AKADÉMIAI KIADÓ

Pollack Periodica • An International Journal for Engineering and Information Sciences

19 (2024) 2, 30–35

DOI: 10.1556/606.2024.00990

ORIGINAL RESEARCH PAPER

Analysis for an open-source library in database management systems

Rama Hayek[1] and Mohammad Zaher Akkad[2*]

[1] Department of Software Engineering and Information Systems, Faculty of Information Engineering, University of Aleppo, Aleppo, Syria

[2] Faculty of Mechanical Engineering and Informatics, Institute of Logistics, University of Miskolc, Miskolc-Egyetemváros, Hungary

Received: November 21, 2023 • Revised manuscript received: February 2, 2024 • Accepted: February 8, 2024
Published online: May 13, 2024

ABSTRACT

Spatial data management is crucial for applications like urban planning and environmental monitoring. While traditional relational databases are commonly used, they struggle with large and complex spatial data. NoSQL databases provide support for unstructured data and scalability. This article compares the performance and disk space usage of SQL Server (a relational database) and MongoDB (NoSQL database) using an open-source library. Experiments conducted with the OpenStreetMap dataset from Central America show that the MongoDB database outperformed the relational SQL Server database in most cases, offering practical advantages for spatial data management in Geographic Information System applications.

KEYWORDS

spatial data, relational database, NoSQL, OpenStreetMap

1. INTRODUCTION

Geographic Information Systems (GIS) are computer-based systems used to manage, store, manipulate, analyze, and present spatial or geographic data [1]. GIS data typically involves many types of data, like maps, satellite images, and geospatial data [2]. The management and analysis of GIS data are traditionally done using Relational DataBase Management Systems (RDBMS) due to their strong data consistency, transactional support, and Atomicity, Consistency, Isolation, and Durability (ACID) properties [3]. Relational databases have been used as a strong base for managing and storing spatial data objects for close to two decades [4]. In today's world, with the increasing growth of geolocated and geospatial data ranging from satellite images with massive volumes to user-generated content that has varied formats, performance, and scalability challenges appeared with Structured Query Language (SQL) relational databases [3]. Due to these limitations, companies need to develop efficient ways to improve response time both when data is provided and retrieved. Non-relational SQL (NoSQL) databases are one of the techniques that have emerged in the recent past to handle requests for large datasets [2, 4]. Many organizations now use NoSQL to manage their GIS data [5]. NoSQL databases are more flexible and scalable and can handle large quantities of unstructured and semi-structured data, making them well-suited for managing geospatial data [6].

The objective of this article is to create an open-source library that enables the analysis of query performance and disk space utilization in two commonly used DataBase Management Systems (DBMSs): SQL Server and MongoDB. Both systems are capable of handling spatial data making them suitable for querying spatial datasets. This article contributes to existing literature by introducing an open-source library developed using C# programming language on a console application that can be utilized within DBMS or GIS modules. It also

*Corresponding author.
E-mail: qgezaher@uni-miskolc.hu

investigates and compares the query performance and disk space usage of relational and NoSQL databases, both of which are extensively employed in spatial analysis.

2. THEORETICAL BACKGROUND

Traditionally, geospatial databases as Oracle Spatial and PostGIS have been effective tools for managing geospatial data. They provide spatial data types, indexing capabilities, and spatial operations that are specifically designed for handling geospatial information. In the past decade, the availability of GPS-enabled devices has led to a significant increase in geospatial data production. This has resulted in the emergence of "Geospatial Big Data," which refers to the unstructured and semi-structured location-based data generated [7]. While SQL relational databases were traditionally effective for managing geospatial data, the rise of geospatial big data has necessitated more efficient handling and storage solutions. NoSQL databases have emerged as promising alternatives, offering improved performance and scalability for managing geospatial big data. These databases are specifically designed to handle large-scale and unstructured data, making them well-suited for the challenges associated with geospatial big data [8]. NoSQL databases do not use tables, rows, and columns to organize and store data. Instead, they use a variety of data models that do not require a predefined schema. Different types of NoSQL DBMSs are

- Key-Value Store (e.g., DynamoDB);
- Document Store (e.g., MongoDB and CouchDB);
- Wide Column Store/Column Families (e.g., Hadoop/HBase and Cassandra); and
- Graph Databases (e.g., Neo4J and OrientDB) [9].

NoSQL databases are considered one of the techniques to handle the emergent requirements of geo-big data [6]. To aid decision-making when selecting and optimizing geospatial database systems, comparative studies between SQL and NoSQL DBMSs have highlighted the varying strengths and limitations of different systems. In addition, there are several frameworks available for evaluating geospatial databases like GEOYCSB, Geographica, and GeoBenchmark Suite. Each framework offers its own set of features and performance metrics for assessing the efficiency and suitability of geospatial database solutions. Researchers have proposed frameworks that consider factors like query optimization, application performance estimation, and measuring disk space, helping to navigate the complex landscape of selecting an appropriate DBMS for GIS application. There are two main reasons behind relying on the DBMSs (SQL Server and MongoDB) [10–12]. First, they both support spatial data types. Second, they possess a large user base.

Starting with a brief literature review, there has been a growing interest in recent years in finding a balance between traditional relational databases and the emerging opportunities offered by NoSQL databases. As a result, several studies have been conducted to compare the performance of these two database types. For example, one study [13] specifically compared the performance of MongoDB, a popular NoSQL document-oriented database, with SQL database in terms of executing common queries. The findings of this study indicated that MongoDB often outperformed SQL in terms of speed and overall performance. Another study [11] focused on examining the performance of MongoDB and PostGIS, a spatial extension for SQL databases, in handling geospatial data. The researchers assessed the loading capabilities of both databases using a NodeJS-based web application that simulated large amounts of geospatial data. Although the results of the study indicated that MongoDB outperformed PostGIS in loading geospatial data, they might differ in real-world scenarios with actual geospatial data and complex queries. In 2018, the study [14] conducted a performance comparison between a document-based NoSQL database and a relational database for Voluntary Geographic Information System (VGIS) data storage architecture. The study suggested the advantages of the document-based NoSQL database in terms of feasibility and performance, but it lacks consideration of more complex types of queries and comparing metrics. Furthermore, the study [15] in 2020 provided a detailed analysis of Create, Read, Update, and Delete (CRUD) operations and their impact on application performance. The study compared the well-known MySQL database with CouchDB, a less-studied non-relational database. The analysis considered a complex database scenario with multiple joins and explored different data structures. This study used a comprehensive analysis of CRUD operations and query complexity. However, the number of records used is relatively small, which may not fully capture scalability and performance challenges at larger scales. Additional studies [4, 16, 17] also contributed to the understanding of the performance advantages of MongoDB in specific scenarios and query types. All these studies primarily focused on reading queries and did not extensively investigate writing operations.

Based on the research gaps in the previously mentioned literature, this article aims to provide a comprehensive evaluation of SQL Server and MongoDB by considering the complete CRUD operations and disk space utilization using the openly available OpenStreetMap (OSM) dataset of Central America. By analyzing both the query performance and disk space aspects, database management systems analysis is covered. The research involves the development of an open-source library for spatial query performance analysis. By comparing the spatial query performance of SQL Server and MongoDB, this research aims to provide valuable insights that can facilitate informed decision-making in the field of spatial data management. To sum up, the contributions of the article are as follows:

- The evaluation of the spatial query performance and disk space usage of SQL Server and MongoDB is provided, considering their support for spatial data types;
- An open-source library is developed that facilitates the analysis of query performance between these two-database management systems;

- The experiments are conducted on the openly available OSM dataset of Central America, ensuring the relevance and practicality of the findings;
- The outcomes of this article provide a valuable resource for organizations and practitioners seeking effective geospatial data management and decision-making processes;
- Outcomes have significant implications for GIS education, as the used methodology enables the teaching of both relational and NoSQL DBMSs using real-life datasets.

3. APPLIED METHODOLOGY

This article focuses on analyzing the query performance of CRUD operations on two well-known database management systems: SQL Server and MongoDB. The methodology involves several steps to ensure a comprehensive analysis. The first step is to import the openly available OSM locations dataset of Central America into the SQL Server. This initial import is crucial to ensure that each record in the dataset has a unique identifier, which may not be readily available in the raw data. Next, the OSM dataset is converted into GeoJSON objects, which serve as the basis for importing the data into MongoDB. This conversion step allows for a comparative analysis between SQL Server and MongoDB, as both databases will have access to the same dataset in a compatible format.

In the experiments, real-world mechanisms used by businesses and enterprises were simulated by considering the actual transaction registration model employed by database engines. Standard commands were used like INSERT instead of more efficient instructions like Bulk Insert to accurately reflect real-world data insertion patterns [13]. This principle applies to other basic commands as well, like DELETE and UPDATE. By simulating the transaction registration model commonly used by businesses, the performance of SQL Server and MongoDB is evaluated accurately under realistic conditions.

Regarding the querying of data, queries are divided into two categories: selecting all data and *k*-Nearest Neighbors (kNN) queries [17]. Selecting all data is important for data verification, exploration, and performance benchmarking, allowing researchers to ensure data accuracy, understand its structure, and establish a performance baseline. On the other hand, kNN queries are essential for spatial analysis and location-based applications, enabling the evaluation of database systems' efficiency and accuracy in handling proximity searches. By dividing the queries into these categories, the article covers fundamental data retrieval and spatial analysis aspects, providing a comprehensive assessment of the database systems' performance and capabilities.

The final test involves measuring the disk space utilization of the two databases: SQL Server and MongoDB. This test aims to evaluate the storage efficiency and space consumption of each database system. By comparing the disk space requirements of the two databases, researchers can assess their effectiveness in managing and storing the dataset from a storage utilization perspective. This analysis provides insights into the efficiency of data storage and can help inform decisions regarding database sizing, resource allocation, and cost optimization. By incorporating these considerations into the experimental design, it was ensured to align closely with the actual mechanisms and challenges faced by GIS businesses and applications.

4. EXPERIMENTS AND ANALYSIS

The aim is to develop an open-source library that facilitates systematic query performance and disk storage analyses between SQL Server and MongoDB. Conducting a comparative study involves performing multiple experiments to gather results for analysis. These results are then used to evaluate and discuss the performance characteristics and disk space utilization of the two database systems. The applied source code can be found on GitHub [18].

4.1. Application data

The data was extracted from the OSM [19], which is a free and open-source project that aims to create a map of the world using crowdsourced data. The OSM dataset is a collection of geospatial data that has been contributed by users all over the world. This dataset includes information on roads, buildings, land use, points of interest, and other geographic features. The OSM dataset is available in a variety of formats, including XML and PBF (used in the article).

The OSM dataset is used by various organizations and individuals for various applications, including navigation, urban planning, disaster response, and environmental analysis. It offers a collaborative and constantly updated source of geospatial data [20]. The experiments were executed on four OSM datasets that contain different numbers of location points. The numbers of location points in all datasets are as follows: (1,000,000, 10,000,000, 20,000,000, 30,000,000). All experiments were conducted on a CORE i5, 12 GB RAM laptop with a Windows 11, 64 bit operating system.

4.2. Applied tests

In the study, the OSM data was stored in both SQL Server and MongoDB (utilizing the 2dsphere index for efficient spatial querying). Additionally, no constraints were applied to the databases. The execution of queries was performed on a single node without distributed computing. The following presents the results of the experiments, as well as the analysis. The limitations of the experiments are also presented and discussed.

4.2.1. Performance analysis. Experiments in this research include the insert, update, delete, and comprehensive select queries on the two database engines. First, the SQL language queries were designed to run on an MS SQL Server database. Then the queries were converted to the same queries in

MongoDB syntax to run in MongoDB. For evaluating the performance operations, 10,000 data points have been chosen as an average sample for the number of records, which is being selected in most of the related studies. The comparison of the taken time to run the queries on both databases will be shown in the experimental results. The execution times were taken in microseconds in both relational and NoSQL databases.

In the following, one subsection is assigned to each experiment, in which the desired operation or query is expressed, and the results are illustrated.

4.2.2. Insert operations. The results of the insert operations in both databases in the previously mentioned size scales, along with 10,000 records at a time, indicate that SQL Server performs insert operations at a higher speed compared to MongoDB with SQL Server being approximately two times faster in completing the insert operations. In fact, MongoDB's insert performance can be optimized when inserting entire documents at once, as it eliminates the need for individual row-level processing. On the other hand, SQL Server's relational structure requires explicit data type definitions for each column, which can lead to faster insert performance in certain scenarios, especially when inserting one record at a time. The results are shown in Fig. 1.

4.2.3. Delete operations. The delete operations have also been evaluated in these experiments. The results in Fig. 2 show that MongoDB outperforms SQL by a factor of about 13 times when the number of records reaches 30 million.

4.2.4. Update operations. Update operations were performed with the same database size scales in this subsection. As can be seen from the results, the MongoDB database engine has absolute superiority especially when data sizes increase. The results are shown in Fig. 3.

4.3. Querying in different modes

Regarding the query on the data, as mentioned earlier, queries have been divided into two main queries (Q1 and Q2), and the results are to be reviewed.

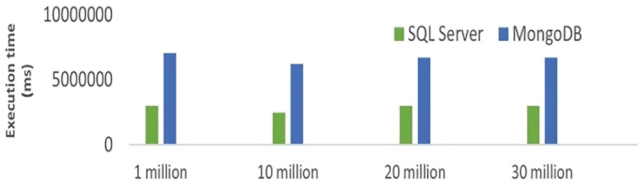

Fig. 1. INSERT execution times in SQL Server and MongoDB

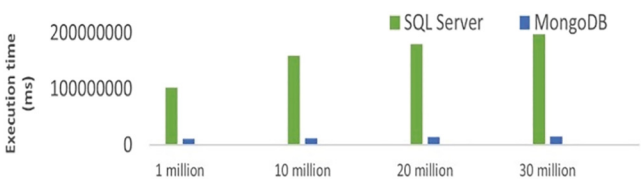

Fig. 2. DELETE execution times in SQL Server and MongoDB

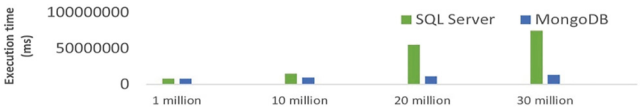

Fig. 3. UPDATE execution times in SQL Server and MongoDB

4.3.1. Q1 loading all locations. The goal of the first query is to retrieve all locations from both the SQL Server and MongoDB databases. This query helps assess the efficiency of retrieving and handling a large volume of geospatial data from the NoSQL database. By testing the query's performance, scalability, and response time, GIS analysts can evaluate the database's ability to handle and process extensive geospatial datasets. This query is particularly useful in assessing the overall data loading capabilities and ensuring that the NoSQL database can effectively handle the entire geospatial dataset without any bottlenecks or performance issues.

4.3.2. Q2 k nearest neighbors' analysis. kNN queries, which involve finding the nearest neighbors of a query instance, are widely used in GIS and other fields for tasks like data classification and clustering. Researchers have dedicated significant attention to optimizing kNN operations. This subsection discusses the experimental setup and findings of a kNN query comparison between SQL Server and MongoDB. In this experiment, a random point is selected, and its nearest neighbors are determined based on its coordinates.

The k of the nearest 1,000 was selected as an average value of k as in the related studies [16, 17]. The aim is to find the nearest points to a specific given point, ordered by distance. This particular query is essential for assessing the efficiency of a database in conducting proximity-based searches, which is crucial in various geospatial analysis tasks. These tasks often involve finding nearby locations, performing spatial clustering, or executing spatial joins. By evaluating the response time and accuracy of the query, GIS analysts can determine the database's suitability for handling geospatial analysis tasks that heavily rely on proximity-based operations.

In SQL Server, to calculate the neighbors' points, using spatial extensions like the Geometry Data Type and spatial functions for distance like Euclidean distance would typically be needed. On the other hand, in MongoDB, the $near operator is used to find and sort the documents based on their proximity to the specified point. The results are then limited to 1,000 points using the limit function. For the first query Q1, the average execution times are cleared in the following figures.

After analyzing the graph in Figs 4 and 5, it becomes evident that MongoDB consistently outperforms SQL Server in all scenarios, ranging from small datasets to large datasets. For the second query Q2, query execution times are presented as follows.

Also, from Figs 6 and 7, it is observed that the kNN loading time for MongoDB in Q2 is significantly smaller

Fig. 4. Q1 average execution times in SQL Server

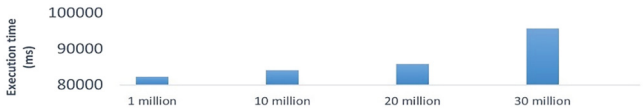

Fig. 5. Q1 average execution times in MongoDB

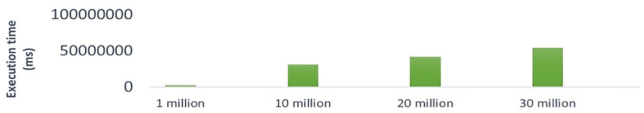

Fig. 6. Q2 average execution times in SQL Server

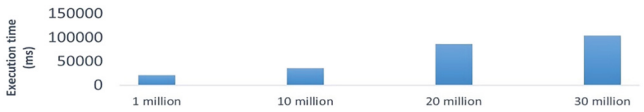

Fig. 7. Q2 average execution times in MongoDB

Fig. 8. Comparing disk space between SQL Server with MongoDB (size in GB)

compared to that of MS SQL Server as the number of data grows. On smaller datasets, the response time difference is less significant compared to the larger dataset.

4.4. Disk space analysis

Regarding disk space usage, the NoSQL proved to be a better solution (see Fig. 8). MongoDB demonstrates more efficient disk space usage compared to SQL Server for the same amount of data, primarily due to its flexible, schema-less format and the absence of relational overhead such as transaction logs and metadata management.

5. CONCLUSIONS

The research conducted in this article aimed to investigate the performance and disk space utilization in GIS applications using both relational and NoSQL databases. The results consistently demonstrated that MongoDB, a NoSQL database, outperformed relational databases in most cases. The superior performance of MongoDB makes it an advantageous choice for handling the complex querying needs of these systems, where real-time data retrieval and processing are critical.

The findings are specific to the investigated versions of SQL Server and MongoDB but remain valuable for teaching GIS based on relational DBMSs. The developed library has the potential to teach both relational and NoSQL DBMSs using real-life datasets and can be expanded to support other DBMSs. Future research directions include analyzing more spatial data types like line and multi-line objects. The article can also propose expanding the test scope to encompass important spatial features, as geometric operations, spatial indexing strategies, and support for 3d/4d spatial data to further evaluate the practical applicability and performance of the geospatial databases in diverse scenarios.

REFERENCES

[1] H. Goyal, C. Sharma, and N. Joshi, "An integrated approach of GIS and spatial data mining in big data," *Int. J. Comput. Appl.*, vol. 169, no. 11, pp. 1–6, 2017.

[2] W. Tampubolon, W. Reinhardt, S. Sumaryono, and S. T. L. Tobing, "NoSQL standard and approach for geospatial database collection," in *Seminar Nasional Geomatika*, Cibinong, Indonesia, April 14, 2021, pp. 321–326.

[3] E. Baralis, A. D. Valle, P. Garza, C. Rossi, and F. Scullino, "SQL versus NoSQL databases for geospatial applications," in *IEEE International Conference on Big Data*, Boston, MA, USA, December 11–14, 2017, pp. 3388–3397.

[4] S. Agarwal and K. S. Rajan, "Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries," in *Conference Proceedings on Free and Open-Source Software for Geospatial*, vol. 17, Boston, USA, September 20, 2017, Art no. 2.

[5] M. Hasan, E. Panidi, and V. Badenko, "Comperative evaluation of NoSQL and relational databases performance while analysing semi structured GeoSpatial Data," in *International Scientific Conference GEOBALCANICA*, Saint Petersburg, Russia, August 22, 2019, pp. 541–549.

[6] D. Guo and E. Onstein, "State-of-the-art geospatial information processing in NoSQL Databases," *ISPRS Int. J. Geo-Information*, vol. 9, no. 5, 2020, Art no. 331.

[7] J. G. Lee and M. Kang, "Geospatial big data: Challenges and opportunities," *Big Data Res.*, vol. 2, no. 2, pp. 74–81, 2015.

[8] Z. Liu, H. Guo, and C. Wang, "Considerations on geospatial big data," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 46, 2016, Art no. 012058.

[9] J. K. Chen and W. Z. Lee, "An introduction of NoSQL databases based on their categories and application industries," *Algorithms*, vol. 12, no. 5, 2019, Art no. 106.

[10] T. Jia, X. Zhao, Z. Wang, D. Gong, and G. Ding, "Model transformation and data migration from relational database to MongoDB," in *Proceedings of IEEE International Congress on Big Data*, San Francisco, CA, USA, June 27–July 02, 2016, pp. 60–67.

[11] D. Laksono, "Testing spatial data deliverance in SQL and NoSQL database using NodeJS fullstack web app," in *Proceedings of 4th*

International Conference on Science and Technology, Yogyakarta, Indonesia, August 7–8, 2018, pp. 1–5.

[12] Geospatial queries, 2023. [Online]. Available: https://www.mongodb.com/docs/manual/geospatial-queries/. Accessed: Nov. 1, 2023.

[13] S. H. Aboutorabi, M. Rezapour, M. Moradi, and N. Ghadiri, "Performance evaluation of SQL and MongoDB databases for big e-commerce data," in *International Symposium on Computer Science and Software Engineering*, Tabriz, Iran, August 18–19, 2015, pp. 1–7.

[14] D. C. M. Maia, M. Holanda, and B. D. C. Camargos, "Performance analysis on voluntary geographic information systems with document-based NoSQL Database," in *Proceedings on Developments and Advances in Intelligent Systems and Applications*, Maristela, Holandia, Maristela, January 1, 2018, pp. 181–197.

[15] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Győrödi, G. A. Gabor, and G. D. Pecherle, "Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application's data storage," *Appl. Sci.*, vol. 10, no. 23, 2020, Art no. 8524.

[16] İ. B. Coşkun, S. Sertok, and B. Anbaroğlu, "K-nearest neighbor query performance analysis on a large-scale taxi dataset: PostgreSQL vs. MongoDB," *Int. Arch. Photogrammetry, Remote Sensing Spat. Inf. Sci.*, vol. XLII-2/W13, pp. 1531–1538, 2019.

[17] B. Anbaroğlu and A. Mobasheri, "Spatial query performance analyses on a big taxi trip origin-destination dataset," in *Proceedings on Open Source Geospatial Science for Urban Studies, Lecture Notes in Intelligent Transportation and Infrastructure*, vol. 2020, pp. 37–53.

[18] Applied source code, 2024. [Online]. Available: https://github.com/remihk94/source_lib. Accessed: Jan. 30, 2024.

[19] OpenStreetMap, 2023. [Online]. Available: https://www.openstreetmap.org/about. Accessed: Nov. 1, 2023.

[20] OpenStreetMap and its use as open data, 2023. [Online]. Available: https://www.e-education.psu.edu/geog585/node/738. Accessed: Nov. 1, 2023.