Bases de Données Spécialisées

Projet : comparaison Cypher 5 et Cypher 25

Le projet est à faire en binôme. 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. La date limite pour soumettre ces documents sur moodle est le 11 janvier (négociable, sachant que je dois rendre les notes de l'UE le 26 février). Un rapide oral en présentiel ou distanciel - sous forme de questions de ma part - pourra être organisé dans le courant du mois de janvier.
Vous êtes encouragés à discuter entre vous et à me poser des questions. En revanche si vous communiquez entre groupes différents, veillez à ne pas me rendre des travaux trop similaires (utilisez en particulier des jeux de données différents).
Il est également possible de proposer un autre sujet ou de demander à travailler sur un problème plus théorique (dans ce cas, contactez moi par mail amelie@irif.fr).

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) ou bien vous récupérerez des données directement sur le web (par exemple en utilisant les librairies de la Python Data Science Stack). Veillez à ce qu'il s'agisse de données *connectées*, pour lesquelles une implémentation graphe est pertinente. Vous implémenterez vos données dans une base de données relationnelle PostgreSQL, ainsi que dans une base de données graphe Neo4j. Veillez à ce que des propriétés numériques soient associés à certaines des arrêtes de votre graphe (e.g., coût, distance, date, etc.), de façon à pouvoir travailler sur l'implémentation d'algorithmes de graphe variés. N'oubliez pas de définir soigneusement vos contraintes d'intégrité et index, ainsi que d'expliciter votre stratégie d'import de données dans le rapport, qui devra également expliquer vos éventuels choix de modélisation (relisez bien les conseils de modélisation donnés en cours, il n'est d'ailleurs pas interdit de s'y référer dans le rapport !). Si vous avez dû nettoyer vos données, expliquez également comment vous avez procédé.

2. Requêtes

L'un des objectifs de ce projet consiste à comparer Cypher 5 avec Cypher 25. Vous commencerez par examiner les requêtes étudiées dans cet article, qui porte sur Cypher 5 et ses versions antérieures. En vous inspirant de cet article et celui-ci, vous essaierez alors de proposer de nouvelles manières d'écrire dans Cypher 25, les requêtes Cypher 5 pointées comme problématiques dans le premier article. Vous testerez également l'évaluation de versions SQL

de certaines de ces requêtes sur votre jeu de données relationnelles. Lorsque l'évaluation ne termine pas sur votre jeu de données, vous essaierez de déterminer un sous-ensemble de votre jeu de données sur lequel elle termine (sous-graphe ou sous-ensemble des tuples). Vous êtes invités à étudier tout problème de graphe qui n'aurait pas été considéré dans le premier article, mais qui vous semble intéressant.
Incluez dans votre rapport quelques plans d'exécution. Qu'en concluez-vous ? Comparez en particulier l'évaluation de requêtes Cypher 5 et 25 a priori équivalentes mais utilisant des syntaxes différentes (d'autant plus intéressant si l'une s'évalue bien et l'autre non). Pour chaque requête considérée, essayez de proposez une requête SQL équivalente : vous comparerez l'efficacité de l'évaluation des requêtes par postgres et neo4j. En particulier, proposez au moins une requête SQL récursive sur votre BD relationnelle, dont vous comparerez l'efficacité avec celle d'une requête Cypher 5 ou 25 équivalente sur votre BD graphe. Essayez également de trouver une requête SQL plus efficace qu'une requête Cypher, à expressivité équivalente. Essayez d'expliquer vos résultats.
J'attends au moins les comparaisons suivantes :

- Pour une certaine propriété d'arête (par exemple, timestamp, âge, poids, etc), considérez une requête filtrant des chemins tels que la valeur de cette propriété soit croissante sur les arêtes du chemin. Utilisez un filtre négatif sur certains types de patterns (`NOT EXISTS`) pour exprimer la requête dans Cypher 5 et utilisez `allReduce` pour l'exprimer dans Cypher 25.

- Considérez au moins une requête Cypher 25 utilisant des quantified graph patterns qu'il vous semblerait difficile d'écrire sans cette fonctionnalité (proposez si possible une écriture sans).

- Vous considérez un ensemble de requêtes calculant des plus courts chemins, non pondérés et pondérés, vous testerez et comparerez l'ensemble des variantes disponibles dans Cypher 5, Cypher 25 et GDS (pour GDS, il vous faudra donc en particulier créer une projection de votre graphe transactionnel sur laquelle vous pourrez ensuite lancer vos algorithmes). Quels sont les algorithmes utilisés par les plans de neo4j ? (Commentez en particulier la différence entre BFS unidirectionnel et bidirectionnel.)

- Essayez d'implémenter directement dans Cypher 25 certains des algorithmes de graphe disponibles dans GDS. Comparez les deux approches.

Bonus : considérez d'autres SGBDs, par exemple Memgraph qui implémente Cypher ou bien DuckDB, qui implémente SQL PGQ.