Rapport du TP Neo4j
Base de Données Orientées Graphe

Mohamed Dhia Eddine THABET
3ème cycle d'ingénieurs informatique
École Nationale d'Ingénieurs de Carthage (ENIC)

26 octobre 2025

Table des matières

1 I. Installation et Configuration (Docker)

L'environnement Neo4j a été déployé à l'aide de Docker.

1.1 Arrêt d'anciens conteneurs (optionnel)

Pour éviter les conflits, les conteneurs existants ont été arrêtés et supprimés.

```
docker stop neo4j-tp
docker rm neo4j-tp
```

1.2 Lancement du conteneur Neo4j

Le conteneur a été lancé en mappant les ports (7474, 7687), en montant un volume pour les fichiers CSV, en définissant un mot de passe et en incluant le plugin APOC (bien que non utilisé finalement pour le LOAD CSV).

Le chemin du volume `/run/media/dhia/New Volume1/NEO4j-Lab/neo4j dataset` a nécessité des guillemets à cause des espaces.

```
docker run \
    --name neo4j-tp \
    -p 7474:7474 \
    -p 7687:7687 \
    -d \
    -v "/run/media/dhia/New Volume1/NEO4j-Lab/neo4j dataset":/var/lib/neo4j/
    import \
    -e NEO4J_AUTH=neo4j/00000000 \
    -e NEO4J_PLUGINS='["apoc"]' \
    neo4j:latest
```

2 II. Correction des Données (Nettoyage CSV)

L'importation initiale avec LOAD CSV a échoué à cause de lignes mal formatées dans le fichier `airports.csv`.

2.1 Ouverture du fichier

Le fichier a été ouvert avec nano pour correction manuelle.

```
nano "/run/media/dhia/New Volume1/NEO4j-Lab/neo4j dataset/airports.csv"
```

2.2 Lignes corrigées

Les lignes suivantes ont été identifiées et nettoyées.

2.2.1 Ligne 1 (Magdeburg)

Avant :

```
332,"Magdeburg \City\"" Airport""",Magdeburg,Germany,\N,EDBM,52.073...
```

Après :

```
332,"Magdeburg City Airport",Magdeburg,Germany,\N,EDBM,52.073...
```

2.2.2 Ligne 2 (Szczecin)

Avant :

```
676,"Szczecin-Goleniów \Solidarność\"" Airport""",Szczecin,Poland,SZZ...
```

Après :

```
676,"Szczecin-Goleniów Solidarność Airport",Szczecin,Poland,SZZ...
```

3 III. Importation et Construction des Graphes

Une fois les données nettoyées, les étapes d'importation ont été exécutées.

3.1 Nettoyage de la base (pré-importation)

Avant de lancer l'importation, la base de données a été vidée.

```
MATCH (n) DETACH DELETE n
```

3.2 1. Importation de "airport.csv"

```
LOAD CSV WITH HEADERS FROM "file:/airports.csv" as l
CREATE (airport:Airport{id:toInteger(l.AirportID), name:l.Name, city:l.City,
country:l.Country, IATA:l.IATA, latitude:toFloat(l.Latitude),
longitude: toFloat(l.Longitude), altitude: toFloat(l.Altitude), TimeZone:l.TZ});
```

3.3 2. Importation de "airlines.csv"

```
LOAD CSV WITH HEADERS FROM "file:/airlines.csv" as l
CREATE (airline:Airline{id:toInteger(l.AirlineID), name:l.Name, alias:l.Alias,
   IATA:l.IATA,
country:l.Country, active:l.Active});
```

3.4 3. Création des indexes (Syntaxe moderne)

Les commandes d'index du TP utilisaient une syntaxe obsolète. Elles ont été mises à jour pour Neo4j 4.x/5.x.

```
CREATE INDEX airport_id_index FOR (n:Airport) ON (n.id);
CREATE INDEX airline_id_index FOR (n:Airline) ON (n.id);
CREATE INDEX route_id_index FOR (n:Route) ON (n.id);
CREATE INDEX airport_country_index FOR (n:Airport) ON (n.country);
CREATE INDEX airport_city_index FOR (n:Airport) ON (n.city);
CREATE INDEX airport_iata_index FOR (n:Airport) ON (n.IATA);
CREATE INDEX route_name_index FOR (n:Route) ON (n.name);
```

3.5 4. Création des nœuds d'itinéraires (Syntaxe moderne)

La clause `USING PERIODIC COMMIT` a été remplacée par `CALL {...} IN TRANSACTIONS`.

3.5.1 Activation du mode :auto

Cette commande est nécessaire dans le Neo4j Browser pour ce type de transaction.

```
:auto
```

3.5.2 Importation des routes

```
LOAD CSV WITH HEADERS FROM "file:/routes.csv" as l
CALL {
    WITH l
    MERGE (airline:Airline {id: toInteger(l.AirlineID)})
    MERGE (source:Airport {id: toInteger(l.SourceAirportID)})
    MERGE (dest:Airport {id: toInteger(l.DestAirportID)})
    CREATE (route:Route {equipment: l.Equipment})
    CREATE (route)-[:from]->(source)
    CREATE (route)-[:to]->(dest)
    CREATE (route)-[:by]->(airline)
} IN TRANSACTIONS OF 1000 ROWS;
```

4 IV. Cypher : Requêtes sur les Graphiques

4.1 1. Nom et IATA des aéroports de France

```
MATCH (a:Airport {country: 'France'})
RETURN a.name AS Nom, a.IATA AS CodeIATA;
```

4.2 2. Compagnies aériennes françaises actives avec IATA

```
MATCH (a:Airline {country: 'France', active: 'Y'})
WHERE a.IATA IS NOT NULL AND a.IATA <> '\\N'
RETURN a.name AS Nom, a.IATA AS CodeIATA;
```

4.3 3. Compagnies françaises avec au moins une route

```
MATCH (a:Airline {country: 'France'})<-[:by]-(:Route)
RETURN DISTINCT a.name AS Nom;
```

4.4 4. Graphe des routes au départ de CDG

```
MATCH p = (cdg:Airport {IATA: 'CDG'})<-[:from]-(r:Route)-[:to]->(dest:Airport)
MATCH q = (r)-[:by]->(airline:Airline)
RETURN p, q
LIMIT 100;
```

4.5 5. Graphe des routes au départ de CDG par A380

```
MATCH p = (cdg:Airport {IATA: 'CDG'})<-[:from]-(r:Route)-[:to]->(dest:Airport)
WHERE r.equipment CONTAINS 'A380'
MATCH q = (r)-[:by]->(airline:Airline)
RETURN p, q;
```

4.6 6. Villes et Pays de destination (départ CDG, A380)

```
MATCH (cdg:Airport {IATA: 'CDG'})<-[:from]-(r:Route)-[:to]->(dest:Airport)
WHERE r.equipment CONTAINS 'A380'
RETURN DISTINCT dest.city AS Ville, dest.country AS Pays;
```

4.7 7. Villes, Pays et Compagnie (départ CDG, A380)

```
MATCH (cdg:Airport {IATA: 'CDG'})<-[:from]-(r:Route)-[:to]->(dest:Airport)
WHERE r.equipment CONTAINS 'A380'
MATCH (r)-[:by]->(airline:Airline)
RETURN DISTINCT dest.city AS Ville, dest.country AS Pays, airline.name AS
   Compagnie;
```

4.8 8. Graphe des itinéraires entre CDG et aéroports français

```
MATCH p = (cdg:Airport {IATA: 'CDG'})<-[:from]-(:Route)-[:to]->(dest:Airport {
    country: 'France'})
RETURN p
UNION
MATCH p = (src:Airport {country: 'France'})<-[:from]-(:Route)-[:to]->(cdg:
    Airport {IATA: 'CDG'})
RETURN p;
```

4.9 9. Graphe de toutes les routes par A380

```
MATCH p = (src:Airport)<-[:from]-(r:Route)-[:to]->(dest:Airport)
WHERE r.equipment CONTAINS 'A380'
MATCH q = (r)-[:by]->(airline:Airline)
RETURN p, q;
```

4.10 10. Graphe routes : France vers Royaume-Uni (UK)

```
MATCH p = (src:Airport {country: 'France'})<-[:from]-(r:Route)-[:to]->(dest:
    Airport {country: 'United Kingdom'})
MATCH q = (r)-[:by]->(airline:Airline)
RETURN p, q;
```

5 V. Requêtes Complexes

5.1 1. Création des relations homogènes :path

```
MATCH (FROM:Airport) <-[:from]- (r:Route) -[:to]-> (TO:Airport), (r) -[:by]-> (comp)
WHERE FROM <> TO
MERGE (FROM)-[:path {airline: comp.name}]->(TO);
```

5.2 2. Création de l'index sur la relation :path

La syntaxe du TP a été mise à jour pour un index de relation.

```
CREATE INDEX path_airline_idx FOR ()-[r:path]-() ON (r.airline);
```

5.3 3. Graphe des paths "Air France" entre aéroports français

```
MATCH p = (a:Airport {country: 'France'})-[r:path {airline: 'Air France'}]->(b:Airport {country: 'France'})
RETURN p;
```

5.4 4. Nombre de paths "Air France" par pays destination

```
MATCH (a:Airport)-[r:path {airline: 'Air France'}]->(b:Airport)
RETURN b.country AS PaysDestination, count(r) AS NombreDePaths
ORDER BY NombreDePaths DESC;
```

5.5 5. Chemins (longueur 2-3) de Nantes à Salt Lake City

```
MATCH p = (nantes:Airport {city: 'Nantes'})-[:path*2..3]->(slc:Airport {city: 'Salt Lake City'})
RETURN p;
```

5.6 6. Chemin le plus court de Nantes à Salt Lake City

```
MATCH p = shortestPath(
  (nantes:Airport {city: 'Nantes'})-[:path*1..10]->(slc:Airport {city: 'Salt Lake City'})
)
RETURN p;
```