Nom :
Prénom :

BDLE – Seconde Partie

Exemen réparti du 18 Novembre 2016
Durée : 2 Heures
Documents autorisés

1 Algèbre RDD de Spark (6 pts)

On s'intéresse à la formulation de requêtes Sparql en utilisant l'algèbre RDD de Spark vue en cours. On considère les triplets de l'encadré ci-dessous qui forment un graphe. Ces triplets, de la forme n_i, p_j, n_k, décrivent qu'un noeud n_k est atteignable à partir d'un noeud n_i via l'arc p_i.

n0, p0, n1	n0, p1, n6	n1, p1, n2	n2, p2, n4	n3, p0, n4	n3, p0, n1
n4, p1, n5	n5, p2, n2	n5, p2, n7	n7, p1, n6	n5, p3, n8	n8, p0, n7

On suppose que les instructions suivantes ont été exécutées :

```
scala> val triples = sc.textFile("...").map(x=>x.split(",")).map(x=>(x(0),x(1),x(2)))
scala> val p0 = triples.filter{case (s,p,o) => p.contains("p0")}
scala> val p1 = triples.filter{case (s,p,o) => p.contains("p1")}
scala> val p2 = triples.filter{case (s,p,o) => p.contains("p2")}
scala> val p3 = triples.filter{case (s,p,o) => p.contains("p3")}
```

Exprimer les requêtes suivantes dans l'algèbre RDD.

Question 1 (3 points)

```
(q1) select ?x ?z where { ?x p0 ?y . ?y p1 ?w. ?w p2 ?z }
```

Réponse : scala> val q1 = . . .

. . .

. . .

. . .

. . .

. . .

. . .

Rappel des données

n0, p0, n1	n0, p1, n6	n1, p1, n2	n2, p2, n4	n3, p0, n4	n3, p0, n1
n4, p1, n5	n5, p2, n2	n5, p2, n7	n7, p1, n6	n5, p3, n8	n8, p0, n7

Question 2 (3 points)

```
(q2) select ?x ?z where { ?x p2 ?z.  ?y p1 ?x. ?x p3 ?w}
```

Réponse : scala> val q2 = . . .

. . .

. . .

. . .

. . .

. . .

. . .

2 Algèbre Dataset de Spark (6 pts)

On s'intéresse à la formulation de requêtes SPARQL en utilisant l'API Dataset de Spark vue en TME. On considère les triplets de l'encadré ci-dessous. Un triplet a la forme *sujet,prop,objet*. Des étudiants (luke, liz, rick, etc.) ont étudié dans une université (ex. *luke,studiedAt,CMU*). Pour certains étudiants on connait le nom du superviseur (ex. *monica,supervisedBy,Yang*) avec éventuellement son université (*Yang,studiedAt,UCSD*). On connait l'état où se situe une université (*CMU,locatedAt,PA*) et l'état où habite un étudiant (*liz,livesIn,MA*).

luke,hasDegree,eng	Yang,studiedAt,UCSD	MIT,locatedAt,MA
luke,studiedAt,CMU	rick,supervisedBy,Horrow	luke,livesIn,PA
liz,studiedAt,CMU	Horrow,studiedAt,MIT	liz,livesIn,PA
rick,studiedAt,MIT	CMU,locatedAt,PA	rick,livesIn,MA
suzan,studiedAt,UCSD	UCSD,locatedAt,CA	monica,livesIn,CA
monica,supervisedBy,Yang	UCSB,locatedAt,CA	

Afin de faciliter la réponse aux questions, on suppose que les instructions suivantes ont été exécutées. case class Triple(sujet :String, prop :String, objet :String)
val TRIPLES = sc.textFile("exo2.txt").map(ligne => ligne.split(",")).map(t => Triple(t(0), t(1), t(2))).toDS()

```
val StudiedAt = TRIPLES.where("prop = 'studiedAt'").select("sujet","objet")
val SupervisedBy = TRIPLES.where("prop = 'supervisedBy'").select("sujet","objet")
val LocatedAt = TRIPLES.where("prop = 'locatedAt'").select("sujet","objet")
val LivesIn = TRIPLES.where("prop = 'livesIn'").select("sujet","objet")
```

Exprimer les requêtes suivantes.

Question 1 (2 points)

(R1) Les personnes *p* qui étudient dans une université *u* qui se situe dans un endroit *l*, ces personnes doivent avoir des encadrants *s*. La requête doit retourner *p, u* et *l*. La requête équivalente en SPARQL a trois motifs de triplets :

```
select ?p ?u ?l
where {?p studiedAt ?u. ?u locatedAt ?l. ?p supervisedBy ?s}
```

Le résultat de cette requête est (rick,MIT,MA). Décomposer votre réponse avec une expression par motif de triplet de la requête (t1 à t3), puis une expression finale (R1).

Réponse :

val t1 = . . .

val t2 = . . .

val t3 = . . .

val R1 = . . .

. . .

. . .

Question 2 (2 points)

(R2) Les personnes *p* qui étudient dans la même université *u* que luke. La requête retourne *p*. La requête équivalente en SPARQL est :

```
select ?p  where {'luke' studiedAt ?u. ?p studiedAt ?u.
FILTER (?p <> 'luke')}
```

Le résultat de cette requête est Liz (il ne contient pas luke).

Réponse :

val t1 = . . .

val t2 = . . .

val R2 = . . .

. . .

...

Question 3 (1 point)

(R3) Exprimer cette requête en français et calculer le résultat affiché.

```
val R3 = TRIPLES.where("prop = 'locatedAt' or prop = 'livesIn'").
                 groupBy("objet").count().sort("count")
```

Réponse :

...

...

le résltat affiché est : ...

Question 4 (1 point)

(R4) Les personnes *p* qui étudient dans la même université *u* que leur superviseur *s*. La requête retourne *p* et *s*. Voici la requête équivalente en SPARQL :

```
select ?p ?s
where {?p studiedAt ?u. ?p supervisedBy ?s. ?s studiedAt ?u}
```

Le résultat de cette requête est (rick,Horrow).

Réponse :

val t1 = ...

val t2 = ...

val t3 = ...

val R4 = ...

...