MongoDB with Python

Jay Chow

Software Engineer, Cisco eCommerce, IT

Cisco, San Jose

12th March 2020

Agenda

1. What is MongoDB?
2. Differences between MongoDB and RDBMS
3. Features of MongoDB
4. When do we use MongoDB?
5. Python with MongoDB
6. Example code
7. Querying and Printing documents
8. References

What is MongoDB?

- **MongoDB**, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

- SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. **Modern applications are more networked, social and interactive than ever**. Applications are storing more and more data and are accessing it at higher rates.

What is MongoDB? (cont'd)

- Relational Database Management System(RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance.

- MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

What is MongoDB?(cont'd)

- MongoDB is a cross-platform, document-oriented database that works on the concept of collections and documents.

- MongoDB offers high speed, high availability, and high scalability.

- Reasons to opt for MongoDB :

1) It supports hierarchical data structure
2) It supports maps like Dictionaries in Python.
3) Built-in Python drivers to connect python-application with Database(i.e PyMongo)
4) It is designed for Big Data.
5) Deployment of MongoDB is very easy.

Differences between MongoDB and RDBMS

MongoDB vs RDBMS

RDBMS	MongoDB
Table	Collection
Row	JSON Document
Index	Index
Join	Embedding & Linking
Partition	Shard

Features of MongoDB

- Document Oriented: MongoDB stores the main subject in the minimal number of documents and not by breaking it up into multiple relational structures like RDBMS. For example, it stores all the information of a computer in a single document called Computer and not in distinct relational structures like CPU, RAM, Hard disk, etc.
- Indexing: Without indexing, a database would have to scan every document of a collection to select those that match the query which would be inefficient. So, for efficient searching Indexing is a must and MongoDB uses it to process huge volumes of data in very less time.
- Scalability: MongoDB scales horizontally using sharding (partitioning data across various servers). Data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that resides across many physical servers. Also, new machines can be added to a running database.
- Replication and High Availability: MongoDB increases the data availability with multiple copies of data on different servers. By providing redundancy, it protects the database from hardware failures. If one server goes down, the data can be retrieved easily from other active servers which also had the data stored on them.
- Aggregation: Aggregation operations process data records and return the computed results. It is similar to the GROUPBY clause in SQL. A few aggregation expressions are sum, avg, min, max, etc

When do we use MongoDB?

MongoDB is preferred over RDBMS in the following scenarios:

- Big Data: If you have huge amount of data to be stored in tables, think oshardingf MongoDB before RDBMS databases. MongoDB has built in solution for partitioning and your database.
- Unstable Schema: Adding a new column in RDBMS is hard whereas MongoDB is schema-less. Adding a new field, does not effect old documents and will be very easy.
- Distributed data Since multiple copies of data are stored across different servers, recovery of data is instant and safe even if there is a hardware failure.
- MongoDB currently provides official driver support for all popular programming languages like C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang.

Python with MongoDB (with simple example)

```
# importing module
from pymongo import MongoClient

# creation of MongoClient
client=MongoClient()

# Connect with the portnumber and host
client = MongoClient("mongodb://localhost:27017/")

# Access database
mydatabase = client['name_of_the_database']

# Access collection of the database
mycollection=mydatabase['myTable']

# dictionary to be added in the database
rec={
title: 'MongoDB and Python',
description: 'MongoDB is no SQL database',
tags: ['mongodb', 'database', 'NoSQL'],
viewers: 104
}

# inserting the data in the database
rec = mydatabase.myTable.insert(record)
```

The following command is used to connect the MongoClient on the localhost which runs on port number 27017.

Querying in MongoDB:

6. **Querying in MongoDB :** There are certain query functions which are used to filer the data in the database. The two most commonly used functions are:
   1. find()
      find() is used to get more than one single document as a result of query.

```
for i in mydatabase.myTable.find({title: 'MongoDB and Python'})
    print(i)
```

      This will output all the documents in the myTable of mydatabase whose title is 'MongoDB and Python'.

   2. **count()**
      count() is used to get the numbers of documents with the name as passed int he parameters.

```
print(mydatabase.myTable.count({title: 'MongoDB and Python'}))
```

      This will output the numbers of documents in the myTable of mydatabase whose title is 'MongoDB and Python'.

Print all documents:

3. **To print all the documents/entries inside 'myTable' of database 'mydatabase' :** Use the following code:

```
from pymongo import MongoClient

try:
    conn = MongoClient()
    print("Connected successfully!!!")
except:
    print("Could not connect to MongoDB")

# database name: mydatabase
db = conn.mydatabase

# Created or Switched to collection names: myTable
collection = db.myTable

# To find() all the entries inside collection name 'myTable'
cursor = collection.find()
for record in cursor:
    print(record)
```

References

https://www.geeksforgeeks.org/mongodb-and-python/