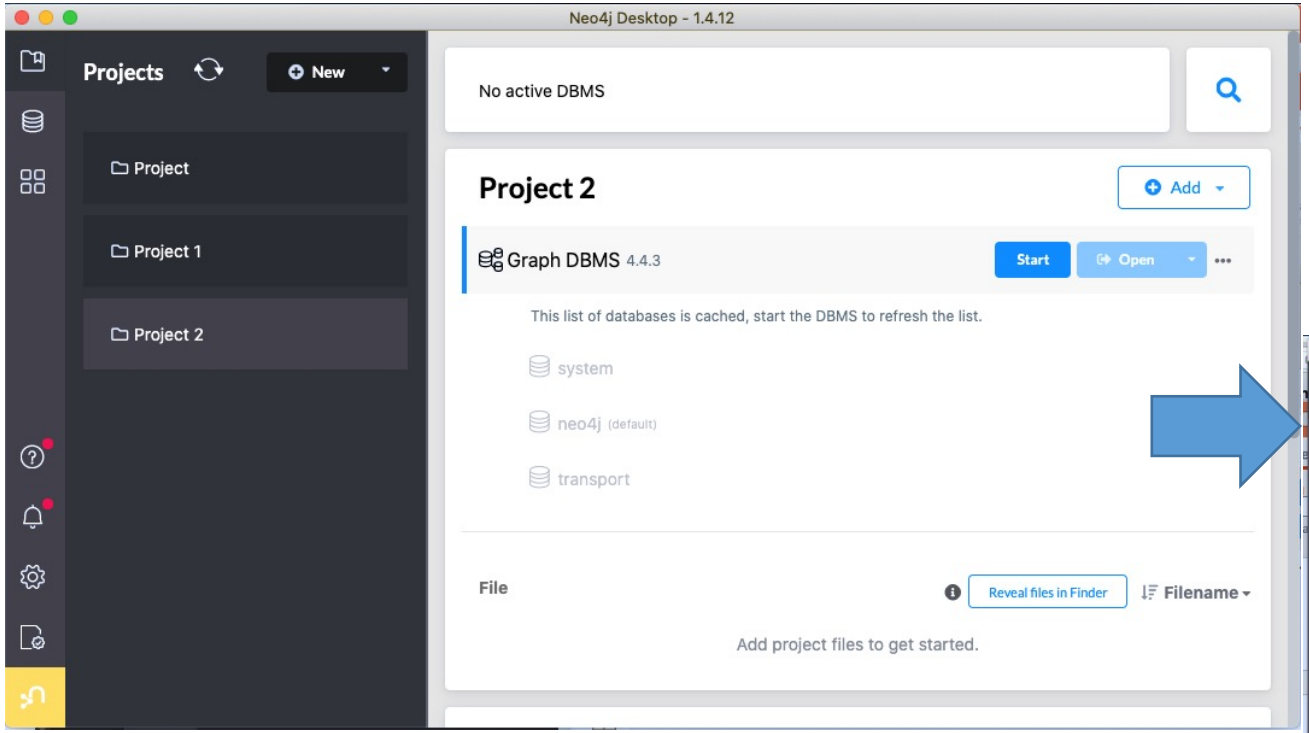
P4. Calculate shortest path from Amsterdam to London. How far is it?

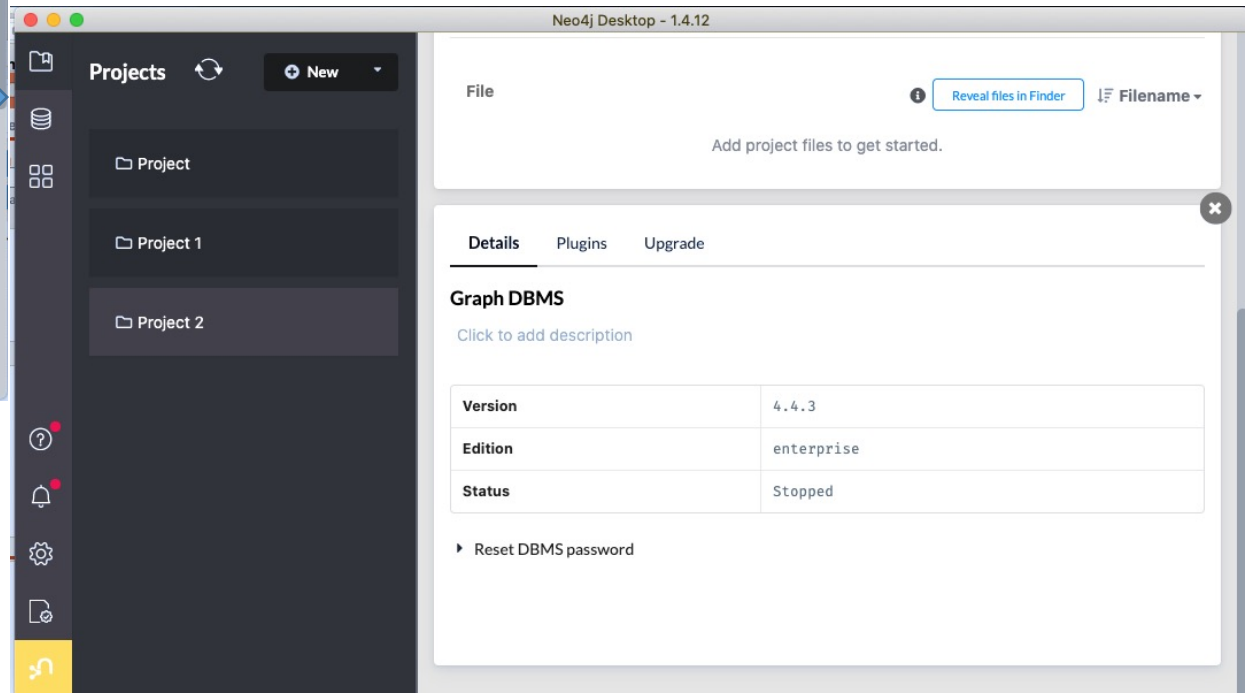
```
MATCH p=shortestPath(  (london:Place {id:"London"})-[*]-(amsterdam:Place {id:"Amsterdam"}))
RETURN p
```

Graph data science lib (gds)
+ APOC=support library
must install those plugins


Neo4j Desktop - 1.4.12
Projects
New
Project 4
Project
Project 1
Details
Plugins
Upgrade
APOC
Compatible version: 4.4.0.2
The APOC library consists of many (about 450) procedures and functions to help with many different tasks in areas like data integration, graph algorithms or data conversion.
GitHub
Documentation
Graph Data Science Library
Neo4j Streams

How to get to that plugins tab: scroll down after choosing database





P5. After running the following code, answer what is the unweighted shortest-path from London to Amsterdam using gds (graph data science lib)

```
CALL gds.alpha.allShortestPaths.stream({
nodeProjection: "*",
relationshipProjection: {
all: {
type: '*',
orientation:"UNDIRECTED"
}
}
})
YIELD sourceNodeId, targetNodeId, distance
WHERE gds.util.asNode(sourceNodeId).id = "London" and gds.util.asNode(targetNodeId).id = "Amsterdam"
RETURN gds.util.asNode(sourceNodeId).id as origin ,gds.util.asNode(targetNodeId).id as destination, distance
```

P6. Now we want you to calculate the Weighted shortest-path from Amsterdam to London using gds (graph data science lib). What is it? Is this the same value you obtained without gds?

```
CALL gds.alpha.allShortestPaths.stream({
nodeProjection: 'Place',
relationshipProjection: {
EROAD: {
type: 'EROAD',
properties: 'distance'
}
},
relationshipWeightProperty: 'distance'
})
YIELD sourceNodeId, targetNodeId, distance
WHERE gds.util.asNode(sourceNodeId).id = "Amsterdam" and gds.util.asNode(targetNodeId).id = "London"
RETURN gds.util.asNode(sourceNodeId).id as origin ,gds.util.asNode(targetNodeId).id as destination, distance
```

Same code...

```
CALL gds.alpha.allShortestPaths.stream({
nodeProjection: 'Place',
relationshipProjection: {
EROAD: {
type: 'EROAD',
properties: 'distance',
orientation:"UNDIRECTED"
}
},
relationshipWeightProperty: 'distance'
})
YIELD sourceNodeId, targetNodeId, distance
WHERE gds.util.asNode(sourceNodeId).id = "London" and gds.util.asNode(targetNodeId).id = "Amsterdam"
RETURN gds.util.asNode(sourceNodeId).id as origin ,gds.util.asNode(targetNodeId).id as destination, distance
```

P7. Run the following code and tell us the distance from London to any other node

```
MATCH (startnode:Place {id: 'London'})
CALL gds.alpha.shortestPath.deltaStepping.stream({
nodeProjection: 'Place',
relationshipProjection: {
EROAD: {
type: 'EROAD',
properties: 'distance'
}
},
startNode: startnode,
relationshipWeightProperty: 'distance',
delta: 3.0
})
YIELD nodeId, distance
RETURN gds.util.asNode(nodeId).id AS Name, distance AS Cost
order by Cost
```

P8. Create a transport graph (nodes+rels)

```
CALL gds.graph.create(
'transportGraph',
'Place',
'EROAD',
{
relationshipProperties: 'distance'
})
```

P9. Now run Dijskstra to get all shortest paths from London to any other node

```
MATCH (source:Place {id: 'London'})
CALL gds.allShortestPaths.dijkstra.stream('transportGraph', {
sourceNode: source,
relationshipWeightProperty: 'distance'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs,
path
RETURN
index,
gds.util.asNode(sourceNode).name AS sourceNodeName,
gds.util.asNode(targetNode).name AS targetNodeName,
totalCost,
[nodeId IN nodeIds | gds.util.asNode(nodeId).name] AS
nodeNames,
costs,
nodes(path) as path
ORDER BY index
```

P10. Get all shortest paths between any pair of nodes

```
CALL gds.alpha.allShortestPaths.stream({ nodeProjection: "*",
relationshipProjection: {
all: {
type: "*",
properties: "distance",
orientation: "UNDIRECTED"
} },
relationshipWeightProperty: "distance"
})
YIELD sourceNodeId, targetNodeId, distance
WHERE sourceNodeId < targetNodeId
RETURN gds.util.asNode(sourceNodeId).id AS source,
gds.util.asNode(targetNodeId).id AS target,
distance
ORDER BY distance DESC
LIMIT 10;
```

Minimum Spanning Tree

With neo4J

P11 (a). Find a spanning tree starting from Amsterdam MINST means minimal spanning tree

This query stores its resultsas MINST relationships.

```
MATCH (n:Place {id:"Amsterdam"})
CALL gds.alpha.spanningTree.minimum.write({
startNodeId: id(n),
nodeProjection: "*",
relationshipProjection: {
EROAD: {
type: "EROAD",
properties: "distance",
orientation: "UNDIRECTED"
} },
relationshipWeightProperty: "distance",
writeProperty: 'MINST',
weightWriteProperty: 'cost'
})
YIELD createMillis, computeMillis, writeMillis, effectiveNodeCount RETURN createMillis,
computeMillis, writeMillis, effectiveNodeCount;
```

P11 (b). What is the MINST route from Amsterdam to Felixstowe?

```
MATCH path = (n:Place {id:"Amsterdam"})-[:MINST*]-()
WITH relationships(path) AS rels
UNWIND rels AS rel
WITH DISTINCT rel AS rel
RETURN startNode(rel).id AS source, endNode(rel).id AS destination, rel.cost AS cost;
```

P11 (c). return the minimum weight spanning tree as graph; from the graph tell us the route from Amsterdam to London

```
MATCH path = (n:Place {id:"Amsterdam"})-[:MINST*]-()
WITH relationships(path) AS rels
UNWIND rels AS rel
WITH DISTINCT rel AS rel
RETURN startNode(rel) AS source, endNode(rel) AS destination, rel AS cost;
```

Centrality

Create database chapter5

:use chapter5

```
WITH 'https://github.com/neo4j-graph-analytics/book/raw/master/data/' AS base
WITH base + 'social-nodes.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (:User {id: row.id});
```

```
WITH 'https://github.com/neo4j-graph-analytics/book/raw/master/data/' AS base
WITH base + 'social-relationships.csv' AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MATCH (source:User {id: row.src})
MATCH (destination:User {id: row.dst}) MERGE (source)-[:FOLLOWS]->(destination);
```

Figure 5-2 illustrates the graph that we want to construct.

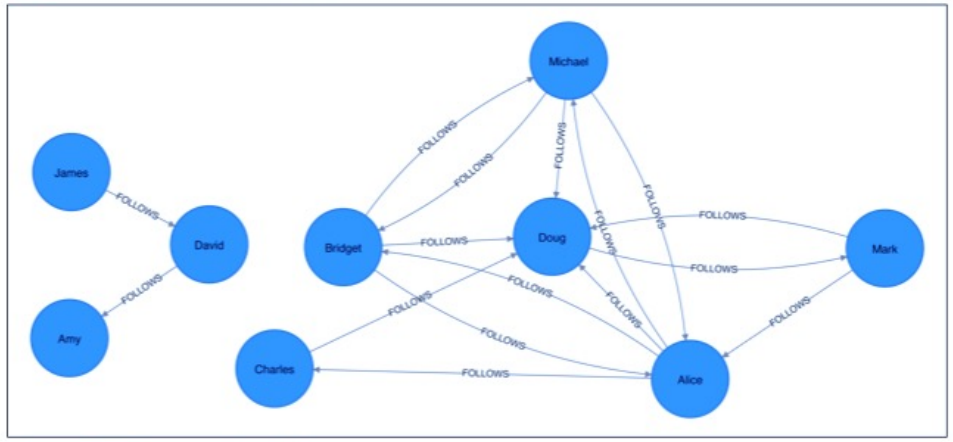


Figure 5-2. The graph model

P12: Closeness Centrality with Neo4j – what are the three most central nodes, in order?

```
CALL gds.alpha.closeness.stream({ nodeProjection: "User", relationshipProjection: "FOLLOWS” })
YIELD nodeId, centrality
RETURN gds.util.asNode(nodeId).id, centrality ORDER BY centrality DESC;
```

P13: What I would like to bve able to do, did not find how to do it yet

- I would like to be able to show nodes of diferente sizes depending on centrality scores, so that I would be able to show larger bubbles for more central nodes in the following:

```
CALL gds.alpha.closeness.stream({ nodeProjection: "User", relationshipProjection: "FOLLOWS” })
YIELD nodeId, centrality
RETURN gds.util.asNode(nodeId), centrality ORDER BY centrality DESC;
```

That way I would be able to show the results graphically, which would be nice

P14: Betweenness Centrality with Neo4j – what are the three most central nodes, in order?

```
CALL gds.betweenness.stream({ nodeProjection: "User",
relationshipProjection: "FOLLOWS"
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).id AS user, score ORDER BY score
DESC;
```

P15: PageRank with Neo4j – what are the three most central nodes, in order?

```
CALL gds.pageRank.stream({ nodeProjection: "User", relationshipProjection: "FOLLOWS",
maxIterations: 20,
dampingFactor: 0.85 })
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).id AS page, score ORDER BY score DESC;
```

Connected componentes, Centrality

P16: please run the three following alternatives and discuss which are the communities based on each. Are they the same? I would like to be able to colour nodes differently based on community

List scc (strongly connected componentes):

```
CALL gds.alpha.scc.stream ({ nodeProjection: "User", relationshipProjection: "FOLLOWS"
})
YIELD nodeId, componentId
RETURN componentId, collect(gds.util.asNode(nodeId).id) AS libraries ORDER BY size(libraries) DESC;
```

As Nodes:

```
CALL gds.alpha.scc.stream ({ nodeProjection: "User", relationshipProjection: "FOLLOWS"
})
YIELD nodeId, componentId
RETURN collect(gds.util.asNode(nodeId)) AS communities ORDER BY size(communities) DESC;
```

List wcc (weakly connected componentes):

```
CALL gds.wcc.stream({
nodeProjection: "User",
relationshipProjection: "FOLLOWS"
})
YIELD nodeId, componentId
RETURN componentId, collect(gds.util.asNode(nodeId).id) AS communities ORDER BY size(communities) DESC;
```

Final words...

- Even if we cannot programmatically change size or colour of nodes, we could at least create diferente types of nodes, one type per community, and then assign a diferente (constant) colour or size to each type of node. That way we would be able to do it.... I did not try that...

The end

…

Extra Cypher questions

Out of class

Questions B

P15. Find actor named 'Taylor Hackford’

P16. Encontra os filmes anteriores a 1995. Mostra (a) os filmes (grafo) e mostra (b) os títulos

P17. Mostra os filmes que incluem “Matrix” no nome. Verifique depois todos os atributos desses filmes

P18. Descobre quem realizou o filme "You've Got Mail”

P19. Quem trabalhou no filme "You've Got Mail”