Chapitre 4

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.

4.1 Map

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

4.1.1 Exécuter la requête suivante :

```
mapFunction = function () {
        emit(this.category, 1);};
reduceFunction = function (key, values) {
        return Array.sum(values);};
queryParam = {"query":{}, "out":{"inline":true}};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
```

Correction : La sortie est affichée directement avec à la fois le résultat et les statistiques d'exécution.

```
{
        "results" : [
        {
                "_id" : "accommodation",
                "value" : 1698
        },
        {
                "_id" : "attraction",
                "value" : 1758
        },
        {
                "_id" : "poi",
                "value" : 2800
        },
        {
                "_id" : "restaurant",
                "value" : 9183
        }
        ],
        "timeMillis" : 164,
        "counts" : {
                "input" : 15439,
                "emit" : 15439,
                "reduce" : 171,
                "output" : 4
        },
        "ok" : 1
}
```

On peut constater que 15439 documents sont traités et produisent dans le map chacun une sortie. Produisant alors 4 clés différentes de regroupement (output : 4). Le nombre de reduce représente le nombre d'opérations "reduce" effectuées en fonction de l'allocation des ressources locales et globales.

Il est possible d'enregistrer le résultat de la requête dans une nouvelle collection :
queryParam = "query" :, "out" :"result_set" ;

4.1.2 Pour chaque catégorie, donner le nombre de commentaires en français (utiliser la taille d'une liste en javascript "reviews.language") ;

Correction :

```
mapFunction = function () {
        if(this.reviews.language == "fr")
                emit(this.category, this.review.fr);
};
```

4.1.3 Pour chaque catégorie, donner le nombre de mots pour les commentaires en français ;
Correction :

```
mapFunction = function () {
        for(var i=0 ; i< this.reviews.length ; i++){
                if (this.reviews[i].language == "fr")
                        emit (this.category, this.reviews[i].wordsCount);
        }
};
```

4.1.4 Pour chaque "type" de contact (s'il existe), donner le nombre de lieux associés ;
Correction : Une version un peu brutale vérifie chacunes des clés disponibles. Mais ce n'est pas très évolutif.

```
mapFunction = function () {
        if(!this.contact)
                return;
        if(this.contact.website && this.contact.website != "")
                emit("website", 1);
        if(this.contact.GooglePlaces && this.contact.GooglePlaces != "")
                emit("GooglePlaces", 1);
        if(this.contact.phone && this.contact.phone != "")
                emit("phone", 1);
        if(this.contact.Foursquare && this.contact.Foursquare != "")
                emit("Foursquare", 1);
        if(this.contact.Booking && this.contact.Booking != "")
                emit("Booking", 1);
        if(this.contact.Facebook && this.contact.Facebook != "")
                emit("Facebook", 1);
};
```

Vous pourrez tester en cherchant l'ensemble des clés contenues dans le sous-document "contact".

```
Object.keys(this.contact)
```

4.1.5 Donner le nombre de lieux pour chaque "nombre moyen de mots" par lieu (moyenne de reviews.wordsCount), pour les messages Facebook. Arrondir la moyenne à l'inférieur (Math.floor()) ;
Correction :

```
mapFunction = function () {
        sum = 0;
        nb = 0;
        for(var i=0 ; i<this.reviews.length ; i++){
                review = this.reviews[i];
                if(review.source == "Facebook"){
                        sum += review.wordsCount;
                        nb++;
                }
        }
        if(nb > 0 && sum > 0)
                emit (Math.floor(sum/nb), 1);
};
```

Il est également possible de rajouter un filtre dans le `queryParam` pour ne chercher que les documents qui ont au moins un commentaire "Facebook".

4.1.6 Compter le nombre de lieux ayant plus de 3 commentaires en anglais ;
Correction :

```
mapFunction = function () {
        nb = 0;
        for(var i=0 ; i< this.reviews.length ; i++){
                if (this.reviews[i].language == "en")
                        nb++;
                if(nb > 3){
                        emit (null, 1);
                        return;
                }
        }
}
```

4.1.7 Compter par langue, le nombre de lieux ayant plus de 3 commentaires de cette langue ;
Correction :

```
mapFunction = function () {
        nb = {};
        for(i=0 ; i< this.reviews.length ; i++){
                lang = this.reviews[i].language;
                if(nb[lang])
                        nb[lang]++;
                else
                        nb[lang] = 1;
        }
        for(key in nb){
                if(nb[key] > 3)
                        emit (key, 1);
        }
}
```

4.1.8 Compter le nombre du lieux par nombre de services ;
Correction :

```
mapFunction = function () {
        if(this.services)
                emit(this.services.length, 1);};
```

4.1.9 Pour chaque nom de lieu, donner ceux qui sont présents au moins deux fois.
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 (stocker le résultat pour faire une requête dessus). Trier le résultat par ordre décroissant du nombre d'occurences ;
Correction :

```
mapFunction = function () {
                emit(this.name, 1);};
queryParam = {"query":{}, "out":"result"};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
db.result.find({"value":{$gte:2}}).sort({"value":-1});
```

4.1.10 Pour la catégorie "accommodation", donner le nombre de commentaires par année (utiliser 'substring' sur la date) ;
Correction :

```
mapFunction = function () {
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(review.time){
                        annee = review.time.substring(0,4);
                        emit(annee, 1);
                } else
                        emit(null, 1);
        }
};
queryParam = {"query":{"category":"accommodation"}, "out":{"inline":1}};
```

4.1.11 Pour les commentaires Foursquare, donner le nombre de commentaires par année ;
Correction :

```
mapFunction = function () {
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if (review.source == "Foursquare"){
                        if(review.time){
                                annee = review.time.substring(0,4);
                        else
                                annee = null;
                        emit(annee, 1);
                }
        }
};
queryParam = {"query":{}, "out":{"inline":1}};
```

4.1.12 Pour chaque clé "source & année", donner le nombre de publications.
Attention, la clé du emit doit être un document ;
Correction :

```
mapFunction = function () {
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(review.time)
                        annee = review.time.substring(0,4);
                else
                        annee = null;
                emit({"src":review.source, "y" : annee}, 1);
        }
};
```

4.1.13 Pour chaque couple de service, donner le nombre de lieux ;
Correction :

```
mapFunction = function () {
        if(this.services)
                for(var i=0;i<this.services.length;i++){
                        s1 = this.services[i];
                        for(var j=0;j<this.services.length;j++){
                                s2 = this.services[j];
                                if(i < j)
                                        emit({"s1":s1, "s2":s2},1);
                        }
                }
};
queryParam = {"query":{}, "out":"result"};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
db.result.find().sort({"value":-1});
```