

HAI914I : Apache CouchDB Suite

I.Mougenot

FDS UM

2023

Une vue Map/Reduce avec Fauxton

Edit View

Design Document ?

_design/D1 ▼

Index name ?

V2

Map function ?

```
1 - function (doc) {  
2   if (doc.type=='player' && doc.nationality)  
3     emit(doc.nationality, 1);  
4 }
```

Reduce (optional) ?

_count ▼

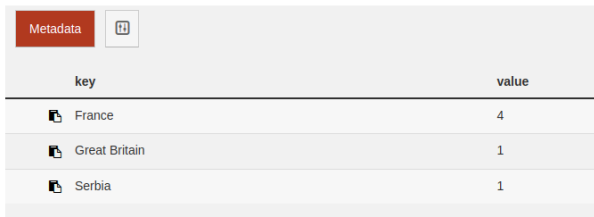
Figure: Vue V2 : clé nationalité et valeur 1 et fonction prédéfinie _count

Résultats de V2 sans faire jouer la fonction Reduce

key	value
France	1
France	1
France	1
France	1
Great Britain	1
Serbia	1

Figure: Résultats Map

Résultats de V2 avec fonction Reduce






Metadata	
Table	
key	value
 France	4
 Great Britain	1
 Serbia	1

Figure: Résultats avec Reduce

Nouveau Map/reduce avec clé composite

Edit View

Design Document ⓘ

_design/D1 ▼

Index name ⓘ

V3

Map function ⓘ

```
1- function (doc) {  
2-   if (doc.type == 'player' && doc.nationality)  
3-     emit([doc.gender, doc.nationality, doc._id], 1)  
4- }
```

Reduce (optional) ⓘ

_count ▼

☒ Save Document and then Build Index Cancel

Figure: clé tableau genre nationalité _id


V3 : plusieurs sorties



key	value
 ["F"]	2
 ["M"]	4

Figure: Résultats du Reduce niveau 1

V3 : plusieurs sorties







key	value
 ["F", "France"]	2
 ["M", "France"]	2
 ["M", "Great Britain"]	1
 ["M", "Serbia"]	1

Figure: Résultats du Reduce niveau 2


V3 : plusieurs sorties


key	value
["F", "France", "caroGa_93"]	1
["F", "France", "kikiMl_93"]	1
["M", "France", "JoTs"]	1
["M", "France", "laMonI"]	1
["M", "Great Britain", "murray"]	1
["M", "Serbia", "djoko"]	1


Figure: Résultats du Reduce niveau 3

Fonction Reduce personnalisée


Edit View

Design Document 

Index name 

Map function 

```
1 function (doc) { if (doc.type === 'player' && doc.age)
2   emit(doc.gender, doc.age);
3 }
```

Reduce (optional) 

Custom Reduce function

```
1 function (keys, values, rereduce) {
2   return Math.max.apply([], values);
3 }
```

Figure: renvoyer l'âge le plus élevé par genre

Résultat Reduce

key	value
 F	28
 M	36

Figure: renvoyer l'âge le plus élevé par genre

Manipuler les vues avec cURL

```
curl http://localhost:5984/tennis/_design/D1/_view/V2?reduce=false
```

```
curl http://localhost:5984/tennis/_design/D1/_view/V3?group_level=1
```

```
curl http://localhost:5984/tennis/_design/D1/_view/V3?group_level=2
```

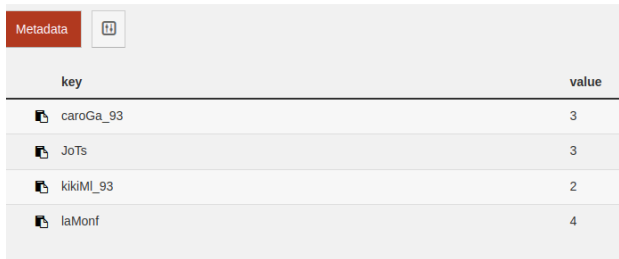
Différents appels de la même fonction

Valeurs composées

```
"victoires": {  
  "map": "function(doc)  
  { if(doc.type=='player' && doc.tournaments.length>0)  
    { for(var i in doc.tournaments) {var tournoi = doc.tournaments[i];  
      emit(doc._id,[tournoi.city, tournoi.year]);    } }",  
  "reduce": "_count"  
}
```

Les clés comme les valeurs peuvent être des objets JSON

Résultat après Reduce



The screenshot shows a web interface with a 'Metadata' tab selected. Below the tab is a table with two columns: 'key' and 'value'. The table contains four rows of data, each with a document icon next to the key. The keys are 'caroGa_93', 'JoTs', 'kikiMI_93', and 'laMonf'. The corresponding values are 3, 3, 2, and 4.





key	value
 caroGa_93	3
 JoTs	3
 kikiMI_93	2
 laMonf	4

Figure: renvoyer le nombre de tournois par joueur

Organiser le "sans schéma"

Bonnes pratiques : décider de l'organisation des documents en amont

- Utiliser `_id` comme seul identifiant du document (recours aux id naturels pour intégration éventuelle avec autres sources de données ou alors uuids)
- typer les documents (entités décrites : personne, voiture ...)
- introduire des estampilles temporelles : dates de création et de mise à jour
- utiliser des champs simples pour décrire l'entité cible et des champs composés pour les entités reliées

Organiser le "sans schéma"

réfléchir en amont à :

- est ce que l'on fait porter sur un seul type de document, toute la description (dénormalisation) ?
penser à la fréquence de mise à jour, notamment pour ce qui concerne les éléments pointés dans le document
- est ce qu'il vaut mieux décomposer la description dans plusieurs documents connexes (passage par référence) ?
penser à la capacité de combiner les données distribuées dans plusieurs documents

Exemple : passage par référence

```
{
  "_id": "murray",
  "firstname" : "Andy",
  "type" : "player",
  "nationality" : "Great Britain"
},
{
  "_id": "Great Britain",
  "population" : 60000000,
  "type" : "country"
},
```

Rapprocher informations nation et joueurs

"Collation view" pour contourner l'absence de jointure

```
function(doc) {  
  if (doc.type == "country") {  
    emit([doc._id, 0], [{"pop":doc.population}, {"players":doc.players}]);  
  } else if (doc.type == "player") {  
    emit([doc.nationality, 1], [{"name":doc.firstname}, {"rank":doc.ranking}]);  
  }  
}
```

Rapprocher informations nation et joueurs

Résultats Collation View

key	value
["France", 0]	[{ "pop": 50000000 }, { "players": "60000" }]
["France", 1]	[{ "name": "Jo" }, { "rank": 12 }]
["France", 1]	[{ "name": "Caroline" }, { "rank": 24 }]
["France", 1]	[{ "name": "Kristina" }, { "rank": 20 }]
["France", 1]	[{ "name": "Gael" }, { "rank": 7 }]
["Great Britain", 0]	[{ "pop": 60000000 }, { "players": "10000" }]
["Great Britain", 1]	[{ "name": "Andy" }, { }]
["Japan", 0]	[{ "pop": 20000000 }, { "players": "5000" }]
["Serbia", 0]	[{ "pop": 10000000 }, { "players": "5000" }]
["Serbia", 1]	[{ "name": "Novak" }, { "rank": 2 }]

Figure: Rapprochement dans l'espace

Retour sur le postulat CAP

Fig. extraite de <https://dev.to/katkelly/cap-theorem-why-you-can-t-have-it-all-gal>

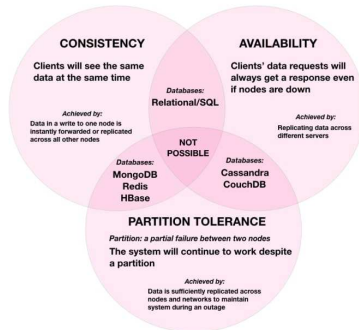


Figure: CAP, CouchDB et MongoDB

Partition et Réplication : deux mécanismes orthogonaux

- **Partitionnement** : BD découpée en blocs, distribués ensuite sur les nœuds du cluster avec des stratégies de partitionnement propres à chaque système (même si grandes tendances).
 - le partitionnement horizontal (ou sharding) : une référence au relationnel (partition au niveau des tuples (horizontal) / au niveau des colonnes (vertical))
 - partitionnement = élément essentiel à la scalabilité
- **Réplication** : les "shards" sont dupliqués et distribués sur les nœuds
 - améliore les accès et pas de SPOF
 - la réplication n'intervient pas dans la scalabilité

Pour faire simple

Partition et Réplication

- données **réparties** sur plusieurs machines : **résistance à la charge**
- données **répliquées** sur plusieurs machines : **résistance aux pannes**

CouchDB et le distribué

Éléments d'intérêt

- AP et éventuellement C (principes CAP)
- pas de SPOF (Single Point of Failure) : tous les nœuds peuvent jouer tous les rôles
- bien adapté à la "scalabilité horizontale" : ajout de nœuds facilités, mécanismes de vue, compression des données qui passent à l'échelle
- transparent pour les applications
- shard = ensemble de documents (placement à partir de l'ID sur lequel une fonction de hachage est appliquée)

Choix opérés

Deux grandes stratégies inspirées du système Amazon Dynamo

- partitionnement : "consistent hashing" (autre stratégie possible "range partitioning" comme pour HBase)
- réplication : "multi-master" (autre stratégie possible : "master-slave" comme pour HBase)

Serveurs distribués CouchDB (depuis 2.0)

<http://webdam.inria.fr/Jorge/html/wdmch21.html>

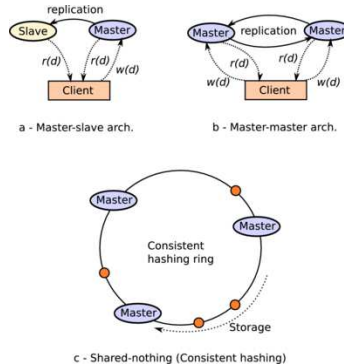


Figure: schémas de distribution

Placement par fonction de hachage

Stratégie de partitionnement et répliquation

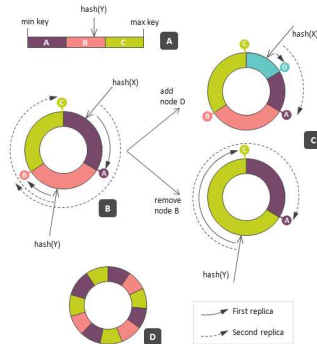
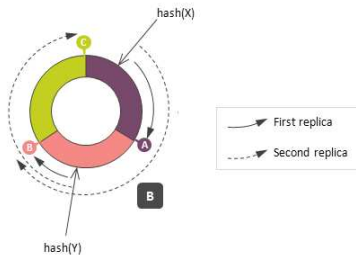


Figure: "hachage" de la clé et placement dans un shard répliqué

Anneau : espace de clés hachées séparé en intervalles

Nœuds qui se voient assigner un objet selon la valeur de la clé hachée, un nœud est responsable de la région qui le sépare de son prédécesseur, Nœuds qui se voient répliquer un objet



Seuls les nœuds voisins sont impactés par l'ajout / suppression de nœuds dans le cluster

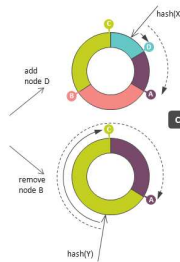


Figure: Redimensionnement/panne du système

Paramètres clés pour le cluster

Paramètres du cluster :

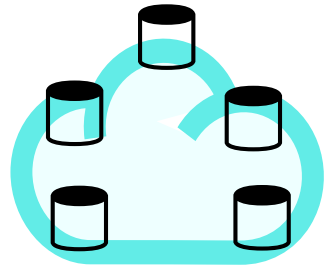
Nodes : nombre de noeuds du système

N : nombre de copies/réplicas des données

Q : nombre de shards (partitions horizontales) pour la BD

R : quorum de noeuds à consulter en lecture
avant réponse au client

W : quorum de noeuds à consulter en écriture
avant réponse au client



cluster de machines

Principes et terminologie : Article Amazon Dynamo 2007

Figure: Le système QNRW

Un exemple

Paramètres du cluster couchdb : exemple

Nodes : nombre de noeuds du système : ici 3

N : par défaut 3 (si N=1 pas de tolérance aux pannes)

Q : par défaut 8

8 * 3 : 24 shards à placer sur 3 noeuds => 8 shards / noeud

R : en général $(N+1)/2$ ici 2

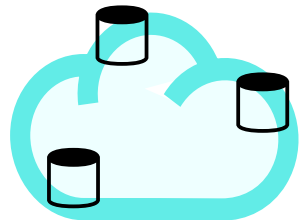
si R=1 diminue l'attente

si R=N maximise la cohérence de la lecture

W : en général $(N+1)/2$ ici 2

si W=1 maximise la bande passante

si W=N maximise la cohérence de l'écriture



cluster de machines

Figure: Principes de placement des partitions

Instructions pour le distribué et autres

```
-- les noeuds du systeme
curl -s $COUCH3/_membership | jq
-- avoir les infos sur le choix systeme QNWR
curl -s $COUCH3/tennis

-- enregistrement et partitions
curl -s $COUCH3/tennis/_shards/caroGa_93 | jq

-- les partitions exploitees par la base
curl -s $COUCH3/tennis/_shards | jq

-- pour pb cache et formatage
curl -H "Cache-Control: no-cache" -s $COUCH3/tennis | jq

-- dernieres revisions des documents
curl -s $COUCH3/tennis/_changes

-- informations sur les revisions d'un document
curl -s $COUCH3/tennis/caroGa_93?revs_info=true
```

Exemple au niveau BD

```
tsabelle.mougenot@umontpellier.fr@prodpeda-x2go-focal1:~$ curl -s $COUCH3/tennis | jq
{
  "db_name": "tennis",
  "purge_seq": "0-g1AAAACXeJzLYWBgYHpgTmEwTM4vTc5ISXIoKMpPKUHNsdSFChjrGull5qUVJeqV5ubnlRSk5uRkphbpbRXlgLTm",
  "update_seq": "0-g1AAAACXeJzLYWBgYHpgTmEwTM4vTc5ISXIoKMpPKUHNsdSFChjrGull5qUVJeqV5ubnlRSk5uRkphbpbRXlgLTm",
  "sizes": {
    "file": 16764,
    "external": 0,
    "active": 0
  },
  "props": {},
  "doc_del_count": 0,
  "doc_count": 0,
  "disk_format_version": 8,
  "compact_running": false,
  "cluster": {
    "q": 2,
    "n": 3,
    "w": 2,
    "r": 2
  },
  "instance_start_time": "0"
}
```

Figure: Parmi les informations générales

Exemple au niveau BD

Usage de la réplication différent, il s'agit de disposer de manière unitaire différentes fragments de BD qui sont distribués (BD fédérées)

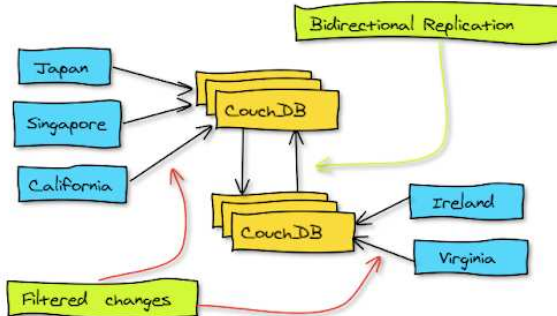


Figure: BD Fragmentée

Les primitives associées à la réplication (ici unidirectionnelle)

```
curl -X POST http://localhost:5984/_replicate -d '{"source":"test_db","target":"replika", "continuous":true}' -H 'Content-Type: application/json'
```

Répliquer la BD dans son ensemble, créer au préalable replika, "continuous" pour synchroniser les mises à jour dans la foulée

Conflits liés à la réplication d'une BD ou de fragment de base

<http://webdam.inria.fr/Jorge/html/wdmch21.html>

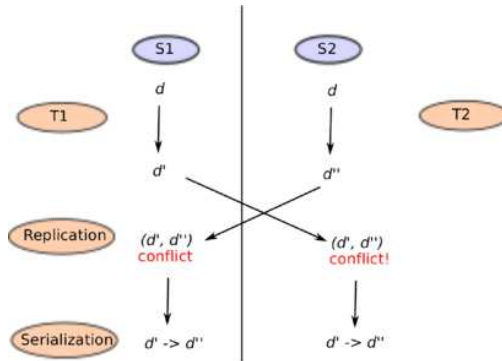


Figure: Mises à jour conflictuelles