HEG

HESConnect

Base de données avancée

Gazaoui Myriam - Antonio Antelo
23.12.2022

Table des matières

Contexte

HESConnect est un réseau social développé par deux étudiants en Informatique de Gestion. Cette application est destinée à réunir les étudiants, assistants et professeurs des différentes Hautes Ecoles Spécialisées afin de faciliter la création d'un réseau professionnel.

Les utilisateurs de l'application HESConnect peuvent contacter les membres qu'ils connaissent ou qu'ils pourraient connaître par le biais d'une connaissance en commun. HESConnect propose aussi le partage d'événements en lien avec les différentes Hautes Ecoles et les filières qu'elles proposent. Les événements peuvent être disponibles sur la page de la HES organisatrice, par le biais d'une barre de recherche et aussi faire l'objet de recommandations.

Simplifier la communication au sein des Hautes Ecoles permet de créer des relations interdisciplinaire, impliquant chaque individu lié à ces institutions.

Prérequis

Voici les différentes étapes nécessaires à l'installation de la librairie APOC, nous avons également inclus au dossier les fichiers nécessaires.

To install the APOC library in your Neo4j database, follow these steps:

Download the latest APOC release from the Neo4j website: https://neo4j.com/download/other-releases/
Extract the contents of the downloaded file to a location on your computer.
Copy the `apoc-X.X.X.jar` file from the extracted folder to the `plugins` directory of your Neo4j installation.
Open the `neo4j.conf` file in a text editor and add the following line:
Copy code

```
dbms.security.procedures.unrestricted=apoc.*
```

Save the changes to the `neo4j.conf` file and restart your Neo4j server.
Once the server has restarted, you should be able to use the APOC functions in your Cypher queries.
Note: The specific steps for installing APOC may vary depending on your operating system and version of Neo4j. If you encounter any issues during the installation process, you may want to consult the APOC documentation or seek additional help from the Neo4j community.

Données utilisées

Avant d'aborder le choix de la base de données NoSQL, il est nécessaire de rentrer dans le détail des données utiles à la conception de ce projet. Toutes les données sont chargées via des documents textes « .csv » sur l'application java. Tous les objets ont été instanciés avant et après utilisation des requêtes.

Personne

Des étudiants, des professeurs, des assistants sont des personnes qui ont comme relations : ***ETUDIE***, ***ENSEIGNE_POUR*** ou ***ASSISTE_POUR*** avec les différentes filières que les HES proposent. Chaque Personne peut avoir une relation ***CONNAIT*** auprès d'une autre Personne.

Ce set de données a été généré aléatoirement à l'aide du site web Mockaroo.

Attributs d'une Personne : NOM, PRÉNOM, MAIL, GENRE, CODE POSTAL

Les mails ont été générés aléatoirement et modifiés dans l'application java afin de répondre aux besoins de l'application HESConnect de manière cohérente. Le nom de domaine du mail détermine la nature du rôle de la personne au sein d'une Haute Ecole Spécialisée.

Ecole *

HEAD, HETS, HES, HEG sont les écoles qui ***COMPOSENT*** le réseau HES.

Attribut : NOM DE L'HES

Filière *

Chaque filière (IG, IBM, EE...) ***APPARTIENT*** à une Haute Ecole Spécialisée.

Attribut : NOM DE LA FILIÈRE

Compétence *

Les compétences sont prédéterminées et ***DISPENSEE_DANS*** dans les différentes filières.

Attribut : Nom de la compétence

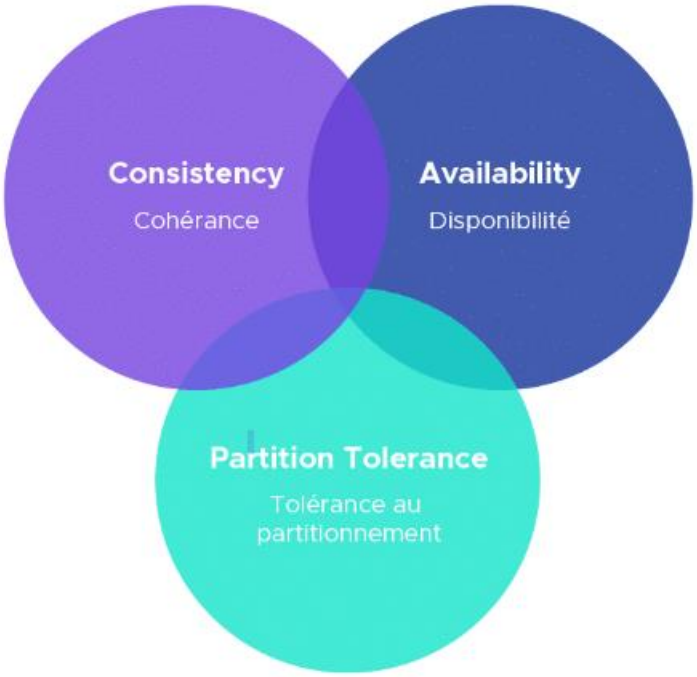
Evénement*

Des conférences, des festivals ou encore des soirées peuvent être ***PROPOSEES PAR*** une HES.

Attributs : Nom de l'événement, thématique(s) concernée(s)<>.

* NB : Ces données ne peuvent pas être générées aléatoirement car elles assurent la cohérence du projet, utiliser des données qui ne sont pas en lien avec le sujet n'aurait pas représenté fidèlement le projet.

Choix de la base de données



.

Comme indiqué dans le théorème CAP, les trois propriétés que sont la cohérence, la disponibilité et la tolérance au partitionnement ne peuvent pas être remplies simultanément.

Le choix d'une base de données NoSQL se justifie dans notre cas de figure par le fait que nous n'avons pas besoin de données identiques et cohérentes pour proposer nos services. Une application répondant aux principes d'un réseaux social est caractérisée, comme pour imiter la vie réelle, par le nombre d'interactions, de partages et de relations très changeantes entre les entités. Les données générées évoluent à chaque seconde. Elles ne doivent donc pas être altérées lors d'un changement quelconque et doivent fournir les services proposés peu importe l'évolution des relations.

La base de données recherchées afin de développer l'application HESConnect devait répondre à ces besoins :

- La disponibilité des données : Il ne faut pas que le fonctionnement du système soit menacé par la disparition d'un nœud.
  Exemple : Suppression d'un événement, il ne faut pas que les autres événements ou que la HES organisatrice rencontre des problèmes à la suite de cette suppression. Il faut simplement que la relation soit supprimée sans conséquence sur les nœuds concernés.
- L'autonomie des nœuds dans leur fonctionnement (partitionnement). Contrairement au cas d'une base de données SQL, il n'est pas nécessaire de supprimer toutes les relations en lien avec un nœud lors de sa suppression. Aucune dépendance ne doit être établie puisque les données sont entres-elles interchangeable.

Base de données orientée graphe VS orientée document

L'application HESConnect est un réseau social permettant de lier les membres des différentes Hautes Ecoles Spécialisées en indiquant les statuts que l'on peut rencontrer, à savoir : les professeurs, leurs assistants et les étudiants. Le principe du réseau social est de lier des personnes qui ont un intérêt commun autour, par exemple, d'événements ou de compétences.

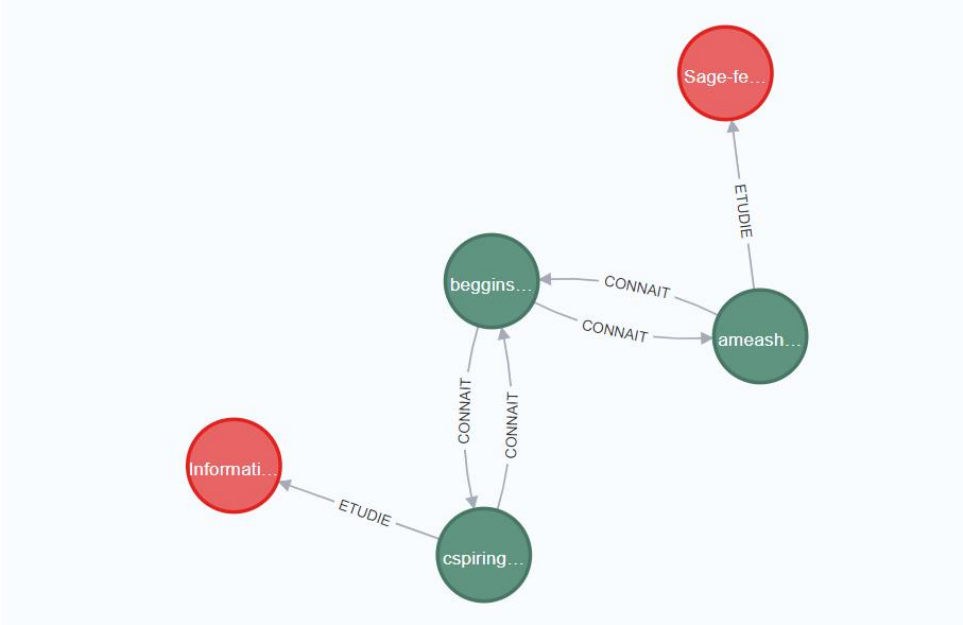
Les entités entres-elles peuvent avoir différents liens plus ou moins complexes, ces liens peuvent évoluer sous différentes formes et que notre besoin principal en matière de base de données est la recherche d'une flexibilité dans la création des relations et dans l'accès aux données.

Notre besoin concernant la gestion des données est de pouvoir créer des relations entre chaque nœud et de favoriser l'accès aux différents nœuds en fonction de relations déjà créées, d'une manière simplifiée. L'utilisation d'une base de données orientée document répond certes aux besoins de rapidité dans l'accès aux données ainsi qu'à la simplicité des valeurs récupérées mais est limitée dans la récupération des potentielles relations entre chaque valeur. Le choix d'une base de données orientée document telle que MongoDB n'aurait été que partiellement justifié.

Une base de données orientée Graphe telle que Neo4j répond à notre besoin principal : celui de pouvoir gérer un réseau et les relations que celui-ci peut engendrer. Les données peuvent être fortement connectées et nous pouvons permettre aux utilisateurs d'avoir des recommandations en fonction de leurs compétences ou de leurs relations avec les autres utilisateurs. La visualisation de la base de données de type graphe permet également de faciliter la représentation du monde réel. Les requêtes de type « chemin le plus court », fortement utilisées dans les applications gérant un réseau, sont simplifiées et permettent de calculer, par exemple, le nombre de nœud qui nous sépare d'un autre nœud.

Explication des requêtes et justifications

Trouver le chemin le plus court

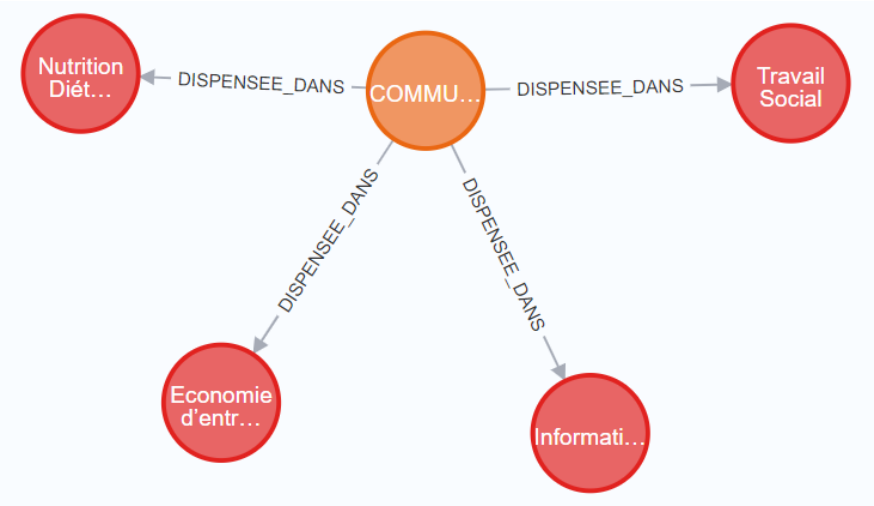


La requête « trouver le chemin le plus court » a pour objectif de permettre à un étudiant X (ici le premier nœud vert lié par la relation « **ÉTUDIE** » à la filière Sage-femme), étudiant en HEDS qui souhaite développer une application liée au domaine de la santé, de contacter une personne (un étudiant, un assistant ou professeur) en informatique de gestion qui pourrait l'aider à réaliser son projet.

Nous avons utilisé la fonction que Neo4j ' « shortestPath » ou « apoc.path.element », les requêtes qui nécessitent de trouver le plus court chemin entre 2 nœuds de départ et d'arrivée qui sont passé en paramètre. L' « apoc.path.elements » est une fonction de l'extension APOC (« Awesome Procedures On Cypher ») de Neo4j qui permet de décomposer un chemin en une liste de nœuds et de relations. Ici, cette fonction nous a permis de décomposer le chemin entre l'étudiant qui est à la HEDS et une personne se trouvant dans la filière Informatique de Gestion.

Cette requête de calcul de chemin illustre l'intérêt d'utiliser une base de données de type Graph car, c'est le seul type de base de données qui permet de retourner ce type de résultat sans complexité.

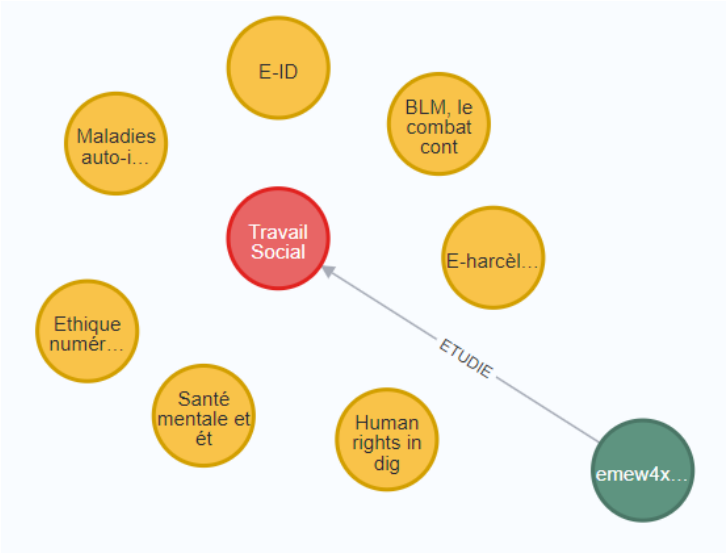
Requête de calcul



Cette requête de calcul permet aux différentes HES de planifier pour l'année 2023 les différentes thématiques d'événements qui pourraient intéresser le plus d'étudiants. Afin de mieux cibler les étudiants ainsi que leurs centres d'intérêts (d'un point de vue des études), ils ont besoin de connaître le nombre d'étudiants par filière ainsi que la compétence dispensée au plus grand nombre entre eux.

Afin de pouvoir retourner un résultat concluant à cette requête de calcul, la relation **ÉTUDIE** est parcouru. Le choix de filière (et donc des compétences liées) des étudiants pouvant évoluer, utiliser la base de données neo4j pour ce type de requête permet d'augmenter et d'adapter très rapidement le résultat en fonction des données existantes.

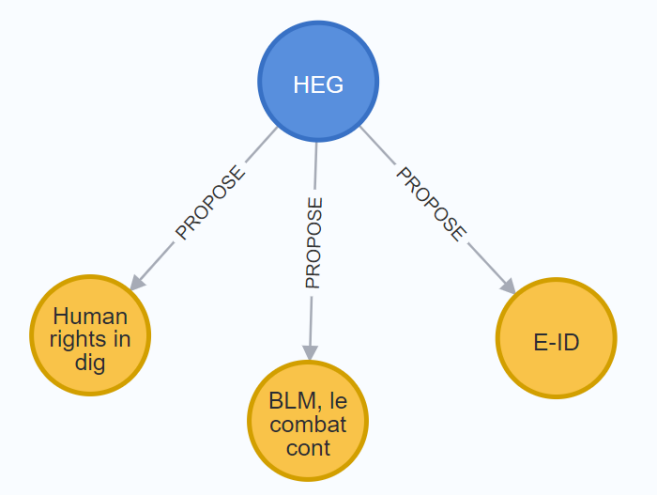
Requête sur un événement



L'application HESConnect dispose d'un algorithme de recommandation qui permet aux utilisateurs de prendre connaissance des événements futurs en fonction de leur domaine de compétence. Une comparaison des mots utilisés dans la liste des thématiques de l'événement et dans les nœuds de compétences permet de « proposer » à l'utilisateur un ou plusieurs événements qui conviendrait à son profil. Il arrive que, parfois, aucun événement ne soit proposé.

Cette troisième requête illustre l'agilité que l'utilisation de la base de données Neo4j nous permet d'obtenir dans la recommandation d'événement en fonction d'un réseau donné.

Barre de recherche

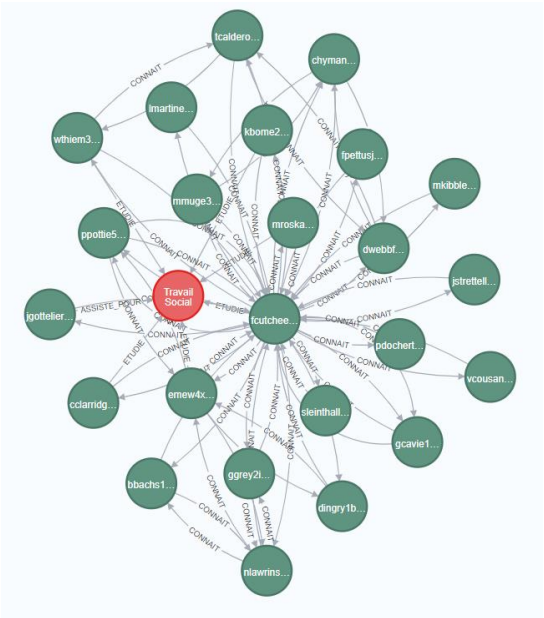


Cette requête nous permet d'obtenir tous les événements que les différentes Haute Ecole Spécialisée proposent suivant le thème que l'utilisateur de l'application passera en paramètre. Le nœud Événement a comme attribut son nom ainsi qu'une liste de thématiques. Si l'utilisateur entre une thématique inconnue, une exception est levée, indiquant qu'il n'y a aucun événement en lien avec la thématique choisie. Ici la complexité de cette requête se trouve dans le fait que l'HES qui était liée aux événements était inconnu.

Pour la réaliser en SQL, nous aurions eu besoin de faire une requête utilisant l'instruction « SELECT » avec plusieurs jointures et ajouter une condition WHERE.

Pour utiliser cette requête : il est nécessaire d'entrer une thématique dans la console à l'endroit spécifié.

Le plus de followers :



Cette requête permet de trouver la PERSONNE qui a le plus de connaissances (relations **CONNAIT**). Une fois la PERSONNE filtrée, nous analysons quelle est la filière à laquelle elle est affiliée.

Bien que cette requête ne soit pas la plus complexe, elle a été une des premières que nous avons réalisées et il nous semblait pertinent de l'inclure à ce dossier pour vous permettre d'avoir une vue d'ensemble des relations existantes. Elle permet d'obtenir la vision d'ensemble d'une base de données orienté graph avec tous ces nœuds qui sont relié avec différents nœuds.

Conclusion

La création de ce projet nous a permis de découvrir de manière approfondi les avantages de l'utilisation d'une base de données NoSQL ainsi que ses conditions d'utilisation. Les requêtes effectuées auraient été beaucoup plus complexes voir infaisable en SQL.

Les requêtes s'exécutent au chargement de la base de donnée, il est possible de les exécuter en commentant dans la classe « Applic » la classe Bdd().