IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 9, Issue 12, December 2020

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

A Comparative Exploration of Unstructured Data with SQL and NO-SQL Database Management Systems

Sakshi Sharma[1], Natasha Dutta[2]

Senior Technical Project Manager, Kforce Global Solutions Inc. [1]

Assistant Vice President, Security Operations, RBL Bank, Mumbai, India[2]

ABSTRACT: This paper aims to conduct a comparative study between a relational database, specifically Microsoft SQL Server, and a non-relational database, MongoDB, focusing on their handling of unstructured data represented in JSON format. While there has been extensive research comparing various database management systems in terms of performance, security, and other factors, there is limited information available that assesses these databases based on the nature of the data they manage. This study will primarily explore the different capabilities that both relational and non-relational databases offer when managing JSON data. To achieve this, we will carry out a series of experiments, ensuring that the data in question does not require normalization. The results of these experiments will then be analyzed to conclude.

KEYWORDS: JSON, SQL, RDBMS, MongoDB, Big Data, Social Media, Internet of Things (IoT), BSON (Binary JSON).

I. INTRODUCTION

The landscape of data management has evolved significantly over the past few decades, driven by the exponential growth of data generation across various industries. Traditionally, relational database management systems (RDBMS) like Microsoft SQL Server have been the cornerstone of data storage and retrieval. These systems are based on structured data models, where data is organized into tables with predefined schemas, making them ideal for scenarios requiring consistent data integrity, complex querying, and transactional accuracy. The structured nature of relational databases ensures that data is normalized, reducing redundancy and promoting efficient data management. Besides, they should allow for missing values to replace null values. RDBMS is generally known for their utility for data that requires transactional reliability and/or can be normalized [1]. This approach has been the backbone of enterprise-level applications, financial systems, and any environment where data consistency and reliability are paramount.

The rise of big data, social media, IoT (Internet of Things), and other modern data-driven applications has led to an explosion in the volume, velocity, and variety of data being generated. The unstructured data is usually populated in JSON or XML format, in which normalization is generally not required. It is noteworthy to know that with this type of data, when and why we should use an RDBMS model database application opposed to NoSQL or document-oriented database (e.g. SQL Server instead of MongoDB), as well as what will be the advantages and disadvantages [2]. Much of this data is unstructured or semi-structured, taking the form of text files, images, videos, and, notably, JSON (JavaScript Object Notation) documents. JSON, with its flexible, hierarchical structure, allows for the representation of complex data without the need for rigid schemas. This kind of data does not easily fit into the traditional rows and columns of relational databases, leading to the emergence and growing popularity of NoSQL (Not Only SQL) databases, such as MongoDB. NoSQL databases are designed to handle large volumes of diverse and dynamic data, offering greater flexibility, scalability, and performance in managing unstructured and semi-structured data formats.

The debate over the efficacy of SQL versus NoSQL databases in handling unstructured data has become increasingly relevant as organizations seek to leverage the vast amounts of data available to them. SQL databases like Microsoft SQL Server have adapted to some extent by incorporating support for JSON data, allowing for the storage and querying of JSON documents within a relational framework. This adaptation, however, raises questions about the performance and efficiency of SQL databases when dealing with unstructured data compared to their NoSQL counterparts, which are inherently designed for such tasks. MongoDB, as a leading NoSQL database, offers native support for JSON-like documents, known as BSON (Binary JSON), providing a more natural and flexible environment for working with unstructured data.

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

This study embarks on a comprehensive exploration of the strengths and weaknesses of both SQL and NoSQL database management systems in the context of unstructured data, specifically focusing on JSON formats. By conducting a series of experiments, this research aims to evaluate how these two types of databases handle unstructured data in terms of performance, scalability, flexibility, and ease of use. The experiments will be designed to reflect real-world scenarios where unstructured data plays a crucial role, providing insights into the practical implications of choosing one database management system over the other. The ultimate goal of this comparative study is to equip data professionals, system architects, and decision-makers with a clearer understanding of the trade-offs involved in using SQL versus NoSQL databases for unstructured data, thereby guiding them in making informed decisions that align with their specific data management needs.

RDBMS operates on a rigid, predefined structure called a schema, which must be established before any data is stored. This schema dictates that all data entries must conform to a uniform set of attributes, and any missing data is typically replaced with null values. RDBMS are well-suited for situations where data integrity and transactional accuracy are essential, especially in scenarios where data can be organized in a structured, normalized manner. However, NoSQL databases offer a more flexible approach. Unlike RDBMS, NoSQL databases do not rely on tables to store data. They use a simpler and more dynamic data model, allowing for a flexible schema that can easily accommodate changes. This flexibility makes NoSQL databases particularly effective at handling unstructured data, such as documents, multimedia files, emails, and social media content. Unstructured data is often stored in formats like JSON or XML, which do not require the data to be normalized, making NoSQL an ideal choice for managing such diverse and complex data types.

1.1 Databases

A database is a collection of data, information, and knowledge that is stored in an organized way, allowing for easy access, management, and updating. As new data is added, modified, or deleted, the existing data within the database is updated accordingly. Database systems function by querying the stored data and then running relevant applications to create or update the database as needed. Database systems work by querying the information or data present, and then executing relevant applications against it (Creating and updating themselves) [3]. Typically, computer databases hold a mix of data records or files, such as sales transactions, inventories, and product catalogs. Database management systems (DBMS) enable users to generate reports, manage read/write operations, and analyze usage patterns. As such, some databases follow the ACID compliance (Atomicity, Consistency, Isolation, and Durability) to ensure that the transactions are complete and the data is compatible [4].

1.2 SQL (Structured Query Language)

SQL, or Structured Query Language, is a standard programming language used in relational database management systems for performing essential operations on data. Developed in the 1970s, SQL has become an indispensable tool for database administrators [5]. It is also widely used by data analysts and developers for writing scripts for data integration and performing analytical queries.

SQL is primarily used for managing and modifying database structures, including tasks such as adding, deleting, and updating data, as well as retrieving specific data or subsets of data from a database for transaction processing and analytics applications [6]. SQL commands, known as statements, are typically written in a command form, with common examples being INSERT, SELECT, CREATE, UPDATE, ADD, DELETE, TRUNCATE, and ALTER.
Relational databases, also known as SQL databases, are structured as sets of tables containing data organized into columns and rows. Each column in a table represents a specific data category, such as name, age, or phone number, while each row contains values corresponding to these categories [7].

SQL has been the foundation for building both proprietary and open-source relational database management systems (RDBMS). These databases are widely used across organizations, with popular examples including Microsoft SQL Server, MySQL (owned by Oracle), Oracle Database, SAP Adaptive Server, IBM DB2, SAP HANA, and PostgreSQL [8]. Many of these database management systems are SQL-centric, incorporating specific extensions to the standard language for enhanced programming and related functions.

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

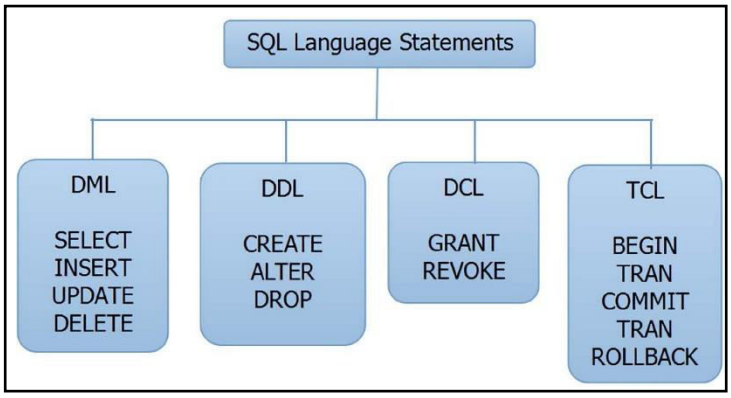

Figure 1: SQL Breakups

1.3 NoSQL (Not Only SQL Database)

NoSQL databases introduce a diverse range of data models, including graph, columnar, document, and key-value formats. "Not Only SQL," commonly referred to as NoSQL, offers an alternative to the traditional relational database approach, which relies on storing data in tables and carefully designing schemas before data can be inserted. When dealing with large sets of distributed data, NoSQL databases provide an optimal solution. Some databases that existed before the development of relational database management systems (RDBMS) are also considered NoSQL, but the term typically refers to databases developed in the early 21st century to manage extensive data clustering in web and cloud applications [8]. In these contexts, performance and scalability needs often surpass the rigid data consistency offered by SQL databases.

NoSQL databases were not created to follow the rules of relational schemas. Companies like Google, Amazon, and other large-scale web organizations have adopted NoSQL databases to focus on smaller operational goals while integrating relational databases where significant data consistency is crucial. Early NoSQL databases emphasized specific aspects of data management tailored for web and cloud applications, particularly the ability to efficiently distribute large volumes of data across computing clusters [10]. To support the rapid evolution of applications that require constant updates, developers opted for flexible schemas—or in some cases, no schema at all—enabling faster and more adaptable application development [11].

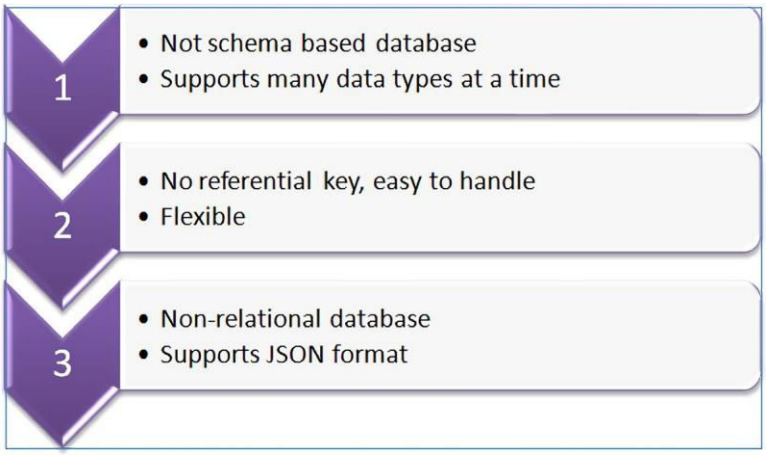

Figure 2: Properties of NoSQL databases.

1.4 Database Management System (DBMS)

A Database Management System (DBMS) is a type of system software designed to create and manage databases. It provides a structured approach for users and programmers to create, retrieve, update, and manipulate data. Essentially, a DBMS serves as an intermediary between the end users and the database, ensuring that data is consistently organized and readily accessible.

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

The DBMS is responsible for managing three core components: the data itself, the database engine that handles data retrieval, locking, and modification, and the database schema, which defines the logical structure of the database [9]. These components collectively ensure data concurrency, security, standardized administrative procedures, and integrity. Additionally, a DBMS facilitates essential database administration tasks such as backup and recovery, change management, and performance monitoring and tuning [13]. Many DBMS platforms are equipped with features for automatic restarts, transaction rollbacks, activity logging, and auditing, making them highly reliable for managing and safeguarding data.

1.5 MongoDB

MongoDB is a prominent NoSQL database known for its flexibility and scalability, designed to handle a wide range of data types and structures. Unlike traditional relational databases that rely on structured schemas and tables, MongoDB uses a document-oriented approach to data storage. In MongoDB, data is stored in JSON-like documents, which are organized into collections rather than tables. This schema-less design allows for dynamic and nested data structures, making it ideal for applications that require a flexible and evolving data model. Each document in MongoDB can contain a varying set of fields, allowing for different data types and structures within the same collection. This flexibility is particularly advantageous for handling unstructured or semi-structured data, such as multimedia files, logs, and user-generated content, which might not fit neatly into a traditional relational schema. MongoDB supports powerful querying and indexing capabilities, enabling efficient retrieval and manipulation of data. Its built-in support for horizontal scaling, through sharding, allows MongoDB to handle large volumes of data and high traffic loads by distributing data across multiple servers [15]. This scalability is crucial for modern applications that experience rapid growth and require reliable performance. Additionally, MongoDB offers features such as replication for high availability and automatic failover, ensuring data resilience and continuous access. The database also provides a rich set of tools for data management and analysis, including aggregation pipelines, which allow for complex data processing and transformation. MongoDB's focus on performance, scalability, and flexibility has made it a popular choice for a wide range of applications, from real-time analytics and big data processing to content management systems and IoT applications. Its ease of use and adaptability to changing data requirements position MongoDB as a powerful solution for developers seeking to build modern, data-intensive applications.

MongoDB is a document-oriented database that operates within the NoSQL framework and is known for its open-source nature. Emerging in the early 21st century, MongoDB distinguishes itself from traditional relational databases by eschewing the conventional tables and rows model. Instead, it employs a system based on documents and collections. In MongoDB, the fundamental unit of data is the document, which consists of key-value pairs. Collections, on the other hand, group these documents and function similarly to tables in a relational database. Unlike relational databases with fixed schemas, MongoDB features a highly dynamic schema design, allowing documents within a collection to have varying structures and fields. This flexibility supports diverse and evolving data requirements, making MongoDB well-suited for applications that need to handle a wide range of data types and formats.

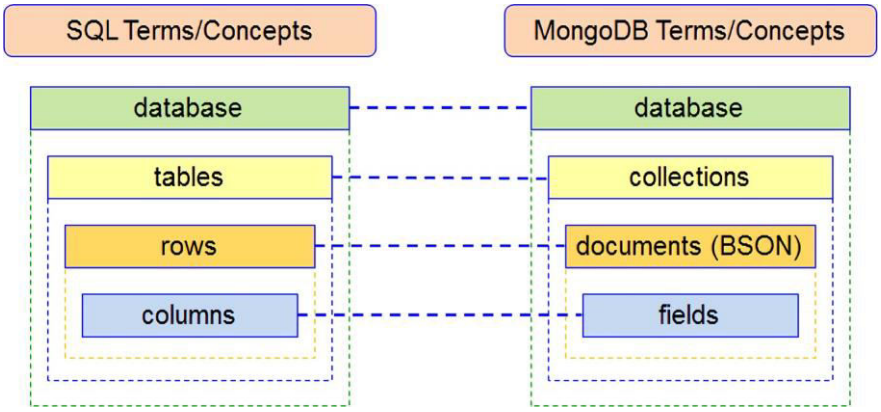

Figure 3: SQL Vs MongoDB terms

II. LITERATURE REVIEW

Yaron Gonen (2011) The increasing demand for applications that require the storage of vast amounts of data has led to the use of distributed databases, which offer high availability and scalability. In recent years, more companies have started using non-relational databases, also known as NoSQL databases, which have gained significant market interest

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

as the applications they support continue to grow. Unlike traditional relational databases, NoSQL databases are not designed to support full SQL functionality. Additionally, they often prioritize performance and scalability over consistency and security. However, as these databases are used to store increasingly sensitive data, security concerns have become more prominent. This paper examines two of the most popular NoSQL databases, Cassandra and MongoDB, and discusses their key security features as well as the challenges they face in this area [5].

N. Jatana (2012) Relational databases have been a cornerstone of data storage for decades. However, for modern interactive web and mobile applications, the need for flexibility and scalability in data models is increasingly important. The term NoSQL encompasses a range of non-relational databases that offer schema-less and scalable solutions. Often referred to as "Internet-age" databases, NoSQL systems are currently utilized by major organizations like Google, Amazon, and Facebook, which operate in the Web 2.0 era. NoSQL databases come in various types, including key-value stores, document databases, column-oriented databases, and graph databases. Each type allows developers to model data in a way that closely aligns with the application's needs. This paper explores the data modeling and query syntax of relational databases as well as several types of NoSQL databases. It includes a case study of a news website similar to Slashdot to illustrate these concepts [14].

Rene Xavier (2015) The rapid progress in high-throughput sequencing technologies has introduced significant computational challenges in the field of bioinformatics, particularly in managing the vast amounts of data generated by automated sequencers. One major challenge is the need to store and analyze these large-scale genomic datasets. The traditional relational database model may not always be the most effective solution for handling such massive and unconventional data, especially when it comes to efficiently writing and retrieving information. In this paper, we explore the use of the Cassandra NoSQL database as an alternative approach for storing genomic data. We analyze data persistency and input/output (I/O) operations using real genomic data within the Cassandra system. Author compare the performance of Cassandra with that of a traditional relational database system and another NoSQL database, MongoDB. This comparison helps to identify which database approach is more suitable for managing large-scale genomic data [16].

Snowflake and DynamoDB. It begins by outlining the key features, architecture, and use cases of each database. The analysis then moves into a detailed comparison, looking at aspects such as data models, scalability, performance, consistency, and cost-effectiveness. The paper also discusses strategies for optimizing data management with both Snowflake and DynamoDB, offering best practices for schema design, query optimization, data loading, and performance tuning. By evaluating Snowflake and DynamoDB in both SQL and NoSQL contexts, this paper aims to enhance the understanding of modern data management strategies. It provides guidance to organizations in selecting the database technology that best fits their needs. Additionally, the paper emphasizes the importance of adopting a data-driven approach to improve database performance, scalability, and cost-efficiency in today's fast-paced business environment. The paper examines the architecture and scalability of both databases, noting that Snowflake's columnar storage and its separation of compute and storage layers make it well-suited for analytical workloads. In contrast, DynamoDB's distributed architecture and flexible schema are designed for real-time, high-volume transactional applications. This comparison helps illustrate the strengths and best use cases for each technology [17].

III. METHODOLOGY

The methodology for this study involves a comprehensive comparative analysis of relational and NoSQL databases through a detailed case study of a news website similar to Slashdot. Initially, the study will select representative databases for each type: a relational database such as MySQL or PostgreSQL, and NoSQL databases including Redis or Riak (key-value store), MongoDB (document database), Apache Cassandra (column-oriented database), and Neo4j (graph database). The case study will design a simplified version of the news website, incorporating essential features like user accounts, news articles, comments, and categories. For the relational database, a normalized schema will be created with separate tables for users, articles, comments, and categories. In contrast, the NoSQL databases will use different approaches: the key-value store will handle session data and metadata with key-value pairs; the document database will model data with collections of flexible documents for articles, comments, and users; the column-oriented database will utilize column families for data storage; and the graph database will represent entities as nodes and their relationships as edges. Following schema design, the databases will be populated with sample data mimicking real-world usage patterns. A set of queries will then be executed to test data retrieval and manipulation, including fetching recent articles, retrieving comments for specific articles, searching by category or keyword, and managing user interactions. Performance evaluation will involve assessing each database's scalability by observing how well it handles increasing data volumes and user loads, as well as measuring query performance in terms of response times. The study will also explore data management and optimization strategies, such as indexing, caching, and data partitioning, to

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

enhance each database's efficiency. Comparative analysis will focus on the suitability of each data model, scalability, performance, flexibility, and cost-effectiveness. The final step will be to compile the findings into a detailed report, highlighting the strengths and weaknesses of each database type in the context of the case study, and offering recommendations based on the comparative results to guide organizations in selecting the most appropriate database technology for their needs.

IV. RESULTS

Experiment 1

In the first experiment, we inserted 100,000 entries into both a relational database (SQL Server) and a NoSQL database (MongoDB). To ensure a fair comparison, we used the same insertion method for both databases, as different methods could skew the results in favor of one system over the other. The experiment was conducted ten times, and the results are illustrated in Figure 4.

Table 1: Computer Specification

Operating System	Windows 10 Pro
Processor	AMD Quad-Core A8-5545M APU@ 1.7 GHz
Installed memory (RAM)	16.00 GB
Disk	1 TB 5400 rpm SATA

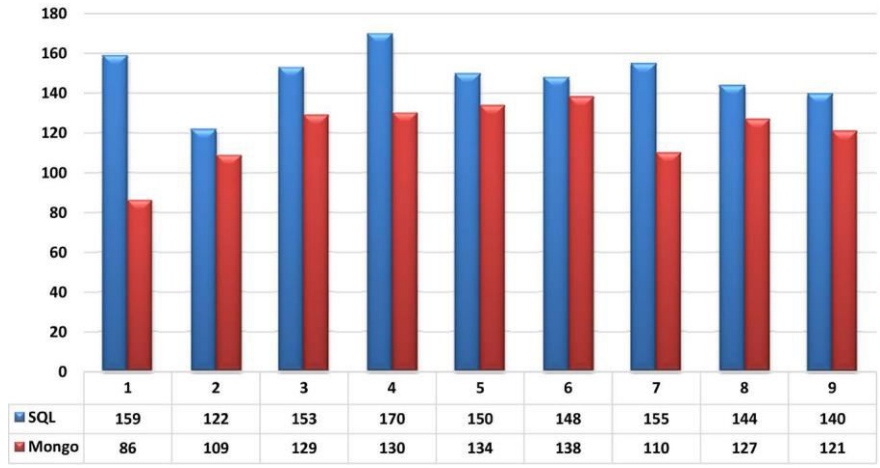

Figure 4: Result of Experiment 1.

From the analysis, it is evident that MongoDB outperformed SQL Server in terms of speed, generally completing the insertion process about fifteen seconds faster. Despite some variation in results—particularly during the 5th and 7th trials, where the performance of both databases was relatively similar—MongoDB consistently proved to be the faster of the two in all ten iterations of the experiment.

Experiment 2

In the second experiment, we performed ten searches for a random string in both databases and compared the results from each iteration. The findings are depicted in Figure 5, revealing some notable outcomes.

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

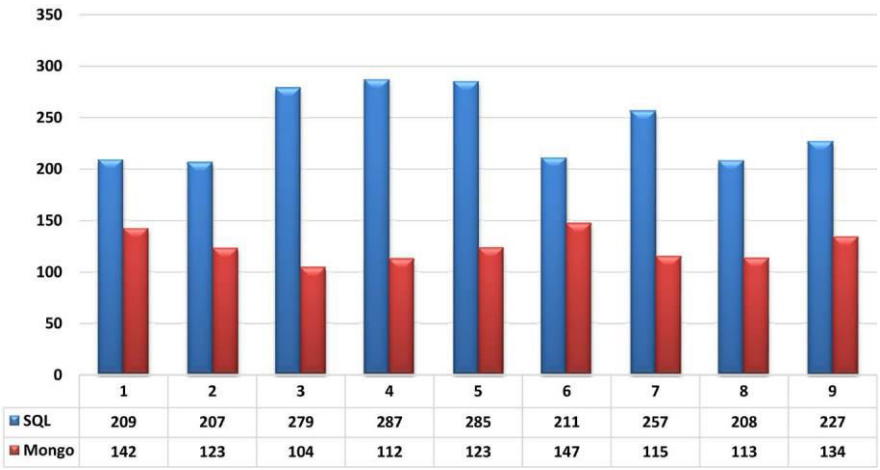

Figure 5: Result of Experiment 2.

The results of these experiments were striking and highlighted a significant disparity between the two types of databases. It is clear that searching for a specific string is notably more efficient and convenient in MongoDB compared to the other database.

V. DISCUSSION

The results of the experiments underscore a profound distinction between relational databases and NoSQL databases, particularly in the context of performance and efficiency. In the first experiment, where we inserted 100,000 entries into both SQL Server and MongoDB, MongoDB demonstrated a clear advantage, completing the task approximately fifteen seconds faster on average. This finding highlights MongoDB's superior speed and efficiency in handling large-scale data insertions, which is a crucial factor for applications requiring rapid data processing. The consistency of MongoDB's performance across the ten trials, despite some fluctuations in individual results, reinforces its reliability in such tasks. Similarly, the second experiment revealed MongoDB's exceptional capability in searching for a specific string. The efficiency and convenience of executing such searches in MongoDB were markedly superior compared to the relational database. This suggests that MongoDB's design, which is optimized for handling unstructured data and its flexible schema, significantly enhances its search capabilities. These observations reflect broader implications for database selection based on application needs. For scenarios involving large volumes of data and frequent search operations, MongoDB's performance benefits can be substantial. The contrast between the databases illustrates how the choice between SQL and NoSQL can impact operational efficiency and effectiveness, depending on the nature of the data and the specific requirements of the application. Therefore, organizations should carefully consider these factors when selecting a database system, as the efficiency and convenience of MongoDB in these experiments suggest it may offer substantial advantages for applications dealing with extensive data and complex search requirements.

VI. CONCLUSION

The experiments conducted underscore the significant differences in performance and efficiency between relational and NoSQL databases, with MongoDB showing notable advantages in both data insertion and search operations. The first experiment demonstrated MongoDB's superior speed in handling large-scale data insertions, completing the process approximately fifteen seconds faster than SQL Server on average. This advantage highlights MongoDB's strength in managing and processing substantial volumes of data, a critical factor for applications that require rapid and efficient data handling. Furthermore, the second experiment revealed that MongoDB excels in performing searches for specific strings, showcasing its efficiency and convenience compared to the relational database. The flexibility and optimized design of MongoDB for unstructured data and dynamic schema greatly enhance its search capabilities. These findings suggest that MongoDB's performance benefits make it a compelling choice for applications with extensive data and frequent search requirements. On the other hand, while relational databases like SQL Server continue to be valuable for their structured data and transactional reliability, the experiments indicate that MongoDB's strengths in scalability and speed are increasingly relevant in modern data-intensive applications. Consequently, organizations must carefully assess their specific needs, including data volume, search complexity, and performance requirements, when choosing between relational and NoSQL databases. The results of this study emphasize the importance of aligning database

ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.512| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 12, December 2020 ||

selection with the unique demands of the application, as MongoDB's demonstrated advantages in these experiments could lead to significant improvements in operational efficiency and effectiveness for suitable use cases.

REFERENCES

1. Sanobar, K. and Vanita, M. (2013) SQL Support over MongoDB Using Metadata. International Journal of Scientific and Research Publications, 3, 1-5. https://technet.microsoft.com/enus/library/bb522562(v=sql.105).aspx
2. MongoDB for Giant Ideas. "Database References". https://docs.mongodb.org/manual/reference/database-references
3. A Comparative Study of Databases with Different Methods of Internal Data Man-agementhttps://pdfs.semanticscholar.org/75b1/c2b5bd593b7a47786c6cf3cb8593554aa746.pd f
4. Jing, H., Haihong, E., Guan, L. and Jian, D. (2011) Survey on NoSQL Database. 2011 6th International Conference on Pervasive Computing and Applications (ICPCA), Port Elizabeth, South Africa, 26-28 October 2011, 363-366. https://doi.org/10.1109/ICPCA.2011.6106531
5. Han, J., Song, M., Gonen, Y. and Song, J. (2011) A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing. Proceedings of the 2011 10th IEEE/ACIS International Conference on Computer and Information Science, Sanya, 16-18 May 2011, 351-355. https://doi.org/10.1109/ICIS.2011.61
6. Tudorica, B.G. and Bucur, C. (2011) A Comparison between Several NoSQL Data- bases with Comments and Notes. 2011 10th Roedunet International Conference (RoEduNet), Iași, Romania, 23-25 June 2011, 1-5. https://doi.org/10.1109/RoEduNet.2011.5993686
7. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J. (2011) Security Issues in NoSQL Databases. 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 16-18 November 2011, 541-547. https://doi.org/10.1109/TrustCom.2011.70
8. Wei-Ping, Z. and Ming-Xin, L. (2011) Using MongoDB to Implement Textbook Management System instead of MySQL. 3rd IEEE International Conference on Communication Software and Networks (ICCSN), Xi'an, China, 27-29 May 2011. https://doi.org/10.1109/ICCSN.2011.6013720
9. Tauro, C.J.M., Aravindh, S. and Shreeharsha, A.B. (2012) Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Databases. Interna- tional Journal of Computer Applications, 48, No. 20. https://doi.org/10.5120/7461-0336
10. Edlich, S. (2011) List of NoSQL Databases. http://nosqldatabase.org
11. Leavitt, N. Will NoSQL Databases Live Up to Their Promise?
12. Strauch, C. NoSQL Databases. http://www.christofstrauch.de/nosqldbs.pdf
13. Bhat, U. and Jadhav, S. (2010) Moving Towards Non-Relational Databases. Interna- tional Journal of Computer Applications, 1, No. 13. https://doi.org/10.5120/284-446
14. Jatana, N., Puri, S., Ahuja, M., Kathuria, I. and Gosain, D. (2012) A Survey and Comparison of Relational and Non-Relational Databases. International Journal of Engineering Research &Technology, 1, 1-5. https://doi.org/10.1142/9781848168701_0002
15. MongoDB for Giant Ideas. "Database References". https://docs.mongodb.org/manual/reference/database-references.
16. Aniceto R, Xavier R, Guimarães V, Hondo F, Holanda M, Walter ME, Lifschitz S. Evaluating the Cassandra NoSQL Database Approach for Genomic Data Persistency. Int J Genomics. 2015;2015:502795. doi: 10.1155/2015/502795. Epub 2015 Oct 19. PMID: 26558254; PMCID: PMC4629038.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details