# Chapitre 1

# Interrogation simple

## 1.4 API Mongo - Requêtes *aggregate*

L'opérateur `aggregate` est une séquence d'opérations représentant une chaîne de pipeline entre chaque opérateur (le résultat d'un opérateur est donné à l'opérateur suivant). Il s'exprime de la sorte :
`aggregate( [ {$op1}, {$op2}, {$op3} ] )`.

*1.4.1* Donner les adresses des lieux de catégorie "accommodation" avec un service "*blanchisserie*", projeter le résultat sur le nom et numéro de téléphone (seulement si elle existe), et trier sur le nom ;
**Correction :**

```
opMatch = {$match : {"services":"blanchisserie", "category":"accommodation",
                "contact.phone" : {$exists:1}} };
opProject = {$project : {"name":1, "contact.phone":1}};
opSort = {$sort : {"name":1}};
db.paris.aggregate([ opMatch, opProject, opSort ]);
```

*1.4.2* Nombre de lieux de catégorie "accommodation" et ayant un service "*chambres non-fumeurs*" ;
**Correction :**

```
opMatch = {$match :  {"category" : "accommodation", "services":"chambres non-fumeurs"} };
opGroup = {$group : {_id:null, "tot":{$sum:1}} };
db.paris.aggregate([ opMatch, opGroup ]);
```

*1.4.3* Donner le nombre de lieux par catégorie ;
**Correction :**

```
opGroup = {$group : {_id:"$category", "tot":{$sum:1}} };
db.paris.aggregate([ opGroup ]);
```

*1.4.4* Pour les lieux de catégorie "accommodation", donner le nombre de lieux pour chaque service ;
**Correction :**

```
opMatch = {$match :  {"category" : "accommodation"} };
opUnwind = {$unwind : "$services"};
opGroup = {$group : {_id:"$services", "tot":{$sum:1}} };
db.paris.aggregate([ opMatch, opUnwind, opGroup ]);
```

Pour éviter de grouper par liste de services, il faut décomposer ces listes (unwind), puis faire l'agrégation.

*1.4.5* Trier le résultat précédent par ordre décroissant ;
**Correction :**

```
opSort = {$sort : {tot:-1}};
db.paris.aggregate([ opMatch, opUnwind, opGroup, opSort ]);
```

*1.4.6* Du résultat précédent, n'afficher que les services présents dans plus de 1000 lieux ;
**Correction :**

```
opMatch2 = {$match : {"tot":{$gt : 1000}}};
db.paris.aggregate([ opMatch, opUnwind, opGroup, opSort, opMatch2 ]);
```

*1.4.7* Pour chaque nom de lieu de catégorie "poi", donner le nombre de commentaires dont la source (reviews.source) est "Facebook". Trier par ordre décroissant ;
**Correction :**

```
opMatch = {$match : {"category" : "poi"}};
opUnwind = {$unwind : "$reviews"};
opMatch2 = {$match : {"reviews.source" : "Facebook"}};
opGroup = {$group :        {_id:"$name", "tot":{$sum:1}} };
opSort = {$sort : {tot:-1}};
db.paris.aggregate([ opMatch, opUnwind, opMatch2, opGroup, opSort ]);
```

*1.4.8* Pour chaque langue d'un commentaire (reviews.language), donner le nombre de commentaires de lieux ayant un service "*chambres non-fumeurs*" ;
**Correction :**

```
opMatch = {$match : {"services" : "chambres non-fumeurs"}};
opUnwind = {$unwind : "$reviews"};
opGroup = {$group : {_id:"$reviews.language", "tot":{$sum:1}}};
opSort = {$sort : {tot:-1}};
db.paris.aggregate([ opMatch, opUnwind, opGroup, opSort]);
```

*1.4.9* Pour chaque nom de lieu de catégorie "restaurant", donner la note moyenne et le nombre de commentaires. Trier le résultat par ordre décroissant de moyenne, puis de nombre ;
**Correction :**

```
opMatch = {$match : {"category" : "restaurant"}};
opUnwind = {$unwind : "$reviews"};
opMatch2 = {$match : {"reviews.rating" : {$nin : ["", 0]}}};
opGroup = {$group : {_id:"$name","moy" : {$avg:"$reviews.rating"}, "tot":{$sum:1}} };
opSort = {$sort : {moy:-1, tot:1}};
db.paris.aggregate([ opMatch, opUnwind, opMatch2, opGroup, opSort]);
```

On pourrait également calculer un score particulier dans le projet pour trier de manière intelligente le résultat : $score = moy \times \log_{10}(tot)$

```
opProject = {$project : {"moy":1, "tot":1, "score":{$multiply:["$moy",{$log:["$tot", 10]}]}} };
opSort = {$sort : {score:-1, moy:-1}};
db.paris.aggregate([ opMatch, opUnwind, opMatch2, opGroup, opProject, opSort]);
```

*1.4.10* Pour chaque catégorie de lieux et langue de commentaire, donner le nombre de commentaires correspondants ;
**Correction :** La clé de groupement n'est pas forcément un type scalaire, il peut être un document JSon (catégorie + langue)

```
opUnwind = {$unwind : "$reviews"};
keyGroup = {"cat" : "$category", "lang":"$reviews.language"};
opGroup = {$group : {_id: keyGroup, "tot":{$sum:1}}}
opSort = {$sort : {tot:-1}};
db.paris.aggregate([ opUnwind, opGroup, opSort ]);
```

*1.4.11* Pour chaque catégorie de lieux, donner le nombre moyen de commentaires par langue (réutiliser le résultat précédent) ;
**Correction :** La clé du nouveau groupement est dans la clé du premier groupe (_id.cat)

```
opUnwind = {$unwind : "$reviews"};
keyGroup = {"cat" : "$category", "lang":"$reviews.language"};
opGroup = {$group : {_id: keyGroup, "tot":{$sum:1}}}
opGroup2 = {$group : {_id: "$_id.cat", "avg":{$avg:"$tot"}}}
db.paris.aggregate([ opUnwind, opGroup, opGroup2 ]);
```

*1.4.12* Donner le nombre de moyen de commentaires par lieu ;

Correction :

```
opUnwind = {$unwind : "$reviews"};
opGroup = {$group : {_id: "$_id", "tot":{$sum:1}}}
opGroup2 = {$group : {_id: null, "avg":{$avg:"$tot"}}}
db.paris.aggregate([ opProject, opUnwind, opGroup, opGroup2 ]);
```