Cours sur l'Agrégation en MongoDB

Introduction L'agrégation en MongoDB est une méthode puissante pour analyser et transformer les données stockées dans une base de données.

Elle permet de réaliser des calculs complexes sur des collections, comme des regroupements, des filtres, et des transformations.

Les pipelines d'agrégation utilisent une série d'étapes qui transforment les documents d'entrée pour produire des résultats.

Objectifs À la fin de ce cours, vous serez capable de : 1. Comprendre les bases des pipelines d'agrégation. 2. Utiliser les principales étapes de l'agrégation. 3. Réaliser des opérations avancées comme le regroupement, les tris et les projections.

---

1. Concepts clés

Pipeline d'agrégation Un pipeline est une séquence d'étapes où chaque étape reçoit les données en entrée, les traite, puis transmet les résultats à l'étape suivante.
Les principales étapes d'un pipeline incluent (respecter cet ordre) - $match**$: $Filtrerlesdocuments. - **$**group** : Regrouper les documents et appliquer des calculs. - $project**$: $Transformerourestructurerlesdocuments. - **$**sort** : Trier les documents. - $limit**/$**skip** : Limiter ou ignorer un nombre de documents. - **$unwind** : Décomposer des tableaux pour traiter chaque élément individuellement.

- pipeline

```
data -> traitement -> data -> traitement ...
```

Exemple de syntaxe de base

```
// compter le nombre de restaurant par quartie
// 1.aggregate c'est un framework d'agrégation
// $group: permet de créer les groupement avec _id : le nom du champ $nom_du_champ
// appliquer une fonction de groupement, ici on applique la fonction count sur chaque groupe
db.restaurants.aggregate([
    { $group: { _id: "$borough", count: { $sum: 1} } },
]);

db.restaurants.aggregate([
    { $match: { borough: "Manhattan" } }, // Étape 1 : Filtrer les documents
    { $group: { _id: "$cuisine", count: { $sum: 1 } } }, // Étape 2 : Regrouper par type de
```

```
    { $sort: { count: -1 } } // Étape 3 : Trier par ordre décroissant
]);
```

---

2. Principales Étapes du Pipeline

****2.1** *match* ∗ ∗ : *FiltrerlesdocumentsL'étape*‘**match‘ sélectionne uniquement les documents répondant à certains critères.** **Exemple** : Trouver les restaurants dans le Bronx :

```
db.restaurants.aggregate([
    { $match: { borough: "Bronx" } } // comme le premier {} du find
]);
```

****2.2** *project*∗∗ : *RestructurerlesdocumentsUtilisez*‘**project‘ pour inclure, exclure ou transformer des champs.** **Exemple** : Afficher uniquement le nom et la cuisine :

```
db.restaurants.aggregate([
    { $project: { name: 1, cuisine: 1, _id: 0 } } // définir les champs à récupérer
]);
```

****2.3** *group* ∗ ∗ : *RegrouperetcalculerUtilisez*‘**group‘ pour regrouper les documents par une clé et effectuer des calculs (somme, moyenne, etc.).**

- `_id` est une clé définit par MongoDB qui permet de définir le clé de regroupement.
- `count` est une clé qui permet de faire un calcul sur les groupements.

Exemple : Compter le nombre de restaurants par type de cuisine :

```
db.restaurants.aggregate([
    { $group: { _id: "$cuisine", count: { $sum: 1 } } }
]);
```

****2.4** *sort* ∗ ∗ : *TrierlesdocumentsUtilisez*‘**sort‘ pour organiser les documents.** **Exemple** : Trier les cuisines par popularité :

```
db.restaurants.aggregate([
    { $group: { _id: "$cuisine", count: { $sum: 1 } } },
    { $sort: { count: -1 } }
]);
```

****2.5** *unwind*∗∗ : *DécomposerdestableauxUtilisez*‘**unwind‘ pour “déplier” les tableaux en plusieurs documents.** **Exemple** : Afficher chaque évaluation (grade) individuellement :

```
db.restaurants.aggregate([
    { $unwind: "$grades" },
    { $project: { name: 1, grades: 1, _id: 0 } }
]);
```

****2.6 $limit et** *skip* ∗ ∗ : *Limiterouignorer* – ‘**limit: Réduit le nombre de documents retournés. -$skip‘ : Ignore un nombre donné de documents. Exemple** : Limiter les résultats à 5 restaurants :

```
db.restaurants.aggregate([
    { $limit: 5 }
]);
```

---

3. Exercices Pratiques

Exercice 1 : Trouver les cuisines les plus populaires

1. Filtrez les restaurants du borough “Manhattan”.
2. Groupez par type de cuisine.
3. Comptez le nombre de restaurants pour chaque cuisine.
4. Triez par ordre décroissant.

Exercice 2 : Calculer la moyenne des scores

1. Filtrez les restaurants de cuisine “Italian”.
2. Unwind les évaluations (`grades`).
3. Calculez la moyenne des scores pour chaque restaurant.

Exercice 3 : Afficher les restaurants avec au moins 3 évaluations

1. Comptez le nombre d’éléments dans `grades`.
2. Filtrez les restaurants avec au moins 3 évaluations.

---

4. Cas Pratique : Analyse des restaurants à New York

Objectif : Utiliser un pipeline d’agrégation pour analyser le jeu de données des restaurants de New York. Pipeline complet :

```
db.restaurants.aggregate([
    { $match: { borough: "Brooklyn", cuisine: "Pizza" } },
    { $unwind: "$grades" },
    { $group: {
        _id: "$name",
        averageScore: { $avg: "$grades.score" }
    }},
```

```
    { $sort: { averageScore: -1 } },
    { $limit: 10 }
]);
```

Ce pipeline : - Filtre les restaurants de Brooklyn servant des pizzas. - Décompose les évaluations en éléments individuels. - Calcule la moyenne des scores pour chaque restaurant. - Trie par score décroissant et limite les résultats aux 10 meilleurs.

---

5. Points Clés à Retenir

1. **Étapes de base** : `$match`, `$group`, `$project`, `$sort`, `$unwind`.
2. Les pipelines d'agrégation sont transformateurs et ne modifient pas les données source.
3. Expérimentez avec différentes étapes pour explorer vos données.

Ressources

- Documentation officielle de MongoDB sur l'agrégation
- Exercices interactifs : MongoDB University