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 ne sont comptabilisées qu'au niveau du département, et non de la commune, 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.

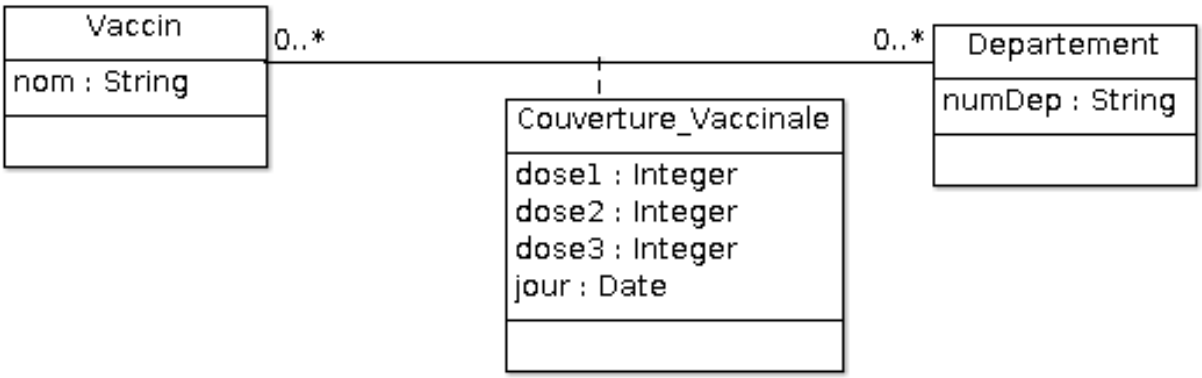

FIGURE 1 – Diagramme de classes pour l'application

Deux exemples distincts de documents sont fournis, qui illustrent les deux structures de document définies,

```
{
"_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.

```
la premiere structuration introduit volontairement une erreur de conception,
   avec l'element "vaccin" qui ne devait pas etre dans le tableau mais un champ
   elementaire du document au meme titre que departement ou jour.
   Syntaxiquement c'est ok car un tableau peut contenir divers elements, mais
   semantiquement cela gene dans la consultation. Un autre element a probleme
   est que pour un jour et un departement donnes, 5 documents sont definis : un
   par vaccin et un pour tous les vaccins. L'information n'est pas concise.
   L'id de chaque document est construit a partir du jour, departement et
   vaccin.
la seconde structuration est plus concise (un seul document par jour et par
   departement quel que soit le vaccin), l'organisation est "meilleure", et
   l'id est defini par le systeme (uuid). La difficulte est que que l'on se
   trouve face a des tableaux imbriques qui rendent difficiles la consultation.
   Un intermediaire entre les deux propositions avec les noms des vaccins en
   guise de champ aurait ete possible. La seconde structuration est plus
   appropriee a des requetes de comparaison entre les doses injectees pour
   chaque vaccin.
```

Listing 4 – Manipulation date

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 5 – Exemple map/reduce

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

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

Listing 6 – 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 7 – Correction

2. lister tous les documents de la BD

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

Listing 8 – 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 9 – 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 10 – 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 11 – 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 12 – 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 13 – Correction

4. donnez le nombre de documents de type couverture_vaccinale par département, par année et par mois écoulés.

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

Listing 14 – 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 15 – 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 16 – 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 17 – Correction

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

```
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 18 – 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 19 – 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 20 – 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 21 – Correction