



FONCTIONS D'AGRÉGATION

Plan module

Introduction

Base de données orienté document : MongoDB

- Modélisation
- CRUD
- Agrégation
- Indexation
- Sharding et Replica Set

Base de données orienté colonne : Cassandra

Base de données orienté graphe : Neo4j

match ... Sort ... Limit ... Skip

- \$match : Afficher les lignes qui correspondent a un critère bien déterminé

```
db.livres.aggregate([ { $match: { year: 2010 } }, { $project: { _id: 0, title: 1 } } ] )
```

- \$sort : ordonner une liste d'éléments

```
db.livres.aggregate([ { $sort: { "year": -1 } }, { $project: { _id: 0, title: 1 } } ] )
```

- \$limit : afficher un nombre bien déterminé de documents

- \$skip : sauter un nombre de documents

```
db.livres.aggregate([ { $sort: { "year": -1 } },  
                      { $project: { _id: 0, title: 1 } },  
                      { $skip: 1 },  
                      { $limit: 2 } ] )
```

\$Project : Chaîne de caractères

Fonctions	Exemples
\$toLowerCase/ \$toUpperCase	<code>db.livres.aggregate([{ \$project: { _id: 0, auteur: { \$toLowerCase: "\$author" }, title: 1 } }])</code>
\$concat	<code>db.livres.aggregate([{ \$project: { author: 1, title_description: { \$concat: ["\$title", "-", "\$description"] } } }])</code>
\$split	<code>db.livres.aggregate([{ \$project: { _id: 0, description: { \$split: ["\$description", " "] } } }])</code>
\$substr	<code>db.livres.aggregate([{ \$project: { _id: 0, description: { \$substr: ["\$description", 0, 4] } } }])</code>
\$strLenBytes	<code>db.livres.aggregate([{ \$project: { _id: 0, title: 1, length: { \$strLenBytes: "\$title" } } }])</code>

\$Project : Fonctions numériques

Fonctions	Exemples
\$add	<code>db.livres.aggregate([{ \$project: { _id: 0, likes: { \$add: ["\$likes", 3] } } }])</code>
\$multiply	
\$ceil	<code>db.livres.aggregate([{ \$project: { _id: 0, likes: { \$ceil: "\$likes" } } }])</code>
\$floor	
\$trunc	<code>db.livres.aggregate([{ \$project: { _id: 0, likes: { \$trunc: "\$likes" } } }])</code>
\$mod	

\$Project : Tableau

Fonctions	Exemples
\$arrayElemAt	<pre>db.livres.aggregate([{ \$project: { _id: 0, first: { \$arrayElemAt: ["\$tags", 0] } } }])</pre>
\$reverseArray	<pre>db.livres.aggregate([{ \$project: { _id: 0, first: { \$reverseArray: ["\$tags"] } } }])</pre>
\$size	<pre>db.livres.aggregate([{ \$project: { _id: 0, size: { \$size: ["\$tags"] } } }])</pre>

\$Project : Fonctions date

Fonctions	Exemples
\$dayOfYear	
\$year	<pre>db.data.aggregate([{ \$project: { _id: 0, year: { \$year: "\$date_pub" } } }])</pre>
\$month	<pre>db.data.aggregate([{ \$project: { _id: 0, year: { \$month: "\$date_pub" } } }])</pre>
\$isoDayOfWeek	<pre>db.data.aggregate([{ \$project: { _id: 0, year: { \$isoDayOfWeek: "\$date_pub" } } }])</pre>
\$isoWeek	
\$dateToString	

Fonctions d'agrégation

- Afficher le nombre de livre par **auteur** :

```
db.livres.aggregate([{$group:{_id:"$author",  
                               nbr_livre:{$sum:1}}}]])
```

```
Select count(*) nbr_livre FROM livres GROUP BY author
```

- Afficher le nombre de livre par **nom_auteur** par **annee**

```
db.livres.aggregate([{$group:{_id:{nom_auteur :"$author",  
                                   annee :"$year"}}, nbr_livre:{$sum:1}}}]])
```

```
SELECT count(*) nbr_livre FROM livres GROUP BY nom_auteur, annee
```


Fonctions d'agrégation

```
db.livres.aggregate([{$group:{_id:"$author",sum_likes:{$sum:"$likes"}}}])
```

Select sum(likes) sum_likes from livres group by author

```
db.livres.aggregate([{$group:{_id:{}},avg_like:{$avg:"$likes"}}]})
```

Select avg(likes) avg_like from livres

```
db.livres.aggregate([{$group:{_id:{}},max_like:{$max:"$likes"}}]})
```

Select max(likes) max_like from livres

```
db.livres.aggregate([{$group:{_id:"$year",min_like:{$min:"$likes"}}}])
```

Select min(likes) min_like from livres group by year

Fonctions d'agrégation : \$first et \$last

\$first et \$last sont des fonctions de groupe qui permettent d'avoir la première et la dernière valeur de chaque groupe.

Utilisées généralement avec la fonction \$sort.

```
db.livres.aggregate([ { $group: { _id: "$year", auteur: { $first: "$author" } } } ] )
```

Construction d'un tableau

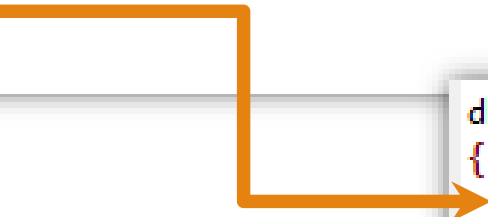
- `$addToSet`: Afficher la liste des livres par auteur. Pas de duplication des valeurs.

```
db.livres.aggregate([{$group:{_id:"$author", livres:{$addToSet:"$title"}}}])
```

- `$push`: similaire à `$addToSet`, mais duplication des valeurs.

Foreign Key : \$lookup

```
db.conference.insert({  
  title: "Plein de mots-clefs hyper cools...",  
  description: "Refaites votre SI avec tout ça, sinon vous êtes mort!",  
  speaker_id: "techno_maniac"  
})
```



```
db.speaker.insert(  
  {  
    _id: "techno_maniac",  
    name: "Jean-Marcel Le Maniac",  
    email: "le_maniac@web2day.org"  
  })
```

```
db.conference.aggregate( [ {"$lookup": {from: "speaker",localField:  
"speaker_id",foreignField: "_id",as: "speaker_infos"} } ] )
```

\$unwind ... \$out

- \$unwind : séparer les éléments d'un tableau

```
db.livres.aggregate([{$unwind: "$tags"}, {$group: {_id: "$_id", nbre_tags: {$sum: 1}}}] );
```

- \$out : Récupérer le résultat d'agrégation dans une autre collection

```
db.livres.aggregate([{$unwind: "$tags"}, {$group: {_id: "$_id", nbre_tags: {$sum: 1}}}], {$out: "results"})
```

```
db.results.find()
```

Options d'agrégation

- Analyse de l'agrégation :

```
db.livres.aggregate([{$group:{_id:"$author",nbr_livre:{$sum:1}}}],{explain:true})
```

- Gestion de l'agrégation sur le disque et non dans l'espace mémoire (cas où le résultat dépasse 100M)

```
db.livres.aggregate([{$group:{_id:"$author",nbr_livre:{$sum:1}}}],{allowDiskUse:true})
```

Limites de l'Agrégation

Résultat limité à 16 MB d'espace mémoire

Peut utiliser au maximum 10% de l'espace mémoire d'une machine; utiliser `$match` pour filtrer les résultats, et utiliser `$project` pour choisir quelques champs.

sharding, c'est la répartition des collections entre plusieurs instances de Mongo, avec un routeur mongos, l'agrégation fonctionne sur un environnement fragmenté, mais après le premier stade `$group` ou `$sort`, l'agrégation doit être ramené aux mongos, afin qu'il puisse recueillir les résultats avant de les envoyer pour la prochaine étape du pipeline (qui a besoin de voir le résultat final)

mapreduce est une alternative de l'agrégation

Hadoop est aussi une alternative de l'agrégation; il y a un connecteur entre Hadoop et MongoDB.