TP n°3 : Indexation et validation

Exercice 1 : Mise en bouche

- Importer le fichier restaurants.json dans MongoDB.
- Vérifier que la collection restaurants est bien créée.

Recherches simples :

Rechercher le restaurant **"Talia'S Steakhouse"** :

Utilisez la méthode .explain() pour avoir le détail de la requête ; notez ces détails.

Exercice 2 : Entrée avec l'indexation

1. **Création et observation d'un index simple :**

   - Créer un index sur le champ borough :
   - db.restaurants.createIndex({ borough: 1 })
   - Réaliser les requêtes suivantes avant et après création de l'index et analyser les différences avec explain() :

     1. Trouver tous les restaurants situés à Manhattan :
     2. Trouver tous les restaurants ayant une cuisine **"Chinese"** dans Brooklyn

2. **Création d'un index composé :**

   - Créer un index sur les champs cuisine et borough :
   - Effectuer les requêtes suivantes :

     1. Rechercher les restaurants ayant une cuisine **"Chinese"** dans **Queens**.
     2. Rechercher les restaurants ayant une cuisine **"American"** dans **Brooklyn**, avec uniquement les champs name et address.

3. **Supprimer un index et observer les impacts :**

   - Supprimer l'index précédent et comparer les performances :

EXERCICE 3 : Plat principal : On valide !

1. **Validation simple :**

   - Ajouter une règle imposant que les restaurants doivent avoir un champ name et un champ borough :

2. **Test de la validation :**

   - Insérer un document incomplet et analyser l'erreur :
   - Ajouter un document valide :

3. Configuration d'une validation avancée :
   Ajoutez une règle qui vérifie :
   - Les restaurants doivent avoir un champ `name` non vide.
   - Le champ `grades.score` (score de l'évaluation) doit être un entier strictement positif.
   - Le champ `borough` doit être l'une des valeurs suivantes : "Manhattan", "Brooklyn", "Queens", "Bronx", "Staten Island".

a) **Cas valide** : Insérez un restaurant conforme :
b) **Cas invalide 1** : Essayez d'insérer un restaurant avec un borough invalide
c) **Cas invalide 2** : Essayez d'insérer un restaurant avec un score négatif :

Exercice 4 : Dessert géospatial

1. **Création d'un index géospatial :**

   - Créer un index 2D sur le champ address.coord.

2. **Recherche géospatiale :**

   - Trouver les restaurants dans un rayon de 1 km autour de **Talia'S Steakhouse** :

3. **Requête combinée :**

   - Trouver des restaurants dans un rayon de 2 km proposant une cuisine **"Italian"**.

4. **Recherche par polygone :**

   Trouvez tous les restaurants situés à l'intérieur de l'île de Manhattan.

   Pour vous aider, voici les coordonnées de Manhattan:

   Nord-Ouest : `[-74.0479, 40.8820]` — Point situé près de Spuyten Duyvil Creek, à l'extrémité nord-ouest de Manhattan.
   Nord-Est : `[-73.9067, 40.8820]` — Point situé près du pont Washington Heights, à l'extrémité nord-est de Manhattan.
   Sud-Est : `[-73.9067, 40.6829]` — Point situé à l'extrémité sud-est, près du pont Brooklyn.
   Sud-Ouest : `[-74.0479, 40.6829]` — Point situé près de Battery Park, à l'extrémité sud-ouest de Manhattan.

   Combien trouvez-vous de restaurants ?

   Comparez votre résultat avec la requête : **db.restaurants.find({ borough: "Manhattan" }).count()**

   Quelle requête est la plus efficace ?