AN ANALYSIS OF QUERY PERFORMANCE: MONGODB n SQL

Harshitha R[1], Vidya Raj C[2]

[1]Student, MTech-Information Technology, The National Institute of Engineering, Mysuru, Karnataka, India
[2]Professor, Dept. of CS&E, The National Institute of Engineering, Mysuru, Karnataka, India

---***---

Abstract *As the internet technology evolves, the data set gets boom, currently it is challenging to handle the Big Data. Complexity and huge data set are the main reasons to come up with MongoDB. Many departments, warehouses and corporate companies are switching from SQL to NoSQL. In this paper, we are using MongoDB - a NoSQL database and MySQL - a SQL database. The NoSQL database provides the best and faster performance for the large volume of data, it is highly scalable and mainly eliminates the duplication of data. This paper aims to provides the detailed performance analysis along with mapping of MongoDB with SQL, advantages and a study of comparisons of CRUD operations. The outcome shows us the MongoDB givers better results and is dynamic.*

***Key Words*: MySQL, MongoDB, NoSQL, Analysis with MongoDB with SQL**

1. INTRODUCTION

From the past few years relational database system is used as a primary DB to store the data. Increase in the usage of internet also increased the structured, unstructured and semi-structured data. While SQL Server is reasonably priced for licensing, it is still expensive as cores grow and there are limits to the small scales of some applications. The complexity of licensing also means that it becomes complex to manage this across time, as well as more expensive. SQL allows duplication - A property appears multiple times in the table. It must be stored across multiple tables in case of indexing and mapping, this makes the data repetition hence tables structure becomes complex and querying takes time. SQL is mainly a schema oriented structured database hence handling with the huge unstructured data becomes challenging and reduces the performance and scalability. The use of NoSQL – not only SQL databases like MongoDB avoids all these main drawbacks and provides the enhanced performance of read and write operations for larger volume of data set. Hence there is a huge transformation of usage in NoSQL databases in this recent and current era and is still growing.

2. RELATED WORKS

As the essential focus is to migrate the data from SQL to MongoDB and to reduce the process overhead involved in migration process. Even though there are multiple algorithms available it is difficult to implement the functionality of SQL queries in to MongoDB. The method implements the graphical user interface to convert the queries automatically. It reduces the burden of syntax studying [1].

Metadata provides the information about other data; A method is proposed where in the Metadata layer acts as an interface between database layer and application layer, which supports the SQL query language to No SQL query language by conversion. Metadata holds the routing information for the conversion form one format to another [2]. A relative comparison of NoSQL to SQL mainly by considering their concepts and commands are focused [4].

An experimental setup comparing the two different databases and measuring the performance by its runtime, it contains four separate test runs with hundred each run. The outcome shows SQL performs better with updating and selecting non-key attributes while MongoDB provides the best results in all the operations [5].

The evaluation of performance in Big e-commerce databases includes the set of experiments with many operations such as read, write, select and delete from different aspects and provides the data with different complexities can be easily handled with MongoDB over SQL. [7]

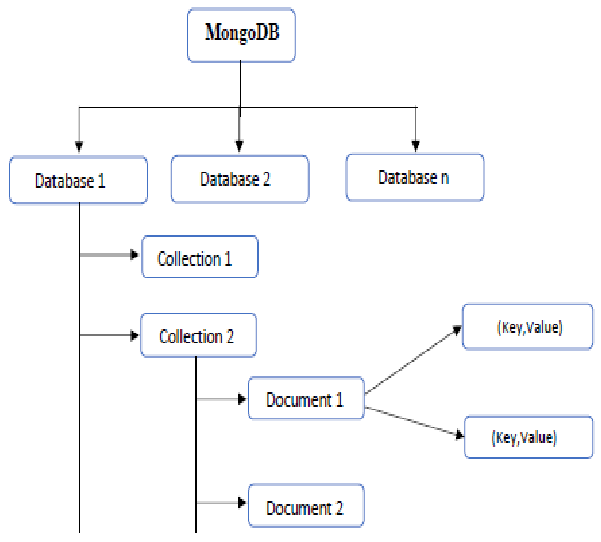

Fig-1: Structure of MongoDB

3. OVERVIEW OF MONGODB

MongoDB is a Document based database of NoSQL type that provides us with increased availability and scaling as its major features. It stores the data in JSON format and provides JavaScript functions to query that data [1], which meaning fields can vary from document to document and data structure can be changed over time. MongoDB is free of cost as it is an open source. It mainly avoids the table structures, it implements a simple collection containing documents. MongoDB uses the port 27017 for the client connection. 'Mongod' is the important process which handles all the tasks of MongoDB server. MongoDB allows ad-hoc querying and aggregations, which provides the real time powerful access and analysing the data. The generic structure of MongoDB is defined in Fig-1.

3.1 Reasons Why MongoDB

3.1.1 Schemaless

The JSON based MongoDB is schemaless, the documents of the database can have varying number of fields with different data types. The data is stored in JSON format or as (Key, Value) pair and there are no limitations on number of pairs in a document. Any database architect can easily design a database without rigid schema structure thus it does not need to follow 3NF form of Normalization as of SQL. In case of SQL, when the ongoing process is needing to be modified, it takes time to redesign the schema plan and to recontinue the process. Thus, MongoDB is dynamic and flexible to use.

3.1.2 Sharding

As the data grows the demand for the storage also increases, MongoDB uses the sharding concept. When a machine is having insufficient storage, this sharding resolves the problem using horizontal scaling. In this horizontal scaling, the data is stored across different servers and the main advantage of it is increase in performance. A MongoDB cluster is made of one or more shards, where each shard node is responsible for storing the actual data. Each shard consists of either one node or a replicated node which just holds data for that shard. Read and write queries are routed to the appropriate shard [2]. This process and horizontal scaling are not available in SQL.

3.1.3 Replication

Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication provides a level of fault tolerance against the loss of a single database server [3]. MongoDB has its replica set- which contains 'mongod' processes with same data. This replication is very much needed for backup and for disaster recover, hence gives the data ready all the time and it does not consume time for replication. Replication provides the synchronization in multiple servers.

3.1.4 CAP Theorem

MongoDB follows CAP theorem, which means that data will be consistence, available and partition tolerant across the distributed systems. Consistency deal with providing same data to all the clients after the execution of certain operations. Availability - MongoDB keeps the data available all the time, with no downtime. Partition Tolerance - System works continuously even if there is no communication between the servers or in case of the message fail. To maintain all three properties is most important factor rather than the ACID properties of SQL. The fig-2 briefs out the reasons why MongoDB can be used.

4. MAPPING OF MONGODB WITH SQL SERVER

The mapping of MongoDB with SQL Server is shown in fig 3. The SQL database have multiple tables in it whereas the MongoDB have collections in it. MongoDB is mainly a document-based database, rather than the rows and columns it contains fields of the documents. The Group by operation of SQL is made by simple Aggregation in MongoDB.

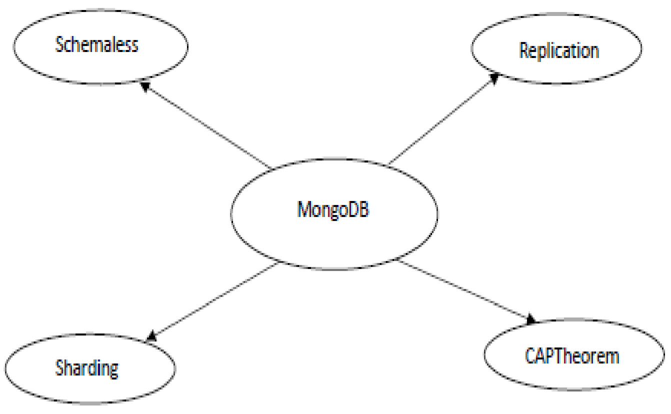

Fig-2: Reason for MongoDB

The following example briefs the simple MongoDB document.

```
{
        id: ObjectID ("5e0c3356017f45dfbf7b199d")
        DepartmentName: "cardiac"
```

PatientAdmit_date: 03:01:2019;

}

The above examples describe the Hospital database, containing a document with fields DepartmentName, PatientAdmit_date, which stores the data in json format and an id which is generated automatically.

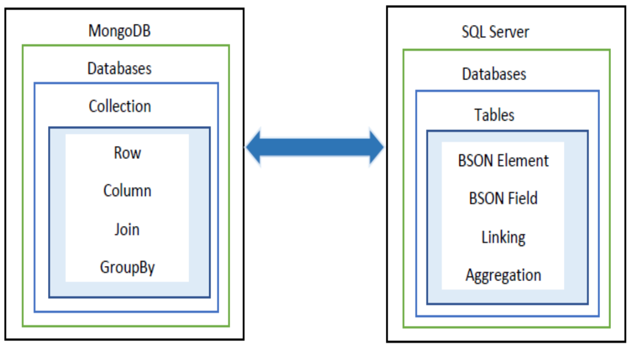

Fig-3: Mapping of MongoDB with SQL

4.1 Comparison with respect to schema

4.1.1 For CREATE Command

SQL:
CREATE TABLE Department (id varchar(30) PRIMARY KEY, DepartmentName varchar(30), PatientAdmit_date date);

MongoDB:
db.Department.insert({id:246, DepartmentName:"cardiac", PatientAdmit_date: 21:11:2018})

4.1.2 For DROP Command

SQL:
DROP TABLE Department;

MongoDB:
db. Department.drop( )

4.1.3 For DELETE Command

SQL:
DELETE FROM Department WHERE DepartmentName="cardiac";
MongoDB:
db.Department.remove({DepartmentName="cardiac"})

4.1.4 For INSERT Command

SQL:
INSERT INTO Department (id, department_name, patient_admitDate)VALUES(842,"nephrology",06:05:2011);

MongoDB:
db.Department.insert({id:843,department_Name:"nephrology",patient_admitDate : 06:05:2011})

4.1.5 For SELECT Command

SQL: SELECT * FROM Department;

MongoDB: db.find.Department ( )

5. EXPERIMENT

5.1 System Specifications:

The experiment is conducted with the following system specifications:

- Operating System: windows 10, 64 bits
- Processor: intel core i5
- RAM: 8GB

5.2 Experimental Details

The experiment is conducted in both MySQL and MongoDB. Both the database has same number of tables or collection and same number of columns or fields in MySQL and MongoDB respectively.

5.2.1 Insertion Operation

Initially created a Hospital Database which contains the tables or collection as follows:

Hospital table or collection: id, name, department_name, patient_name, patient_admitDate;

Department table or collection: id, department_name, patient_admitDate;

Patient table or collection: id, Fname, Lname, age, hospital, patient_admitDate;

This hospital database which includes the foreign keys and primary keys in it. Example: department_name is the primary key in department table or collection and it is a foreign key in hospital table. The database is inserted with 10,000 records of patients and have designed a simple UI using c# for interaction. The timings are recorded using microtime function. The results after the insertion of both shows that in MySQL, it is inserted in 480sec and in MongoDB, it takes 0.40 seconds. The fig-4 shows the graphical representation of insertion operation.

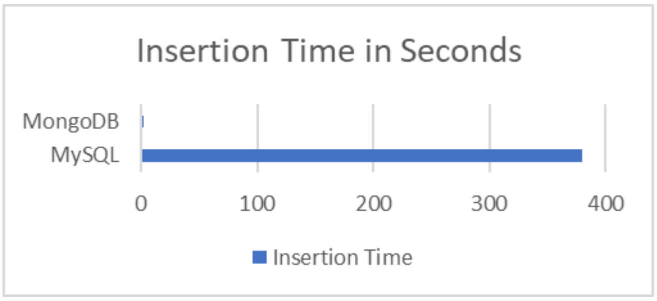

Fig-4: Insert Operation

5.2.2 Query Operation

The query which is used here is to retrieve all the patient's details of particular department on particular date. The MySQL uses the join operation on patient and department table to retrieve the data. The query operation takes 0.0045 seconds in MongoDB which is negligible. The results are shown in fig-5.

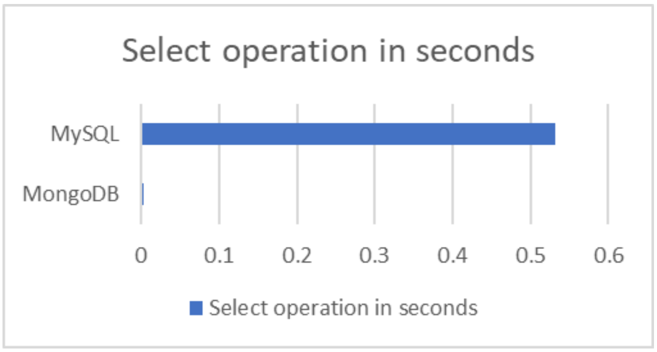

Fig-5: Query Operation

5.2.3 Delete Operation

As the results observed in insertion and selection operation, the deletion also operates faster than MySQL. The deletion of all the patient's on specified date they admitted is as shown in the fig-6, which represents that time taken to delete patients record in MySQL is 0.4873 seconds and time taken in MongoDB is 0.00195 seconds.

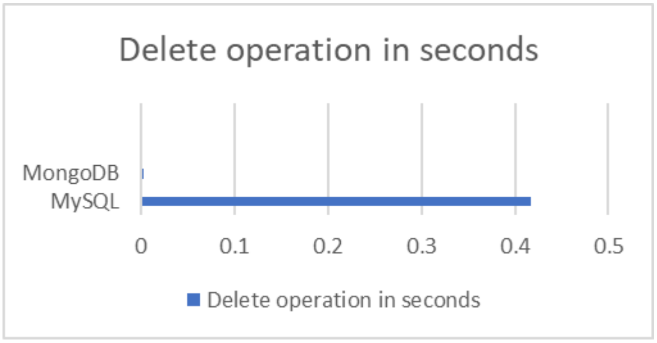

Fig-6: Delete Operation

By comparing overall results, it clearly shows that MongoDB generates more faster results than MySQL. MongoDB is the choice for users who need a less rigid database structure. MongoDB could be a good solution for larger data sets in which the schema is constantly changing or in the case that queries performed will be less complex [5].

6. CONCLUSIONS

MongoDB is the most popular among the NoSQL databases. It is a great tool for building data warehouses, especially because of its ability to fully utilize so called "shared-nothing cluster architecture." The flexible and scalable nature of MongoDB provides the user to quick change of their databases from rational DB's. There is no need of mapping of application objects to database objects which makes the user friendly.

The only limitation of this MongoDB is the data size is limited upto 16 mb per document but this problem can be easily solved by using MongoDB's GridFS, which allows us to load more than 16mb of data in each document. MongoDB is best suitable for hierarchical data storage which is not suitable for SQL. Thus, the NoSQL database, MongoDB is best suitable for massive data storage.

REFERENCES

[1]. Shaikh, N. F., Jadhav, A., Raina, C., Nagoshe, G., & Kale, S. (2018). Data Migration From SQL To Mongodb. HELIX, 8(5), 3701-3704.

[2]. Khan, S., & Mane, V. (2013). SQL support over MongoDB using metadata. International Journal of Scientific and Research Publications, 3(10), 1-5.

[3]. MongoDB https://www.mongodb.com./

[4]. Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015, June). A comparative study: MongoDB vs. MySQL. In 2015 13th International Conference on Engineering of Modern Electric Systems (EMES) (pp. 1-6). IEEE.

[5]. Parker, Z., Poe, S., & Vrbsky, S. V. (2013, April). Comparing nosql mongodb to an sql db. In Proceedings of the 51st ACM Southeast Conference (pp. 1-6).

[6]. MongoDB Documentation https://docs.mongodb.com./

[7]. Aboutorabi, S. H., Rezapour, M., Moradi, M., & Ghadiri, N. (2015, August). Performance evaluation of SQL and MongoDB databases for big e-commerce data. In 2015 International Symposium on Computer Science and Software Engineering (CSSE) (pp. 1-7). IEEE.