N° anonymat :		page 1

BDLE – 5IN852 - Examen réparti du 7 février 2020

Exercice 1 : Données touristiques — 8 pts

On considère le schéma. Chaque table est un dataset.

Visite (photoID, personID, date, lat, lon) *on connait la (latitude, longitude) d'une photo*

Intérêt (POI, lat, lon, catégorie) *catégorie vaut hôtel, musée, restau, ...*
on connait la (latitude, longitude) d'un point d'intérêt POI.

Place (photoID, pays, ville) *il y a 50 pays et 20 villes par pays en moyenne*

Il y a 10 000 tuples dans Visite, 11 000 dans Place et 1000 dans Intérêt.

On suppose que tous les attributs sont indépendants.

La répartition initiale d'un dataset est aléatoire (*ie.*, ne dépend pas d'un attribut) sur **10** machines, avec **une** partition par machine, et le même nombre d'objets dans chaque partition.
L'ordonnancement du traitement en plusieurs étapes suit le principe map-reduce de spark : rassembler dans une étape (*stage*) toutes les opérations pouvant être faites avant un transfert (*shuffle*) qui répartit les données pour d'autres étapes. On quantifie, si possible, les transferts de données en nombre d'objets transférés.

Question 1. Dans Place, une photo peut être associée à plusieurs pays. Par exemple, on peut avoir les tuples (photo1, *England, London*) et (photo1, *United-Kingdom, London*) pour une photo prise à Londres. On considère la requête :

```
E1 = Place.groupBy("photoID", "ville").agg(collect_list("pays").as("listeP"))
```

Réponse =>

Requete : SELECT photoID, ville, collect_list(pays) as listP
FROM Places
Group By photoID, ville

Voir le explain d'une requete groupeBy en TP, attention l'aggregation n'est pas un count ni un avg mais un count list

Fonction partielle dans chaque partition contenant 1100 tuples : aggregé les pays pour avoir un ensemble A(photo, ville, listePays) – Rmq : card(A)<= 1100

Repartition selon (photo,ville) correspond au shuffle

Fonction complete dans chaque nouvelle partition obtenue apres la répartition Pour tous les tuples ayant les memes (photo,ville) faire l'union des listPays. On obtient E1

Question 2. On considère **Place1** (photoID, pays, ville) où photoID est **unique**. Expliquer brièvement comment obtenir Place1 à partir de Place. Vous pouvez répondre par une expression en syntaxe Dataframe.

En SQL :

E2a: On suppose que photoID → Ville

```
with T as (select photoID, ville, collect_list(pays) as listeP
           From Places
           Group by photoID, ville)
```

```
Select photoID, ville, first(listeP) from T
```

as Pays

E2b: sans supposer que photoID → Ville

```
with T as (select photoID, collect_list(ville, pays) as listePV
                From Places
                Group by photoID)
        Select photoID, first(listePV) from T
```

as Pays

Remarque sur l'exécution parallèle de E2a et E2b : leur contenu n'est pas déterministe.

Pour rendre le résultat déterministe il faut imposer un ordre avant de faire le first

E2c :

```
with T as (select photoID, ville, collect_list(pays) as listeP
                From Places
                Group by photoID, ville)
        Select photoID, ville, first(array_sort(listeP)) from T
```

En syntaxe dataframe :

```
Place1 = E1.withColumn('pays', first(array_sort(listeP)).alias('pays')  ).select('photoID', 'ville', 'pays')
```

Question 3. Soit la requête affichant le nombre de villes dans chaque pays :

E3 = Place1.groupBy("pays").agg(**countDistinct**("ville").as("nv")).orderBy("pays")

Proposer une exécution avec **un seul** transfert (**= 1 seule repartition**) qui répartit les données par **intervalle** (et non par hachage). Décrire les étapes partielles et complètes du traitement.

Rappel répartition par intervalle : opérateur Exchange RangePartitionning dans le explain

On veut E3(pays, nv) trié par pays

Initialement Place1 est partitionné par photoID.

~~Fonction partielle : regrouper par pays et compter le nb de villes distinctes et trier par pays.~~

~~On obtient A(pays, nv)~~

Fonction partielle : regrouper par pays et agréger pour garder la liste des villes **distinctes** et trier par pays

On obtient A(pays, listeV) trié par pays

Chaque partition peut contenir 50 pays. Echanger 50* 9/10 éléments (pays, listeV)

Répartition par 10 intervalles de valeurs sur le domaine de pays (1 intervalle destiné à une machine)

Fonction complète : par **fusion** des morceaux (shuffle read) reçus car ils sont déjà triés. La fusion doit éliminer les villes en double pour chaque pays. Puis compter le nombre de villes

Rmq : autre solution A n'est pas trié par pays, et faire le tri dans la fonction complète.

Proposer une expression E3b donnant un résultat équivalent à celui de E3. E3b doit utiliser la fonction distinct() mais pas la fonction countDistinct().

E3b = Place1.select('pays', 'ville').distinct().groupBy(pays).agg(**count(1)**.as("nv")).orderBy("pays")

Rmq :plusieurs réponses possibles : **count(1)** ou **count(*)** ou **count(ville)**

Fonction partielle : projeter sur pays,ville obtenir l'ensemble des (pays,ville) **distincts**

Repartition par (pays,ville) par hachage (car le distinct() s'appuie sur une répartition par hachage sans tenir compte des opérations qui sont faites par la suite)

Fontion pour compléter le distinct : rassembler les (pays,ville) identiques provenant de différentes partitions

On obtient A(pays,ville) sans doubles

puis fonction partielle (pour le group by+agg) agréger par pays et **count** pour obtenir (pays, **nv**) puis tri par pays

Repartition par intervalle du domaine de pays

Fonction pour compléter le group by+agg : fusion de morceaux reçus et sum des nv pour chaque pays