Gloire BAYOUNDOULA
INFOA3

Matière : **Bases de données NoSQL**
Partie TP4 : **MapReduce avec CouchDB**
Enseignant : **Monsieur Samir Youcef**

Rendu TP4 MapReduce avec CouchDB

Disclaimer : J'ai réalisé ce TP sur Windows en utilisant Docker desktop. Une fois, Docker desktop installé, il comporte un terminal linux dans lequel on peut utiliser toutes les commandes linux requises pour le TP.

- Documentation officielle de CouchDB : couchdb - Official Image

Introduction à MapReduce et CouchDB

MapReduce est un modèle de programmation utilisé pour le traitement parallèle de grandes quantités de données. Il est particulièrement utilisé dans le domaine du Big Data. CouchDB, un système de gestion de bases de données NoSQL, implémente MapReduce pour effectuer des calculs distribués de manière efficace.

CouchDB est une base de données NoSQL open-source orientée documents, développée par Apache. Elle stocke les données sous forme de documents JSON flexibles, sans schéma rigide, ce qui facilite la gestion de données semi-structurées sur divers appareils comme les mobiles ou navigateurs.

CouchDB utilise une API REST qui peut être installée localement ou via Docker. Les documents sont stockés au format JSON, et l'identifiant (_id) est unique. Il supporte le versionnement des documents.

Cette API REST est utilisée avec HTTP pour les opérations CRUD, avec un système de vues dynamiques basées sur JavaScript (MapReduce) pour interroger les données efficacement.

Mais on va utiliser le client curl avec Docker Desktop à la place. Mais il y a d'autres clients ou des plugins que l'on peut ajouter à son navigateur également.
Pour afficher la représentation d'une ressource

Installation avec Docker

Pour installer CouchDB avec Docker et mapper les volumes :

```
docker run -d   --name couchdb   -p 5984:5984   -e COUCHDB_USER=youssef
  -e COUCHDB_PASSWORD=samir   couchdb:3
```

-e : pour définir une variable d'environnement
Il y'a deux variables d'environnement:

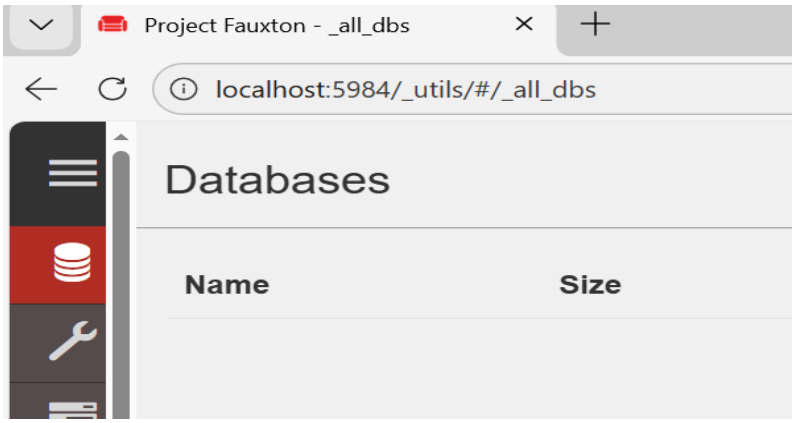
- ➢ COUCHDB_USER : nom d'utilisateur
- ➢ COUCHDB_PASSWORD : mote de passe

Mapping des ports, port d'écoute par défaut de CouchDB : 5984
couchDB : Nom de l'image

Vérifiez que le conteneur est en cours d'exécution : docker ps

1. Commande vidéo 1 CouchDB

CouchDB expose une API REST facilitant la manipulation des bases de données et documents. L'interface graphique est accessible via http://localhost:5984/_utils.



Tester la connexion à CouchDB

Pour avoir le client en ligne de commande, la commande sur linux est :
curl -X GET http://youssef:samir@localhost:5984/

Mais comme j'utilise Docker Desktop, j'utilise la commande
curl.exe -X http://youssef:samir@localhost:5984/

```
PS C:\Users\cloud> curl.exe http://youssef:samir@localhost:5984/
{"couchdb":"Welcome","version":"3.5.1","git_sha":"44f6a43d8","uuid":"0d44ff413633d2d093a1b031201f6447","fe
atures":["access-ready","partitioned","pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":
"The Apache Software Foundation"}}
PS C:\Users\cloud>
```


Importation et Manipulation de Données

Pour afficher toutes les bases de données

curl -X GET http://localhost:5984/_all_dbs

Cette commande liste toutes les bases de données disponibles sur CouchDB.

Pour créer une base de données, la commande est en général :

curl -X PUT http://localhost:5984/films

Il renvoie ok si le document est bien créé.
Elle crée une nouvelle base de données nommée films.

curl.exe -X PUT http://youssef:samir@localhost:5984/films

```
PS C:\Users\cloud> curl.exe -X PUT http://youssef:samir@localhost:5984/films
{"ok":true}
PS C:\Users\cloud>
```

Pour afficher la représentation de la ressource films:

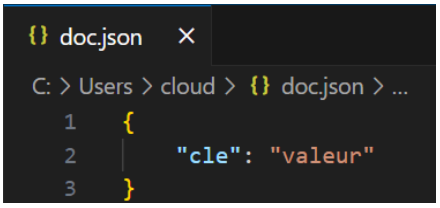
PS C:\Users\cloud> curl.exe -X GET http://youssef:samir@localhost:5984/films

```
PS C:\Users\cloud> curl.exe -X GET http://youssef:samir@localhost:5984/films
{"instance_start_time":"1766017180","db_name":"films","purge_seq":"0-g1AAAABPeJzLYWBgYMpgTmHgzcvPy09JdcjLz
8gvLskBCeexAEmGBiD1HwiyEhlwqEtkSKqHKMgCAIT2GV4","update_seq":"0-g1AAAACLeJzLYWBgYMpgTmHgzcvPy09JdcjLz8gvLs
kBCeexAEmGBiD1HwiyMpgTGXKBAuxGRuZpFsZG6HpwmJLIkFSPoj0lLdnUyDINXXEWAPNMKos","sizes":{"file":16692,"external
":0,"active":0},"props":{},"doc_del_count":0,"doc_count":0,"disk_format_version":8,"compact_running":false
,"cluster":{"q":2,"n":1,"w":1,"r":1}}
PS C:\Users\cloud>
```


Pour insérer un document dans la base films

On utilise la commande :
PS C:\Users\cloud> curl.exe -X PUT http://youssef:samir@localhost:5984/films/doc -d '{"cle":"valeur"}'

Mais comme je suis sur Docker Desktop, j'ai créé un fichier doc.json dans lequel, j'ai stocké les informations et ensuite j'ai exécuté la commande :



curl.exe -X PUT http://youssef:samir@localhost:5984/films/doc -H "Content-Type: application/json" --data-binary "@doc.json"

```
curl: (3) URL rejected: Port number was not a decimal number between 0 and 65535
PS C:\Users\cloud> curl.exe -X PUT http://youssef:samir@localhost:5984/films/doc -H "Content-Type: applica
tion/json" --data-binary "@doc.json"
{"ok":true,"id":"doc","rev":"1-0f3beea1048634b34d8091e22ab3c6fb"}
```


Si vous souhaitez ajouter un autre document avec :

curl.exe -X PUT http://youssef:samir@localhost:5984/films/doc -d '{"nom":"gloire"}'

Cela va créer une erreur car l'identifiant doc est déjà connu, il faut donc changer et mettre un identifiant uniquement pour chaque document.

Insérer un document sauvegardé dans un fichier

curl curl.exe -X POST http://localhost:5984/exemple -d @movie:10098.json -H "Content-Type: application/json"

NB: Ce n'est pas une bonne idée de définir soi-même une clé car les systèmes de gestion de bases de données NOSQL gèrent des données dans une infrastructure distribuée afin de faciliter la gestion de la panne des serveurs et de l'arrêt d'un nœud.

Insertion d'une collection de documents (bulk insert)

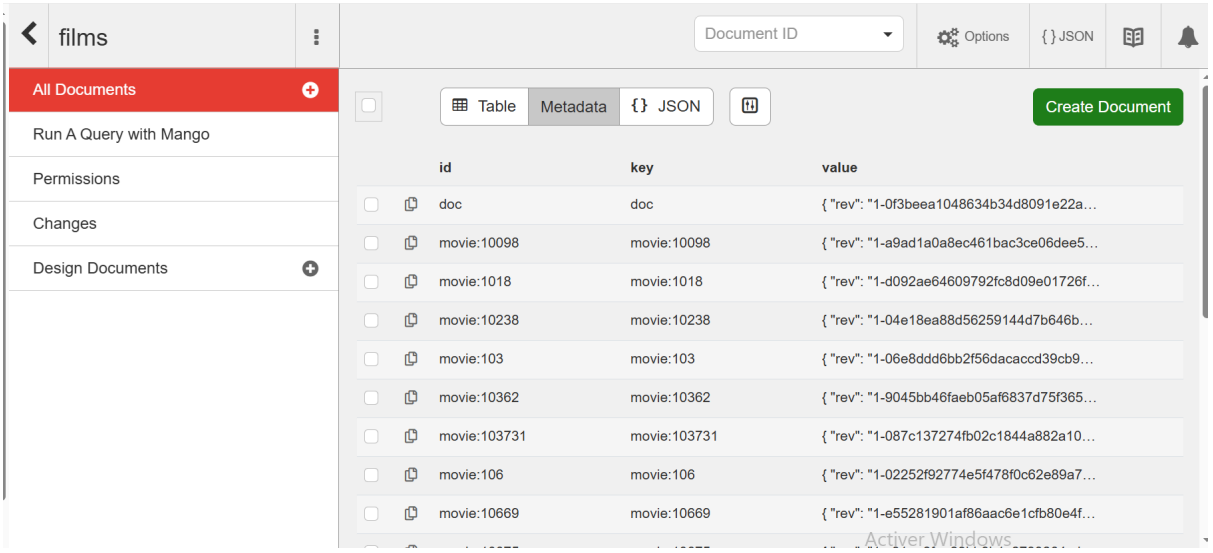
En supposant que nous ayons le fichier films_couchdb.json :

curl -X POST http://youssef:samir@localhost:5984/films/_bulk_docs -d @films_couchdb.json -H "Content-Type: application/json"

J'ai rajouté - -data-binary car j'utilise Docker Desktop

```
PS C:\Users\cloud> curl.exe -X POST http://youssef:samir@localhost:5984/films/_bulk_docs -H "Content-Type:
 application/json" --data-binary "@C:\Users\cloud\films_couchdb.json"
[{"ok":true,"id":"movie:11","rev":"1-c404285f85cc593f351855821ebe5fc7"},{"ok":true,"id":"movie:24","rev":"
1-7f851a642ab7108adad8354952d4c560"},{"ok":true,"id":"movie:28","rev":"1-5ef74f3007d597da5c1a41d73e00f308"
},{"ok":true,"id":"movie:33","rev":"1-210992fbbd105dd91ceb02a1f6b1811d"},{"ok":true,"id":"movie:38","rev":
"1-902184f7cc63bc4f802f3b33ffd2eb27"},{"ok":true,"id":"movie:59","rev":"1-2ca4990b59fbaee2006e0b0ef74481d0
"},{"ok":true,"id":"movie:62","rev":"1-89d7541cf67625fbeda283dadf7294bb"},{"ok":true,"id":"movie:74","rev"
:"1-ea1b40608799bd603ebc7f82cf4511ac"},{"ok":true,"id":"movie:75","rev":"1-522355c47de05179064d6218542b6ca
7"},{"ok":true,"id":"movie:77","rev":"1-85291b834cfda40e18739d1b37d6deef"},{"ok":true,"id":"movie:78","rev
":"1-c2b126bd26e20d4256ea573e5bc5a11a"},{"ok":true,"id":"movie:85","rev":"1-ae348bef3600f3a445ed329201ccd1
```

On peut observer que sur l'outil en ligne, il y'a les films qui se sont rajoutés au document "doc":



Pour un film comme le movie:10098, par exemple

Il faut toujours regarder la structure des documents, il y a un identifiant et ce sont des documents JSON imbriqués les uns dans les autres pour créer des documents complexes en violent la première forme normale qui est exigée par les systèmes de gestion de bases de données relationnels.ici on peut avoir plusieurs acteurs qui participent dans le film dans le même fichier. Contrairement à une base de données relationnelle, où il faut faire le lien entre le film et chaque acteur avec des clés étrangères, ça permet d'éviter la redondance des données et l'incohérence .

Pour récupérer un document

curl -X GET http://localhost:5984/films/movie:10098:
Cette commande récupère le document ayant _id = movie:10098.

Comme je suis sur Docker Dekstop, j'ai utilisé cette commande :

curl.exe -X GET http://youssef:samir@localhost:5984/films/movie%3A10098

Pour supprimer un document

curl -X DELETE http://localhost:5984/films/movie:10098?rev=<REV_ID>

Remarque : <REV_ID> est la version du document, obtenue avec la commande GET. Cela supprime le document movie:10098 en précisant sa révision (_rev).

Pour supprimer une base de données

curl -X DELETE http://localhost:5984/films

Pour lister les documents de la base

curl -X GET http://youssef:samir@localhost:5984/films/_all_docs

Pour **lister les bases existantes**
curl -X GET http://youssef:samir@localhost:5984/_all_dbs

2. Commandes vidéo 2 MapReduce CouchDB

Commandes MapReduce

MapReduce repose sur deux fonctions principales : **Map** et **Reduce**.

2.1 Fonction Map

La fonction Map est appliquée individuellement à chaque document d'une base de données JSON. Elle extrait des informations et émet des paires (clé, valeur) qui seront ensuite regroupées.

function (doc) {

```
  if (doc.type === "user") {
    emit(doc._id, doc.name);
  }
}
```

- Cette fonction parcourt chaque document.
- Si le type du document est "user", elle émet l'identifiant du document (doc._id) comme clé et son nom (doc.name) comme valeur.

2.2 Fonction Reduce

La fonction Reduce agrège les résultats produits par la fonction Map.

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

- Cette fonction prend en entrée une liste de clés et leurs valeurs associées.
- Elle retourne le nombre total de valeurs, permettant par exemple de compter le nombre d'utilisateurs.

Lister les bases existantes

curl -X GET http://youssef:samir@localhost:5984/_all_dbs

3. Exemple d'application :

Prenons un exemple pratique où l'on veut calculer le nombre de films par année.

3.1 Fonction Map pour extraire les films par année

```
function (doc) {
  if (doc.year && doc.title) {
    emit(doc.year, doc.title);
  }
}
```

3.2 Fonction Reduce pour compter le nombre de films par année

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

Réponses aux questions TP

1. Nombre total de films

```
map = function() { emit("total", 1); }
```

```
reduce = function(key, values) { return Array.sum(values); }
```

Exécutez db.movies.mapReduce(map, reduce) pour obtenir le total.

2. Films par genre

```
map = function() { emit(this.genre, 1); }
```

```
reduce = function(key, values) { return Array.sum(values); }
```

3. Films par réalisateur

```
map = function() { emit(this.director.last_name, 1); }
```

```
reduce = function(key, values) { return Array.sum(values); }
```

Utilise director.last_name (ex: "Chaplin").

4. Acteurs uniques

Compte les apparitions uniques d'acteurs sur tous les films.

```
map = function() {
  this.actors.forEach(function(actor) {
    emit(actor.last_name, 1);
  });
}
```

```
reduce = function(key, values) { return Array.sum(values); }
```

5. Films par année

```
map = function() { emit(this.year, 1); }
```

```
reduce = function(key, values) { return Array.sum(values); }
```

11. Acteurs par genre (sans doublons)

```
map = function() {
  var actorsSet = {};
  this.actors.forEach(function(actor) {
    actorsSet[actor.last_name] = true;
  });
  emit(this.genre, Object.keys(actorsSet));
}
reduce = function(key, values) {
  var uniqueActors = {};
  values.forEach(function(actorsList) {
    actorsList.forEach(function(actor) { uniqueActors[actor] = true; });
  });
  return Object.keys(uniqueActors);
}
```

12. Acteurs les plus prolifiques

```
map = function() {
  this.actors.forEach(function(actor) {
    emit(actor.last_name, 1);
  });
}
reduce = function(key, values) { return Array.sum(values); }
```

14. Note moyenne par année (si grades présent)

```
map = function() { if (this.grades) emit(this.year, this.grades.mean || 0); }
reduce = function(key, values) {
  var sum = 0, count = 0;
```

```
  values.forEach(function(v) { sum += v; count++; });
  return sum / count;
}
```