TP : couchDB

Ghassen BEN JANNET

Préambule

CouchDB, ou Apache CouchDB, est un système open-source de gestion de bases de données NoSQL orienté documents. Il utilise un modèle de données flexible basé sur JSON, offrant une alternative performante et intuitive aux bases de données relationnelles traditionnelles.

Une particularité de CouchDB est son utilisation intégrale d'une **API RESTful**, facilitant l'accès et la manipulation des données via les méthodes HTTP classiques : `GET`, `PUT`, `POST` et `DELETE`. Cette approche rend CouchDB particulièrement adaptée aux applications web modernes et aux environnements distribués.

CouchDB dispose également d'un puissant moteur d'exécution **MapReduce**, permettant de créer des vues dynamiques et persistantes des données. Ces vues sont automatiquement mises à jour lorsque les données changent, offrant ainsi une performance accrue lors des requêtes répétées.

— **Note :** Les fonctions MapReduce dans CouchDB sont principalement écrites en **JavaScript**, facilitant leur développement et leur intégration avec des applications web.

Partie 1 : Initiation avec CouchDB

Lancement de CouchDB avec Docker

Pour démarrer rapidement CouchDB, on utilise Docker afin de récupérer et exécuter une image prête à l'emploi :

```
docker run -d --name couchdb \
  -e COUCHDB_USER=admin \
  -e COUCHDB_PASSWORD=admin123 \
  -p 5984:5984 apache/couchdb
```

Après le lancement réussi du conteneur, vous pouvez accéder à l'interface d'administration **Fauxton** via l'URL suivante :

— `http://localhost:5984/_utils/`

Tester CouchDB avec Curl

Pour vérifier le bon fonctionnement de CouchDB, vous pouvez utiliser l'outil en ligne de commande **Curl** :

```
curl -X GET http://user_name:password@localhost:5984
```

Dans notre cas spécifique, la commande devient :

```
curl -X GET http://admin:admin123@localhost:5984
```

La réponse attendue devrait ressembler à ceci :

```
PS C:\Users\ghass\Desktop\couchDB> curl.exe -X GET http://admin:admin123@localhost:5984
{"couchdb":"Welcome","version":"3.4.2","git_sha":"6e5ad2a5c","uuid":"941e4515a98c53fbc180827c61bb0002","features":["acce
ss-ready","partitioned","pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"The Apache Software Foundat
ion"}}
PS C:\Users\ghass\Desktop\couchDB>
```

FIGURE 1 – Exemple de réponse CouchDB via Curl

Ceci confirme que CouchDB est correctement lancé et prêt à être utilisé.

Création d'une ressource

Pour créer une ressource (base de données), on utilise la méthode **PUT** :

```
curl -X PUT http://username:password@localhost:5984/nom_ressource
```

Par exemple, pour créer la base **films** :

```
curl -X PUT http://admin:admin123@localhost:5984/films
```

La réponse suivante indique le succès de l'opération :

```
PS C:\Users\ghass\Desktop\couchDB> curl.exe -X PUT http://admin:admin123@localhost:5984/films
{"ok":true}
PS C:\Users\ghass\Desktop\couchDB>
```

FIGURE 2 – Confirmation de la création de la ressource *films*

Cette base peut également être visualisée depuis l'interface Fauxton du navigateur :

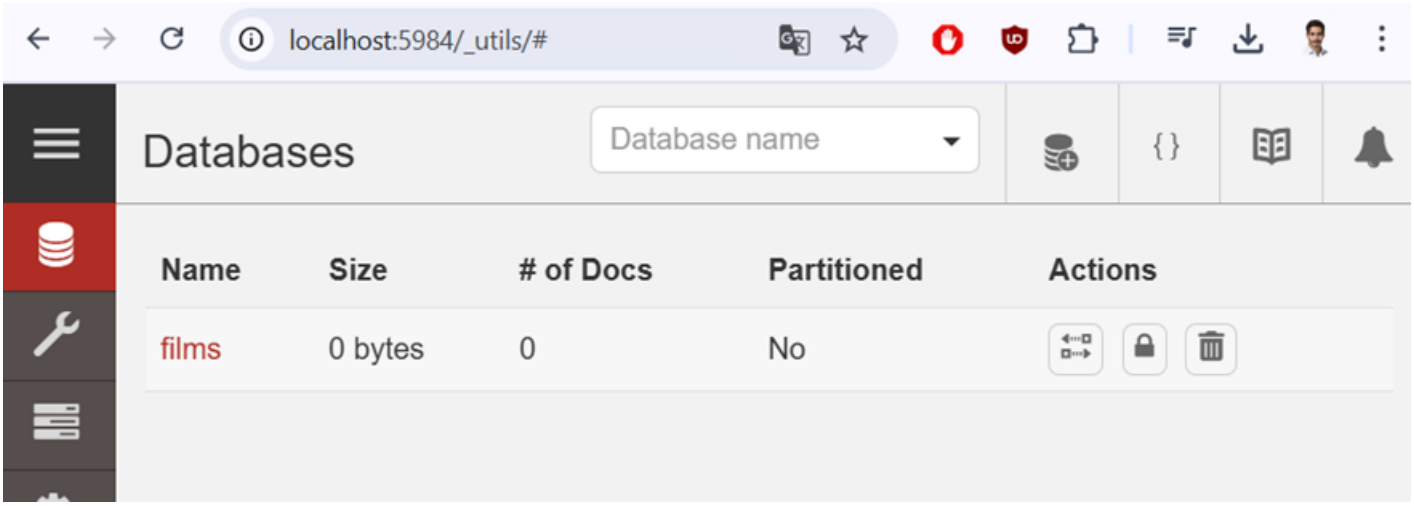

FIGURE 3 – Visualisation de la ressource *films* via Fauxton

Visualisation d'une ressource

Pour visualiser une ressource, utilisez la méthode **GET** :

```
curl -X GET http://admin:admin123@localhost:5984/films
```

Quelques points clés de la réponse :

— `"db_name": "films"` : Confirme que la base *films* existe.
— `"doc_count": 0` : Aucun document présent dans cette base.
— `"disk_format_version": 8` : Version du format de stockage.
— `"compact_running": false` : Indique qu'aucune compaction n'est en cours.

```
PS C:\Users\ghass\Desktop\couchDB> curl.exe -X GET http://admin:admin123@localhost:5984/films
{"instance_start_time":"1739436985","db_name":"films","purge_seq":"0-g1AAAABPeJzLYWBgYMpgTmHgzcvPy09JdcjLz8gvLskBCeexAEm
GBiD1HwiyEhlwqEtkSKqHKMgCAIT2GV4","update_seq":"0-g1AAAACLeJzLYWBgYMpgTmHgzcvPy09JdcjLz8gvLskBCeexAEmGBiD1HwiyMpgTGXKBAu
ymKanJiQZG6HpwmJLIkFSPot082dDMLNEAXXEWABaTKqY","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\ghass\Desktop\couchDB>
```

FIGURE 4 – Visualisation détaillée de la ressource *films*

Ceci confirme la création correcte et l'état actuel de la base **films**.

Insertion d'un document

Pour insérer un document dans la base **films**, utilisez la commande suivante :

```
curl -X PUT http://admin:admin123@localhost:5984/films/doc -d '{"cle" : "valeur"}' -H "Content-Type: application/json"
```

Application :

```
PS C:\Users\ghass\Desktop\couchDB> curl.exe -X PUT http://admin:admin123@localhost:5984/films/doc -d "{ \`"cle\`": \`"valeur\`" }" -H "Content-Type: application/json"
{"ok":true,"id":"doc","rev":"1-0f3beea1048634b34d8091e22ab3c6fb"}
PS C:\Users\ghass\Desktop\couchDB>
```

FIGURE 5 – Insetion du document

Ce document sera visible dans Fauxton.

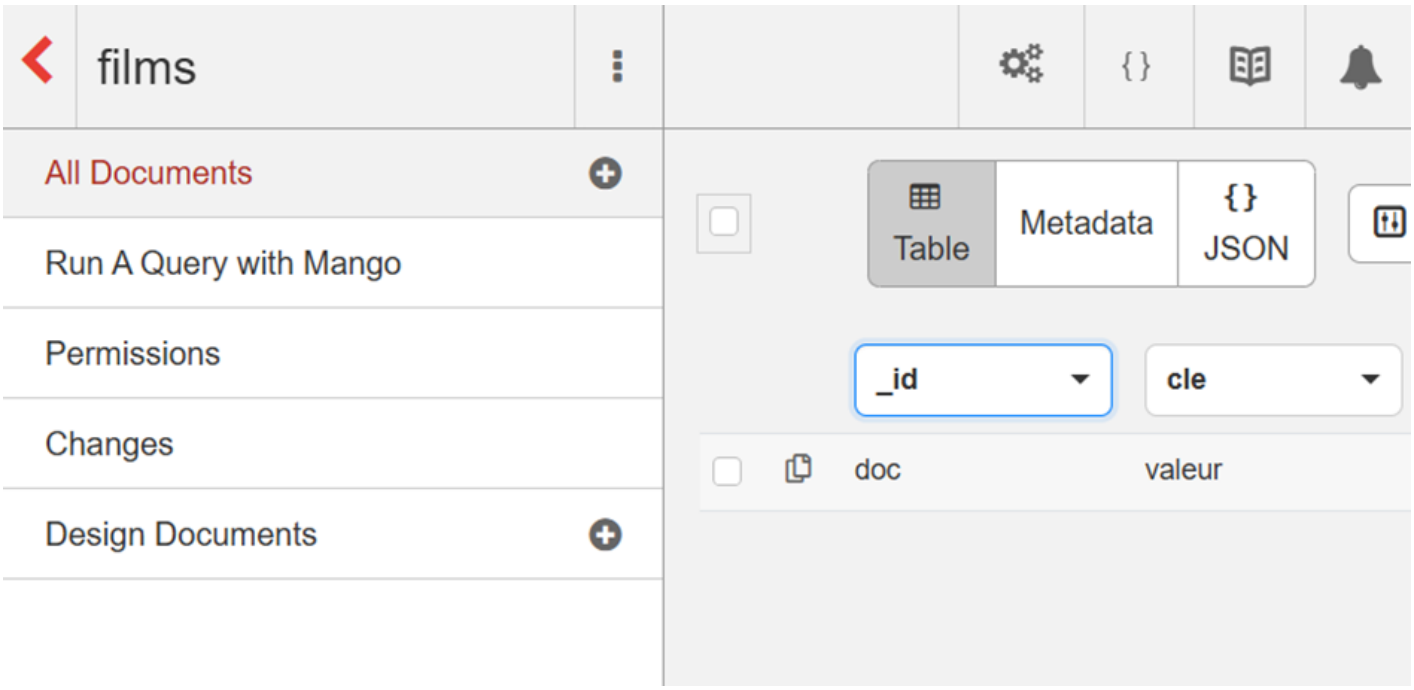

FIGURE 6 – Visualisation du document inséré

Gestion des conflits

Si vous essayez d'insérer un document avec le même ID existant, vous aurez un conflit :

```
curl -X PUT http://admin:admin123@localhost:5984/films/doc -d '{"nom" : "ghass"}'
```

Réponse :

```
{"error":"conflict","reason":"Document update conflict."}
```

Pour contourner ceci, utilisez un nouvel identifiant :

```
curl -X PUT http://admin:admin123@localhost:5984/films/doc1 -d '{"nom": "ghass"}' -H "Content-Type: application/json"
```

Insertion en masse

Vous pouvez insérer plusieurs documents simultanément à partir d'un fichier JSON :

```
curl -X POST http://admin:admin123@localhost:5984/films/_bulk_docs -H "Content-Type: application/json" --data-binary "@C:\\Users\\ghass\\Desktop\\couchDB\\films_couchdb.json"
```

[h !]

Si aucun identifiant n'est spécifié, CouchDB en générera automatiquement un pour chaque document.

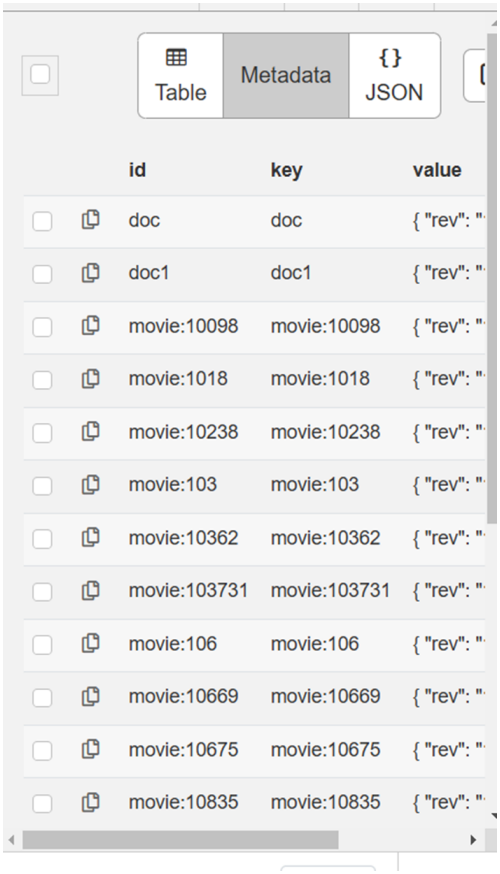

FIGURE 7 – Insertion en masse

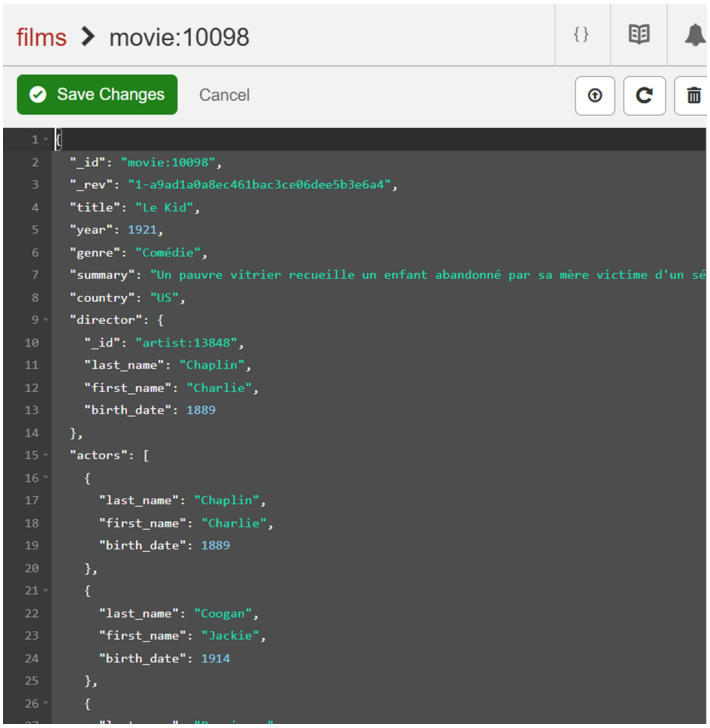

FIGURE 8 – Détails : insertion en masse

Vérifier l'existence d'un document

Pour vérifier l'existence et récupérer les données d'un document spécifique :

```
curl -X GET http://admin:admin123@localhost:5984/films/movie:10098
```

Cette commande affiche le contenu détaillé du document identifié par `movie:10098`, confirmant ainsi son existence et permettant de visualiser ses informations complètes.

Partie 2 : MapReduce avec CouchDB

Présentation de MapReduce dans CouchDB

MapReduce est une approche de traitement des données utilisée dans CouchDB pour créer des vues indexées. Une fonction `map` est utilisée pour émettre des paires clé-valeur, et une fonction `reduce` permet d'agréger ces valeurs.

Exemple 1 : Nombre de films par année

Fonction Map

Pour calculer le nombre de films par année, utilisez cette fonction `map` :

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

Cette fonction produit des paires clé-valeur année-titre.

Fonction Reduce

Cette fonction `reduce` compte le nombre de films par année :

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

Requête MapReduce

Vous pouvez exécuter cette vue avec la commande suivante :

```
curl -X GET "http://admin:password@localhost:5984/movies/_design/movies/_view/movies_by_year?group=
    true"
```

Exemple 2 : Nombre de films réalisés par acteur

Fonction Map

Cette fonction `map` émet les acteurs avec le titre des films dans lesquels ils ont joué :

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

Fonction Reduce

Cette fonction `reduce` compte le nombre de films réalisés par acteur :

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

Requête MapReduce

Exécutez la vue avec :

```
curl -X GET "http://admin:password@localhost:5984/movies/_design/movies/_view/movies_by_actor?group=true"
```

Conclusion

CouchDB est une base de données NoSQL puissante et flexible permettant une gestion simple et efficace des documents. Grâce à Docker et `curl`, il est facile de configurer et d'interagir avec CouchDB. L'utilisation de MapReduce permet de créer des vues indexées pour exploiter efficacement les données stockées.

Fonction map par défaut : Cette fonction par défaut traite chaque document et l'émet tel quel :

```
function(doc) {
    emit(null, doc);
}
```

Application Exercice n°1 : Traitement MapReduce

1. Modèle de représentation

La matrice M peut être représentée sous forme d'une collection C de documents structurés, chaque document représentant une page web avec ses liens sortants accompagnés de leur poids (importance) respectif. Ce modèle est adapté au traitement par MapReduce :

```
{
  "_id": "page_i",
  "liens": [
    {"page": "page_j", "poids": 0.5},
    {"page": "page_k", "poids": 0.3}
    // ... autres liens ventuels
  ]
}
```

2. Calculer la norme des vecteurs lignes

Chaque ligne i de la matrice M est un vecteur représentant la page P_i. La norme d'un vecteur V est définie par :

$$\|V\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_N^2}$$

Fonction Map

La fonction `map` traite chaque document individuellement. Elle calcule la somme des carrés des poids des liens sortants d'une page donnée (vecteur ligne) :

```
function(doc) {
  var sommeCarres = 0;
  for (var i = 0; i < doc.liens.length; i++){
    sommeCarres += Math.pow(doc.liens[i].poids, 2);
  }
  emit(doc._id, sommeCarres);
}
```

Cette fonction émet ainsi une paire clé-valeur : la clé est l'identifiant de la page et la valeur est la somme des carrés des poids.

Fonction Reduce

La fonction `reduce` reçoit les sommes intermédiaires émises par la fonction `map`, et agrège ces valeurs pour calculer la norme finale en prenant la racine carrée de cette somme :

```
function(keys, values) {
  var sommeTotale = 0;
  for(var i = 0; i < values.length; i++){
    sommeTotale += values[i];
  }
  return Math.sqrt(sommeTotale);
}
```

3. Produit matrice-vecteur avec MapReduce

On souhaite calculer le produit de la matrice M par un vecteur W :

$$\phi_i = \sum_{j=1}^{N} M_{ij} W_j$$

Le vecteur W est supposé stocké en mémoire RAM accessible à toutes les fonctions Map ou Reduce.

Fonction Map

Pour chaque page (ligne de la matrice M), la fonction `map` calcule la somme partielle en multipliant chaque poids du lien sortant par la valeur correspondante dans le vecteur W :

```
function(doc) {
  var sommeProduit = 0;
  for(var i = 0; i < doc.liens.length; i++){
    var pageCible = doc.liens[i].page;
    sommeProduit += doc.liens[i].poids * W[pageCible];
  }
  emit(doc._id, sommeProduit);
}
```

Ici, chaque clé émise est l'identifiant de la page, et la valeur correspond au résultat partiel du produit ligne-vecteur.

Fonction Reduce

La fonction `reduce` agrège simplement les résultats partiels pour obtenir le produit final pour chaque page :

```
function(keys, values) {
  var resultatFinal = 0;
  for(var i = 0; i < values.length; i++){
    resultatFinal += values[i];
  }
  return resultatFinal;
}
```