Nom :
Prénom :

BDLE – Gestion des données graphes

Exemen réparti 2 du 3 Février 2017
Durée : 2 Heures
Documents autorisés

1 Requetes Cipher (3 pts)

Considérons le graphe de propriétés de la Figure 1 dont les noeuds représentent soit des films (*Movie*) soit des personnes (*Person*) et dont les arcs reliant exclusivement des personnes à des films indiquent soit l'appréciation d'un film (*ranks*) soit la participation à un film (*plays*). Les noeuds et les arcs comportent des attributs dont le sens est clair : par exemple, le noeud *Person* dont l'attribut *id* est 001 désigne la personne 'alex' qui attribue la note 3 au film 'lala' produit en 2016. Les noeuds films ne comportent pas d'identifiant dans cet exemple.

Question 1 (1 point)

Que calculent les deux requêtes ci-dessous ? Quel sont leurs résultats sur le graphe ci-dessus.

```
Match (p:Person)-[:ranks {note:2}]->(m:Movie)
return m
```

Réponse : Cette requête retourne

...
...
Son résultat est

```
Match (p:Person)-[:ranks | :plays]->(m:Movie)
return p, m
```

Réponse : Cette requête retourne

...
...
Son résultat est

Question 2 (1 point)

Formuler une requête Cipher qui extrait les paires de personnes ayant noté un film en commun.

Réponse :

Question 3 (1 point)

Formuler une requête Cipher qui extrait les films dans lesquels jouent au moins deux personnes et qui ont été notés par au moins deux autres personnes.

Réponse :

2 Traitement avec GraphX (5 pts)

Considérons la Table 1 qui fournit une représentation relationnelle du graphe Cinéma. Chaque ligne de vertices.txt représente les noeuds du graphe en associant à un id de noeud une chaine de caractères qui représente le type de ce noeud (Person ou Movie) et la valeur de l'attribut (Nom ou Titre) associée. Chaque ligne de edges.txt représente les arcs du graphe en associant à chaque paire d'identifiants de noeuds le type de l'arc ainsi que la valeur de l'attribut correspondant (note ou cachet).

Question 4 (1 point)

Comlpéter le code ci-dessous afin de permettre l'instanciation de la structure Graph de GraphX et tel que la structure obtenue pour les noeuds et les arcs possède le type ci-dessous

```
org.apache.spark.rdd.RDD[(VertexId, (String, String))]
org.apache.spark.rdd.RDD[Edge[(String, Int)]]
```

Réponse :

```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

var fvertices = sc.textFile("vertices.txt")
val vertexList:RDD[(VertexId,(String, String))] = fvertices. .....




var fedges = sc.textFile("edges.txt")
val edgesList:RDD[Edge[(String, Int)]] = fedges. ....




val graph = Graph.apply(vertexList, edgesList)
```

Question 5 (2 points)

Compléter le code ci-dessous qui calcule pour les films (noeuds de type Movie) la moyenne de ses notes.

Réponse :

```
val aggNotes = graph.aggregateMessages[(Int, Int)](
triplet => ....

....


,


.....


)

val moyNotes = ....
```

Question 6 (2 points)

Compléter le code ci-dessous qui calcule pour les films (noeuds de type Movie) la somme des cachets des Personnes qui y jouent.

Réponse :

```
val sumCachet = graph.aggregateMessages[(Int)](
triplet => ....

....


,


.....


)
```

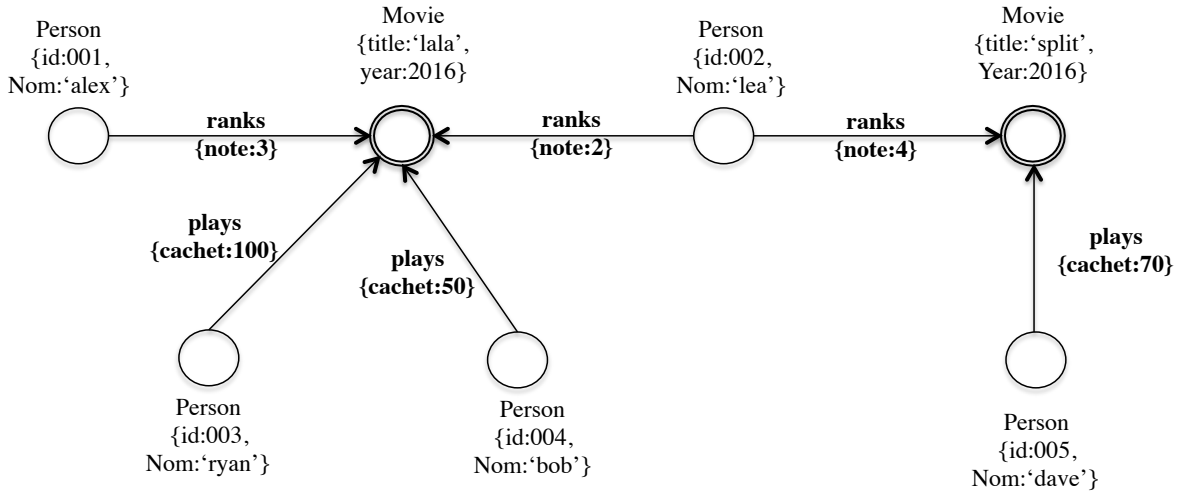

FIGURE 1 – Données cinéma (exercice 1)

id type nom
001,person,alex
002,person,lea
003,person,ryan
004,person,bob
005,person,dave
006,movie,lala
007,movie,split

vertices.txt

src dst type val
001,006,ranks,3
003,006,plays,100
004,006,plays,50
002,006,ranks,2
002,007,ranks,4
005,007,plays,7

edges.txt

TABLE 1 – Représentation relationnelle des données Cinéma (exercice 2)