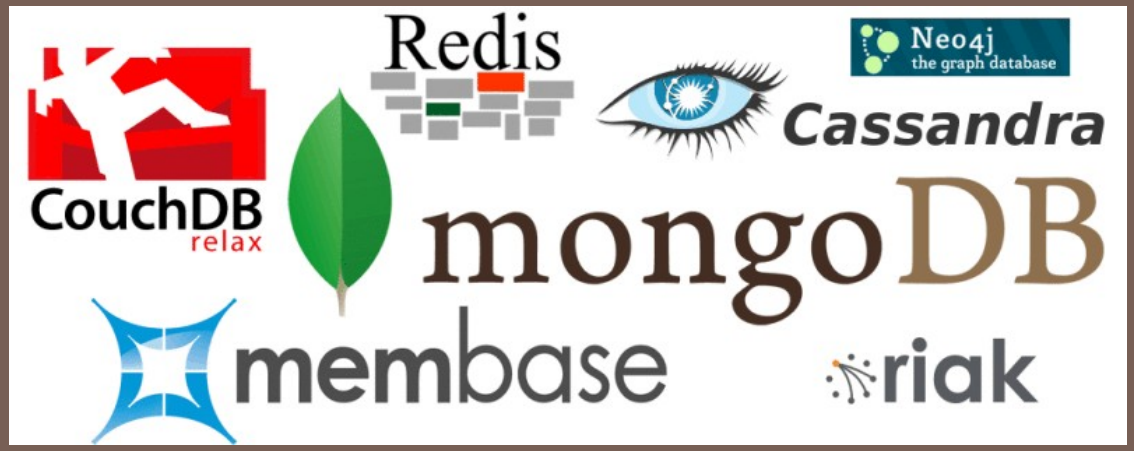

NoSQL : Neo4J

BD et SI

Frédéric Moal
Master 1 MIAGE

Qu’est-ce que Neo4J ?

- Neo4j est un système de gestion de base de données au code source libre basée sur les graphes, développé en Java par la société suédo-américaine Neo technology. Le produit existe depuis 2000, la version 1.0 est sortie en février 2010
- v2 : dec 2013, v3 : avr 2016
- 2 versions : community (pas de clustering), entreprise

Principes

- Basée sur des graphes : nœuds et arcs orientés
- Chaque objet possède un label (type) et des attributs clé-valeur (type simple : chaine, numérique...)
- Langage de requête Cypher, pattern sur les graphes

Installation

- Serveur(s) Docker :

```
docker run --publish=7474:7474 \
--publish=7687:7687 \
--env=NEO4J_AUTH=none \
neo4j
```

- Client web intégré sur le port 7474

Cypher

- Langage de requête https://neo4j.com/docs/
- Toutes les opérations de base sur les données
- Interface web [JS dynamique] pour exécuter les requêtes Cypher dans Neo4j

Cypher : create

- Creation nœuds :

```
CREATE ( {name:"Anny"}),
( {name:"JérômeB"}),( {name:"Alexis"}),
( {name:"Nicolas"}),( {name:"JérômeM"}),
( {name:"Corinne"})
```

Avec Labels et noms :

```
CREATE (Fred:Personne {nom:'Moal',
age:22})
```

Cypher : create

- ou plusieurs labels :

```
CREATE (Fred)-[:WORK_WITH
{module:'BD'}]->(Yohan)
```

Cypher : create

- Creation arcs :

```
CREATE (Fred)-[:WORK_WITH {module:'BD'}]->(Yohan)
```

Sinon recherche et création :

```
MATCH (a),(j)
WHERE a.nom = 'Moal' AND j.nom = 'Boichut'
CREATE (a)-[r:AMI]->(j)
RETURN r;
```

Cypher : requête

- Matching sur le graphe :

  match (r) return r;
  MATCH (n) RETURN n LIMIT 100

- La sélection dans le graphe peut être orientée ou non orientée :

```
MATCH (qq1)-[:AMI]->(qq2) WHERE qq1.name = 'José' RETURN qq2;

MATCH (qq1)-[:AMI]-(qq2) WHERE qq1.name = 'José' RETURN qq2;

MATCH (qq2)<-[:AMI]-(qq1) WHERE qq1.name = 'José' RETURN qq2;
```

Cypher : requête

- Graphe :

les amis des amis de :
MATCH (j {name:"Moal"})-[:AMI]->()-[:AMI]->(aa)
RETURN aa;

les amis des amis des amis :
MATCH (j {name:"Moal"})-[:AMI]->()-[:AMI]->()-
[:AMI]->(aaa) RETURN aaa;

ou en version courte :
MATCH (j {name:"Moal"})-[:AMI*3]->(aaa) RETURN
aaa;

Cypher : MAJ

- les mises à jour :

```
MATCH (fred{name:"Moal"}) SET
fred.pseudo='fred' RETURN fred;
```

la suppression :

```
MATCH (fred{name:"Moal"}) DELETE fred;
```

la suppression de relations :

```
MATCH (a{name:"Moal"})-[relation]-
(qqun:Personne) DELETE relation;
```

Cypher : refcard

- USE database
- CREATE INDEX FOR (p:Person) ON (p.name)
- CREATE CONSTRAINT ON (p:Person)
  ASSERT p.name IS UNIQUE
- CREATE CONSTRAINT ON ()-[l:LIKED]-()
  ASSERT l.when IS NOT NULL
- CREATE USER alice SET PASSWORD $password
- CREATE ROLE my_role
- GRANT ACCESS ON DATABASE * TO my_role

Cypher : refcard

- Cf https://neo4j.com/docs/cypher-refcard/4.3/