Bases de données NoSQL

MapReduce

Nom: HAMRANI
Prénom: Amziane

Introduction:

Dans le TP précédent, nous avons abordé des concepts fondamentaux des bases de données NoSQL liés à la **montée en charge** et à la **tolérance aux pannes**, notamment le **sharding** et la **réplication**.

- Le **sharding** permet de répartir les données sur plusieurs nœuds afin de gérer de très grands volumes de données et d'augmenter les performances.
- La **réplication** permet de dupliquer les données sur plusieurs nœuds afin d'assurer la disponibilité du système et sa résistance aux pannes.

Cependant, ces mécanismes soulèvent une nouvelle problématique essentielle : **chaque nœud ne possède qu'une partie des données**. Dès lors, comment effectuer des **calculs globaux** (comptages, moyennes, agrégations, statistiques) sur un ensemble de données réparti sur plusieurs machines ?

C'est pour répondre à cette problématique qu'intervient le modèle **MapReduce**.

Qu'est-ce que MapReduce ?

MapReduce est un modèle de programmation conçu pour effectuer des **traitements distribués** sur de grandes quantités de données. Il permet de paralléliser les calculs sur plusieurs nœuds tout en masquant la complexité de la distribution des données.

Le principe repose sur deux phases principales :

- **Phase Map:**

Chaque nœud traite localement la partie des données qu'il possède. La fonction map transforme les données d'entrée en paires **clé / valeur**.
Exemples :

- Associer un genre à un film
- Associer un réalisateur à une note
- Associer une année de sortie à un compteur

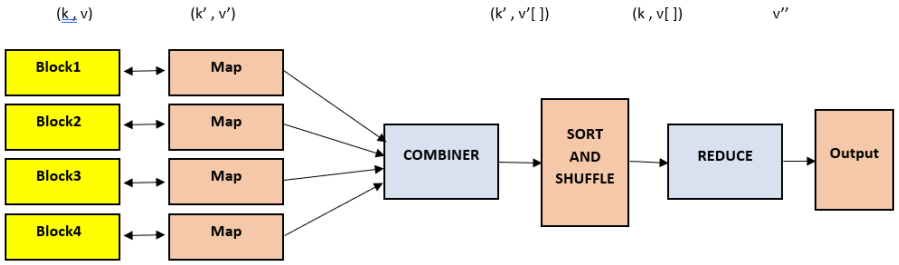
- **Phase *Reduce***

Les résultats produits par la phase map sont regroupés par clé, puis la fonction reduce agrège les valeurs associées à chaque clé.

Exemples :

- Additionner des compteurs
- Calculer une moyenne
- Fusionner des listes sans doublons

Grâce à ce mécanisme, MapReduce permet d'effectuer des calculs globaux sans jamais avoir besoin de rapatrier toutes les données sur une seule machine.



CouchDB:

Apache CouchDB est une base de données **NoSQL orientée documents**, conçue pour fonctionner dans des environnements distribués et tolérants aux pannes. Chaque document CouchDB est identifié par un identifiant unique (_id) et une révision (_rev), ce qui permet une gestion fine des conflits lors de la réplication.

Son architecture repose sur des principes simples : immutabilité des données, écriture append-only et synchronisation par réplication.

CouchDB et MongoDB : similitudes et différences:
CouchDB et MongoDB appartiennent tous deux à la famille des bases NoSQL orientées documents, ce qui implique plusieurs points communs importants.

Malgré ces points communs, les deux systèmes diffèrent sur plusieurs aspects clés:

- MongoDB propose un langage de requêtes riche et un framework d'agrégation avancé, CouchDB repose principalement sur des **vues MapReduce persistantes** pour interroger les données.
- CouchDB met fortement l'accent sur la réplication et la gestion des conflits, MongoDB privilégie la performance des requêtes temps réel.

MapReduce dans CouchDB

Dans CouchDB, MapReduce est le mécanisme principal d'interrogation des données. Contrairement à d'autres bases NoSQL, il n'existe pas de langage de requêtes complexe : les requêtes s'appuient sur des vues, définies dans des design documents.
Un point fondamental de CouchDB est que les résultats MapReduce sont **persistants**. Une fois une vue créée, CouchDB met automatiquement à jour son index lorsque les documents changent.

Cela permet d'obtenir des requêtes rapides sans recalculer systématiquement l'ensemble des données.

Installation:

Afin de simplifier l'installation et d'assurer un environnement reproductible, CouchDB est déployé à l'aide de **Docker**.

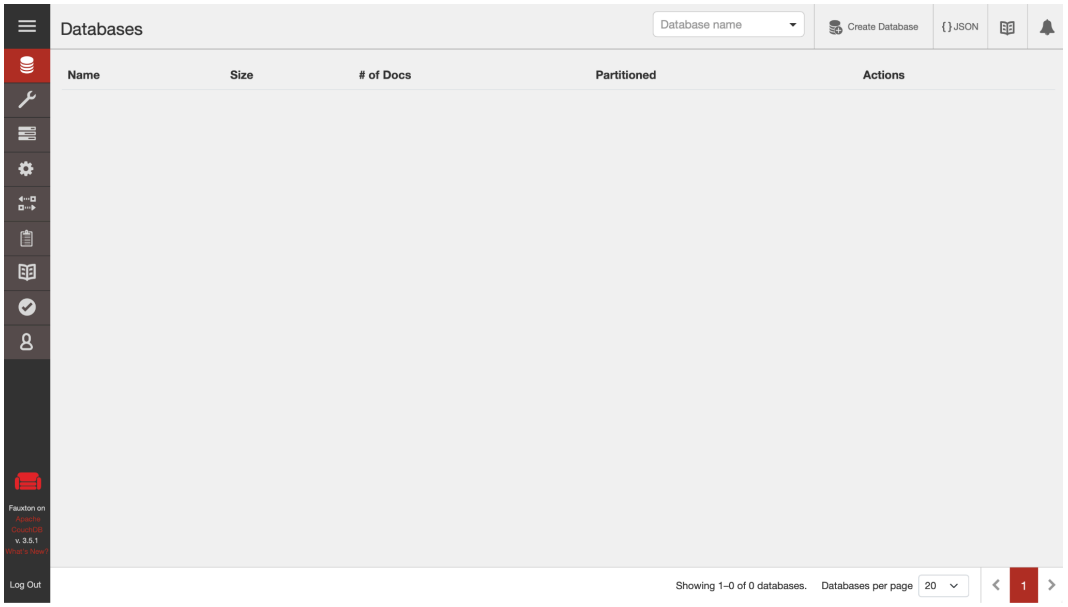
Lancement du conteneur:

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

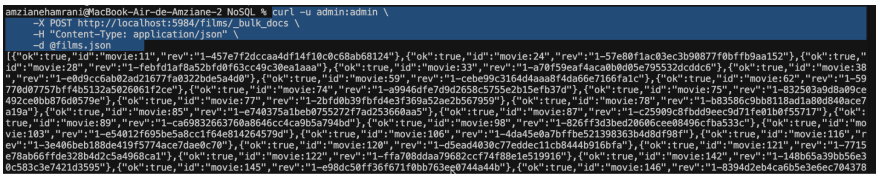
```
amzianehamrani@MacBook-Air-de-Amziane-2 NoSQL % docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                                  NAMES
12911004c7e4   couchdb:latest   "tini -- /docker-ent…"   40 seconds ago   Up 39 seconds   4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp   couchdb
```

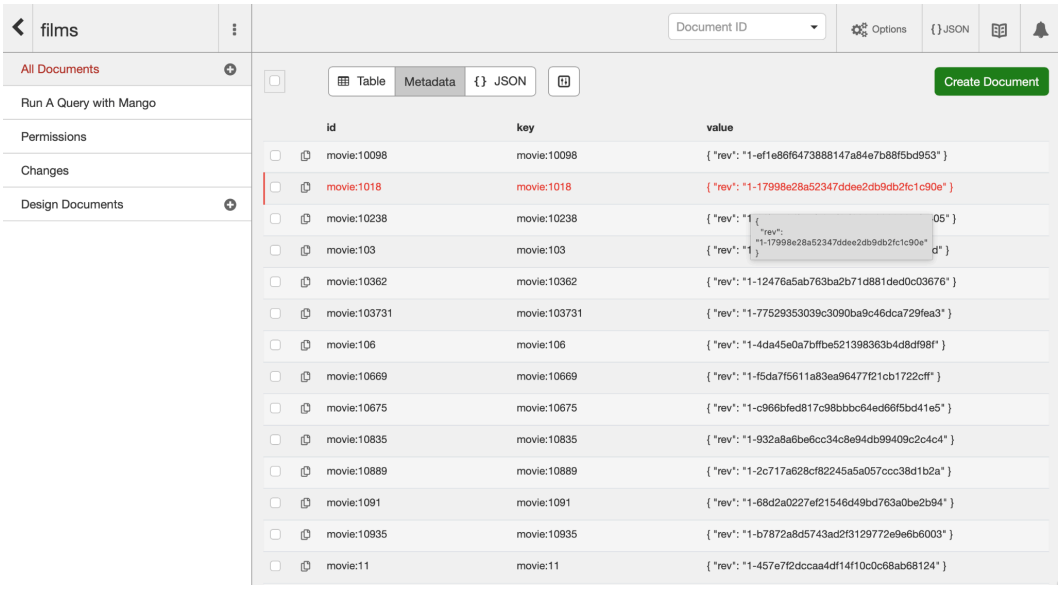
CouchDB fournit une interface web d'administration appelée Fauxton.

http://localhost:5984/_utils



Création de la DB et insertion de documents:





Exercices MapReduce avec CouchDB

1. Nombre total de films

Map:

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

Reduce:

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

2. Nombre de films par genre

Map:

```
function (doc) {
    emit(doc.genre, 1);
}
```

Reduce:

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

key	value
Action	36
Adventure	3
Aventure	22
Comédie	25
Comedy	1
Crime	29
Drama	14
Drame	96
Fantastique	4

Fantastique

3. Nombre de films par réalisateur

Map:

```
function (doc) {
  emit([doc.director.first_name, doc.director.last_name],
1);
}
```

Reduce:

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

key	value
["Abel", "Ferrara"]	2
["Adam", "McKay"]	1
["Akira", "Kurosawa"]	3
["Alain", "Corneau"]	1
["Alain", "Resnais"]	1
["Alan", "J. Pakula"]	1
["Alejandro", "González Iñárritu"]	2
["Alfred", "Hitchcock"]	10
["André", "Téchiné"]	1

4. Nombre d’acteurs uniques

Map:

```
function (doc) {
    doc.actors.forEach(function (a) {
    emit(a.first_name + " " + a.last_name, null);
    });
}
```

Reduce:

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

key	value
A.J. Cook	null
Aaron Eckhart	null
Abdelghafour Elaaziz	null
Abigail Breslin	null
Adam Driver	null
Adam Goldberg	null
Adel Bencherif	null
Adèle Haenel	null
Adolfo Celi	null

5. Nombre de films par année de sortie

Map:

```
function (doc) {
    emit(doc.year, 1);
}
```

Reduce:

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

key	value
1921	1
1927	1
1931	1
1936	2
1937	1
1940	3
1941	1
1942	1
1943	1

6. Note moyenne par film

Map:

```
function (doc) {
    doc.grades.forEach(function (g) {
        emit(doc.title, { sum: g.note, count: 1 });
    });
}
```

Reduce:

```
function (keys, values, rereduce) {
    var res = { sum: 0, count: 0 };
    values.forEach(function (v) {
    res.sum += v.sum;
    res.count += v.count;
    });
    return res;
}
```

key	value
2001 l'Odyssée de l'espace	{ "sum": 311, "count": 4 }
3 Billboards : Les Panneaux de la vengeance	{ "sum": 207, "count": 4 }
58 minutes pour vivre	{ "sum": 127, "count": 4 }
À bout de souffle	{ "sum": 223, "count": 4 }
A History of Violence	{ "sum": 225, "count": 4 }
Ali	{ "sum": 175, "count": 4 }
Alice	{ "sum": 178, "count": 4 }
Alice et le maire	{ "sum": 25
Alien : Covenant	{ "sum": 15

9. Film avec la meilleure note maximale

Map:

```
function (doc) {
    var max = Math.max.apply(null, doc.grades.map(g =>
    g.note));
    emit(doc.title, max);
}
```

Reduce:

```
function (keys, values, rereduce) {
    return Math.max.apply(null, values);
}
```

key	value
2001 l'Odyssée de l'espace	97
3 Billboards : Les Panneaux de la vengeance	90
58 minutes pour vivre	58
À bout de souffle	100
A History of Violence	91
Ali	73
Alice	69
Alice et le maire	81
Alien : Covenant	60

À bout de souffle

10. Nombre de notes strictement supérieures à 70

Map:

```
function (doc) {
    doc.grades.forEach(function (g) {
    if (g.note > 70) emit("sup70", 1);
    });
}
```

Reduce:

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

key	value
sup70	317

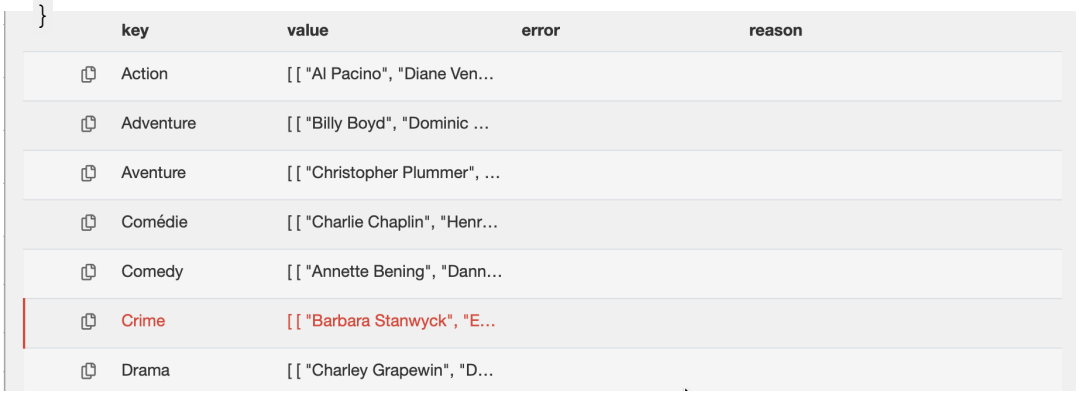
11. Acteurs par genre (sans doublons)

Map:

```
function (doc) {
    doc.actors.forEach(function (a) {
    emit(doc.genre, a.first_name + " " + a.last_name);
    });
}
```

Reduce:

```
function (keys, values, rereduce) {
    return Array.from(new Set(values));
}
```



key	value	error	reason
Action	[["Al Pacino", "Diane Ven…		
Adventure	[["Billy Boyd", "Dominic …		
Aventure	[["Christopher Plummer", …		
Comédie	[["Charlie Chaplin", "Henr…		
Comedy	[["Annette Bening", "Dann…		
Crime	[["Barbara Stanwyck", "E…		
Drama	[["Charley Grapewin", "D…		

Questions ouvertes:

Modèle de documents pour représenter la matrice M: On a une très grande matrice N×N, donc on ne stocke pas une matrice dense, mais uniquement les liens existants, comme pour PageRank.

```
{
  "i": i,
  "j": j,
  "value": v
}
```

Calcul de la norme des vecteurs (une norme par ligne i):

Map: produire $M^2[i][j]$ pour chaque ligne i.

```
function (doc) {
  emit(doc.i, doc.value * doc.value);
}
```

Reduce: sommer puis calculer la racine carrée

```
function (keys, values, rereduce) {
  let sommeCaree = 0;
  values.forEach(function (v) {
    sommeCaree+= v;
  });
  return Math.sqrt(sommeCaree);
}
```

Produit matrice M × vecteur W

Map: regrouper tous les M[i][j] par ligne i

```
function (doc) {
  emit(doc.i, { j: doc.j, v: doc.value });
}
```

Reduce: calculer la somme M[i][j]* W[j]

```
function (keys, values, rereduce) {
  let s= 0;
  values.forEach(function (e) {
    s+= e.v * W[e.j];
  });
  return s;
}
```