Abdullah ÇAY

MongoDB - Introduction to MongoDB

30 minutes Easy

DataScientest • com

NoSQL databases (MongoDB)

NoSQL databases (MongoDB)

1. Introduction to MongoDB

a) What is NoSQL?

Relational databases have been used since the 1970s. They are widely used and well documented and are very useful to represent relationships between different entities. However, they require a strict schema. Nowadays, developers have storage systems that are simpler to use.

With Big Data, we are faced with a multitude **of different types of data**. For example, if we want to store the activity of a user on a website. This website will evolve, have new pages, new features. If we want to track the use of these new features, we may need to update the schema of our database.

Recently, the focus has been on **NoSQL databases** which allow for greater flexibility on the data schema and thus manage data with a wide range of formats.

> Originally, NoSQL means **No SQL**, which means, database management systems that are not based on the relational paradigm. It has been **Not Only SQL** (not just SQL) as the boundary between the two families has become blurrier.

NoSQL DBMSs are made up of a wide range of technologies, techniques and paradigms, but there are three main families:

- **Document-oriented databases**
- **Column-oriented databases**
- **Graph-oriented databases**

Below, we will detail what a document-oriented database is.

b) Document-oriented databases

In a relational database, data is stored in rows and columns, forming a table, following a strict schema.

```
{"menu": {
    "sandwichs": [
        {"name": "Melthouse","bread":"Wheat","sauce": "Ceasar"},
        {"name": "Italian BMT", "extras": ["pickles","onions","lettuce"],"sauce":["Chipotle","Aioli"]},
        {"name": "Steakhouse Melt","bread":"Parmesan Oregano"}
}}
```

It would be complex to store them in a relational database. Indeed, there is no strict schema and we notice that our data (here sandwiches) have different attributes.

However, this is not the case with document-oriented databases where we will store information in **documents**. A document is a set of key-values. That is why these databases are adequate to process our semi-structured data.

These documents are then organized into what we call **collections**. In this way, we make sure that similar documents are grouped together. Finally, we put our collections into **databases**. This way, we can organize our data.

The following tools provide document-oriented databases:

- **Couchbase**
- **CouchDB**
- **MongoDB**

We will see in particular MongoDB which is the reference in this field.

c) Installation

We will use Docker to handle MongoDB in order to avoid wasting time during the installation.

> Write the following lines in a `docker-compose.yml` file:

```
version: "3.3"
services:
  mongodb:
    image : mongo:5.0
    container_name: my_mongo
    volumes:
    - ./sample_training:/data/db
    ports:
    - 27017:27017

```

> It is possible to associate a user to our MongoDB server. To do this, we can use this file instead.

> Run the following command:

```
docker-compose up -d
```

With the `docker ps` command, we can verify that our `MongoDB` container is operational. Now, let's switch to the environment with Mongo using the following command:

```
docker exec -it my_mongo bash
```

We are in a different environment from our personal machine, we can write `mongo` and press the tab key to see that several commands dedicated to Mongo are available whereas they were not there initially.

```
root@47359760cda5:/# mongo
mongod        mongodump     mongoexport   mongofiles    mongoimport   mongorestore  mongos        mongosh
mongostat     mongotop
```

We will see how to manipulate data on MongoDB in the next chapter. To exit the MongoDB environment, simply issue the `exit` command.

Validate

Machine status

Ubuntu Server 18.04 LTS

SSD Volume Type

64-bit x86

Stopped

Connect Reset Start