

HAI914I : Apache CouchDB

I.Mougenot

FDS UM

2024

Typologie des systèmes NoSQL

Au regard du mode de représentation choisi

- principe de base : clé/valeur
 - **Systèmes clé/valeur distribués**
 - **Systèmes orientés colonne**
 - **Systèmes orientés document**
- **Systèmes orientés graphe**
 - dans un certaine mesure les triplestores et les SGBDOO

Agrégats : unités naturelles pour le distribué

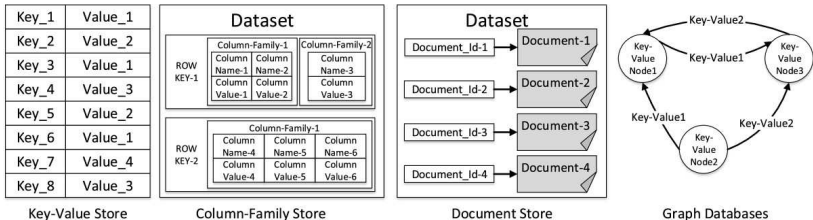


Figure: Extrait de K. Grolinger et al, 2013

Agrégat : collection d'objets de domaine liés par une entité racine

Agrégat (pattern Domain Driven Design (Martin Fowler))

Disposer d'une unité d'information complexe qui est traitée, stockée et échangée de façon atomique, et qui n'est référencée que par sa racine = clé (importance de la clé)

Pattern Domain Driven Design (DDD) Aggregate (Martin Fowler, NoSQL distilled, 2012), inspiré du livre de Eric Evans (Domain-Driven Design 2003)

Définition : semi-structuré

Repris de Dan Suciu, Encyclopedia of Database Systems, 2009

The semi-structured data model is designed as an evolution of the relational data model that allows the representation of data with a **flexible structure**. Some items may have **missing attributes**, others may have **extra attributes**, some items may have **two or more occurrences of the same attribute**. **The type of an attribute is also flexible**: it may be an atomic value or it may be another record or collection. Moreover, collections may be heterogeneous, i.e., they may contain items with different structures. The semi-structured data model is a **self-describing data model**, in which the data values and the schema components co-exist.

Langages support du "semi-structuré"

Parmi ces langages

- Langages de balisage
 - XML (eXtended Markup Language) : très orienté balise (tag)
- Formats de données
 - JSON (Javascript Object Notation) : orienté données
 - YAML (Yet Another Markup Language) : orienté configuration et lisibilité

JSON vs XML

JSON : les annotations sur les arêtes et les valeurs sur les nœuds

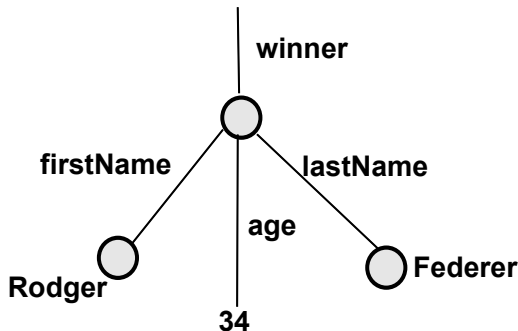


Figure: La structuration JSON

XML vs JSON

XML : les annotations et les valeurs sont sur les nœuds

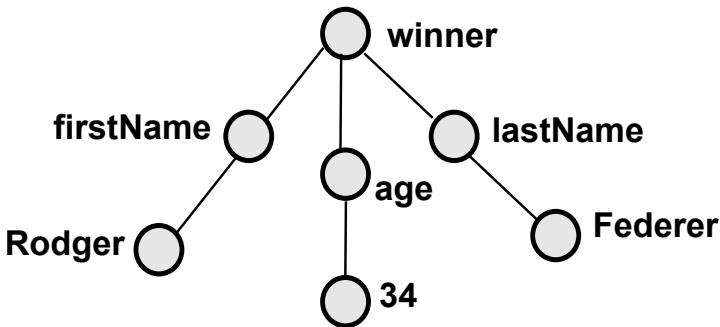


Figure: La structuration XML

JSON, JavaScript Object Notation

JSON (Notation Objet issue de JavaScript) : format d'échange de données notoirement simple

- exploitable par les machines et plutôt lisible par les humains
- sous-ensemble de JavaScript mais indépendant de tout langage
- en train de s'imposer comme le langage du Web : format de sérialisation des objets Javascript (mais pas que)

JSON, deux principes structuraux

La paire nom (ou clé)/valeur et le tableau de valeurs

- collection de paires nom/valeur réifiée selon les langages comme un objet, un enregistrement, ou une autre structure (comme une table associative) ;
- liste de valeurs ordonnées réifiée selon les langages comme un tableau, un vecteur, une liste ou une suite.

JSON, des exemples d'objets simples à composés

Ensemble de paires nom/valeur

- objet simple : `"lastName": "Federer"` ou encore `"age": 40`
- objet composé : ensemble de couples non ordonnés :
`{"lastName": "Federer", "firstName": "Rodger", "age": 40 }`
- objet composé : `"winner" : {"lastName": "Federer",
"firstName": "Rodger", "age": 40 }`

JSON, tableaux de valeurs

- valeur composée : un tableau est une collection ordonnée de valeurs qui peuvent ne pas être de même type
- "players" : ["Ozaka", "Garcia", "Sinner", "Alcaraz"]
- "people" : [{"name": "Nadal", "age": 36}, {"name": "Federer", "age": 40, "gender": "male"}, {"name": "Ferro", "age": 24}]

JSON, exemple d'objets composés

Vision document

```
{  
  "tournament": "The French Open",  
  "year": 2015,  
  "director": {"lastName": "Ysern",  
    "firstName": "Gilbert"},  
  "players": ["Williams", "Nadal", "Djokovic", "Murray",  
    "Simon", "Tsonga", "Cornet"]  
}
```

JSON, des formats dédiés

Pour l'échange et le traitement a posteriori de données thématiques

GeoJSON

BioJSON

TopoJSON

...

GeoJSON : structure pour les entités spatiales

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [2.25, 48.84]
  },
  "properties": {
    "name": "The French Open", "year": 2015, "director": {"lastName": "Ysern", "
      firstName": "Gilbert"},
    "players": [ "Williams", "Nadal", "Murray", "Simon", "Tsonga", "Cornet" ]
  }
}
```

Listing 1: Roland Garros en GeoJSON

Faciliter les traitements au travers du format dédié

Visualisation du contenu du fichier GeoJSON sous QGIS

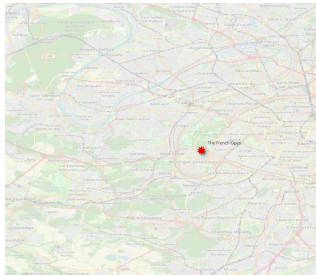


Figure: Site de Roland Garros Porte d'Auteuil Paris (fond de carte OSM)

les systèmes NoSQL à document

Les plus en vue

MongoDB (exploite BSON - binary JSON)

Apache CouchDB (inspiré par Lotus Notes, logiciel pour le travail collaboratif dans les entreprises)

CouchBase (dérivé de CouchDB)

IBM Cloudant

Realm (pour les BD embarquées, rachat par MongoDB)

...

CouchDB

Cluster of Unreliable Commodity Hardware

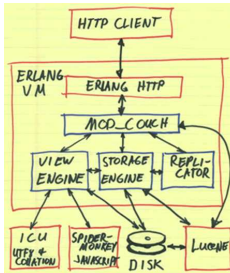


Figure: Architecture générale

Apache CouchDb : une double orientation

- Système de données orienté documents
 - un document est une unité informationnelle autonome et composite : textuel mais aussi image, son, ...
 - fonctionnalités attendues : gestion de versions, évolution et restructuration, réplication, synchronisation
- Système de données orienté web
 - un document est une ressource web (toute chose concrète ou abstraite, susceptible d'être identifiée, et qui peut être manipulée au travers de diverses représentations)
 - défini au travers d'une URI
 - manipulable au travers d'une architecture REST et du protocole HTTP

Principes CouchDB

Extrait de <http://b3d.bdpedia.fr/bddoc.html>

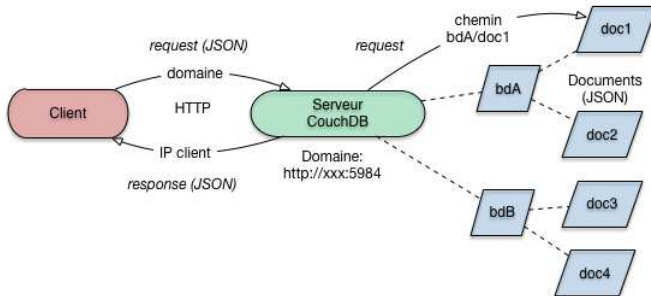


Figure: Principes Généraux CouchDB

La vision document

Modèle de données à base de graphe avec des structures très flexibles

- auto-description des données
- fort pouvoir d'expressivité : le contenu se décrit avec des agrégats de listes, d'ensembles et d'enregistrements
- typage et structure flexibles : données pouvant être typées et/ou structurées (contraintes a priori ou contrôle a posteriori)
- sérialisation : le contenu avec sa structure doit pouvoir être publié sous la forme d'une chaîne de caractères

Efficacité et simplicité des principes REST (Representational State Transfer)

- des échanges type web services pour créer, accéder ou gérer des ressources (exploitation des URI à cet effet)
 - GET : retourner une ressource
 - PUT : créer / mettre à jour une ressource
 - POST : créer une ressource à attacher à la collection, et faire évoluer une ressource existante par envoi de message (annoter, envoi de données à un processus)
 - DELETE : supprimer la ressource
- Système de données orienté web
 - une interface web est seule nécessaire, disposer de librairies clients est un plus

Exemples d'interaction avec le client REST CURL (Client URL Library)

```
-- server banner
$ curl -X GET localhost:5984
{"couchdb":"Welcome","version":"3.3.3", ..., "features":["access-ready","partitioned",
  "pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"The
  Apache Software Foundation"}}

-- create a db= put a resource (authentification required)
$ curl -X PUT admin:pwd@localhost:5984/db_test
{"ok":true}

-- call to test_db
$ curl -X GET admin:pwd@localhost:5984/db_test
{"instance_start_time":"1727864848","db_name":"db_test","purge_seq" .....}

-- listing of all databases
$ curl -X GET admin:pwd@localhost:5984/_all_dbs
["db_test","exam","tennis",...]
```

Propriétés d'un document : _id et _rev

_id : clé ou racine de l'agrégat, _rev : plusieurs versions du document peuvent co-exister dans la base de données

```
--putting a resource in a db
$ curl -X PUT admin:pwd@localhost:5984/db_test/o1 -d '{"name":"Nadal"}'
{"ok":true,"id":"o1","rev":"1-f28a4a5baf607e908cea5863c324d147"}

--getting the document using its URI:
$ curl -X GET admin:pwd@localhost:5984/db_test/o1
{"_id":"o1","_rev":"rev":"1-f28a4a5baf607e908cea5863c324d147","name":"Nadal"}
```


Différentes versions d'un même document

Tentative de mise à jour d'un document

```
--updating a doc : failure
$ curl -X PUT admin:pwd@localhost:5984/db_test/o1 -d '{"name":"Nadal", "age":36}'
{"error":"conflict","reason":"Document update conflict."}

--updating = adding a new version.
-- multi-version concurrency control protocol which requires the current version
  number:
$ curl -X PUT admin:pwd@localhost:5984/db_test/o1 -d '{"_rev":"1-f28a4a5baf607e908cea5863c324d147","name":"Nadal", "age":36}'
{"ok":true,"id":"o1","rev":"2-b3a6b46d95c7671046d98826c2010595"}
```

Illustration Contrôle de concurrence multi-version

Extrait de <http://b3d.bdpedia.fr/bddoc.html>, pré suppose de pouvoir conserver au moins 2 versions à la fois



Figure: MVCC et verrouillage pessimiste

Obtenir un UUID, lister les documents et versions d'un document

```
--get an Universally Unique Identifier (uuid) for document's identification
$ curl -X GET admin:pwd@localhost:5984/_uuids
{"uuids":["415da9c4f3bd245ea2fbbeaab600057c"]}

--display documents' metadata
$ curl admin:pwd@localhost:5984/db_test/_all_docs

--display information about revision
$ curl -X GET admin:pwd@localhost:5984/db_test/o1?revs=true
```

charger un document : fichier sans identifiant

```
{
  "lastname" : "Tsonga",
  "firstname" : "Jo",
  "type" : "player",
  "age" : 37,
  "tournaments": [
    {
      "city": "Marseille",
      "year": 2017
    }
  ],
  "nationality" : "France"
}
```

Listing 2: fichier tsonga.json

charger un document : fichier avec identifiant

```
{  
  "_id": "laMonf",  
  "lastname" : "Monfils",  
  "firstname" : "Gael",  
  "type" : "player",  
  "age" : 36,  
  "ranking" : 57,  
  "nationality" : "France"  
}
```

Listing 3: fichier monfils.json

Exploiter les fichiers précédents

Les appels passent par PUT ou POST selon le choix fait autour de l'identification du document

```
--explicit id : PUT
$ curl -X PUT admin:pwd@localhost:5984/db_test/o5 -d @tsonga.json -H "Content-Type:
  application/json"
{"ok":true,"id":"o5","rev":"1-87f50d9c01b37db251d7e12a8ad0ce69"}

-- id within the document : POST
$ curl -X POST admin:pwd@localhost:5984/db_test -d @monfils.json -H "Content-Type:
  application/json"
{"ok":true,"id":"laMonf","rev":"1-341422aac64aa47ff3a933bb9d62c5b1"}
```

chargement par lots : fichier intégrant plusieurs documents

Ici identification requise au sein de chaque document

```
{
  "docs": [
    { "_id": "murray",
      "type" : "player",
      "nationality" : "Great Britain"
    },
    { "_id": "djoko",
      "lastname" : "Djokovic",
      "type" : "player",
      "ranking" : 2,
    }
  ]
}
```

Listing 4: fichier murrayDjoko.json

Documents de la base, ajout par lots

```
--adding documents : fail = content type is required
$ curl -d @murrayDjoko.json -X POST admin:pwd@localhost:5984/db_test/_bulk_docs

--correct
$ curl -X POST admin:pwd@localhost:5984/db_test/_bulk_docs -d @murrayDjoko.json -H "
  Content-Type: application/json"
[{"ok":true,"id":"murray","rev":"1-7d4b633c14d3b66fd2c333947627f7ef"}, {"ok":true,"id":
  "djoko","rev":"1-0620108d9d04bbaa9b55c826664a244d"}]
```


Suppression d'un document ou d'une base de données

Paradoxalement, plus facile de supprimer une base de données

```
-- document deleting (last version) A new version has been created = logical deletion
$ curl -X DELETE admin:pwd@localhost:5984/db_test/o1?rev=2-8
e0031775b443f73681b8c2566895f8b
{"ok":true,"id":"o1","rev":"3-c3e4361ab165df874c0ee1d92966d8de"}

-- drop database
$ curl -X DELETE admin:pwd@localhost:5984/other_db
{"ok":true}
```

CouchDb : La gestion de documents

- document = objet JSON de taille arbitraire
- Chaque document possède un identifiant (`_id`) et un numéro de version (revision number : `_rev`) amené à changer lors d'une modification
- Des fonctions de validation peuvent venir valider l'insertion ou la modification de documents (type-checking).
- Une vue est une nouvelle collection de paires clé-document (spécification Map/Reduce)
- Un document peut être répliqué sur d'autres instances CouchDb

Le requêtage et la notion de vue

Une vue s'apparente à un document JSON d'une tâche Map/Reduce définie en Javascript, et son résultat est aussi un document JSON

Deux types de vues

- vue permanente (intégrée au sein d'une collection dénommée "design document") : matérialisée et indexée sur la clé à l'aide d'une structure B+Tree
- vue temporaire : calculée à la volée (possiblement inefficace)

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



Organisation Générale CouchDB

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

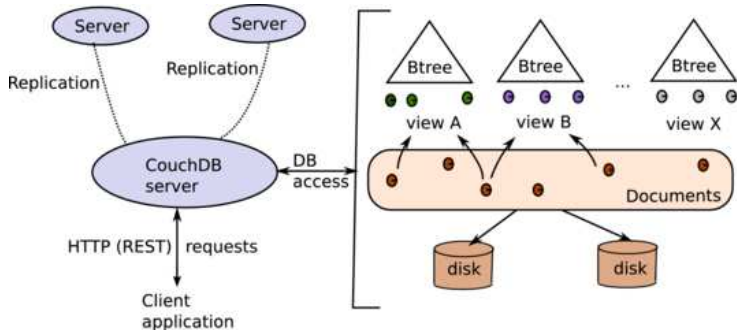


Figure: Vue générale d'un serveur CouchDB

Applicatif Web Fauxton

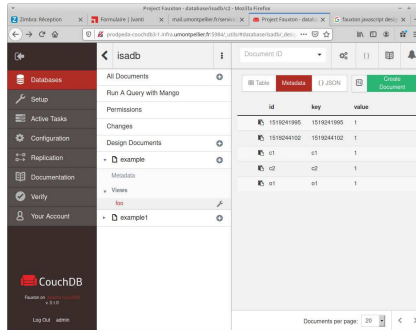


Figure: Exemple d'interface Web Fauxton (CouchDB 3.1)

Rappel Principes Map-Reduce

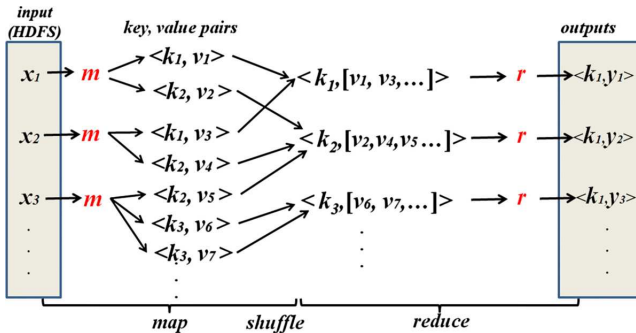


Figure: Map-Reduce : modèle programmation distribuée

Exemple 1 (juste une fonction MAP)

Renvoyer tous les documents de la base

```
{
  "views": {
    "all": {
      "map": "function(doc) { emit(null, doc.name) }",
    }
  }
}
```

Listing 5: MAP Function

Exemple 2 (également une fonction MAP)

Identique à la fonction prédéfinie `_all_docs`

```
{
  "views": {
    "allDocs": {
      "map": "function(doc) { emit(doc._id, {"rev" : doc._rev}); }",
    }
  }
}
```

Listing 6: MAP Function

Exemple 3 (juste MAP)

Index sur clé (firstname) pointant sur tuples valeurs (nationality)

```
{
  "views": {
    "players": {
      "map": "function(doc) { if (doc.type=='player') {emit(doc.firstname, doc.
        nationality);}}",
    }
  }
}
```

La vue players une fois enregistrée (au sein du "document design") peut ensuite être rappelée

Exemple très simplifié B+Tree

B+Tree
clés index
Clé, valeur, donnée

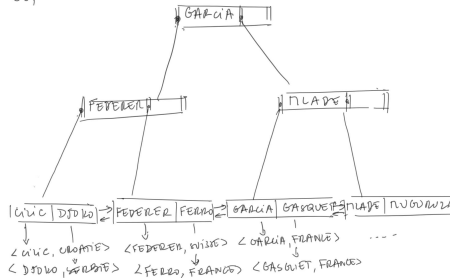


Figure: Illustration B+Tree pour cette vue

Organisation des vues au sein de documents

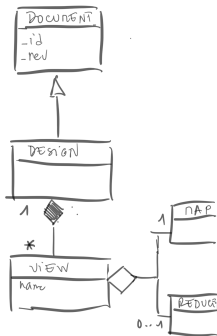


Figure: Design Document

Détails Map : exemple document joueur

```
isabelle.mougenot@unontpellier.fr@prodpeda-x2go-focal1:~/Desktop/BasesDeDonnees/CouchDB$ curl -s $COUCH3/tennis/caroGa_93 | jq
{
  "_id": "caroGa_93",
  "_rev": "1-62098e47935455bd782a636f08629088",
  "lastname": "Garcia",
  "firstname": "Caroline",
  "type": "player",
  "age": 28,
  "gender": "F",
  "ranking": 24,
  "coach": {
    "nom": "Dupont",
    "prenom": "Anne"
  },
  "nationality": "France",
  "spokenLanguages": [
    "fr",
    "en",
    "sp"
  ],
  "tournaments": [
    {
      "city": "Bogota",
      "year": 2012
    },
    {
      "city": "Llinoges",
      "year": 2014
    },
    {
      "city": "Strasbourg",
      "year": 2015
    }
  ]
}
```

Figure: Structure générale d'un document

Détails Map : code Javascript

clé : une ville de tournoi une année donnée, valeur : 1

```
l.sab@lille.mougenot@umontpellier.fr@prodpeda-x2go-focal1:~/desktop/basesbebonnees/CouchDB/exemples_2021$ curl -s $COUCH3/tennts/_design/tbls/  
{  
  "id": "_design/tbls",  
  "rev": "1-8fa26f9880695c42c7ff1994533eb7af",  
  "views": {  
    "Tournaments": {  
      "map": "function(doc) { if (doc.type=='player' && doc.tournaments) for (var t in doc.tournaments) { emit(doc.tournaments[t], 1);}}"  
    }  
  }  
}
```

Figure: Design Document

Une vue Map : résultats

```
isabelle.mougenot@umontpellier.fr@prodpeda-x2go-focal1:~/Desktop/BasesDeDonnees/CouchDB/Exemples_2021$ curl -s $COUCH3/te
{"total_rows":12,"offset":0,"rows":[
{"id":"JoTs","key":{"city":"Anvers","year":2017},"value":1},
{"id":"caroGa_93","key":{"city":"Bogota","year":2012},"value":1},
{"id":"JoTs","key":{"city":"Cassis","year":2019},"value":1},
{"id":"laMonf","key":{"city":"Doha","year":2018},"value":1},
{"id":"kikiML_93","key":{"city":"Hong Kong","year":2016},"value":1},
{"id":"caroGa_93","key":{"city":"Limoges","year":2014},"value":1},
{"id":"JoTs","key":{"city":"Montpellier","year":2019},"value":1},
{"id":"laMonf","key":{"city":"Montpellier","year":2020},"value":1},
{"id":"laMonf","key":{"city":"Rotterdam","year":2019},"value":1},
{"id":"laMonf","key":{"city":"Rotterdam","year":2020},"value":1},
{"id":"caroGa_93","key":{"city":"Strasbourg","year":2015},"value":1},
{"id":"kikiML_93","key":{"city":"Taipei","year":2012},"value":1}
]}
```

Figure: Résultats

Détails Map/reduce : code Javascript

map : clé : une ville de tournoi, valeur : 1

reduce : compter par ville les joueurs

```
(base) isa@Faraman:~/Enseignement/BasesDeDonnees/NoSQL/SupportsCours/HAI914I/Couch_22$ curl -X GET http://admin:isa@localhost:5984/tennis/_design/tournois | jq
{
  "_id": "_design/tournois",
  "_rev": "9-186306420adc4e4a58cd77e8c1126351",
  "views": {
    "cityYear": {
      "map": "function (doc) { if (doc.tournaments) { for (var i in doc.tournaments){\n  emit([doc.tournaments[i].city, doc.tournaments[i].year,], 1); }\n}",
      "reduce": "_count"
    },
    "city": {
      "map": "function (doc) { if (doc.tournaments) { for (var i in doc.tournaments){\n  emit(doc.tournaments[i].city, 1); }\n}",
      "reduce": "_count"
    }
  },
  "language": "javascript"
}
```

Figure: Design Document

Détails Map/reduce : Resultats

```
curl -X GET admin:pwd@localhost:5984/tennis/_design/tournois/_view/city?group_level=1
```

```
base) isa@Faraman:~/Enseignement/BasesDeDonnees/NoSQL/SupportsCours/HAI914I/Couch_22$ curl -X GET admin:isa@localhost:5984/tennis/_design/tournois/_view/city?group_level=1
{"rows":[{"key":"Anvers","value":1},
{"key":"Bogota","value":1},
{"key":"Cassis","value":1},
{"key":"Doha","value":1},
{"key":"Hong Kong","value":1},
{"key":"Limoges","value":1},
{"key":"Montpellier","value":2},
{"key":"Rotterdam","value":2},
{"key":"Strasbourg","value":1},
{"key":"Taipei","value":1}
]}
base) isa@Faraman:~/Enseignement/BasesDeDonnees/NoSQL/SupportsCours/HAI914I/Couch_22$
```

Figure: Resultats Reduce

Avec REDUCE et cURL : autre exemple

Contenu du fichier tennismen.js

```
{ "_id": "_design/allTennismen", "language": "javascript", "views": { "allT": { "map": "function (doc) {\n  if (doc.type=='player') {emit(doc._id,{\"nationality\" : doc.nationality});}\n}\n", "reduce": "_count" } }
```

```
curl -X PUT http://localhost:5984/test_db/_design/allTennismen -d tennismen.js -H "Content-Type: application/json"
```

Avec cURL : résultats

```
curl -X GET http://localhost:5984/test_db/_design/allTennismen/_view/allT?reduce=false
```

```
{ "total_rows": 6, "offset": 0, "rows": [
  { "id": "caroGa_93", "key": "caroGa_93", "value": { "nationality": "France" } },
  { "id": "djoko", "key": "djoko", "value": { "nationality": "Serbia" } },
  { "id": "JoTs", "key": "JoTs", "value": { "nationality": "France" } },
  ... ] }
```

La vue nommée allT retourne la nationalité des joueurs

Construire une vue avec cURL

```
curl -X POST -d '{
  "map":"function(doc){ { emit(doc.nationality, 1) } }",
  "reduce":"function(keys, values){ return sum(values) }" }' -H 'Content-Type:
  application/json' 'http://localhost:5984/test_db/_temp_view?group=true'
```

```
retourne :
{"rows":[
{"key":"France","value":4},
{"key":"Great Britain","value":1},
{"key":"Serbia","value":1}
]}
```

Un exemple Map/Reduce Complet

Vue by_country : renvoyer le nombre de joueurs par pays

```
{
  "_id": "_design/tennismen",
  "language": "javascript",
  "views": {
    "by_country": {
      "map": "function(doc) { \n if (doc.nationality)  { emit(doc.nationality,
        1);}\n}",
      "reduce": "function(keys, values) {\n    return sum(values);\n}\n"
    }
  }
}
```

Un exemple Map/Reduce Complet

Vue by_country : exploiter la clé

```
$ curl 'http://.../_design/tennismen/_view/by_country?group_level=1'
{"rows": [
  {"key": "France", "value": 3},
  {"key": "Great_Britain", "value": 1},
  {"key": "Serbia", "value": 1}]}
$ curl 'http://.../test_db/_design/tennismen/_view/by_country'
{"rows": [
  {"key": null, "value": 5}]}

```

cURL et sans le reduce

```
curl 'http://127.0.0.1:5984/test_db/_design/tennismen/_view/by_country?key="France"&
    reduce=false'
{"total_rows":5,"offset":0,"rows":[
{"id":"laMonf","key":"France","value":1},
{"id":"o5","key":"France","value":1},
{"id":"o6","key":"France","value":1}
]}
```

Différents niveaux d'agrégats (1)

```
{
  "_id": "_design/tennis",
  "_rev": "2-4c82905cc71abedd3671d8de02c26ed8",
  "language": "javascript",
  "views": {
    "TestReduce": {
      "map": "function (doc) { if (doc.type=='player')\n{   emit([doc.gender, doc\n                .nationality, doc._id], 1);}}\n",
      "reduce": "_count"
    }
  }
}
```


Différents niveaux d'agrégats (2)

```
-- all keys
curl 'http://127.0.0.1:5984/urbain/_design/tennis/_view/TestReduce'
-- group on first attribute of the key
curl 'http://127.0.0.1:5984/urbain/_design/tennis/_view/TestReduce?group_level=1'
-- group on first and second attributes of the key
curl 'http://127.0.0.1:5984/urbain/_design/tennis/_view/TestReduce?group_level=2'
...
```

Différents niveaux d'agrégats (3)

```
$ curl 'http://.../_view/TestReduce'
{"rows":[
{"key":null,"value":6}
]}
$ curl 'http://.../_view/TestReduce?group_level=1'
{"rows":[
{"key":["F"],"value":2},
{"key":["M"],"value":4}
]}
$ curl 'http://.../_view/TestReduce?group_level=2'
{"rows":[
{"key":["F","France"],"value":2},
...]}
```