TP MONGODB - INDEXATION, GÉOSPATIAL ET AGRÉGATION

Formation MongoDB UA3-14 ISITECH

Introduction

Dans ce TP, vous allez travailler sur trois aspects avancés de MongoDB en vous appuyant sur le projet de bibliothèque (bibliotheque_amazon) que vous avez déjà développé. Vous explorerez les performances apportées par l'indexation, les capacités géospatiales de MongoDB, et la puissance analytique du framework d'agrégation.

Prérequis

- Une instance MongoDB fonctionnelle (locale ou Atlas)
- Une base de données nommée `bibliotheque_amazon` contenant au minimum :
  - Une collection `livres` avec des données sur les livres
  - Une collection `utilisateurs` avec des données sur les utilisateurs

Si vous n'avez pas encore ces collections, ou si vous souhaitez générer plus de données pour vos tests, des scripts de génération sont fournis à la fin de ce document.

Structure du TP

Ce TP est divisé en trois parties distinctes mais complémentaires :

1. **Indexation et optimisation des performances**
2. **Requêtes géospatiales**
3. **Framework d'agrégation**

Pour chaque partie, vous réaliserez des exercices pratiques puis vous rédigerez un bref rapport d'analyse.

TP 1 : Indexation et optimisation des performances

Objectifs

- Analyser l'impact des index sur les performances des requêtes
- Créer et configurer différents types d'index
- Comprendre les compromis entre performance de lecture et d'écriture

Exercice 1.1 : Préparation et analyse des performances sans index

1. Si vous n'avez pas suffisamment de données, utilisez le script de génération de livres fourni pour ajouter au moins 1000 documents dans votre collection `livres`.

2. Analysez les performances des requêtes suivantes **sans index** en utilisant `explain("executionStats")` :

- Une recherche par titre exact
- Une recherche par auteur
- Une recherche par plage de prix (ex: entre 10€ et 20€) et note minimale
- Une recherche filtrée par genre et langue avec tri par note décroissante

3. Pour chaque requête, notez dans un tableau :
   - Le nombre de documents examinés (totalDocsExamined)
   - Le temps d'exécution (executionTimeMillis)
   - Le type d'étape utilisée (COLLSCAN, etc.)

Exercice 1.2 : Création d'index simples et composites

1. Créez des index appropriés pour optimiser chacune des requêtes précédentes. Réfléchissez au choix entre index simples et composites en fonction des critères de recherche.
2. Exécutez à nouveau les mêmes requêtes avec `explain("executionStats")` et comparez les résultats. Notez particulièrement l'amélioration des performances.

Exercice 1.3 : Index spécialisés

1. Créez un index de texte (text index) sur les champs titre et description des livres.
2. Testez une recherche de texte simple et analysez ses performances.
3. Créez une nouvelle collection `sessions_utilisateurs` avec des documents contenant :
   - Un ID d'utilisateur
   - Une date de dernière activité
   - Des données de session
4. Créez un index TTL sur cette collection pour supprimer automatiquement les sessions après 30 minutes d'inactivité.

Exercice 1.4 : Optimisations avancées

1. Créez un index qui permet d'obtenir une requête couverte (covered query). Vérifiez que la requête n'examine aucun document (totalDocsExamined = 0).
2. Créez un index unique sur le champ ISBN des livres et testez son comportement en essayant d'insérer un document avec un ISBN dupliqué.
3. Créez un index partiel qui n'indexe que les livres disponibles.
4. Activez le profiler MongoDB pour identifier les requêtes lentes (prenant plus de 100ms).
5. Identifiez et supprimez un index redondant ou peu utilisé.

Livrables TP1

1. Un document présentant les résultats de vos tests de performance (avant/après indexation) sous forme de tableau comparatif.

2. Un rapport concis (environ 1 page) répondant aux questions suivantes :
    - Quelles améliorations de performance avez-vous observées après l'ajout d'index ?
    - Quels types d'index ont été les plus efficaces pour vos requêtes ?
    - Avez-vous identifié des compromis entre performance de lecture et d'écriture ?
    - Comment choisiriez-vous les index pour une application de bibliothèque en production ?

TP 2 : Requêtes géospatiales

Objectifs

- Enrichir votre base de données avec des données géospatiales
- Créer des index géospatiaux
- Implémenter des fonctionnalités basées sur la localisation

Exercice 2.1 : Enrichissement des données

1. Modifiez le schéma de vos utilisateurs pour inclure des coordonnées géographiques dans leur adresse. Utilisez le format GeoJSON Point :

```
adresse: {
  rue: String,
  ville: String,
  code_postal: String,
  localisation: {
    type: "Point",
    coordinates: [longitude, latitude] // Attention à l'ordre:
longitude puis latitude
  }
}
```

2. Créez une nouvelle collection `bibliotheques` avec au moins 3 bibliothèques dans différentes villes. Chaque document doit contenir :
    - Un nom et une adresse
    - Des coordonnées GeoJSON Point pour la localisation
    - Une zone de service définie comme un polygone GeoJSON
3. Créez les index géospatiaux nécessaires sur les collections `utilisateurs` et `bibliotheques`.

Exercice 2.2 : Requêtes de proximité

1. Trouvez les 5 utilisateurs les plus proches d'un point donné (par exemple, le centre de Paris) dans un rayon de 5km.
2. Trouvez les bibliothèques les plus proches d'un utilisateur spécifique.
3. Utilisez l'opérateur `$geoNear` dans un pipeline d'agrégation pour obtenir les bibliothèques triées par distance et calculer précisément cette distance (en km).

Exercice 2.3 : Requêtes géospatiales avancées

1. Utilisez `$geoWithin` pour trouver tous les utilisateurs à l'intérieur d'une zone définie par un polygone (par exemple, un quartier de Paris).
2. Trouvez tous les utilisateurs qui se trouvent dans la zone de service d'une bibliothèque spécifique.
3. Créez une collection `rues` avec au moins une rue représentée comme un LineString GeoJSON, puis utilisez `$geoIntersects` pour trouver les bibliothèques dont la zone de service intersecte cette rue.

Exercice 2.4 : Cas d'utilisation métier

1. Créez une collection `livraisons` pour suivre les livraisons de livres, avec :
    - Des références vers les livres et les utilisateurs
    - Un point de départ (bibliothèque) et un point d'arrivée (adresse utilisateur)
    - Une position actuelle du livreur (Point GeoJSON)
    - Un itinéraire planifié (LineString GeoJSON)
2. Implémentez une fonction pour mettre à jour la position d'une livraison.
3. Créez une requête pour trouver toutes les livraisons en cours dans un rayon de 1km autour d'un point donné.

Livrables TP2

1. Les requêtes MongoDB réalisées pour chaque exercice avec des commentaires explicatifs.
2. Un rapport concis (environ 1 page) répondant aux questions suivantes :
    - Comment les fonctionnalités géospatiales peuvent-elles améliorer un service de bibliothèque ?
    - Quels défis avez-vous rencontrés lors de l'implémentation des requêtes géospatiales ?
    - Comment intégreriez-vous ces fonctionnalités dans une application web ou mobile ?

TP 3 : Framework d'agrégation

Objectifs

- Maîtriser les différentes étapes du pipeline d'agrégation MongoDB
- Réaliser des analyses statistiques sur les données de la bibliothèque
- Comprendre les performances des opérations d'agrégation

Exercice 3.1 : Agrégations de base

1. Créez un pipeline d'agrégation pour calculer les statistiques suivantes par genre de livre :
    - Nombre de livres
    - Note moyenne
    - Prix moyen, minimum et maximum
2. Analysez la répartition des livres par éditeur :
    - Nombre de livres
    - Nombre de genres différents
    - Nombre d'auteurs différents

- Note moyenne

3. Enrichissez votre collection d'utilisateurs avec des données d'emprunt si ce n'est pas déjà fait, puis créez un pipeline d'agrégation pour analyser les habitudes d'emprunt par ville d'utilisateur.

Exercice 3.2 : Agrégations avancées

1. Analysez les durées d'emprunt en calculant :
   - La durée moyenne de prêt
   - Les durées minimale et maximale
   - Le pourcentage d'emprunts retournés en retard
2. Utilisez l'opérateur `$lookup` pour joindre les collections `livres` et `utilisateurs` afin d'analyser quels livres sont les plus empruntés et par qui.
3. Utilisez l'opérateur `$facet` pour créer un tableau de bord complet avec plusieurs analyses en un seul pipeline :
   - Statistiques globales (nombre total de livres, prix moyen, note moyenne)
   - Top 5 des livres les mieux notés
   - Répartition par langue
   - Répartition par décennie de publication

Exercice 3.3 : Cas d'usage métier

1. Créez un système de recommandation basique qui suggère des livres à un utilisateur en fonction des genres qu'il a déjà empruntés.
2. Analysez les tendances d'emprunt par mois ou par saison pour identifier les périodes de forte activité.
3. Identifiez les livres jamais empruntés et depuis combien de temps ils sont en stock.

Exercice 3.4 : Optimisations et performances

1. Créez une vue (view) MongoDB pour une de vos agrégations fréquemment utilisées.
2. Analysez les performances d'une de vos agrégations avec `explain()` et identifiez les étapes qui pourraient bénéficier d'un index.
3. Utilisez l'option `allowDiskUse: true` pour une agrégation complexe qui traite un grand volume de données.

Livrables TP3

1. Les pipelines d'agrégation réalisés pour chaque exercice, avec des commentaires explicatifs.
2. Un rapport concis (environ 1 page) répondant aux questions suivantes :
   - Comment le framework d'agrégation améliore-t-il l'analyse des données par rapport aux simples requêtes find() ?
   - Quels opérateurs d'agrégation avez-vous trouvé les plus utiles et pourquoi ?

- Quels défis avez-vous rencontrés lors de l'implémentation des pipelines d'agrégation complexes ?
- Comment pourriez-vous optimiser davantage les performances de vos agrégations ?

Conseils généraux

- **Approche progressive** : Construisez vos requêtes étape par étape et testez chaque étape individuellement.
- **Documentation** : Consultez régulièrement la documentation officielle MongoDB.
- **Données de test** : Assurez-vous d'avoir suffisamment de données pour que vos résultats soient significatifs.
- **Performance** : Utilisez `explain()` pour comprendre et optimiser vos requêtes.
- **Préfixe GeoJSON** : Respectez l'ordre des coordonnées [longitude, latitude] dans GeoJSON.

Annexe : Scripts de génération de données

Script de génération d'utilisateurs

Voici le début d'un script pour générer des utilisateurs. Complétez-le et adaptez-le à vos besoins :

```
// À exécuter dans mongosh connecté à votre base de données
// Génération d'utilisateurs avec localisation
const prenoms = ["Jean", "Marie", "Pierre", "Sophie", "Thomas",
"Isabelle"];
const noms = ["Martin", "Bernard", "Dubois", "Thomas", "Robert",
"Richard"];
const villes = [
  { nom: "Paris", coordinates: [2.3522, 48.8566] },
  { nom: "Lyon", coordinates: [4.8357, 45.7640] },
  { nom: "Marseille", coordinates: [5.3698, 43.2965] }
];

// Complétez ce script pour générer des utilisateurs avec localisation
```

Script de génération de livres

Voici le début d'un script pour générer des livres. Complétez-le et adaptez-le à vos besoins :

```
// À exécuter dans mongosh connecté à votre base de données
// Génération de livres
const genres = ["Fiction", "Science-fiction", "Philosophie", "Histoire"];
const langues = ["Français", "Anglais", "Espagnol", "Allemand"];
const editeurs = ["Gallimard", "Hachette", "Flammarion", "Actes Sud"];

// Complétez ce script pour générer des livres avec des attributs
aléatoires
```

Ressources

- Documentation MongoDB sur les index
- Documentation sur les requêtes géospatiales
- Documentation sur le framework d'agrégation
- MongoDB Atlas pour visualiser les résultats géospatiaux
- GeoJSON Validator pour vérifier vos données géospatiales

Évaluation

Vous serez évalué sur :

1. La maîtrise technique des fonctionnalités MongoDB
2. La pertinence des solutions proposées
3. La qualité des analyses et des rapports
4. La capacité à optimiser les performances
5. La clarté des commentaires et de la documentation, ça marche ?

Bon courage et bon TP !