

# Bases de données NoSQL

MapReduce



# CouchDB

**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

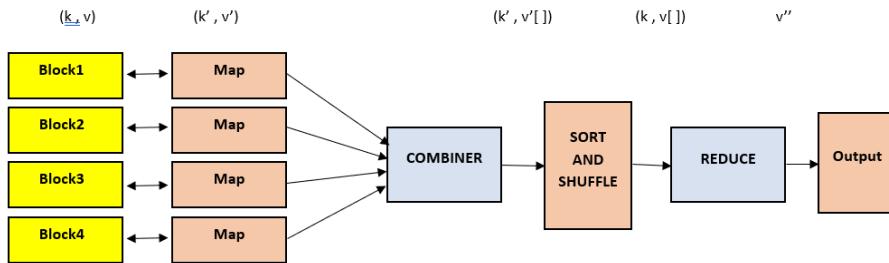
#### - Phase Reduce

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 **persistentes**. 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
12911004c7e4        couchdb:latest     "tini -- /docker-ent..."   40 seconds ago   Up 39 seconds      4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
NAMES               couchdb
```

CouchDB fournit une interface web d'administration appelée Fauxton.

[http://localhost:5984/\\_utils](http://localhost:5984/_utils)

The screenshot shows the Fauxton interface, which is a web-based administration tool for Apache CouchDB. The page has a header with the title 'Databases'. On the left, there is a sidebar with various icons for database management. The main content area displays a table with columns: Name, Size, # of Docs, Partitioned, and Actions. At the bottom of the table, it says 'Showing 1–0 of 0 databases.' There is also a 'Databases per page' dropdown set to 20, and navigation buttons for the first page (number 1).

## Création de la DB et insertion de documents:

```
amzianehamran@MacBook-Air-de-Amziane-2 NoSQL % curl -u admin:admin \
-X POST http://localhost:5984/_temp/_bulk_docs \
-H "Content-Type: application/json" \
-d "[{"ok":true,"id":"movie:11","rev":"1-457e7f2dccaa4df1af18c0c68ab68124"}, {"ok":true,"id":"movie:24","rev":"1-57e80ff1ac03ec3b90877f0ffb9aa152"}, {"ok":true,"id":"movie:38","rev":"1-a70f59ea f4aca0bb0d05e795532ddcd6"}, {"ok":true,"id":"movie:38","rev":"1-e09cc6ab2ad21677f0322bde5a0d"}, {"ok":true,"id":"movie:59","rev":"1-cebe99e31d4d4aa8f4da6e7166fa1c"}, {"ok":true,"id":"movie:62","rev":"1-5976007516532842641220"}, {"ok":true,"id":"movie:87","rev":"1-47a0375a1eb97552727ad253660aa5"}, {"ok":true,"id":"movie:85","rev":"1-47a0375a1eb97552727ad253660aa5"}, {"ok":true,"id":"movie:89","rev":"1-ca6983266376a8646cc4ca b5b794b"}, {"ok":true,"id":"movie:106","rev":"1-4da5e0a7bfbe521398363b4d8d98f"}, {"ok":true,"id":"movie:103","rev":"1-54017f6950e5a8cc1f64e814264579d"}, {"ok":true,"id":"movie:120","rev":"1-d5ead403c77dedec11cb444b916fa"}, {"ok":true,"id":"movie:123","rev":"1-71b8780ead6f7a328a4025-5e08c5a1"}, {"ok":true,"id":"movie:145","rev":"1-e98dc50ff36197febb7632g0744a4b"}, {"ok":true,"id":"movie:146","rev":"1-83942e42e4ca0b5e3e6ec784378"}]
```

| id           | key          | value   |
|--------------|--------------|---|
| movie:1098   | movie:1098   | {"rev": "1-e1e86f6473888147a84e7b88f5bd953"}  |
| movie:1018   | movie:1018   | {"rev": "1-17998a28a52347ddee2db2fc1c90e"}    |
| movie:10238  | movie:10238  | {"rev": "1-17998a28a52347ddee2db2fc1c90e"}    |
| movie:103    | movie:103    | {"rev": "1-17998a28a52347ddee2db2fc1c90e"}    |
| movie:10362  | movie:10362  | {"rev": "1-12476a5ab763ba2b71d881ded0c03676"} |
| movie:103731 | movie:103731 | {"rev": "1-7729353039c3090ba9c46dca729fea3"}  |
| movie:106    | movie:106    | {"rev": "1-4da5e0a7bfbe521398363b4d8d98f"}    |
| movie:10669  | movie:10669  | {"rev": "1-f5da7f5611a83ea96477f21cb172cff"}  |
| movie:10675  | movie:10675  | {"rev": "1-c966bfed817c98bbc64ed66f5bd41e5"}  |
| movie:10835  | movie:10835  | {"rev": "1-932a8a6be6cc34c8e94db99409c2c4c4"} |
| movie:10889  | movie:10889  | {"rev": "1-2c717a628cf82245a5a057ccc381b2a"}  |
| movie:1091   | movie:1091   | {"rev": "1-6d2a0227ef2154dd9bd763a0be2b94"}   |
| movie:10935  | movie:10935  | {"rev": "1-b7872a8d5743ad2f3129772e9e6b6003"} |
| movie:11     | movie:11     | {"rev": "1-457e7f2dccaa4df14f10c0c68ab68124"} |

## Exercices MapReduce avec CouchDB

### 1. Nombre total de films

#### Map:

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

#### Reduce:

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

| key   | value |
|-------|-------|
| total | 278   |

### 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     |

## 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 Celli         | 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": 253, "count": 4 } |
| Alien : Covenant                            | { "sum": 15, "count": 4 }  |

## 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    |

## 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 |
|---|-----------|-------------------------------|-------|--------|
| 1 | Action    | [["Al Pacino", "Diane Ven...  |       |        |
| 2 | Adventure | [["Billy Boyd", "Dominic ...  |       |        |
| 3 | Aventure  | [["Christopher Plummer", ...  |       |        |
| 4 | Comédie   | [["Charlie Chaplin", "Henr... |       |        |
| 5 | Comedy    | [["Annette Bening", "Dann...  |       |        |
| 6 | Crime     | [["Barbara Stanwyck", "E...   |       |        |
| 7 | Drama     | [["Charley Grapewin", "D...   |       |        |