

8INF803 Bases de données réparties - Hiver 2020

Rapport : Devoir 2 - Exercice 1

Groupe :

Gabriel LEBIS
Antoine ORHANT
Clément CARIOU
Tomy GUICHARD

Code et rapport par Gabriel LEBIS

Le code et les fichiers sont disponibles sur mon GitHub :

<https://github.com/G4bleb/distributed-databases/tree/master/Devoir2/Exercice1>

Crawling

J'ai fait le crawler en **Python** avec **Scrapy**

J'ai crawlé ces 5 indexes :

```
start_urls = ['http://legacy.aonprd.com/bestiary/monsterIndex.html',  
'http://legacy.aonprd.com/bestiary2/additionalMonsterIndex.html',  
              'http://legacy.aonprd.com/bestiary3/monsterIndex.html',  
              'http://legacy.aonprd.com/bestiary4/monsterIndex.html',  
              'http://legacy.aonprd.com/bestiary5/index.html',]
```

Le site était difficile à parser avec seulement des sélecteurs css (la méthode recommandée par scrapy), donc j'ai plutôt utilisé du regex sur le html.

C'est ce regex qui match les sorts dans la page, et récupère leur nom dans le groupe 1 :

```
p = re.compile('href=".*?\s/spells/\s.*?">(.*?)</a>')
```

Il repère un sort grâce au lien dans l'attribut href, qui contient "spells"

Le crawler exporte ses données dans un tableau json (*data.json*):

```
[...  
{"name": "Bat Swarm", "spells": []},  
{"name": "Basilisk", "spells": ["flesh to stone"]},  
...]
```

Je le formate avec sed pour qu'il puisse être importé en dataframe Spark, car spark lit le format *JSON Lines*, des objets json séparés en lignes :

```
sed '1d;$d' data.json # supprime les crochets au début et à la fin du fichier  
| sed '/\[ \]/d' #Supprime les lignes où une créature n'a pas de sorts  
| sed 's/},/}/' #Supprime les virgules après un bracket fermant  
| sort > scala_readable.json #Trie les créatures par ordre alphabétique (pourquoi pas)
```

Le fichier *scala_readable.json* se présente comme suit :

```
...
{"name": "Bebilith", "spells": ["plane shift"]}
{"name": "Beheaded", "spells": ["animate dead", "air walk", "fly"]}
{"name": "Belker", "spells": ["gaseous form"]}
...
```

Map reduce avec Spark :

Le schéma du dataframe récupéré avec Spark est le suivant :

```
|-- name: string (nullable = true)
|-- spells: array (nullable = true)
|     |-- element: string (containsNull = true)
```

J'en fait un RDD :

```
val myCreatureArray = myDataFrame.distinct()
  .collect()
  .map{ row:Row =>
    (row.getAs[String]("name"), row.getAs[mutable.WrappedArray[String]]("spells").array)
  }
```

Map :

Pour chaque sort d'une créature, on émit un couple *nom_sort*, [*nom_créature*]. Le nom de la créature est dans un tableau pour pouvoir le concaténer avec les autres, formant ainsi un tableau de tous les noms des créatures ayant le sort

```
var mapped = creatures.flatMap(elem => {
  val results = new ArrayBuffer[(String, Array[String])]
  for (spellname <- elem._2) { // Pour chaque sort d'une créature
    results += Tuple2(spellname, Array(elem._1)) // Emit (nomdusort, [nomdelacréature])
  }
  results
})
```

Reduce :

```
var reduced = mapped.reduceByKey((a, b) => {
  a ++ b // Concaténer les tableaux de noms
})
```

Export :

Export du RDD:

Je sérialise le tableau de nom avec des ";" entre chaque nom

```
val serialized = reduced.map(x => (x._1, x._2.mkString("; ")))  
serialized.saveAsTextFile("/home/gableb/bddreparties/Devoir2/Exercice1/output_rdd")
```

Une fois exporté :

```
...  
(magic weapon,Lukwata; Maftet; Drowning Devil (Sarglagon); Disenchanter)  
(leashed shackles,Guardian Dragon; Shadow Collector)  
(command plants,Zomok; Ancient Blue Dragon; Fungal Nymph; Fastachee)  
...
```

Export en Json :

Je recrée un dataframe à partir du RDD (il faut nommer les colonnes)

```
spark.createDataFrame(reduced).toDF("spell", "creatures").write.format("json")  
    .save("/home/gableb/bddreparties/Devoir2/Exercice1/output_json");
```

Une fois exporté :

```
...  
{ "spell": "leashed shackles", "creatures": ["Guardian Dragon", "Shadow Collector"] }  
{ "spell": "command plants", "creatures": ["Zomok", "Ancient Blue Dragon", "Fungal  
Nymph", "Fastachee"] }  
...
```