



RAPPORT BIG DATA

**MEDDAH Amine
FEGHOUL Ghiles**

**M2 GIL
TP A**

**Année Universitaire
2017/2018**

Introduction	1
Environnement et Pré-requis	2
Les données	2
I- MongoDB	6
Installation :	6
Execution :	7
Page d'accueil	8
Insertion des Données	10
Importation de fichiers	10
Select, Update et Delete des Données	11
Architecture MongoDB :	12
Architecture multimodèle MongoDB	13
Indexation MongoDB :	13
Création d'un index	13
Types d'index	13
II- CouchDB	15
Installation :	15
Création d'un compte Admin:	17
Création d'une Base de données:	19
Insertion des données :	20
Utilisation CURL :	23
Architecture CouchDB :	25
Indexation CouchDB :	26
Vue	26
Type Vues :	27
Qui utilise ces bases de données?	28
Systèmes d'exploitation du serveur ?	28
Conclusion	28

Introduction

Le but de ce projet est de réaliser un comparatif entre 2 systèmes de gestion des bases de données NoSQL orientées documents, en l'occurrence **MongoDB** et **CouchDB** (il en existe bien évidemment d'autres mais on a choisi ces 2 là car c'est les plus répandus sur le marché).

La comparaison se fera au niveau performance calculée notamment lors des différentes opérations de manipulations des données et qui est exprimée par le temps de réponse de chaque moteur de base de données pour chaque requête émise.

On commencera d'abord ce rapport par les pré-requis nécessaires pour le fonctionnement et la mise en place des 2 moteurs ainsi que l'environnement sur lequel on les a utilisés, ensuite, on enchaîne avec la première étude qui concerne MongoDB avec notamment les différentes étapes allant de l'installation jusqu'à l'exploitation des données, puis on passe à la deuxième étude qui concerne CouchDB où l'on effectuera aussi l'explication des mêmes étapes que précédemment, et finalement on terminera par notre conclusion tirée de cette étude.

Environnement et Pré-requis

Les données

On crée un programme **java (JAR)** qui permet de parser les données extraites depuis le site format fichier XML et les transformés en format Json, Ce programme lit les fichiers fournis en entrée et les convertit selon les formats sélectionnés en option. pour qu'on puissent les utiliser dans MongoDB et CouchDB comme le montre la figure suivante :

1- parser le fichier récupéré en plusieurs fichiers **XML**. Comme Suit :






```
C:\Users\QLF\Downloads> Useful-Projects-master\Useful-Projects-master\ArticleParser\demo>java -jar parser.jar test.xml test2.xml
Début du parsing des documents !
-----
>> Parsing de test2.xml ...
>> >> Fichier ./xml/article_10022410.xml généré
>> Document test2.xml parsé ! (12 ms)
-----
>> Parsing de test.xml ...
>> >> Fichier ./xml/article_29427970.xml généré
>> >> Fichier ./xml/article_29427917.xml généré
>> >> Fichier ./xml/article_29427743.xml généré
>> >> Fichier ./xml/article_29427516.xml généré
>> >> Fichier ./xml/article_29427485.xml généré
>> >> Fichier ./xml/article_29427481.xml généré
>> >> Fichier ./xml/article_29427444.xml généré
>> >> Fichier ./xml/article_29427399.xml généré
>> >> Fichier ./xml/article_29427281.xml généré
>> >> Fichier ./xml/article_29427233.xml généré
>> >> Fichier ./xml/article_29427232.xml généré
>> >> Fichier ./xml/article_29427231.xml généré
>> >> Fichier ./xml/article_29427230.xml généré
>> >> Fichier ./xml/article_29427049.xml généré
>> >> Fichier ./xml/article_29426734.xml généré
>> >> Fichier ./xml/article_29426719.xml généré
>> >> Fichier ./xml/article_29426583.xml généré
>> >> Fichier ./xml/article_29426508.xml généré
>> >> Fichier ./xml/article_29426385.xml généré
>> >> Fichier ./xml/article_29426333.xml généré
>> Document test.xml parsé ! (44 ms)
-----
Fin du parsing des documents !
>> Nombre de fichiers parsés : 2
>> Temps total : 85 ms
```






















2- génération des fichiers json à partir des fichier XML généré précédemment, comme suit :

```
C:\Users\QLF\Downloads> Useful-Projects-master\Useful-Projects-master\ArticleParser\demo>java -jar parser.jar -f=json test.xml test2.xml
Début du parsing des documents !
-----
>> Parsing de test2.xml ...
>> >> Fichier ./json/article_10022410.json généré
>> Document test2.xml parsé ! (58 ms)
-----
>> Parsing de test.xml ...
>> >> Fichier ./json/article_29427970.json généré
>> >> Fichier ./json/article_29427917.json généré
>> >> Fichier ./json/article_29427743.json généré
>> >> Fichier ./json/article_29427516.json généré
>> >> Fichier ./json/article_29427485.json généré
>> >> Fichier ./json/article_29427481.json généré
>> >> Fichier ./json/article_29427444.json généré
>> >> Fichier ./json/article_29427399.json généré
>> >> Fichier ./json/article_29427281.json généré
>> >> Fichier ./json/article_29427233.json généré
>> >> Fichier ./json/article_29427232.json généré
>> >> Fichier ./json/article_29427231.json généré
>> >> Fichier ./json/article_29427230.json généré
>> >> Fichier ./json/article_29427049.json généré
>> >> Fichier ./json/article_29426734.json généré
>> >> Fichier ./json/article_29426719.json généré
>> >> Fichier ./json/article_29426583.json généré
>> >> Fichier ./json/article_29426508.json généré
>> >> Fichier ./json/article_29426385.json généré
>> >> Fichier ./json/article_29426333.json généré
>> Document test.xml parsé ! (100 ms)
-----
Fin du parsing des documents !
>> Nombre de fichiers parsés : 2
>> Temps total : 190 ms
```






















Résultat:

Dans notre cas le fichier récupérer dans le site d'appelle **test.xml** :

 json	14/02/2018 16:08	Dossier de fichiers	
 xml	14/02/2018 16:06	Dossier de fichiers	
 parser.jar	13/02/2018 22:54	Executable Jar File	75 Ko
 test.xml	13/02/2018 22:54	Fichier XML	166 Ko
 test2.xml	13/02/2018 22:54	Fichier XML	11 Ko

<input type="checkbox"/> Nom	Modifié le	Type	Taille
 article_10022410.xml	14/02/2018 16:06	Fichier XML	11 Ko
 article_29426333.xml	14/02/2018 16:06	Fichier XML	9 Ko
 article_29426385.xml	14/02/2018 16:06	Fichier XML	8 Ko
 article_29426508.xml	14/02/2018 16:06	Fichier XML	6 Ko
 article_29426583.xml	14/02/2018 16:06	Fichier XML	9 Ko
 article_29426719.xml	14/02/2018 16:06	Fichier XML	12 Ko
 article_29426734.xml	14/02/2018 16:06	Fichier XML	10 Ko
 article_29427049.xml	14/02/2018 16:06	Fichier XML	10 Ko
 article_29427230.xml	14/02/2018 16:06	Fichier XML	12 Ko
 article_29427231.xml	14/02/2018 16:06	Fichier XML	10 Ko
 article_29427232.xml	14/02/2018 16:06	Fichier XML	8 Ko
 article_29427233.xml	14/02/2018 16:06	Fichier XML	8 Ko
 article_29427281.xml	14/02/2018 16:06	Fichier XML	9 Ko
 article_29427399.xml	14/02/2018 16:06	Fichier XML	7 Ko
 article_29427444.xml	14/02/2018 16:06	Fichier XML	9 Ko
 article_29427481.xml	14/02/2018 16:06	Fichier XML	6 Ko
 article_29427485.xml	14/02/2018 16:06	Fichier XML	7 Ko
 article_29427516.xml	14/02/2018 16:06	Fichier XML	7 Ko
 article_29427743.xml	14/02/2018 16:06	Fichier XML	7 Ko
 article_29427917.xml	14/02/2018 16:06	Fichier XML	11 Ko
 article_29427970.xml	14/02/2018 16:06	Fichier XML	10 Ko

Useful-Projects-master > ArticleParser > demo > json

<input type="checkbox"/> Nom	Modifié le	Type	Taille
 article_10022410.json	14/02/2018 19:38	Fichier JSON	10 Ko
 article_29426333.json	14/02/2018 16:38	Fichier JSON	8 Ko
 article_29426385.json	17/02/2018 14:58	Fichier JSON	7 Ko
 article_29426508.json	17/02/2018 14:59	Fichier JSON	5 Ko
 article_29426583.json	17/02/2018 14:59	Fichier JSON	8 Ko
 article_29426719.json	17/02/2018 15:00	Fichier JSON	10 Ko
 article_29426734.json	17/02/2018 15:00	Fichier JSON	8 Ko
 article_29427049.json	17/02/2018 15:02	Fichier JSON	8 Ko
 article_29427230.json	17/02/2018 15:02	Fichier JSON	10 Ko
 article_29427231.json	17/02/2018 15:03	Fichier JSON	8 Ko
 article_29427232.json	17/02/2018 15:03	Fichier JSON	7 Ko
 article_29427233.json	17/02/2018 15:05	Fichier JSON	7 Ko
 article_29427281.json	17/02/2018 15:06	Fichier JSON	7 Ko
 article_29427399.json	17/02/2018 15:06	Fichier JSON	7 Ko
 article_29427444.json	17/02/2018 15:06	Fichier JSON	7 Ko
 article_29427481.json	17/02/2018 15:07	Fichier JSON	5 Ko
 article_29427485.json	17/02/2018 15:07	Fichier JSON	7 Ko
 article_29427516.json	17/02/2018 15:07	Fichier JSON	6 Ko
 article_29427743.json	17/02/2018 15:08	Fichier JSON	6 Ko
 article_29427917.json	17/02/2018 15:08	Fichier JSON	10 Ko
 article_29427970.json	17/02/2018 15:09	Fichier JSON	8 Ko

Enfin, le résultat obtenu après l'exécution de notre programme qui permet de parser des fichiers fournis en entrées et les transformer aux fichiers Json qui seront exploités pour alimenter nos bases de données MongoDB et CouchDB.

Sachant que le programme permet aussi de transformer des fichiers en plusieurs formats selon les formats sélectionnés en option c'est-à-dire selon les arguments de la requête.

I- MongoDB

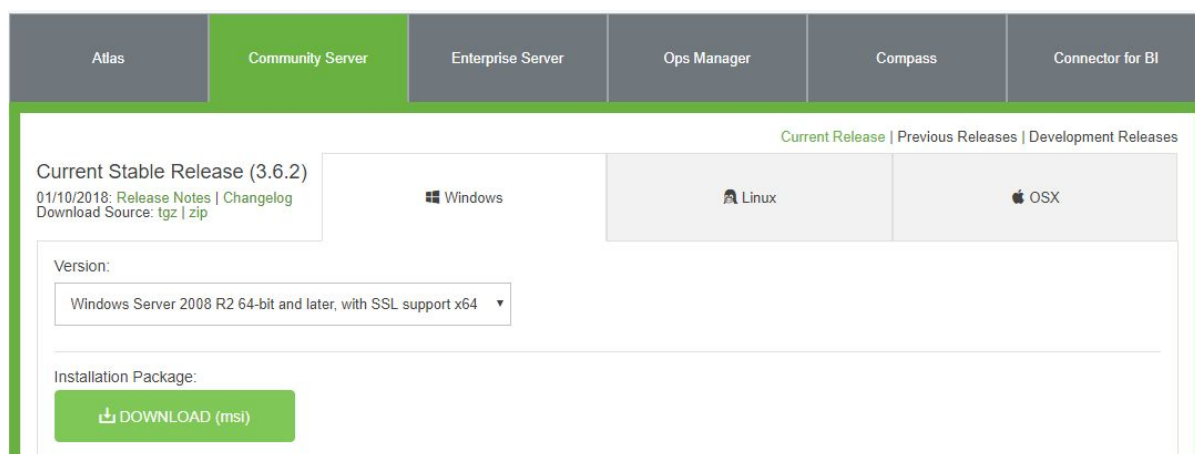
MongoDb est un système de base de données dans la mouvance NoSQL. Il est orienté documents. Son nom vient de *Humongous* qui veut dire énorme ou immense. L'objectif est donc de pouvoir gérer de grandes quantités de données. Comment ? Le moteur de base de données facilite l'extension (on parle de *scaling*) si bien que l'on pourra supporter l'accroissement de la quantité de données par l'ajout de machines.

Dans MongoDB, l'information est modélisée sur un document au format JSON (Javascript Object Notation).

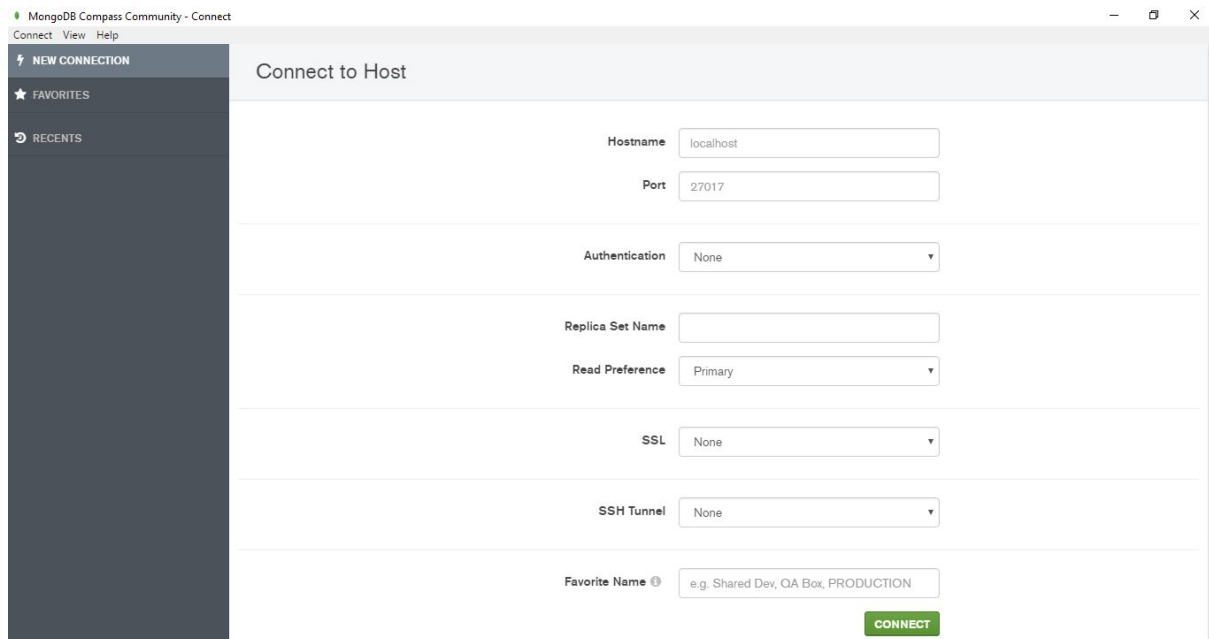
Installation :

Vous pouvez trouver MongoDB en téléchargement ici:

Lien de téléchargement : <https://www.mongodb.com/download-center#community>



L'installation se fait en lançant le fichier .exe téléchargé dans le site précédent un fois l'installation est terminée, mongodb nous offre une interface graphique pour faciliter la manipulation au utilisateurs.



Execution :

Pour pouvoir se connecter à MongoDB. Il faudra d'abord lancer le serveur MongoDB, et pour cela, Ouvrez un premier invite de commande pour lancer le moteur de base de données :

```

mongodb

Microsoft Windows [version 10.0.16299.125]
(c) 2017 Microsoft Corporation. Tous droits réservés.

C:\Users\guett\mongodb
2018-02-14T06:26:37.619-0800 I CONTROL [initandlisten] MongoDB starting : pid=9252 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-SBN95BP
2018-02-14T06:26:37.631-0800 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-02-14T06:26:37.632-0800 I CONTROL [initandlisten] db version v3.6.2
2018-02-14T06:26:37.632-0800 I CONTROL [initandlisten] git version: 489d177dbd0f0420a8ca04d39fd78d0a2c539420
2018-02-14T06:26:37.643-0800 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips 22 Sep 2016
2018-02-14T06:26:37.643-0800 I CONTROL [initandlisten] allocator: tcmalloc
2018-02-14T06:26:37.643-0800 I CONTROL [initandlisten] modules: none
2018-02-14T06:26:37.643-0800 I CONTROL [initandlisten] build environment:
2018-02-14T06:26:37.643-0800 I CONTROL [initandlisten] distmod: 2008plus-ssl
2018-02-14T06:26:37.644-0800 I CONTROL [initandlisten] distarch: x86_64
2018-02-14T06:26:37.644-0800 I CONTROL [initandlisten] target_arch: x86_64
2018-02-14T06:26:37.644-0800 I CONTROL [initandlisten] options: {}
2018-02-14T06:26:37.726-0800 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1496M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(reco
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten]
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten]
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-02-14T06:26:38.570-0800 I CONTROL [initandlisten] **      Remote systems will be unable to connect to this server.
2018-02-14T06:26:38.571-0800 I CONTROL [initandlisten] **      Start the server with --bind_ip <address> to specify which IP
2018-02-14T06:26:38.571-0800 I CONTROL [initandlisten] **      addresses it should serve responses from, or with --bind_ip all to
2018-02-14T06:26:38.571-0800 I CONTROL [initandlisten] **      bind to all interfaces. If this behavior is desired, start the
2018-02-14T06:26:38.572-0800 I CONTROL [initandlisten] **      server with --bind_ip 127.0.0.1 to disable this warning.
2018-02-14T06:26:38.572-0800 I CONTROL [initandlisten]
2018-02-14T06:26:38.572-0800 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This
2018-02-14T06:26:38.573-0800 I CONTROL [initandlisten] can lead to increased memory pressure and poor performance.
2018-02-14T06:26:38.573-0800 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/ut-windows-system-file-cache
2018-02-14T06:26:38.573-0800 I CONTROL [initandlisten]
2018-02-14T15:26:38.676+0100 I STORAGE [initandlisten] createCollection: admin.system.version with provided UUID: 857e7868-8903-40c7-b050-d9e4e04b2dbd
2018-02-14T15:26:38.926+0100 I COMMAND [initandlisten] setting featureCompatibilityVersion to 3.6
2018-02-14T15:26:39.047+0100 I STORAGE [initandlisten] createCollection: local.startup_log with generated UUID: bf97d018-2a11-4620-bb8d-1dbcac2a6645
2018-02-14T15:26:43.440+0100 W FTDC [initandlisten] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 'L'objet
2018-02-14T15:26:43.440+0100 I FTDC [initandlisten] spécifié n'a pas été trouvé sur l'ordinateur.' for counter 'Memory\Available Bytes'
2018-02-14T15:26:43.440+0100 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2018-02-14T15:26:43.468+0100 I NETWORK [initandlisten] waiting for connections on port 27017

```

Une autre méthode pour lancer le client mongodb en utilisant l'invite de commande, ouvrez un second invite de commande pour lancer l'interpréteur de commandes tout en gardant celle du serveur ouvert, puis tapez la commande suivante :

```

mongo

```



```

C:\Users\QLF>mongo
MongoDB shell version v3.6.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-02-17T07:48:30.949-0800 I CONTROL [initandlisten]
2018-02-17T07:48:30.949-0800 I CONTROL [initandlisten] ** WARNING: Access control
is not enabled for the database.
2018-02-17T07:48:30.949-0800 I CONTROL [initandlisten] **      Read and write
access to data and configuration is unrestricted.
2018-02-17T07:48:30.950-0800 I CONTROL [initandlisten]
2018-02-17T07:48:30.951-0800 I CONTROL [initandlisten] ** WARNING: This server is
bound to localhost.
2018-02-17T07:48:30.952-0800 I CONTROL [initandlisten] **      Remote systems
will be unable to connect to this server.
2018-02-17T07:48:30.953-0800 I CONTROL [initandlisten] **      Start the serve
r with --bind_ip <address> to specify which IP
2018-02-17T07:48:30.955-0800 I CONTROL [initandlisten] **      addresses it sh
ould serve responses from, or with --bind_ip_all to
2018-02-17T07:48:30.956-0800 I CONTROL [initandlisten] **      bind to all int
erfaces. If this behavior is desired, start the
2018-02-17T07:48:30.957-0800 I CONTROL [initandlisten] **      server with --b
ind_ip 127.0.0.1 to disable this warning.
2018-02-17T07:48:30.982-0800 I CONTROL [initandlisten]
2018-02-17T07:48:30.983-0800 I CONTROL [initandlisten]
2018-02-17T07:48:31.003-0800 I CONTROL [initandlisten] ** WARNING: The file system
cache of this machine is configured to be greater than 40% of the total memory. Th
is can lead to increased memory pressure and poor performance.
2018-02-17T07:48:31.003-0800 I CONTROL [initandlisten] See http://dochub.mongodb.o
rg/core/wt-windows-system-file-cache
2018-02-17T07:48:31.006-0800 I CONTROL [initandlisten]

```

Les ports utilisés pour la connexion sont

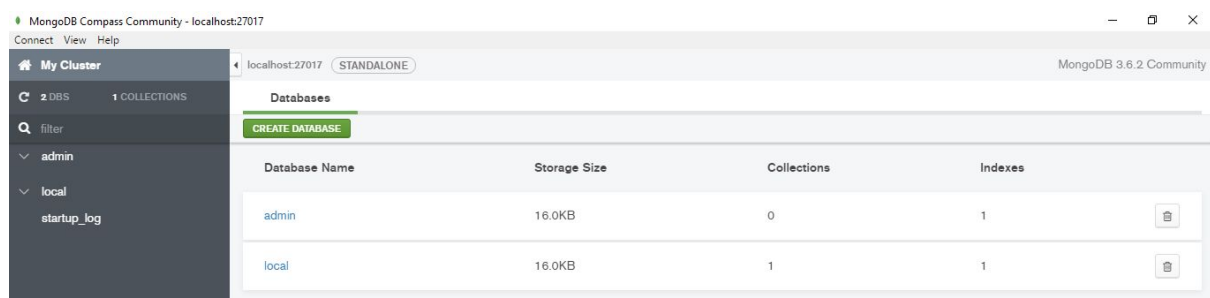
```

2018-02-14T15:29:35.545+0100 I NETWORK [listener] connection accepted from 127.0.0.1:51728 #1 (1 connection now open)
2018-02-14T15:29:35.736+0100 I NETWORK [conn1] received client metadata from 127.0.0.1:51728 conn: { driver: { name: "nodejs", version: "2.2.33" }, os: { type: "Window
s NT", name: "win32", architecture: "x64", version: "10.0.16299" }, platform: "Node.js v7.4.0, LE, mongodb-core: 2.1.17" }
2018-02-14T15:29:36.436+0100 I NETWORK [listener] connection accepted from 127.0.0.1:51730 #2 (2 connections now open)
2018-02-14T15:29:36.447+0100 I NETWORK [listener] connection accepted from 127.0.0.1:51731 #3 (3 connections now open)

```

Page d'accueil

Après avoir lancé le serveur et se connecter à mongodb, on sera redirigé à l'interface suivante :



Création de la base de données et de collection

la figure suivante montre comment créer une base de données mongodb avec une collection qui va contenir les documents (les données). et pour cela il suffit de cliquer sur le bouton

CREATE DATABASE

Create Database

Database Name

Collection Name

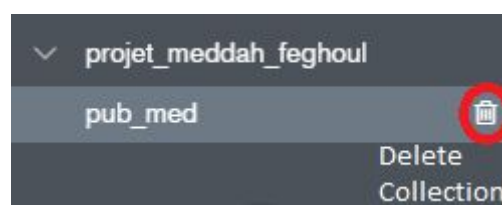
☐ Capped Collection ⓘ

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

CANCEL
CREATE DATABASE

En cliquant sur le bouton create database, voici bien que la base a été bien créé et elle contient une collection, avec la possibilité d'ajouter ou de supprimer de la base qu'on vient de créer aussi la possibilité d'ajouter ou supprimer une collection ou plusieurs

BigData_mongodb	Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	
admin	pub_med	21	5.3 KB	111.3 KB	1	36.0 KB	
config							
local							
projet_meddah_feghoui							
pub_med							



Afficher la liser des bases de dnnées

>show dbs

```
> show dbs
admin          0.000GB
config         0.000GB
local          0.000GB
projet_meddah_feghoui 0.000GB
```

Autre méthode pour la création de la base de données en utilisant l'invite de commande client en tapant la commande suivante, il va la créer tout seul: :

```
>use <nom-database>
```

```
> use projet_meddah_feghouli  
switched to db projet_meddah_feghouli
```

La création de la collection est implicite, elle se fait à l'insertion du premier document, il suffit de faire une insertion direct en donnant un nom pour la collection qui va contenir les documents qu'on veut insérer, comme suit :

```
>bd.<nom-collection>.insert(.....)
```

Afficher la liste des collections de la base de données :

```
>show tables
```

```
> show tables  
pub_med
```

Insertion des Données

Pour insérer des documents dans une base de données mongodb :

```
> db.pub_med.insert({name:"meddah_feghouli", prenom:"amine_ghiles", etude:"GIL"})  
WriteResult({ "nInserted" : 1 })
```

Résultat :



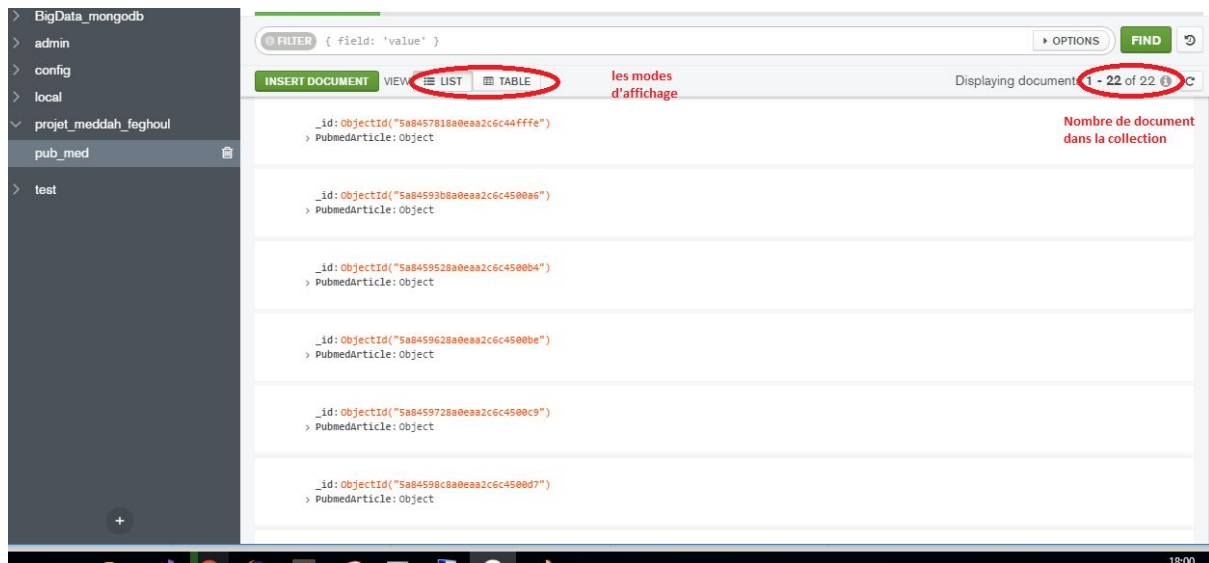
The screenshot shows the MongoDB Compass interface. On the left, a sidebar lists the database 'projet_meddah_feghouli' and the collection 'pub_med'. A red circle highlights the 'pub_med' collection. On the right, the 'Details' tab is active, showing the document details: '_id: ObjectId("5a8857195bb1270386ee1cd7")', 'name: "meddah_feghouli"', 'prenom: "amine_ghiles"', and 'etude: "GIL"'. A red arrow points to the 'WriteResult' output from the terminal above.

Importation de fichiers

Concernant les fichiers json généré précédemment depuis les données XML pour leur insertion dans la base de données, la commande est la suivante qui permet d'insérer un seul fichier json dans une collection :

```
C:\Program Files\MongoDB\Server\3.6\bin>mongoimport --jsonArray -d projet_meddah_feghouli -c pub_med --file C:\Users\QLF\Downloads\Useful-Projects-master\Useful-Projects-master\ArticleParser\demo\json\article_10022410.json  
2018-02-14T16:41:39.032+0100 connected to: localhost  
2018-02-14T16:41:39.245+0100 imported 1 document
```

donc pour insérer tout les document il suffit de refaire la même requête pour les autre documents, le résultat après avoir importé tout les fichiers :



Select, Update et Delete des Données

Details  `_id: ObjectId("5a8857195bb1270386ea1cd7")`
 name: "meddah_feghoui"
 prenom: "amine_ghiles"
 etude: "GIL"

Edit DOC  **Dupliquer DOC**  **Delete Doc** 

ou bien en utilisant l'invite de commande

```
>db.<nom-collection>.find() => sélection de tout le contenu de la collection
>db.<nom-collection>.find({--- : {---:---}, {---:---}}) => sélection d'un ou plusieurs documents donnés, selon les critères donnés en paramètres.
>db.<nom-collection>.update({---:---} , {$set : {---:---}})
>db.<nom-collection>.remove({---:---})
```

Remarque :

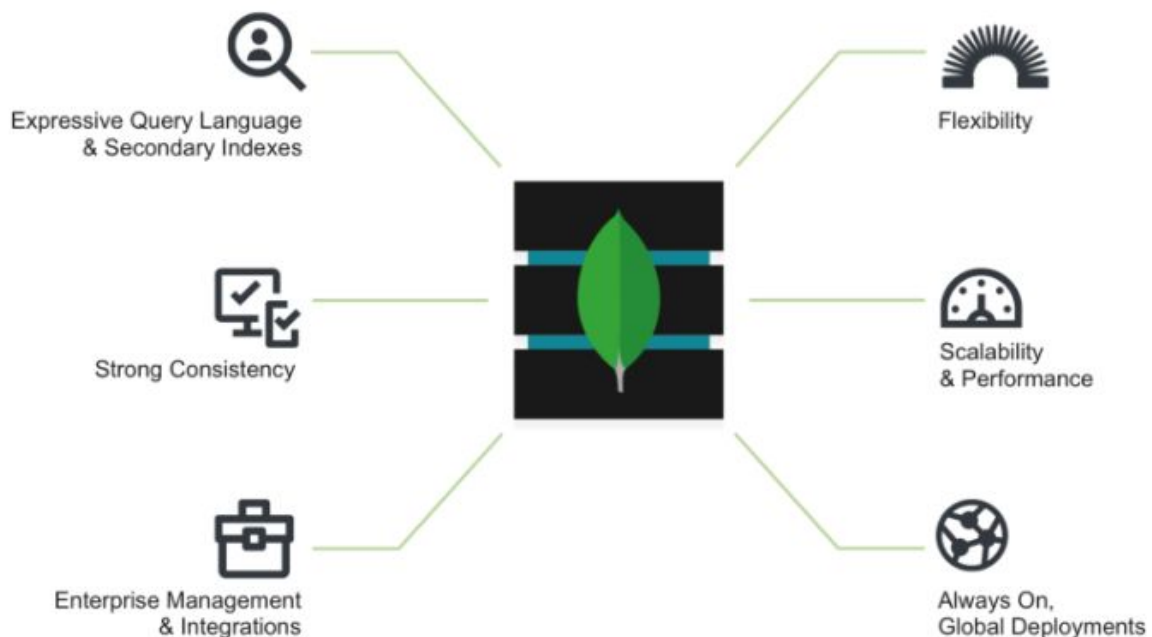
MongoDb limite la taille des documents à 16MB, ce qui est une taille importante pour un document JSON.

Si vos documents sont plus importants, vous pouvez les découper en plusieurs collections (et vous l'aurez probablement fait avant d'approcher cette limite).

Comme dans une base relationnelle, on utilise les `_id` pour faire le liens entre plusieurs collections. La différence est qu'il n'y a pas de jointure. Il vous faudra alors faire 2 requêtes.

Architecture MongoDB :

La philosophie de conception de MongoDB est axée sur la combinaison des capacités critiques des bases de données relationnelles avec les innovations des technologies NoSQL. Notre vision est de tirer parti du travail qu'Oracle et d'autres ont accompli au cours des 40 dernières années pour faire des bases de données relationnelles ce qu'elles sont aujourd'hui. Plutôt que d'abandonner des décennies de maturité de base de données prouvée, MongoDB reprend là où elle s'était arrêtée en combinant des capacités de bases de données relationnelles clés avec le travail que les pionniers d'Internet ont fait pour répondre aux exigences des applications modernes.



Cependant, les applications modernes imposent des exigences non traitées par les bases de données relationnelles, ce qui a conduit au développement de bases de données NoSQL qui offrent: Modèle de données flexible, Évolutivité et performance, Toujours des déploiements globaux.

Tout en offrant ces innovations, les systèmes NoSQL ont sacrifié les capacités critiques auxquelles les gens s'attendent et s'appuient dans les bases de données relationnelles.

MongoDB propose une approche différente. Avec son architecture Nexus, MongoDB est la seule base de données qui exploite les innovations de NoSQL tout en maintenant la base des bases de données relationnelles.

Architecture multimodèle MongoDB

MongoDB permet aux utilisateurs de mélanger et de faire correspondre plusieurs moteurs de stockage au sein d'un même déploiement. Cette flexibilité offre une approche plus simple et plus fiable pour répondre aux divers besoins d'applications pour les données. Traditionnellement, plusieurs technologies de base de données doivent être gérées pour répondre à ces besoins, avec un code d'intégration complexe et personnalisé pour déplacer les données entre les technologies et assurer un accès sécurisé et cohérent. Grâce à l'architecture de stockage flexible de MongoDB, la base de données gère automatiquement le transfert de données entre les technologies de moteur de stockage à l'aide de la réplication native.

Indexation MongoDB :

Fondamentalement, les index dans MongoDB sont similaires aux index dans d'autres systèmes de base de données. MongoDB définit les index au niveau de la collection et prend en charge les index sur n'importe quel champ ou sous-champ des documents d'une collection MongoDB.

MongoDB crée un index unique sur le champ `_id` lors de la création d'une collection. L'index `_id` empêche les clients d'insérer deux documents avec la même valeur pour le champ `_id`. Vous ne pouvez pas supprimer cet index sur le champ `_id`

Remarque

Dans les clusters partitionnés, si vous n'utilisez pas le champ `_id` comme clé de partition, votre application doit garantir l'unicité des valeurs dans le champ `_id` pour éviter les erreurs. Cela est le plus souvent effectué en utilisant un ObjectId standard généré automatiquement.

Création d'un index

Pour créer un index dans l'environnement Mongo, utilisez

```
db.collection.createIndex ().
db.collection.createIndex( <key and index type specification>, <options> )
```

Types d'index

MongoDB fournit un certain nombre de types d'index différents pour prendre en charge des types spécifiques de données et de requêtes.

Single Field (Champ unique) : En plus de l'index `_id` défini par MongoDB, MongoDB prend en charge la création d'index ascendants / descendants définis par l'utilisateur sur un seul champ d'un document.

Compound Index (Index composé) : MongoDB prend également en charge les index définis par l'utilisateur sur plusieurs champs, c'est-à-dire les index composites.

MultiKey Index (Index multi-clés) : MongoDB utilise des index multi-clés pour indexer le contenu stocké dans les tableaux. Si vous indexez un champ qui contient une valeur de

tableau, MongoDB crée des entrées d'index séparées pour chaque élément du tableau. Ces index multi-clés permettent aux requêtes de sélectionner des documents contenant des tableaux en faisant correspondre des éléments ou des éléments des tableaux. MongoDB détermine automatiquement s'il faut créer un index multi-clés si le champ indexé contient une valeur de tableau; vous n'avez pas besoin de spécifier explicitement le type multi-clé.

Geospatial Index (Index géospatial) : Pour prendre en charge des requêtes efficaces de données de coordonnées géospatiales, MongoDB fournit deux index spéciaux: les index 2d qui utilisent la géométrie plane lors du renvoi des résultats et les index 2dsphere qui utilisent la géométrie sphérique pour renvoyer les résultats.

Text Indexes (Index de texte) : MongoDB fournit un type d'index de texte qui prend en charge la recherche de contenu de chaîne dans une collection. Ces index de texte ne stockent pas de mots d'arrêt spécifiques à la langue (par exemple "the", "a", "or") et empêchent les mots d'une collection de stocker uniquement les mots racines.

Hashed Indexes (Index hachés) : Pour prendre en charge le sharding basé sur le hachage, MongoDB fournit un type d'index haché qui indexe le hachage de la valeur d'un champ. Ces index ont une distribution plus aléatoire des valeurs le long de leur plage, mais ne prennent en charge que les correspondances d'égalité et ne peuvent pas prendre en charge les requêtes par plage.

Remarque

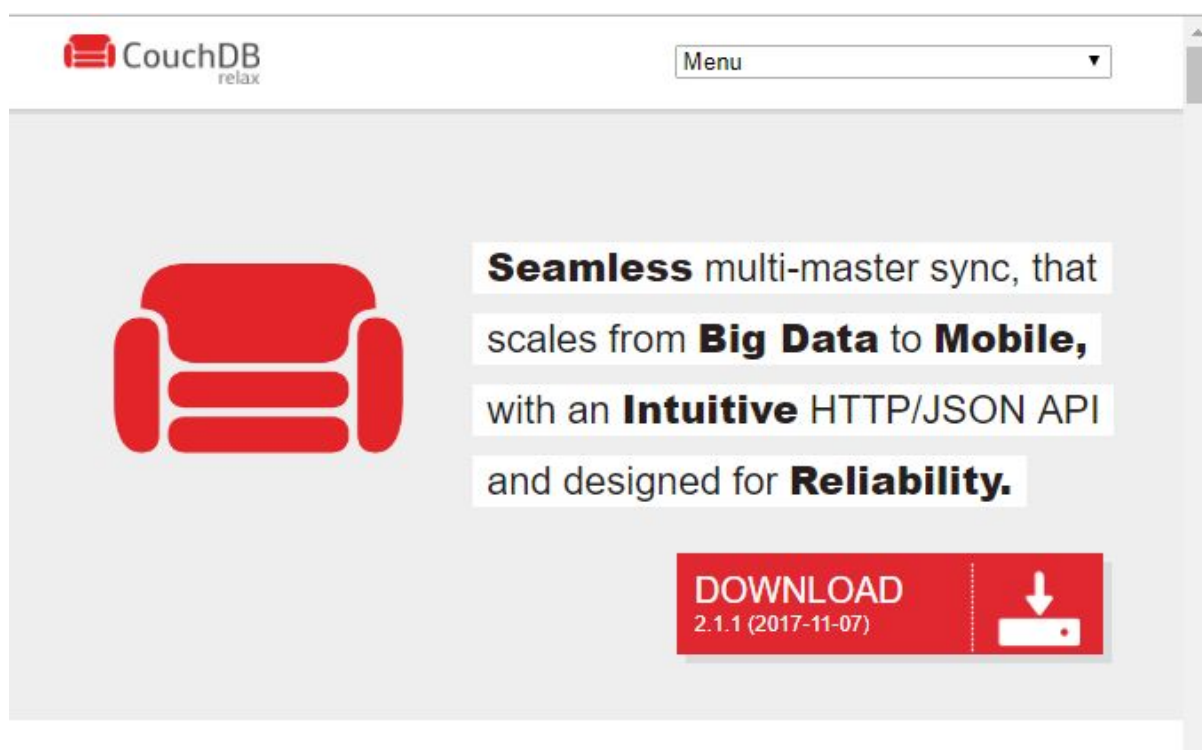
Les index sont la méthode préférée dans MongoDB, et ne pas avoir d'index peut ralentir les temps de lecture.

II- CouchDB

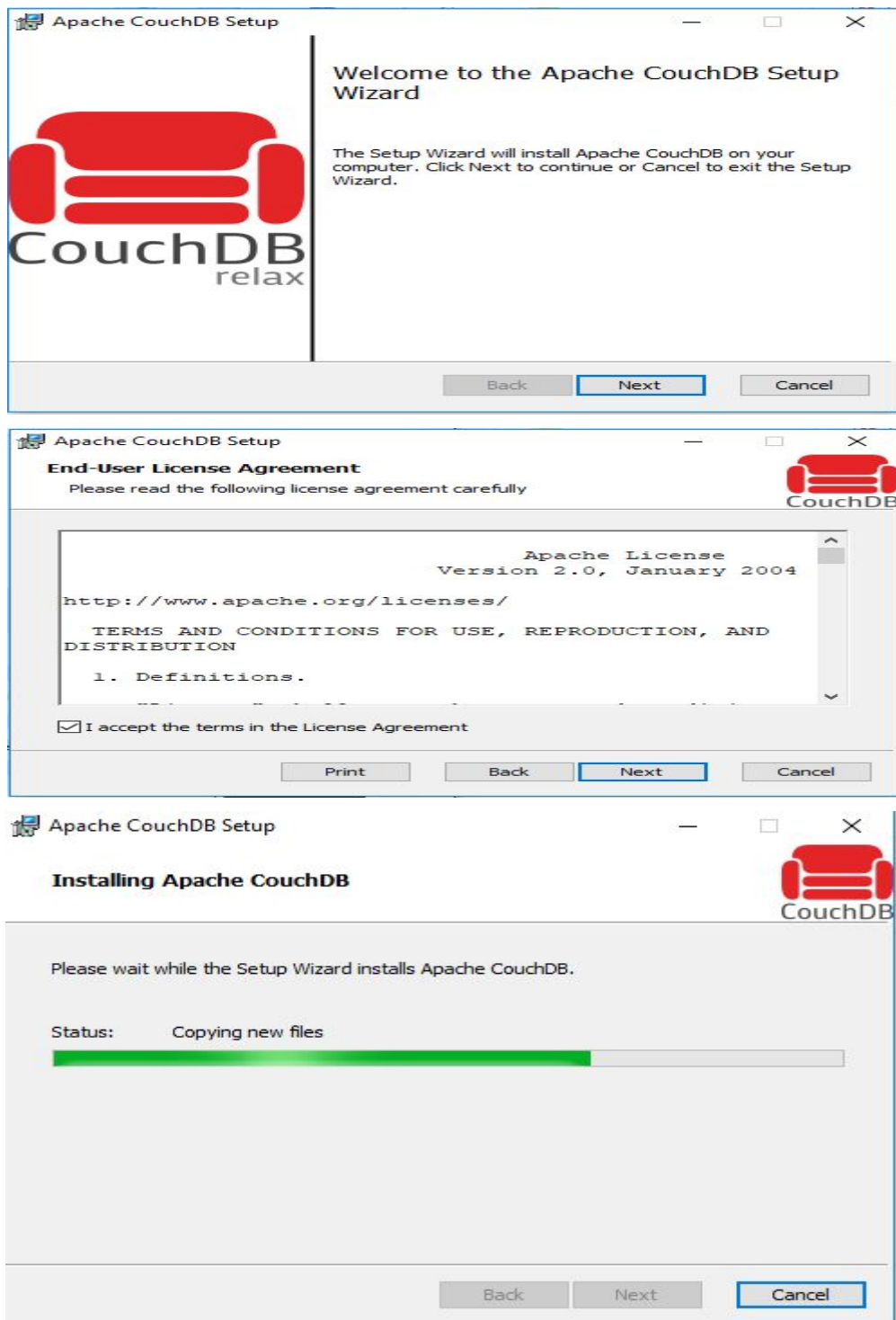
CouchDB est une base de données qui embrasse complètement le web. Stockez vos données avec des documents JSON. Accédez à vos documents avec votre navigateur Web, via HTTP . Interrogez , combinez et transformez vos documents avec JavaScript . CouchDB fonctionne bien avec les applications Web et mobiles modernes. Vous pouvez distribuer vos données, en utilisant efficacement la réplication incrémentale de CouchDB . CouchDB prend en charge les configurations maître-maître avec détection automatique des conflits . CouchDB est livré avec une suite de fonctionnalités, telles que la transformation de documents à la volée et les notifications de changement en temps réel , qui facilitent le développement Web. Il est même livré avec une console d'administration Web facile à utiliser, directement à partir de CouchDB! Nous nous soucions beaucoup de la mise à l'échelle distribuée . CouchDB est hautement disponible et tolérant à la partition, mais il est également cohérent . Et nous nous soucions beaucoup de vos données. CouchDB dispose d'un moteur de stockage tolérant aux pannes qui met la sécurité de vos données en premier.

Installation :

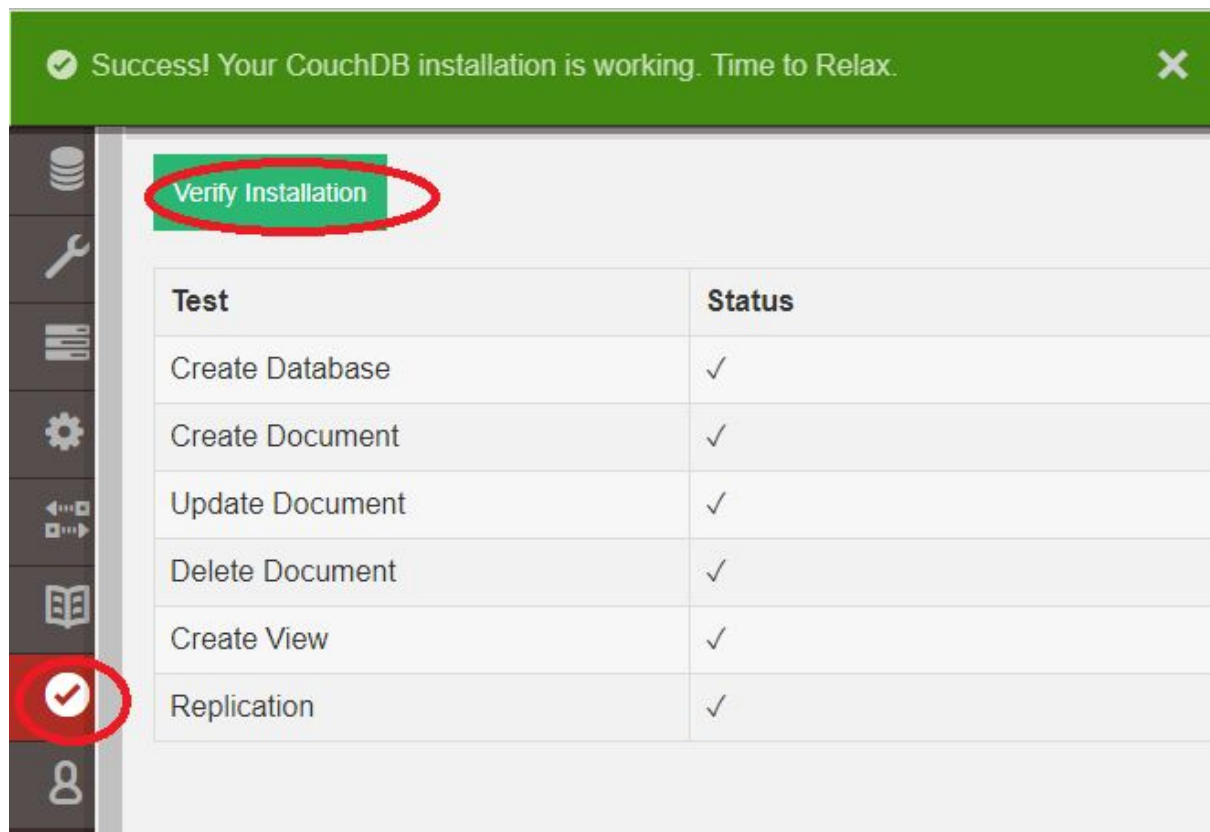
Lien de téléchargement : <http://couchdb.apache.org/>



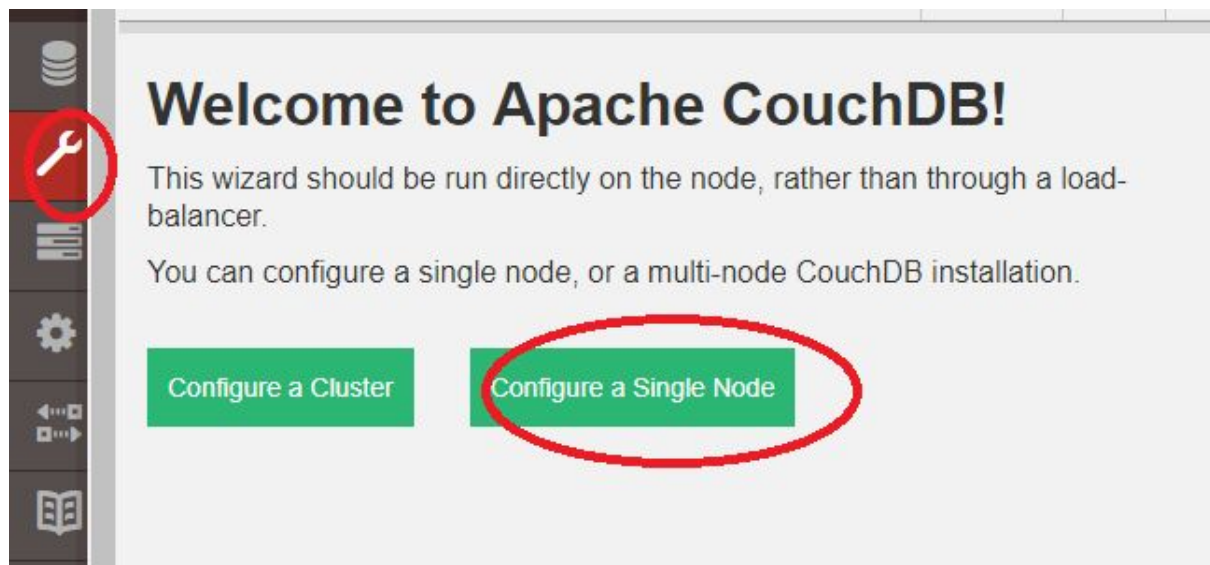
Quelque figures qui détaillent les phases de l'installation



un fois l'installation est terminée, CouchDb nous offre une interface graphique pour faciliter la manipulation au utilisateurs, et pour accéder à cette interface il suffit de taper sur le navigateur l'url suivant : http://127.0.0.1:5984/_utils/index.html#/all_dbs
Ensuite, vérifier que tous les outils Couch ont été bien installés



Création d'un compte Admin:



Project Fauxton

127.0.0.1:5984/_utils/index.html#setup/singlenode

Applications Portail Wifirst etudes films inconnu Coinbase

Setup Apache CouchDB

Create Admin credentials.

admin

.....

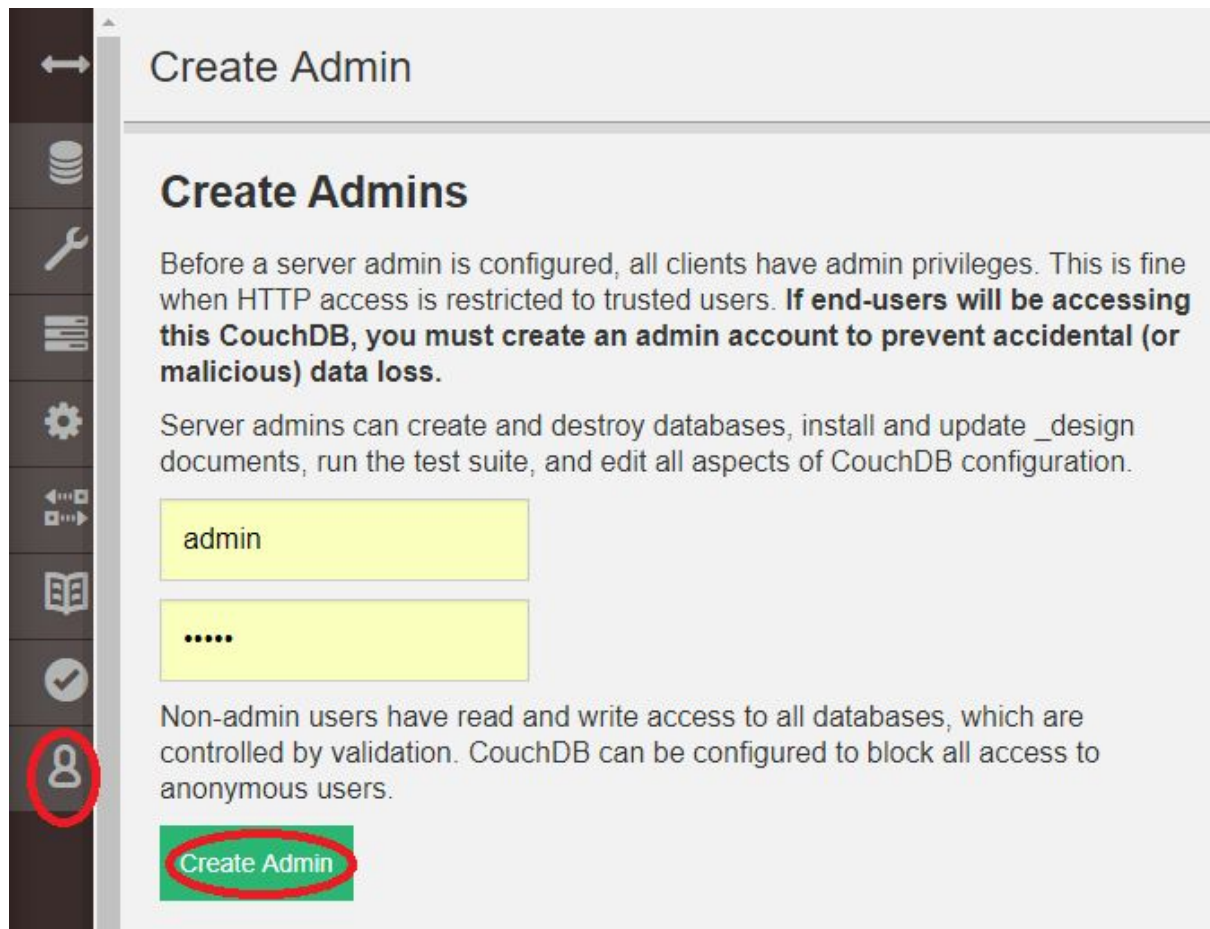
Bind address the node will listen on

0.0.0.0

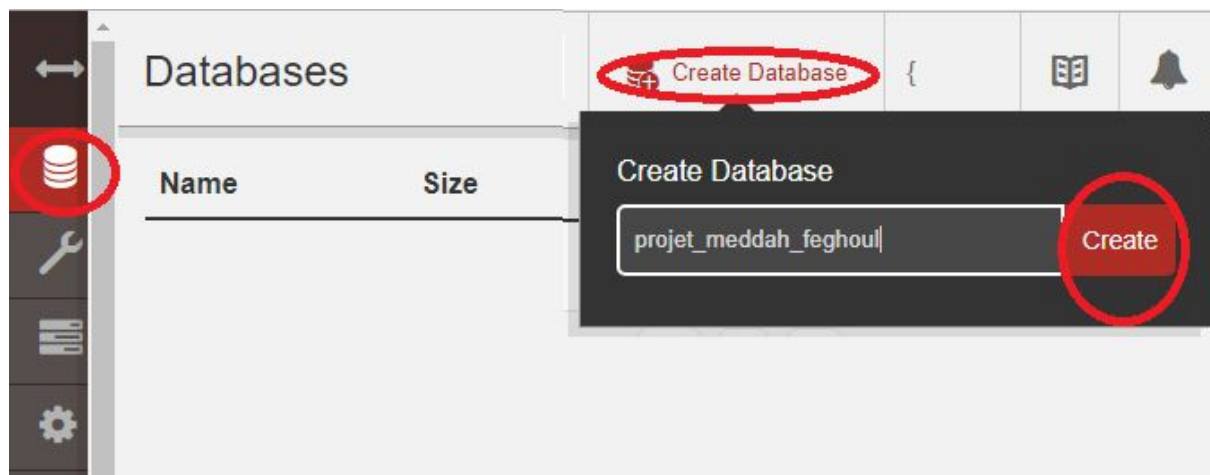
Port that the node will use

5984

✓ Configure Node



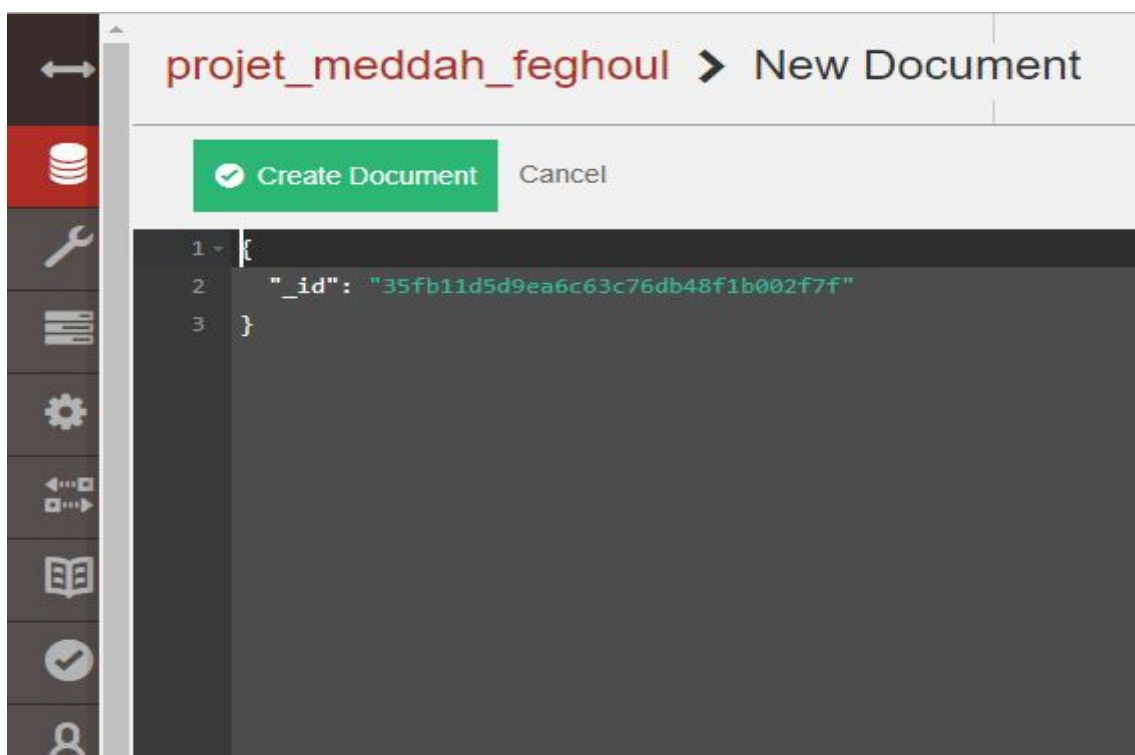
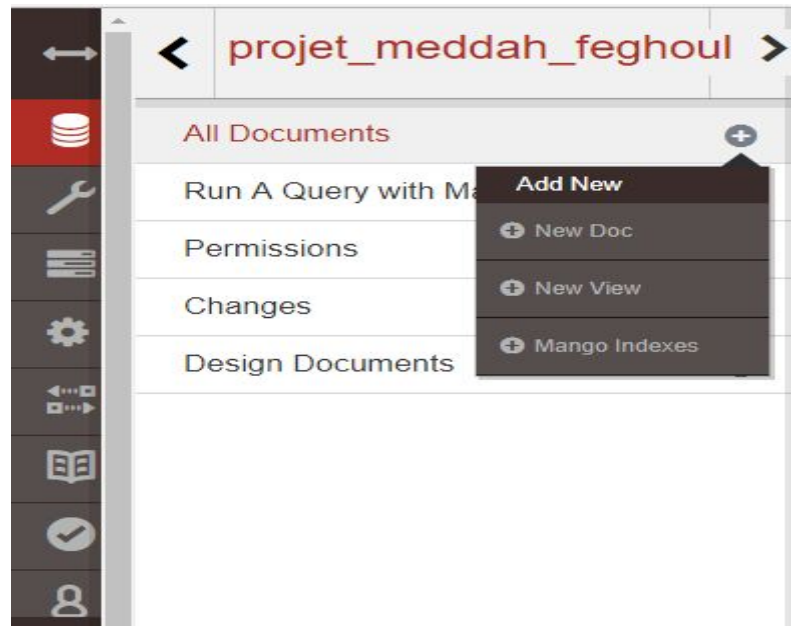
Création d'une Base de données:



Résultat

Name	Size	# of Docs	Actions
projet_meddah_feghoui	66.0 KB	21	   

Insertion des données :



projet_meddah_feghouli > New Document

☒ Create Document Cancel

```

158   "PMID": {
159     "Version": 1,
160     "content": 29427970
161   }
162 },
163 "PubMedData": {
164   "History": {"PubMedPubDate": [
165     {
166       "Month": "08",
167       "Year": 2017,
168       "Day": 14,
169       "PubStatus": "received"
170     },
171     {
172       "Month": 12,
173       "Year": 2017,
174       "Day": "06",
175       "PubStatus": "revised"
176     },
177     {
178       "Month": "01",
179       "Year": 2018,

```

Mode d'affichage

☐ Table ☐ Metadata ☒ JSON

id "35fb11d5d9ea6c63c76db48f1b002f7f"

```

{
  "id": "35fb11d5d9ea6c63c76db48f1b002f7f",
  "key": "35fb11d5d9ea6c63c76db48f1b002f7f",
  "value": {
    "rev": "1-b4c600d2c8c198d7761d8e8fe09093fa"
  },
  "doc": {
    "_id": "35fb11d5d9ea6c63c76db48f1b002f7f",
    "_rev": "1-b4c600d2c8c198d7761d8e8fe09093fa",
    "PubMedArticle": {
      "MedlineCitation": {
        "Status": "MEDLINE",
        "MedlineJournalInfo": {
          "NlmUniqueID": "0375362",
          "Country": "United States",
          "ISSNLinking": "0021-972X",
          "MedlineTA": "J Clin Endocrinol Metab"
        },
        "Owner": "NLM",
        "CitationSubset": [
          "AIM",
          "IM"

```

l'attribut **_rev** permet d'indiquer le nombre de modifications effectuée sur le document tout en incrémentant le premier indice i+1 (i-xx..) et en attribuant une nouvel suite aléatoire (xx..)

Après avoir insert tous les documents le résultat de requête qui permet de retourner tous les document de la collection est le suivant

Query history

Mango Query ?

```

1 {
2   "selector": {
3     "_id": {
4       "$gt": null
5     }
6   }
7 }

```

Run Query Explain manage indexes

Executed in 28 ms

Table JSON

PubMedArticle Resultat _id

<input type="checkbox"/>	{ "MedlineCitation": { "Status": "MEDLINE", "Medli...	35fb11d5d9ea6c63c76db48f1b01cdc3
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "In-Process", "Medli...	35fb11d5d9ea6c63c76db48f1b01db9a
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "In-Data-Review", "...	35fb11d5d9ea6c63c76db48f1b01f007
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "In-Process", "Medli...	35fb11d5d9ea6c63c76db48f1b020918
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "Publisher", "Medlin...	35fb11d5d9ea6c63c76db48f1b02154a
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "Publisher", "Medlin...	35fb11d5d9ea6c63c76db48f1b022802
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "Publisher", "Medlin...	35fb11d5d9ea6c63c76db48f1b0244a5
<input type="checkbox"/>	{ "MedlineCitation": { "Status": "Publisher", "Medlin...	35fb11d5d9ea6c63c76db48f1b025496

projet_meddah_feg... > Mango Q...

Query history

Mango Query ?

```

1 {
2   "selector": {
3     "_id": {
4       "$gt": null
5     }
6   }
7 }

```

Run Query Explain manage indexes

```

{
  "dbname": "projet_meddah_feghou1",
  "index": {
    "ddoc": null,
    "name": "_all_docs",
    "type": "special",
    "def": {
      "fields": [
        {
          "_id": "asc"
        }
      ]
    }
  },
  "selector": {
    "_id": {
      "$gt": null
    }
  },
  "opts": {
    "use_index": [],
    "bookmark": "nil",
    "limit": 25,
    "skip": 0,
    "sort": {},
    "fields": "all_fields",
    "r": [
      49
    ]
  }
}

```

Sélectionner un document avec un `_id` donnée :

The screenshot shows the Mango Query interface. On the left, the 'Mango Query' section contains a query: `{ "selector": { "_id": "35fb11d5d9ea6c63c76db48f1b01cdc3" } }`. Below the query are buttons for 'Run Query' and 'Explain', and a status message 'Executed in 11 ms'. On the right, the 'JSON' tab is selected, displaying the document: `{ "_id": "35fb11d5d9ea6c63c76db48f1b01cdc3", "_rev": "1-b4c600d2c8c198d7761d8e8fe09093fa", "PubmedArticle": { "MedlineCitation": { "Status": "MEDLINE", "MedlineJournalInfo": { "NlmUniqueID": "0375362", "Country": "United States", "ISSNLinking": "0021-972X", "MedlineTA": "J Clin Endocrinol Metab" }, "Owner": "NLM", "CitationSubset": ["AIM", "IM"] } } }`.

Utilisation CURL :

il permet d'interroger les base de données CouchDB en utilisant l'invite de commande

Lien de telechargement et tuto d'installation :

<https://help.zendesk.com/hc/en-us/articles/229136847-Installing-and-using-cURL#install>

Examinons l'Application Programming Interface (API) à l'aide de l'utilitaire en ligne de commande curl. Notez qu'il s'agit d'une manière parmi d'autres de s'adresser à CouchDB et que nous vous en indiquerons de nouvelles dans la suite de l'ouvrage. Ce qui est intéressant avec curl, c'est qu'il vous permet de forger votre requête HTTP et de voir ce qui se trouve « sous le capot » de votre base de données. Assurez-vous que CouchDB est démarré et exécutez :

```
>curl http://127.0.0.1:5984/
```

Cela envoie une requête de type GET à l'instance de CouchDB que vous venez d'installer.

```
C:\Users\QLF>curl
curl: try 'curl --help' or 'curl --manual' for more information

C:\Users\QLF>curl http://127.0.0.1:5984
{"couchdb": "Welcome", "version": "2.1.1", "features": ["scheduler"], "vendor": {"name": "The Apache Software Foundation"}}

C:\Users\QLF>curl http://127.0.0.1:5984/projet_meddah_feghou1
{"db_name": "projet_meddah_feghou1", "update_seq": "21-g1AAAAFeJzLYWBg4MhgTmEQTM4vTc5ISXLIyU90zMnIly7JAUoxJTikyf__z8rkRmPoiQFIJlKd1bHhE-dA0hdPGHzEkDq6gm18cCJBkagBRQ6fysRBaCahda107Hbz9E7QG12vtZiYwE1T6AqAX5KwsA4TfvQw", "sizes": {"file": 140758, "external": 156717, "active": 67593}, "purge_seq": 0, "other": {"data_size": 156717, "doc_del_count": 0, "doc_count": 21, "disk_size": 140758, "disk_format_version": 6, "data_size": 67593, "compact_running": false, "cluster": {"q": 8, "n": 1, "w": 1, "r": 1}, "instance_start_time": "0"}
```

Remarque

La commande curl envoie une requête GET par défaut. Vous pouvez produire des requêtes POST, PUT, DELETE avec curl `-X <METHODE>`. Afin de nous y retrouver dans l'historique de la console, nous utilisons l'option `-X` même pour les requêtes de type GET. Par la suite, si nous voulons envoyer la requête en POST, PUT, DELETE, il suffit de changer la méthode.

HTTP effectue davantage d'opérations « sous le capot » que celles que vous voyez ici. Si vous cherchez à voir tout ce qui passe sur le câble, ajouter l'option -v (c.-à-d. curl -vX GET) et vous verrez la tentative de connexion au serveur, les en-têtes de la requête et ceux de la réponse. Très utile pour déboguer !

Créons une base de données :

```
>curl -X PUT http://127.0.0.1:5984/projet_meddah_feghoui
```

Réponse

```
{"ok": true}
```

Nous pouvons lister les bases :

```
>curl -X GET http://127.0.0.1:5984/_all_dbs
```

SELECT avec _id donné :

requête Get permet la sélection des données

```
C:\Users\QLF>curl -X GET http://127.0.0.1:5984/projet_meddah_feghoui/35fb11d5d9ea6c63c76db48f1b01cdc3
{"_id": "35fb11d5d9ea6c63c76db48f1b01cdc3", "_rev": "1-b4c600d2c8c198d7761d8e8fe09093fa", "PubmedArticle": {"MedlineCitation": {"Status": "MEDLINE", "MedlineJournalInfo": {"NlmUniqueID": "0375362", "Country": "United States", "ISSNLinking": "0021-972X", "MedlineTA": "J Clin Endocrinol Metab"}, "Owner": "NLM", "CitationSubset": ["AIM", "IM"], "ChemicalList": {"Chemical": [{"NameOfSubstance": {"UI": "D019314", "content": "Dehydroepiandrosterone Sulfate"}, "RegistryNumber": "57B0907FJR"}, {"NameOfSubstance": {"UI": "D003907", "content": "Dexamethasone"}, "RegistryNumber": "7S5I7G3JQL"}, {"NameOfSubstance": {"UI": "D000324", "content": "Adrenocorticotrophic Hormone"}, "RegistryNumber": "9002-60-2"}, {"NameOfSubstance": {"UI": "D006854", "content": "Hydrocortisone"}, "RegistryNumber": "WI4X0X7BPJ"}]}, "DateCompleted": {"Month": "02", "Year": 1999, "Day": 25}, "Article": {"Pagination": {"MedlinePgn": "520-6"}, "PublicationTypeList": {"PublicationType": [{"UI": "D016428", "content": "Journal Article"}, {"UI": "D013485", "content": "Research Support, Non-U.S. Gov't"}]}, "Language": "eng", "Abstract": {"AbstractText": "The natural course of adrenal incidentalomas and the risk that such lesions evolve toward hormonal hypersecretion or malignancy are still under evaluation. Of 246 consecutive patients with adrenal incidentaloma studied at our institution in the last 15 yr, 91 underwent surgery. Of the remaining patients, a group of 75 (52 females and 23 males; median age, 56 yr; range, 19-77 yr) with incidentally discovered asymptomatic adrenal masses (60 unilateral and 15 bilateral; median diameter, 2.5 cm; range, 1.0-5.6) was enrolled in an endocrine and morphological follow-up of at least 2 yr after diagnosis (median, 4 yr; range, 2-10). During follow-up, no patients developed malignancy; 9 showed mass enlargement, with appearance of a new mass in the contralateral gland in 2; 3 developed adrenal hyperfunction (overt Cushing's syndrome in 2, nonclinical hypercortisolism in 1); and 3 showed adrenal mass enlargement associated with adrenal hyperfunction (nonclinical hypercortisolism in 2, pheochromocytoma in 1). The estimated cumulative risks to develop mass enlargement and hyperfunction were 8% and 4%, respectively, after 1 yr, 18% and 9.5% after 5 yr, and 22.8% and 9.5% after 10 yr. Nine risk factors for adrenal mass enlargement or hyperfunction were arbitrarily selected and evaluated: sex, age, presence of obesity, hypertension, diabetes, abnormal endocrine tests, mass size, mass location, and scintigraphic uptake pattern. Three of them attained statistical significance: mass size of 3 cm or more at diagnosis and exclusive radiocholesterol uptake by the mass at scintigraphy had relevance for the occurrence of adrenal hyperfunction, whereas the presence of endocrine test abnormalities at diagnosis had predictive value for mass enlargement. It is concluded that subtle hormonal abnormalities are risk factors for mass size increase, which is not a sign of malignant transformation. Both mass size of 3 cm or more at diagnosis and exclu
```

INSERT

```
C:\Users\QLF>curl -X POST http://127.0.0.1:5984/projet_meddah_feghou1 -d {"name\":"meddah_feghou1","\prenom\":"amine_ghiles"} -H "Content-Type:application/json" {"ok":true,"id":"35fb11d5d9ea6c63c76db48f1b03d011","rev":"1-2598a751b7b1cdc95d11b81b1a3486fb"}
```

Résultat

<input type="checkbox"/>		35fb11d5d9ea6c63c76db4...	{ "MedlineCitation": { "Statu...
<input type="checkbox"/>		35fb11d5d9ea6c63c76db4...	meddah_feghou1 amine_ghiles

UPDATE

```
C:\Users\QLF>curl -X PUT http://127.0.0.1:5984/projet_meddah_feghou1/35fb11d5d9ea6c63c76db48f1b03d011 -d {"_rev\":"1-2598a751b7b1cdc95d11b81b1a3486fb","\name\":"meddah_feghou1_put","\prenom\":"amine_ghiles_put","\etude\":"GIL_put"} -H "Content-Type:application/json" {"ok":true,"id":"35fb11d5d9ea6c63c76db48f1b03d011","rev":"2-df63048b6d308323142f2b2413365999"}
```

Résultat

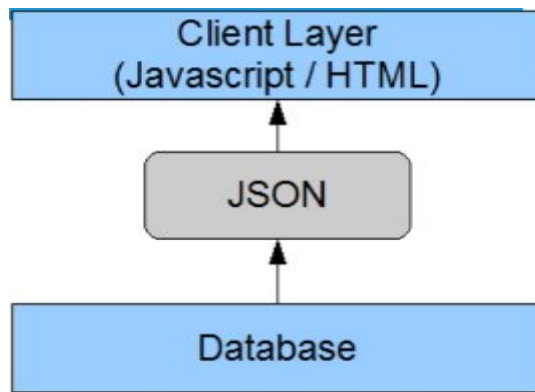
	_id	name	prenom	PubmedArticle	etude
<input type="checkbox"/>		35fb11d5d9ea6...	{ "MedlineCitati...		
<input type="checkbox"/>		35fb11d5d9ea6...	meddah_fegho...	amine_ghiles_put	GIL_put
<input type="checkbox"/>		gil	meddah_fegho...	amine_ghiles_put	

DELETE

```
C:\Users\QLF>curl -X DELETE http://127.0.0.1:5984/projet_meddah_feghou1/gil?rev=2-555ff2ce73de52ee29a2bcc3df57365c {"ok":true,"id":"gil","rev":"3-a99ed491ec71dcc95ad5360722f21b8"}
```

Architecture CouchDB :

C'est une architecture à deux tiers, utilise HTTP comme interface de programmation principale et JSON pour le stockage de données. Comme nous avons du JavaScript côté client, les performances vont augmenter grâce à la compatibilité de JSON avec JavaScript CouchDB est compatible avec des plateformes telles que Windows, Linux, Mac-iOS, mobiles Android. elle est plus approprié pour les applications client telles que les applications Web.



CouchDB a été conçu avec la réplication bidirectionnelle (ou la synchronisation) et l'opération hors ligne en tête. Cela signifie que plusieurs réplicas peuvent avoir leurs propres copies des mêmes données, les modifier, puis synchroniser ces modifications ultérieurement.

CouchDB garantit une cohérence éventuelle pour pouvoir fournir à la fois la disponibilité et la tolérance de partition.

Indexation CouchDB :

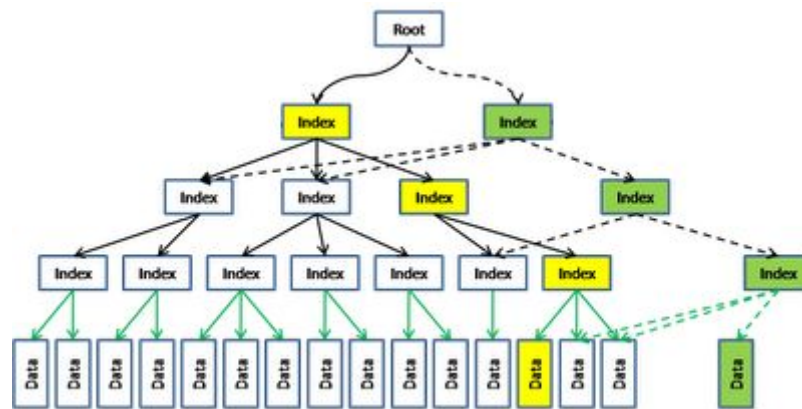
L'interface Query est (actuellement) une API simplificatrice permettant de créer et d'accéder aux vues CouchDB non-existantes. Les index que vous définissez via le noeud final **_index** sont en fait traduits en vues, et ces vues peuvent être consultées et utilisées de la même manière qu'une vue CouchDB normale, ainsi que via le point de terminaison **_find** (remarque: l'inverse n'est pas vrai - Query doesn't utiliser pas actuellement les vues javascript existantes). Les vues restent dans la couche erlang, ce qui nous donne l'opportunité d'améliorer les performances, etc.

Également, on peut filtrer les données de résultat pour ne renvoyer que les champs de document qui vous intéressent, plutôt que de coder en dur les champs renvoyés dans la vue ou d'exécuter le résultat de la vue via une fonction de liste.

Vue

Les vues dans CouchDB sont similaires aux index dans SQL. Les vues dans CouchDB peuvent être utilisées pour filtrer des documents, récupérer des données dans un ordre spécifique et créer des index efficaces pour que vous puissiez trouver des documents en utilisant des valeurs. Une fois que vous avez des index, ils peuvent représenter les relations entre les documents. Ces résultats de vue sont stockés dans une structure d'index B-tree.

CouchDB utilise MapReduce, un processus en deux étapes qui examine tous les documents et crée un résultat de carte composé d'une liste ordonnée de paires clé / valeur. Le mappage se produit une fois après la création d'un document. Après cela, il n'est pas modifié à moins que le document ne soit mis à jour.



Index B-tree CouchDB

Type Vues :

Les vues sont l'outil principal utilisé pour l'interrogation et la création de rapports sur les documents CouchDB. Il existe deux types de vues: les vues permanentes et temporaires.

1- Les vues permanentes sont stockées dans des documents spéciaux appelés documents de conception, et peuvent être accédées via une requête *HTTP GET* à l'URI / {nom_db} / {docid} / {nom_vue}, où {docid} a le préfixe *_design* / de sorte que CouchDB reconnait le document comme document de conception, et {viewname} a le préfixe *_view* / de sorte que CouchDB le reconnaisse comme une vue.

2- Les vues temporaires ne sont pas stockées dans la base de données, mais plutôt exécutées à la demande. Pour exécuter une vue temporaire, vous envoyez une requête *HTTP POST* à l'URI / {nom_bd} / *_temp_view*, où le corps de la requête contient le code de la fonction d'affichage et l'en-tête *Content-Type* est défini sur *application / json*.

Remarque :

Les vues temporaires ne sont bonnes que pendant le développement. Le code final ne doit pas compter sur eux, car ils sont très coûteux à calculer chaque fois qu'ils sont appelés et ils sont de plus en plus lents, car plus vous avez de données dans une base de données. Si vous pensez que vous ne pouvez pas résoudre quelque chose dans une vue permanente que vous pouvez résoudre dans une vue ad-hoc, vous pouvez vouloir reconsidérer.

Pour les deux types de vues, la vue est définie par une fonction JavaScript qui associe les clés de vue aux valeurs (bien qu'il soit possible d'utiliser d'autres langages que JavaScript en branchant des serveurs de vue tiers).

Par défaut, les vues ne sont pas créées et mises à jour lorsqu'un document est enregistré, mais plutôt lorsque vous y accédez. Par conséquent, le premier accès peut prendre un certain temps en fonction de la taille de vos données pendant que CouchDB crée la vue. Si cela est préférable, les vues peuvent également être mises à jour lorsqu'un document est enregistré à l'aide d'un script externe qui appelle les vues lorsque des mises à jour ont été effectuées.

Toutes les vues d'un document de conception unique sont mises à jour lorsque l'une des vues de ce document de conception est interrogée.

Qui utilise ces bases de données?

Voici quelques exemples d'entreprises qui utilisent ces bases de données:

CouchDB: Talend SA, Akamai Technologies, Hothead Games, Inc., GenCorp Technologies, Vivint Solar, Inc.

MongoDB : Adobe, BBVA, CERN, Département des Anciens Combattants, Electronic Arts, Forbes, Under Armour

Systèmes d'exploitation du serveur ?

CouchDB : Android, BSD, Linux, OS X, Solaris, Windows.

MongoDB : Linux, OS X, Solaris, Windows.

Conclusion

CouchDB: CouchDB utilise un magasin de documents avec des données présentées au format JSON. Il offre une API HTTP RESTful pour lire, ajouter, modifier et supprimer des documents de base de données. Chaque document comprend des champs et des pièces jointes. Les champs peuvent contenir des nombres, du texte, des booléens, des listes, etc.

Le modèle de mise à jour pour CouchDB est optimiste et sans verrou. Cette structure de base de données, inspirée de Lotus Notes, peut être étendue des clusters globaux aux périphériques mobiles.

MongoDB : MongoDB stocke les données sans schéma en utilisant des documents au format BSON. Ces collections de documents n'ont pas besoin d'une structure prédéfinie et les colonnes peuvent varier pour différents documents de la collection.

MongoDB est sans schéma, vous permettant de créer des documents sans avoir à créer d'abord la structure de ce document. Dans le même temps, il possède encore de nombreuses fonctionnalités d'une base de données relationnelle, y compris une cohérence forte et un langage de requête expressif.

CouchDB est plus approprié pour des données plus stables, où tous les documents pour une collection donnée ont les mêmes champs et les requêtes ne changeront pas beaucoup. Dans ce projet, l'un des principaux objectifs est la flexibilité des données et des requêtes. Ainsi, en fonction des requêtes précédemment définies, vous pouvez lier le développement et l'itération de l'utilisateur. Un autre point important est la croissance de notre taille de données: MongoDB est conçu pour fournir une solution robuste pour stocker les données et leur permettre de croître au fil du temps, et pour cela, ce que nous devons faire est d'insérer de nouveaux fragments dans le clusters, sans avoir à changer le code du projet.

Donc, pour choisir entre ces deux:

MongoDB: Si j'ai besoin de requêtes dynamiques. Si je préfère définir des index, ne pas mapper / réduire les fonctions. Si j'ai besoin de bonnes performances sur une grosse base de données. Si je voulais CouchDB, mais les données changent trop, remplissant les disques.

CouchDB: Pour accumuler, occasionnellement modifier des données, sur lesquelles des requêtes prédéfinies doivent être exécutées. Endroits où le contrôle des versions est important

Donc en conclusion, MongoDB est plus rapide, CouchDB est plus sûr.

