TP n°4, MapReduce avec CouchDB

Partie 1 : Installation et utilisation de CouchDB :

Mise en place de CouchDB dans docker :

Docker étant installé, on va dans un premier temps créer un volume pour CouchDB :
```
docker volume create couchdb_data
```

On va ensuite run le container en précisant le nom et le mot de passe d'un utilisateur admin. On va également préciser le volume qu'on utilise :
```
docker run --name couchdb -e COUCHDB_USER=arthur -e
COUCHDB_PASSWORD=bassette -p 5984:5984 -v couchdb_data:/opt/couchdb/data -d
couchdb
```

Il possible de se connecter à une interface graphique en accéder à l'url suivante :
http://localhost:5984/_utils

Il faut bien sûr s'assurer d'être sur le bon le bon port. Il suffira ensuite de se connecter avec les identifiants définis dans la commande précédente.

Utilisation de CouchDB :

Il faut savoir que CouchDB repose sur un API REST. C'est-à-dire qu'on va pouvoir interagir avec cette dernière via les commandes GET, PUT, POST et DELETE. On peut utiliser divers moyens d'actionner ces commandes, par exemple via CURL en ligne de commande ou via POSTMAN en interface graphique.

On va voir les différentes actions possibles via ces instructions.

Afin de récupérer des informations, nous allon utiliser la commande GET :

```
curl -X GET http://arthur:bassette@localhost:5984
```

Une fois de plus, il faut bien s'assurer de mettre votre propre identifiant et mot de passe avant le localhost.

Si on veut créer un document on va utiliser PUT :
```
curl -X PUT http://arthur:bassette@localhost:5984/films
```

On peut bien sûr récupérer les informations de la collection nouvellement créée :
```
curl -X GET http://arthur:bassette@localhost:5984/films
```

Il est possible d'importer des documents qui sont enregistrés sur la machine depuis laquelle on lance les requêtes grâce à POST :
```
curl -X POST http://arthur:bassette@localhost:5984/films/_bulk_docs -H "Content-Type:
application/json"  --data-binary "@films_couchdb.json"
```

Il est possible de récupérer les données d'un film précis de cette collection selon son id :
```
curl -X GET http://arthur:bassette@localhost:5984/films/movie:10098
```

Map Reduce avec CouchDB :

Avec CouchDB, les fonctions MapReduce sont exprimées en JavaScript.

Principe de Map Reduce : supposons une collection composée de documents indépendants. La fonction Map va être appliquée sur chaque document indépendamment. Il va appliquer un certain nombre d'opérations sur ce dernier, peut en retourner aucun, un ou plusieurs. Il va également y appliquer des transformations. Par la suite, une étape intermédiaire, le sort and shuffle a lieu.

Dans couchDB, on peut uniquement mettre en place la fonction Map sans nécessairement implémenter la fonction Reduce. On peut donc mettre en place des transformations intermédiaires et les sauvegarder.

On va d'abord implémenter la fonction map suivante afin de tester, en javascript. Cette fonction va récupérer les document un par un puis va renvoyer uniquement l'année du film et son titre:
```
function (doc) { emit(doc.year, doc.title); }
```

Voici un exemple de sortie de cette fonction :

key	value
1921	Le Kid
1927	L'Aurore
1931	M le maudit
1936	Les Temps modernes
1936	La Charge de la brigade légère
1937	La Grande Illusion

On peut donc noter que doc en argument permet de nommer le document sur laquelle va agir la fonction, emit va indiquer les champs qu'on va récupérer, ici doc.year et doc.title.

On peut bien sûr également définir une fonction reduce. Voici une fonction reduce qui va agir sur le map précédent afin de compter le nombre de films par année :
```
function (keys, values) { return values.length; }
```

Exemple d'affichage de cette fonction :

1952	2
1954	1
1955	1
1956	2

Ainsi, les fonctions Map et Reduce permettent de transformer les données et de les agréger selon le besoin. On peut comparer ça aux GroupBy que l'on connaît en SQL associé aux opérateurs tels que SUM, MIN, MAX etc…sauf qu'ici nous sommes beaucoup plus libres dans ce que nous pouvons faire puisque nous pouvons définir ces fonctions.

Par exemple, voici une fonction map bien plus complexe, qui associée à la fonction Reduce permet de calculer le nombre de film dans lequel a pu jouer chaque acteur.

Map :

```
function(doc) {
  for (i = 0; i < doc.actors.length; i++) {
    emit({"prénom": doc.actors[i].first_name, "nom": doc.actors[i].last_name}, doc.title) ;
  }
}
```

Reduce :

```
function (keys, values) { return values.length; }
```

Affichage :

{ "prénom": "Adolfo", "nom": "Celi" }	1
{ "prénom": "Adolphe", "nom": "Menjou" }	1
{ "prénom": "Adrian", "nom": "Rawlins" }	1
{ "prénom": "Adrien", "nom": "Brody" }	2