TP : Exploration des Projections Natives et Orientation des Relations dans Neo4j GDS

Objectif :
Appliquer les concepts de projections natives et explorer les impacts de l'orientation des relations sur les analyses graphiques.

Consignes :
Étape 1 : Création d'une projection de base

1. Projetez un graphe contenant les nœuds `User` et `Movie` ainsi que les relations `RATED`.
2. Listez les graphes projetés pour vérifier la projection.
3. Utilisez l'algorithme `degree` pour calculer le nombre de connexions de chaque nœud, et affichez les résultats pour les nœuds `Movie`.

Étape 2 : Modification de l'orientation des relations

1. Projetez un graphe où la relation `RATED` est inversée (`RATED_BY`).
2. Réutilisez l'algorithme `degree` pour calculer combien de fois chaque film a été noté.
3. Comparez les résultats avec ceux obtenus lors de la projection de base.

Étape 3 : Relations non orientées

1. Projetez un graphe en spécifiant que les relations `RATED` sont non orientées.
2. Appliquez l'algorithme `degree` pour analyser les connexions dans ce graphe.
3. Affichez les résultats pour tous les nœuds.

Étape 4 : Analyse avancée

1. Ajoutez une propriété `weight` aux relations `RATED`.
2. Projetez un graphe en incluant cette propriété et utilisez-la dans un calcul de degré pondéré.
3. Comparez les résultats avec ceux des étapes précédentes.

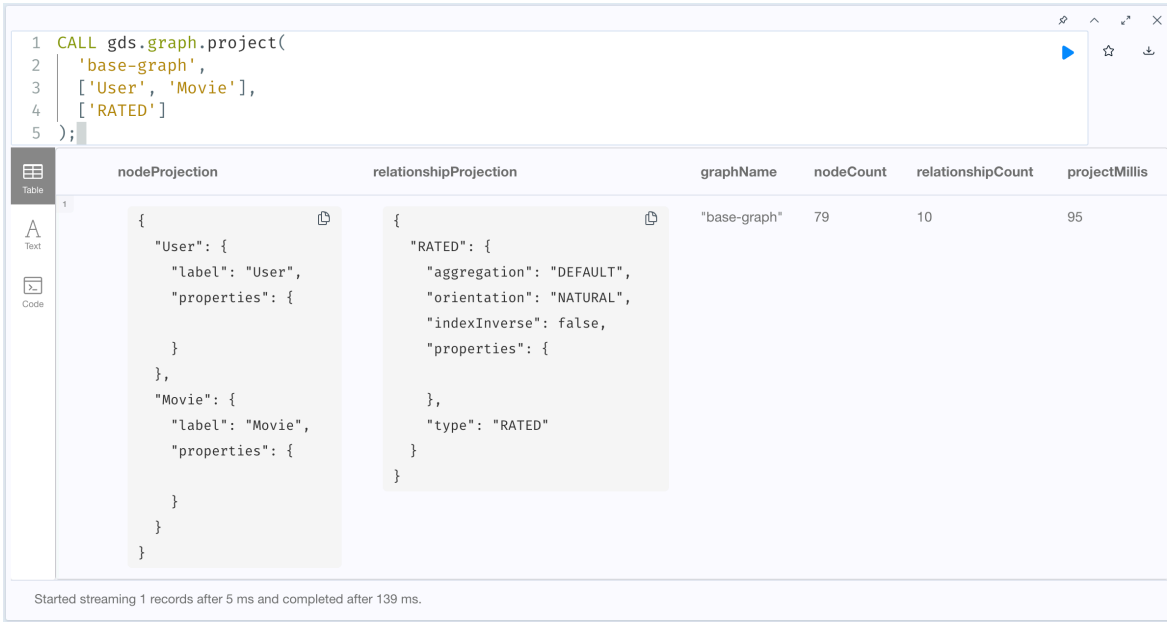
Questions analytiques :

1. Quel type d'orientation (naturelle, inversée ou non orientée) fournit les informations les plus pertinentes pour analyser les films les plus notés ?

2. Quels sont les avantages d'utiliser des relations non orientées ou inversées dans des analyses spécifiques ?
3. Proposez une situation où l'ajout d'une propriété pondérée pourrait améliorer les résultats d'analyse.

Étape 1 : Création d'une projection de base

1. Créer la projection de base :

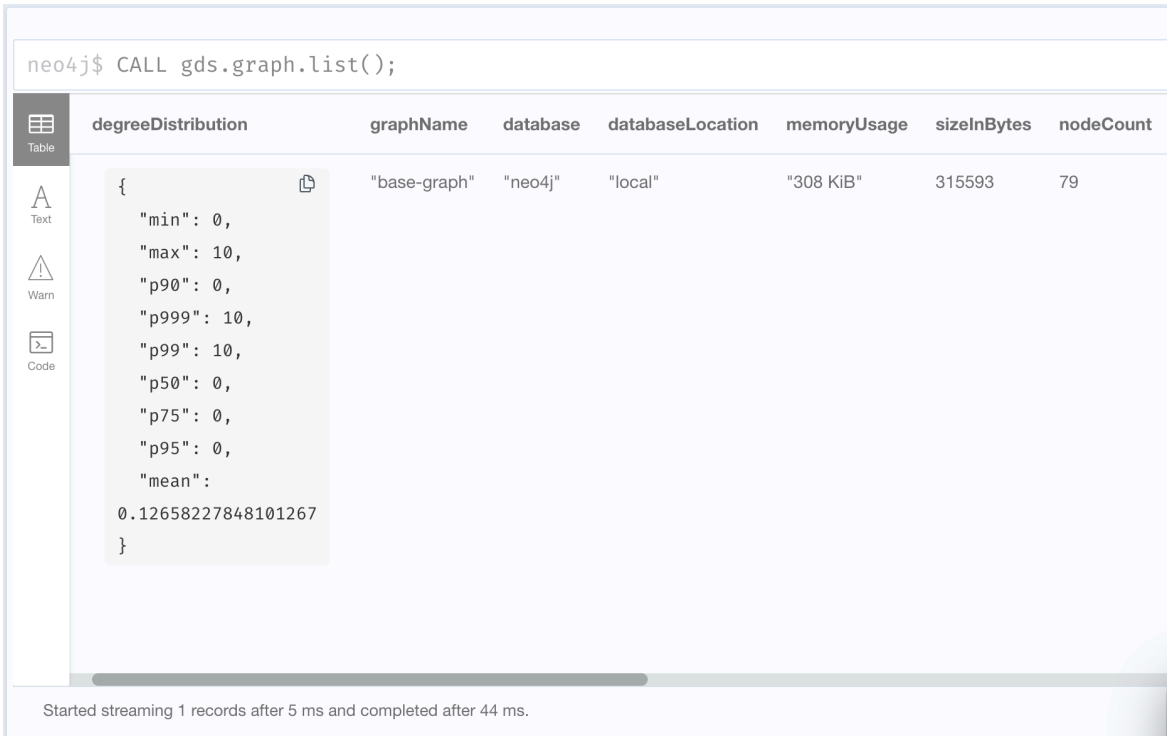


```
CALL gds.graph.project(
  'base-graph',
  ['User', 'Movie'],
  ['RATED']
);
```

nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
`{ "User": { "label": "User", "properties": { } }, "Movie": { "label": "Movie", "properties": { } } }`	`{ "RATED": { "aggregation": "DEFAULT", "orientation": "NATURAL", "indexInverse": false, "properties": { }, "type": "RATED" } }`	"base-graph"	79	10	95

Started streaming 1 records after 5 ms and completed after 139 ms.

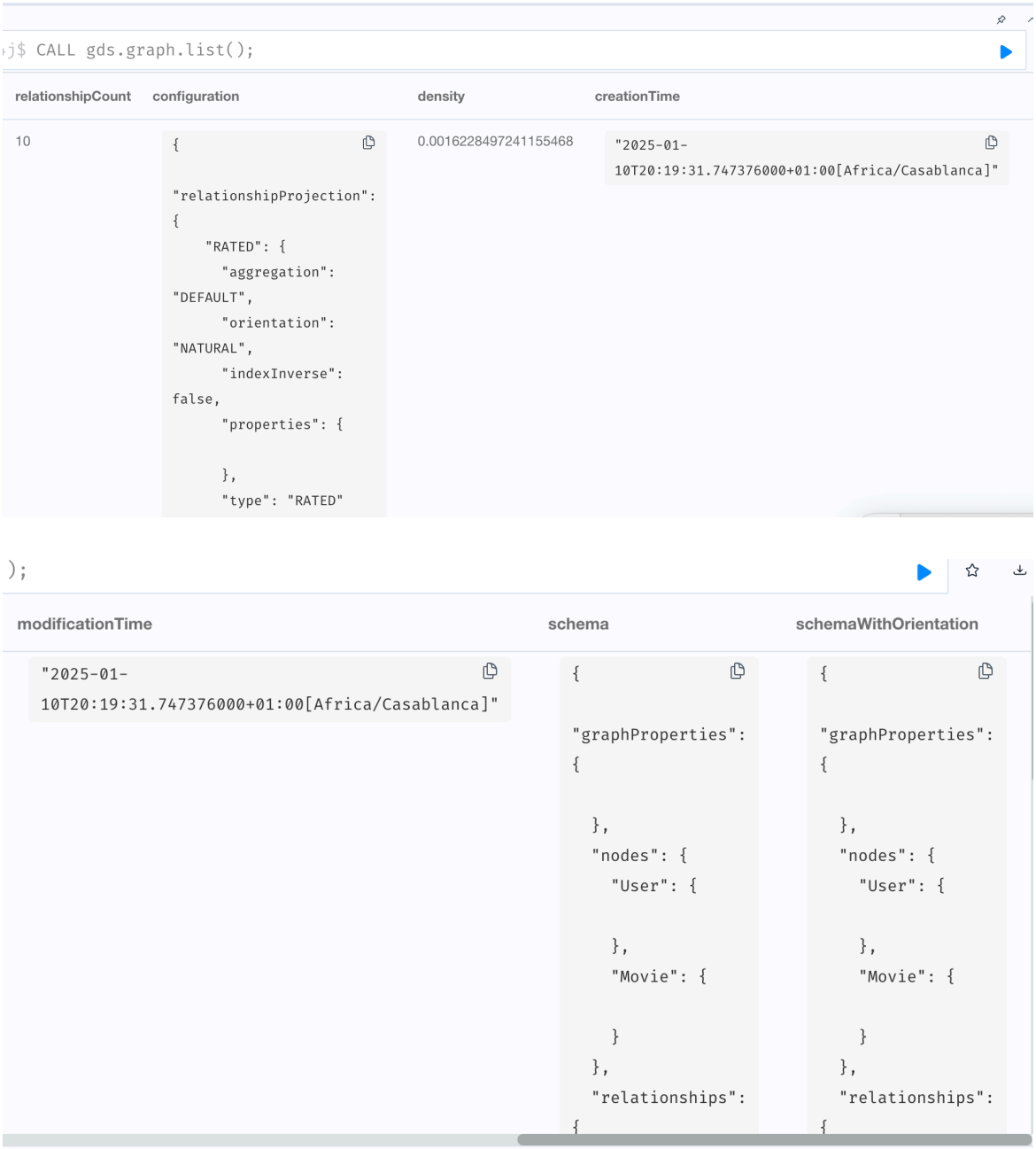
2. Lister les graphes projetés pour vérifier la création :



```
neo4j$ CALL gds.graph.list();
```

degreeDistribution	graphName	database	databaseLocation	memoryUsage	sizeInBytes	nodeCount
`{ "min": 0, "max": 10, "p90": 0, "p999": 10, "p99": 10, "p50": 0, "p75": 0, "p95": 0, "mean": 0.12658227848101267 }`	"base-graph"	"neo4j"	"local"	"308 KiB"	315593	79

Started streaming 1 records after 5 ms and completed after 44 ms.



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

relationshipCount	configuration	density	creationTime
10	{ "relationshipProjection": { "RATED": { "aggregation": "DEFAULT", "orientation": "NATURAL", "indexInverse": false, "properties": { }, "type": "RATED"	0.0016228497241155468	"2025-01-10T20:19:31.747376000+01:00[Africa/Casablanca]"

```
);
```

modificationTime	schema	schemaWithOrientation
"2025-01-10T20:19:31.747376000+01:00[Africa/Casablanca]"	{ "graphProperties": { }, "nodes": { "User": { }, "Movie": { } }, "relationships": {	{ "graphProperties": { }, "nodes": { "User": { }, "Movie": { } }, "relationships": {

3. Calculer le degré (degree) :

```
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;
```

Table | Text | Code

	movieTitle	degree
1	*null*	10.0
2	"The Dark Knight"	0.0
3	"Inception"	0.0
4	"Inception"	0.0
5	"The Dark Knight"	0.0
6	"The Matrix"	0.0
7		

Started streaming 10 records after 6 ms and completed after 31 ms.

Étape 2 : Modification de l'orientation des relations

1. Supprimer le graphe projeté précédent :

```
neo4j$ CALL gds.graph.drop('base-graph', false);
```

Table | Text | Warn | Code

	graphName	database	databaseLocation	memoryUsage	sizeInBytes	nodeCount	relationshipCount	configuration
1	"base-graph"	"neo4j"	"local"	""	-1	79	10	{ "relationshipProjection": { "RATED": { "aggregation": "DEFAULT", "orientation": "NATURAL", "indexInverse": false, "properties": { }, "type": "RATED"

Started streaming 1 records after 5 ms and completed after 13 ms.

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

density	creationTime
0.0016228497241155468	"2025-01-10T20:19:31.747376000+01:00[Africa/Casablanca]"

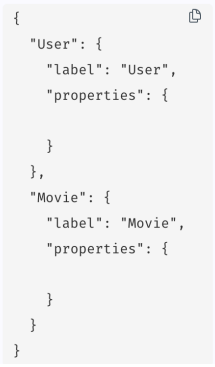
5 ms and completed after 13 ms.

mpleted after 13 ms.

2. Créer une projection avec orientation inversée :

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

Table | Text | Code

	nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
1			"reverse-graph"	79	10	11

3. Calculer le degré (degree) :

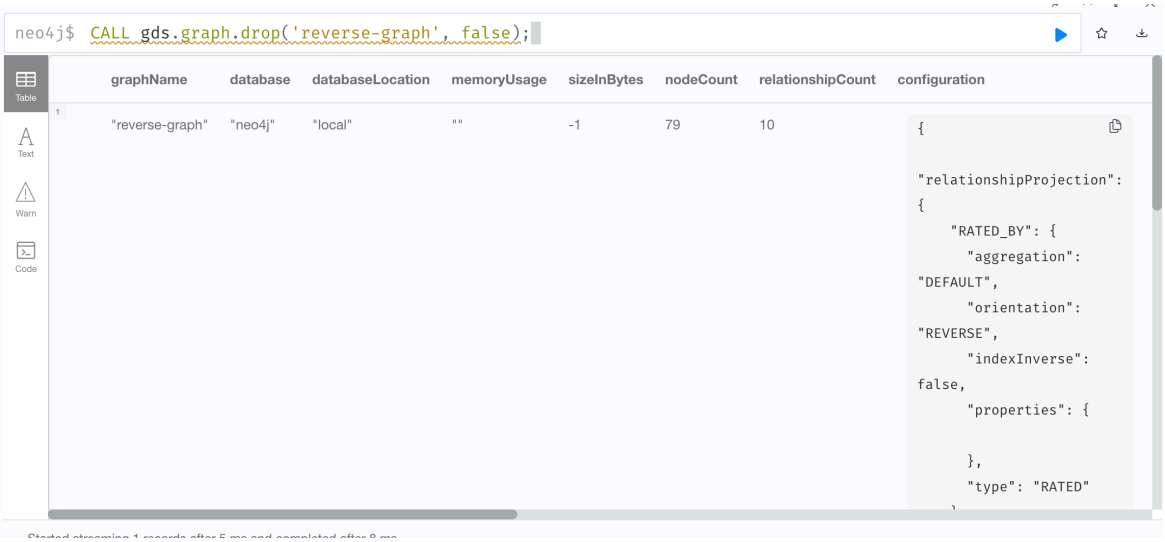
```
CALL gds.degree.stream('reverse-graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS movieTitle, score AS ratingCount
ORDER BY ratingCount DESCENDING
LIMIT 10;
```

Table | Text | Code

	movieTitle	ratingCount
1	"The Matrix"	1.0
2	"The Dark Knight"	1.0
3	"The Matrix"	1.0
4	"Inception"	1.0
5	"The Dark Knight"	1.0
6	"The Matrix"	1.0
7		

Étape 3 : Relations non orientées

1. Supprimer le graphe précédent :



2. Créer une projection avec des relations non orientées :

3. Calculer le degré (degree) :

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

	nodeTitle	degree
1	*null*	10.0
2	"The Dark Knight"	1.0
3	"Inception"	1.0
4	"Inception"	1.0
5	"The Dark Knight"	1.0
6	"The Matrix"	1.0
7		

Started streaming 10 records after 8 ms and completed after 14 ms.

Étape 4 : Analyse avancée avec pondération

1. Supprimer le graphe précédent :

```
neo4j$ CALL gds.graph.drop('undirected-graph', false);
```

graphName	database	databaseLocation	memoryUsage	sizeInBytes	nodeCount	relationshipCount	configuration	de
"undirected-graph"	"neo4j"	"local"	""	-1	79	20	{ "relationshipProjection": { "RATED": { "aggregation": "DEFAULT", "orientation": "UNDIRECTED", "indexInverse": false, "properties": { }, "type": "RATED"	0.

Started streaming 1 records after 4 ms and completed after 8 ms.

3. Calculer le degré pondéré :

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

Table | Text | Code

	nodeTitle	weightedDegree
1	*null*	10.0
2	"The Dark Knight"	0.0
3	"Inception"	0.0
4	"Inception"	0.0
5	"The Dark Knight"	0.0
6	"The Matrix"	0.0
7		

Started streaming 10 records after 1 ms and completed after 7 ms.

Questions analytiques

1. Quel type d'orientation fournit les informations les plus pertinentes pour analyser les films les plus notés ?

→ L'orientation inversée « RATED_BY »

2. Quels sont les avantages d'utiliser des relations non orientées ou inversées ?

- Les relations non orientées ont une vue globale des connexions ainsi que les relations inversées sont utiles pour des analyses spécifiques

3. Proposez une situation où l'ajout d'une propriété pondérée pourrait améliorer les résultats d'analyse.

On ajoute la propriété weight