NoSQL

Une base de données SQL est entrée dans un bar NoSQL. Peu de temps après, elles en sont ressorties parce qu'elles n'ont pas pu trouver de table.

Introduction

Les bases de données NoSQL offrent la capacité de stocker d'importantes quantités de données. Aujourd'hui, il existe une multitude de solutions NoSQL telles que MongoDB, Cassandra, Redis, etc.

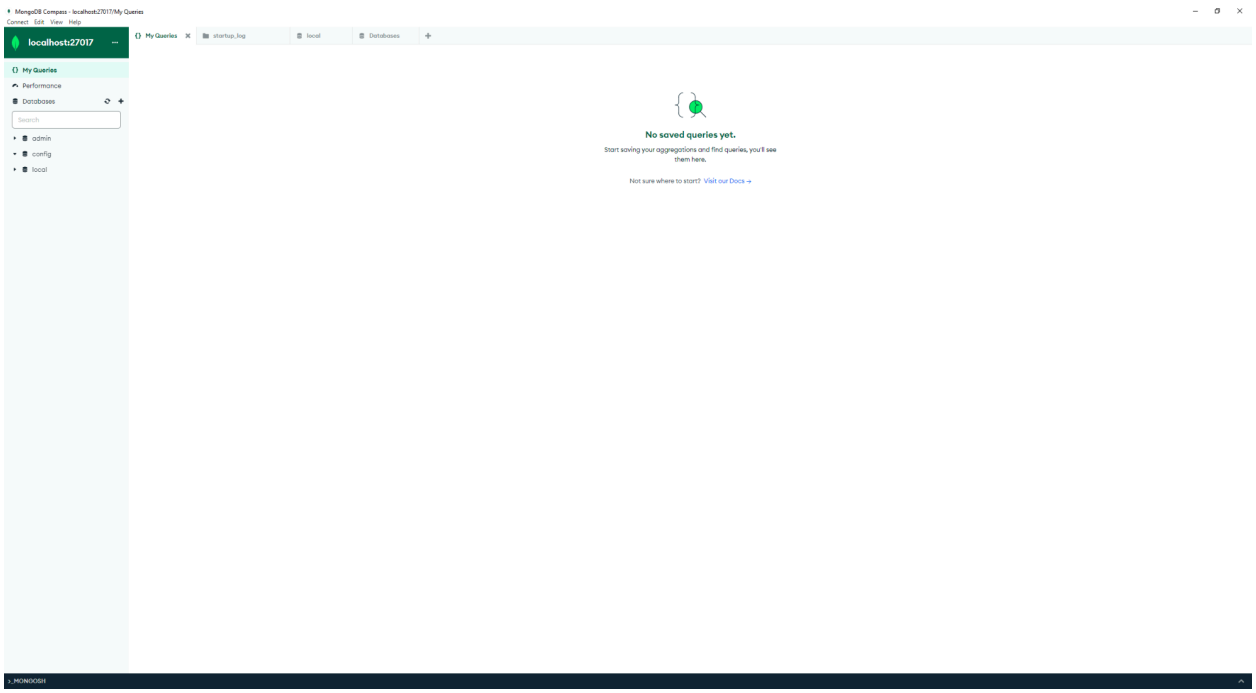
À la différence des bases de données relationnelles, les bases de données NoSQL n'utilisent pas de tables avec des colonnes, mais plutôt des documents, ce qui confère une grande flexibilité et des performances élevées. Chaque document est composé de paires clé valeur et est regroupé dans ce que l'on appelle une collection. Une base de données peut contenir plusieurs collections, tandis qu'une collection peut comprendre plusieurs documents. Cette modularité permet aux développeurs de façonner chaque collection selon leurs besoins spécifiques.

Commençons la découverte du NoSQL avec MongoDB !

Récupérez votre répertoire GitHub nommé “**runtrack-nodeJS**”. Dans ce répertoire créez un dossier “**jour3**”. Créer un fichier “**jour3.txt**” possédant pour chaque job, un titre “**jobXX**” ou **XX** est le numéro du job et les instructions No SQL demandées. N’oubliez pas d’envoyer vos modifications dès qu’un job est avancé ou terminé et mettez des commentaires explicites lors de vos commits.

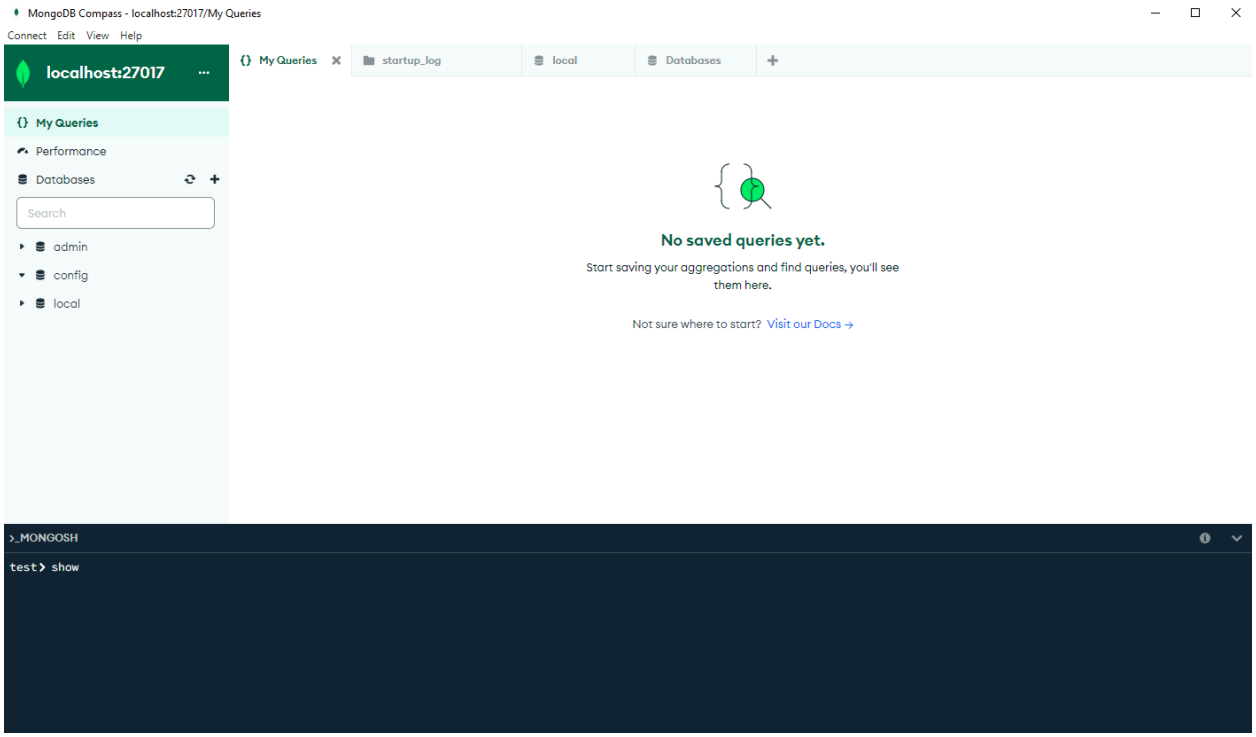
Job 1

Il est possible d’utiliser MongoDB de plusieurs façons, à **distance** ou **localement** en l’installant sur votre ordinateur. Installez sur votre ordinateur MongoDb en suivant les instructions de la documentation officielle.
Si tout s'est bien passé, vous devriez avoir accès à MongoDB compass :

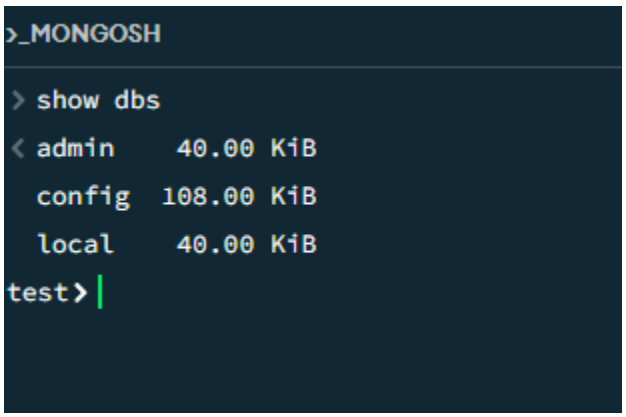


Job 2

MongoDb Compass vous met à disposition une console afin de pouvoir modérer votre base de données avec un système d'auto-complétion pour vous faciliter l'écriture des requêtes.



À l'aide de ce terminal, afficher l'**ensemble des bases de données** présentent dans Compass. Si tout s'est bien passé, vous devriez avoir le résultat suivant :



Job 3

Utiliser le **terminal MongoSH** pour créer une base de données nommée "**shop**".

Ajouter une collection nommée "**product**" et possédant les champs suivants :

- ➔ "**name**" ⇒ chaussure
- ➔ "**description**" ⇒ une paire de chaussures
- ➔ "**price**" ⇒ 35
- ➔ "**quantity**" ⇒ 120

Si tout s'est bien passé, vous devriez avoir dans le terminal :

```
< {
    acknowledged: true,
    insertedId: ObjectId('660db9f9d74a2b47eb20ab37')
  }
```

Job 4

Ajoutez les documents suivants à votre collection "**product**" :

- ➔ **name** ⇒ "Pantalon noir", **description** ⇒ "Un pantalon noir", **price** ⇒ 14,99, **quantity** ⇒ 12
- ➔ **name** ⇒ "chaussettes", **description** ⇒ "chaussettes cool !", **price** ⇒ 6, **quantity** ⇒ 6

Job 5

Créez une nouvelle collection nommée "**category**" et insérez différentes catégories **en une seule ligne de commande** (au minimum 3).

Résultat attendu :

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660dbc5fd74a2b47eb20ab38'),
    '1': ObjectId('660dbc5fd74a2b47eb20ab39'),
    '2': ObjectId('660dbc5fd74a2b47eb20ab3a')
  }
}
```

Job 6

À l'aide d'une ligne de commande, afficher **l'ensemble des collections** de votre base de données. Si tout s'est bien passé, vous devriez avoir ce message dans le terminal :

```
category
product
```

Job 7

Récupérez l'ensemble des données de la collection "**category**".

Résultat attendu :

```
{
  _id: ObjectId('660dbc5fd74a2b47eb20ab38'),
  name: 'vetement'
}
{
  _id: ObjectId('660dbc5fd74a2b47eb20ab39'),
  name: 'chaussure'
}
{
  _id: ObjectId('660dbc5fd74a2b47eb20ab3a'),
  name: 'bébé'
}
```

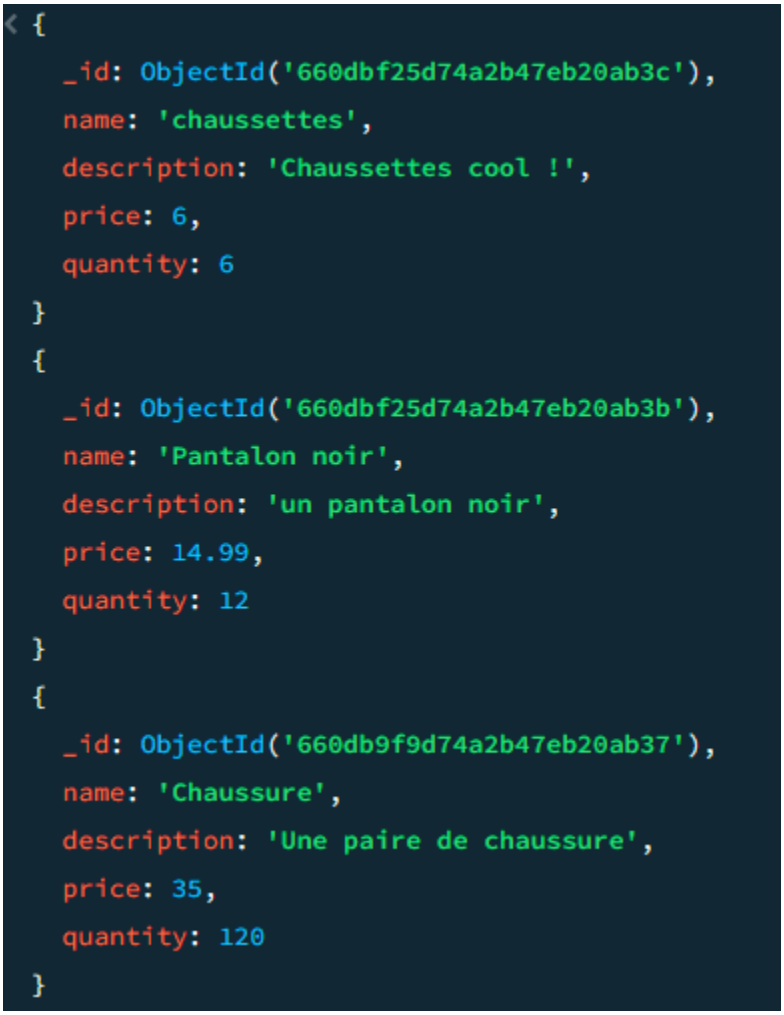
Job 8

Écrire une requête permettant d'afficher seulement deux catégories.

Job 9

Écrire une requête permettant de récupérer l'ensemble des produits classés par **prix croissant**.

Résultat attendu :



```
{
  _id: ObjectId('660dbf25d74a2b47eb20ab3c'),
  name: 'chaussettes',
  description: 'Chaussettes cool !',
  price: 6,
  quantity: 6
}
{
  _id: ObjectId('660dbf25d74a2b47eb20ab3b'),
  name: 'Pantalon noir',
  description: 'un pantalon noir',
  price: 14.99,
  quantity: 12
}
{
  _id: ObjectId('660db9f9d74a2b47eb20ab37'),
  name: 'Chaussure',
  description: 'Une paire de chaussure',
  price: 35,
  quantity: 120
}
```

Job 10

Écrire une requête permettant de récupérer l'ensemble des produits classés par **prix décroissant**.

Job 11

Écrire une requête permettant de récupérer les produits ayant un prix **supérieur à 5 euros et un stock inférieur à 100**.

Résultat attendu :

```
< {
    _id: ObjectId('660dbf25d74a2b47eb20ab3b'),
    name: 'Pantalon noir',
    description: 'un pantalon noir',
    price: 14.99,
    quantity: 12
  }
  {
    _id: ObjectId('660dbf25d74a2b47eb20ab3c'),
    name: 'chaussettes',
    description: 'Chaussettes cool !',
    price: 6,
    quantity: 6
  }
```

Job 12

Écrire une requête permettant de récupérer les produits ayant un prix **supérieur à 14.99 euros ou un stock supérieur à 100**.

Résultat attendu :

```
< {
    _id: ObjectId('660db9f9d74a2b47eb20ab37'),
    name: 'Chaussure',
    description: 'Une paire de chaussure',
    price: 35,
    quantity: 120
  }
```

Job 13

Récupérer l'ensemble des produits **sans leurs id**.

Résultat attendu :

```
{
  name: 'Chaussure',
  description: 'Une paire de chaussure',
  price: 35,
  quantity: 120
}
{
  name: 'Pantalon noir',
  description: 'un pantalon noir',
  price: 14.99,
  quantity: 12
}
{
  name: 'chaussettes',
  description: 'Chaussettes cool !',
  price: 6,
  quantity: 6
}
```

Job 14

Récupérer une catégorie à l'aide de son **id**.

Job 15

Changer le nom de la catégorie "**bébé**" en "**Vêtements pour enfant**".

Job 16

Créer une relation entre la collection “**product**” et “**category**” à l’aide d’un champ nommé “**category_id**”.
Modifier et récupérer l’ensemble des produits et leurs catégories.
Résultat attendu :

```
{
  _id: ObjectId('660db9f9d74a2b47eb20ab37'),
  name: 'Chaussure',
  description: 'Une paire de chaussure',
  price: 35,
  quantity: 120,
  category_id: ObjectId('660dbc5fd74a2b47eb20ab39'),
  category: {
    _id: ObjectId('660dbc5fd74a2b47eb20ab39'),
    name: 'chaussure'
  }
}
```

Job 17

Récupérer la catégorie qui possède le moins d'articles.

Résultat attendu :

```
{
  count: 1,
  _id: ObjectId('660dbc5fd74a2b47eb20ab39'),
  name: 'chaussure'
}
```

Compétences visées

- NoSQL

Rendu

Le projet est à rendre sur **https://github.com/prenom-nom/runtrack-nodeJs**.

Base de connaissance

- ➔ Documentation officielle
- ➔ Tout savoir sur les bases de données non relationnelles
- ➔ Requêtes en MongoDB