
# Neo4j & Graph Algorithms

Graph and Mining

A5

**ESILV**
nicolas.travers (at) devinci.fr

# Chapter

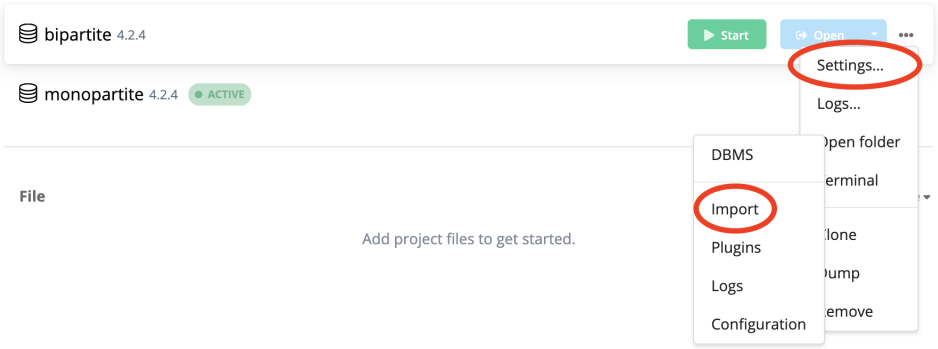
# Contents

# Chapter 1
# Import the Tourism Circulation dataset

The dataset has been used to study the Circulation of tourists on a geographic territory. You will find the corresponding scientific publication here: `https://link.springer.com/chapter/10.1007%2F978-3-030-62005-9_29`

## 1.1 Create your database

- Check the *Neo4j desktop version* (if not already installed). Make an update before.
  Tested versions: Neo4j Desktop 1.5.0, with Neo4j 4.4.8;
- Check if your firewall blocks ports **7474** (Neo4j browser) & **7687** (Bolt protocol[1]);
- Create a projet "*Graph Mining*", and create two DBMS "**bi-partite**" and "**mono-partite**".
  If needed, refer to the guide used last year: `https://chewbii.com/neo4j-travaux-pratiques/`
- In order to guarantee good performances, edit on both graphs the "**Settings...**" in order to put sufficient memory[2]:




```
dbms.memory.heap.initial_size=1G
dbms.memory.heap.max_size=6G
```

**Edit settings**

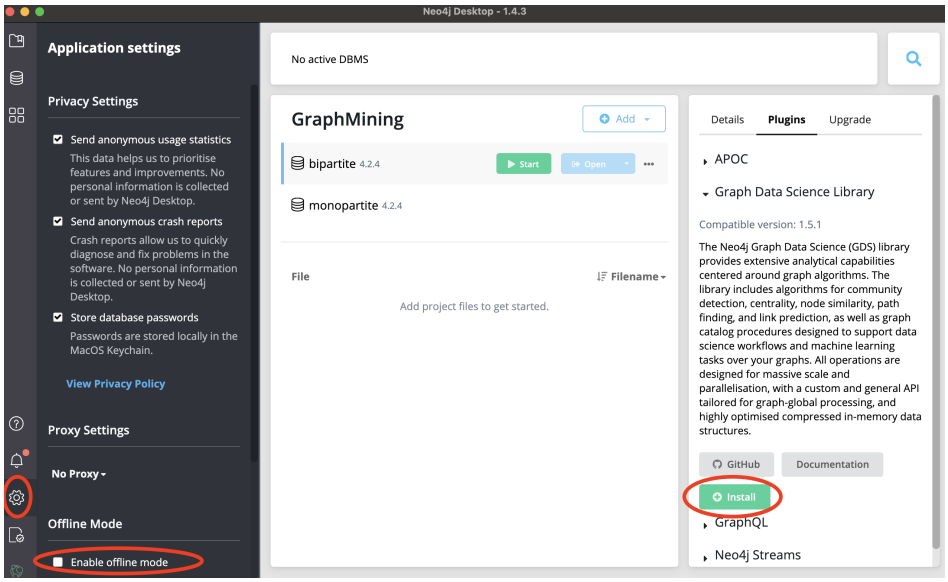
```
# Java Heap Size: by default the Java heap size is dynamically calculated based
# on available system resources. Uncomment these lines to set specific initial
# and maximum heap size.
dbms.memory.heap.initial_size=1G
dbms.memory.heap.max_size=6G
```

Can be 6G if you want to keep more space for the graph. Becareful, do not exceed the amount of memory *left* on your laptop (OS, browser, apps, services take a lot of memory).

## 1.2 Activate GDS

- Click on your database in order to show the *details* bar (right side)[3];
- Click on the *Plugins* tab, then on "Graph Data Science Library";
- Install the plugin;

[1] If you restart the database, it should change the ports. Becareful!
[2] Do not exceed the capacity of your laptop (see "system" on Windows)
[3] If necessary on older versions, remove the "*Enable offline mode*" in **Settings** (bottom left of the Desktop - see picture above)



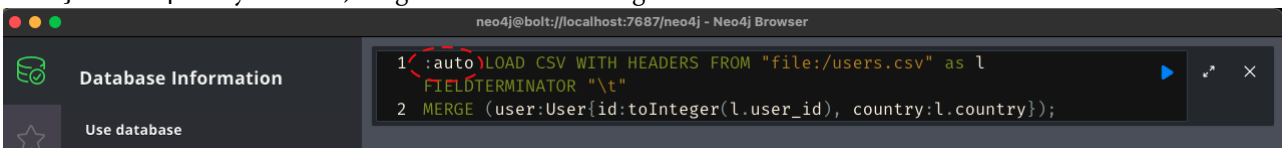

- Do it on both graphs;
- If necessary, restart the database in order to take into account the plugin.

## 1.3 Create your database

Now download the two datasets from DVO, unzip the archives, and put the files in the import folder ("...", "Open Folder", "Import" - see Figure above) for "bipartite" and "monopartite" graphs;

There are **three main ways** to import data:

*1.3.1* `Neo4j Desktop`: easy to show, longer to execute on huge files. Use *":auto"* for autocommit[4]




*1.3.2* `Cypher-shell`: need to be launched in a console (command: *cypher-shell*), use less memory and queries are more efficient. Use *"using periodic commit"* for autocommit.

```
~> cypher-shell
username: neo4j
password: *****
Connected to Neo4j using Bolt protocol version 4.4 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j> USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM "file:/users.csv" as l FIELDTERMINATOR "\t"
             MERGE (user:User{id:toInteger(l.user_id), country:l.country});

0 rows
ready to start consuming query after 22883 ms, results consumed after another 0 ms
Added 1007287 nodes, Set 2014574 properties, Added 1007287 labels
neo4j@neo4j>
```

*1.3.3* `APOC`: driver where you develop your own program to make the connection with the Neo4j database. More useful for most projects, transformations, etc.

[4] A commit is used for transaction databases to keep consistency. However it requires to maintain an history of all updates, consequently it uses a lot of memory. Autocommit helps to flush the transaction automatically and free memory space.

### 1.3.1 Bi-partite Graph

This dataset is an extraction from Tripadvisor reviews where you can find correlations between users (anonymized) and French locations.

- Start the "*bipartite*" database;
- Two ways to import data as mentionned above: *desktop* or *cypher-shell*. I recommend *cypher-shell* which is more efficient (but in shell mode). If you choose "Neo4j desktop"[5] anyway, modify "`USING PERIODIC COMMIT`" by "`:auto`"
- Create indexes as in the following but *one query at a time only*:

```
CREATE INDEX IF NOT EXISTS FOR (u:User) on u.id;
```

  Older Neo4j versions use another syntax[6].

```
CREATE INDEX IF NOT EXISTS FOR (u:User) on u.country;
```

  *one query at a time only*!!!!! (I said it before...)

```
CREATE INDEX IF NOT EXISTS FOR (u:Area_4) on u.gid;
```

```
CREATE INDEX IF NOT EXISTS FOR (u:Area_4) on u.gid_4;
```

- Import `User` nodes:

```
CALL {
   LOAD CSV WITH HEADERS FROM "file:/users.csv" as l FIELDTERMINATOR "\t"
   MERGE (user:User{id:toInteger(l.user_id), country:l.country})
};
```

  As said earlier, "*:auto*" is the autocommit

- Import `Area_4` nodes:

```
CALL {
   LOAD CSV WITH HEADERS FROM "file:/gadm36_4.csv" as l FIELDTERMINATOR "\t"
   MERGE (loc:Area_4{gid:toInteger(l.gid),
        gid_0:l.gid_0,name_0:l.name_0, gid_1:l.gid_1,name_1:l.name_1, gid_2:l.gid_2,name_2:l.name_2,
        gid_3:l.gid_3,name_3:l.name_3, gid_4:l.gid_4,name_4:l.name_4})
};
```

  Gadm3.6 is a database which stores all information according to administrative areas all around the world. Here are extracted information from France.

  - Area_0: Country
  - Area_1: Region
  - Area_2: Department
  - Area_3: District (*Canton* in French)
  - Area_4: Cities (*Communauté de communes* in French)
  - Area_5: Towns (*villes/villages* in French) - not shown in this file

[5] Open the "**bi-partite**" DBMS browser.
If the button is not clickable, use in your own Web browser: `http://localhost:7474`.
[6] CREATE INDEX ON :User(id);

- Import Reviews relationships.
  It can take a while - sometimes several minutes - **do not forget to index nodes and change the heap size!**
  The global file has been split in 10 distinct "reviews" file. Change the file number to import it 10 times.

```
CALL {
  LOAD CSV WITH HEADERS FROM "file:/reviews_0.csv" as l FIELDTERMINATOR "\t"
  MERGE (area:Area_4{gid_4:l.gid_to} )
  MERGE (user:User{id:toInteger(l.user_id)} )
  MERGE (user) -[:review{year:toInteger(l.year),rating:toFloat(l.rating),NB:toInteger(l.NB)}]-> (area)
};
```

  **Execute the same query** with "*reviews_1.csv*" and so on, until "reviews_9".

### 1.3.2 Mono-partite graph

This dataset is a transformation of the bi-partite graph imported previously. Here are the steps already applied:

- Locations are grouped by GADM3.6 at level 4 : **Cities**
- For each couple of reviews from a same user, create a link between the two corresponding cities.
- Group all the links from a given country of origin and year of review to create a weighted relationship.

A Java program has been developped to extract this circulation graph.

- Stop the "*bipartite*" database;
- Start the "*monopartite*" database;
- Open the "**mono-partite_circulation**" DBMS. After downloading the dataset on DVO, unzip the archive, and put the files in the import folder ("...", "Open Folder", "Import"). Then, open the Neo4j browser.
- In order to guarantee good performances, edit the settings in order to put sufficient memory
- Create indexes as in the following but *one query at a time only*:

```
CREATE INDEX IF NOT EXISTS FOR (u:Area_4) on u.gid;
```

```
CREATE INDEX IF NOT EXISTS FOR (u:Area_4) on u.gid_4;
```

- Import circulation relationships:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/circulationGraph_4.csv" as l FIELDTERMINATOR "\t"
MERGE (from:Area_4{gid:toInteger(l.gid_from)} )
MERGE (to:Area_4{gid:toInteger(l.gid_to)} )
MERGE (from) -[:trip{year:toInteger(l.year),NB:toInteger(l.NB),country:l.country}]-> (to);
```

- Import Area nodes (update existing nodes with metadata):

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/gadm36_4.csv" as l FIELDTERMINATOR "\t"
MATCH (loc:Area_4{gid:toInteger(l.gid)})
SET
     loc.gid_0=l.gid_0, loc.name_0=l.name_0, loc.gid_1=l.gid_1, loc.name_1=l.name_1,
     loc.gid_2=l.gid_2,loc.name_2=l.name_2, loc.gid_3=l.gid_3,loc.name_3=l.name_3,
     loc.gid_4=l.gid_4, loc.name_4=l.name_4, loc.lat=l.centroidlat, loc.long=l.centroidlong;
```

- Some algorithms do not work with directed graphs (like for this circulation graph). In order to enable this, you must create a graph undirected[7]. For this, create a new graph database called "**mono-partite undirected**" with the same queries except the one on relationships (the ">" has been removed):

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/circulationGraph_4.csv" as l FIELDTERMINATOR "\t"
MERGE (from:Area_4{gid:toInteger(l.gid_from)} )
MERGE (to:Area_4{gid:toInteger(l.gid_to)} )
MERGE (from) -[:trip{year:toInteger(l.year),NB:toInteger(l.NB),country:l.country}]- (to);
```

Done! You can work on the practice work on both graphs.

[7]Cypher projections keep the native direction property and cannot be changed

# Chapter 2

# Mining Bi-partite Graphs

First, close all graph databases except the "**bi-partite**" graph. Open the browser.

## 2.1 Similarity

*2.1.1* Take the two French users who reviewed the most (sum of NB);

*2.1.2* Give their *Jaccard* Similarity[1] between distinct areas (use `WITH` clause to exploit previous result - user 1 and then user 2);

*2.1.3* Take the two French users who reviewed the most distinct areas. Give their similarity;

*2.1.4* Explain the difference;

*2.1.5* For those couples, give the *overlap* and explain the difference with *Jaccard*;

*2.1.6* For those couples, give the *Euclidean* and *cosine* similarities, using the NB. Explain the difference (between couples and other similarities);

*2.1.7* Idem with ratings (and explanation);

*2.1.8* Do the same for areas in common **only**;

*2.1.9* Give the average *jaccard* and *overlap* similarities[2] for **Spanish** where they visited at least 5 places per area (NB >= 5);

*2.1.10* Give the one for British, American and Italians. Explain the differences.

## 2.2 Link Prediction

*2.2.1* Give the number of common neighbors between the two French who reviewed the most (seen before);

*2.2.2* Give the link prediction on *total neighbors*, *preferential attachment*, *resource allocations* and *Adamic Adar*;

*2.2.3* Explain the differencies;

*2.2.4* Give the top 10 shared neighbors between the top 10 spanish reviewers (sum of NB). Give for all similarities (*total neighbors*, *preferential attachment*, *resource allocations* and *Adamic Adar*) ordered by adamic adar.

*2.2.5* Discuss the result by looking at common neighbors.

[1] gds v2.2 gds.similarity.jaccard
gds v1.5 s.alpha.similarity.jaccard

[2] Euclidean and Cosine must have same vector size which is not always the case.

# Chapter 3

# Mining Mono-partite Graphs

First, close all graph databases except the "**mono-partite**" graph. Open the browser.

## 3.1 Cypher Projection

In the following, we need to create several sub-graphs in order to understand various behavior from the users.

*3.1.1* Create a Cypher Projection[1] named "French2019" where you extract the graph for the French in 2019 population with NB;

*3.1.2* Idem with "French2020", "British2019", "British2020", "US2019", "US2020";

## 3.2 Community Detection

*3.2.1* Give the number of triangles per node for French2019 and French2020, in decreasing order;

*3.2.2* Idem but grouped by department (Area_2). Discuss the result;

*3.2.3* Idem with the *clustering coefficient*. Discuss the result (Infinity and different results);

*3.2.4* Extract communities with "*Label Propagation*" on different Cypher projections;

*3.2.5* From the previous result, give the list of communities per department. Discuss the result;

*3.2.6* Idem with "*Louvain*";

*3.2.7* Group previous result per communityId. Discuss the result;

## 3.3 Path finding

*3.3.1* Give all pairs of shortest paths in French2019 based on NB properties. Need to use a Map configuration instead on CypherProjection (with "relationshipWeightProperty");

*3.3.2* Extract the Minimum Spanning Tree starting from "Paris 1° arrondissement";

*3.3.3* Extract the **Maximum** Spanning Tree;

## 3.4 Centrality

*3.4.1* Extract *PageRank* centralities from nodes in different various cypher projection. Discuss the order of results (weights are dependant on the graph);

*3.4.2* Give the average, min and max PageRank score for corresponding departments. Explain the differences;

*3.4.3* Give Degree, Closeness, Betweenness centralies for those graphs and explain differences.

[1] gds v2: CALL gds.graph.project.cypher(...) (https://neo4j.com/docs/graph-data-science/2.1/algorithms/astar/)
gds v1: CALL gds.graph.create.cypher(...)