IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol-6, Special Issue-6, July 2018 E-ISSN: 2347-2693

Analysis of Aggregate Functions in Relational Databases and NoSQL Databases

Benymol Jose[1*], Sajimon Abraham[2]

[1]School of Computer Sciences, Mahatma Gandhi University, Kottayam, India
[2]School of Management and Business Studies, Mahatma Gandhi University, Kottayam, India

[*]*Corresponding Author: benymol.jose@mariancollege.org, Tel.: +0091-9995237344*

Available online at: www.ijcseonline.org

Abstract— The attractions in Big Data Analytics made a progress from relational databases to NoSQL databases. A NoSQL structure can be utilized to enhance the distribution of storage and analysis work of data in the world of big data. MongoDB is a type of NoSQL database which represents data as a collection of documents. Ordinary database systems like MySQL can store only organized data in tabular form as rows and columns. As the majority of the data created now is in unstructured or semi structured format, it is difficult for conventional database systems to store or process this data. NoSQL data stores like MongoDB can store this huge data which additionally have very powerful query engines and indexing features. These features made it simple and fast to execute extensive variety of queries including aggregate ones. The aggregation pipeline and map reduce concepts in MongoDB provides support for aggregate operations. This paper primarily makes a comparison of performance of aggregate queries in MySQL and MongoDB. A set of experiments were performed with two datasets of different size in the two databases. The results show that MongoDB performs better in all the cases. The results can be a boost for companies to change the structure of their databases from conventional form to NoSQL.

Keywords— Relational Databases, NoSQL Databases, MongoDB, MySQL, Aggregation.

I. INTRODUCTION

The relational model has ruled the computer industry since the 1980s for the most part to store and retrieve data. Gradually, relational database systems lost its highlight because of the existence of a rigid schema. This inflexibility caused the difficulty in making new relationships between the entities [1]. Another critical reason of its failure is that as the accessible data is coming in varying formats, it is getting difficult to process this huge volume of data with relational model. This is because of the time consumed in joining a large number of tables [2].

Due to the boom of huge mass of unstructured data in these years, non-conventional databases like MongoDB are coming up to manage the issues which exists in connection with conventional databases. MongoDB is exceptionally valuable which can even replace the existing conventional relational foundation [3].

MongoDB is a document-based NoSQL database developed by MongoDB Inc, which is accessible as an open source. MongoDB systems use documents and collections instead of the tables used in traditional databases. JSON, BSON based documents or sub documents are the fundamental components of the collections in MongoDB. The capacity issues of traditional database systems related with managing huge volumes of unstructured data can be rectified by using NoSQL systems like MongoDB [4].

An aggregation is a task that passes through a set of records and does a calculation on a group of values and gives a single value such as sum, average, max, min or count as result. Often, people are interested in summarizing data to determine trends or produce top level reports which can help in decision making in commercial organizations. Aggregate functions can assist with the summarization of large volumes of data [5]. These aggregation tasks can be achieved with both relational databases like MySQL and document-based NoSQL databases like MongoDB. In a distributed environment, each partition gives a fractional outcome and after that these results are aggregated productively so that it can give one response to the application and client. The different aggregate functions and operations available are count, min, max, sum, distinct, group, sort etc.

The main aim of this paper is to make the comparison and analysis of performance of queries using aggregate functions both in MongoDB and MySQL and to see which one performs well. The aggregate queries are executed with two datasets of different sizes and the results show that MongoDB performs well for both the cases. This is a motive for business organizations to shift from conventional

databases to NoSQL databases for easy storage and fast decision making.

The paper is composed as follows: section II, depicts few related studies and in section III, MongoDB Query Aggregation options are compared with that of MySQL. In section IV, an evaluation on performance of both the databases is made by executing different queries with aggregate functions and the time taken for the execution is noted. Finally, in section V, graphs are drawn and the performance is analyzed and the paper is concluded with the comments on the comparison of aggregate query performance in MySQL and MongoDB.

II. RELATED WORK

These days, there have been lot of discussions happening worldwide about the performance of SQL and NoSQL databases. But very few researches have occurred with the performance comparison of SQL and NoSQL databases with large datasets and simple queries. Here, we are referring to few studies and works happened in this area. In a recent study, a method was proposed to combine the properties of MySQL and MongoDB by adding a middleware between the layers of application and database. It consists of metadata which includes different types of packages [6]. In another contribution, different database operations were performed in the SQL and NoSQL databases with the same dataset for an e-commerce system. And it is concluded that MongoDB does better for all operations except for few aggregate operations [7]. In another work, attempt is made to utilize NoSQL database in place of the relational database. It is applied to traditional information management systems, compared the two database technologies, gave the key code of NoSQL implementation, and finally listed the performance comparison of the two schemes [8]. Another research is endeavoring to evaluate the execution speed of five NoSQL databases (Redis, MongoDB, Couchbase, Cassandra, HBase) with an evaluating device called - YCSB [9].

III. MongoDB Query Aggregation Options

There are two different aggregation methods available in MongoDB. They are the aggregation pipelining and map reducing methods.

A. Aggregation Pipeline in MongoDB

The aggregation pipeline's working is similar to that of the pipe command in Unix. The query and the diagram explaining the working of the pipeline is shown in the following figure, Figure 1.

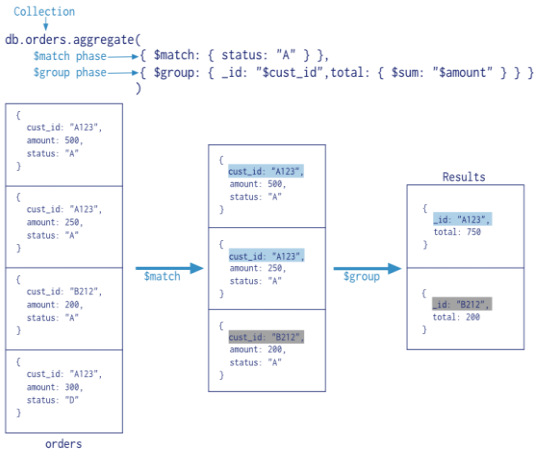

Figure 1. Aggregation Pipeline from Mongodb.com documentation

As the documents are moved through the pipeline, filtering and changes can be applied as required on every operator. Skip, match, and sort are the various functions that each pipeline operator is supposed to perform. The performance can be improved by reducing the quantity of the of data being processed by applying filtering at the beginning of the pipeline itself [10].

B. Map-Reduce in MongoDB

The automatic query processing flexibility which is not included in aggregation pipeline is associated with the map-reduce capability of MongoDB. It uses massively parallel processing to manage the data whose size is very big. The map-reduce codes are executed with a particular command 'map-Reduce' in MongoDB. In this, before applying filtering and gathering of documents with the reduce task, documents are matched with the map task.

The reduce process considers a group of (key, value) pairs as input which is produced as the output of the map procedure. Normally the input to a map procedure is a collection and the reduce procedure can give a collection as output or can be returned inline. Also, if the reduce process outputs a collection it can act as input to one more map-reduce process [10]. The working of map reduce process is demonstrated in the following figure, Figure 2.

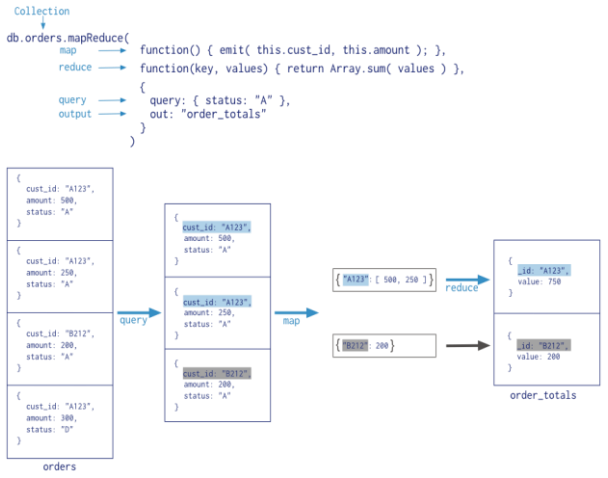

Figure 2. Map-Reduce from Mongodb.com documentation

IV. PERFORMANCE EVALUATION

The experiments are performed in a machine running with Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, with Windows10, 64-bit operating system. The machine has 8GB of physical memory and 256GB of SSD hard disk space. The MySQL Workbench 6.3 and MongoDB Studio 3T 3.4.5 are the respective softwares used to test the data. The data is first collected in Studio 3T and then migrated to MySQL Workbench with the import option available there. Queries with different aggregate functions are executed on both the databases and the execution time is recorded.

The first dataset utilized comprises of the details of permits issued by the Department of Buildings in the City of Chicago from 2006 to the present. The dataset for every year contains in excess of 65,000 records. Furthermore, the dataset taken for the experiment and analysis consists of around 1.5 lakhs of records. It is denoted as D1.

The second data set considered consists of radiation and environmental data from all over the world which begun in response to the nuclear disaster in Japan in March, 2011. It consists of around 10 lakhs 50 thousand records of 1GB size. To denote it in experiments the name D2 is used.

The comparison between the databases is done by performing a series of queries using aggregate functions like sum, avg, min and max. The time taken for the execution of these queries is noted in the corresponding tables for both the datasets. To avoid complication, only queries with dataset D1 are shown.

A. Sum() Function

The sum function is often applied to a set of numeric values and it returns the sum of those values. The following table, Table 1 shows the queries used in both MySQL and MongoDB for the aggregate function sum and the time taken for query execution in case of D1 and D2.

Table 1. Query and Execution time for MySQL and MongoDB for the aggregate function sum with D1 and D2.

Query in MySQL	**Query in Studio 3T**	**Time in Milliseconds**			
		D1		**D2**	
		MySQL	**MongoDB**	**MySQL**	**MongoDB**
SELECT STREET_NUMBER,sum(ID) FROM building GROUP BY STREET_NUMBER;	**db.building.aggregate([{$group:{_id:"$STREET_NUMBER", num:{$sum: "$ID"}}}])**	**578**	**40**	**21641**	**2476**

Here documents were grouped based on the field "street_number" and on each appearance of street_number, the existing value of sum is updated.

B. Avg() Function

The avg () function is applied to a set of numeric values and it returns the average or mean of those values. The following table, Table 2 shows the queries used in both MySQL and MongoDB for the aggregate function avg and the time taken for query execution in case of D1 and D2.

Here documents were grouped based on street_number and the average of their ID values are taken.

Table 2. Query and Execution time for MySQL and MongoDB for the aggregate function avg with D1 and D2.

Query in MySQL	Query in Studio 3T	Time in Milliseconds			
		D1		D2	
		MySQL	MongoDB	MySQL	MongoDB
select avg(ID) from building group by street_number;	db.building.aggregate([{$group:{_id:"$STREET_NUMBER", num:{$avg:"$ID"}}}])	516	63	18656	2016

C. Max() Function

This aggregate function is applied to a set of numeric values and it returns the maximum from a group. The following table, Table 3 shows the queries used in both MySQL and MongoDB for the aggregate function max and the time taken for query execution in case of D1 and D2.

Table 3. Query and Execution time for MySQL and MongoDB for the aggregate function max with D1 and D2.

Query in MySQL	Query in Studio 3T	Time in Milliseconds			
		D1		D1	
		MySQL	Mongo DB	MySQL	Mongo DB
select max(ID) from building group by street_number;	db.building.aggregate([{$group:{_id:"$STREET_NUMBER",num:{$max:"$ID"}}}])	484	87	19500	2019

In this, documents are grouped based on street number and the maximum ID value is chosen from it.

D. Min() Function

Here also the aggregate function min is applied to a set of numeric values and it returns the minimum of those values. The following table, Table 4 shows the queries used in both the databases for the aggregate function min and the time taken for query execution in case of D1 and D2.

Table 4. Query and Execution time for MySQL and MongoDB for the aggregate function min with D1 and D2.

Query in MySQL	Query in Studio 3T	Time in Milliseconds			
		D1		D2	
		MySQL	Mongo DB	MySQL	Mongo DB
select min(ID) from building group by street_number;	db.building.aggregate([{$group:{_id:"$STREET_NUMBER",num:{$min:"$ID"}}}])	484	114	21641	1998

In this, documents were grouped based on street_number and the minimum ID value is chosen from it.

V. Analysis and Evaluations

The performance of relational and NoSQL databases is compared with MySQL Workbench 6.3 and MongoDB Studio 3T. For the different aggregate functions, sum, avg, min and max, the queries are executed with D1 and D2 and the time taken for the executions is noted in the following tables, Table 5, Table 6 and combinedly in Table 7.

Table 5: Execution time for query in MySQL and MongoDB with the different aggregate functions for D1

Aggregate Function	**Time in Milliseconds**	
	D1	
	MySQL	**MongoDB**
sum()	**578**	**40**
Avg()	**516**	**63**
Max()	**484**	**87**
Min()	**484**	**114**

Table 6: Execution time for query in MySQL and MongoDB with the different aggregate functions for D2

Aggregate Function	**Time in Milliseconds**	
	D2	
	MySQL	**MongoDB**
sum()	**21641**	**2476**
Avg()	**18656**	**2016**
Max()	**19500**	**2019**
Min()	**21641**	**1998**

Table 7: Combined execution time for query in MySQL and MongoDB for the different aggregate functions with D1 and D2

Aggregate Function	**Time in Milliseconds**			
	D1		**D2**	
	MySQL	**MongoDB**	**MySQL**	**MongoDB**
sum()	**578**	**40**	**21641**	**2476**
Avg()	**516**	**63**	**18656**	**2016**
Max()	**484**	**87**	**19500**	**2019**
Min()	**484**	**114**	**21641**	**1998**

Based on the data given in Table 5 and Table 6, graphs are plotted and is shown in the following figures, Figure 3 and Figure 4.

On analyzing and looking at it, it is clear that, the execution of MongoDB is great compared with that of MySQL. The increased performance of the Aggregation Pipeline is used to accomplish this.

But there is not much difference in the execution time taken by both relational and NoSQL databases in case of aggregate queries. In any case, MongoDB shows great improvement by taking less time for the completion of queries using aggregate functions compared to MySQL which is a relational database.

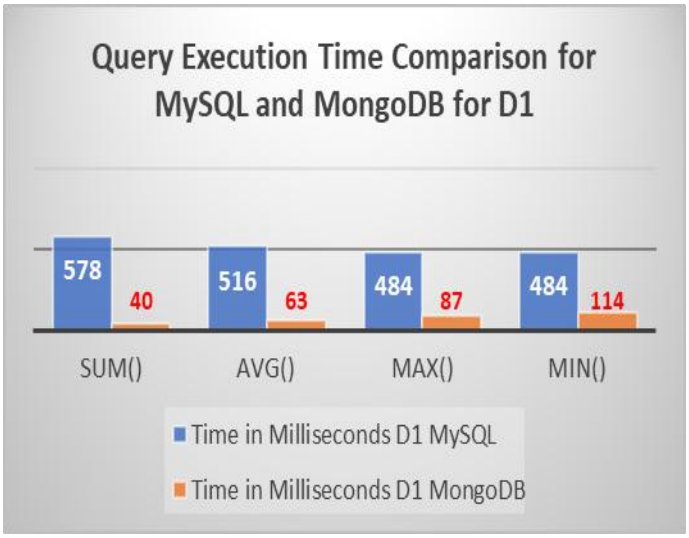

Figure 3: Performance comparison of MySQL Workbench with MongoDB Studio3T for the aggregate functions with D1.

Figure 4: Performance comparison of MySQL Workbench with MongoDB Studio3T for the aggregate functions with D2.

Also, in the following tables, Table 8 and Table 9 the performance of MySQL and MongoDB is shown separately for the two datasets D1 and D2. The corresponding graphs are drawn and shown in figures, Figure 5 and Figure 6.

Table 8: Execution time for query in MySQL for the different aggregate functions with D1 and D2

Aggregate Function	Time in Milliseconds	
	D1	D2
	MySQL	MySQL
sum()	578	21641
Avg()	516	18656
Max()	484	19500
Min()	484	21641

Table 9: Execution time for query in MongoDB for the different aggregate functions with D1 and D2

Aggregate Function	Time in Milliseconds	
	D1	D2
	MongoDB	MongoDB
sum()	40	2476
Avg()	63	2016
Max()	87	2019
Min()	114	1998

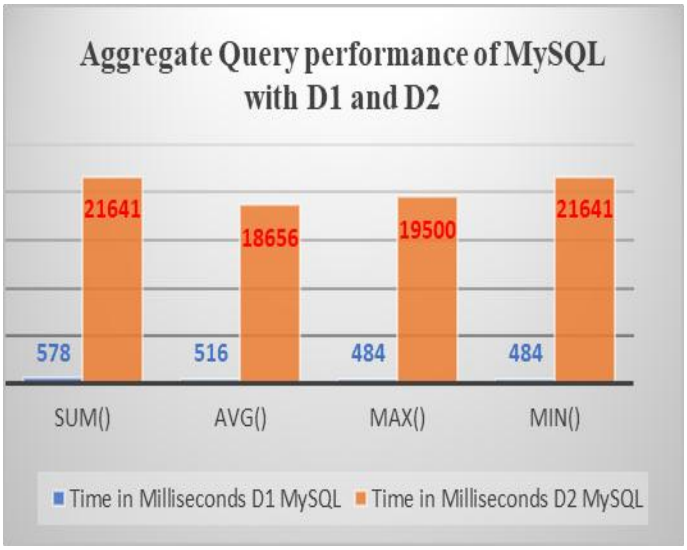

Figure 5: Performance comparison of MySQL Workbench for aggregate functions with D1 and D2.

On analyzing the above two graphs it is clear that the time taken by both MySQL and MongoDB increases as the size of the dataset increases. Both takes less time to execute queries with D1. And when D2 is used there is a relative increase in the time taken compared to D1. There is a proportionate increase in time both in case of MySQL and MongoDB. MongoDB performs well in all cases by taking less time for execution.

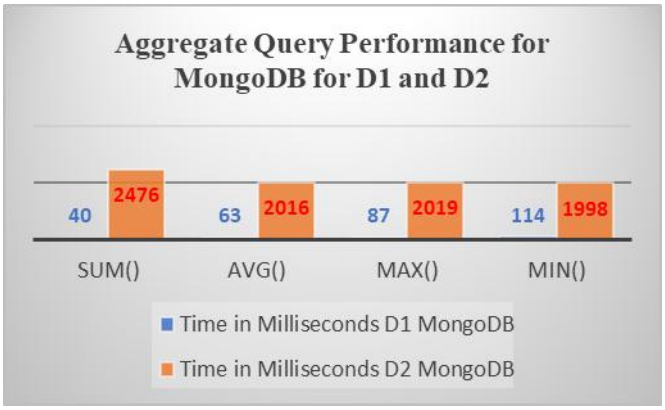

Figure 6: Performance comparison of MongoDB Studio3T for the aggregate functions with D1 andD2.

VI. CONCLUSION

NoSQL databases show good performance and scalability for most operations over huge datasets. In this paper, the experiments are done with different workloads to find the contrasts in execution time which is there with relational and NoSQL databases. It was performed by executing queries with the different aggregate functions with MySQL workbench 6.3 and MongoDB studio3T 3.4.5. The differences in execution time is shown using the graphs, but there is not much difference in the execution time taken by both in case of aggregation. But anyway, MongoDB performs well by taking less time and thus can be favored for its performance. It is a motive for the commercial business organizations to shift from conventional database systems to NoSQL databases in managing todays unstructured data. This research can be further improved by using several different types of queries with a higher number of records for different types of NoSQL databases.

REFERENCES

[1] Ramez Elmasri, Shamkant B. Navathe, *"Fundamentals of Database Systems"*, Pearson, India, pp. 621-622, 2007.

[2] Mary Femy P.F, Reshma K.R, Surekha Mariam Varghese, *"Outcome Analysis Using Neo4j Graph Database"*, International Journal on Cybernetics & Informatics, Vol 5, No.2, pp.229-236, 2016.

[3] Dipina Damodaran B, Shirin Salim, Surekha Marium Varghese, *"Performance Evaluation of MySQL and MongoDB databases"*, International Journal of Cybernetics and Informatics", Vol.5, No.2, pp. 387-394, 2016

[4] Guoxi Wang, and Jianfeng Tang, *"The NoSQL Principles and Basic Application of Cassandra Model"*, In the proceeding of the International Conference on Computer Science and Service Systems, Washington, pp.1332-1335, 2012.

[5] Yue Cui, William Perrizo, *"Aggregate Function Computation and Iceberg Querying in Vertical databases"*, A Thesis submitted to the Graduate Faculty of the North Dakota State University, North Dakota, pp. 10, 2005.

[6] Sanobar Khan, Vanita Mane, *"SQL support over MongoDB using metadata"*, International Journal of Scientific and Research publications, Vol.3, Issue.10, pp.1-5, 2013.

[7] Seyyed Hamid Aboutorabi, Mehdi Rezapour, Milad Moradi, Nasser Ghadiri, *"Performance evaluation of SQL and MongoDB databases for big e-commerce data"*, In the proceeding of the International Symposium on Computer Science and Software Engineering (CSSE), Iran,pp.72-78, 2015.

[8] Zhu Wei-ping, Li Ming-xin, Chen Huan, *"Using MongoDB to implement textbook management system instead of MySQL"*, In the proceeding of the IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, pp.828-830, 2011.

[9] Enqing Tang, Yushun Fan, *"Performance Comparison between Five NoSQL Databases"*, In the proceeding of the 7th International Conference on Cloud Computing and Big Data (CCBD), China, pp.105-109, 2016.

[10] https://docs.mongodb.com/manual/

Authors Profile

Benymol Jose pursed Masters in Computer Science from Bharathidasan University, Thiruchirappally, Tamil Nadu in 1999 and M.Phil in Computer Science from Madurai Kamaraj University, Madurai, Tamil Nadu in the year 2009. She is currently working as Assistant Professor in Department of Computer Applications, Marian College, Kuttikkanam, Idukki, and is pursuing Ph.D. in Mahatma Gandhi University, Kottayam in the topic of Unstructured data mining and NoSQL databases. She had published many papers in journals and conference proceedings including IEEE. Her main research work focuses on unstructured data mining, NoSQL databases and Big Data Analytics. She has 18 years of teaching experience and 3 years of Research Experience.

Sajimon Abraham M.Sc. Mathematics, M.C.A, M.B.A, Ph.D. in Computer Science. He has been working as Assistant Professor in Computer & IT, School of Management and Business Studies, Mahatma Gandhi University, Kottayam. He currently holds the additional charge of Director(Hon), University Center for International Co-operation. He was previously working as System Analyst at Institute of Human Resource Development and as Database Architect at Royal University of Bhutan under Colombo Plan on deputation through Ministry of External Affairs, Govt. of India. His research area includes Data science, Mobility Mining, Spatio Temporal Databases, Big Data Analytics and E-learning and has published 52 articles in National, International journals and Conference proceedings.

