UNIVERSITÉ DE MONTPELLIER
-
FACULTÉ DES SCIENCES

Session : 1 — Durée de l'épreuve : 2 heures
Date : 18 janvier 2019 — Documents autorisés : tous
Mention Informatique — Matériel utilisé : aucun
Master 2ème année 2018 : Nouvelles Approches pour la Persistance des Données (HMIN340)

1 Questions de cours (5 points)

Donner des définitions concises pour trois (au choix) des six grandes notions suivantes (vous pouvez vous aider de schémas explicatifs si vous le jugez nécessaire)

1. Scalabilité horizontale
2. Réplication
3. Partitionnement
4. Principes du "théorème" CAP énoncé pour tout système distribué, pensé pour la gestion de gros volumes de données
5. Persistance polyglotte
6. Modèle semi-structuré

2 Le réseau de transport de la TAM et NoSQL

2.1 Enoncé

Les parties portant sur les systèmes de persistance entrevus en cours exploitent tous le contexte des transports urbains de l'agglomération de Montpellier (TAM)

2.2 Partie Neo4J (6 points)

1. Des ordres de création de nœuds et d'arêtes vous sont donnés dans la syntaxe Cypher. Vous construirez un premier graphe à partir des deux ordres de création donnés ci-dessous.

```
CREATE (s1:Station {nom:'Occitanie', latitude:43.63435811, longitude:3.84863696})
-[:STOP_DE]-> (la1:Liaison {distance:500}),
(la1) -[:APPARTIENT_A]->
(li1:Ligne {nom:'Mosson-Odysseum', num:1, type:'tram',longueur:15700, tempsTotal:50}),
(la2:Liaison {distance:1000}) -[:APPARTIENT_A]-> (li1), (s2:Station
{nom:'Hopital Lapeyronie',latitude:43.63166867, longitude:3.85260055}) -[:STOP_DE]->
(la1), (s1) -[:STOP_DE]-> (la2), (s3:Station
{nom:'Chateau d O',latitude:43.63131727, longitude:3.84299936}) -[:STOP_DE]-> (la2),
(s4:Station {nom:'Univ. Sciences et Lettres',latitude:43.63131727, longitude:3.84299936})
-[:STOP_DE]->
(la3:Liaison {distance:1200}),
(s2) -[:STOP_DE]-> (la3), (la3)-[:APPARTIENT_A]-> (li1)
RETURN s1, s2, s3, s4, li1, la1, la2, la3


MATCH (o:Station {nom:'Occitanie'})
CREATE (sp:Station {nom:'Saint-Priest', latitude:43.836699, longitude:4.360054})
-[:STOP_DE]-> (l:Liaison {distance:400}),
(l) -[:APPARTIENT_A]->(li:Ligne {nom:'Euromedecine-Pas du Loup', num:6, type:'bus'}),
(o) -[:STOP_DE]-> (l)
```

Vous choisirez le nom des stations, le numéro (num) des lignes, et la distance des liaisons pour donner une étiquette aux nœuds visualisés et vous mentionnerez les types de chacun des nœuds. Les autres propriétés des nœuds ne seront pas représentées.

2. Une requête de consultation en langage Cypher, vous est donnée qui porte sur le graphe qui vient d'être créé. Vous donnerez la signification de cette requête, ainsi qu'un exemple partiel du résultat renvoyé par cette requête

```
MATCH (l:Ligne {nom:'Mosson-Odysseum'}) <-[a:APPARTIENT_A]- ()
<-[st:STOP_DE]- (s:Station)
RETURN DISTINCT l.nom, s.nom
```

3. Une nouvelle requête Cypher vous est donnée (toujours sur le graphe créé). Vous donnerez également la signification de cette requête et et vous proposerez le sous-graphe qui est le résultat de la requête.

```
MATCH (l:Liaison) <-[:STOP_DE]- (s:Station)
RETURN l, s
```

4. Vous écrirez en langage Cypher, la requête : "donner le nom des stations qui sont des stops pour le même tronçon (liaison de ligne)"
5. Le choix de représentation arrêté pour représenter les lignes, les stations et leurs connexions manque d'efficacité et de simplicité, proposez une simplification du modèle pour le rendre plus performant, sans perdre de l'information.

2.3 Partie CouchDB (4 points)

Un exemple de document JSON décrivant une station, et géré au sein d'une BD CouchDB, vous est donné.

```
{
  "nom" : "Occitanie",
  "latitude" : 43.63435811,
  "longitude" : 3.84863696,
  "type" : "station",
  "lignes": [     {
          "libelle": "Mosson-Odysseum", "num":1,
          "typeLigne": "tram", "destination":"Hopital Lapeyronie"}, {
          "libelle": "Mosson-Odysseum", "num":1,
          "typeLigne": "tram", "destination":"Chateau d O"}]
  }
```

1. Vous modifierez ce document pour y rajouter la correspondance de la ligne de bus numéro 6 avec pour libellé "Euromedecine-Pas du Loup" et à destination de la station Saint-Priest
2. Donnez votre compréhension de la vue Map suivante (écriture en javascript du corps de la fonction map), que renvoie t'elle (illustrez le résultat obtenu en prenant le document précédent comme exemple) :

```
function(doc) { if (doc.type=='station')
  emit(doc.nom, doc.lignes.length);
}
```

3. Construisez une vue Map-Reduce qui compte le nombre de stations de la base de documents et donnez l'écriture javascript du corps de la fonction map et du corps de la fonction reduce

2.4 Partie HBase (4 points)

Une proposition de table et familles de colonnes vous est faite. Des tuples sont également fournis au travers des ordres JRuby associés.

```
create 'station', 'general', 'ligne'
put 'station', 'Occitanie', 'general:nom', 'Occitanie'
put 'station', 'Occitanie', 'general:latitude', '43.63435811'
put 'station', 'Occitanie', 'general:longitude', '3.84863696'
put 'station', 'Occitanie', 'ligne:libelle', 'Mosson-Odysseum'
put 'station', 'Occitanie', 'ligne:num', '1'
put 'station', 'Occitanie', 'ligne:typeLigne', 'tram'
put 'station', 'Occitanie', 'ligne:destination', 'Hopital Lapeyronie'
```

1. Vous reprendrez les ordres précédents sous une forme tabulaire
2. Cette représentation HBase vous semble valide ? Si oui justifiez, si non, expliquez pourquoi et modifiez le schéma en conséquence (table ou ordres JRuby).