# TP1 CouchDB (3h) (Correction)

## 1. Préalable

Vous travaillerez sur le cluster CouchDB (version 3.1) installé sur les serveurs de la DSIN, et accessible aux adresses des trois nœuds disponibles dans le cluster qui sont :
`prodpeda-couchdb3-1.infra.umontpellier.fr:5984`,
`prodpeda-couchdb3-2.infra.umontpellier.fr:5984`
et `prodpeda-couchdb3-3.infra.umontpellier.fr:5984`.

## 2. Création et alimentation de la base de données

Vous définirez au préalable une variable d'environnement nommée COUCH3 dans votre .bashrc pour vous simplifier l'accès au cluster en ligne de commande. Les informations concernant la chaîne de connexion sont données dans un fichier textuel à part.

— Vous définirez une base de données nommée **occitanie** en la préfixant avec votre nom de famille (tout en minuscules). Réalisez cette opération en ligne de commandes avec curl ;

```
-- attention bd en minuscules _ ici un exemple avec le prefixe zoe
 curl -X PUT $COUCH3/zoe_occitanie
-- reponse du systeme
{"ok":true}

-- verifier avec
-- curl -X GET $COUCH3/_all_dbs
```

Listing 1 – Exemple de création de bd

— Vous exploiterez le mode "ajout par lots" via curl pour insérer les documents des fichiers `herault.json`, `gard.json`, `aveyron.json`, `hauteGaronne.json` et `regions_partiel.json` dans la base de données.

```
-- exemple de chargement dans zoe_occitanie
curl -X POST $COUCH3/zoe_occitanie/_bulk_docs -d @herault.json -H "Content-Type:
    application/json"
```

Listing 2 – Exemple de chargement par lots

Le contexte applicatif est le même que celui abordé pour Neo4J, à savoir des informations administratives sur des communes françaises. Des exemples de documents json extraits des fichiers de commune et de région sont proposés ici. Vous noterez le champ **type** qui est une tentative de définition d'un schéma.

```
{
```






```
  "_id": "34100",
  "codeInsee": "34100",
  "longitude": "2.8925",
  "latitude": "43.4858",
  "type": "commune",
  "nom": "FERRIERES-POUSSAROU",
  "dep": "34",
  "old_reg": "91",
  "populations": [
    {
      "pop_1975": 38.0961632716736
    },
    {
      "pop_1985": 35.8391614288578
    },
    {
      "pop_1995": 45.2742630644936
    },
    {
      "pop_2005": 56.941818739832
    },
    {
      "pop_2010": 60.6582030010204
    }
  ]
}
```


Listing 3 – Exemple de commune


```
{
  "_id": "91",
  "reg": "91",
  "chef_lieu_reg": "34172",
  "type": "old_region",
  "new_reg": "occitanie",
  "nom_reg": "LANGUEDOC-ROUSSILLON",
  "departements": [
    {
      "dep": "34",
      "nom_dep": "HERAULT",
      "chef_lieu_dep": "34172"
    },
    {
      "dep": "30",
      "nom_dep": "GARD",
      "chef_lieu_dep": "30189"
    },
    {
      "dep": "11",
      "nom_dep": "AUDE",
      "chef_lieu_dep": "11069"
    },
    {
      "dep": "66",
      "nom_dep": "PYRENEES-ORIENTALES",
      "chef_lieu_dep": "66136"
    }
  ]
}
```

Listing 4 – Exemple d'ancienne région

```
{
  "_id": "occitanie",
  "chef_lieu_reg": "31555",
  "nom_reg": "Occitanie",
  "type": "region",
  "president": {
    "nom": "Delga",
    "prenom": "Carole"
  }
}
```

Listing 5 – Exemple de nouvelle région

## 3. Appropriation de la base de données

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. lister les informations générales concernant le serveur couchdb, à l'aide du mécanisme GET

```
curl -X GET $COUCH3/
```

Listing 6 – Informations serveur

2. lister les informations générales concernant concernant la base occitanie, à l'aide du mécanisme GET. Pouvez vous connaître le nombre de documents contenus dans la base occitanie ?

```
curl -X GET $COUCH3/zoe_occitanie/
 avec "doc_count"
-- remarquer la philosophie tout est document y compris la base
```

Listing 7 – Informations bd

3. lister tous les documents de la BD

```
curl -X GET $COUCH3/zoe_occitanie/_all_docs
```

Listing 8 – Liste des documents

4. faire afficher le contenu d'un document. Quel est son numéro de révision ? Comment savoir si ce document a déjà été modifié ?

```
curl -X GET $COUCH3/zoe_occitanie/73/
regarder l'attribut "_rev" et sa valeur : le premier chiffre est un compteur
    1, 2, 3 etc en fonction des modifications
```

Listing 9 – Infos document



## 4. Définition de vues

Vous définirez des vues (en javascript) dans votre base pour consulter les documents décrivant les régions et les communes. Ce travail peut se faire soit en ligne de commande avec curl, soit avec l'interface Web Fauxton (depuis un navigateur à l'adresse prodpeda-couchdb3-2.infra.umontpellier.fr :5984/_utils/).

### 4.1 MAP seulement

1. donnez toutes les informations sur les régions (de type old_region) de la base

```
-- juste la vue javascript a coder
function(doc) { if (doc.type=="old_region") {
  emit(doc._id, doc); }
}

-- exemple en ligne de commande avec le document json correspond a la vue
curl -X PUT $COUCH3/zoe_occitanie/_design/Q1 -d '{"views":{"v1":
    {"map":"function(doc) { if (doc.type=='old_region') {emit(doc._id, doc) ;} }"
    }}}' -H "Content-Type: application/json"
{"ok":true,"id":"_design/Q1","rev":"1-2ae053c303a0125fba76a79a416fe39d"}

-- possibilite de mettre le document json dans un fichier
curl -X PUT $COUCH3/zoe_occitanie/_design/d2 -d @vue.json -H "Content-Type:
    application/json"
{"ok":true,"id":"_design/d2","rev":"1-aa50cfb893896efbe5d9cc0f3ab99214"}
```

Listing 10 – Map1

2. donner les noms (clés) et latitude et longitude de chaque commune

```
-- juste la vue javascript a coder
function(doc) { if (doc.type=="commune") {
  emit(doc.nom, {"latitude":doc.latitude, "longitude":doc.longitude} ); }
}
```

Listing 11 – Map2

3. donner le code insee (clé), le département, la latitude et la longitude de MONTPELLIER (nom de la commune)

```
-- juste la vue javascript a coder
function(doc) { if (doc.nom=="MONTPELLIER") {
  emit(doc.codeInsee, [{"departement":doc.dep}, {"latitude":doc.latitude},
      {"longitude":doc.longitude}] ); }
}
```

Listing 12 – Map3

4. donnez le nom et le prénom de la présidente de la région Occitanie

```
-- juste la vue javascript a coder
function (doc) {
  if(doc.type == 'region'){
    if(doc.nom_reg== 'Occitanie'){
      var val = doc.president.prenom + '-' + doc.president.nom
      emit(doc.nom_reg, val);
    }
```

```
  }
}
```

Listing 13 – Map4

### 4.2 MAP et REDUCE

1. donner le nombre de communes au total puis par département et enfin par région (old_region)

```
-- le format json de la vue est donne
"REDUCE_Un":{"map":"function(doc) { if (doc.type==’commune’)\n emit([doc.old_reg,
    doc.dep, doc._id], 1);\n}",
"reduce":"_count"}
```

Listing 14 – MR1

2. donner le nombre d’habitants par commune en 1985

```
-- la vue javascript a coder identique au partitionnement niveau 3
"REDUCE_Deux":{"map":"function(doc) { if (doc.type==’commune’)\n
for (var i = 0; i<doc.populations.length; i++)\n{ \nvar pop = doc.populations[i];\n
if (pop.pop_1985)\n emit([doc.old_reg, doc.dep, doc.nom], pop.pop_1985);\n\n}\n}",
"reduce":"_sum"}
```

Listing 15 – MR2

3. donner le nombre d’habitants par département en 1985

```
-- partitionnement niveau 2
```

Listing 16 – MR3

4. donner le nombre d’habitants par région (anciennes régions) en 1985

```
-- partitionnement niveau 1
```

Listing 17 – MR4

### 4.3 Autres requêtes

1. donner les communes qui ont vu leurs populations décroître entre 1985 et 1995

```
function(doc)
{ if (doc.type==’commune’)
{ for (var t in doc.populations)
{ var pop = doc.populations[t];
if (pop.pop_1975)
{var p75 = doc.populations[t].pop_1975;}
if (pop.pop_1985)
{var p85 = doc.populations[t].pop_1985;}
}
var test = p85 - p75;
if (test < 0)
{emit(doc._id, [doc.nom, p75, p85, test]);}
}
}
```



Listing 18 – Communes en déclin

2. donner les informations sur la nouvelle région Occitanie ainsi que sur les anciennes régions Languedoc-Roussillon et Midi-Pyrénées (forme de jointure)

```
"REDUCE_Quatre":{"map":"function(doc) { if (doc.type==’region’)\n
{ \n emit([doc._id, 0], doc.nom_reg);\n}\n
if (doc.type==’old_region’)\n{ \n emit([doc.new_reg, 1], doc.nom_reg);\n}\n}"}}
```

Listing 19 – Collation view

## 5. Distribution de la base de données

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. quels sont les nœuds mobilisés pour l’organisation du serveur, et la gestion de la base ?

```
curl -X GET $COUCH3/_membership
-- les 3 noeuds prodpeda-couchdb3-1.infra.umontpellier.fr,
    prodpeda-couchdb3-2.infra.umontpellier.fr,
-- prodpeda-couchdb3-3.infra.umontpellier.fr
-- participent a la mise en oeuvre du systeme et des bases de donnees
```

Listing 20 – Les nœuds

2. Combien de partitions sont définies (avant recopie) ? Quel est le nombre de copies ? Combien de partitions répliquées sont définies ?

```
 2 partitions qui sont repliquees sur les 3 noeuds
 curl -X GET $COUCH3/zoe_occitanie/_shards
{"shards":{"00000000-7fffffff":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-3.infra.umontpellier.fr"],
"80000000-ffffffff":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr","couchdb@prodpeda-
 couchdb3-3.infra.umontpellier.fr"]}}
```

Listing 21 – Shards et copies

3. Comment savoir dans quelle(s) partition(s) se trouve un des documents de la base ?

```
curl -X GET $COUCH3/zoe_occitanie/_shards/34172
{"range":"00000000-7fffffff","nodes":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-3.infra.umontpellier.fr"]}
 dans la premiere partition intervalle "00000000-7fffffff"
```

Listing 22 – Shards et documents

4. Est ce que des copies de toutes les partitions sont présentes sur tous les nœuds ?

```
oui visible avec curl -X GET $COUCH3/zoe_occitanie/_shards
```

Listing 23 – Shards et nœuds

5. Est que toutes les lectures sont consistentes ?

```
dans la grande majorite des cas, oui. On lit 2 copies (quorum en lecture) avant de
    retourner la reponse, il y a au moins une copie qui est a jour sur les 3
    partitions et deux qui pour des temps tres courts peuvent ne pas etre a jour
    mais on a de fortes probabilites de lire une copie qui est la derniere sur les
    deux lues
voir "cluster":{"q":2,"n":3,"w":2,"r":2} quand curl -X GET $COUCH3/zoe_occitanie/

-- avec q = nombre de shards, n = nombre de copies, r = quorum en lecture (nombre
    de shards parcourus avant lecture),
w = quorum en ecriture (nombre de shards lus avant ecriture)
```

Listing 24 – Nombre de lectures

# TP2 CouchDB

## 1. Création et alimentation d'une base de données vaccination

### 1.1 Sans schéma et structure JSON

L'objectif du premier exercice est de réfléchir en amont à la structuration des documents JSON. Cette structuration est essentielle puisque une base de données CouchDB est dite sans schéma. Il est donc impératif de veiller en amont à disposer d'une auto-description des documents qui puisse faciliter ensuite l'organisation des documents et la consultation des données.

— Vous créerez une nouvelle base de données nommée **vaccination** en la préfixant avec votre nom de famille (tout en minuscules). Réalisez cette opération en ligne de commandes avec curl, et en modifiant le nombre de partitions pour la base ;

```
curl -X PUT $COUCH3/zzz_vaccination?q=8
```

Listing 1 – Exemple changement nombre partitions

— Un diagramme de classes est donné, qui reprend la structure qui sera à retrouver. Pour des raisons d'anonymat des patients, les doses (dose1 pour la première injection, dose2 pour la seconde injection, dose3 pour la troisième injection) pour chaque vaccin sont comptabilisées au niveau du département à une date donnée (jour). Les vaccins sont à prendre parmi Pfizer, Moderna, AstraZeneca, Janssen et TousVaccins (la somme des 4). Vous noterez le champ **type** qui est un début de définition d'un schéma, et ce, quelle que soit la structuration (fichier) choisie.





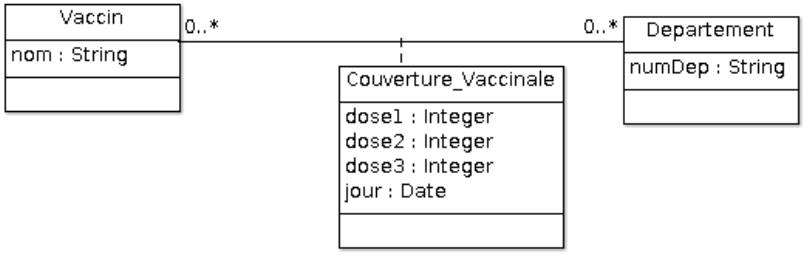


FIGURE 1 – Diagramme de classes pour l'application

```
{
"_id": "12_27-DEC-20_AstraZeneka",
"dep": "12",
"jour": "27-DEC-20" ,
"type": "couverture_vaccinale" ,
"doses": [ {"vaccin": "AstraZeneka" } , {"dose1": 0 } , { "dose2": 0 } , {
"dose3": 0} ]
}
```

Listing 2 – Exemple un (vaccin1.json)

```
{
"jour": "04-JAN-21",
"dep": "12",
"type": "couverture_vaccinale",
"vaccinations": [
{"vaccin": "AstraZeneka", "doses": [ {"dose1": 0 }, { "dose2": 0 }, {
   "dose3":
0} ]} ,{"vaccin": "Janssen", "doses": [ {"dose1": 0 }, { "dose2": 0 }, {
"dose3": 0} ]} ,{"vaccin": "Moderna", "doses": [ {"dose1": 0 }, { "dose2":
   0 },
{ "dose3": 0} ]} ,{"vaccin": "Pfizer", "doses": [ {"dose1": 1 }, {
   "dose2": 0 },
{ "dose3": 0} ]} ,{"vaccin": "TousVaccins", "doses": [ {"dose1": 1 }, {
   "dose2":
0 }, { "dose3": 0} ]} ]
}
```

Listing 3 – Exemple deux (vaccin2.json)

— Vous ferez un choix entre les deux structures proposées. Vous expliquerez les raisons de ce choix (en accord avec les transparents du cours sur les bonnes pratiques). Les deux structurations proposées sont discutables et pourraient être améliorées. Vous ferez des propositions d'amélioration.
— Vous exploiterez le mode "ajout par lots" via curl pour insérer les documents du fichier `vaccin1.json` ou `vaccin2.json` en fonction de votre choix concernant la structuration.

## 2. Appropriation de la base de données

### 2.1 Ajout de documents de type "departement"

Vous ajouterez également les quatres documents du fichier `departements.json`, et vous construirez la fonction map/reduce suivante :

```
function (doc) {
  if (doc.type==’departement’)
  emit(doc.type,doc.population);
}

function (keys, values, rereduce) {
 return Math.min.apply({}, values);

}
```

Listing 4 – Exemple map/reduce

— que renvoie cette requête (au niveau map, puis au niveau reduce et enfin au niveau rereduce) ?

```
map : renvoie type departement (cl\’e) et population pour chaque
    departement
reduce : retourne la population du departement qui a la plus faible
    population.
rereduce : idem que reduce
```

Listing 5 – Correction

### 2.2 Jeu de questions à définir en fonction du modèle choisi

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. lister les informations générales concernant concernant la base vaccination, à l'aide du mécanisme GET. Pouvez vous connaître le nombre de documents contenus dans la base ?

```
curl -X GET $COUCH3/vaccination/
avec "doc_count"
```

Listing 6 – Correction



2. lister tous les documents de la BD

```
curl -X GET $COUCH3/vaccination/_all_docs
```

Listing 7 – Correction

3. faire afficher le contenu d'un document.

```
curl -X GET $COUCH3/vaccination1/12_27-DEC-20_AstraZeneka/
 regarder l’attribut "_rev" et sa valeur : le premier chiffre est un
    compteur 1, 2, 3 etc en fonction des modifications

% pour choix 2, _id attribue par le system avec uuid
% curl -X GET $COUCH3/vaccination2/90b092dc35e079c5ea6f405c5a24978d
```

Listing 8 – Correction

## 3. Définition de vues

Vous définirez des vues (en javascript) dans votre base pour consulter les documents décrivant les couvertures vaccinales. Ce travail peut se faire soit en ligne de commande avec curl, soit avec l'interface Web Fauxton (depuis un navigateur à l'adresse prodpeda-couchdb3-2.infra.umontpellier.fr :5984/_utils/).

### 3.1 MAP et MAP/REDUCE

1. renvoyez pour tous les documents de type couverture_vaccinale du département de l'Hérault (34), l'identifiant du document (clé doc._id) et le jour de vaccination (valeur doc.jour)

```
vaccination1
function (doc) { if ((doc.type=’couverture_vaccinale’ && doc.dep==’34’))
    emit(doc._id, doc.jour); }
vaccination2
function (doc) { if (doc.type==’couverture_vaccinale’ && doc.dep==34)
    emit(doc._id, doc.jour);}
```

Listing 9 – Correction

2. donnez le nombre de documents de type couverture_vaccinale du département de l'Hérault (34) (clé doc.dep, valeur 1)

```
pour les deux modeles
map : function (doc) { if (doc.type==’couverture_vaccinale’ &&
    doc.dep==34) emit(doc.dep, 1); }
 "reduce": "_count"
vaccination1 1450 documents et vaccination2 290 documents
```

Listing 10 – Correction

3. donnez le nombre de documents de type couverture_vaccinale du département de l'Hérault (34) pour chaque année écoulée. Un exemple de manipulation de date est fourni.

```
var d = new Date(doc.jour) ;
-- fonctions pour retourner annee ou mois
```

```
-- d.getFullYear() ou d.getMonth()
```

Listing 11 – Manipulation date

```
pour les deux modeles
map : function (doc) { if (doc.type=='couverture_vaccinale' && doc.dep==34)
{ var d = new Date(doc.jour) ;
  emit(d.getFullYear(), 1);}
}

reduce avec _count
```

Listing 12 – Correction

4. donnez le nombre de documents de type couverture_vaccinale par département et par année écoulée.

```
pour les deux modeles
 function (doc) { if (doc.type=='couverture_vaccinale')
{ var d = new Date(doc.jour) ;
  emit([doc.dep, d.getFullYear()], 1);
}
}
 reduce avec _count
```

Listing 13 – Correction

5. donnez pour les documents de type couverture_vaccinale pour le vaccin Pfizer, l'identifiant du document (clé doc._id) et la date et le département de vaccination (valeur : doc.jour et doc.dep)

```
 vaccination1
function (doc) { if (doc.type=='couverture_vaccinale' &&
    Array.isArray(doc.doses))
{ for (var dos in doc.doses) { if (doc.doses[dos].vaccin &&
    doc.doses[dos].vaccin=='Pfizer')
 emit(doc._id, {"date":doc.jour, "dep": doc.dep});
}}}
 vaccination2
function (doc) { if (doc.type=='couverture_vaccinale' &&
    Array.isArray(doc.vaccinations))
{ for (var dos in doc.vaccinations) { if
    (doc.vaccinations[dos].vaccin=='Pfizer')
 emit(doc._id, {"date":doc.jour, "dep":doc.dep});
}}}
```

Listing 14 – Correction

6. donnez le nombre de documents de type couverture_vaccinale pour le vaccin Pfizer par département

```
 vaccination1
function (doc) { if (doc.type=='couverture_vaccinale' &&
    Array.isArray(doc.doses))
{ for (var dos in doc.doses) { if (doc.doses[dos].vaccin &&
    doc.doses[dos].vaccin=='Pfizer')
 emit(doc.dep, 1);
```



```
}}}
 _count
 vaccination2
function (doc) { if (doc.type=='couverture_vaccinale' &&
    Array.isArray(doc.vaccinations))
{ for (var dos in doc.vaccinations) { if
    (doc.vaccinations[dos].vaccin=='Pfizer')
 emit(doc.dep, 1);
}}}
 _count
```

Listing 15 – Correction

7. donnez le nombre de documents de type couverture_vaccinale pour le vaccin Pfizer par département, par mois et par an

```
 vaccination1
function (doc) { if (doc.type=='couverture_vaccinale' &&
    Array.isArray(doc.doses))
{ for (var dos in doc.doses) { if (doc.doses[dos].vaccin &&
    doc.doses[dos].vaccin=='Pfizer')
{ var dte = new Date(doc.jour) ;
 emit([doc.dep, dte.getFullYear(), dte.getMonth()+1], 1);
}}}}
 _count
 vaccination2
 function (doc) { if (doc.type=='couverture_vaccinale' &&
     Array.isArray(doc.vaccinations))
 { for (var dos in doc.vaccinations) { if
     (doc.vaccinations[dos].vaccin=='Pfizer') {
  var dte = new Date (doc.jour) ;
 emit([doc.dep, dte.getFullYear(), dte.getMonth()+1], 1);
 }}}}
 _count
```

Listing 16 – Correction

8. donnez la somme de dose1 (et autres statistiques) pour le vaccin Pfizer par département, par mois et par an

```
 vaccination1
   "map": "function (doc) { if ((doc.type=='couverture_vaccinale') &&
       (Array.isArray(doc.doses)) )\n { for (var v in doc.doses)\n { if
       (doc.doses[v].vaccin=='Pfizer') {\n for (var d in doc.doses)\n{ if
       (doc.doses[d].dose1) \n{ var dte = new Date(doc.jour) ;\n
       emit([doc.dep, dte.getFullYear(), dte.getMonth()+1],
       doc.doses[d].dose1); } } }\n }\n }\n}",
     "reduce": "_stats"

 vaccination 2
 function (doc) { if (doc.type=='couverture_vaccinale' &&
     Array.isArray(doc.vaccinations))
 { for (var dos in doc.vaccinations) { if
     (doc.vaccinations[dos].vaccin=='Pfizer') {
  var dte = new Date (doc.jour) ; var tabPfizer = doc.vaccinations[dos] ;
```

```
for (var d1 in tabPfizer.doses)
{ if (tabPfizer.doses[d1].dose1 ) {
emit([doc.dep, dte.getFullYear(),
    dte.getMonth()+1],tabPfizer.doses[d1].dose1); } }
}}}}
    "reduce": "_stats"
```

Listing 17 – Correction

## 4. Distribution de la base de données

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. Combien de partitions sont définies (avant recopie) ? Quel est le nombre de copies ? Combien de partitions répliquées sont définies ?

```
Si q=8 8 partitions sont repliquees sur les 3 noeuds donc 24
%curl -X GET $COUCH3/zoe_vaccine_3/_shards
%{"shards":{"00000000-1fffffff":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
```

Listing 18 – Correction

2. Comment savoir dans quelle(s) partition(s) se trouve un des documents de la base ?

```
curl -X GET $COUCH3/zoe_vaccine_3/_shards/12_27-DEC-20_AstraZeneka
```

Listing 19 – Correction

3. Est ce que des copies de toutes les partitions sont présentes sur tous les nœuds ?

```
oui visible avec curl -X GET $COUCH3/occitanie/_shards
```

Listing 20 – Correction

# TP1 CouchDB (3h) (Correction)

## 1. Préalable

Vous travaillerez sur le cluster CouchDB (version 3.1) installé sur les serveurs de la DSIN, et accessible aux adresses des trois nœuds disponibles dans le cluster qui sont :
`prodpeda-couchdb3-1.infra.umontpellier.fr:5984`,
`prodpeda-couchdb3-2.infra.umontpellier.fr:5984`
et `prodpeda-couchdb3-3.infra.umontpellier.fr:5984`.

## 2. Création et alimentation de la base de données

Vous définirez au préalable une variable d'environnement nommée COUCH3 dans votre .bashrc pour vous simplifier l'accès au cluster en ligne de commande. Les informations concernant la chaîne de connexion sont données dans un fichier textuel à part.

— Vous définirez une base de données nommée **occitanie** en la préfixant avec votre nom de famille (tout en minuscules). Réalisez cette opération en ligne de commandes avec curl ;

```
-- attention bd en minuscules _ ici un exemple avec le prefixe zoe
 curl -X PUT $COUCH3/zoe_occitanie
-- reponse du systeme
{"ok":true}

-- verifier avec
-- curl -X GET $COUCH3/_all_dbs
```

Listing 1 – Exemple de création de bd

— Vous exploiterez le mode "ajout par lots" via curl pour insérer les documents des fichiers `herault.json`, `gard.json`, `aveyron.json`, `hauteGaronne.json` et `regions_partiel.json` dans la base de données.

```
-- exemple de chargement dans zoe_occitanie
curl -X POST $COUCH3/zoe_occitanie/_bulk_docs -d @herault.json -H "Content-Type:
    application/json"
```

Listing 2 – Exemple de chargement par lots

Le contexte applicatif est le même que celui abordé pour Neo4J, à savoir des informations administratives sur des communes françaises. Des exemples de documents json extraits des fichiers de commune et de région sont proposés ici. Vous noterez le champ **type** qui est une tentative de définition d'un schéma.

```
{
```






```
  "_id": "34100",
  "codeInsee": "34100",
  "longitude": "2.8925",
  "latitude": "43.4858",
  "type": "commune",
  "nom": "FERRIERES-POUSSAROU",
  "dep": "34",
  "old_reg": "91",
  "populations": [
    {
      "pop_1975": 38.0961632716736
    },
    {
      "pop_1985": 35.8391614288578
    },
    {
      "pop_1995": 45.2742630644936
    },
    {
      "pop_2005": 56.941818739832
    },
    {
      "pop_2010": 60.6582030010204
    }
  ]
}
```


Listing 3 – Exemple de commune


```
{
  "_id": "91",
  "reg": "91",
  "chef_lieu_reg": "34172",
  "type": "old_region",
  "new_reg": "occitanie",
  "nom_reg": "LANGUEDOC-ROUSSILLON",
  "departements": [
    {
      "dep": "34",
      "nom_dep": "HERAULT",
      "chef_lieu_dep": "34172"
    },
    {
      "dep": "30",
      "nom_dep": "GARD",
      "chef_lieu_dep": "30189"
    },
    {
      "dep": "11",
      "nom_dep": "AUDE",
      "chef_lieu_dep": "11069"
    },
    {
      "dep": "66",
      "nom_dep": "PYRENEES-ORIENTALES",
      "chef_lieu_dep": "66136"
    }
  ]
}
```

Listing 4 – Exemple d'ancienne région

```
{
  "_id": "occitanie",
  "chef_lieu_reg": "31555",
  "nom_reg": "Occitanie",
  "type": "region",
  "president": {
    "nom": "Delga",
    "prenom": "Carole"
  }
}
```

Listing 5 – Exemple de nouvelle région

## 3. Appropriation de la base de données

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. lister les informations générales concernant le serveur couchdb, à l'aide du mécanisme GET

```
curl -X GET $COUCH3/
```

Listing 6 – Informations serveur

2. lister les informations générales concernant concernant la base occitanie, à l'aide du mécanisme GET. Pouvez vous connaître le nombre de documents contenus dans la base occitanie ?

```
curl -X GET $COUCH3/zoe_occitanie/
 avec "doc_count"
-- remarquer la philosophie tout est document y compris la base
```

Listing 7 – Informations bd

3. lister tous les documents de la BD

```
curl -X GET $COUCH3/zoe_occitanie/_all_docs
```

Listing 8 – Liste des documents

4. faire afficher le contenu d'un document. Quel est son numéro de révision ? Comment savoir si ce document a déjà été modifié ?

```
curl -X GET $COUCH3/zoe_occitanie/73/
regarder l'attribut "_rev" et sa valeur : le premier chiffre est un compteur
    1, 2, 3 etc en fonction des modifications
```

Listing 9 – Infos document



## 4. Définition de vues

Vous définirez des vues (en javascript) dans votre base pour consulter les documents décrivant les régions et les communes. Ce travail peut se faire soit en ligne de commande avec curl, soit avec l'interface Web Fauxton (depuis un navigateur à l'adresse prodpeda-couchdb3-2.infra.umontpellier.fr :5984/_utils/).

### 4.1 MAP seulement

1. donnez toutes les informations sur les régions (de type old_region) de la base

```
-- juste la vue javascript a coder
function(doc) { if (doc.type=="old_region") {
  emit(doc._id, doc); }
}

-- exemple en ligne de commande avec le document json correspond a la vue
curl -X PUT $COUCH3/zoe_occitanie/_design/Q1 -d '{"views":{"v1":
    {"map":"function(doc) { if (doc.type=='old_region') {emit(doc._id, doc) ;} }"
    }}}' -H "Content-Type: application/json"
{"ok":true,"id":"_design/Q1","rev":"1-2ae053c303a0125fba76a79a416fe39d"}

-- possibilite de mettre le document json dans un fichier
curl -X PUT $COUCH3/zoe_occitanie/_design/d2 -d @vue.json -H "Content-Type:
    application/json"
{"ok":true,"id":"_design/d2","rev":"1-aa50cfb893896efbe5d9cc0f3ab99214"}
```

Listing 10 – Map1

2. donner les noms (clés) et latitude et longitude de chaque commune

```
-- juste la vue javascript a coder
function(doc) { if (doc.type=="commune") {
  emit(doc.nom, {"latitude":doc.latitude, "longitude":doc.longitude} ); }
}
```

Listing 11 – Map2

3. donner le code insee (clé), le département, la latitude et la longitude de MONTPELLIER (nom de la commune)

```
-- juste la vue javascript a coder
function(doc) { if (doc.nom=="MONTPELLIER") {
  emit(doc.codeInsee, [{"departement":doc.dep}, {"latitude":doc.latitude},
      {"longitude":doc.longitude}] ); }
}
```

Listing 12 – Map3

4. donnez le nom et le prénom de la présidente de la région Occitanie

```
-- juste la vue javascript a coder
function (doc) {
  if(doc.type == 'region'){
    if(doc.nom_reg== 'Occitanie'){
      var val = doc.president.prenom + '-' + doc.president.nom
      emit(doc.nom_reg, val);
    }
```

```
  }
}
```

Listing 13 – Map4

### 4.2 MAP et REDUCE

1. donner le nombre de communes au total puis par département et enfin par région (old_region)

```
-- le format json de la vue est donne
"REDUCE_Un":{"map":"function(doc) { if (doc.type==’commune’)\n emit([doc.old_reg,
    doc.dep, doc._id], 1);\n}",
"reduce":"_count"}
```

Listing 14 – MR1

2. donner le nombre d’habitants par commune en 1985

```
-- la vue javascript a coder identique au partitionnement niveau 3
"REDUCE_Deux":{"map":"function(doc) { if (doc.type==’commune’)\n
for (var i = 0; i<doc.populations.length; i++)\n{ \nvar pop = doc.populations[i];\n
if (pop.pop_1985)\n emit([doc.old_reg, doc.dep, doc.nom], pop.pop_1985);\n\n}\n}",
"reduce":"_sum"}
```

Listing 15 – MR2

3. donner le nombre d’habitants par département en 1985

```
-- partitionnement niveau 2
```

Listing 16 – MR3

4. donner le nombre d’habitants par région (anciennes régions) en 1985

```
-- partitionnement niveau 1
```

Listing 17 – MR4

### 4.3 Autres requêtes

1. donner les communes qui ont vu leurs populations décroître entre 1985 et 1995

```
function(doc)
{ if (doc.type==’commune’)
{ for (var t in doc.populations)
{ var pop = doc.populations[t];
if (pop.pop_1975)
{var p75 = doc.populations[t].pop_1975;}
if (pop.pop_1985)
{var p85 = doc.populations[t].pop_1985;}
}
var test = p85 - p75;
if (test < 0)
{emit(doc._id, [doc.nom, p75, p85, test]);}
}
}
```



Listing 18 – Communes en déclin

2. donner les informations sur la nouvelle région Occitanie ainsi que sur les anciennes régions Languedoc-Roussillon et Midi-Pyrénées (forme de jointure)

```
"REDUCE_Quatre":{"map":"function(doc) { if (doc.type==’region’)\n
{ \n emit([doc._id, 0], doc.nom_reg);\n}\n
if (doc.type==’old_region’)\n{ \n emit([doc.new_reg, 1], doc.nom_reg);\n}\n}"}}
```

Listing 19 – Collation view

## 5. Distribution de la base de données

Vous répondrez aux questions suivantes en vous aidant de requêtes CURL :

1. quels sont les nœuds mobilisés pour l’organisation du serveur, et la gestion de la base ?

```
curl -X GET $COUCH3/_membership
-- les 3 noeuds prodpeda-couchdb3-1.infra.umontpellier.fr,
    prodpeda-couchdb3-2.infra.umontpellier.fr,
-- prodpeda-couchdb3-3.infra.umontpellier.fr
-- participent a la mise en oeuvre du systeme et des bases de donnees
```

Listing 20 – Les nœuds

2. Combien de partitions sont définies (avant recopie) ? Quel est le nombre de copies ? Combien de partitions répliquées sont définies ?

```
 2 partitions qui sont repliquees sur les 3 noeuds
 curl -X GET $COUCH3/zoe_occitanie/_shards
{"shards":{"00000000-7fffffff":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-3.infra.umontpellier.fr"],
"80000000-ffffffff":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr","couchdb@prodpeda-
 couchdb3-3.infra.umontpellier.fr"]}}
```

Listing 21 – Shards et copies

3. Comment savoir dans quelle(s) partition(s) se trouve un des documents de la base ?

```
curl -X GET $COUCH3/zoe_occitanie/_shards/34172
{"range":"00000000-7fffffff","nodes":["couchdb@prodpeda-couchdb3-1.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-2.infra.umontpellier.fr",
"couchdb@prodpeda-couchdb3-3.infra.umontpellier.fr"]}
 dans la premiere partition intervalle "00000000-7fffffff"
```

Listing 22 – Shards et documents

4. Est ce que des copies de toutes les partitions sont présentes sur tous les nœuds ?

```
oui visible avec curl -X GET $COUCH3/zoe_occitanie/_shards
```

Listing 23 – Shards et nœuds

5. Est que toutes les lectures sont consistentes ?

```
dans la grande majorite des cas, oui. On lit 2 copies (quorum en lecture) avant de
    retourner la reponse, il y a au moins une copie qui est a jour sur les 3
    partitions et deux qui pour des temps tres courts peuvent ne pas etre a jour
    mais on a de fortes probabilites de lire une copie qui est la derniere sur les
    deux lues
voir "cluster":{"q":2,"n":3,"w":2,"r":2} quand curl -X GET $COUCH3/zoe_occitanie/

-- avec q = nombre de shards, n = nombre de copies, r = quorum en lecture (nombre
    de shards parcourus avant lecture),
w = quorum en ecriture (nombre de shards lus avant ecriture)
```

Listing 24 – Nombre de lectures

# Neo4J dans la pratique (TP2)

## 1. Préalable Neo4J

### 1.1 Version Neo4J autre que celle déposée sur moodle

Avant de démarrer le serveur, il faut modifier le fichier neo4j.conf (dans répertoire conf) et ajouter deux instructions (si vous ne travaillez pas avec la version Neo4J déposée sur moodle).

```
# ajout pour import/export RDF
dbms.unmanaged_extension_classes=semantics.extension=/rdf

# ajout pour autoriser export dans un fichier
apoc.export.file.enabled=true
```

Listing 1 – ajouts dans neo4j.conf

Il faut également disposer les plugins adaptés pour les procédures APOC et le plugin NeoSemantics. Pour la version 3.5.21, il s'agit des archives :

```
apoc-3.5.0.14-all.jar
neosemantics-3.5.0.4.jar
```

Listing 2 – archives utiles

### 1.2 Mise en route

Vous démarrerez ensuite le serveur :

```
... ./bin/neo4j start
```

Listing 3 – ordre de mise en route du serveur

## 2. Enrichissement du modèle

Les derniers maires successifs de Montpellier sont à ajouter au modèle

Créer les objets Personne en relation avec la commune de Montpellier (déjà présente dans la base) suivants

```
MATCH (c:Commune {name:'MONTPELLIER'})
CREATE (gf:Personne {nom:"FRECHE",prenom:"GEORGES"}) <-[ap1:ADMINISTREE_PAR
    {date_debut:1997, date_fin:2004}]- (c),
(hm:Personne {nom:"MANDROUX",prenom:"HELENE"})<-[ap2:ADMINISTREE_PAR {date_debut:2004,
    date_fin:2014}]-(c), (ps:Personne
    {nom:"SAUREL",prenom:"PHILIPPE"})<-[ap3:ADMINISTREE_PAR {date_debut:2014,
    date_fin:2020}]-(c), (md:Personne
    {nom:"DELAFOSSE",prenom:"MICKAEL"})<-[ap4:ADMINISTREE_PAR {date_debut:2020,
    date_fin:2026}]-(c)
return *
```

Listing 4 – Les maires de Montpellier





### 2.1 Exercices après enrichissement

1. définir l'ordre Cypher qui permet d'ajouter un label Maire aux objets de label Personne, qui sont associés à un objet de label Commune, au travers d'une relation de type ADMINISTREE_PAR

```
%MATCH (p:Personne) <-[:ADMINISTREE_PAR]- (c:Commune) SET p:Maire
```

Listing 5 – Correction

2. lister l'ensemble des procédures rendues disponibles grâce à l'ajout d'archives Java dans le répertoire plugins. Lister aussi l'ensemble des fonctions.

```
CALL dbms.procedures()
CALL dbms.functions()
```

Listing 6 – Correction

3. utiliser une de ces procédures du paquetage d'extension général (préfixe apoc) pour export une partie des objets du graphe au format json (apoc.export.json.query). Vous renverrez l'identifiant, les labels et le nom de la commune, ainsi que le non de son département et de sa région. Quels sont les autres formats disponibles ?

```
call apoc.export.json.query("MATCH (a:Commune) -[w:WITHIN]-> (d:Departement)
Return id(a), labels(a), a.name, d.name", "test.json", {} )

CSV ou GRAPHML par exemple
call apoc.export.graphml.all("test.xml", {} ) pour ouvrir le fichier avec un autre
    editeur de graphe
```

Listing 7 – Correction

#### 2.1.1 Utilisation du plugin Neosemantics

Neosemantics permet d'exploiter Neo4J à la manière d'un triplestore. Nous en explorons quelques fonctionnalités élémentaires.

1. vous renverrez le modèle de connaissances de la base au format RDF. Vous pouvez dessiner sous forme de graphe une partie du résultat. Que remarquez vous comme différence avec ce que vous savez contenir le modèle ?

```
:GET /rdf/onto

absence de toutes les proprietes concretes des noeuds comme etant des relations
```

Listing 8 – Correction

2. Vous renverrez au format RDF, la description du noeud correspondant à la commune de MONTPELLIER à partir de son identifiant interne. Vous pouvez dessiner sous forme de graphe une partie du résultat.

```
MATCH (m:Commune {name:'MONTPELLIER'}) return ID(m) pour avoir l'ID

ensuite par exemple si ID : 37
:GET /rdf/describe/id/37
```

Listing 9 – Correction

3. Renvoyez le résultat de la requête suivante au format RDF

```
MATCH (c:Commune {name:'MONTPELLIER'}) RETURN c
```

Listing 10 – Requête

Est ce que ce résultat est équivalent au résultat de la description du noeud de la commune de MONTPELLIER ?

```
:POST /rdf/cypher { "cypher":"MATCH (c:Commune {name:'MONTPELLIER'}) RETURN c" ,
    "format" : "N3"}

ici juste les proprietes litterales donc non ce n'est pas equivalent
```

Listing 11 – Correction

4. Renvoyez les informations sur MONTPELLIER et ses différents maires au format RDF

```
:POST /rdf/cypher { "cypher":"MATCH (c:Commune {name:'MONTPELLIER'})
    -[ADMINISTREEPAR]-> (m:Maire) RETURN *" , "format" : "N3"}
```

Listing 12 – Correction

5. Enrichissez l'ordre précédent pour renvoyer le plus d'informations possibles sur MONTPELLIER

```
:POST /rdf/cypher { "cypher":"MATCH (r:Region) <-[w2:WITHIN]- (d:Departement)
    <-[w1:WITHIN]- (c:Commune {name:'MONTPELLIER'})-[ap1:ADMINISTREE_PAR]->
    (p:Personne), (c) -[:NEARBY]- (c1:Commune) RETURN * " , "format" : "N3"}
```

Listing 13 – Correction

6. Comment faire pour renvoyer les informations qui correspondent aux propriétés valuées de la relation ADMINISTREE_PAR au format RDF ?

```
:POST /rdf/cypher { "cypher":"MATCH (c:Commune {name:'MONTPELLIER'})
    -[ap1:ADMINISTREE_PAR]-> (p:Personne) MERGE (c)-[:gouvernance]->
    (m:Municipalite {dateDeb:ap1.date_debut, dateFin:ap1.date_fin})
    <-[:dirige]-(p) RETURN c, m, p" , "format" : "N3"}
```

Listing 14 – Correction



## 3. Import de données au format RDF

Il est possible d'importer au sein d'une base de données Neo4J, des triplets provenant de points d'accès SPARQL. Ici une requête SPARQL exploitant une des procédures du plugin neosemantics (semantics.importRDF) vous est donnée. Cette requête de type CONSTRUCT exploite le point d'accès SPARQL de Wikidata pour retourner différentes informations sur des communes (ici le concept City est emprunté au vocabulaire schema.org). Auparavant, il faudra créer l'index :

```
CREATE INDEX ON :Resource(uri)
```

Listing 15 – Requête CYPHER/SPARQL

```
WITH ' PREFIX sch: <http://schema.org/>
CONSTRUCT{ ?item a sch:City;
        sch:address ?inseeCode;
        sch:name ?itemLabel ;
        sch:geoTouches ?otherItem .
     ?otherItem a sch:City;
     sch:name ?otheritemLabel ;
     sch:address ?otherinseeCode . }
WHERE { ?item wdt:P374 ?inseeCode .
   ?item wdt:P47 ?otherItem .
   ?otherItem wdt:P374 ?otherinseeCode .
      ?item rdfs:label ?itemLabel .
       filter(lang(?itemLabel) = "fr") .
      ?otherItem rdfs:label ?otheritemLabel .
       filter(lang(?otheritemLabel) = "fr") .
    FILTER regex(?inseeCode, "^34") .
      } limit 400 ' AS sparql CALL semantics.importRDF(
  "https://query.wikidata.org/sparql?query=" +
    apoc.text.urlencode(sparql),"JSON-LD",
   { headerParams: { Accept: "application/ld+json"} })
YIELD terminationStatus, triplesLoaded,namespaces,extraInfo
RETURN terminationStatus, triplesLoaded,namespaces,extraInfo
```

Listing 16 – Requête CYPHER/SPARQL

Une fois cet import terminé, vous répondrez à une série de questions.

### 3.1 Questions d'appropriation

1. Expliquer en langage naturel ce que renvoie la requête SPARQL construite. Faîtes un graphe rapide sur 2 cités du résultat de l'import dans Neo4J.
2. Les nœuds importés correspondent à des villes et villages de l'Hérault. Vous ferez en sorte de les lier au nœud correspondant au département de l'Hérault via la relation WITHIN

```
MATCH (ci:sch__City)
MATCH (n:Departement {id:'34'})
CREATE (ci) -[:WITHIN]-> (n)
```

Listing 17 – Correction

3. Vous supprimerez les nœuds de type commune du graphe

```
MATCH (co:Commune) DETACH DELETE co
```

Listing 18 – Correction

4. Renvoyez le nombre de communes limitrophes de la commune de Montpellier

```
MATCH (m:sch__City {sch__name:'Montpellier'})-[:sch__geoTouches]-(x)
RETURN count(x) as limitrophes
```

Listing 19 – Correction

### 3.2 Questions sur les chemins

1. renvoyez un des plus courts chemins entre Montpellier et Beaulieu

```
MATCH p=shortestpath((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) ) RETURN p
```

Listing 20 – Correction

2. renvoyez tous les plus courts chemins entre Montpellier et Beaulieu

```
MATCH p=allshortestpaths((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) ) RETURN p
```

Listing 21 – Correction

3. renvoyez le nom des cités traversées par un des plus courts chemins entre Montpellier et Beaulieu

```
MATCH p=shortestpath((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) )
RETURN extract(n in nodes(p)|n.sch__name) as noeudsDuChemin
```

Listing 22 – Correction

4. renvoyez le nom et l'adresse des cités traversées par un des plus courts chemins entre Montpellier et Beaulieu

```
match p=shortestpath((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) )
RETURN extract(n IN nodes(p) | {name: n.sch__name, codeInsee: n.sch__address})

ou alors

match p=shortestpath((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) )
UNWIND nodes(p) as n
RETURN n.sch__name, n.sch__address
```

Listing 23 – Correction

5. renvoyez le nombre des plus courts chemins entre Montpellier et Beaulieu



```
MATCH p=allshortestpaths((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) )
RETURN count(p)
```

Listing 24 – Correction

6. renvoyez tous les chemins entre Montpellier et Beaulieu (très coûteuse). Que faire pour réduire la complexité ?

```
MATCH p=((m:sch__City {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) )
RETURN p

creer un index
create index on :sch__City(sch__name)

ou alors ne pas renvoyer tous les chemins
ensuite MATCH p=((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}) ) RETURN p limit 5
```

Listing 25 – Correction

7. retourner un des plus courts chemins qui ne passe pas par Clapiers ?

```
match p=shortestpath((m:sch__City
    {sch__name:'Montpellier'})-[:sch__geoTouches*]-(g:sch__City
    {sch__name:'Beaulieu'}))
where not ('Clapiers' in (extract (n in nodes(p)| n.sch__name))) return p
```

Listing 26 – Correction