JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

LEVERAGING MONGODB: A COMPREHENSIVE STUDY ON DATABASE MANAGEMENT

Neha Lidoriya[1] , Jagriti Chand[2]

Department of Computer Science and Engineering
Barkatullah University,University Institute of Technology, Bhopal,M.P.

***Abstract*—**An open-source document-oriented database called MongoDB is designed to hold a lot of data and facilitate effective manipulation of that data. Because data in MongoDB is not stored and retrieved as tables, it falls under the category of NoSQL (Not Just SQL) databases.

Keywords—Leverage, MongoDB, Fields, documents

I. INTRODUCTION

In 2007, the American software firm 10gen started working on MongoDB as part of an intended platform-as-a-service offering. The business switched to an open-source development approach in 2009 and started providing services including commercial support. The name 10gen was changed to MongoDB Inc. in 2013.An open-source document-oriented database is called MongoDB. Because data in MongoDB is not stored and retrieved as tables, it falls under the category of NoSQL (Not Just SQL) databases. After reviewing the overall overview of MongoDB, let's learn how to install it on Windows.MongoDB may be installed via MSI. Let's look at a step-by-step tutorial on using MSI to install MongoDB on Windows. One well-known open-source NoSQL database software is MongoDB. Because it is a document-oriented database, information is kept as adaptable documents that resemble JSON. Because of its ease of use, scalability, and flexibility, MongoDB is a well-liked option for a variety of applications.

Here's a brief overview of some key aspects of MongoDB:

1. **Document-Oriented:** MongoDB stores data in documents, which are JSON-like objects composed of key-value pairs. These documents can have nested structures, allowing for flexible and hierarchical data modeling. This document-oriented approach makes it easier to represent complex data structures compared to traditional relational databases.
2. **Schema less:** Unlike relational databases, MongoDB does not require a predefined schema for storing data. Each document in a collection can have its own unique structure, and fields can be added or removed dynamically as needed. This flexibility simplifies the development process, especially in environments where the data schema evolves over time.
3. **Scalability:** MongoDB is designed to scale horizontally across multiple servers, allowing it to handle large volumes of data and high traffic loads. It supports sharding, which involves partitioning data across multiple servers (shards) to distribute the workload and increase performance. MongoDB's architecture also includes features like replica sets for high availability and data redundancy.
4. **Query Language and Indexing:** MongoDB uses a rich query language that supports a wide range of operations for querying and manipulating data. It supports queries based on document fields, as well as advanced features like aggregation pipelines for complex data processing. MongoDB also provides indexing capabilities to optimize query performance, including support for compound indexes and text search.
5. **High Availability and Fault Tolerance:** MongoDB provides features for ensuring high availability and fault tolerance of data. Replica sets allow data to be replicated across multiple nodes, providing automatic failover and data redundancy in case of node failures. Additionally, MongoDB Atlas, the managed cloud service for MongoDB, offers built-in backup and disaster recovery capabilities.

MongoDB is widely used in various industries and applications, including web and mobile applications, content management systems, real-time analytics, and IoT (Internet of Things) platforms. Its flexibility, scalability, and ease of use make it a versatile choice for developers and organizations seeking a modern database solution.

II. EASE OF USE

A. *Selecting a Template*

MongoDB is known for its ease of use and developer-friendly features. Here are some aspects that contribute to its simplicity:

1. **Document-Oriented Model:** MongoDB's document-oriented

data model, based on JSON-like documents, is intuitive and flexible. Developers can store data in a format that closely resembles the objects used in their application code, simplifying the mapping between application objects and database documents.

2. **Dynamic Schema:** MongoDB does not require a predefined schema, allowing developers to insert documents into collections without needing to define the structure beforehand. This flexibility is particularly useful during the development phase when the data model is still evolving.
3. **Query Language:** MongoDB's query language is expressive and easy to use. It supports a wide range of operations for querying, updating, and manipulating data. Queries are written in a JSON-like syntax, making them straightforward to understand and write.
4. **Indexing:** MongoDB supports indexing to improve query performance. Developers can create indexes on fields to speed up common queries. MongoDB's automatic index management feature can also simplify the process of optimizing query performance.
5. **Aggregation Pipeline:** MongoDB's aggregation pipeline allows developers to perform complex data processing and analysis operations. It provides a framework for composing a series of data processing stages, such as filtering, grouping, and projecting, to transform and aggregate data.
6. **Drivers and Libraries:** MongoDB provides official drivers for a variety of programming languages, including Python, JavaScript (Node.js), Java, and others. These drivers abstract away the complexity of interacting with the database, providing simple APIs for performing CRUD (Create, Read, Update, Delete) operations.
7. **Documentation and Community:** MongoDB offers comprehensive documentation and resources for developers, including tutorials, guides, and examples. The MongoDB community is active and supportive, with forums, user groups, and online communities where developers can seek help and share knowledge.
8. **MongoDB Atlas:** MongoDB Atlas is a fully managed cloud database service that simplifies the deployment and management of MongoDB databases. It offers features such as automated backups, monitoring, and scaling, allowing developers to focus on building their applications without worrying about infrastructure management.

Overall, MongoDB is a great option for a variety of applications, from small projects to large-scale business installations, because to its emphasis on simplicity, flexibility, and developer productivity. Make sure you have the right template for the size of your paper first. This template is designed to print on A4-sized paper. Please dismiss this file and download the Microsoft Word, Letter file if you are using US letter-sized paper.

PREPARE YOUR PAPER BEFORE STYLING

A. Abbreviations and Acronyms

Here are some common abbreviations and acronyms related to MongoDB:

1. **MongoDB:** The name of the database itself, often abbreviated as "MDB" or simply "Mongo".
2. **CRUD:** Stands for Create, Read, Update, Delete - the basic operations performed on data in a database. MongoDB is well-known for its support of CRUD operations.
3. **BSON:** Binary JSON. BSON is the binary-encoded format used by MongoDB to store data in a more compact and efficient manner compared to JSON.
4. **API:** Application Programming Interface. MongoDB provides APIs for various programming languages (e.g., pymongo for Python, mongo-java-driver for Java) to interact with the database.
5. **CLI:** Command Line Interface. MongoDB offers a command-line tool called mongo or mongosh for interacting with databases from the command line.
6. **ODM:** Object-Document Mapper. An ODM is a programming library that abstracts away the details of working with MongoDB documents, providing an interface similar to an Object-Relational Mapper (ORM) for relational databases.
7. **ORM:** Object-Relational Mapper. While not specific to MongoDB, ORMs are commonly used in the context of relational databases. However, similar concepts are applied to MongoDB with the term "ODM" (Object-Document Mapper).
8. **RS:** Replica Set. A group of MongoDB servers that maintain the same data set, providing redundancy and high availability.
9. **Shard:** A subset of data in a MongoDB cluster. Sharding is a method for distributing data across multiple machines to improve scalability and performance.
10. **MMS:** MongoDB Management Service. Now known as MongoDB Cloud Manager, it provides monitoring, backup, and automation features for MongoDB deployments.
11. **CS:** Config Server. In a sharded MongoDB cluster, config servers store metadata and configuration settings.

MQL: MongoDB Query Language. The query language used to interact with MongoDB databases.

These are just a few examples, but MongoDB has a rich ecosystem with various tools and concepts, each with its own set of abbreviations and acronyms.

Units

In MongoDB, data is organized into several logical units:

1. **Database:** A database in MongoDB is a container for collections. It is the highest-level container for data storage. Each database has its own set of collections and permissions. You can think of a database as being analogous to a relational database.
2. **Collection:** A collection is a group of MongoDB documents. It is roughly equivalent to a table in a relational database. Collections do not enforce a schema, meaning that the documents within a collection can have different fields and structures.
3. **Document:** A document is a set of key-value pairs. It is the basic unit of data in MongoDB and corresponds to a row in a relational database. Documents are stored in collections and are represented using BSON (Binary JSON), which is a binary-encoded format similar to JSON.
4. **Field:** A field is a key-value pair within a document. Each document can have multiple fields, each with its own value. Fields can contain various types of data, including strings, numbers, arrays, and nested documents.

5. **Index:** An index is a data structure that improves the speed of data retrieval operations on a collection. Indexes can be created on one or more fields within a collection and can significantly improve query performance, especially for frequently queried fields.
6. **GridFS:**GridFS is a specification for storing and retrieving large files, such as images, videos, and audio files, in MongoDB. It allows you to store files that exceed the BSON document size limit of 16 megabytes by dividing them into smaller chunks and storing each chunk as a separate document.

These are the primary units of MongoDB, each serving a specific purpose in storing and organizing data within the database. Understanding these units is essential for effectively designing and working with MongoDB databases and collections.

- Leads to confusion because equations do not balancedimensionally. If you must use mixed units, clearlystate the units for each quantity that you use in anequation.
- Do not mix complete spellings and abbreviations ofunits: “Wb/m2” or “webers per square meter”, not“webers/m2”.Spell out units when they appear intext: “. . . a few henries”, not “. . . a few H”.
- Use a zero before decimal points: “0.25”, not “.25”.Use “cm3”, not “cc”. (*bullet list*)

B. Equations

MongoDB is a NoSQL (Not Only SQL) database, so it doesn't have a strict "equation" in the way that a mathematical function does. However, we can think of MongoDB in terms of its components and operations:

1. **Data Model:** MongoDB stores data in flexible, JSON-like documents called BSON (Binary JSON). Documents are organized into collections, which are similar to tables in relational databases. Each document can have its own structure, allowing for a dynamic schema.
2. **Query Language:** MongoDB provides a rich query language for retrieving and manipulating data. The query language includes operators for filtering, sorting, updating, and aggregating data. Queries are written in a JSON-like syntax and can be composed using various operators and expressions.
3. **Indexes:** MongoDB supports indexes to optimize query performance. Indexes can be created on one or more fields within a collection, allowing for efficient retrieval of data based on specific criteria. MongoDB uses B-tree and hashed indexes to speed up query execution.
4. **Aggregation Pipeline:** MongoDB's aggregation pipeline allows for complex data processing and analysis. The pipeline consists of stages, such as $match, $group, $project, and $sort, which are applied sequentially to transform and aggregate data. This enables operations like grouping, filtering, and computing aggregates on large datasets.
5. **Replication and Sharding:** MongoDB provides features for scalability and high availability. Replica sets are used for data redundancy and fault tolerance, while sharding allows data to be distributed across multiple nodes to support large-scale deployments.
6. **Transactions:** Starting from version 4.0, MongoDB supports multi-document transactions, allowing developers to perform atomic operations across multiple documents within a single transaction.

In conclusion, MongoDB is a feature-rich, scalable, and adaptable database system designed to meet a variety of data processing and storage requirements. Its elements and functions provide developers strong instruments for creating and overseeing contemporary apps. Although MongoDB lacks a single "equation," its data architecture, query language, indexing, aggregating capabilities, replication, sharding, and transaction support may all be used to understand it.

C. Some Common Mistakes

Common mistakes in MongoDB development and administration can occur due to various reasons, including misunderstandings of the data model, misconfigurations, inefficient queries, and lack of understanding of MongoDB's features and best practices. Here are some common mistakes to be aware of:

1. **Overusing Embedding:** While MongoDB's document model allows for embedding related data within a single document, overusing embedding can lead to performance issues and document size limitations. It's important to carefully consider when to embed documents and when to reference them using document references or linking.
2. **Inefficient Queries:** Writing inefficient queries, such as queries without proper indexes or queries that perform unnecessary data scans, can lead to slow query performance and increased resource consumption. It's essential to create appropriate indexes based on query patterns and use query optimization techniques like projection and aggregation pipeline stages.
3. **Lack of Indexing:** Failing to create indexes on fields used for querying can result in slow query performance, especially on large collections. It's important to identify frequently used query patterns and create indexes to support them effectively.
4. **Unnecessary Overhead:** MongoDB provides various features and functionalities, but enabling unnecessary features can introduce additional overhead and complexity. For example, enabling unnecessary validation rules, secondary indexes, or verbose logging can impact performance and resource utilization.
5. **Ignoring Data Consistency:** MongoDB offers different levels of consistency guarantees, and developers need to understand these trade-offs when designing applications. Ignoring data consistency requirements or not properly handling concurrency can lead to data inconsistency issues.
6. **Overlooking Sharding Considerations:** Sharding is a powerful feature for horizontal scaling in MongoDB, but it requires careful planning and consideration of factors such as shard key selection, data distribution, and cluster topology. Ignoring these considerations can lead to uneven data distribution, hot spots, and performance issues.
7. **Lack of Monitoring and Maintenance:** Failing to monitor MongoDB deployments and perform routine maintenance tasks such as index optimization, data compaction, and hardware upgrades can lead to performance degradation and stability issues over time.
8. **Not Keeping Up with Updates:** MongoDB releases regular updates and patches to improve performance, security, and stability. Failing to keep the MongoDB deployment up to date with the latest versions and patches can expose the system to

security vulnerabilities and compatibility issues.

9. **Not Understanding Transactions:** MongoDB introduced multi-document transactions in version 4.0, but misunderstanding how transactions work or not using them appropriately can lead to data integrity issues and unexpected behavior in applications that require transactional semantics.

Developers and administrators may create and manage reliable and effective MongoDB installations by being aware of these typical errors and adhering to best practices for MongoDB development and management. Frequent code reviews, performance tweaking, and training may help reduce these risks and guarantee that MongoDB applications run smoothly.

Here, choose the appropriate financing source. Delete this text field if it is empty. Punctuation should exist outside of quote marks in typeface to emphasize a word or phrase. When a sentence ends, a parenthetical phrase or statement is punctuated outside of the closing parentheses (like this). (Parenthesis surround the punctuation in a parenthetical statement.)

An "inset" is a graph within another graph, not a "insert." Unless you really mean anything that alternates, the term alternatively is chosen over the phrase "alternately."

Do not use the word “essentially” to mean “approximately” or “effectively”.

In your paper title, if the words “that uses” can accurately replace the word “using”, capitalize the “u”; if not, keep using lower-cased.

Be aware of the different meanings of the homophones “affect” and “effect”, “complement” and “compliment”, “discreet” and “discrete”, “principal” and “principle”.

Do not confuse “imply” and “infer”.

The prefix “non” is not a word; it should be joined to the word it modifies, usually without a hyphen.

There is no period after the “et” in the Latin abbreviation “et al.”.

The abbreviation “i.e.” means “that is”, and the abbreviation “e.g.” means “for example”.

An excellent style manual for science writers is [7].

III. USING THE TEMPLATE

In MongoDB, there isn't a concept of "templates" in the same way as in some other database systems or programming frameworks. However, you can achieve similar functionality by creating standard document structures or schemas for your collections.

Here's how you can create a "template-like" structure in MongoDB using collections and documents:

1. **Define a Standard Schema:** Determine the structure of your documents and define a standard schema that all documents within a collection should adhere to. This schema should include the fields and their data types that your application requires.

2. **Use Validation Rules:** MongoDB allows you to enforce schema validation rules at the collection level. You can define validation rules using JSON Schema to ensure that documents inserted or updated in the collection meet certain criteria. This helps maintain data consistency and integrity.

3. **Create Sample Documents:** You can create sample documents that represent the "template" for your collection. These documents can serve as examples for developers to follow when inserting or updating data in the collection. They can also provide guidance on the expected structure and format of documents.

4. **Leverage Indexes:** Define indexes on fields that are commonly queried to improve query performance. Indexes can help speed up data retrieval operations and ensure that queries are executed efficiently, especially for large collections.

5. **Document-Level Validation:** MongoDB also supports document-level validation rules using the $jsonSchema operator in queries. This allows you to perform additional validation checks on individual documents before inserting or updating them in the collection.

6. **Use Data Modeling Tools:** There are various data modeling tools and libraries available for MongoDB that can help you design and manage your document schemas effectively. These tools often provide features for visualizing schemas, generating code, and enforcing best practices.

By following these practices, you can create a standardized structure or "template" for your MongoDB collections, ensuring consistency and integrity of your data while also improving query performance and developer productivity.

After the text edit has been completed, the paper is readyfor the template. Duplicate the template file by using theSaveAscommand,andusethenamingconventionprescribedbyyour conferenceforthenameofyourpaper.In this newly created file, highlight all of the contents andimport your prepared text file. You are now ready to styleyour paper; use the scroll down window on the left of theMS Word Formatting toolbar.

A. Authors and Affiliations

B. MongoDB was originally developed by a company called 10gen, which was founded by Dwight Merriman, Eliot Horowitz, and Kevin Ryan in 2007. The company later changed its name to MongoDB, Inc. in 2013 to align with the name of its flagship product.

C. As for the affiliation, MongoDB, Inc. is the primary entity behind the development and maintenance of MongoDB. The company provides commercial services and support for MongoDB, as well as offering a managed cloud database service called MongoDB Atlas.

D. MongoDB, Inc. also oversees the open-source development of MongoDB, which is distributed under the terms of the GNU Affero General Public License (AGPL) and the Apache License. The open-source community contributes to the development of MongoDB through code contributions, bug reports, and discussions on forums and mailing lists.

The template is designed for, but not limited to, sixauthors. Aminimumofoneauthorisrequiredforallconference articles. Author names should be listed startingfrom left to right and then moving down to the next line.Thisistheauthorsequencethatwillbeusedinfuturecitations and by indexing services. Names should not belisted in columns nor group by affiliation. Please keep youraffiliationsassuccinctaspossible(forexample,donotdifferentiate among departments of the same organization).

1) ***For papers with more than six authors:***Add authornameshorizontally,movingtoathirdrowifneededformo re than 8 authors.

2) ***For papers with less than six authors:***To changethe default, adjust the template as follows.

a) ***Selection:***Highlight all author and affiliation lines.

b) ***Change number of columns:***Select the Columnsicon from the MS Word Standard toolbar and then select thecorrect number of columns from the selection palette.

c) ***Deletion:***Delete the author and affiliation lines forthe extra authors.

B. Identify the Headings

In MongoDB, a "heading" typically refers to the field names or keys within a document. These headings represent the different attributes or properties of the data stored in a MongoDB collection.

For example, consider a MongoDB document representing a user profile:

```
{
 "username": "john_doe",
 "name": "John Doe",
 "email": "john.doe@example.com",
 "age": 30,
 "city": "New York"
}
```

In this document, the headings or field names include "username", "name", "email", "age", and "city". Each heading corresponds to a specific attribute of the user profile, such as the username, name, email address, age, and city.

Headings in MongoDB collections are analogous to column names in relational database tables. They define the structure and organization of the data within the collection, allowing applications to access and manipulate individual attributes of documents.

Headings, or heads, are organizational devices that guidethereaderthroughyourpaper.Therearetwotypes:component heads and text heads.

Component heads identify the different components ofyour paper and are not topically subordinate to each other.Examples include Acknowledgments and References and,for these, the correct style to use is “Heading 5”. Use “figurecaption” for your Figure captions, and “table head” for yourtabletitle.Run-inheads,suchas“Abstract”,willrequireyou to apply a style (in this case, italic) in addition to thestyle provided by the drop down menu to differentiate thehead from the text.

Textheadsorganizethetopicsonarelational,hierarchical basis. For example, the paper title is the primarytextheadbecauseallsubsequentmaterialrelatesandelaborates on this one topic. If there are two or more sub-topics,thenextlevelhead(uppercaseRomannumerals)should be used and, conversely, if there are not at least twosub-topics, then no subheads should be introduced. Stylesnamed“Heading1”,“Heading2”,“Heading3”,and“Heading 4” are prescribed.

D. Figures and Tables

In MongoDB, there isn't a direct concept of "figures" and "tables" as in traditional relational databases. However, you can achieve similar functionalities using MongoDB collections and documents:

1. **Collections:** Collections in MongoDB are analogous to tables in relational databases. They are logical groupings of documents, and each collection can store multiple documents. Collections are schema-less, meaning that documents within the same collection can have different structures.
2. **Documents:** Documents in MongoDB are JSON-like data structures that contain field-value pairs. Each document represents a single record in the collection. Documents can have nested structures, allowing for complex and flexible data modeling.
3. **Schema Flexibility:** MongoDB's flexible schema allows you to store heterogeneous data within the same collection. This flexibility enables you to store data of varying structures and types without the need for a predefined schema.
4. **Embedded Documents:** MongoDB allows you to embed documents within other documents, similar to the concept of nested tables or sub-tables in relational databases. This allows you to represent hierarchical or nested data structures easily.
5. **Indexes:** MongoDB supports indexing to optimize query performance. You can create indexes on fields within a collection to improve the speed of data retrieval operations, similar to how you would create indexes on columns in relational database tables.
6. **Aggregation Framework:** MongoDB provides a powerful aggregation framework for performing data aggregation and analysis operations. You can use aggregation pipelines to transform, group, and analyze data within a collection, similar to the functionality provided by SQL queries and aggregate functions in relational databases.

While MongoDB doesn't have explicit "figures" and "tables" like relational databases, you can achieve similar functionalities using collections and documents. MongoDB's flexible data model, indexing capabilities, and aggregation framework provide powerful tools for storing, querying, and analyzing data in a variety of use cases.

Top of Form

a) Positioning Figures and Tables: In MongoDB, there isn't a direct concept of "figures" and "tables" as in traditional relational databases. However, you can achieve similar functionalities using MongoDB collections and documents:

1. **Collections:** Collections in MongoDB are analogous to tables in relational databases. They are logical groupings of documents, and each collection can store multiple documents. Collections are schema-less, meaning that documents within the same collection can have different structures.
2. **Documents:** Documents in MongoDB are JSON-like data structures that contain field-value pairs. Each document represents a single record in the collection. Documents can have nested structures, allowing for complex and flexible data modeling.
3. **Schema Flexibility:** MongoDB's flexible schema allows you to store heterogeneous data within the same collection. This flexibility enables you to store data of varying structures and types without the need for a predefined schema.
4. **Embedded Documents:** MongoDB allows you to embed documents within other documents, similar to the concept of nested tables or sub-tables in relational databases. This allows you to represent hierarchical or nested data structures easily.

5. **Indexes:** MongoDB supports indexing to optimize query performance. You can create indexes on fields within a collection to improve the speed of data retrieval operations, similar to how you would create indexes on columns in relational database tables.

6. **Aggregation Framework:** MongoDB provides a powerful aggregation framework for performing data aggregation and analysis operations. You can use aggregation pipelines to transform, group, and analyze data within a collection, similar to the functionality provided by SQL queries and aggregate functions in relational databases.

While MongoDB doesn't have explicit "figures" and "tables" like relational databases, you can achieve similar functionalities using collections and documents. MongoDB's flexible data model, indexing capabilities, and aggregation framework provide powerful tools for storing, querying, and analyzing data in a variety of use cases.

You

a) Positioning Figures and Tables in MongoDB

In MongoDB, as a NoSQL database, there isn't a direct concept of "figures" and "tables" as you'd find in a relational database management system (RDBMS). However, you can still organize data and achieve similar outcomes by structuring your collections and documents effectively.

Here's how you can think about organizing and positioning "figures" and "tables" in the context of MongoDB:

1. **Collections as Tables:** MongoDB collections serve a similar purpose to tables in an RDBMS. Each collection can be thought of as a logical grouping of related data, similar to how you might organize data into tables based on their relationships in a relational database.

2. **Documents as Rows:** Within MongoDB collections, documents represent individual records or rows of data. Each document contains fields (like columns in a table) that store specific pieces of information.

3. **Structuring Data:** When designing your MongoDB schema, consider how you want to structure your data. You can denormalize data by embedding related information within a single document, similar to how you might denormalize tables in a relational database. Alternatively, you can use references between documents to represent relationships between different entities.

4. **Indexing for Performance:** MongoDB allows you to create indexes on fields within your collections to improve query performance. Consider creating indexes on fields that are frequently queried or used for sorting to speed up data retrieval.

5. **Aggregation for Analysis:** MongoDB's aggregation framework allows you to perform complex data analysis operations, similar to how you might use SQL queries and aggregate functions in a relational database. You can use aggregation pipelines to group, filter, and analyze data within your collections.

6. **Data Modeling Considerations:** When positioning "figures" and "tables" in MongoDB, consider your data modeling requirements, including data access patterns, query performance, and scalability. Design your schema to optimize for the specific needs of your application.

While MongoDB doesn't have the exact concepts of "figures" and "tables" found in traditional relational databases, you can still achieve similar outcomes by effectively structuring your collections and documents and leveraging MongoDB's features for data organization, indexing, and analysis.

IV. Conclusion

MongoDB is a flexible, document-oriented database platform that is designed to be the cloud database of choice for enterprise applications. MongoDB provides a number of features that make it a great choice for a wide variety of applications.

REFERENCES

1. Cornelia A. Gyorödi* , Diana V. Dum¸se-Burescu, Doina R. Zmaranda and Robert¸S. Gyorödi, “A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management” Big Data Cogn. Comput. 2022, 6, 49. https://doi.org/10.3390/bdcc6020049.

2. Anjali Chauhan M.tech Scholar, CSE Department, Rawal Institute of Engineering and Technology, Faridabad, Haryana, India, “A Review on Various Aspects of MongoDb Databases” International Journal of Engineering Research & Technology (IJERT) http://www.ijert.org ISSN: 2278-0181 IJERTV8IS050031 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 8 Issue 05, May-2019.

3. Leveraging MonoDB : A Comprehensive Study on Database Management