Présentation de NEO4J

Parcours d’un graphe : Concepts
Plus court chemin entre deux nœuds
Relation directe et indirecte entre deux noeuds

Parcours d'un graphe

ANCHOR: Point de départ d'une requête: un ou plusieurs noeuds
Objectif : Minimiser le nombre de nœuds

Exemple : Graphe Movies comporte 42 nœuds Movie et 137 nœuds Person

```
MATCH (p:Person)-[:ACTED_IN]->(m)
RETURN p.name, m.title LIMIT 100   => Anchor = Noeud Person

MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
RETURN p.name, m.title LIMIT 100 ==> Anchor = Noeud Movie

ROFILE MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE p.name = 'Clint Eastwood'
RETURN p.name, m.title  => Noeud Person/ « Clint Eastwood »

MATCH (p1:Person)-[:ACTED_IN]->(m1)
MATCH (m2)<-[:ACTED_IN]-(p2:Person)
WHERE p1.name = 'Tom Hanks'AND p2.name = 'Meg Ryan'AND m1 = m2
RETURN m1.title  => 2 Noeuds Person « Tom Hanks » et « Meg Ryan »
```

Parcours d’un graphe

PATH: Relations qui partent du (des) noeud(s) vu précédemment (anchor)

Objectif : Eliminer le parcours de relations inutiles

```
MATCH (m:Movie)<--(p:Person)
WHERE p.name = 'Tom Hanks'
RETURN  m.title   => 2 relations (DIRECTED, ACTED_IN) et 13 nœuds

MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
WHERE p.name = 'Tom Hanks'
RETURN  m.title   => 1 relation et 1 nœud
```

Parcours d'un graphe

Exemple :

```
MATCH (m:Movie)<-[r:ACTED_IN]-(p:Person)
WHERE p.name = 'Tom Cruise'
RETURN  m.title  => 3 relations , 3 nœuds (Films)
```

m.title
"A Few Good Men"
"Top Gun"
"Jerry Maguire"

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie),
(coActors:Person)-[:ACTED_IN]->(m)
WHERE p.name = 'Tom Cruise'
RETURN m.title AS movie ,collect(coActors.name) AS coActors
=>
3 relations, 3 nœuds (Films)
+ 11 nœuds issus de A few Good Men
+  5 nœuds issus de Top Gun
+  8 nœuds issus de Jerry MAguire
```

movie	coActors
"A Few Good Men"	["Jack Nicholson", "Demi Moore", "Kevin Bacon", "Kiefer Sutherland", "Noah Wyle", "Cuba Gooding Jr.", "Kevin Pollak", "J.T. Walsh", "James Marshall", "Christopher Guest", "Aaron Sorkin"]
"Top Gun"	["Kelly McGillis", "Val Kilmer", "Anthony Edwards", "Tom Skerritt", "Meg Ryan"]
"Jerry Maguire"	["Cuba Gooding Jr.", "Renee Zellweger", "Kelly Preston", "Jerry O'Connell", "Jay Mohr", "Bonnie Hunt", "Regina King", "Jonathan Lipnicki"]

Parcours d'un graphe

Attention à l'écriture de la requête
La requête ci-dessous est proche de la précédente, cependant elle ramène Tom Cruise en plus

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE p.name = 'Tom Cruise'
MATCH (allActors:Person)-[:ACTED_IN]->(m)
RETURN m.title AS movie, collect(allActors.name) AS allActors
```

movie	allActors
"A Few Good Men"	["Tom Cruise", "Jack Nicholson", "Demi Moore", "Kevin Bacon", "Kiefer Sutherland", "Noah Wyle", "Cuba Gooding Jr.", "Kevin Pollak", "J.T. Walsh", "James Marshall", "Christopher Guest", "Aaron Sorkin"]
"Top Gun"	["Tom Cruise", "Kelly McGillis", "Val Kilmer", "Anthony Edwards", "Tom Skerritt", "Meg Ryan"]
"Jerry Maguire"	["Tom Cruise", "Cuba Gooding Jr.", "Renee Zellweger", "Kelly Preston", "Jerry O'Connell", "Jay Mohr", "Bonnie Hunt", "Regina King", "Jonathan Lipnicki"]

Type PATH

PATH est le type de données qui représente la structure du graphe
Récupérer le path d'une requête : MATCH p=requête CQL

Exemple :
Affiche le sous graphe correspondant à la requête

```
MATCH p=(:Person)-[:ACTED_IN]->(:Movie {title: "Cast Away"}) <-
[:DIRECTED]-(:Person) RETURN p;
```

Affiche les nœuds du sous graphe correspondant à la requête

```
MATCH p=(:Person)-[:ACTED_IN]->(:Movie {title: "Cast Away"}) <-[:DIRECTED]-(:Person)
RETURN nodes(p);
```

Affiche les relations du sous graphe correspondant à la requête

```
MATCH p=(:Person)-[:ACTED_IN]->(:Movie {title: "Cast Away"}) <-[:DIRECTED]-(:Person)
RETURN relationships(p);
```

Longueur du chemin (nombre de relations du sous graphe correspondant à la requête

```
MATCH p=(:Person)-[:ACTED_IN]->(:Movie {title: "Cast Away"}) <-[:DIRECTED]-(:Person)
RETURN length(p);
```

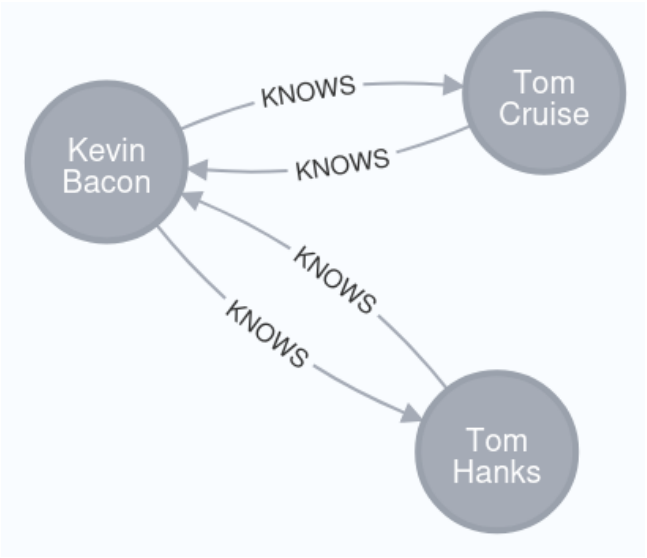
Plus cours chemin entre 2 noeuds

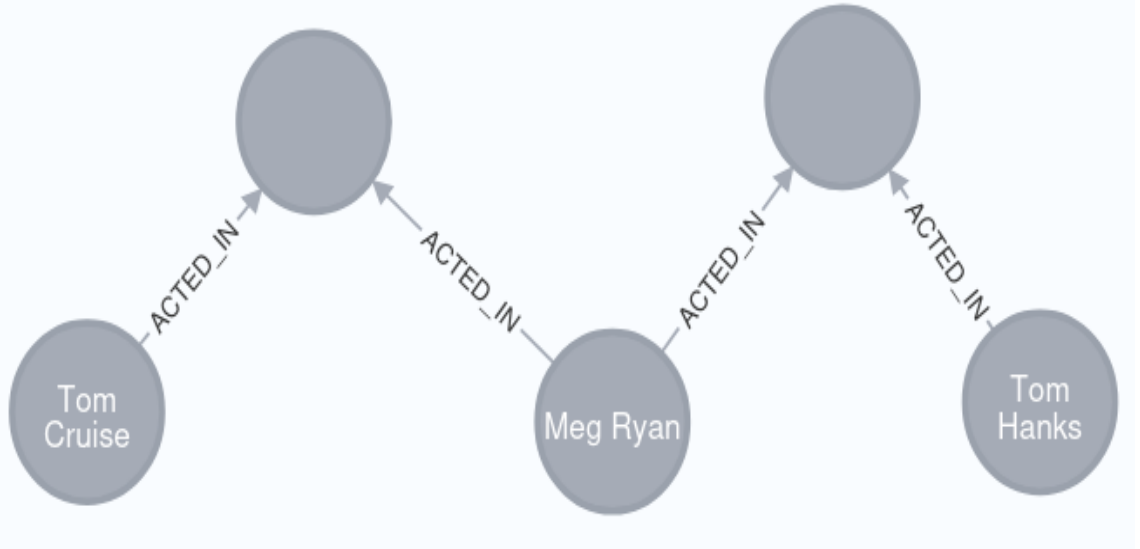
ShortestPath est une fonction disponible dans CQL

Exemple :

```
MATCH p=shortestPath((p1:Person)-[*]-(p2:Person))
WHERE p1.name = "Tom Hanks"
AND p2.name = "Tom Cruise"
RETURN p ;
```

Le résultat va dépendre des relations existantes



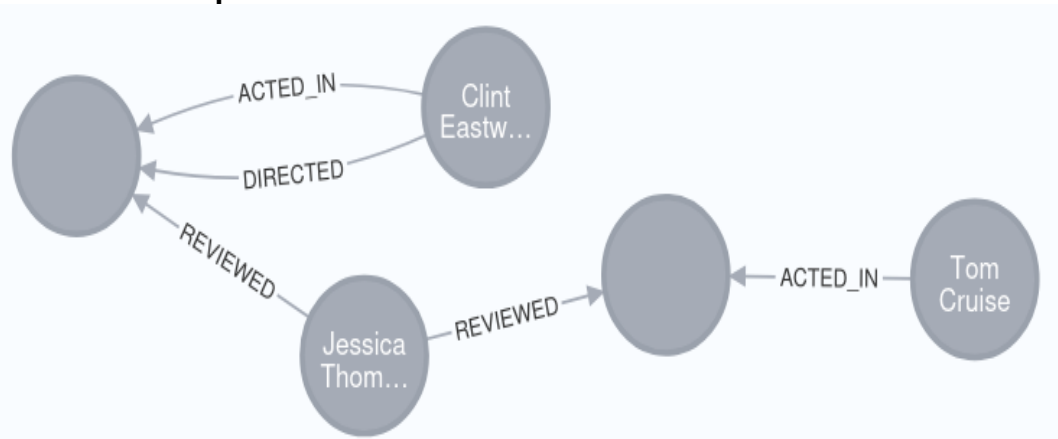


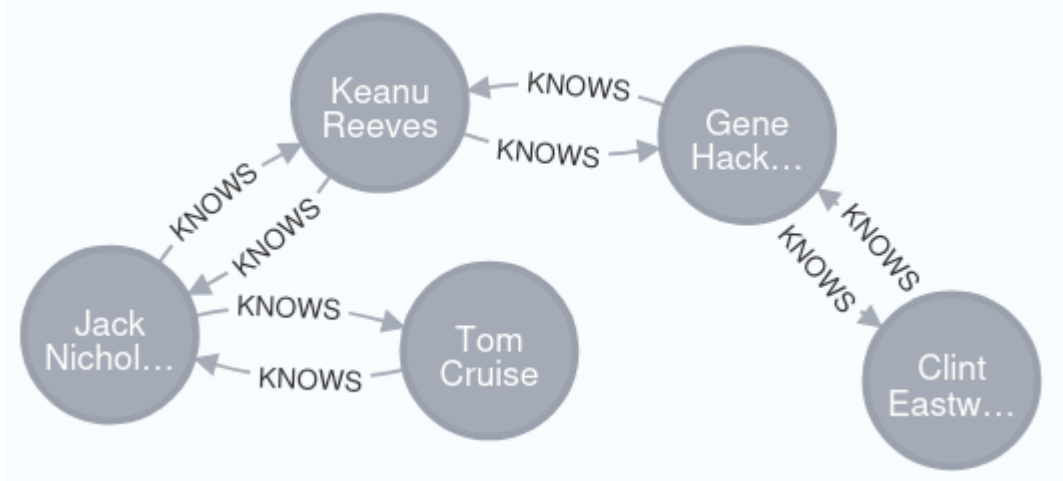
Plus cours chemin entre 2 noeuds

Filtrage sur les relations

Exemple :

```
MATCH p=shortestPath((p1:Person)-[*]-(p2:Person))
WHERE p1.name = "Clint Eastwood"
AND p2.name = "Tom Cruise"
RETURN p
```

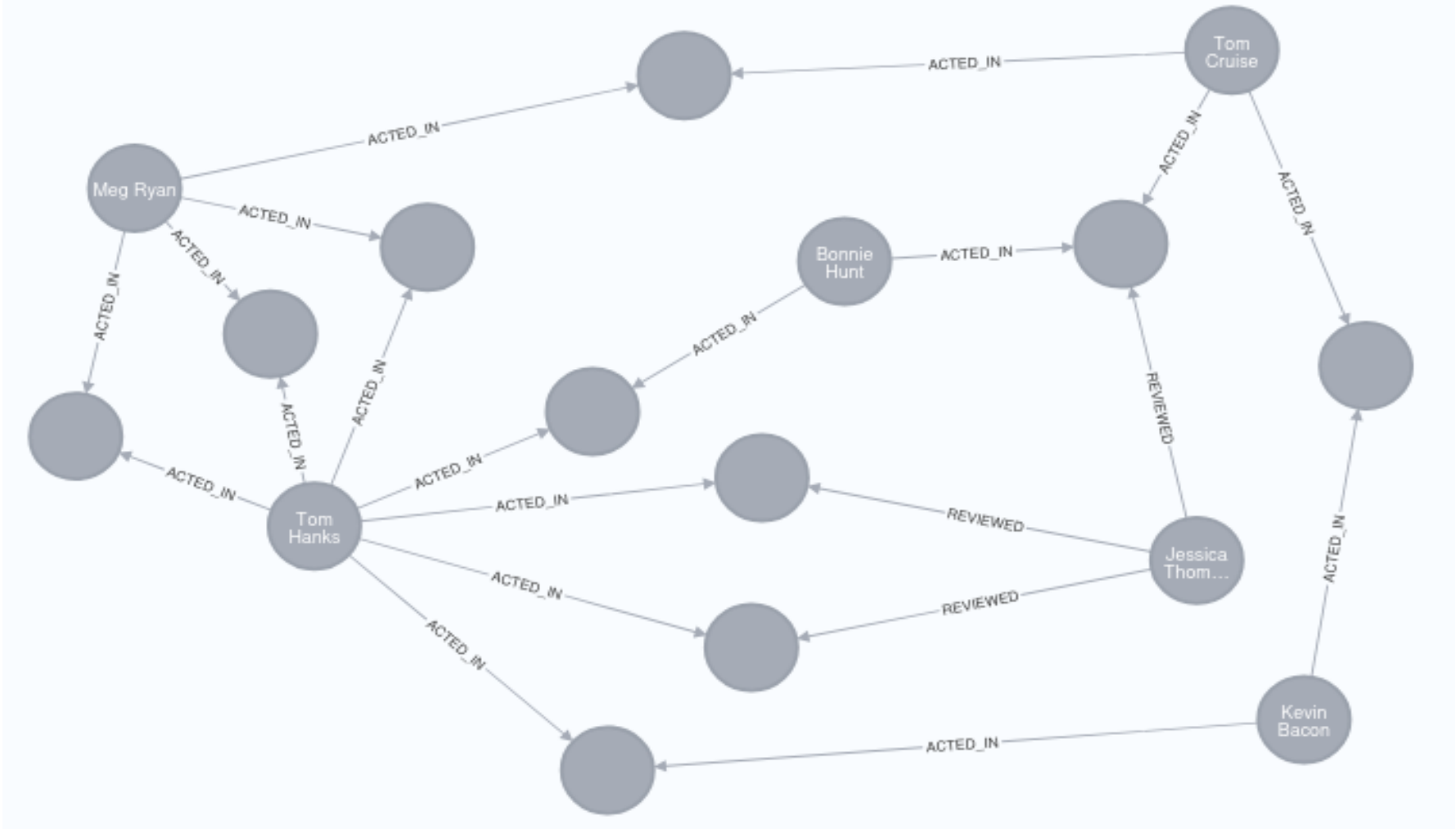





```
MATCH p=shortestPath((p1:Person)-[:ACTED_IN]-(p2:Person))
WHERE p1.name = "Clint Eastwood"
AND p2.name = "Tom Cruise"
RETURN p   => Pas de chemin trouvé
```

Plus cours chemins entre 2 noeuds

allShortestPath est une fonction disponible dans CQL



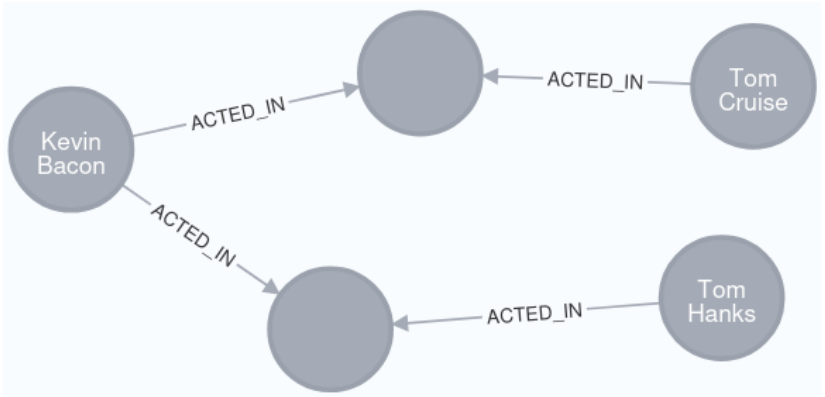
Base Graphe Neo4J

Plus cours chemins entre 2 noeuds

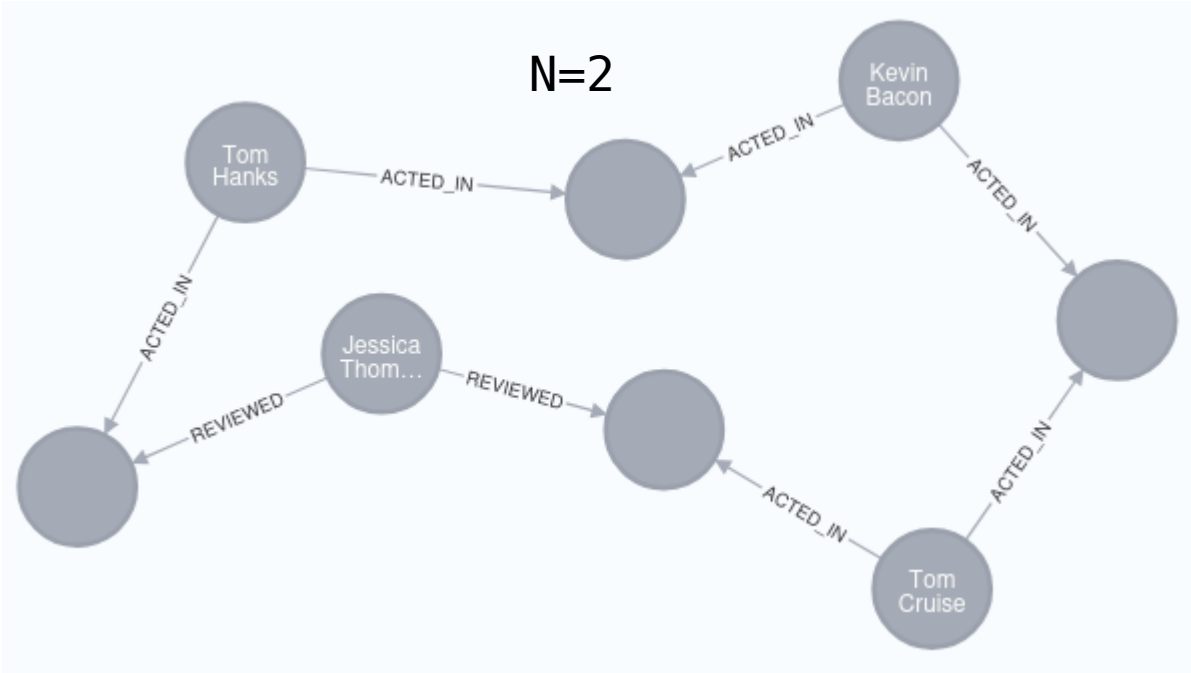
Shortest / All shortest remplace les deux fonctions précédentes pour une conformité à GQL

```
MATCH p = Shortest N (p1:Person)-[*]-(p2:Person)
WHERE p1.name = "Tom Hanks"
AND p2.name = "Tom Cruise"
RETURN  p
Avec N : le nombre de chemin recherché
```

N=1



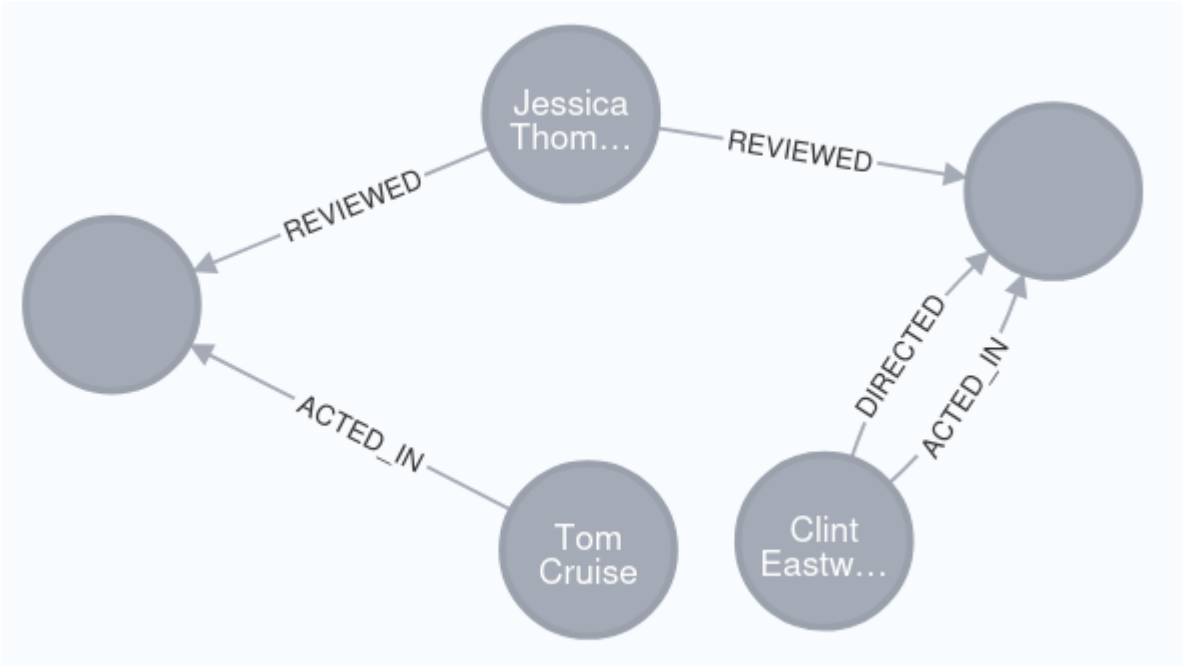
N=2



Plus cours chemins entre 2 noeuds

ANY : Permet de vérifier l'existence d'un chemin entre 2 nœuds (Equivalent à Shortest 1)

```
MATCH p = Any (p1:Person)-[*]-(p2:Person)
WHERE p1.name = "Clint Eastwood"
AND p2.name = "Tom Cruise"
RETURN  p
```



Relations directe & indirecte entre 2 noeuds

Rechercher liens directs et indirects entre deux nœuds
(« Varying Length Traversal »)

Concept: Nombre de sauts (« hop ») pour aller d'un nœud à un autre

Connexion directe entre deux nœuds

```
MATCH (p1)-[r:KNOWS*1]->(p2) :
```

p1 connaît directement p2 (1 relation traversée) (=[r:KNOWS])

Connexion indirecte entre deux nœuds

```
MATCH (n)-[r:KNOWS*2]->(p2)
```

p1 connait p2 indirectement (2 relations traversées)

```
MATCH (p1)-[r:KNOWS*1..5]->(p2) : de 1 à 5 sauts
```

p1 connaît p2 directement & indirectement (de 1 à 5 relations traversées)

```
MATCH (p1)-[r:KNOWS*1..]→(p2)
```

p1 connaît p2 directement & indirectement (de 1 à n relations traversées)