TP – Prise en main de la base graph Neo4j

1 Introduction

Ce rapport décrit les étapes de prise en main d'une base de données graphe avec Neo4j. L'objectif est de créer une base contenant des informations sur des aéroports, des compagnies aériennes, et les routes les reliant, en utilisant des fichiers CSV et les requêtes Cypher.

2 Étapes d'installation et de création de la base de données

1. Lancer Neo4j Desktop.
2. Créer une nouvelle base de données dans un projet.
3. Démarrer la base de données (modifier les ports si nécessaire).
4. Accéder au dossier d'import via *Open Folder*.
5. Ouvrir l'interface web : `http://localhost:7474`.
6. Utiliser le mot de passe par défaut : `neo4j`.

3 Préparation des fichiers CSV

Les fichiers CSV nécessaires sont les suivants :

- `airports.csv` : informations sur les aéroports.
- `airlines.csv` : informations sur les compagnies aériennes.
- `routes.csv` : informations sur les routes.

Ces fichiers doivent être placés dans le dossier import de Neo4j Desktop : `$Neo4j_folder/import/`

4 Commandes Cypher utilisées

Création des nœuds Airport

```
LOAD CSV WITH HEADERS FROM "file:/airports.csv" AS l
CREATE (: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
});
```

Création des nœuds Airline

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

Création des index

```
CREATE INDEX airport_id_index FOR (a:Airport) ON (a.id);
CREATE INDEX airline_id_index FOR (a:Airline) ON (a.id);
```

Création des routes et relations

```
LOAD CSV WITH HEADERS FROM "file:/routes.csv" AS l
MATCH (source:Airport {id: toInteger(l.SourceAirportID)})
MATCH (dest:Airport {id: toInteger(l.DestinationAirportID)})
MATCH (airline:Airline {id: toInteger(l.AirlineID)})
CREATE (f:Flight { routeId: toInteger(l.RouteID) })
CREATE (f)-[:SOURCE]->(source)
CREATE (f)-[:DESTINATION]->(dest)
CREATE (f)-[:OF]->(airline);
```

Création des nœuds City et relations IS_IN

```
MATCH (a:Airport)
MERGE (c:City {name: a.city})
MERGE (a)-[:IS_IN]->(c);
```

Création des nœuds Country et relations EXIST

```
MATCH (a:Airport)
MERGE (co:Country {name: a.country})
WITH a, co
MATCH (a)-[:IS_IN]->(c:City)
MERGE (c)-[:EXIST]->(co);
```

Relation Airline vers Country (BELONGS)

```
MATCH (air:Airline)
MERGE (co:Country {name: air.country})
MERGE (air)-[:BELONGS]->(co);
```

5 Visualisation du graphe

Le graphe a été visualisé dans Neo4j Browser. Les nœuds représentent les entités (`Airport`, `Airline`, `Flight`, `City`, `Country`) et les arêtes montrent leurs relations (`SOURCE`, `DESTINATION`, `OF`, `IS_IN`, `EXIST`, `BELONGS`). Chaque type de nœud est coloré pour faciliter la lecture.

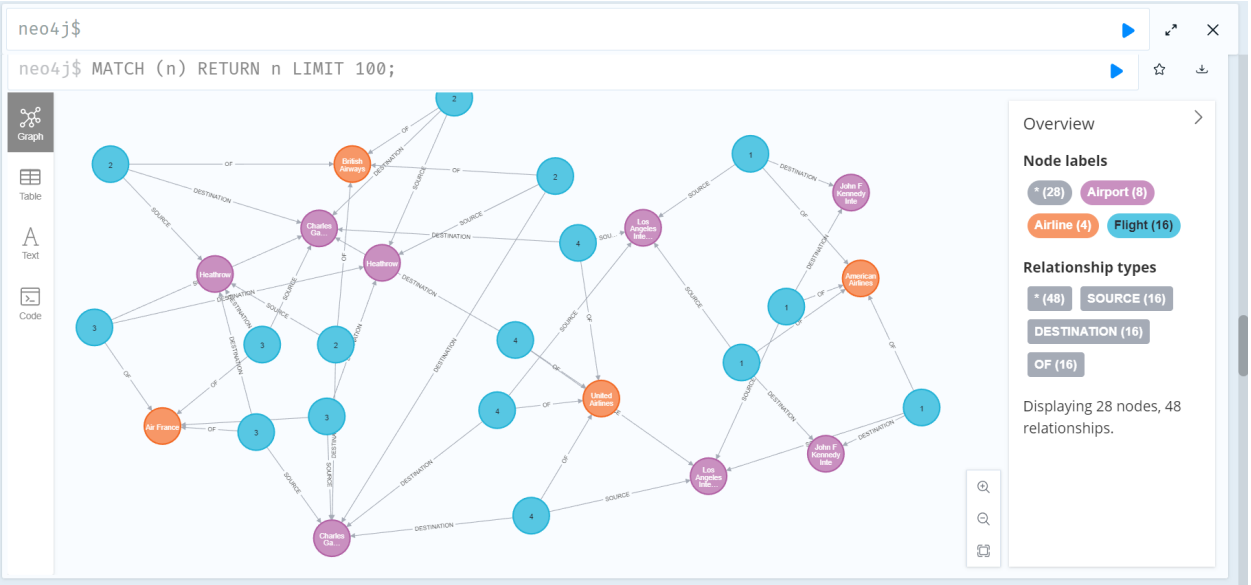

Figure 1: Visualisation du graphe dans Neo4j Browser

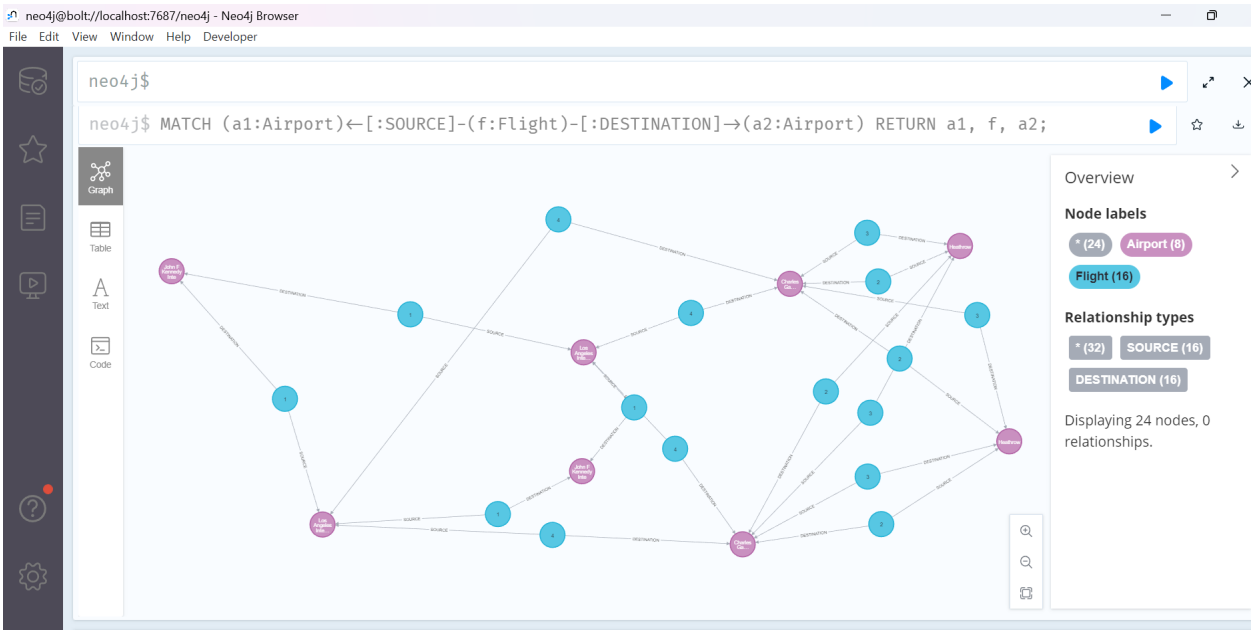

Figure 2: Visualisation du graphe dans Neo4j Browser

Figure 3: Visualisation du graphe dans Neo4j Browser

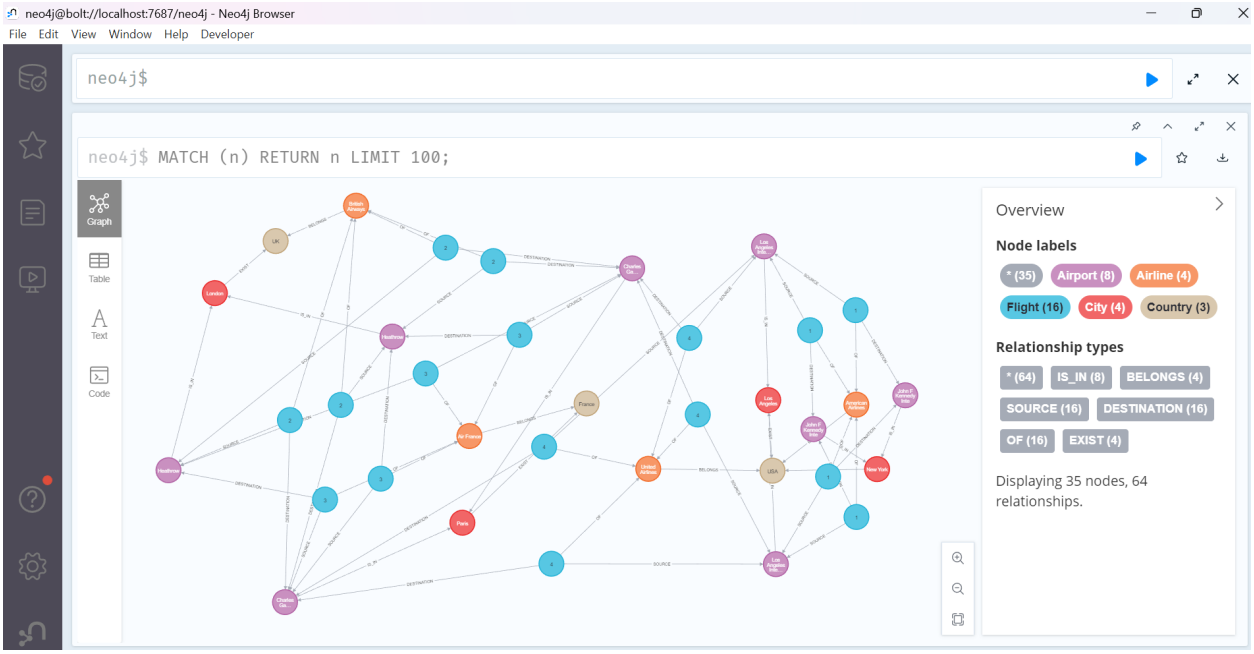

Figure 4: Visualisation du graphe dans Neo4j Browser

6 Conclusion

Ce TP m'a permis de découvrir les bases de données graphes avec Neo4j, de manipuler des données issues de fichiers CSV et de modéliser des relations complexes entre sentités. La création de nœuds, d'index et de relations à l'aide de Cypher est un outil puissant pour explorer des données interconnectées.