MongoDB integration with Python and Node.js, Express.js

Conference Paper · March 2024

DOI: 10.1109/ICAECT60202.2024.10460546

7 authors, including:

Santi KUMARI Behera
Veer Surendra Sai University of Technology
113 PUBLICATIONS **1,906** CITATIONS

SEE PROFILE

Prabira kumar Sethy
Guru Ghasidas Vishwavidyalaya
193 PUBLICATIONS **3,506** CITATIONS

SEE PROFILE

2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) | 979-8-3503-4367-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAECT60202.2024.10469546

MongoDB integration with Python and Node.js, Express.js

AVSS Somasundar
Department of Information Technology,
Sagi Rama Krishnam Raju (S.R.K.R) Engineering College (A),
Bhimavaram, A. P, 534204, India.
somasundar@srkrec.ac.in

M Chilakarao
Department of Information Technology,
Sagi Rama Krishnam Raju (S.R.K.R) Engineering College (A),
Bhimavaram, A. P, 534204, India
chilakarao@gmail.com

BHVS Rama Krishnam Raju
Department of Information Technology,
Sagi Rama Krishnam Raju (S.R.K.R) Engineering College (A),
Bhimavaram, A. P, 534204, India.

Santi Kumari Behera, SM-IEEE
Department of CSE
VSSUT Burla
Odisha, India
b.santibehera@gmail.com

Ch Venkata Ramana
Department of CSE,
Shri Vishnu Engineering College for Women (A), Bhimavaram, A. P, India
chvramanacse@svecw.edu.in

Prabira Kumar Sethy
Department of ECE
Guru Ghasidas Vishwavidyalaya,
Bilaspur, C.G.
prabirsethy.05@gmail.com

***Abstract*-This scholarly article presents a detailed examination of the seamless integration of MongoDB, a well-established NoSQL database, with two widely used programming languages—Python and Node.js. MongoDB's inherent adaptability, extensive scalability, and focus on document-centricity position it as a premier choice for contemporary web and application development. Python and Node.js, recognized for their versatility and comprehensive libraries, serve as robust frameworks for interacting with MongoDB. The study delves into the intricate process of integrating MongoDB with Python and Node.js, meticulously analyzing the distinct drivers and Application Programming Interfaces (APIs) associated with each language. The document also highlights a comprehensive analysis of the PyMongo Driver, a crucial component facilitating the seamless integration of MongoDB with the Python programming language. Furthermore, the article duly acknowledges the advantages of incorporating the Mongoose driver in the integration of MongoDB with Node.js and Express.js. Overall, this scholarly inquiry provides valuable insights into the integration process, emphasizing the significance of language-specific drivers and APIs, and offering a holistic perspective on the advantages and considerations associated with MongoDB integration with Python and Node.js.**

Keywords: MongoDB; PyMongoose; NoSQL Integration

1. INTRODUCTION

The introduction of this scholarly review highlights MongoDB as a widely used NoSQL database renowned for its document-oriented architecture, particularly suited for unstructured and semi-structured data that may pose challenges for traditional relational database management systems. MongoDB stands out as a versatile and ubiquitous solution in web applications and enterprise frameworks.

This review paper provides an extensive exploration of MongoDB, focusing on its seamless integration with the MEAN (MongoDB, ExpressJS, AngularJS, NodeJS) stack and its harmonious integration with the popular programming language, Python.

MongoDB, introduced in 2009 as a cost-free and unencumbered document-centric database system, operates by leveraging JSON-esque documents with discretionary schemas. Its primary goal is to deliver exceptional performance, scalability, and availability.

The database system employs collections to store data, akin to tables in traditional relational databases. However, it's crucial to note that MongoDB collections do not impose strict schemas, allowing for diverse structures within the documents they contain. MongoDB's comprehensive support for various data types, including strings, integers, floating-point numbers, dates, and binary data, further enhances its versatility and applicability in diverse data scenarios.

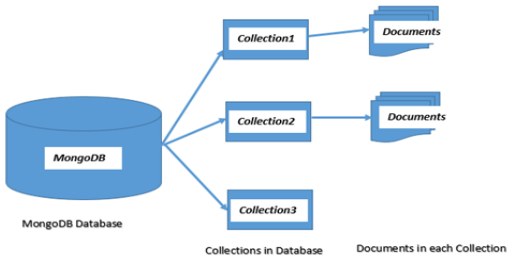

Fig. 1. MongoDB Basic Data Storage

The document holds Key and Value Pairs. The sample documents are as follows:

```
{
  "_id": ObjectId("61751d2d11c34a12b88e60a1"),
  "name": "Ruthvik",
  "email": "ruthvika@xyz.com",
  "age": 25,
  "address": {
    "street": "Srirampuram",
```

```
      "city": "Bhimavaram",
      "state": "AP",
      "zipcode": 534204"
   },
   "phone_numbers": [
      {"type": "home", "number": "9876678912"},
      {"type": "work", "number": "9678443256"}
   ]
}
```

The **_id** field is a unique identifier for the document that is automatically generated by MongoDB when the document is inserted. The names, emails, and age fields were simple. The address field is a sub-document or embedded that contains a person's street, city, state, and zip code. The phone number field is an array of subdocuments that contain a person's phone numbers.

Collection is akin to a Table in RDBMS while Document is akin to a Row/Record in a Table. MongoDB also provides advanced features, such as horizontal scaling, replication, and automatic sharding. These features allow MongoDB to handle large volumes of data and provide a high availability and fault tolerance.

MongoDB is suitable as a data server for analytical purposes. Products from vendors, which are used for Data Analytics, use MongoDB to supply data to models and dashboards.

II. LITERATURE REVIEW

[1] Found that most of the research focused on MongoDB's performance and scalability and that the majority of the studies used experimental and benchmarking methods.[2] Focused on the use of MongoDB in healthcare applications, MongoDB is suitable for storing and processing healthcare data owing to its scalability, flexible data model, and support for geospatial queries. [3] analyzed security vulnerabilities in MongoDB and their impact on data privacy and security. MongoDB is susceptible to various security threats such as injection attacks, unauthorized access, and data leakage. They recommended several measures to enhance the security of MongoDB deployments such as proper authentication and authorization, network security, and software patching. [4] provides an overview of NoSQL databases for big data and compares the performance of MongoDB with that of other databases. MongoDB is suitable for big data applications owing to its scalability, data modeling, and query processing capabilities. They also found that MongoDB outperformed other NoSQL databases such as Cassandra and HBase in terms of query execution time and data ingestion rate. It is summarized in [5] their findings on MongoDB's architecture, data modeling, query processing, performance, and security. [6] discussed MongoDB's suitability for big data, its data modeling, indexing, and query processing, as well as its performance and scalability. They also identified challenges, such as data. [7] emphasized applications in various domains such as healthcare, e-commerce, and social media. Security measures that can be taken to protect MongoDB deployments are recommended in [8]. They recommended several best practices, such as secure authentication, access control, and encryption, to enhance MongoDB security.

III. ANALYSIS OF INDEXING IN MONGODB

MongoDB utilizes advanced data structures, specifically B-trees and hash tables, for its indexing mechanism. These indexes play a crucial role in accelerating document retrieval by utilizing indexed fields, eliminating the need to scan the entire collection. Consider a scenario of managing a collection of commodities with attributes such as _id, appellation, and monetary valuation. When creating an index for the "name" field (product name), MongoDB employs a B-tree data structure. This B-tree stores distinct "name" values along with references to their respective documents, potentially utilizing the _id field as a reference point.

A B-tree, or Balanced Tree, is a self-balancing tree data structure commonly used in database management and file systems. It maintains ordered data, providing efficient operations for insertion, deletion, and search. B-trees are characterized by consistent leaf node positioning at an equivalent level, ensuring structural balance. Each node in the structure contains a range of keys and pointers to subordinate nodes, with keys arranged in a sequential manner.

In computer science, the B-tree is fundamental, possessing a well-defined structure for efficient data storage and retrieval. Each B-tree node can accommodate a specific number of keys, denoted by the symbol "t," with a minimum of t-1 keys and a maximum of 2t-1 keys. The number of child pointers in a node exceeds the number of keys by one, resulting in k+1 child pointers for a node containing k keys.

Creating a B-tree index on a MongoDB collection field involves meticulous construction to enhance data retrieval and querying efficiency. The B-tree maintains balance during key insertion and removal, optimizing the tree structure to improve both search and modification operations. The limited number of keys in each node contributes to the overall shallowness of the tree, facilitating expedited search operations.

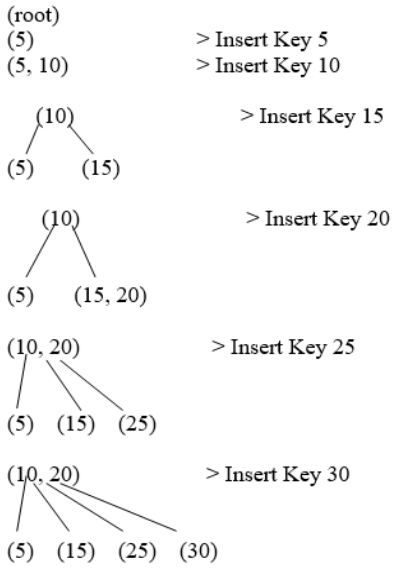

Fig. 2. Example of a B-tree with a maximum of 2 keys per node (2-3 B-tree).

When embarking on the journey to locate a key within a B-tree, the search initiates at the root node. Traversing down the branches of the tree, one meticulously examines the encountered keys, determining the appropriate path to follow. This iterative process continues until either the desired key is found or a terminal node without the key is reached. This efficient tree-based framework significantly enhances database indexing and file system management, ensuring the swift and predictable execution of data retrieval and modification operations. Figure 3 visually illustrates the concept of database indexing.

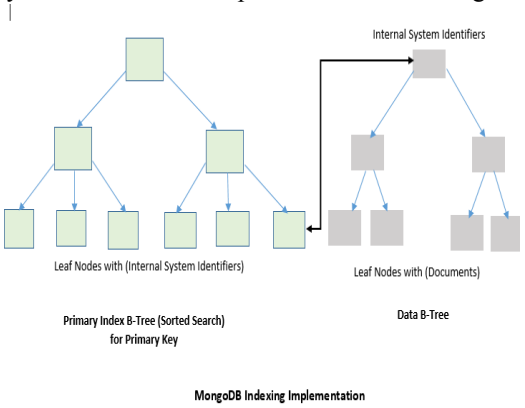

Fig 3. MongoDB Indexing Implementation

Here, the Internal System Identifier generates a 16-digit hexadecimal number and the External Identifier has a custom number. Thus, one may be chosen.

IV. INTEGRATION WITH PYTHON AND EXPRESS.JS

A key advantage of using Python with MongoDB is the seamless accommodation of both synchronous and asynchronous programming paradigms by the driver. This built-in flexibility allows developers to choose the most suitable methodology that aligns with the specific requirements of their application.
Express.js, a well-regarded web framework for Node.js, provides a streamlined and minimalist approach to building web applications. With its highly adaptable and modular framework, Express.js empowers developers to effortlessly integrate it with various technologies, with MongoDB being a notable example.

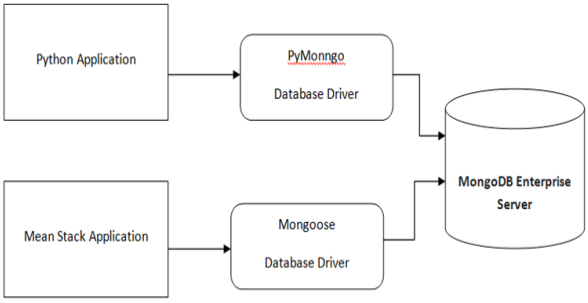

Fig 4. MongoDB integration Architecture.

Numerous libraries and frameworks facilitate the smooth integration of MongoDB with both the Python programming language and the Express.js web application framework. I'll now enumerate several widely recognized methodologies for your understanding.

- PyMongo: This prominent software tool serves as the designated MongoDB driver for Python. PyMongo provides a Pythonic Application Programming Interface (API) for interacting with MongoDB, accommodating both synchronous and asynchronous programming paradigms.
- Motor: Representing an asynchronous Python driver, Motor facilitates seamless interaction with MongoDB. Leveraging Python's asyncio library, this framework enables the execution of non-blocking I/O operations, enhancing overall application performance.
- Mongoose: A widely acclaimed object modeling library for Node.js, Mongoose has gained popularity in the MongoDB realm. It offers a straightforward and adaptable API for interacting with MongoDB, seamlessly integrable into the Express.js framework.
- Mongo Engine: This widely acclaimed MongoDB object modeling library is designed for the Python programming language. Mongo Engine provides a robust and versatile API for seamless interaction with MongoDB, incorporating advanced functionalities such as schema validation and indexing.

A notable advantage of using an object modeling library like Mongoose or Mongo Engine lies in its capacity to streamline the complexities associated with MongoDB utilization. This is achieved by providing a more sophisticated conceptual framework for data modeling and query construction.

V. USE OF AGGREGATION AND INTEGRATION

The integration of the Mongoose library played a pivotal role in establishing a connection between the MongoDB Database and the trio of node.js, express.js, and angular.js. Leveraging the Mongolian library, modules are available for seamlessly incorporating Aggregation and MapReduce techniques. By utilizing the application programming interfaces (APIs) offered by these modules, we can compose code within the Visual Studio code environment. This facilitates the development of Node.js programs, which can then be employed for integration and testing endeavors.

- **Aggregations:** Aggregation operations process data records and return computed results. Aggregation operations group values from multiple documents and can perform a variety of operations on the grouped data to return a single result.

- **MapReduce:** MongoDB provides map-reduce operations for aggregation. In general, map-reduce operations have two phases: a map stage that processes each document and emits one or more objects for each input document, and a reduced phase that combines the output of the map operation. Optionally, MapReduce can have a finalized stage to make final modifications to the results. Like other aggregation operations, MapReduce can specify a query condition to select the input documents, as well as sort and limit the results.

VI. THE ISSUE FACED IN INTEGRATION

The PyMongo API documentation for PyMongo was available with the driver source. The commands for CRUD that are used in MongoDB Shell like "insertOne," "updateOne," "insertMany, etc." should not be used as they are in Python language and the Python API is a bit different. Pymongo uses insert_one, insert_many, and update_one to perform these operations. This is illustrated in the following example:

```
import pymongo
url='mongodb://localhost:27017/'
client=pymongo.MongoClient(url)
db1=client['sports']
col1=db1['col11']
doc1={'name':'cricket','level':'international','no':11}
col1.insertOne(doc1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2936\939103756.py in <module>
      5 col1=db1['col11']
      6 doc1={'name':'cricket','level':'international','no':11}
----> 7 col1.insertOne(doc1)

~\AppData\Roaming\Python\Python39\site-packages\pymongo\collection.py in __call__(self, *args, **kwargs)
   3211                 exists.  % self.__name
   3212             )
-> 3213         raise TypeError(
   3214             "'Collection' object is not callable. If you meant to "
   3215             "call the '%s' method on a 'Collection' object it is "

TypeError: 'Collection' object is not callable. If you meant to call the 'insertOne' method on a 'Collection' object it is fail
ing because no such method exists.
```

Figure 5. Python Open source in PyMongo for API information (Collection Methods)

We wrote a program in Jupyter to create a' sports' database, 'col1' collection in MongoDB, as shown above. We then attempted to insert a document using the insertOne method. This shows an error, as above. To check the correct method names, open the collection.py file shown in the highlighted black box, and then it is found that the correct method name is insert_one. The results were obtained after changing the insert command. The correct method signature of CRUD with the PyMongo driver may also be found in the API documentation available for PyMongo.

Mongoose: Mongoose is a driver that connects Express.js with MongoDB. Upto Mongoose 6.0 call back functions in res, req methods of express modules are allowed. From Mongoose 7.0, support for call back functions was removed. Therefore, If a development project code is to be written with callback functions, we have to install Mongoose 6.0.
To install a version-specific Mongoose instead of the latest version the command is,
Npm install mongoose@6.0

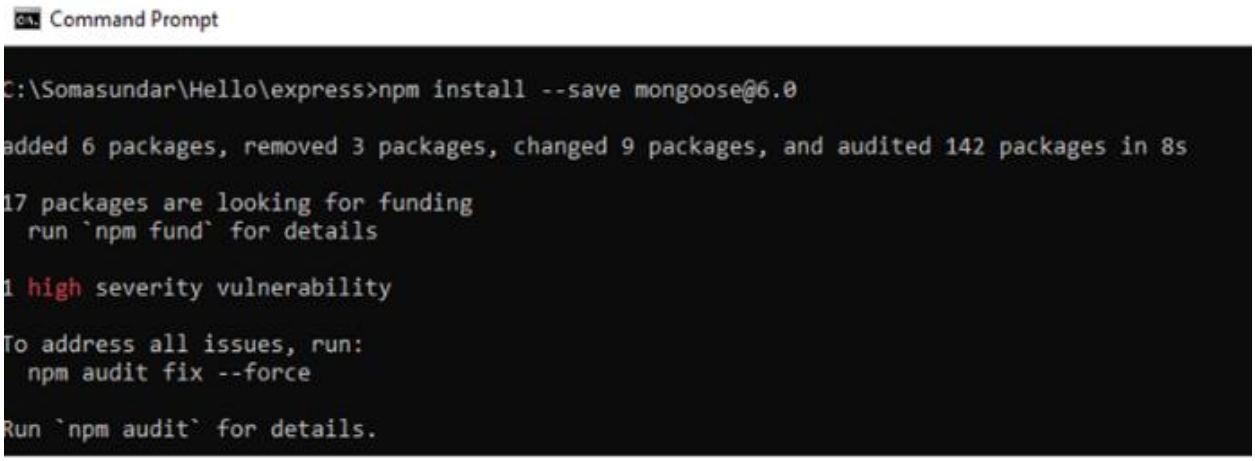

Figure 6. Installing specific version Mongoose driver using @ operator with Node Package Manager(npm).

VII. CONCLUSION

Integrating MongoDB with web frameworks such as Python and Express.js can help developers to build powerful and scalable web applications. Several libraries and frameworks are available that make it easy to integrate MongoDB with Python and Express.js. The PyMongo and Motor libraries provide a powerful and flexible interface for working with MongoDB in Python, whereas the Mongoose and MongoEngine libraries provide a higher-level abstraction for data modeling and query building in Node.js. The use of these libraries can help simplify the process of working with MongoDB and improve the performance of the web applications. Before Integration Version Compatibility should be checked, and programmers must refer to version-specific APIs for Integration.

REFERENCES

[1] Ahmed, S. & Qureshi, M. U. (2021). A systematic literature review on MongoDB. International Journal of Advanced Computer Science and Applications 133(144).

[2] Altwaijri, H., et al. (2020). A Review of MongoDB and Its Applications in Healthcare. Journal of Healthcare Engineering, 26(36).

[3] Kiran, P. S., & Vijayakumar, V. (2018). A review of security vulnerabilities in MongoDB. Journal of Computer Virology and Hacking Techniques, 149(154).

[4] Moshiri, B., & Shojafar, M. (2018). A systematic literature review on NoSQL databases for big data. Journal of Big Data, 189(211).

[5] Kumar, S., & Tarar, S. (2019). A systematic review of MongoDB: A NoSQL Database. International Journal of Computer Applications, 19(25).

[6] Huang, J., et al. (2018). A Review of MongoDB in Big Data. International Journal of Advanced Computer Science and Applications, 241(247).

[7] Ramezani, A., et al. (2017). A review of MongoDB databases and their applications. Journal of Advanced Research in Dynamical and Control Systems, 322(329).

[8] Kokate, M. A., & Patil, P. A. (2016). A Review on Security Issues in MongoDB NoSQL Database. International Journal of Computer Applications, 27(30).

[9] Chouhan, A. (2019). A Review on Various Aspects of MongoDB Databases. International Journal of Engineering Research & Technology (IJERT), 8(05).

[10] Krishnan, H., et al. (2016). MongoDB – a comparison with NoSQL databases. International Journal of Scientific and Engineering Research, 1035(1037).