SDAD

24 jan 2022

NOSQL

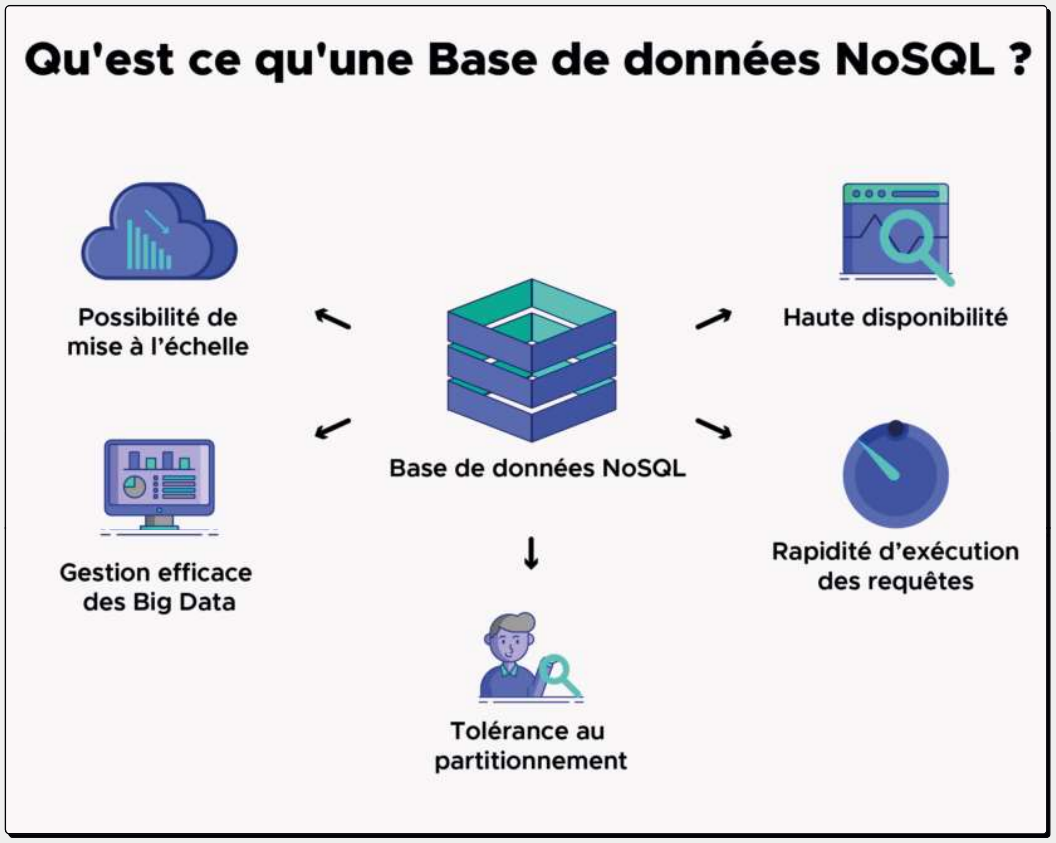
mongoDB

Plan

1. introduction NOSQL
2. définition mongodb
3. installation de mongoDB
4. Découvrez le fonctionnement de MongoDB

NOSQL (NOT ONLY SQL)

Le **NoSQL** est un type de bases de données, dont la spécificité est d'être non relationnelles. Ces systèmes permettent le stockage et l'analyse du Big Data. Ainsi, **NoSQL** est utilisé pour le Big Data et les applications web en temps réel.



NOSQL (NOT ONLY SQL)

Types de SGBD NoSQL :

1. **Orienté document (MongoDB, CouchDB)**
2. **Clé / valeur (Redis, DynamoDB, Voldemort)**
3. **Orienté colonne (Cassandra, Hbase)**
4. **Orienté graphe (Neo4J, Titan)**

NOSQL (NOT ONLY SQL)

La notion de document:

- Un **document** est la représentation d'une donnée en BSON
- BSON = Binary JSON. Extension du JSON (support officiel du type Date, ...).

Exemple d'un document:

```
{ "name" : "MongoDB",
  "type" : "Database" ,
  "info" : {
   x : 200 ,
   y : 404 }}
```

Dans le monde **NoSQL,** la **NORMALISATION** ne se pose pas.

Mongodb

définition mongodb :

MongoDB est un système de gestion de base de données **orientée documents**, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données Il est écrit en C++.

- Il gère d'énorme quantités de données.
- Le moteur de base de données facilite l'extension appelée **Scaling.**
- La quantité de données est supportée par l'accroissement et l'ajout de machines.

Installation de Mongodb

- Vous pouvez télécharger **MongoDb** à partir du site: http://www.mongodb.org/downloads
- ouvrir l'invite de commande (cmd)
- tapez **mongod** démarre le moteur de base de données
- ouvrir l'invite de commande (nouvelle fenêtre)
- tapez **mongo** est un interpréteur de commandes.

Mongodb

JSON:

- **JSON** est utilisé pour insérer et restituer des documents.
- Format **JSON**: se base sur 2 types d'éléments:

1- Soit des paires **clé/valeur**, par exemple : "**nom**": "*test*"

2- Soit des tableaux , par exemple :

"**Jours**" : **[***" Lundi", "Mardi", "mercredi", "jeudi" ,"vendredi", " samedi", "Dimanche"***]**

Mongodb

La notion de document:

- Un **serveur MongoDB** est composé de bases de données.
- Une base de données contient **des collections.**
- Chaque **collection** possède des **documents.**
- Chaque **document** possède un **identifiant** unique généré par **MongoDB**, le champ _id.

Mongodb

afficher les bases les bases de données disponibles :

- Une fois installé **mongodb** correctement, allez dans la ligne de console et tapez la commande suivante :
  - **› mongod** => pour démarrer le serveur
  - **› mongo** => sur une autre fenêtre pour démarrer la partie client

› show dbs

Mongodb

Création de la base de données :

› **use** *namedb*

=› Cette fonction (› **use namedb**) permet de créer une BDD si elle n'existe pas. Si elle est déjà existante, de se placer à l'intérieur.

Création d'une collection :

› **db.createCollection ("***nameCollection***")**

lister toutes les collections disponibles dans notre base :

› **show collections**

Mongodb

insertion d'un document JSON dans une collection :

› **db.*nameCollection*.insert({ "*nom*" :** "*name*" **, "*Age*" :** *34* **})**

afficher le contenu d'une collection :

› **db.*nameCollection*.find()**

› **db.*nameCollection*.find().pretty()**

combien de documents dans une collection :

› **db.*nameCollection*.count ()**

Mongodb

insertion d'un document dans une collection :

- **"id"** qui est un **identifiant unique** pour le **document**.
- Dans le cas contraire, si ce **"id"** n'est pas spécifié, **MongoDb** génère un unique **ObjectId** qui **identifie** le **document**.
- On peut affecter un **identifiant explicitement** dans une **collection**

› **db.*nameCollection*.insert({ _id** : 2, **"produit":** "Coca", **prix:** 10, **"enStock"** : true **})**

- Les bases de données MongoDB sont schemaless , c'est à dire que les documents ne doivent pas tous respecter le même format.

Mongodb

importer un fichier JSON dans une collection :

- En supposant que ce tableau est sauvegardé dans un fichier **file.json** , on peut l'importer dans la collection ***nameCollection*** de la base ***nameDb*** avec la commande suivante :

› **mongoimport -d** *nameDb* **-c** *nameCollection* **--file** *file.json* **--jsonArray**

Mongodb

Faire une recherche d'informations :

- la méthode **find()** nous permettra de récupérer notre document après une opération d'insertion.
- Elle n'est donc pas utilisable sur une collection qui contient beaucoup de documents.
- **Exemple :** de recherche de tous les documents ayant un attribut "**type**" ayant pour valeur "***Book"*** :

  › **db.*nameCollection*.find({** "type" **: "**Book**" });**

- Il est également possible d'**enlever les doublons** d'un attribut donné :

  › **db.*nameCollection*.distinct("title");**

Mongodb

Faire une recherche d'informations : (skip() & limit())

- Comme en SQL (étendu), les options **skip** et **limit** permettent de " **paginer** " le résultat.
- **Exemple**: › **db.*nameCollection*.find().skip(9).limit(12)**

=> La requête suivante affiche **12 documents** à partir du **dixième** inclus.

Mongodb

Faire une recherche d'informations : (sort())

- Exemple:

=> La requête suivante **trie** les **documents** sur le **titre** du film : **› db.*nameCollection*.find().sort ({** "title" **: 1 })**

=> ascendante (valeur **1**) ou **descendante** (valeur **1**).

Mongodb

Faire une recherche d'informations : (findOne())

- **Exemple**:

=> Une requête sur **l'identifiant** ramène (au plus) un seul document. Dans un tel cas, on peut utiliser **findOne**

› **db.*nameCollection*.findOne ({"_id": "id1"})**

Mongodb

Faire une recherche d'informations :

- **Exemple**:

=> Tous les films dont le titre commence par **"Re"**

› **db.*nameCollection*.find({** "title" **: "/^**Re**/" })**

Mongodb

Faire une recherche d'informations :

(opérateur de comparaison)

- **$gt** : plus grand que
- **$lt** : plus petit que
- **$gte** : plus grand ou égal à
- **$lte** : plus petit ou égal à
- **Exemple :**

› **db.*nameCollection*.find ({** "year"**: { $gte :** 2000, **$lte :** 2005 **}})**

Mongodb

Faire une recherche d'informations :

(opérateur ensemblistes)

- **$in , $and , $or, $all, $exist , $type** et **$regex**
- **Exemple :**
  - **db.nameCollection.find({ "num": {$in: [**500, 600, 700**] }})**

Mongodb

Faire une mise à jour : (update())

- **Exemple :**

  › **db.nameCollection.update({ "_id": *"id1"* },**

  { $set : { *"title":"update"* }})

Faire une mise à jour d'un tableau :

- On va ajouter un tableau à un document
  › **db.nameCollection.insert ({"cmpt":**101**}, tab : ['a','b','c'] })**
- Maintenant pour ajouter un élément, on utilise l'opérande **$push**
  › **db.produits.update ({"cmpt":**101**}, {$push : {tab : 'd'}})**

=› le nouveau tableau **tab** contient**: ['a', 'b', 'c', 'd']**

Mongodb

Suppression d'une propriété :

- Exemple :

=> Nous allons supprimer la propriété que l'on vient d'ajouter.

Pour cela, nous allons utiliser **$unset**

› **db.nameCollection.update({ "_id": "id1"} , { $unset : { "title":"update" }})**

Mongodb

Suppression d'une collection:

=> On peut supprimer une collection :

› **db.*nameCollection*.drop()**

Mongodb

Suppression d'une base de donnée :

=> On peut supprimer une base de donnée:

- **use** *dbname*
- ***db.dropDatabase()***