Neo4j

Insertion de données

ajout des labels

```
create (:Personne:Homme{nom:"Christoph",prenom:"Samuel",age:21})
create (:Personne:Femme{nom:"Mouche",prenom:"Valentin",age:20})
create (:Specialite{libelle:"Informatique"})
create (:Specialite{libelle:"Chimie"})
create (:EntrepriseInstitut{nom:"UBDX",adresse:"Talence"})
create (:EntrepriseInstitut{nom:"UPPA",adresse:"Pau"})
create (:EntrepriseInstitut{nom:"CapGemini",adresse:"Pau"})
create (:EntrepriseInstitut{nom:"CapGemignon",adresse:"Talence"})
```

ajout des relations

```
match (a{nom:"Mouche"})
match (b{nom:"Christoph"})
create (a)-[:EST_AMI_AVEC]->(b)

match (c{nom:"Mouche"})
match (d{nom:"Christoph"})
create (d)-[:EST_AMI_AVEC]->(c)

match (e{nom:"Mouche"})
match (f{nom:"CapGemignon"})
create (e)-[:TRAVAILLE_A]->(f)

match (g{nom:"Christoph"})
match (h{nom:"CapGemini"})
create (g)-[:TRAVAILLE_A]->(h)

match (i{nom:"Mouche"})
match (j{nom:"UBDX"})
create (i)-[:ETUDIE_A]->(j)

match (k{nom:"Christoph"})
match (l{nom:"UPPA"})
create (k)-[:ETUDIE_A]->(l)

match (m{libelle:"Chimie"})
match (n{nom:"UBDX"})
create (n)-[:A_POUR_SPECIALITE]->(m)

match (o{libelle:"Informatique"})
match (p{nom:"UPPA"})
create (p)-[:A_POUR_SPECIALITE]->(o)
```

```
create (:Personne:Homme{nom:"Penne",prenom:"Alexandre",age:70})
match (m{nom:"Penne"})
match (c{nom:"UPPA"})
create (m)-[:ETUDIE_A]->(c)
```

ajout d'un poste (nœud)

```
match (p:Personne)-[r:TRAVAILLE_A]->(e:EntrepriseInstitut)
set r.poste = "Alternant"
```

ajout d'une formation (arc)

```
match (p:Personne)-[r:ETUDIE_A]->(i:EntrepriseInstitut)
set r.formation = "Master"
```

Opérations sur les données

/* 1 */

```
match (p:Personne {nom: "Christoph"})
set p.age = 22
```

/* 2 */

```
match (p:Personne)
optional match (p)-[:TRAVAILLE_A]->(emploi)
set p.travaille = case when emploi is not null then 'oui' else 'non'
end
```

Interrogation de la base de données

/* 1 */

```
match (p:Personne)
where (p.travaille = "oui")
return p
```

/* 2 */

```
create (:Personne:Homme{nom:"Lattard",prenom:"Greg",age:3})

match (p:Personne)
match (p)-[:ETUDIE_A]->()
return p
```

/* 3 */

```
create (:Personne:Femme{nom:"Lavergne",prenom:"Sabrina",age:53})
match (g{nom:"Lavergne"})
match (h{nom:"CapGemignon"})
create (g)-[:TRAVAILLE_A]→(h)
```

```
create (:Personne:Homme{nom:"Janko",prenom:"Matt",age:54})
match (g{nom:"Janko"})
match (h{nom:"CapGemini"})
create (g)-[:TRAVAILLE_A]→(h)

match (p:Personne:Homme)
match (p)-[:TRAVAILLE_A]->()
where (p.age>49)
return p

/* 4 */
match (p:Personne)
match (b:Personne)
where (b.prenom="Bruno")
match (p)-[:EST_AMI_AVEC]->(b)
return p

create (:Personne:Homme{nom:"Kinder",prenom:"Bruno",age:4})
match (c{nom:"Kinder"})
match (d{nom:"Lattard"})
create (d)-[:EST_AMI_AVEC]->(c)

match (c{nom:"Kinder"})
match (d{nom:"Lattard"})
create (c)-[:EST_AMI_AVEC]->(d)

/* 5 */
create (:Personne:Femme{nom:"Presse",prenom:"Ilya",age:30})
match (c{nom:"Presse"})
match (d{nom:"Christoph"})
create (c)-[:EST_AMI_AVEC]→(d)

match (b:Personne {prenom:"Bruno"})-[:EST_AMI_AVEC*1..2]-(p:Personne)
where p <> b return p

/* 6 */
match (c{nom:"Janko"})
match (d{nom:"Lattard"})
create (c)-[:EST_AMI_AVEC]->(d)

match (c{nom:"Janko"})
match (d{nom:"Lattard"})
create (d)-[:EST_AMI_AVEC]→(c)

match (b:Personne {prenom:"Bruno"})-[:EST_AMI_AVEC]-(p:Personne)
match (p)-[:EST_AMI_AVEC]->(c:Personne)
where not c.prenom="Bruno"
return distinct c
```

/* 7 */

```
match (p:Personne), (q:Personne), (r:Personne)
where (r)-[:EST_AMI_AVEC]->(q)
and (p)-[:EST_AMI_AVEC]->(q)
and not ((r)-[]->(p))
and not r.prenom="Bruno"
return r
```

/* 8 */

```
create (:Personne:Femme{nom:"Wang",prenom:"Ada",age:27})
match (g{nom:"Wang"})
match (h{nom:"CapGemignon"})
create (g)-[:TRAVAILLE_A]→(h)

match (p:Personne), (e:EntrepriseInstitut)
where (e)<-[:TRAVAILLE_A]-(p)
and (p.travaille="oui")
return count(e), e.nom order by count(e)
```

/* 9 */

```
match (c{nom:"Christoph"})
match (d{nom:"Lattard"})
create (d)-[:EST_AMI_AVEC]→(c)

match (p:Personne)
where size([(p)-[:EST_AMI_AVEC]->(ami:Personne)-[:TRAVAILLE_A]->() | ami]) >= 2
return p.nom
```

/* 10 */

```
match chemin = shortestPath((d)-[*]-(s))
where d.prenom = "Bruno" and s.libelle = "Informatique"
return chemin
```