Le 17/12/2018

Théophane Tassy, Corentin DOUÉ
ICM 2016

MINES
Saint-Étienne

Défi Big Data
Big Graphs

Introduction

Les données de Big Data sont souvent ultra connectées. Dans certain cas les relations entre les entités est autant (voir même plus important) que les attributs de l'entité C'est pourquoi leur représentation et implémentation sous forme de graphs est parfois adaptée. Dans ce TP nous allons nous familiariser avec neo4j, une implémentation de bases de données à base de graphes. Puis pour interagir avec cette base de données nous allons utiliser le langage GraphQL qui permet de créer une API ultra flexible est efficace au-dessus d'une base de données.

Neo4j

Prise en main

Nous avons tout d'abord appréhendé les concepts de Neo4j en lisant sa documentation.

Puis nous avons suivis le tutoriel suivant :

https://neo4j.com/graphacademy/online-training/introduction-graph-databases/

Il nous a permis de nous familiariser avec Cypher, le langage de Neo4j pour interagir avec la base (créations et modifications)

Installation

Pour pouvoir implémenter un petit projet personnel, nous avons installer Neo4j sur nos machines personnelles. Nous avons alors accès a une application web interface graphique avec la base de données disponible à l'adresse : http://localhost:7474/browser/ et a un point d'accès à la base de données pour les drivers à l'adresse : bolt://localhost:7687

Mini projet

Étant administrateur des sites BDE et Cercle (le bar de la maison des élèves) nous avons accès à une base de données contenant tous les élèves et s'ils sont cotisant ou non à ces deux associations. De plus

on peut savoir qui est membre de ces associations en regardant qui sont les administrateurs de ces sites.

Nous avons donc décidé de modéliser cette base de données sous forme de graph ayant deux types de nœuds : les utilisateurs (Users) et les associations (Unions). Nous modélisons ensuite deux types de relation entre ces nœuds : le fait que l'utilisateur soir un cotisant de l'association ((:User)-[:CONTRIBUTE]->(:Union)) et le fait que l'utilisateur soit un membre de l'association ((:User)-[:IS_MEMBER_OF]->(:Union)).

Implémentation

Nous avons choisi d'implémenter ce projet avec un script Python à l'aide des bibliothèques neo4j-driver pour interagir avec la base de données de néo4j et mysql-connector pour interagir avec la base de données existante.

Le code ce trouve sur le GitHub suivant : https://github.com/CorentinDoue/tp_big_graph.git.

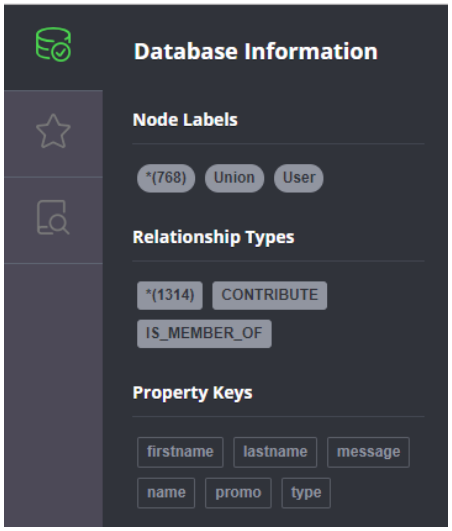
Il faut suivre le readme pour installer correctement le projet.

On y effectue les requêtes Cypher nécessaire à la création des deux types de nœud puis des relations entre eux. On effectue ensuite quelques requêtes pour vérifier que tout à bien été importé correctement.

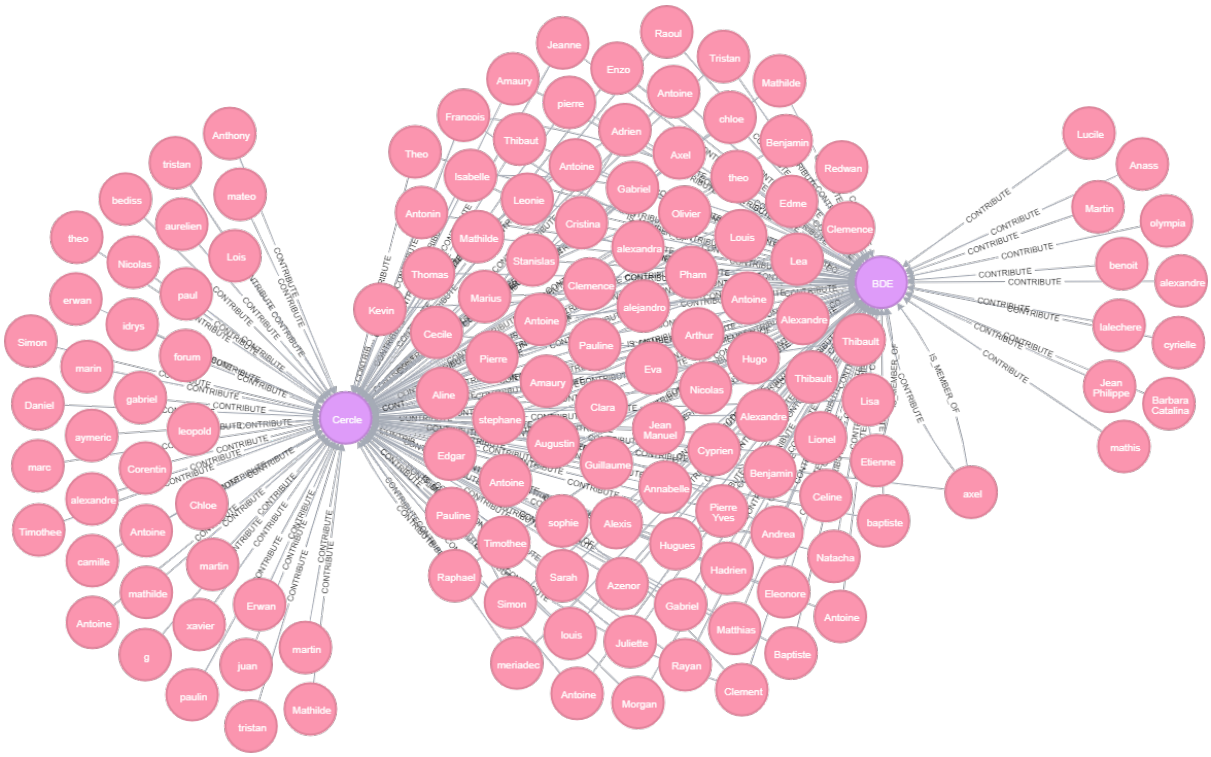
Pour faciliter la reproduction de l'expérience sur votre propre machine, nous avons stocké les variables contenant l'export de la base données dans des fichiers pickle.

Résultats

Après l'exécution script, sur l'interface graphique on remarque que la base de données a été peuplé de 768 nœuds qui sont de type Union ou User et qu'il y a 1314 relations entre ces nœuds répartie entre des CONTRIBUTE ou des IS_MEMBER_OF.



Il est impossible d'afficher le graph complet mais en ce limitant à la promotion 2017 on obtient :



On remarque aux extrémité les utilisateurs n'étant cotisant que d'une des associations et qu'au centre les autres sont cotisants aux deux associations. En tirant le nœud « axel » du bloc central on se rend bien compte qu'il est bien membre du BDE (ce qui implique qu'il est cotisant) et qu'il est cotisant Cercle.

L'exécution des requêtes de test nous renvoie d'abord tout les cotisants BDE (la liste est longue), puis tout les membres de Cercle puis les associations dont Corentin est cotisant puis celle dont il est membre.

```
[<Node id=720 labels={'User'} properties={'promo': 2014, 'firstname': 'myan', 'type': 'ICM', 'lastname': 'nguyen'}>]
[<Node id=721 labels={'User'} properties={'promo': 2014, 'firstname': 'Cecile', 'type': 'ICM', 'lastname': 'PATTE'}>]
[<Node id=722 labels={'User'} properties={'promo': 2014, 'firstname': 'valentin', 'type': 'ICM', 'lastname': 'giraudon'}>]
[<Node id=723 labels={'User'} properties={'promo': 2014, 'firstname': 'paul', 'type': 'ICM', 'lastname': 'cussac'}>]
[<Node id=724 labels={'User'} properties={'promo': 2014, 'firstname': 'Alice', 'type': 'ICM', 'lastname': 'HELFT'}>]
[<Node id=725 labels={'User'} properties={'promo': 2014, 'firstname': 'kim', 'type': 'ICM', 'lastname': 'verhertbruggen'}>]
Members of the Cercle
[<Node id=620 labels={'User'} properties={'promo': 2017, 'firstname': 'Adrien', 'type': 'ICM', 'lastname': 'NAUTRE'}>]
[<Node id=598 labels={'User'} properties={'promo': 2017, 'firstname': 'Antoine', 'type': 'ICM', 'lastname': 'NORET'}>]
[<Node id=581 labels={'User'} properties={'promo': 2017, 'firstname': 'Antoine', 'type': 'ICM', 'lastname': 'LALECHERE'}>]
[<Node id=551 labels={'User'} properties={'promo': 2017, 'firstname': 'Olivier', 'type': 'ICM', 'lastname': 'PANICO'}>]
[<Node id=536 labels={'User'} properties={'promo': 2017, 'firstname': 'Nicolas', 'type': 'ICM', 'lastname': 'VAN DER LAAN'}>]
[<Node id=525 labels={'User'} properties={'promo': 2017, 'firstname': 'Clement', 'type': 'ICM', 'lastname': 'DUBOST'}>]
[<Node id=524 labels={'User'} properties={'promo': 2017, 'firstname': 'Simon', 'type': 'ICM', 'lastname': 'LARDIT'}>]
[<Node id=522 labels={'User'} properties={'promo': 2017, 'firstname': 'Andrea', 'type': 'ICM', 'lastname': 'BARONCELLI'}>]
[<Node id=424 labels={'User'} properties={'promo': 2016, 'firstname': 'Corentin', 'type': 'ICM', 'lastname': 'DOUE'}>]
Unions where Corentin Doué is contributor:
[<Node id=675 labels={'Union'} properties={'name': 'Cercle'}>]
[<Node id=674 labels={'Union'} properties={'name': 'BDE'}>]
Unions where Corentin Doué is member:
[<Node id=674 labels={'Union'} properties={'name': 'BDE'}>]
[<Node id=675 labels={'Union'} properties={'name': 'Cercle'}>]

Process finished with exit code 0
```

Ce n'est qu'un petit aperçu de ce qu'il est possible de faire avec neo4j mais on imagine bien qu'avec cet outil on pourrait modéliser l'ensemble de la cartographie associative de l'école et d'autres nœuds pourrait y être lié comme les événements par exemple. (Avec les relations Union-ORGANIZE->Event et User-PARTICIPE->Event)

GraphQL

Prise en main

Nous sommes d'abord familiarisés avec les concepts autour de GraphQL en suivant les vidéos du tutoriel https://www.howtographql.com/

Mini Projet

Nous avons essayé d'utiliser notre base de données précédemment construite avec Neo4j pour en faire une API GraphQL. Mais après avoir regardé la documentation autour de graphql-python, il nous est apparu qu'il n'était pas possible (ou du moins pas facilement) de relier graphql-python à une bdd déjà existante. Il vaut mieux en effet créer un projet adapté à GraphQL avec la syntaxe qui lui est propre (en fonction du langage)

Nous créons donc une application en Django qui modélisera notre projet précédent en suivant le tutoriel de howtographql. La base donnée ne sera pas neo4j mais une sqlite.

Implémentation

Nous suivons donc le tutoriel est construisons un backend en Django (Python) mais nous utilisons un model relationnel classique dans lequel les relations entre User et Union sont des relations many to many :

```
class Union(models.Model):
    name = models.TextField()

class User(models.Model):
    firstname = models.TextField()
    lastname = models.TextField()
    type = models.TextField(blank=True)
    promo = models.IntegerField()
    contibuted_unions = models.ManyToManyField(Union,
related_name="contributors")
    personal_unions = models.ManyToManyField(Union, related_name="members")
```

Le server se démarre en ligne de commande : python manage.py runserver

Nous avons accès à une interface graphique à l'adresse http://localhost:8000/graphql/ pour simuler des requêtes à l'API. Nous ajoutons donc deux utilisateurs et deux associations. Puis nous ajoutons le fait que les utilisateurs ont cotiser à certaines associations et que certain sont membres des associations.

A des fins de test nous effectuons quelques requêtes pour vérifier le bon comportement de l'API

Résultats

Ajout des utilisateurs et des associations

GraphiQL Prettify History

```
mutation {
  createUser (
    firstname: "Théophane",
    lastname: "Tassy",
    type: "ICM",
    promo: 2016
  ) {
    id
    firstname
    lastname
    type
    promo
  }
}
```

GraphiQL Prettify History

```
mutation {
  createUnion (
    name: "Cercle"
  ) {
    id
    name
  }
}
```

GraphiQL Prettify History

```
query {
  users {
    id
    firstname
    lastname
    type
    promo
  }
}
```


```
{
  "data": {
    "users": [
      {
        "id": "1",
        "firstname": "Corentin",
        "lastname": "Doué",
        "type": "ICM",
        "promo": 2016
      },
      {
        "id": "2",
        "firstname": "Théophane",
        "lastname": "Tassy",
        "type": "ICM",
        "promo": 2016
      }
    ]
  }
}
```


GraphiQL

```
query {
  unions {
    id
    name
  }
}
```


```
{
  "data": {
    "unions": [
      {
        "id": "1",
        "name": "BDE"
      },
      {
        "id": "2",
        "name": "Cercle"
      }
    ]
  }
}
```

Ajout des relations

```
mutation{
  addContribution(
    idUnion: 1,
    idUser: 1
  ) {
    user {
      id
      firstname
      lastname
    }
    union {
      id
      name
    }
  }
}
```

```
mutation{
  addMembership(
    idUnion: 1,
    idUser: 1
  ) {
    user {
      id
      firstname
      lastname
    }
    union {
      id
      name
    }
  }
}
```

```
query{
    unions {
      id
      name
      contributors {
        id
        firstname
        lastname
      }
      members{
        id
        firstname
        lastname
      }
    }
}
```


```
"data": {
  "unions": [
    {
      "id": "1",
      "name": "BDE",
      "contributors": [
        {
          "id": "1",
          "firstname": "Corentin",
          "lastname": "Doué"
        }
      ],
      "members": [
        {
          "id": "1",
          "firstname": "Corentin",
          "lastname": "Doué"
        }
      ]
    },
    {
      "id": "2",
      "name": "Cercle",
      "contributors": [
        {
          "id": "1",
          "firstname": "Corentin",
          "lastname": "Doué"
        },
        {
          "id": "2",
          "firstname": "Théophane",
          "lastname": "Tassy"
        }
      ],
      "members": [
        {
          "id": "1",
          "firstname": "Corentin",
          "lastname": "Doué"
        }
      ]
    }
  ]
```

```
query {
  users{
    id
    firstname
    lastname
    type
    promo
    contibutedUnions{
      name
    }
    personalUnions{
      name
    }
  }
}
```


```
{
  "data": {
    "users": [
      {
        "id": "1",
        "firstname": "Corentin",
        "lastname": "Doué",
        "type": "ICM",
        "promo": 2016,
        "contibutedUnions": [
          {
            "name": "BDE"
          },
          {
            "name": "Cercle"
          }
        ],
        "personalUnions": [
          {
            "name": "BDE"
          },
          {
            "name": "Cercle"
          }
        ]
      },
      {
        "id": "2",
        "firstname": "Théophane",
        "lastname": "Tassy",
        "type": "ICM",
        "promo": 2016,
        "contibutedUnions": [
          {
            "name": "Cercle"
          }
        ],
        "personalUnions": []
      }
```