COUCHDB

I - CONCEPTS DE BASE

CouchDB est une base de données **non relationnelle** (pas de tables et de jointures comme en SQL) développée par Apache..
Elle est orientée « **document** ». Les documents écrits au format des objets **JSON**.

CouchDB a été développée pour être déployée sur internet (cloud) et permet :

- Toutes les manipulations sont réalisables via une **API HTTP** REST (y compris sur une même machine).
- La **réplication** des **bases** sur plusieurs serveurs.

Correspondances CouchDB entre MySQL :

CouchDB	Équivaut en MySQL :
Base non relationnelle. Toutes les données sont dans chaque documents.	Base relationnelle. Il faut effectuer des jointures pour agréger les données.
Base de données	Table
Document	Enregistrement
Design et vues	Webservice

II - DOCUMENT COUCHDB

Les documents sont des objets JSON qui peuvent être hétérogènes. Il n'y a pas de schéma prédéfini de format de documents.

```
{
 "_id": "6e1295ed6c29491254cc05947f18c8af",
 "_rev": "2-de6894cc7b57b662d8f387d52bd24028",
 "date": "2000-01-01",
 "valeur": 500,
 "texte": "blabla"
}
```

Les propriétés `_id` et `_rev` sont gérées par couchdb.

- `_id` est l'identifiant unique du document.
- `_rev` permet de gérer les versions de documents notamment lors des réplications de bases où plusieurs versions peuvent coexister en même temps. Il est également requis lors de la suppression ou de la modification d'un document.

III - INSTALLATION SOUS DEBIAN 9

En cas d'installation derrière un proxy, il est nécessaire de configurer apt.
Pour cela créer un fichier `/etc/apt/apt.conf` contenant les lignes suivantes (adapter le `login` et le `password`) :

```
Acquire::http::Proxy "http://login:password@172.17.2.254:3128/";
Acquire::https::Proxy "http://login:password@172.17.2.254:3128/";
```

Sur une debian 9 installer le paquet `dirmngr` :

```
apt-get install  dirmngr
```

Rajouter le paquet `couchdb`:

```
$ sudo apt-get install -y apt-transport-https gnupg ca-certificates
$ echo "deb https://apache.bintray.com/couchdb-deb stretch main" \
    |  tee /etc/apt/sources.list.d/couchdb.list
```

Installer la clé du dépôt:

```
$ apt-key adv --keyserver keyserver.ubuntu.com --recv-keys \
  8756C4F765C9AC3CB6B85D62379CE192D401AB61
```

Mettre à jour le cache et installer le paquet :

```
$ apt update
$ apt install -y couchdb
```

Ouvrir dans un navigateur web :

```
172.17.XX.XX:5984
```

La réponse doit être de la forme :

```
{"couchdb":"Welcome","version":"2.2.0","git_sha":"2a16ec4","features":["pluggable-
storage-engines","scheduler"],"vendor":{"name":"The Apache Software Foundation"}}
```

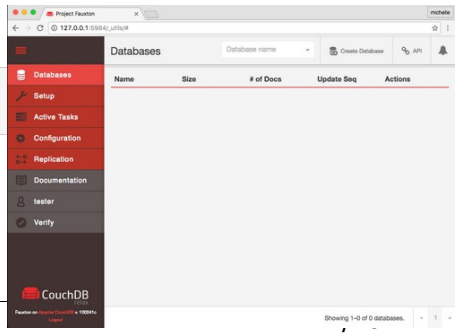
Avec l'outil d'administration Fauxton (voir ci-après) :

- Saisir les identifiants d'accès.
- Mettre `0.0.0.0` en `bind address` afin d'être accessible à distance.

IV - ADMINISTRATION FAUXTON

L'outil Fauxton est un page administration web. Il équivaut à phpmyadmin pour MySQL. Le port par défaut est le 5984.

```
http://adresseIP:5984/_utils/
```

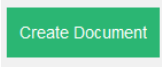


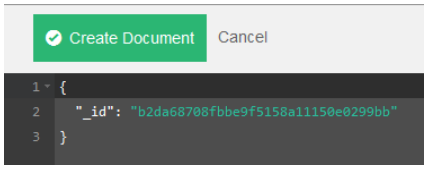
C'est l'outil à utiliser pour :

- Gérer (créer, modifier, supprimer) une base,
- Gérer les documents,
- Gérer les vues,
- Configurer la réplication de base.

http://couchdb.apache.org/fauxton-visual-guide/#intro

Pour créer une nouvelle base, il suffit de renseigner son nom.

Il est alors possible de créer un document. 



L'identifiant `_id` est fourni par CouchDB. Il suffit de compléter le document en respectant le format JSON.

V - LES VUES : MAP REDUCE

V.1. *Présentation*

Les données n'étant pas à priori structurées (comme dans des tables SQL), CouchDB utilise les **vues** pour apporter une **représentation efficace** des données.
Le concept de vue équivaut au développement d'un web service qui répond un flux JSON.
Les vues doivent être conçues de façon à retourner exactement les données nécessaires au client.
Quelle que soit la forme du document, la réponse sera de la forme objet JSON avec les attributs `key` et `value`.

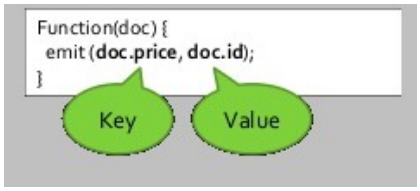
```
{
      "id":"114c927e993c31d12c94bfe99657aba1",
      "key":"2017-02-02",
      "value":4
}
```

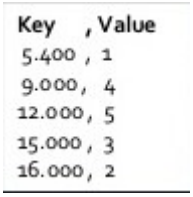
Les manipulations REST (tris, filtrage...) sont réalisées par rapport à la clé. Il faudra générer la clé judicieusement avec une valeur ou plusieurs valeurs (par exemple : date et numéro de capteur...) regroupées dans un tableau.

Les vues sont codées en Javascript et reposent sur le concept de `Map/Reduce`.
Map construit une liste de paires clé/valeurs grâce à la fonction `emit` sur l'ensemble des documents.

Soit la base de données formée par les documents suivants :

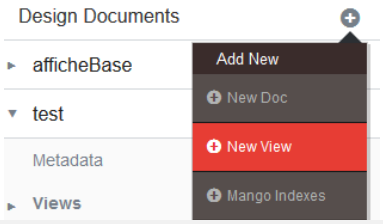
Map construit une liste de paires clé/valeurs grâce à `emit`.





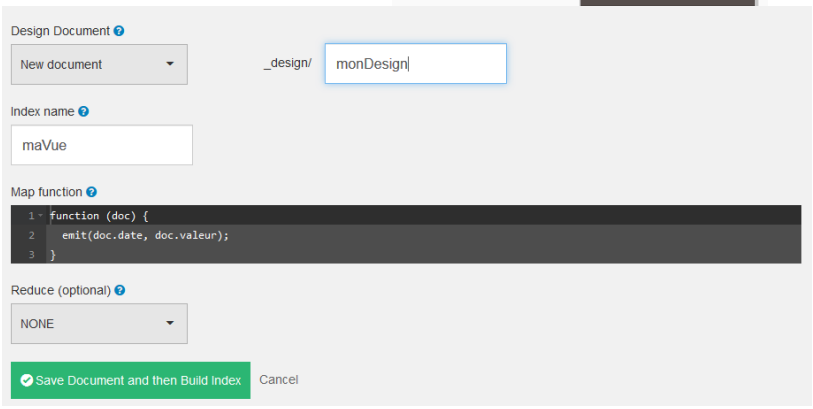
V.2. Création d'une vue

Pour créer une vue, sous Fauxton, sélectionner une base puis développer le menu « Design Documents », puis « New View ».



Les vues sont regroupées dans un « Design Document » à créer.

La partie « Map function » permet de déclarer les vues en Javascript. Les vues sont stockées elles même en JSON comme tout document CouchDB.




```
{
  "_id": "_design/monDesign",
  "_rev": "1-8606c636f7665a0538585e9974e5dc54",
  "views": {
    "maVue": {
      "map": "function (doc) { emit(doc.date, doc.valeur);}"
    }
  },
  "language": "javascript"
}
```

Pour appeler une vue, il faut faire une requête GET à l'url :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/maVue
```

La réponse sera alors du type :

```
{
"total_rows":5,
"offset":0,
"rows":[
      {"id":"6afd5d78fa56703bf98e5429e22179","key":"1999-02-02","value":9},
      {"id":"52237e0443856195c0035104a4af8688","key":"2014-02-02","value":9},
      {"id":"7b24c867b088c3107d9353a0ca122957","key":"2015-01-02","value":8},
      {"id":"e917ac1a8c4e5886990e2d75987297f6","key":"2015-02-02","value":4},
      {"id":"5eca785e9af31a43ba81a03449e31b5a","key":"2018-02-02","value":83}
      ]
}
```

V.3. Statistiques

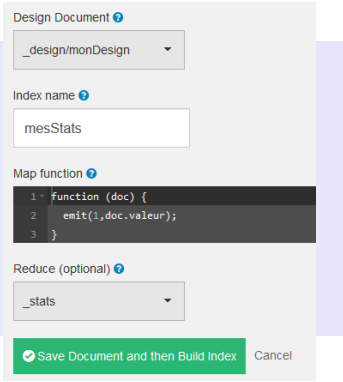
L'option « Reduce » permet de générer des statistiques sur les valeurs générées par la fonction `emit` sur l'ensemble des documents.

Exemple de vue utilisant l'option `_stats` :
Url :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/mesStats
```

Réponse :

```
{
"rows":[
      {"key":null,
      "value":
      {"sum":113,"count":5,"min":4,"max":83,"sumsqr":7131}
      }
      ]
}
```



VI - API HTTP

Tout peut être réalisé via des requêtes HTTP via une API REST. Il est cependant plus simple d'utiliser Fauxton pour manipuler la base de données. On se limitera via HTTP à manipuler les documents en lecture, écriture, mise à jour et suppression.

VI.1. Requêtes HTTP

Tout outil ou librairie qui gère HTTP peut être utilisé pour manipuler une base CouchDB.
Par exemple l'utilitaire en ligne de commande curl :

```
curl -X GET http://adresseIP:5984/mabase/_design/monDesign/_view/mesStats
```

On peut approximativement faire correspondre la sémantique des méthodes HTTP avec les 4 opérations élémentaires d'une base de données (**CRUD** – *Create, Read, Update et Delete*).
L'interface REST utilise les verbes HTTP :

Opération	Méthode HTTP	Équivalence SQL
Create	POST (ou PUT)	INSERT
Read	GET	SELECT
Update	PUT	UPDATE
Delete	DELETE	DELETE

VI.2. Lecture des documents

```
GET /{db}/_design/{ddoc}/_view/{view}
```

Tous les documents (par défaut classés par clé ascendante) :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/maVue
```

Limiter par valeur de clé :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/maVue?key="2000-01-01"
```

Limiter avec une valeur de clé minimale :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/maVue?startkey="2002-01-01"
```

Limiter le nombre de valeurs :

```
http://adresseIP:5984/mabase/_design/monDesign/_view/maVue?limit=3
```

Ordre inverse :

```
http://adresseIP:5984/mabase/_design/test/_view/new-view?descending=true
```

Dernière valeur :

```
http://adresseIP:5984/mabase/_design/test/_view/new-view?limit=1&descending=true
```

VI.3. Ajout d'un document

C'est l'application qui insère les valeurs qui doit fournir l'`_id` (mot binaire de 128bits aléatoire) du nouveau document. CouchDB est développé pour être géré par plusieurs maîtres éventuellement; l'auto incrémentation n'est donc pas utilisée...

CouchDB peut fournir un `_id` non utilisé. Pour obtenir un `uid.`

```
http://adresseIP:5984/_uuids
```

Ajout d'un document

```
curl -X PUT http://adresseIP:5984/mabase/6e1295ed6c29495e54cc05947f18c8af -d
"{""date"":""2000-01-01"",""valeur"":50}"
```

VI.4. Effacement d'un document

Il faut récupérer le `_rev` du document en utilisant l'`_id` et faire une requête `DELETE.`
Requête pour obtenir le `_rev` à partir de l'`_id` :

```
http://adresseIP:5984/mabase/6e1295ed6c29491254cc05947f18c8af
```

Réponse :

```
{"_id":"6afd5d78fa56703bf98e5429e22179",
"_rev":"1-7d79035ee8fdcd86e023334cc2d9df8d",
"date":"1999-02-02",
"valeur":9}
```

Requête DELETE

```
curl -X DELETE http://adresseIP:5984/mabase/6e1295ed6c29491254cc05947f18c8af?rev=3-6477808596fbbd0e2a5c8534cc4b23e3
```

VI.5. Modification d'un document

Comme pour la suppression il faut d'abord récupérer le _rev du document à modifier.
Requête pour obtenir le `_rev` à partir de l'`_id` :

```
http://adresseIP:5984/mabase/3167193d328a2b673959f38d393d126/
```

Réponse :

```
{
"_id":"3167193d328a2b673959f38d393d126",
"_rev":"1-7a938ef1c014f5cfc3e046bee00ae9e8",
"date":"2015-02-02",
"valeur":1
}
```

Requête PUT de modification du document :

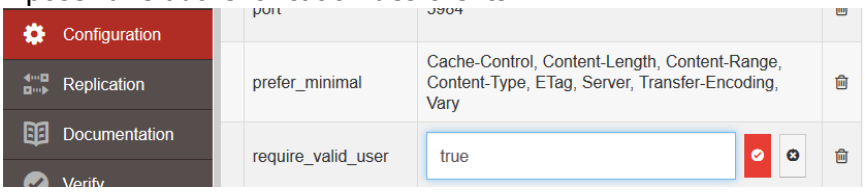
```
curl -X PUT http://adressIP:5984/mabase/3167193d328a2b673959f38d393d126/ -d "
{ ""valeur"" : ""24"" , ""_rev"" :""1-7a938ef1c014f5cfc3e046bee00ae9e8"" } "
```

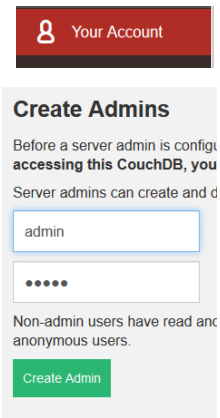
Réponse (avec un nouveau numéro de révision `rev`):

```
{
"ok":true,"id":"3167193d328a2b673959f38d393d126",
"rev":"2-e112ae97e35437fa68ee7f29865240f3"
}
```

VII - AUTHENTIFICATION BASIQUE

Avant de créer un administrateur, tous les clients ont des privilèges administrateurs...
Dans l'outil Fauxton, créer un administrateur de serveur. Renseigner un identifiant et un mot de passe (par exemple admin/admin).
Dans la page de configuration, passer l'option `require_valid_user` à `true` pour imposer une authentification des clients HTTP.





Les clients peuvent alors utiliser l'authentification basique (RFC 2617) pour effectuer les requêtes HTTP.

Requête non authentifiée :

```
curl -X GET http://adresseIP:5984/mabase
```

Réponse :

```
{"error":"unauthorized","reason":"Authentication required."}
```

Requête authentifiée :

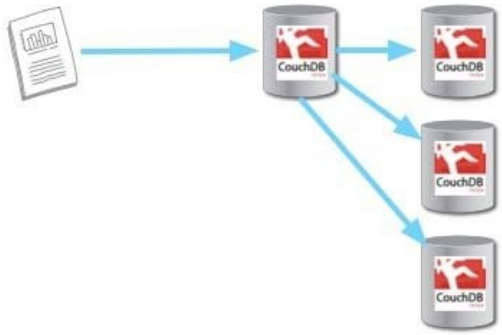
```
curl -X GET http://admin:admin@adresseIP:5984/mabase
```

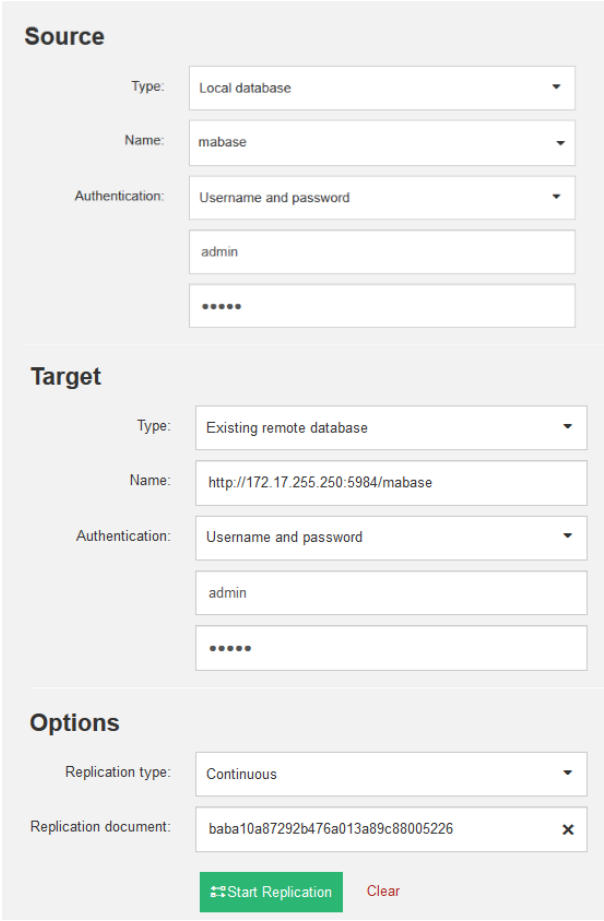
Réponse :

```
{"db_name":"mabase","update_seq":"1...}
```

VIII - REPLICATION

La réplication consiste à synchroniser deux copies de la même base de données. Ces bases peuvent être hébergées sur le même serveur CouchDB ou sur deux serveurs différents.
Il faut configurer la base source et la base destination. Tout document modifié sur la base source sera immédiatement mis à jour sur la base cible. Les vues étant des documents seront également répliquées.





IX - CLIENT C

Exemple de client qui insère une valeur en base de données.
Il faut envoyer une requête PUT avec un identifiant unique dans l'url et la donnée formatée en JSON dans le corps de la requête :

```
curl -X PUT http://127.0.0.1:5984/mabase/6e1295e99c29495e54cc05947f18c8af -d "{""date"":""2099-01-01"",""valeur"":50}"
```

La réponse HTTP est `201 CREATED` avec un corps :

```
{"ok":true,"id":"6e1295e99c29495e54cc05947f18c8af","rev":"1-4b57d68024011e5a3bc6f0f75d0619dc"}
```

Si l'identifiant est déjà utilisé la réponse HTTP est `409 CONFLICT` avec un corps :

```
{"error":"conflict","reason":"Document update conflict."}
```

Le paquet `libcurl4-gnutls-dev` doit être installé sur la cible.

```
apt-get install libcurl4-gnutls-dev
```

```
#include <stdio.h>
#include <string.h>
#include <curl/curl.h>

int main(void) {
    CURL *curl;
    CURLcode res;

    static const char *postthis = "{\"date\":\"2099-01-01\",\"valeur\":50}";

    curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL,
      "http://127.0.0.1:5984/mabase/6e1295e99c55495e541105947f18c999");
        curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "PUT");
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postthis);
        res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(res));
        } else {
            long response_code;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &response_code);
            if (response_code != 201) {
                fprintf(stderr, "ERREUR! Code reponse HTTP %ld\n", response_code);
            } else {
                printf("OK!!");
```

```
            }
        }
    }
    /* always cleanup */
    curl_easy_cleanup(curl);
    return 0;

}
```

X - CLIENT PHP

X.1. Page dynamique PHP

```
<?php
	$ch = curl_init();
	$url = "http://admin:admin@127.0.0.1:5984/mabase/_design/monDesign/_view/maVue";
	curl_setopt($ch, CURLOPT_URL,$url);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-type: application/json',
'Accept: */*'));
	$response = curl_exec($ch);
	curl_close($ch);
	$reponse = json_decode($response);
	foreach($reponse->rows as $data){
		echo "valeur = ".$data->value;
		echo " le ";
		echo $data->key."<br>";
	}
?>
```

X.2. Web-service

```
<?php
	$ch = curl_init();
	$url = "http://admin:admin@127.0.0.1:5984/mabase/_design/monDesign/_view/maVue";
	curl_setopt($ch, CURLOPT_URL,$url);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-type: application/json',
'Accept: */*'));
	$response = curl_exec($ch);
	curl_close($ch);
	header('Content-Type: application/json');
	echo $response;
?>
```