le cnam

Intéraction avec une base MongoDB

Travaux Pratiques

CNAM Paris
nicolas.travers (at) cnam.fr

Chapitre

Table des matières

CHAPITRE 1

INTERROGATION SIMPLE

Pour l'installation de l'environnement, des instructions de connexion et de requêtes, veuillez vous référer au guide disponible sur `http://www.chewbii.com/tp-mongodb`.

Une fois MongoDB installé, connectez vous à la base "`dblp`" (cf Guide).

Pour vous aider dans ces travaux pratiques, veuillez trouver ci-dessous deux exemples de documents JSon différents, illustrant le contenu de la base DBLP. Vous pourrez noter des clés pouvant être présentes ou non dans certains cas (clé "booktitle", "cites"...), voire des données de type différents (clé "pages").

```
{
        "_id" : "books/daglib/0025185",
        "type" : "Book",
        "title" : "MongoDB - The Definitive Guide: Powerful and Scalable Data Storage.",
        "pages" : "I",
        "publisher" : "O'Reilly",
        "year" : 2010,
        "authors" : [ "Kristina Chodorow", "Michael Dirolf" ],
        "isbn" : [ "978-1-449-38156-1" ]
}
```

```
{
        "_id" : "conf/icod/BouzeghoubG83",
        "type" : "Article",
        "title" : "The Design of an Expert System for Database Design.",
        "booktitle" : "ICOD-2 Workshop on New Applications of Data Bases",
        "pages" : { "start" : 203, "end" : 223 },
        "year" : 1983,
        "url" : "db/conf/icod/icod83w.html#BouzeghoubG83",
        "authors" : [ "Mokrane Bouzeghoub", "Georges Gardarin" ],
        "cites" : ["journals/tods/AstrahanBCEGGKLMMPTWW76", "books/bc/AtzeniA93",
                "journals/tcs/AtzeniABM82", "journals/jcss/AbiteboulB86"]
}
```

Cette base contient une extraction des **publications** de recherche en informatique fournie par DBLP.org. Vous y trouverez donc :

- Des articles de conférences/journaux scientifique, de, "type" "Article", dont le titre est "title" mais l'ouvrage de référence (s'il existe) "booktitle" ;
- Des chapitres de livre, de, "type" "Book" ayant également title et booktitle ;
- Des thèses, du, "type" "Phd".
- Chaque publication est associé à une liste d'auteurs (authors) et éventuellement un éditeur (publisher).
- Certaines publications peuvent contenir des références internes avec la clé "cites"

1.1 API Mongo - Requêtes *find*

Créer des requêtes simples pour les phrases suivantes :

1.1.1 Liste de toutes les publications dont le, "type" est un livre ('Book') ;

1.1.2 Liste des publications publiées depuis 2011 ;

1.1.3 Liste des livres depuis 2014 ;

1.1.4 Liste des publications ayant un éditeur (publisher) ;

1.1.5 Liste des publications dont l'auteur est "Jeffrey D. Ullman" ;

1.1.6 Titre des publications dont "Jeffrey D. Ullman" le premier auteur ;

1.1.7 Nom des éditeurs (publisher) et titre des publications dont "Jeffrey D. Ullman" est le seul auteur ;

1.1.8 Titre des publications dont le titre contient le mot "database" ;

1.2 Distincts

1.2.1 Liste distincte de tous les editeurs (“publisher”) ;

1.2.2 Liste de tous les auteurs distincts ;

1.3 API Mongo - Requêtes *aggregate*

1.3.1 Pour les publications contenant le mot "database" et dont le titre de l’ouvrage (booktitle) et la page de départ existent ; trier par titre d’ouvrage et pages de départ ;

1.3.2 Projeter le résultat précédent par titres, titres d’ouvrage et pages ;

1.3.3 Compter, pour le résultat précédent, le nombre de publications retournées ;

1.3.4 Pour le résultat précédent, compter par, "type" de publication ;

1.3.5 Compter le nombre de publications par auteur. Trier le résultat par ordre décroissant ;

1.3.6 Pour chaque éditeur (s’il existe), donner l’année moyenne et le nombre de publications. Trier le résultat par le nombre de publications décroissant ;

1.3.7 Compter par année et éditeur (s’il existe), le nombre de publications. Pour ces derniers, ne retourner que ceux ayant plus 200 publications ;

1.3.8 Pour chaque éditeur, calculer la moyenne du nombre de publications par année. Trier le résultat par moyenne décroissante ;

Chapitre 2

Map/Reduce

Dans ce chapitre, nous allons étudier le comportement d'une requête Map/Reduce sous MongoDB. Pour ce faire, nous allons commencer par s'intéresser au Map, puis à la partie Reduce.

2.1 Map

Cette section s'intéresse aux variantes sur la partie `Map` de la requête.

2.1.1 Exécuter la requête du Guide ;

2.1.2 Pour chaque, "type" de publications (clé "type"), donner le nombre de documents associé ;

2.1.3 Pour chacun des documents de "type" livres (type :"Book"), donner le nombre de ses auteurs (this.authors.length) par titre de livre ("booktitle"), lorsqu'il y en a un ;

2.1.4 Produire le même résultat que la requête précédente, mais en utilisant une boucle '*for*' sur la liste d'auteurs et faire un `emit` par auteur. Pourquoi y a-t-il moins de documents en sortie ?

2.1.5 Compter les publications de plus de 3 auteurs ;

2.1.6 Compter le nombre du publications par nombre d'auteurs ;

2.1.7 Pour chaque publication ayant un "booktitle" (chapitre de livre) publié par Springer ("publisher"), donner le nombre de ses chapitres. N'afficher que ceux qui ont plus d'un chapitre.
Attention, la fonction reduce n'est évaluée que lorsqu'il y a au moins 2 documents pour une même clé. Il est donc nécessaire d'appliquer un filtre après génération du résultat ;

2.1.8 Pour l'éditeur "Springer" ("publisher"), donner le nombre de publications par année ;

2.1.9 Pour chaque clé "publisher & année" (pour ceux qui ont un publisher), donner le nombre de publications.
Attention, la clé du emit doit être un document ;

2.1.10 Pour l'auteur "Jeffrey D. Ullman", donner le nombre de publications par année ;

2.1.11 Pour chaque auteur et année, donner le nombre de publications ;

2.1.12 Donner pour chaque couple d'auteurs (ordre alphabétique), le nombre de fois qu'ils ont publié ensemble. Trier le résultat par ordre décroissant ;

2.2 Reduce

Dans cette section, nous allons modifier la fonction `reduce`.

2.2.1 Pour l'auteur "Jeffrey D. Ullman", donner le nombre moyen de pages pour ses articles (type Article) ;

2.2.2 Pour chaque auteur, donner la liste des titres de ses publications.
Attention, la sortie de map et du reduce doivent être des documents (pas des tableaux) ;

2.2.3 Pour chaque éditeur, donner le nombre moyen de pages par publication ;

2.2.4 Pour chaque auteur, donner le minimum et le maximum des années, ainsi que le nombre total de publications ;

2.2.5 Pour l'éditeur "Springer", donner le nombre d'auteurs distincts par année

CHAPITRE 3

MISES À JOUR

3.1 Insertion

3.1.1 Créer un document (définition sur une ligne) :

```
w={"_id" : "MDS", "type": "Book",
   "title": "Modern Database Systems: The Object Model, Interoperability, and Beyond.",
   "year": 1995, "publisher": "ACM Press and Addison-Wesley", "authors": ["Won Kim"],
   "source": "DBLP"};
```

3.1.2 Insérer le document dans la collection “publis” :

```
db.publis.save(w);
```

3.1.3 Consulter le résultat :

```
db.publis.find({"_id" : "MDS"});
```

‘

3.1.4 Créer et insérer deux autres publications à partir de cette page de conférence (type de publication) : `http://www.informatik.uni-trier.de/~ley/db/journals/vldb/vldb23.html`.

3.2 Update/Delete

Une fois la base intégralement téléchargée, vous pourrez faire des mises à jour, avec les instructions suivantes :

—

```
db.publis.update ({"_id" : "MDS"}, {$set : { "Genre" : "Science" } } );
//Premier JSon : Mapping, Second JSon : Update ($set, $unset)
```

—

```
db.publis.find({"Genre":{$exists:1}});
//Vérifier quels documents ont été modifiés
```

—

```
db.publis.remove({"_id" : "MDS"})
//Mapping de suppression
```

—

```
db.publis.find({"Genre":{$exists:1}});
//Vérifier si les documents existent encore
```

— Il est possible de faire des fonctions itératives (javascript) sur le résultat du `find` :

```
db.publis.find(
        {"type":"Phd", "pages.start":{$exists:1}, "pages.end":{$exists:1}}
).forEach(
  function(pub){
    pub.pp = pub.pages.end - pub.pages.start + 1;
    db.publis.save(pub);
  }
);
```

— Tester le résultat :

```
  db.publis.find({"type":"Phd"}, {"title":1, "pp":1});
```

3.3 Exercices

Pour les mises à jour suivantes, vérifier le contenu des données avant et après la mise à jour.

3.3.1 Mettre à jour tous les livres contenant “database” en ajoutant l’attribut “Genre” : “Database”

3.3.2 Supprimer le champ “number” de tous articles

3.3.3 Supprimer tous les articles n’ayant aucun auteur

3.3.4 Modifier toutes les publications ayant des pages pour ajouter le champ “pp” avec pour valeur le motif suivant : “start–end”

CHAPITRE 4

INDEXATION

Il est possible d'indexer des clés sous MongoDB. Nous allons voir comment les créer, et comment constater les modifications sur les plans d'exécution (explain() à partir de la version 2.6).

4.1 Indexation simple (type scalaire)

4.1.1 Pour la requête ci-dessous, regarder le plan d'exécution généré avec " `.explain()` " à la fin de la requête :

```
db.publis.find({"type" : "Book", year : {$gte : 2014}}).explain();
```

4.1.2 On remarquera qu'une opération de, "type" `COLSCAN` est appliqué (WinningPlan). Cela correspond à un parcours intégral de la collection.

4.1.3 Créer un index sur l'attribut année " `db.publis.createIndex({year:1});` ";

4.1.4 Exécuter à nouveau la requête en regardant le plan d'exécution généré ;

4.1.5 Nous pourrons alors constater qu'une opération `IXSCAN` est appliquée sur la clé "year". Le nouvel index est utilisé. Créer un index sur la clé "type". Vous pourrez alors constater que le plan d'exécution est identique, toutefois, un plan d'exécution a été rejeté "rejectedPlans" (celui sur le, "type")

4.1.6 Il est également possible de consulter le plan d'exécution pour les agrégats avec l'option (2° paramètre de "aggregate") {explain :true} :

```
db.publis.aggregate([{$match:{year:2012}}, {$group:{_id:"$publisher", total : { $sum : 1}}}], {explain:true});
```

4.1.7 Pour MapReduce, il n'est pas possible de consulter le plan d'exécution. Toutefois, on peut constater l'utilisation de l'index via le nombre d'éléments en 'input' :

```
var mapFunction = function () {
        if(this.year > 2010)
                emit({"publisher":this.publisher, year:this.year}, 1);
};
var reduceFunction = function (key, values) {
        return Array.sum(values);};
var queryParam = {"query":{}, "out":"result_set", explain:true};
db.publis.mapReduce(mapFunction, reduceFunction, queryParam);
```

4.1.8 Vous pourrez constater que l'ensemble des documents sont interrogés (input :118015). En effet, le prédicat "year" est présent dans le map. L'optimiseur de MongoDB ne peut analyser la fonction map (appliquée à chaque document), il faut pour cela utiliser le paramètre "query" du queryParam pour que l'index soit pris en compte :

```
var mapFunction = function () {
        emit({"publisher":this.publisher, year:this.year}, 1);
};
var reduceFunction = function (key, values) {
        return Array.sum(values);};
var queryParam = {"query":{year : {$gt : 2010}}, "out":"result_set", options:{$explain:1}};
db.publis.mapReduce(mapFunction, reduceFunction, queryParam);
```

4.2 Indexation 2D avec 2DSphere

Nous pouvons indexer les données avec 2DSphere qui permet de faire des recherches en 2 dimensions. Le schéma des coordonnées doit être structuré de la manière suivante [1] :

1. Documentation : `http://docs.mongodb.org/manual/applications/geospatial-indexes/`

```
"location" : {
      "type" : "Point",
      "coordinates" : [
          1.53414,
          42.50729
      ]
  }
```

La clé de localisation dans ce document est "location".

Pour faire le TP sur des données géospatiales, télécharger le fichier http://chewbii.com/cities.json/.
Décompressez le.
Importez les données dans une collection “cities”.

L’attribut location sera alors indexé :

```
db.cities.ensureIndex( { location : "2dsphere" } );
```

Pour interroger l’index, il faut utiliser un opérateur 2D et l’utiliser sur “location”
Documentation : http://docs.mongodb.org/manual/tutorial/query-a-2d-index/

4.2.1 Tout d’abord récupérer les coordonnées de la ville de Paris, Lyon et Bordeaux. Affecter des variables Paris, Lyon et Bordeaux à ces coordonnées pour les réutiliser par ailleurs ;

4.2.2 Afficher les noms des villes autour de Paris dans un rayon de 100km. Pour cela, utiliser la syntaxe suivant :
{"location":{$near:{$geometry:{"type":"Point","coordinates":[LAT,LONG]},$maxDistance:XXX}}};

4.2.3 Calculer la somme des populations de cette zone. Dans un aggregate il faut utiliser l’opérateur **$geoNear** :
{"$geoNear": {"near":{"type":"Point", "coordinates" : [LAT, LONG]},"maxDistance":XXX,
"distanceField" : "outputDistance", "spherical":true}};

4.2.4 Afficher le nom des villes comprises dans le triangle Paris-Lyon-Bordeaux avec l’opérateur **$geoWithin** :
{location:{$geoWithin:{$geometry:{"type":"Polygon","coordinates":POLYGON}}}}
Le polygon étant une liste de liste de points.

4.2.5 Trouver les villes de plus de 100 000 habitants dans cette zone et afficher leur nom ;

4.2.6 Faire la somme des populations de cette zone ;