encyclopedia

MDPI

Entry

MongoDB: Meeting the Dynamic Needs of Modern Applications

Mukesh Rathore and Sikha S. Bagui *

Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA; mr198@students.uwf.edu
* Correspondence: bagui@uwf.edu

Definition: This entry reviews MongoDB's fundamentals, architectural features, advantages, and limitations, providing a comprehensive understanding of its capabilities. MongoDB's impact on the database landscape is profound, challenging traditional relational databases and influencing the adoption of NoSQL solutions globally. With its continued growth, innovation, and commitment to addressing evolving market needs, MongoDB remains a pivotal player in modern data management, empowering organizations to build scalable, efficient, and high-performance applications.

Keywords: database; MongoDB; NoSQL; unstructured data; document-oriented data model

1. Introduction

MongoDB stands as a pioneering NoSQL database management system, offering a modern alternative to traditional relational databases. At its core, MongoDB operates on a document-oriented model, where data are stored as flexible JSON-like documents within collections. These collections act as containers for related documents, providing scalability and ease of management. A standout feature of MongoDB is its support for Binary JSON (BSON), a binary-encoded serialization of JSON-like documents, enhancing efficiency and performance.

Since its inception, MongoDB has experienced significant growth and evolution, continuously adapting to meet the evolving needs of modern applications. Initially released in 2009, MongoDB has undergone multiple iterations, introducing enhancements and new features to cater to diverse use cases. Its evolution reflects the dynamic nature of the database landscape, with updates focusing on performance optimization, scalability improvements, and enhanced security measures.

MongoDB is favored for its flexible schema design, which enables developers to iterate quickly and accommodate changing data requirements without the need for costly schema migrations. Its document-oriented data model aligns well with the JSON-like data structures commonly used in web and mobile applications, simplifying data storage and retrieval. Additionally, MongoDB's distributed architecture facilitates horizontal scalability, making it well-suited for handling large volumes of data and high-throughput workloads.

According to the most recent data, MongoDB commands between 6.21% [1] to 6.7% [2] of the total market share in the database management system category. MySQL by Oracle claims a substantial 24.64% share, while Microsoft SQL Server follows with 13.42%; PostgreSQL holds 8.03%, and Microsoft Access secures 7.73%. The collective market share of all NoSQL databases amounts to 7.08%. When examining MongoDB customers industry-wise, the most prominent sectors include information technology and services (26%) [2], computer software (16%), and the internet (5%). Additionally, MongoDB sees usage in other industries, such as financial services, marketing, advertising, higher education, human resources, retail, hospitals, and healthcare, albeit with smaller contributions to its overall usage [2].

MongoDB excels in use cases where flexibility and agility are paramount, such as agile development, rapid prototyping, and iterative product development. Across various

check for updates

Citation: Rathore, M.; Bagui, S.S. MongoDB: Meeting the Dynamic Needs of Modern Applications. *Encyclopedia* **2024**, *4*, 1433–1453. https://doi.org/10.3390/encyclopedia4040093

Academic Editors: Raffaele Barretta

Received: 7 August 2024
Revised: 23 September 2024
Accepted: 26 September 2024
Published: 27 September 2024

industries, MongoDB has gained widespread adoption, with notable usage in fields such as e-commerce, finance, healthcare, and technology. Its adoption rate continues to rise steadily, driven by its flexibility, scalability, and developer-friendly approach. MongoDB's impact on the database landscape is profound, challenging traditional relational databases and influencing the adoption of NoSQL solutions globally. Its ability to handle diverse data types, support for distributed architectures, and seamless integration with modern development frameworks contribute to its growing influence in the database ecosystem. MongoDB's support for geospatial queries, text search, and complex data structures makes it a preferred choice for location-based services, search applications, and data-intensive industries such as the Internet of Things (IoT) and healthcare [3].

Looking ahead, MongoDB's future usage is projected to continue growing as organizations increasingly demand agile, scalable, and high-performance database solutions. As data volume and complexity expand, MongoDB is expected to play a pivotal role in enabling organizations to harness the power of their data for insights and innovation. The rise of cloud computing and microservices architecture further reinforces MongoDB's position as a preferred database solution, offering flexibility, scalability, and ease of deployment in cloud environments [4].

2. MongoDB

MongoDB, a popular NoSQL database, is highly relevant in today's data-driven world due to its performance, scalability, and versatility [5,6]. Its document-oriented model allows for high read- and write-throughput, making it ideal for big data applications [5,6]. A comparative study of MongoDB and document-based MySQL demonstrated MongoDB's efficiency in handling Create, Read, Write, and Delete (CRUD) operations, especially query operations, which are crucial for data access and processing [6]. Furthermore, MongoDB's ability to fully utilize the "sharding-nothing cluster architecture" makes it a great tool for building high-performance data warehouses [5]. These features make MongoDB an excellent choice for managing large amounts of data originating from diverse sources like social media, cloud computing services, and the Internet of Things [6].

2.1. Features of MongoDB

- Document-Oriented Data Model [7]: MongoDB employs a document-oriented data model, storing data in flexible JSON-like documents instead of fixed schemas, enabling easier handling of unstructured data. This model supports nested fields and arrays, facilitating the storage of complex data and making it suitable for applications such as content management systems and real-time data analysis.
- Horizontal Scalability [8]: MongoDB achieves horizontal scalability through sharding and distributing data across multiple servers to handle large volumes of data effectively. Additional nodes can be added to the cluster as data volumes increase, ensuring scalability to meet application demands, which is crucial for managing big data [9].
- Indexes and Query Optimization [10]: MongoDB supports various types of indexes, including single-field, multi-field, and geospatial indexes, enhancing query performance by quickly locating the required data. Text search indexes enable users to search for specific keywords within documents, facilitating fast and efficient data retrieval, especially in large databases.
- Aggregation Operations [11]: MongoDB's aggregation framework allows complex queries on stored data, providing a set of powerful operators for filtering, grouping, and transforming data in real-time analysis and reporting scenarios. This framework handles large data volumes efficiently, enabling intricate data analysis tasks and report generation.
- Replication [12]: MongoDB's replication feature ensures high availability and data redundancy by automatically replicating data across multiple nodes. In case of node failure, MongoDB promotes a secondary node to maintain database availability, ensur-

ing data durability and preventing data loss through a clocking mechanism known as heartbeat.

- MapReduce [13]: MapReduce in MongoDB is a flexible but complex feature. It processes large datasets in parallel across multiple nodes, enabling complex data analytical tasks, statistical analysis, and report generation. It is utilized for the batch processing of extensive datasets, providing flexibility and power in data processing.
- GridFS [14,15]: MongoDB's GridFS allows storage and retrieval of large files of the size 16 MB or more, like images, videos, and audio files, by breaking them into smaller chunks distributed across multiple nodes. This efficient storage and retrieval mechanism is beneficial for applications requiring the handling of large multimedia files or content management systems.
- Automatic Sharding [16]: MongoDB's sharding feature distributes data across multiple servers to enhance database performance by balancing the load. Automatic sharding distributes data based on specified shard keys, with MongoDB's balancer ensuring even distribution across shards, thereby optimizing database performance [14]. MongoDB includes ranged sharding, hashed sharding, and zoned sharding.
- Security [17]: MongoDB's security features include authentication, authorization, encryption, and network security. It supports various authentication mechanisms and encryption at rest, injection prevention [18], as well as TLS/SSL for secure connections. MongoDB also provides a security checklist for protecting deployments and a process for reporting security bugs [19].

2.2. Architecture of MongoDB

The MongoDB application data platform depicted in the *MongoDB Architecture Guide* [19] embodies a sophisticated architecture designed to accommodate diverse data management needs. The components include the Document Model, key-value pairs, relationships, objects, graphs, geospatial, unified interface, transactional, search, mobile, real-time analytics, security, multi-cloud, and distributed architecture.

Each layer has its constituent component as follows:

1. The Bottom Layer: This foundational layer of the platform signifies the core infrastructure and security measures essential for robust data management.
    - Security is the cornerstone of this layer, encompassing robust protocols and mechanisms designed to safeguard sensitive data from unauthorized access and breaches. With data security being a top priority in today's digital landscape, this component ensures that MongoDB users can trust their data to be protected against potential threats.
    - Multi-cloud capabilities extend the reach of MongoDB's platform across multiple cloud environments, enabling organizations to leverage the benefits of different cloud providers while maintaining consistency and flexibility in their data management strategies. This component empowers users to seamlessly deploy and manage their MongoDB instances across diverse cloud infrastructures, enhancing resilience and scalability.
    - Distributed Architecture is a fundamental principle of MongoDB's architecture that enables the platform to handle large volumes of data across geographically distributed locations. By distributing data and processing tasks across multiple nodes, this component ensures high availability, fault tolerance, and performance scalability, even in the face of network failures or hardware issues.
2. The Middle Layer: The middle layer occupies a pivotal position within MongoDB's application data platform, acting as a bridge between the foundational infrastructure and the advanced data processing capabilities. This layer encompasses a diverse range of functionalities aimed at facilitating dynamic data interactions and analyses.
    - Transactional capabilities enable MongoDB users to perform data transactions with efficiency and reliability. Whether it is ensuring data integrity, supporting

concurrency control, or managing complex transactional workflows, this component provides the necessary tools and mechanisms to maintain consistency and reliability across various operations.
- Search functionality is another key aspect of the middle layer, empowering users to efficiently retrieve and query information within their MongoDB datasets. By facilitating comprehensive search capabilities, this component enhances the discoverability and accessibility of data, enabling users to extract valuable insights and information from their databases with ease.
- The mobile component plays a crucial role in enabling seamless integration and synchronization of data across different devices and platforms. Whether it is ensuring offline access, supporting real-time data synchronization, or optimizing performance for mobile environments, this component enables MongoDB users to deliver responsive and connected experiences to their mobile users.
- The unified interface component serves as a central access point for interacting with MongoDB's data platform, providing users with a cohesive and intuitive user experience across different functionalities and interfaces. By streamlining access to MongoDB's diverse set of capabilities, this component enhances usability and accessibility, empowering users to leverage the full potential of the platform without unnecessary complexity or friction.
- The real-time analytics component leverages MongoDB's real-time data processing and analysis capabilities to deliver timely insights and decision-making based on up-to-date information. Whether monitoring key performance indicators, detecting anomalies, or generating actionable insights, this component enables users to derive value from their data in real time, driving informed decision-making and business outcomes.

3. The top layer: This crowns MongoDB's application data platform with advanced data modeling and processing capabilities, enabling users to unlock the full potential of their data. As the uppermost tier of the architecture, this layer embodies MongoDB's commitment to flexibility, versatility, and innovation in data management.
    - The Document Model component represents MongoDB's signature approach to data storage and retrieval. Unlike traditional relational databases, which rely on rigid schemas, MongoDB embraces a schema-less, document-oriented model, allowing users to store and query data more flexibly and naturally. This component empowers users to represent complex data structures as rich, hierarchical documents, facilitating agile development and iteration.
    - Key-value pairs offer a lightweight yet powerful data structure for storing and retrieving simple data pairs consisting of keys and corresponding values. Ideal for scenarios such as caching, configuration management, and session storage, this component provides a straightforward and efficient solution for managing key-value datasets within MongoDB.
    - Relationships play a crucial role in modeling complex data interactions and dependencies, enabling users to establish connections between different data entities and navigate relationships with ease. Whether it is representing one-to-one, one-to-many, or many-to-many relationships, this component provides the tools and mechanisms to model and manage data relationships effectively within MongoDB's flexible document model.
    - Objects encompass a diverse range of data types and structures, allowing users to represent and manipulate complex data objects within MongoDB. From arrays and nested documents to custom data types and user-defined objects, this component provides the flexibility to accommodate a wide variety of data structures, empowering users to express their data in the most natural and meaningful way possible.
    - Graphs represent interconnected data entities and relationships in a visually intuitive manner, enabling users to analyze and traverse complex networks of data

with ease. Whether it is representing social networks, recommendation systems, or network infrastructure, this component provides the tools and algorithms to model and analyze graph-based data structures within MongoDB, facilitating powerful graph-based analytics and insights.

- Geospatial capabilities enable users to store, query, and analyze geographic information within MongoDB, supporting spatial data types, indexing, and querying capabilities. Whether it is representing points, lines, polygons, or multi-dimensional shapes, this component empowers users to build location-aware applications and perform spatial analysis within MongoDB's versatile data platform.

The interplay between these layers and components forms the bedrock of MongoDB's application data platform, enabling organizations to build robust, scalable, and versatile data solutions to meet their evolving business needs [19]. Apart from these essential components, MongoDB's architecture encompasses other components like drivers and storage engines. MongoDB's features enable developers to build flexible, scalable, and high-performance applications capable of handling large volumes of data efficiently using tools like the document-oriented data model, horizontal scalability, indexing, aggregation, replication, MapReduce, sharding, and GridFS. These features cater to the requirements of modern businesses, ensuring robust and efficient application development [19].

2.3. Comparing MongoDB with SQL Databases

Table 1 compares MongoDB and document-based SQL databases [6,20]. MongoDB and SQL databases diverge notably in their data models and scalability options. MongoDB's document-oriented approach facilitates the flexible storage of unstructured data, with documents akin to rows in SQL databases but offering nested fields, favoring complex data structures and aggregation. On the other hand, SQL databases adhere to a rigid table-based schema, enabling efficient joins across tables and precise control over data consistency [20,21]. While MongoDB excels in horizontal scalability through sharding, SQL databases typically offer vertical scalability, ensuring robust transactional capabilities but potentially limiting scalability for massive datasets [22].

Table 1. Comparing MongoDB vs. SQL databases.

MongoDB	**SQL Databases**
Collection	Table
Document	Row
Field	Column
Aggregation	Joins

In SQL databases, a primary key is a field or combination of fields that uniquely identifies each record in a table [6,20]. A primary key serves two crucial functions: ensuring uniqueness and maintaining referential integrity. It uniquely identifies each record in a table, preventing duplicate entries, and acts as a reference point for foreign keys in related tables, facilitating the establishment of relationships between them [20]. In MongoDB, the equivalent of a primary key is the _id field [23] for each document (row in SQL) in the collection (table in SQL). The _id field in MongoDB, which is immutable and generated automatically, guarantees that each document is unique and constant. If an ObjectId is not specifically given, MongoDB will assign one to ensure this uniqueness.

While both SQL and MongoDB employ unique identifiers, they diverge in several aspects. In SQL, the primary key can be any chosen field [20], while MongoDB mandates _id as the unique identifier. SQL's primary key can be altered, albeit not recommended, whereas MongoDB's _id remains immutable. Additionally, MongoDB auto-generates a

unique _id if absent, whereas, in SQL, the primary key must be set manually or through auto-increment mechanisms [6,21].

2.4. Comparing MongoDB with Other NoSQL Databases

Other popular NoSQL databases besides MongoDB include Apache Cassandra [24], Redis [25], Apache CouchDB [26], Apache Hbase [27], and Amazon DynamoDB [28]. MongoDB's simplicity in data representation and querying can reduce development time and complexity compared to databases like Cassandra [24] and Hbase [27], which may require more effort to manage and query data. Databases like Redis and CouchDB offer scalability features, and MongoDB's sharding capabilities are particularly robust and well-suited for high-volume applications. While Redis offers excellent performance for specific use cases like caching, MongoDB's versatility and ability to handle complex data structures contribute to its overall performance superiority [29].

In the NoSQL databases category, MongoDB reportedly leads with an estimated market share of approximately 44.29% [30]. Following MongoDB, DynamoDB holds a share of 10.17%, while Cassandra and HBase hold 5.16% and 4.06%, respectively. Amazon ElastiCache rounds out the list with a share of 2.32% [2].

MongoDB introduced multi-document transactions in version 4.0, allowing developers to perform atomic operations across multiple documents within a single transaction. This feature enhances data consistency and integrity, making MongoDB more suitable for transactional applications compared to databases like Redis and Cassandra, which may have limitations in this area.

While each NoSQL database has its strengths and uses cases, MongoDB's combination of ease of use, scalability, performance, transaction support, and ecosystem make it a compelling choice for a wide range of applications and industries.

MongoDB's native sharding capability allows seamless scaling across multiple nodes to handle write-intensive workloads and growing data sizes without adding complexity to applications. Sharding enables the easy adjustment of shards and shard keys, ensuring system availability while automatically rebalancing data across the shards as needed. Unlike other distributed databases that randomly distribute data, MongoDB offers multiple sharding policies tailored to specific query patterns or data placement needs, including ranged, hashed, and zoned sharding [19].

Additionally, MongoDB offers tiered scaling through the MongoDB Atlas Online Archive, enabling economical storage scaling by automatically moving aged data to lower-cost storage tiers while maintaining accessibility through federated queries [19].

2.5. Use Case Comparison of MongoDB with Other NoSQL Databases

Table 2 compares MongoDB with other document-based NoSQL databases like CouchDB and Amazon DynamoDB.

Table 2. Comparing use cases of MongoDB to other NoSQL databases.

Use Case	MongoDB	CouchDB [26]	DynamoDB [28]	Use Case Winner
Scalability and high write-throughput Applications like social media platforms or real-time analytic systems that need to scale with millions of users require high write-throughput, horizontal scalability, and low latency.	MongoDB provides excellent horizontal scalability using sharding across multiple nodes. It supports automatic failover, replica sets, and distributed data.	CouchDB [26] is not as scalable as MongoDB. It is ideal for distributed setups with master–master replication and offline synchronization features but struggles with massive data loads.	DynamoDB [28] is fully managed and provides near-infinite scalability with automatic partitioning and horizontal scaling. It is ideal for applications requiring seamless scaling.	MongoDB is a good fit for recommendation engines, social media feeds, and event logging due to its dynamic schema and support for complex queries.

Table 2. *Cont.*

Use Case	MongoDB	CouchDB [26]	DynamoDB [28]	Use Case Winner
Scalability and high write-throughput Applications like social media platforms or real-time analytic systems that need to scale with millions of users require high write-throughput, horizontal scalability, and low latency.	MongoDB excels in write-heavy workloads, especially in real-time, high-throughput applications, with its write-concern options for durability and consistency.	CouchDB offers a lower write-throughput as it is append-only B-tree architecture but provides strong consistency with its ACID-compliant design.	DynamoDB has the capability of very high write-throughput, but it needs operations to be explicitly set to strongly consistent. Provisioned throughput allows fine-tuning capacity based on workload.	DynamoDB can be used for serverless applications, IoT backends, and large-scale retail apps that need predictable performance and handle unpredictable spikes in traffic.
Applications with flexible data modeling requirements like CMS (Content Management Systems) [31], CRM (Customer Relationship Management) [32], and e-commerce.	MongoDB is known for its flexible schema, which allows handling a wide range of document structures. New fields can be added dynamically to store complex nested objects.	CouchDB supports schema-less documents, but its focus is more on replication and distributed document management rather than complex querying or flexible schema manipulation.	DynamoDB uses a key-value store with optional secondary indexes, allowing semi-structured data to be modeled in a flexible way. However, it is less flexible in terms of querying and modeling complex relationships compared to MongoDB.	MongoDB shines in environments where data modeling requirements change frequently or are unstructured, making it ideal for CMS [31], CRM systems [32], and e-commerce, where product attributes vary widely.
Applications requiring advanced query capabilities such as data analytics, business intelligence, or reporting tools that require complex querying, aggregations, and secondary indexes.	MongoDB supports a powerful querying engine with support for aggregations, joins (with $lookup), and secondary indexes. The aggregation framework is robust, allowing for complex data processing pipelines.	CouchDB's querying model is more limited, relying on MapReduce views. It does not provide the same level of advanced query capabilities as MongoDB.	DynamoDB provides basic query capabilities and secondary indexes, but it does not support joins or complex aggregations. For advanced querying, you might need to supplement DynamoDB with Amazon Athena [33] or ElasticSearch [34].	MongoDB is ideal for analytics-driven applications that require complex queries, filters, and aggregations, such as business intelligence tools or reporting systems.
Offline-first applications with replication needs, such as mobile apps or field service apps, require offline functionality and seamless data replication when the connection is restored.	MongoDB supports replica sets and sharding, but offline-first capabilities are not as streamlined compared to CouchDB.	CouchDB excels in multi-master replication and offline-first use cases. It allows local storage and synchronization of databases across remote locations.	DynamoDB does not natively support offline-first scenarios, but Amazon provides services like DynamoDB Streams [35] and AWS AppSync [36] to handle synchronization.	CouchDB is perfect for offline-first applications where users need local copies of the data, such as field service, remote monitoring, or mobile apps that operate in intermittent connectivity environments.
				MongoDB can be used with external tools for offline functionality but is not as seamless for offline-first mobile applications.

3. Advantages

MongoDB offers several advantages, which include the following:

- Flexible Schema and Ease of Use: MongoDB's document-oriented data model allows for flexible schemas, enabling developers to store data without a predefined structure. This flexibility accommodates evolving data requirements and simplifies database management [14].
- Scalability: MongoDB supports horizontal scalability through sharding, distributing data across multiple nodes to handle large volumes of data and high traffic loads efficiently. This scalability ensures that MongoDB can grow with the needs of an application [8].
- High Performance: MongoDB's indexing, aggregation, and query optimization capabilities contribute to high performance, enabling fast data retrieval and processing. Additionally, MongoDB's in-memory storage engine, WiredTiger [37], further enhances performance for read-heavy workloads.
- Rich Query Language: MongoDB's query language (MQL) [38] offers powerful features for querying and manipulating data, including support for complex queries, aggregation pipelines, geospatial queries, and text search through their MongoDB shell (Mongosh) [39] platform. This allows developers to express a wide range of data access patterns.
- Replication and High Availability: MongoDB supports automatic data replication across multiple nodes, ensuring high availability and fault tolerance. In the event of node failure, MongoDB can automatically promote a secondary node to primary status, minimizing downtime and data loss [14].
- Document-Level Transactions: MongoDB introduced multi-document transactions, allowing developers to perform atomic operations across multiple documents within a single transaction. This ensures data consistency and integrity, especially in transactional applications [14].
- Rich Ecosystem and Community: MongoDB offers a comprehensive ecosystem of tools, libraries, and services to support development, deployment, and management tasks. The MongoDB community [40] is active and vibrant, providing resources, documentation, and support to developers worldwide.
- Real-Time Analytics: MongoDB's real-time analytical [41] capabilities provide organizations with immediate insights by processing data as it is generated, enabling agile decision-making. It seamlessly integrates diverse data sources and facilitates streamlined data collection, ensuring relevant information is captured promptly. MongoDB's agile data convergence eliminates the need for slow ETL processes, allowing for swift analysis of combined data streams. The platform's seamless integration of transactional and historical data enhances the completeness and accuracy of real-time analysis while keeping costs low. With in-place analysis and a multitude of use cases spanning personalization, fraud prevention, process optimization, and preemptive maintenance, MongoDB empowers businesses to derive actionable insights swiftly and efficiently, driving success in dynamic environments [42].

Overall, MongoDB's advantages make it a popular choice for building modern, scalable, and flexible applications across various industries and use cases.

4. Limitations

While MongoDB offers many advantages, it also has some limitations:

- Memory Consumption: MongoDB's memory usage can be relatively high, especially when handling large datasets or complex queries. This can lead to increased hardware requirements and costs for memory-intensive workloads [14].
- Transaction Support: Transactional support in MongoDB refers to the ability of the database system to maintain the atomicity, consistency, isolation, and durability (ACID) properties for operations that involve multiple documents or collections and enable

MongoDB to ensure that a series of operations either all succeed, or all fail, preserving data integrity, even in the face of concurrent access and failures. Although MongoDB introduced multi-document transactions in version 4.0, transactions are not supported across distributed shards in a sharded cluster. This limitation sometimes impacts the consistency and integrity of data in distributed environments [14]. Before MongoDB 4.0, the database did not support multi-document ACID transactions [43], which are critical for complex operations that need strong consistency. MongoDB 4.0+ supports multi-document transactions, but these transactions come with a performance overhead, especially compared to the relational databases that handle transactions more efficiently and are designed around them. The MongoDB 7.0 version does not yet handle transactions as robustly as the traditional relational databases. MongoDB may fall short if applications require strong ACID compliance [43] and complex transaction handling [44,45].

- Data Size Limitations: MongoDB has a document size limit of 16 MB, which may restrict the storage of large documents or datasets. While this limit can be circumvented by using GridFS [15] for storing large files, it adds complexity to data management.
- Schema Design Complexity [46]: While MongoDB's flexible schema can be advantageous, it also requires a careful schema design to ensure efficient query performance and data integrity. Poorly designed schemas can lead to performance issues or data inconsistency. In contrast, the schema design complexity in SQL databases is often characterized by a rigid structure, normalization requirements, and management of complex relationships and constraints.
- Join Operations: MongoDB's document-oriented data model does not support the traditional SQL-style joins between collections. While MongoDB provides mechanisms such as embedded documents and denormalization to address this limitation, complex join operations may be challenging to perform efficiently.
- Indexing Overhead/Unique Indexes: Creating indexes in MongoDB incurs overhead in terms of storage space and write performance. Maintaining indexes for frequently queried fields is essential for efficient query execution but requires careful consideration to balance performance and storage requirements [47].
- Maturity of Ecosystem: While MongoDB's ecosystem is comprehensive, some components may not be as mature or feature-rich as those of traditional relational databases. For example, tools for analytics, reporting, and data integrations are less developed compared to SQL-based solutions.
- Concurrency and locking [46]: MongoDB employs a global write lock at the database level, which can impact write scalability in high-concurrency environments. While MongoDB offers fine-grained locking at the collection level for read operations, write-heavy workloads may experience contention and performance bottlenecks.
- Buit-in Reporting: MongoDB lacks advanced, built-in reporting tools like some relational databases (e.g., MySQL, PostgreSQL) or specialized databases like Oracle, which offer built-in reporting and analytics capabilities. External tools (e.g., BI tools like Tableau [48] or MongoDB aggregation queries [11]) are required to generate reports, which adds complexity. MongoDB does not offer the same level of built-in reporting features found in relational databases like MySQL. Reporting tools must be integrated into MongoDB Atlas via third-party solutions like a Microsoft-certified Power BI connector and a Tableau connector [45,48].
- SQL Query Support: MongoDB does not support SQL natively [49]. It has its own proprietary query language, MongoDB Query Language (MQL) [38], instead of SQL, which is more flexible for certain NoSQL use cases but lacks the comprehensive features of SQL, especially for complex querying, joins, and subqueries. This limitation has been overcome by using the MongoDB Atlas SQL interface [50], which uses SQL-style queries that again do not support write operations [44].
- Data Duplication Problem: In MongoDB, data duplication (or denormalization [51]) is a common design pattern to optimize performance and avoid expensive joins. This,

however, can lead to data redundancy and potential consistency issues, as multiple copies of the same data might need to be updated simultaneously. This is unlike relational databases, which uses normalization to eliminate duplication and enforce referential integrity with foreign keys. MongoDB's flexible schema design often leads to data duplication, especially in denormalized database models, which can result in storage inefficiencies and challenges with data consistency [44,49].

Despite these limitations, MongoDB remains a popular choice for many applications due to its flexibility, scalability, and rich feature set. However, it is essential for developers and architects to carefully evaluate these limitations in the context of their specific use cases and requirements.

5. CRUD Operations in MongoDB

5.1. Create a Database and Collection

Figure 1 demonstrates the creation of a database called "StudentRecords" and a collection called "student" under that database in MongoDB using the Mongosh shell. The "use" command switches the database to "StudentRecords" if it already exists; otherwise, MongoDB will create it before making the switch. Upon successful execution, Mongosh responds with "ok".

```
>_MONGOSH

> // Switch to the StudentRecords database or create it if it doesn't exist
> use StudentRecords
< switched to db StudentRecords
> // Create the student collection
  db.createCollection("student");
< { ok: 1 }
StudentRecords>
```

Figure 1. Creating a database and collection in MongoDB using Mongosh shell.

5.2. Insert Documents into the Collection

Figure 2 illustrates the process of creating documents in MongoDB. The method "insertOne()" is employed in MongoDB for inserting a singular document into a collection, while "insertMany()" enables the insertion of multiple documents simultaneously.

The student documents inserted into the "student" collection comprise various fields, each serving to encapsulate specific information about the students. The "name" field represents an object encompassing details regarding the student's name, including "firstName", "middleName", and "lastName", all of which are of the string data type. The "college" field, also of the string data type, denotes the name of the college the student attends. Additionally, the "dateOfBirth" field, another string data type, signifies the student's date of birth. The "sports" field is an array of objects, with each object containing information about a sport the student participates in. Within each object, there are "sport", "location", and "teamName" fields, all of which are of the string data type. Lastly, the "courses" field represents an array of objects, with each object detailing a course in which the student is enrolled. Each course object contains "courseName" and "courseNumber" fields of the string data type, along with a "creditHrs" field of the number data type, representing the credit hours associated with the course, ranging from 0 to 4.

```
>_MONGOSH
> db.student.insertMany([
    {
        name: { firstName: "John", middleName: "", lastName: "Doe" },
        college: "XYZ University",
        dateOfBirth: "05/15/1998",
        sports: [
            { sport: "Football", location: "Campus Stadium", teamName: "XYZ Tigers" },
            { sport: "Basketball", location: "Campus Gymnasium", teamName: "XYZ Eagles" }
        ],
        courses: [
            { courseName: "Computer Science", courseNumber: "CS101", creditHrs: 3 },
            { courseName: "Mathematics", courseNumber: "MATH201", creditHrs: 4 }
        ]
    },
    {
        name: { firstName: "Alice", middleName: "Elizabeth", lastName: "Smith" },
        college: "ABC College",
        dateOfBirth: "02/28/1999",
        sports: [
            { sport: "Tennis", location: "College Tennis Courts", teamName: "ABC Aces" },
            { sport: "Swimming", location: "College Swimming Pool", teamName: "ABC Sharks" }
        ],
        courses: [
            { courseName: "Physics", courseNumber: "PHYS202", creditHrs: 4 },
            { courseName: "Chemistry", courseNumber: "CHEM101", creditHrs: 3 }
        ]
    }
]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('660e3380853cc17542c70823'),
      '1': ObjectId('660e3380853cc17542c70824')
    }
  }
StudentRecords>
```

Figure 2. Inserting records into the 'student' collection.

The output from Mongosh in Figure 2 indicates that the insertion operation using "insertMany()" was acknowledged as successful, and it provides the ObjectIds of the inserted documents as "insertedIds", with each ObjectId corresponding to its respective document.

Figure 3 shows the output of the fields displayed in the student collection under MongoDB Compass.

5.3. Update Documents under 'Student' Collection

Just as with the "insertOne()" and "insertMany()" commands, MongoDB offers "updateOne()" and "updateMany()" commands specifically designed for updating documents.

Figure 4 demonstrates the usage of the "updateOne()" command, which updates a document based on a specific condition.

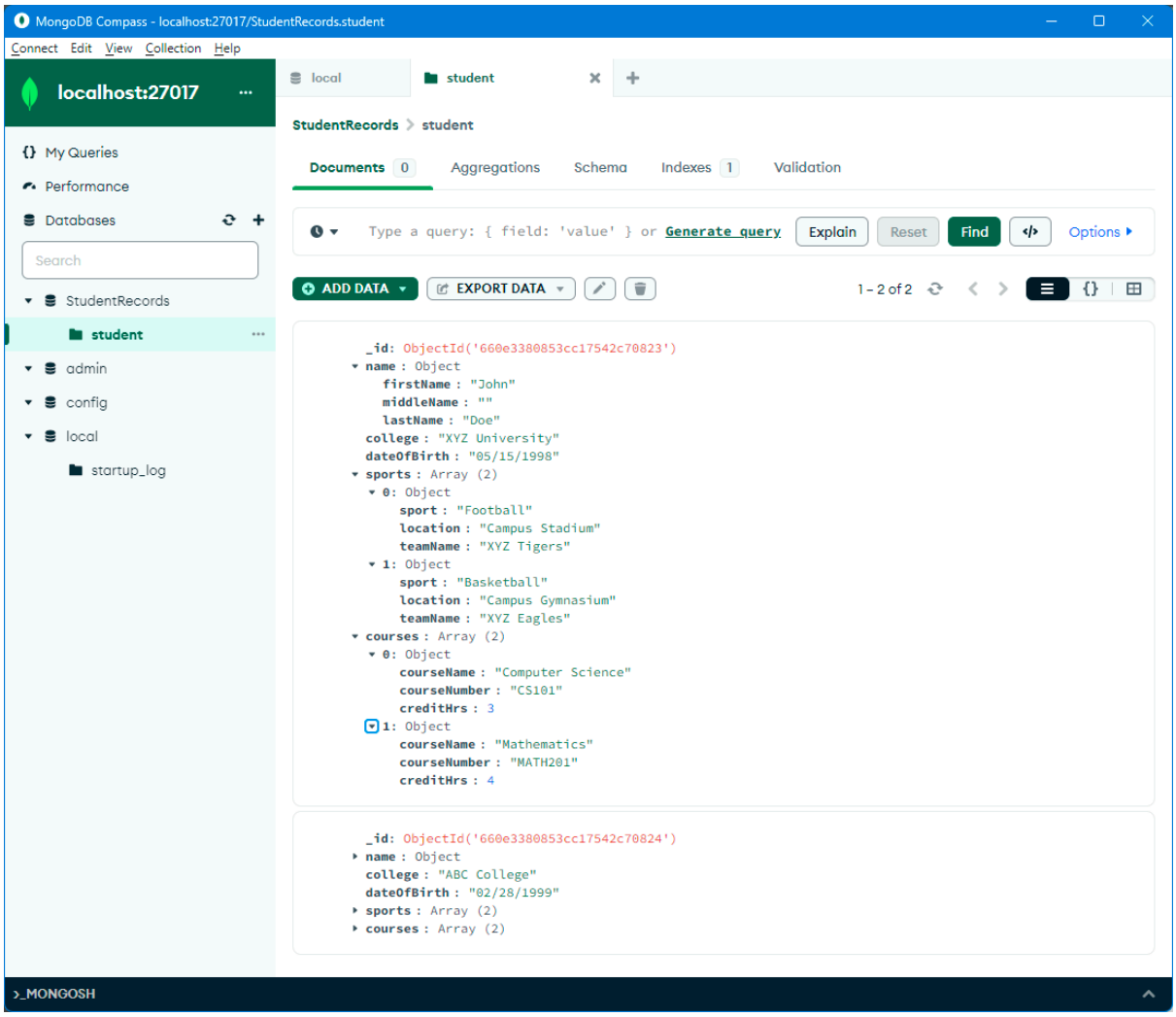

Figure 3. Displaying documents in the 'student' collection under MongoDB Compass.

```
>_MONGOSH
> // update dateOfBirth field to "12/12/2002" for a student with the first name "John" in the student collection
> db.student.updateOne(
    { "name.firstName": "John" },
    { $set: { dateOfBirth: "12/12/2002" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
StudentRecords>
```

Figure 4. Update document that is based on specific conditions using Mongosh shell.

5.4. Querying Documents Using Mongosh

The "find()" method employs a query filter to locate documents. In Figure 5, it is utilized to retrieve the names of students with three credit hours and subsequently project

only the "name" and "_id" fields. The "{ "_id": 0 }" filter can be employed to omit the "_id" field from the output.

```
>_MONGOSH
> /**
  * documents where the creditHrs field within the courses array equals 3
  * and returns only the name field of those documents
  */
> db.student.find(
    { "courses.creditHrs": 3 },
    { "name": 1 }
  )
< {
    _id: ObjectId('660e3380853cc17542c70823'),
    name: {
      firstName: 'John',
      middleName: '',
      lastName: 'Doe'
    }
  }
  {
    _id: ObjectId('660e3380853cc17542c70824'),
    name: {
      firstName: 'Alice',
      middleName: 'Elizabeth',
      lastName: 'Smith'
    }
  }
StudentRecords>
```

Figure 5. Querying documents in Mongosh.

5.5. Deleting Documents in MongoDB

The query shown in Figure 6 is employed to remove all students from the "ABC College" within the student collection. It utilizes the "deleteMany()" method alongside a query filter to identify documents where the college field matches "ABC College".

```
>_MONGOSH
> // delete all students from the "ABC College" college in the student collection
> db.student.deleteMany({ college: "ABC College" })
< {
    acknowledged: true,
    deletedCount: 1
  }
StudentRecords>
```

Figure 6. Deleting documents in MongoDB.

In Figure 7, the MongoDB collection is depicted following the execution of the deletion operation. It illustrates the updated list of documents post-deletion, demonstrating the impact of removing all records associated with the "ABC College" from the collection.

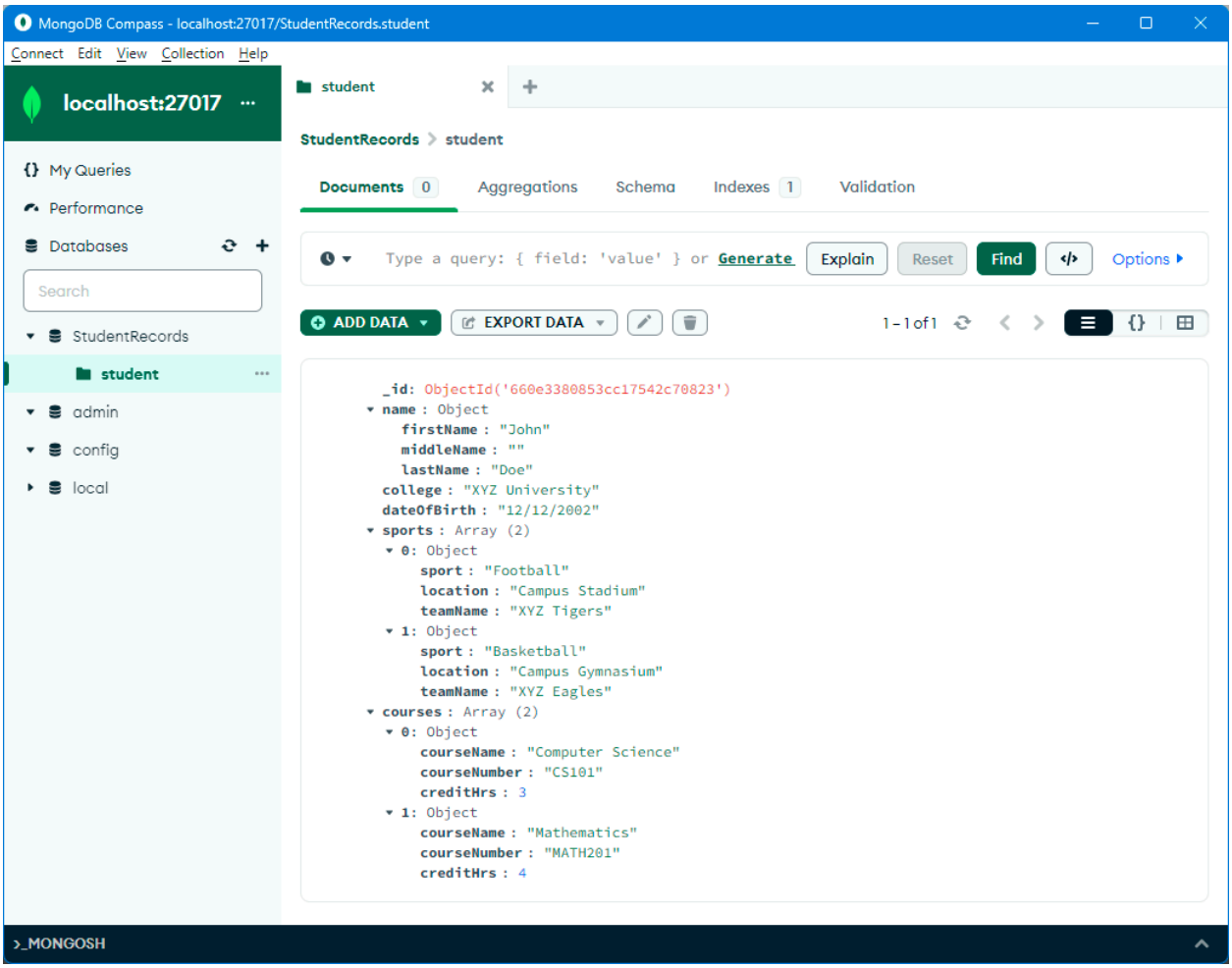

Figure 7. List of documents after the delete operation in MongoDB.

6. Structure of MongoDB

The structure of MongoDB is based on a document-oriented model, which means that data are stored in the form of documents rather than tables, as in traditional relational databases. Each document is a JSON-like object composed of field-value pairs. These documents are then organized within collections, which are akin to tables in relational databases but with more flexibility.

One of the key features of MongoDB is its support for embedded documents and arrays within these documents. Embedded documents allow for the nesting of one document within another, which is useful for representing hierarchical data structures. Arrays, on the other hand, enable the storage of multiple values within a single field, providing a convenient way to handle lists or sets of related data.

The guidelines mentioned in [52] offer a comprehensive approach to the schema design in MongoDB, focusing on efficient data access and the effective modeling of relationships. Firstly, they advocate for grouping data that are frequently accessed together within the same document, aiming to enhance query performance. Secondly, for one-to-one relationships, the recommendation is to embed related data within a single document, streamlining queries and ensuring data coherence. Similarly, for one-to-few relationships, embedding the related data within the parent document is advised to improve data locality and reduce the need for complex joins. For one-to-many and many-to-many relationships, the guidelines

suggest using child references, which promotes data consistency and facilitates efficient updates. Lastly, for unbounded one-to-many relationships, utilizing parent references is recommended, enabling efficient querying while avoiding document size limitations.

Overall, these guidelines aim to optimize data access, ensure data consistency, and simplify the querying processes by harnessing MongoDB's document-oriented schema design capabilities [52]. This structure offers several advantages. It allows for schema flexibility, as documents within a collection can have different structures, making it easy to adapt to changing data requirements. Moreover, the document-oriented model aligns well with the way developers think about data, as objects in their programming languages often map directly to documents in MongoDB.

7. Applications of MongoDB

MongoDB is extensively utilized across numerous industries due to its flexibility and scalability. Some key use cases include the IoT, mobile applications, content management, e-commerce, mainframe offloading, and configuration management. MongoDB plays a crucial role in artificial intelligence, cloud-based applications, IoT, and smart environments, offering robust solutions for managing and analyzing large volumes of diverse data for the financial and IT industries [53].

MongoDB offers a range of use cases across various industries, enhancing different aspects of application development. For artificial intelligence (AI), it streamlines the process of building AI-enriched applications. Edge computing is made more accessible by simplifying data management at the edge. On the IoT, MongoDB helps analyze and act on real-world data from physical devices. For mobile app development, it accelerates the process with ease.

MongoDB also supports payments by modernizing architecture and enables serverless development for scalable applications. It offers single-view functionality, giving real-time access to critical data and personalization to deliver relevant content to users. MongoDB also aids in catalog management, content management, and mainframe modernization and builds reliable, scalable gaming applications globally [54]. The following section presents specific use cases in these contexts.

7.1. MongoDB and AI in Financial Services

MongoDB plays a significant role in enabling AI-driven innovations in the financial services sector. Private banking relationship managers' responsibilities include tracking bond coupon dates, analyzing transaction costs, summarizing research, and generating conversation summaries. Traditionally, these tasks were laborious and manual, but generative AI systems like RAG and integration with databases such as MongoDB further streamline information retrieval for managers [55].

Financial institutions, including retail banks and capital market firms, handle diverse documents crucial to their operations, often arriving in unstructured formats, making information retrieval challenging. MongoDB offers a solution by efficiently storing vast amounts of both live and historical data, essential for AI applications like generative AI. With its vector search capabilities and support for transactions, MongoDB facilitates rapid access to relevant information, enhancing the efficiency of financial institutions in document analysis and decision-making [55].

7.2. MongoDB in IT Services

MongoDB provides a rich ecosystem of tools, libraries, and services to support development, deployment, and management tasks. Features such as MongoDB Atlas (managed database service), MongoDB Compass (GUI for database management), MongoDB Charts (data visualization tool), and MongoDB Cloud Services [56] offer developers a comprehensive suite of solutions to build and maintain MongoDB databases effectively [57].

MongoDB offers two server options for different user needs. The community server provides a flexible document data model with features like ad hoc queries and real-time

aggregations. It can be downloaded for free and is compatible with MongoDB Atlas, which offers additional managed services and advanced functionalities like auto-scaling and a full-text search across various cloud platforms [58]. On the other hand, the Enterprise Server, available with the MongoDB Enterprise Advanced subscription, provides commercial-grade features such as in-memory storage and advanced security controls, catering to businesses' more complex requirements while also offering a free evaluation and development version [59].

MongoDB is widely utilized across various industries due to its versatility. It efficiently manages product data, aids in real-time decision-making for operational intelligence, and enhances product catalog management through dynamic schema capabilities. Additionally, MongoDB supports scalability and application mobility in mobile app development, simplifies customer analytics by facilitating data aggregation, and streamlines mainframe data offloading processes. Moreover, it enables real-time data integration, providing flexibility and efficient query capabilities for aggregating and organizing distributed data, thereby empowering companies with better decision-making and insights. Overall, MongoDB's diverse applications underscore its effectiveness in modern data management scenarios [60].

7.3. MongoDB in Cloud-Based Applications

Cloud computing has transformed businesses by enabling global collaboration, reducing infrastructure costs, and allowing easy scalability. Cloud computing provides flexibility, as users can access files and services from any device with internet connectivity, like using MongoDB Atlas, without needing installation on local machines.

The architecture of cloud computing includes components like the front end (client interface) and back end (servers, storage, and management tools), connected via networks. Virtualization allows multiple users to share resources like computing power and storage seamlessly.

Cloud services come in the following various models: (i) IaaS (Infrastructure-as-a-Service) offers IT infrastructure like virtual machines; (ii) PaaS (Platform-as-a-Service) provides platforms for software development; and (iii) SaaS (Software-as-a-Service) delivers software applications via the internet. Cloud computing is increasingly critical in emerging technologies like AI, gaming, and hybrid cloud solutions. Serverless computing is a newer model that scales resources automatically based on events, enhancing cost efficiency and flexibility.

MongoDB is well-suited for cloud environments due to its ability to scale horizontally and handle distributed databases. It enables seamless data storage, retrieval, and management across cloud platforms, supporting modern microservices architectures, containerization, and serverless computing. Cloud-based applications benefit from MongoDB's high availability, replication, and sharding features, ensuring reliable and fast data access at scale. MongoDB Atlas exemplifies a popular cloud database service supporting these advancements [61].

7.4. MongoDB for Internet of Things (IoT)

IoT refers to a vast network of interconnected devices that collect and share data, ranging from personal gadgets to industrial equipment. These devices generate continuous streams of data that require efficient management and processing. IoT applications span across various sectors. Industrial IoT (IIoT) monitors manufacturing processes and equipment, while healthcare devices enable telemedicine and remote monitoring. MongoDB outlines the essential layers of the IoT architecture and their roles in ensuring a seamless flow of information.

IoT architecture consists of four crucial stages that streamline data flow and processing. First, devices such as sensors and actuators collect data, which is then transmitted to the internet gateway stage, where these data are pre-processed. The gateway enables communication between the sensors and the internet. Next, edge computing processes the data closer to its source, reducing latency and bandwidth usage. This local processing allows for

real-time decision-making, making it ideal for applications needing quick responses and minimal cloud reliance. Finally, the cloud or data center stage stores the data in databases like MongoDB for deeper analysis, machine learning, and long-term operations, providing large-scale processing and storage capacity. This structure ensures efficient, scalable data management in IoT systems [62].

MongoDB is widely used in the IoT for handling the massive influx of real-time data generated by connected devices. It efficiently stores unstructured and semi-structured sensor data, facilitating quick retrieval and analysis. Its flexible schema allows for easy adjustments as new data types emerge. MongoDB enables real-time processing and aggregation of IoT data, which is critical for applications like predictive maintenance, asset tracking, smart metering, and connected vehicles. Additionally, it helps integrate IoT data into existing cloud platforms for further analysis and insights [54].

MongoDB provides an efficient platform for managing the complexities of Internet of Things (IoT) applications by supporting the entire IoT data lifecycle—from data ingestion to storage, real-time analytics, and archiving. With its flexible document model and horizontal scaling, MongoDB handles the high volume and velocity of IoT data seamlessly. It enables real-time data replication, operational workloads, and analytics within a single platform, simplifying data synchronization and access.

Companies like Bosch [63], Toyota [64], and Vaillant [65] leverage MongoDB to power IoT platforms that support intelligent vehicles, smart factories, and advanced IoT infrastructures, delivering actionable insights for diverse industries [66].

7.5. MongoDB in Smart Environments

In the context of smart environments, such as smart cities, homes, and industrial setups, MongoDB is utilized to manage complex and diverse datasets from multiple sources. These environments generate vast amounts of real-time data from sensors, cameras, and other devices, requiring a database that can scale dynamically. MongoDB supports the integration and processing of data from multiple sources, enabling applications like smart traffic management, energy optimization, smart grids, and environmental monitoring. It also allows for the integration of AI and machine learning models to automate decision-making and improve efficiency in smart environments [67].

As an example, as elaborated in [68], Rome is undergoing a digital transformation to address the modern needs of its four million citizens and tourists. The goal is to turn Rome into a smart city by enhancing its infrastructure, services, and digital systems. The challenge is to develop a centralized data platform that could ingest vast amounts of data from various city services, streamlining operations and providing real-time insights. To achieve this, Roma Capitale, the city's governing body, partnered with MongoDB. MongoDB's scalable and flexible data model allowed the city to integrate diverse data types—such as transport, healthcare, and administrative records—into a unified system. This transformation proved crucial during the COVID-19 pandemic when efficient access to real-time data improved the delivery of healthcare and welfare services. Currently, Rome's new data ecosystem manages over 110 million documents and continues to grow, significantly improving the efficiency of city services. With MongoDB's help, Rome has risen in global smart city rankings and has provided residents with seamless access to personal documents and services [68].

8. Conclusions

In conclusion, MongoDB has emerged as a leading NoSQL database management system, offering a modern alternative to traditional relational databases. Its document-oriented model, supported by features like Binary JSON (BSON) serialization and flexible schema capabilities, provides scalability and ease of management for diverse data types. Since its inception in 2009, MongoDB has continually evolved to meet the dynamic needs of modern applications, witnessing widespread adoption across industries such as e-commerce, finance, healthcare, and technology. Overall, the structure of MongoDB, with its

document-oriented model, collections, and support for embedded documents and arrays, provides a powerful and efficient platform for storing and managing diverse types of data.

This research paper delves into MongoDB's fundamental concepts, architectural features, advantages, and limitations, providing a comprehensive understanding of its capabilities. Notably, MongoDB's rich ecosystem, including tools like MongoDB Atlas and MongoDB Compass, further enhances its usability and effectiveness for developers and businesses alike.

This entry also explores MongoDB's use cases across various industries, highlighting its role in product data management, operational intelligence, mobile application development, customer analytics, and more. Additionally, MongoDB's integration with AI technologies and its utility in IT services underscore its versatility and relevance in today's data-driven world.

MongoDB's impact on the database landscape is profound, challenging traditional relational databases and influencing the adoption of NoSQL solutions globally. With its continued growth, innovation, and commitment to addressing evolving market needs, MongoDB remains a pivotal player in modern data management, empowering organizations to build scalable, efficient, and high-performance applications.

MongoDB supports various use cases across industries. It enhances AI applications, simplifies edge computing, and enables real-time IoT data analysis. For mobile apps, it speeds up development, modernizes payment systems, and offers scalable serverless development. MongoDB provides real-time single-view access and personalization and aids in catalog and content management. It also supports mainframe modernization and powers scalable, global gaming applications. Overall, MongoDB's scalability, flexibility, and real-time data capabilities make it ideal for data-intensive environments.

Author Contributions: Conceptualization, M.R. and S.S.B.; methodology, M.R. and S.S.B.; software, M.R.; validation, S.S.B.; formal analysis, M.R.; investigation, M.R.; resources, S.S.B.; writing—original draft preparation, M.R.; writing—review and editing, S.S.B.; visualization, M.R.; supervision, S.S.B.; project administration, S.S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received not external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data Sharing is not applicable.

Acknowledgments: This paper has been partially supported by the Askew Foundation at the University of West Florida.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Datanyze LLC. *MongoDB Market Share*. Available online: https://www.datanyze.com/market-share/databases--272/mongodb-market-share (accessed on 25 April 2024).
2. Enlyft Inc. *Companies Using MongoDB*. Available online: https://enlyft.com/tech/products/mongodb (accessed on 25 April 2024).
3. The Future of MongoDB in 2024: Shaping the Data Landscape, MARTECH QUEST. Available online: https://www.martechquest.com/marketing-strategy/the-future-of-mongodb-in-2024-shaping-the-data-landscape/ (accessed on 13 April 2024).
4. Where Is MongoDB Going Next? Database Trends and Applications. 15 July 2021. Available online: https://www.dbta.com/Columns/MongoDB-Matters/Where-is-MongoDB-going-next-147582.aspx (accessed on 13 April 2024).
5. Anjali, C. A Review on Various Aspects of MongoDB Databases. *Int. J. Eng. Res. Technol.* **2019**, *8*, 90–92.
6. Győrödi, C.A.; Dumşe-Burescu, D.V.; Zmaranda, D.R.; Győrödi, R.S.A. Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management. *Big Data Cogn. Comput.* **2022**, *6*, 49. [CrossRef]
7. Data Modeling, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/data-modeling/ (accessed on 25 April 2024).

8. Scalability with MongoDB Atlas—WhitePaper. July 2022. Available online: https://www.mongodb.com/collateral/scalability-with-mongodb-atlas (accessed on 25 April 2024).
9. Ancuta-Pentronela, B.; Mihai, B.; Mihai, C. Horizontal scalability towards server performance improvement. In Proceedings of the 16th RoEduNet Conference: Networking in Education and Research (RoEduNet), Targu-Mures, Romania, 21–23 September 2017.
10. Indexes, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/indexes/ (accessed on 25 April 2024).
11. Aggregation Operations, MongoDB Manual 7.0. Available online: https://www.mongodb.com/docs/manual/aggregation/ (accessed on 25 April 2024).
12. Replication, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/replication/ (accessed on 25 April 2024).
13. Map-Reduce, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/core/map-reduce/#std-label-map-reduce (accessed on 25 April 2024).
14. Sharma, M.; Trivedi, N.K. A Comprehensive Review of Mongo DB: Features, Advantages, and Limitations. In Proceedings of the 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 6–8 July 2023.
15. GridFS, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/core/gridfs/ (accessed on 25 April 2024).
16. Sharding, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/sharding/ (accessed on 25 April 2024).
17. Security, MongoDB Manual 7.0. 15 August 2023. Available online: https://www.mongodb.com/docs/manual/security/ (accessed on 25 April 2024).
18. Trudeau, M.; Kolodny, J. An Analysis and Overview of MongoDB Security. 2017. Available online: https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/07/Paper.pdf (accessed on 25 April 2024).
19. MongoDB Architecture Guide. November 2021. Available online: https://www.mongodb.com/collateral/mongodb-architecture-guide (accessed on 2 April 2024).
20. Bagui, S.; Earp, R. *Database Design Using ER Diagrams*, 3rd ed.; Taylor and Francis Group: New York, NY, USA, 2023.
21. Earp, R.; Bagui, S. *A Practical Guide to Using SQL in Oracle*, 3rd ed.; BVT Publishing: Redding, CA, USA, 2021.
22. MongoDB Documentation Manual, MongoDB Manual 7.0.7. 18 March 2024. Available online: https://www.mongodb.com/docs/manual/introduction/ (accessed on 25 April 2024).
23. MongoDB Inc. *Documents*. Available online: https://www.mongodb.com/docs/manual/core/document/ (accessed on 13 April 2024).
24. Overview, The Apache Software Foundation, Cassandra Documentation 5.0. Available online: https://cassandra.apache.org/doc/latest/cassandra/architecture/overview.html (accessed on 27 July 2024).
25. Redis Documentation, Redis Inc, Redis GitHub Documentation 7.4. Available online: https://github.com/redis/redis/tree/7.4 (accessed on 29 July 2024).
26. Overview, Apache Software Foundation, Apache CouchDB 3.3.3 Documentation. Available online: https://docs.couchdb.org/en/stable/ (accessed on 29 July 2024).
27. Apache HBase Reference Guide, Apache Software Foundation, Apache HBase Reference Guide. Available online: https://hbase.apache.org/book.html (accessed on 29 July 2024).
28. Kalid, S.; Syed, A.; Mohammad, A.; Halgamuge, M.N. Big-data NoSQL databases: A comparison and analysis of Big-Table, DynamoDB, and Cassandra. In Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 10–12 March 2017.
29. Kaur, G.K.; Singla, S.; Khawas, V. Database Management System: A Study of Increasing Impact of NoSQL Databases. In Proceedings of the 2023 International Conference on Advanced Computing & Communication Technologies (ICACCTech), Banur, India, 23–24 December 2023.
30. 6Sense Insights Inc. *Market Share of MongoDB*. Available online: https://6sense.com/tech/nosql-databases/mongodb-market-share (accessed on 25 April 2024).
31. What Is A Content Management System (CMS)? TechTarget. Available online: https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS (accessed on 5 September 2024).
32. Salesforce, Inc. What Is CRM (Customer Relationship Management)? Available online: https://www.salesforce.com/crm/what-is-crm/ (accessed on 5 September 2024).
33. Amazon Web Services, Inc. Interactive SQL—Amazon Athena—AWS. Available online: https://aws.amazon.com/athena/ (accessed on 5 September 2024).
34. Elasticsearch, B.V. *What Is Elasticsearch? Elasticsearch Guide [8.15] Elastic*. Available online: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html (accessed on 5 September 2024).
35. Amazon Web Services, Inc. *Change Data Capture for DynamoDB Streams—Amazon DynamoDB*. Available online: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html (accessed on 5 September 2024).
36. Amazon Web Services, Inc. What Is AWS AppSync?—AWS AppSync. Available online: https://docs.aws.amazon.com/appsync/latest/devguide/what-is-appsync.html (accessed on 5 September 2024).

37. MongoDB Inc. *WiredTiger Storage Engine, WiredTiger*. Available online: https://www.mongodb.com/docs/manual/core/wiredtiger/ (accessed on 25 April 2024).
38. MongoDB Inc. *MongoDB Query Language, MQL*. Available online: https://www.mongodb.com/docs/manual/tutorial/query-documents/ (accessed on 25 April 2024).
39. MongoDB Inc. *Mongosh, MongoDB Shell*. Available online: https://www.mongodb.com/docs/mongodb-shell (accessed on 25 April 2024).
40. MongoDB Inc. *MongoDB Community, Community*. Available online: https://www.mongodb.com/community/ (accessed on 25 April 2024).
41. MongoDB Inc. *Real-Time Analytics*. Available online: https://www.mongodb.com/use-cases/analytics/real-time-analytics (accessed on 11 April 2024).
42. MongoDB Inc. *Real-Time Analytics Explained*. Available online: https://www.mongodb.com/basics/real-time-analytics-examples (accessed on 11 April 2024).
43. MongoDB Inc. *A Guide to ACID Properties in Database Management Systems*. Available online: https://www.mongodb.com/resources/basics/databases/acid-transactions (accessed on 8 September 2024).
44. Darel Lasrado, MongoDB: The Good, The Bad, and The Ugly, DZone.com. Available online: https://dzone.com/articles/mongodb-the-good-the-bad-and-the-ugly (accessed on 8 September 2024).
45. Migration from MongoDB to MySQL, Oracle. Available online: https://www.mysql.com/why-mysql/migration/mongodb/compare.html (accessed on 8 September 2024).
46. StackChief LLC. *Problems with MongoDB*. 9 December 2020. Available online: https://www.stackchief.com/blog/Problems%20with%20MongoDB (accessed on 25 April 2024).
47. MongoDB Inc. *MongoDB Limitations, Limitations*. Available online: https://www.mongodb.com/docs/manual/core/csfle/reference/limitations/ (accessed on 25 April 2024).
48. MongoDB Inc. *Business Intelligence (BI) Tools Overview*. Available online: https://www.mongodb.com/resources/basics/cloud-explained/business-intelligence-bi-tools (accessed on 8 September 2024).
49. RedSwitches Pte. Ltd. *Waleed, MongoDB Vs SQL Databases: A Comprehensive Comparison*. Available online: https://www.redswitches.com/blog/mongodb-vs-sql/ (accessed on 8 September 2024).
50. MongoDB Inc. *Query with Atlas SQL—MongoDB Atlas*. Available online: https://www.mongodb.com/docs/atlas/data-federation/query/query-with-sql/ (accessed on 8 September 2024).
51. MongoDB Inc. *6 Rules of Thumb for MongoDB Schema Design*. Available online: https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design (accessed on 8 September 2024).
52. Mateusz Papiernik, How to Design a Document Schema in MongoDB. 29 November 2021. Available online: https://www.digitalocean.com/community/tutorials/how-to-design-a-document-schema-in-mongodb (accessed on 4 April 2024).
53. MongoDB Inc. *Use Case Guidance: Where to use MongoDB*. Available online: https://www.mongodb.com/resources/solutions/use-cases/use-case-guidance-where-to-use-mongodb (accessed on 18 September 2024).
54. MongoDB Inc. *MongoDB Use Cases*. Available online: https://www.mongodb.com/solutions/use-cases (accessed on 18 September 2024).
55. Schmuecker, J.; Pan, W.Y.; Pullepu, S. How Leading Industries are Transforming with AI and MongoDB Atlas. 3 2024. Available online: https://www.mongodb.com/collateral/how-leading-industries-are-transforming-with-ai-and-mongo-db-atlas (accessed on 25 April 2024).
56. MongoDB. *MongoDB Cloud Services*. Available online: https://www.mongodb.com/products/platform/cloud (accessed on 25 April 2024).
57. MongoDB. *Introduction to Atlas App Services for Backend and Web Developers*. Available online: https://www.mongodb.com/docs/atlas/app-services/introduction/ (accessed on 25 April 2024).
58. MongoDB Inc. *MongoDB Community Edition*. Available online: https://www.mongodb.com/try/download/community-edition (accessed on 2 April 2024).
59. MongoDB Inc. *MongoDB Enterprise Server*. Available online: https://www.mongodb.com/try/download/enterprise (accessed on 2 April 2024).
60. Jain, H. Best 7 Real-World MongoDB Use Cases. 11 February 2022. Available online: https://hevodata.com/learn/mongodb-use-case/ (accessed on 7 April 2024).
61. MongoDB Inc. *Cloud Explained—What It Is and How It Works*. Available online: https://www.mongodb.com/resources/basics/cloud-explained (accessed on 18 September 2024).
62. MongoDB Inc. *IoT Architecture Explained*. Available online: https://www.mongodb.com/resources/basics/cloud-explained/iot-architecture (accessed on 18 September 2024).
63. MongoDB Inc. *Unleashing the Power of IoT*. Available online: https://www.mongodb.com/solutions/customer-case-studies/bosch (accessed on 19 September 2024).
64. MongoDB Inc. *How Toyota Material Handling Is Creating Smarter Factories by Moving to MongoDB Atlas*. Available online: https://www.mongodb.com/library/iot/how-toyota-material-hanldling-is-creating-smarter-factories?lb-mode=overlay (accessed on 19 September 2024).
65. MongoDB Inc. *Vaillant Solves Scaling IoT Challenges with MongoDB Atlas*. Available online: https://www.mongodb.com/library/iot/vaillant-solves-scaling-iot-challenges?lb-mode=overlay (accessed on 19 September 2024).

66. MongoDB Inc. *MongoDB for Internet of Things (IoT)*. Available online: https://www.mongodb.com/solutions/use-cases/internet-of-things (accessed on 19 September 2024).
67. MongoDB Inc. *Enterprise Advanced. The Best Way to Run MongoDB Yourself*. Available online: https://www.mongodb.com/products/self-managed/enterprise-advanced (accessed on 19 September 2024).
68. MongoDB Inc. *The Eternal Smart City Powered by Roma Capitale and MongoDB*. Available online: https://www.mongodb.com/solutions/customer-case-studies/roma (accessed on 19 September 2024).