
AJC
FORMATION


mongo
DB

Objectifs

Comprendre le mouvement NoSQL

Installer MongoDB

Comprendre l'architecture de MongoDB

Travailler avec les objets dans MongoDB

Optimiser les performances

Plan

Présentation de MongoDB

Prise en main de MongoDB

Travailler avec les documents

Performance et indexation

Environnement

Prés-requis

- Un espace disque de 10G
- OS
  - Linux Ubuntu 20

Présentation de MongoDB

MongoDB

MongoDB (hu**mongo**us = énorme)

Base de données orientée document

Open-source

Développé en C++

Données stockées sous forme documents (par exemple JSON)

Absence de tables

Utilise la notion de « schemaless » (sans schéma)

Avantages de MongoDB

Répond aux besoins de performances

Garantit la scalabilité horizontale (réplication et sharding)

Nombreuses fonctionnalités (count, group by, order by, SUM, MIN, etc.)

Supporte l'indexation pour optimiser les performances

Scalabilité - Vertical vs Horizontal

Lorsque la charge serveur augmente, et qu'il est temps de rajouter des ressources matérielles à votre infrastructure, deux approches s'offrent à vous : augmenter vos ressources **verticalement ou horizontalement**. **On parle de scaling vertical versus scaling horizontal**.

Scalabilité - Vertical vs Horizontal

Scalabilité Verticale

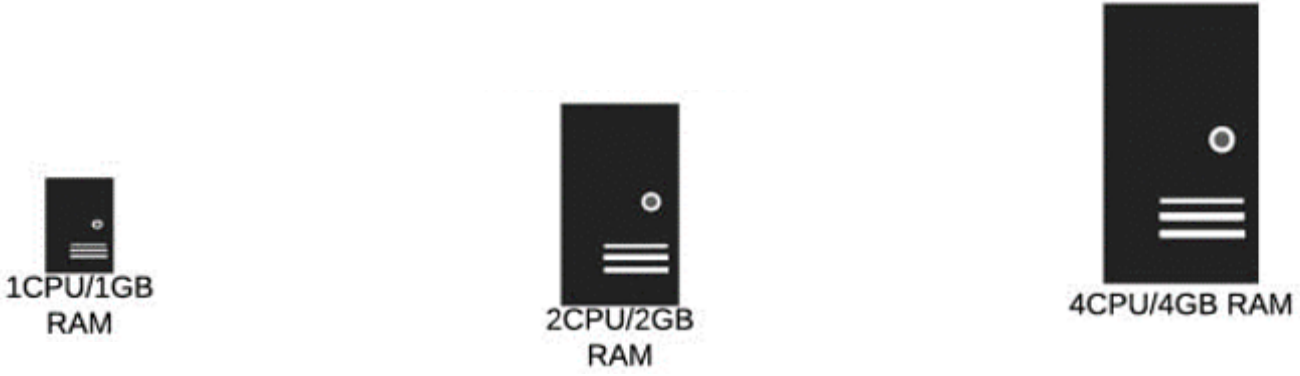
La scalabilité verticale est la plus intuitive : cela revient à ajouter des ressources à un unique serveur (en lui ajoutant de la RAM, en changeant son CPU pour un plus véloce…). C'est simple ! Mais :

Le coût est rapidement exponentiel de la capacité matérielle.

Scalabilité - Vertical vs Horizontal

Scalabilité Verticale

Cette solution a des limites : vous pourrez probablement multiplier par 2 les capacités de votre serveur, peut-être même par 5, mais pas par 100 !
Cela reste dans la majorité des cas la solution la plus pragmatique, et en particulier pour des applicatifs en production depuis de nombreuses années.



Scalabilité - Vertical vs Horizontal

Scalabilité Horizontale

La scalabilité horizontale revient à ajouter de nouveaux serveurs réalisant le même type tâche. Cela permet de n'utiliser que des serveurs standards (on parle de commodity hardware). Mais les implications logicielles sont rapidement importantes !

La présence sur le marché de MongoDB

Utilisé par de grands comptes

- MTV
- Disney
- Doodle
- CERN (Organisation Européenne pour la Recherche Nucléaire)

Analogie avec le SQL

NoSQL (Base) = SQL (Base)

NoSQL (Collection) = SQL (Table)

NoSQL (Document) = SQL (Enregistrement)

Dans les bases SQL chaque enregistrement de la table contient exactement les mêmes champs, seul le contenu varie

Dans une collection MongoDB, les documents peuvent avoir des champs totalement différents

Prise en main de MongoDB

Installation sous Linux

Installation sous Ubuntu / Debian

Instance MongoDB

Arrêt / Démarrage d'une instance MongoDB

✓ Sous Linux

- Démarrage
  - **`sudo service mongod start`** (démarrage de mongoDB via script)
  - **`mongod --dbpath <racine_instance>`** (démarrage de mongoDB)
  - **`mongod -f <fichier.conf>`**
- Arrêt
  - **`sudo service mongod stop`** (permet d'arrêter mongoDB)
  - **`sudo service mongod restart`** (permet le redémarrage de mongoDB)
  - Commande **`db.shutdownServer()`** à partir du client mongo et la base admin

Les bases de données

Création / Suppression

Création de bases de données

- Automatique
  - mongo <db_name>
  - use <db_name>

Suppression de bases de données

- mongo
- use <db_name>;
  - db.runCommand({dropDatabase: 1});
  - db.dropDatabase();

mongo DB

Travailler avec les documents

Les collections

Une Collection c'est quoi ?

- Equivalent de la table en relationnel
- Deux modes de création
  - Automatiquement lors d'une insertion de document (tuple)
  - En exécutant la commande db.createCollection('<nom_collection>');

Les collections

Les opérations sur les Collections

Créer

```
db.createCollection('collection_name');
```

Lister

```
show collections;
db.getCollections();
```

Insérer un document

```
db.<collection_name>.insert( { var1: "valeur", var2: "valeur",
var3: "valeur"} );
```

Supprimer

```
db.<nom_collection>.drop();
```

Les documents

Un document c'est quoi ?

Equivalent d'un enregistrement (tuple) en relationnel.

Deux modes d'insertion

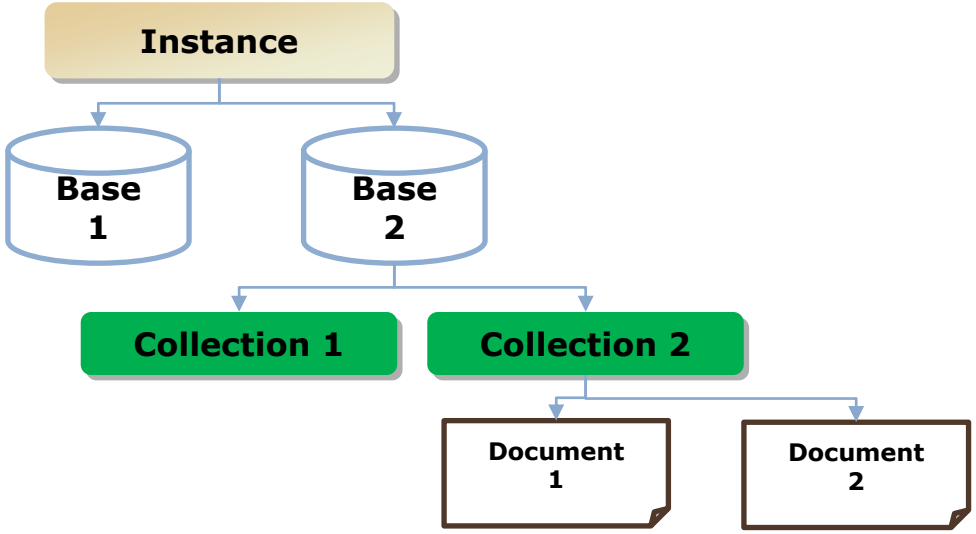
- En utilisant la commande **db**.<nom_collection>.**insert()** du shell
- En utilisant la commande **db**.<nom_collection>.**save()** du shell

Chaque document crée contient le champ « _id »

- Généré automatiquement
- Type primary key
- Indexé

Les documents

Rappel de la hiérarchie des objets



Les documents

Le format JSON/BSON

- MongoDB stocke les documents au format BSON (Binary JSON)
- BSON est la représentation binaire des objets JSON (JavaScript Object Notation)

Les documents

Le format JSON c'est quoi ?

JSON (JavaScript Object Notation)

- Permet de représenter de l'information structurée
- Format de données textuelle
- Utilise une notation JavaScript
- Un document JSON, ne comprend que deux éléments structurels :

  - des ensembles de paires nom / valeur ;
  - des listes ordonnées de valeurs.

Les documents

Le format JSON

Exemple

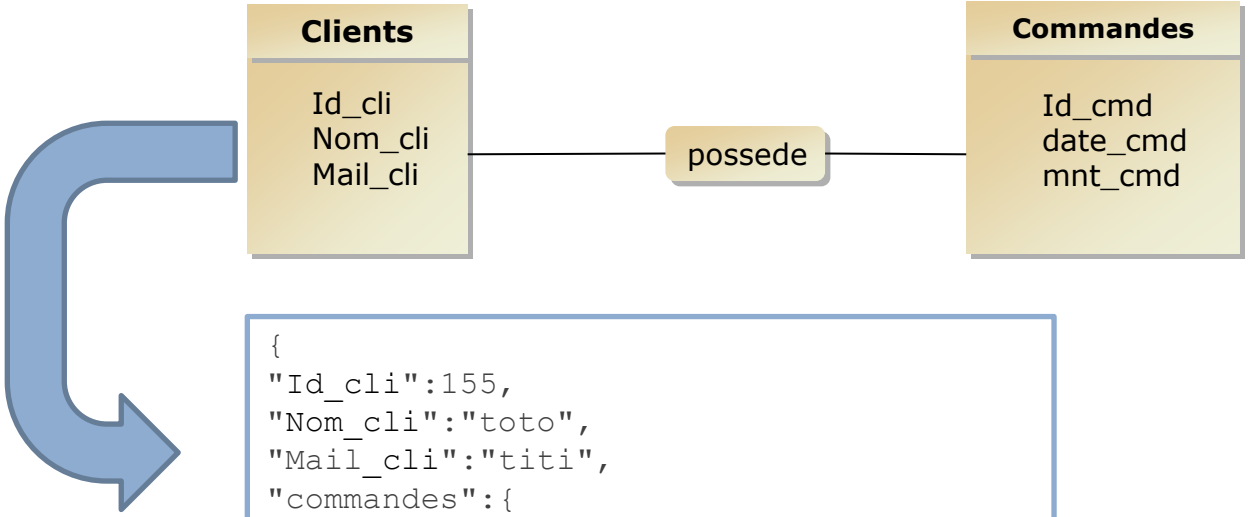
```
{
" nom" : " Johny ",
"prenom" : "TITI",
"adresse" : {
                "numero" : 77,
                "codepostal" : "31000",
                "ville" : "Toulouse"
        },
"nom" : "ALaoui",
"prenom" : "TITI",
"adresse" : {
                "numero" : 177,
                "codepostal" : "33000",
                "ville" : « Paris"  }
```

Document1

Document2

Les documents

Exemple pratique d'utilisation




```
{
"Id_cli":155,
"Nom_cli":"toto",
"Mail_cli":"titi",
"commandes":{
                "Id_cmd":12021,
                "date_cmd":"02-03-2015",
                "mnt_cmd":99.99
              }
}
```

Les documents

Avantage du format JSON

- Format abstrait pour une représentation simplifiée dans les différents langages
- Indépendant du langage de programmation
- Simple et complet pour la représentation des objets
- Utilise des types de données connus et simples à décrire
- Une bonne lisibilité de la syntaxe
- Temps de traitement très proche de celui des fichiers XML
- Moins volumineux en terme de stockage
- Pour JavaScript un document JSON représente un objet

Les documents

Types disponibles en JSON

- string
- number
- object
- array
- true
- false
- null

Les documents

Les types de données

- Une base MongoDB est constituée de Collection(s)
- Une collection est constituée de document(s) regroupant des paires cle/valeur dont la cle est une string et la valeur de type :
  - des objets ;
  - des tableaux ;
  - des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.

Les opérations CRUD

Les instructions CRUD

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

Les opérations CRUD

Insertion

Shell Javascript

- `db.collection_name.insert({var1:"valeur", var2:"valeur",var3:"valeur"})`
- `db.collection_name.find()`

ou

- `db.collection_name.save({var1:"valeur", var2:"valeur",var3:"valeur"})`
- `db.collection_name.find()`

Les opérations CRUD

Mise à jour
Shell

- db.collection_name.**update**(query,**update**,**options1**,**options2**)
- Spécification pour **update**
  - **$set** (permet de mettre a jour un champ)
  - **$unset** (permet de supprimer un champ)
  - **$inc** (permet d'incrémenter un champ)
  - …
- **Options1 : true** création du document s'il n'existe pas
  false pas de création de document
- **Options2 : true** mettre à jour tous les documents
  false mettre à jour uniquement le 1er document

Les opérations CRUD

La sélection

Shell

- `db.collection_name.find(critere)`

Les opérations CRUD

Les opérateurs de comparaison

- **'$gt'** = plus grand que
- **'$gte'** = plus grand ou égal à
- **'$lt'** = plus petit que
- **'$lte'** = plus petit ou égal à
- **'$ne'** = différent de
- **'$all':[v1,v2,vN]** = comporte toutes les valeurs
- **'$in':[v1,vN]** = comporte au moins une des valeurs
- **'$exist':true** = Le champ doit exister (ou pas si false)
- **/<chaine caractère>/** = équivalent du like en SQL

Les opérations CRUD

Suppression

- Shell
  - `db.collection.remove(critère)`

Performance et indexation

Performance et indexation

Les indexes sous MongoDB

Pourquoi faire ?

- Améliorer les temps d'exécution des requêtes

Les différents types d'indexes

- Index sur un seul champ
- Index unique
- Index composé

Indexation par défaut

- Le champ _id est indexé par défaut (impossible de le supprimer)

Performance et indexation

Index sur un seul champ

Deux modes

- Ordre ascendant (1 pour ASC)
- Ordre descendant (-1 pour DESC)

Poser un index (ascendant et descendant)

- db.<collection>.ensureIndex({<nom_champ>:1});
  Exemple: db.Personne.ensureIndex({'nom':1});
- db.<collection>.ensureIndex({<nom_champ>:-1});

Performance et indexation

Index composés

Deux modes

- Ordre ascendant (1 pour ASC)
- Ordre descendant (-1 pour DESC)

Poser un index (ascendant et descendant)

- db.<collection>.ensureIndex({<nom_champ1>:1, <nom_champ2>:1});
- db.<collection>.ensureIndex({<nom_champ1>:-1,<nom_champ2>:-1});

Performance et indexation

Les index unique

Deux modes

- Ordre ascendant (1 pour ASC)
- Ordre descendant (-1 pour DESC)

Poser un index unique (ascendant et descendant)

- db.<collection>.ensureIndex({<nom_champ1>:1, {unique: true});
- db.<collection>.ensureIndex({<nom_champ1>:-1, {unique: true});

Un index unique garantit que les champs indexés ne stockent pas de valeurs en double.

Performance et indexation

Visualisation des index

Index d'une collection

- db.<collection>.getIndexes()

Performance et indexation

- Reconstruction d'index

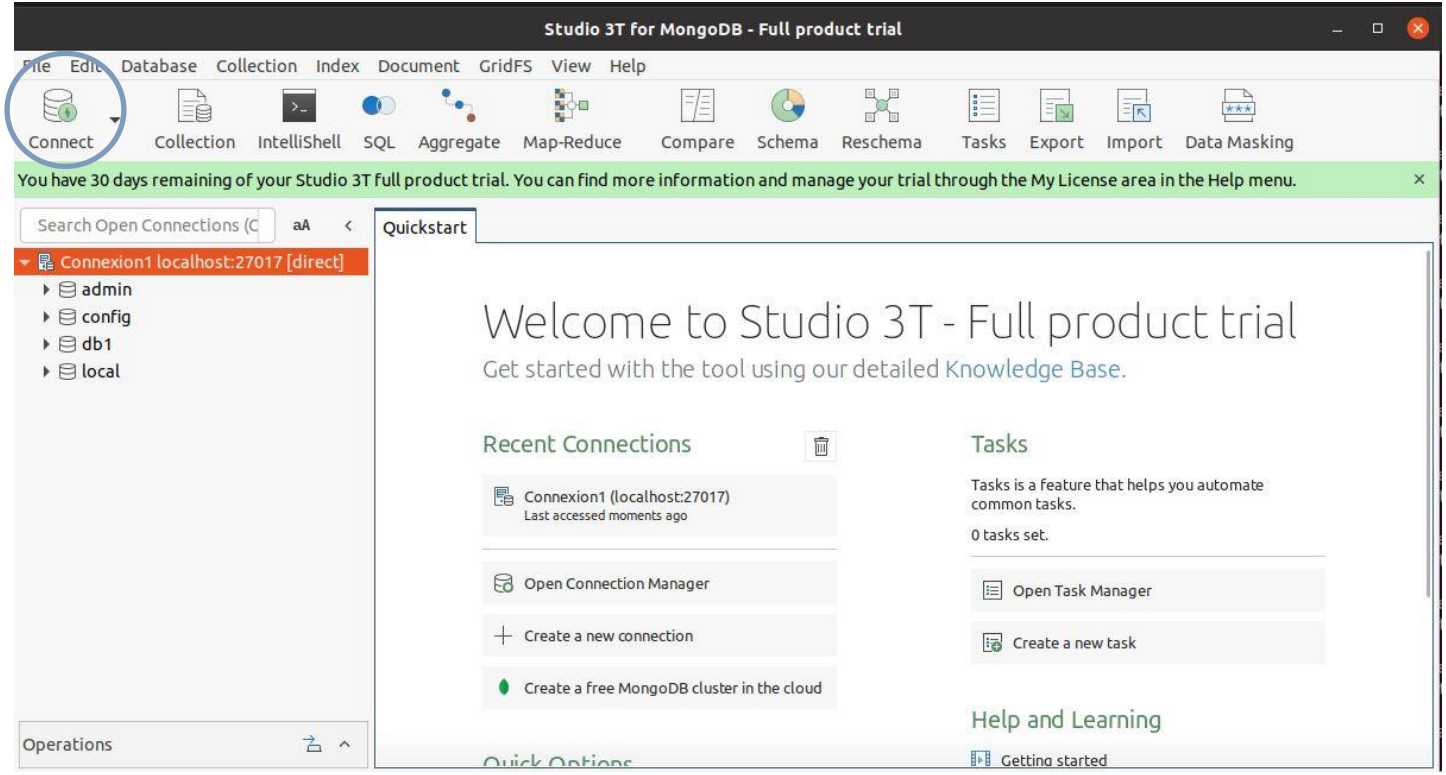
  Utilisation de la méthode db.collection.reIndex()

- Suppression d'index

  db.<collection>.dropIndex('<nom_index>')

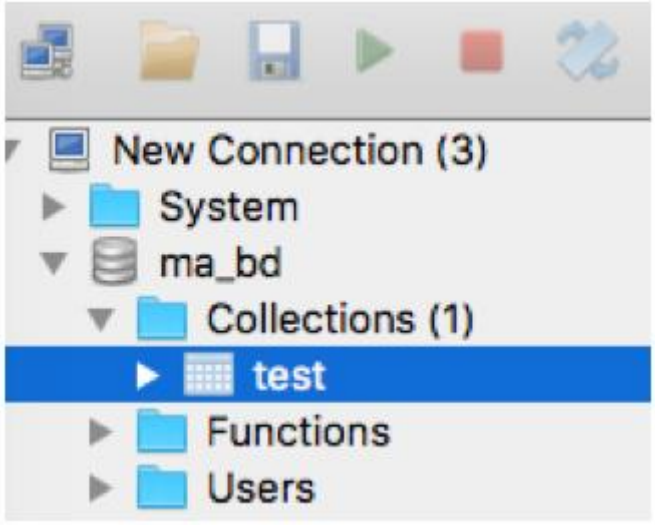
Travail Pratique

Création d'une nouvelle connexion



Création d'une collection

Créons tout d'abord une base de données 'ma_bd' (bouton droit sur la connexion "Create database"), puis sur les collections de cette base, créer une collection "test" (bouton droit sur "Collections(0)")



Modélisation de documents

On peut insérer un document dans notre collection “test” (double click sur la collection, champ de texte en noir pour exécuter une requête.

```
db.test.save (
{
  "cours" : "NoSQL",
  "chapitres" : ["familles", "CAP", "sharding", "choix"],
  "auteur" : {
     "nom" : "Travers",
     "prenom" : "Nicolas"
  }
} )
```

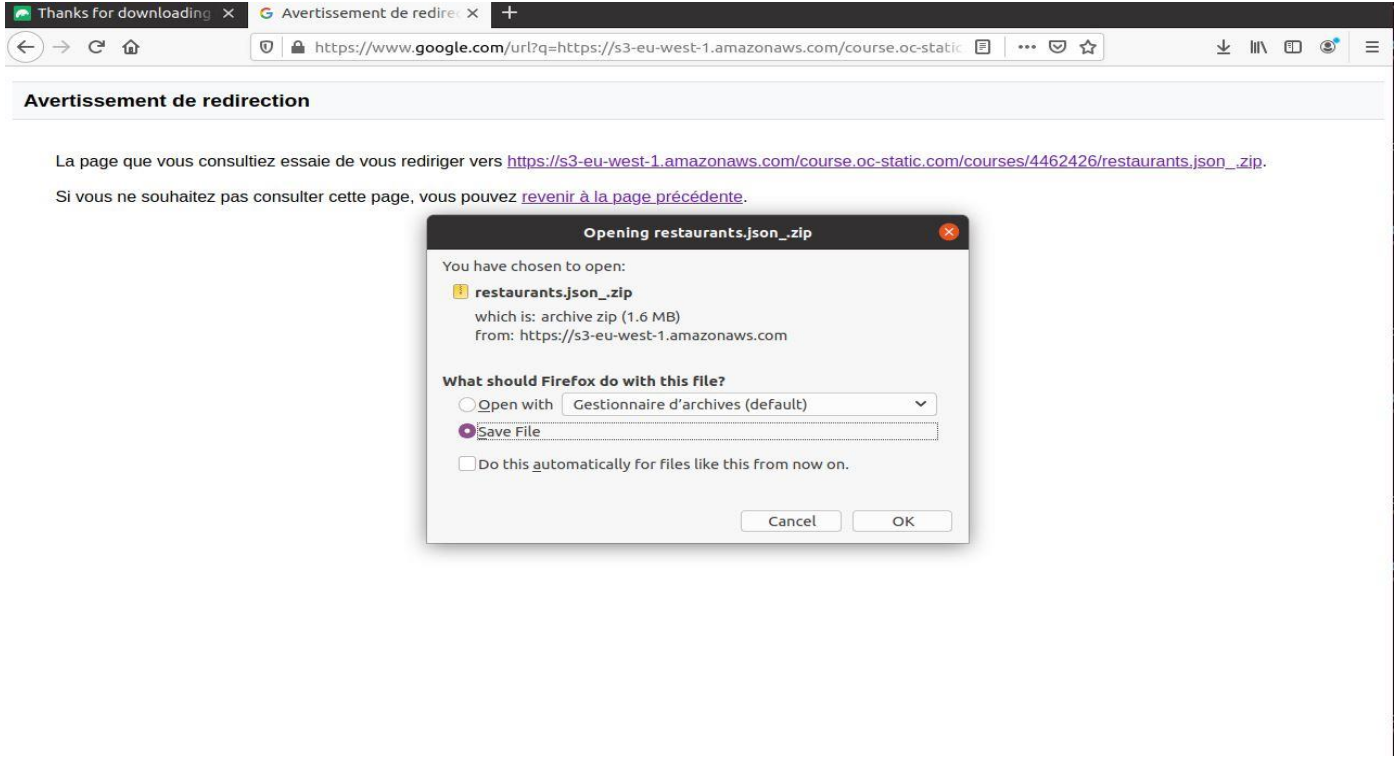
Importer les données

Nous allons maintenant importer un jeu de données déjà formaté à notre base de données. Un jeu de données OpenData est disponible sur des restaurants de New York.

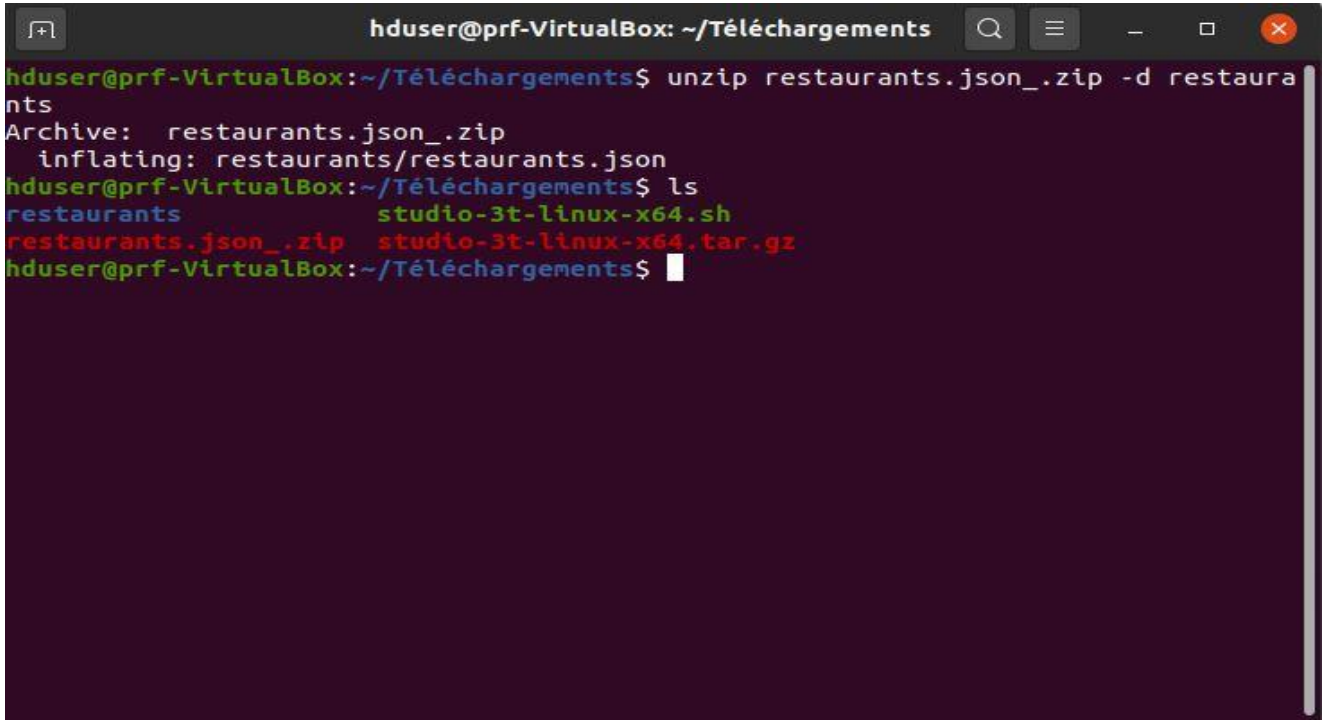
Pour l'importation :

1. Téléchargez l'archive suivante : restaurants.zip
2. Décompresser l'archive
3. Nous allons créer une base de données "new_york" (paramètre --db) et une collection "restaurants" (paramètre --collection). Attention, il ne faut pas de majuscules !

Importer les données



Importer les données

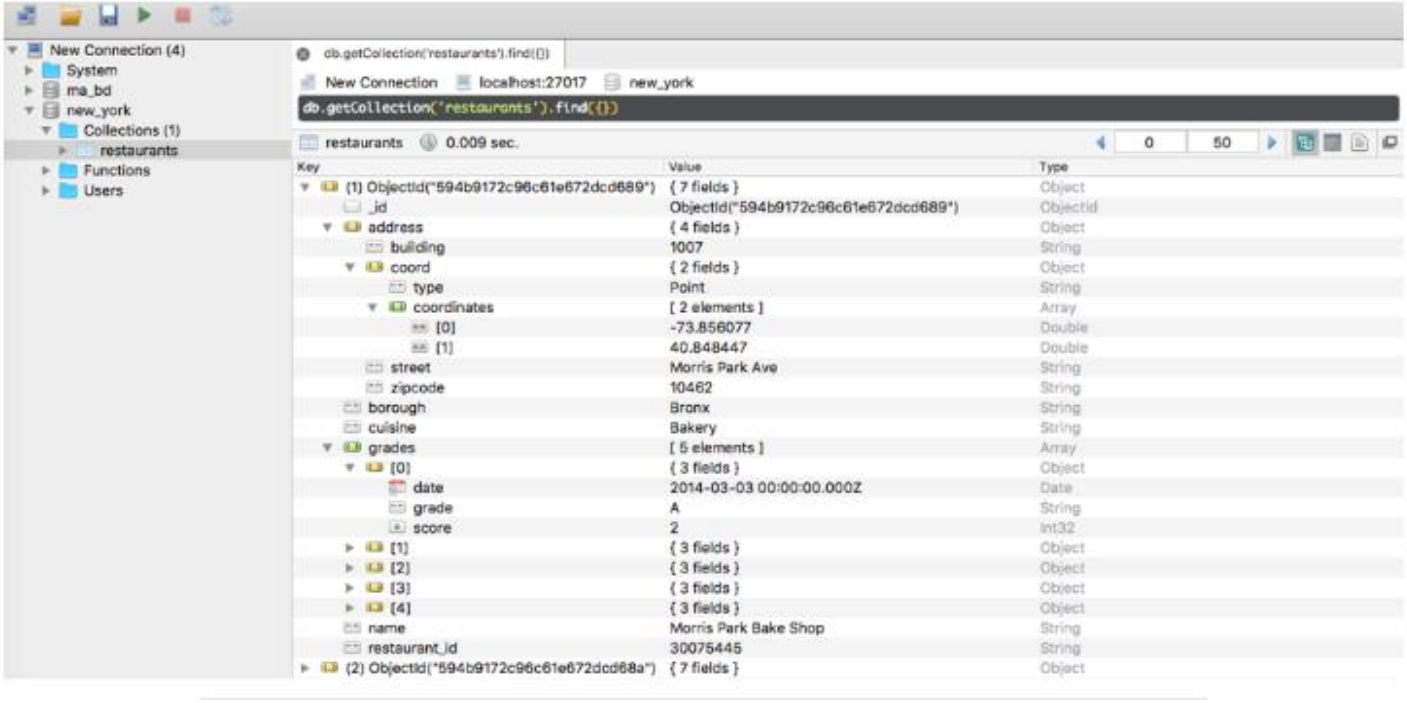


Importer les données

```
hduser@prf-VirtualBox:~$ mongoimport --db new_york --collection restaurants ~/Téléchargements/restaurants/restaurants.json
2022-09-01T00:34:42.687+0200    connected to: mongodb://localhost/
2022-09-01T00:34:43.972+0200    25357 document(s) imported successfully. 0 document(s) failed to import.
hduser@prf-VirtualBox:~$
```

Importer les données

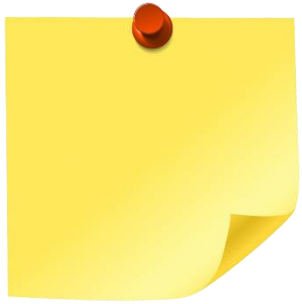
Vous avez une base de données de 25 357 restaurants que vous pouvez consulter directement avec Robo3T.



Interroger les données

Toute commande sur la collection restaurants utilise le préfixe : `"db.restaurants".`

Il suffira d'y associer la fonction souhaitée pour avoir un résultat.



Interroger les données

Avant de commencer, il faut voir à quoi ressemble un document de notre collection restaurants. Utilisons la fonction "findOne()".

```
db.restaurants.findOne()

{
    "_id" : ObjectId("594b9172c96c61e672dcd689"),
    "restaurant_id" : "30075445",
    "name" : "Morris Park Bake Shop",
    "borough" : "Bronx",
    "cuisine" : "Bakery",
    "address" : {
        "building" : "1007",
        "coord" :{"type":"Point","coordinates":[-73.856077,40.848447]},
        "street" : "Morris Park Ave",
        "zipcode" : "10462"
    },
    "grades" : [
        {"date" : ISODate("2014-03-03T00:00:00.000Z"),"grade" : "A","score" : 2},
        {"date" : ISODate("2013-09-11T00:00:00.000Z"),"grade" : "A","score" : 6},
        {"date" : ISODate("2013-01-24T00:00:00.000Z"),"grade" : "A","score" : 10},
        {"date" : ISODate("2011-11-23T00:00:00.000Z"),"grade" : "A","score" : 9},
        {"date" : ISODate("2011-03-10T00:00:00.000Z"),"grade" : "B","score" : 14}
    ]
}
```

Interroger les données

Chaque restaurant a un nom, un quartier (borough), le type de cuisine, une adresse (document imbriqué, avec coordonnées GPS, rue et code postale), et un ensemble de notes (résultats des inspections).

```
db.restaurants.findOne()

{
    "_id" : ObjectId("594b9172c96c61e672dcd689"),
    "restaurant_id" : "30075445",
    "name" : "Morris Park Bake Shop",
    "borough" : "Bronx",
    "cuisine" : "Bakery",
    "address" : {
        "building" : "1007",
        "coord" :{"type":"Point","coordinates":[-73.856077,40.848447]},
        "street" : "Morris Park Ave",
        "zipcode" : "10462"
    },
    "grades" : [
        {"date" : ISODate("2014-03-03T00:00:00.000Z"),"grade" : "A","score" : 2},
        {"date" : ISODate("2013-09-11T00:00:00.000Z"),"grade" : "A","score" : 6},
        {"date" : ISODate("2013-01-24T00:00:00.000Z"),"grade" : "A","score" : 10},
        {"date" : ISODate("2011-11-23T00:00:00.000Z"),"grade" : "A","score" : 9},
        {"date" : ISODate("2011-03-10T00:00:00.000Z"),"grade" : "B","score" : 14}
    ]
}
```

Interroger les données

Voir le TP