Bases de Données Spécialisées

Projet : implémentation d'une BD graphe dans Neo4j ou étude des langages théoriques

Vous traiterez au choix l'un des deux sujets. Il est également possible de proposer un autre sujet (dans ce cas, contactez moi par mail amelie@irif.fr). L'implémentation d'une BD graphe est à faire de préférence en binôme, le projet théorique est à faire seul (me contacter si vous souhaitez le faire en binôme). Dans le premier cas, vous soumettrez une archive ou un lien vers vos fichiers sources (e.g. csv, requêtes Cypher), ainsi qu'un rapport en anglais ou français au format pdf décrivant votre travail. Dans le second cas, vous fournirez un pdf seulement. La date limite pour soumettre ces documents sur moodle est le 3 janvier. Un rapide oral en distanciel - sous forme de questions de ma part - sera organisé début janvier. Vous êtes encouragés à discuter entre vous et à me poser des questions, par exemple sur discord. En revanche si vous communiquez entre groupes différents, veillez à ne pas me rendre des travaux trop similaires (utilisez par exemple des jeux de données différents).

1. Implémentation d'une BD graphe dans Neo4j

1.1. Choix et import d'un jeu de données

Vous choisirez un jeu de données dans l'open data (vous pouvez par exemple en sélectionner un sur https://www.kaggle.com/datasets). Ensuite, vous implémenterez une base de données relationnelle dans PostgreSQL, ainsi qu'une base de données graphe dans Neo4j. N'oubliez pas de définir soigneusement vos contraintes d'intégrité et index (dont éventuellement des index full-text de type lucene), ainsi que d'expliciter votre stratégie d'import de données dans le rapport, qui devra également expliquer vos choix de modélisation.

1.2. Requêtes

Vous proposerez une dizaine de requêtes Cypher interrogeant votre base de données Neo4j. Incluez dans votre rapport quelques plans d'exécution, avec et sans index. Qu'en concluez-vous ? De même proposez au moins une requête SQL récursive sur votre BD relationnelle, dont vous comparerez l'efficacité avec celle d'une requête Cypher équivalente sur votre BD graphe. Essayez également de trouver une requête SQL plus efficace qu'une requête Cypher, à expressivité équivalente.

1.3. Analytique de graphe

Faites tourner quelques algorithmes de la Graph Data Science Library. Expliquez vos choix (pourquoi tel algorithme est-il particulièrement adapté à vos données ?). Utilisez des pro-

jections nommées. Réutilisez-en au moins une plusieurs fois. Essayez plusieurs modes (pas seulement stream). Utilisez éventuellement bloom pour visualiser l'analyse de vos données. Quelles connaissances pouvez-vous inférer de vos données à partir de cette analyse ?

1.4. Bonus

Vous pouvez également choisir d'inclure à votre projet tout autre aspect que vous jugerez pertinent (e.g., implémentation alternative avec Apache TinkerPop et comparaison de l'expressivité de Gremlin et Cypher, analyse de l'expressivité de vos requêtes dans des langages théoriques de type CRPQ et extensions, import de données RDF via neosemantics, etc...). Aucune prise d'initiative de ce type ne vous pénalisera, bien au contraire. Soyez curieux.

2. Questions théoriques

En théorie des bases de données, on modélise les requêtes SQL via des extensions du calcul relationnel et de l'algèbre relationnelle, ce qui permet d'en analyser la complexité (on s'intéresse classiquement à la complexité de l'évaluation, mais également à celle - d'habitude plus élevée - de l'implication). Les langages de requête pour les bases de données graphe peuvent être approchés de la même manière. A ce sujet vous pouvez consulter la référence suivante (il existe également d'autres surveys, que je pourrai vous indiquer si vous le souhaitez) :

- Tutorial de Wim *Martens Graph Data Management* à l'epit 2019 (`https://conferences.cirm-math.fr/1934.html`) :
  - slides `https://www.cirm-math.fr/RepOrga/1934/Slides/martens-graphs.pdf`
  - vidéo 1 de l'exposé `https://www.youtube.com/watch?v=9Ms4T6AGhuU`
  - vidéo 2 de l'exposé `https://www.youtube.com/watch?v=Vnp-7cq3F80`

Les différents sujets proposés ci dessous partent de la lecture d'un ou deux articles. Je vous demande de faire une rapide synthèse des articles en question et de lister quelques directions nouvelles de recherche ouvertes par ces articles, à la lumière de votre connaissance du langage Cypher. Je n'attends évidemment pas que vous traitiez tous les sujets, en traiter un seul demandera déjà beaucoup de travail. Chacun de ces sujets (en particulier le 2.1, qui est probablement le plus exigeant) peut être prolongé par un stage de recherche avec Cristina et moi.

2.1. Langages de requêtes avec données et *trail semantics*

On parle au sujet de la sémantique des chemins sous jacente dans Cypher de *trail semantics* (ou bien de *relationship homomorphism*, c.f. Section 1.5 p7 du manuel de Cypher https://neo4j.com/docs/pdf/neo4j-cypher-manual-4.2.pdf) : dans un chemin, chaque arrête est visitée au plus une fois. Or, les langages de requête pour les bases de données graphe

n'ont été analysés que récemment sous cette hypothèse sémantique (https://www.theoinf.uni-bayreuth.de/pool/documents/Paper2016-20/Paper2019/Dichotomies_for_Evaluating_Simple_Regular_Path_Queries-preprint.pdf, article également présenté dans les vidéos ci dessus). En revanche, cette analyse a été faite pour des langages qui parlent uniquement de la topologie des données, pas pour ceux qui parlent également des données. Un exemple de requête Cypher parlant à la fois de la topologie et des données du graphe est par exemple la requête vue en TP retournant tous les vols avec correspondance entre deux villes assurés sur tout le parcours uniquement par la compagnie air france (et aucune autre). Je vous demanderai de lire et synthétiser brièvement l'article cité ci dessus, ainsi que l'article https://homepages.inf.ed.ac.uk/libkin/papers/jacm2016.pdf. L'article https://arxiv.org/abs/1507.07911 pourrait également être pertinent. Je souhaiterais que vous apportiez quelques éléments de réponse initiaux à la question suivante : quels résultats connus concernant les langages de requêtes sur les graphes avec données vous semblent pouvoir s'appliquer également dans le contexte de la trail semantics ? Pouvez vous identifier un ou plusieurs exemple de preuves qui ne fonctionneraient plus avec ce changement de sémantique ? Si oui, avez vous une idée de stratégie alternative ? A votre avis la complexité du problème est-elle différente ?

2.2. Analytique de graphe

Je vous demanderai de lire et synthétiser http://aidanhogan.com/docs/queralytics.pdf (ici le modèle sous jacent est RDF, que vous étudierez avec Cristina), puis de discuter l'adaptation possible de cette approche aux graphes de propriété. Il s'agirait d'étendre Cypher de manière similaire pour faire de l'analytique de graphe directement dans le langage, toute piste ou idée est la bienvenue.

2.3. Cypher versus G-CORE

Le projet de standardisation https://www.gqlstandards.org/ a été mentionné rapidement en cours. L'article https://g-core.org/wp/wp-content/uploads/2020/05/sigmod2018.pdf est central dans ce projet (https://www.gqlstandards.org/existing-languages). Je vous demanderai de synthétiser cet article et de compléter ses analyses à la lumière de votre connaissance de Cypher. Vous pouvez en particulier proposer d'autres exemples de requête afin de compléter ceux donnés dans l'article. Vous pouvez également évoquer le cas de Sparql ou Gremlin et réfléchir à des extensions possibles du pouvoir expressif de Cypher.

2.4. Sémantique formelle de Cypher

Il existe deux articles posant les fondements d'une sémantique formelle pour Cypher (https://homepages.inf.ed.ac.uk/libkin/papers/sigmod18.pdf et http://www.vldb.org/pvldb/vol12/p2242-green.pdf). Je vous demanderai de synthétiser ces deux articles et à la lumière de votre connaissance de Cypher, de m'expliquer quelles sont maintenant les prochaines étapes.