1. Projection de base

Nous avons commencé par projeter un graphe contenant les nœuds `User` et `Movie`, ainsi que les relations `RATED` :

```
CALL gds.graph.project(
'base-graph',
['User', 'Movie'],
['RATED']
);
```

2. Liste des graphes projetés

Pour vérifier les graphes projetés, nous avons utilisé la commande suivante :

```
CALL gds.graph.list();
```

3. Calcul du degré avec l'algorithme `degree`

Nous avons appliqué l'algorithme `degree` pour calculer le nombre de connexions de chaque nœud dans le graphe projeté :

```
CALL gds.degree.stream('base-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS movieTitle, score AS degree
ORDER BY degree DESCENDING
LIMIT 10;
```

Observation :
Les résultats ont montré un degré de 0 pour les films, car la direction des relations dans la base de données (User $\rightarrow$ Movie) empêchait le calcul correct du degré.

Étape 2 : Modification de l'orientation des relations

Nous avons ensuite inversé l'orientation des relations. Cela a été réalisé en projetant un graphe où la relation `RATED` était orientée en `REVERSE` (inversée) :

```
CALL gds.graph.drop('base-graph', false);

CALL gds.graph.project(
'reverse-graph',
['User', 'Movie'],
{RATED_BY: {type: 'RATED', orientation: 'REVERSE'}}
);
```

4. Calcul du degré avec l'algorithme `degree` sur le graphe inversé

Après avoir inversé l'orientation des relations, nous avons utilisé l'algorithme `degree` pour calculer combien de fois chaque film a été noté. Cela permet d'analyser les films du point de vue des utilisateurs.

```
CALL gds.degree.stream('reverse-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS movieTitle, score AS ratingCount
ORDER BY ratingCount DESCENDING
LIMIT 10;
```

Étape 3 : Relations non orientées

Nous avons projeté un graphe où les relations `RATED` sont non orientées :

```
CALL gds.graph.drop('reverse-graph', false);

CALL gds.graph.project(
'non-orientated-graph',
['User', 'Movie'],
{RATED: {type: 'RATED', orientation: 'UNDIRECTED'}}
);
```

Calcul du degré avec `degree` pour les relations non orientées :

```
CALL gds.degree.stream('non-orientated-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS movieTitle, score AS degree
ORDER BY degree DESCENDING
LIMIT 10;
```

Étape 4 : Analyse avancée

Nous avons ajouté une propriété `weight` aux relations `RATED` pour effectuer une analyse de degré pondéré :

```
CALL gds.graph.drop('non-orientated-graph', false);

CALL gds.graph.project(
'weighted-graph',
['User', 'Movie'],
{RATED: {type: 'RATED', properties: ['weight']}}
);
```

Calcul du degré pondéré avec `degree` :
```
CALL gds.degree.stream('weighted-graph', {relationshipWeightProperty: 'weight'})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS movieTitle, score AS weightedDegree
ORDER BY weightedDegree DESCENDING
LIMIT 10;
```

Questions analytiques

1. Quel type d'orientation (naturelle, inversée ou non orientée) fournit les informations les plus pertinentes pour analyser les films les plus notés ?
- L'orientation naturelle (User → Movie) est la plus pertinente pour analyser la popularité des films en fonction des notes des utilisateurs.
- L'orientation inversée (Movie → User) est utile pour analyser combien de fois un film a été noté.
- Les relations non orientées sont moins spécifiques et peuvent être utilisées pour une analyse plus globale.

2. Quels sont les avantages d'utiliser des relations non orientées ou inversées dans des analyses spécifiques ?
- Relations non orientées : Idéales pour des analyses générales où la direction des relations n'a pas d'importance.
- Relations inversées : Utiles pour analyser les films du point de vue des utilisateurs, afin de voir combien de fois un film a été noté.

3. Proposez une situation où l'ajout d'une propriété pondérée pourrait améliorer les résultats d'analyse.
- L'ajout d'une propriété pondérée permet d'accorder plus de poids aux relations importantes, comme des notes élevées, ce qui améliore l'analyse en identifiant les films les plus appréciés, plutôt que de simplement compter le nombre de notes.

Conclusion

Ce TP a permis d'explorer différents types de projections dans Neo4j GDS et d'observer les effets de l'orientation des relations sur les analyses. L'ajout de propriétés pondérées permet de rendre les analyses plus significatives en tenant compte de l'intensité des relations.