1 Manipulation des bases de données

Création d'une base de données

Pour créer une base de données appelée films :

curl -X PUT http :// abc :123 @ localhost :5984/ films

Insertion de documents

Insertion d'un document simple :

```
curl -X PUT http://abc:123@localhost:5984/films/doc1 -d '{"titre
":"Inception", "annee":2010}'
```

Insertion document

Pour insérer plusieurs documents :

```
curl -X POST http://abc:123@localhost:5984/films/_bulk_docs -d
@films_couchdb.json -H "Content-Type: application/json"
```

" docs ": [

{" _id ": " movie :1001" , " titre ": " Inception " , " annee ": 2010} ,

{" _id ": " movie :1002" , " titre ": " Interstellar " , " annee ": 2014}]

Récupération d'un document

Pour récupérer un document avec un identifiant donné :

```
curl -X GET http://abc:123@localhost:5984/films/movie:1001
```

2 MapReduce avec CouchDB

Qu'est-ce que MapReduce ?

MapReduce est un mécanisme qui permet d'analyser et d'organiser les données dans CouchDB. Il fonctionne en deux étapes :

- **Map** : chaque entrée de données est traitée indépendamment, et une paire clé-valeur est produite.
- **Reduce** :les paires clé-valeur sont agrégées par clé pour produire un résultat final.

Exemple : Nombre de films par année

Fonction Map

La fonction Map émet l'année comme clé et le titre comme valeur :

```
function (doc) {
  emit(doc.annee, doc.titre);
}
```

Fonction Reduce

La fonction Reduce compte le nombre de titres par année :

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

Exemple : Nombre de films pour chaque acteur

Fonction Map

Cette fonction extrait l'année comme clé et le titre du film comme valeur :

```
function (doc) {
  emit(doc.annee, doc.titre);
}
```

Fonction Reduce

Cette fonction compte le nombre de films par année :

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

3 Conclusion

CouchDB, avec son approche simple et son moteur MapReduce, est idéal pour travailler avec des bases de données documentaires et traiter de grandes quantités de données.

4 Exercice n°1 : Modèle et traitement MapReduce

Exercice n° 1 : soit une matrice M de dimension $N \times N$ représentant des liens d'un très grand nombre de pages web (soit N). Chaque lien est étiqueté par un poids (son importance).

1. Proposer un modèle, sous forme de ducuments structurés, pour représenter une telle matrice (s'inspirer du cas Page Rank du moteur de recherche Google, vu en cour). Soit C la collection ainsi obtenue.
2. La ligne i peut être vue comme un vecteur à N dimensions décrivant la page P_i. Spécifiez le traitement MapReduce qui calcule la norme de ces vecteurs à partir des documents de la collection C. La norme d'un vecteur $V(v_1, v_2, \ldots, v_N)$ est le scalaire $||V|| = \sqrt{v_1^2 + v_2^2 + \ldots + v_N^2}$
3. Nous voulons calculer le produit de la matrice M avec un vecteur de dimension N, $W(w_1, w_2, \ldots, w_N)$. Le résultat est un vecteur $\phi = \sum_{j=1}^{N} M_{ij} w_j$, On suppose que le vecteur W tient en mémoire RAM et est accessible comme variable statique par toutes les fonctions de Map ou de Reduce. Spécifiez le traitement MapReduce qui implante ce calcul.

Modèle de données

Dans cet exercice, nous allons représenter une matrice **M**, qui symbolise les liens entre plusieurs pages web. Chaque élément **Mij** de la matrice indique la relationentre la page **Pi** et la page **Pj**, avec un poids qui mesure l'importance du lien.

Structure des documents dans CouchDB

Chaque ligne de la matrice est stockée sous forme de document JSON, où :

- L'identifiant (_id) représente une page web.
- La clé links contient les liens vers d'autres pages avec leur poids associé.

Exemple de document dans CouchDB :

```
{
 "_id": "page_1",
 "links": {
  "page_1": 0.1,
  "page_2": 0.3,
  "page_3": 0.5
 }
}
```

Ce document représente la **page_1**. Elle contient des liens vers **page_1, page_2 et page_3**. Chaque lien a un poids (exemple : **0.3** pour page_2).

Calcul de la norme d’un vecteur

Dans notre cas, chaque ligne de la matrice **M** représente un vecteur, où chaque élément correspond au poids d’un lien entre une page et une autre.
Pour mesurer l’importance de ces liens, nous allons calculer la **norme** de chaque vecteur en utilisant la formule mathématique :

Étape Map

La fonction **Map** récupère tous les poids des liens d'une page et calcule la somme des carrés.

```
function map(doc) {
  let sum = 0;
  if (doc.links) {
    for (let page in doc.links) {
      sum += Math.pow(doc.links[page], 2);
    }
    emit(doc._id, sum);
  }
}
```

Étape Reduce

La fonction **Reduce** applique la racine carrée à la somme obtenue pour obtenir la norme du vecteur.

```
function reduce(keys, values, rereduce) {
  return Math.sqrt(sum(values));
}
```

Calcul du produit de la matrice *M* avec un vecteur *W*

Le produit d'une matrice M par un vecteur W permet de pondérer les liens entre les pages par des coefficients spécifiques.

Dans notre cas, M représente les relations entre les pages, et W est un vecteur stocké en mémoire qui sert à ajuster l'importance des liens.

Étape Map

La fonction **Map** multiplie chaque poids de la matrice par l'élément correspondant du vecteur **W** et émet le résultat.

```
function map(doc) {
  const vectorW = [1, 2, 3, 4, 5]; // Exemple de vecteur W
  let result = 0;

  if (doc.links) {
    for (let page in doc.links) {
      let index = parseInt(page.split('_')[1]); // Extraction de l'index
      result += doc.links[page] * (vectorW[index - 1] || 0);
    }
    emit(doc._id, result);
  }
}
```

Étape Reduce

La fonction Reduce additionne les valeurs obtenues pour chaque page

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