SQL Support over MongoDB using Metadata

Sanobar Khan[*], Prof.Vanita Mane[**]

[*] Department of Computer Engineering, RAIT
[**] Department of Computer Engineering,RAIT

Abstract- New requirements are arising in environments where we have higher volumes of data with high operation rates, agile development and cloud computing. In recent years, a growing number of companies have adopted various types of non-relational database, commonly referred to as NoSQL database. This reflects the growing interactivity of applications which are becoming more networked and social, driving more requests to the database where high-performance NoSQL database such as MongoDB becomes favorable. This paper attempts to use NoSQL database to replace the relational database. It mainly focuses on one of the boosting technology of NoSQL database i.e. MongoDB, and makes a comparison with MySQL and thus justifies why MongoDB is preferred over MySQL. Lastly, a method is proposed to integrate these two types of database by adding a middleware (Metadata) between application layer and database layer.

Index Terms- ACID, BASE,MySQL,MongoDB, Metadata, NoSQL,RDBMs

I. INTRODUCTION

Relational database is widely used in most of the application to store and retrieve data. They work best when they handle a limited amount of data. Handling a huge volume of data like internet was inefficient in RDBMS. To overcome this problem "NO SQL" came into existence. The term NOSQL is short for "Not Only SQL" and was introduced in 2009, when it was chosen a title of a conference "for folks interested in distributed structured data storage" [8].The name attempted to label the emergence of a growing number of non-relational, distributed data stores that often did not attempt to provide ACID.NO SQL, is not a tool, but a methodology composed of several complementary and competing tool.

The primary advantage of NOSQL database is that, unlike relational database they handle unstructured data such as documents, e-mail, multimedia and social media efficiently. Most of the common features of NOSQL database can be summarized as schema is not fixed, does not support join operations, high scalability and reliability, very simple data model, very simple query language, high availability at the price of loosing the ACID trait of the traditional database in exchange with keeping a weaker BASE (Basic Availability, Soft State, Eventual Consistency) feature,uses cheap commodity server to manage exploding data and thus leads to low cost, efficient use of distributed indexes and RAM for data storage, ability to dynamically add new attributes to data records, ability to replicate and to distribute data over many servers [1] [4].Therefore, NoSQL database systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook which had significantly different challenges in dealing with data that the RDBMs solutions could not cope with. There are many advantage of NoSQL as compared to RDBMs, but also there are many obstacles to overcome before they can appeal to mainstream enterprises. Few of the challenges are Maturity which means RDBMs systems are stable and richly functional whereas NoSQL are in pre-production versions with many key features are yet to be implemented,Support,Administration and NoSQL database is still in learning mode.

There are three core categories of NoSQL data model, a) Key-Value stores; in this a value corresponds to a key and data is stored as a key-value pairs. E.g. Redis, flare; b) Column-Oriented stores; in this database contain one extendable column of closely related data and uses table as the data model but do not support table association. E.g. Cassandra, HBase; c) Document Based stores; in this data is stored and organized as a collection of document but the value of document database is stored in JSON or XML format. E.g. MongoDB, CouchDB. MongoDB is a document database developed by 10gen. It manages collection of JSON-like documents. Many applications can thus model data in a more natural way, as data can be nested in complex hierarchies and still be query-able and index-able.

With the rapid development of social web as well as cloud computing, the traditional database cannot cope with the basic demands of availability, scalability, storage of huge data and fast data backup and recovery. Thus numerous internal portals and the emerging WEB 2.0 website make use of open source technology of NoSQL. But we all know that use of relational database will never come to an end, because it provides us with an unparalleled feature set, by maintaining data integrity and scalability. Practically,it is the developers' job to decide which database should be used to meet the application requirement. Nowadays, mostly NoSQL database is used by developer for storing large amount of database. However, most NoSQL systems employ a distributed architecture,with the data held in a redundant manner on several servers, and partitioning scheme relies on consistent hashing to distribute the load across multiple storage hosts [1].Therefore, relational database can be combined with NoSQL, so that it can handle relational structure efficiently.

This paper proposes a methodwhich provides SQL Query language support to the non-relational databaseMongoDB by appending an interface (Metadata), between the application layer and database layer. Application does not have to consider about the storage location, data model and memory requirement.In order to communicate with the database layer, the Metadata contains all the routing information as the conversion rules to convert from one format to another. This model will fulfill the scalability of the massive data without affecting the logical implementation.

Depending on the requirement of the application, we can use different type of NoSQL database in combination with the relational database and the Metadata.Each NoSQL database has its own features, data model and architecture choice of the database depends on the application. We have used MongoDB for our proposed system as it has gain more popularity in recent years.

The rest of the paper is structured as follows. Section II gives a brief overview of MongoDB functionality. Section III gives a performance comparison between the most popular relational database (MySQL) and MongoDB by using reference paper [2].Section IV gives a descriptive theoryand flowchart of our proposed method. Section V concludes this paper.

II. OVERVIEW OF MONGODB

A. MONGODB

MongoDB [4] is a schema less document-oriented database. The name MongoDB comes from "humongous". The database is intended to be scalable and is written in C++. The primary reason for moving away from relational model is to make scaling easier. The basic idea is to replace the concept of a "row" with a more flexible model, the "document".By making use of embedded documents and arrays, this approach makes it possible to represent complex hierarchicalrelationship with a singlerecord. MongoDB is also schema free: a document's keys are not predefined or fixed [12] [13].

1)*Features* MongoDB support Bson data structures to store complex datatypes;supports powerfuland complex query language; high speed access to mass data; stores and distribute large binary files like images and videos; instead of stored procedures, developers can store and use JavaScript functions and values on the server side; supports an easy-to-use protocol for storing large files and file metadata [13]; it gives fast serial performance for single clients; uses memory mapped files for faster performance. Because of these characteristics of MongoDB, many projects with increasing data are considering using MongoDBinstead of relational database.

2) *Data Design*MongoDB database holds a set of collections. A collection has no pre-defined schema like tables, and stores data as documents. BSON (binary encoded JSON like objects) are used to store documents. A document is a set of fields and can be thought of as a row in a collection. It can contain complex structures such as lists, or even document. Each document has an ID field, which is used as a primary key and each collection can contain any kind of document, but queries and indexes can only be made against one collection [7][12].MongoDB supports indexing over embedded objects and arrays thus has a special feature for arrays called "multikeys". This feature allows using an array as index, which can then be used to search documents by their associated tags.Figure1 shows the structure of MongoDB.

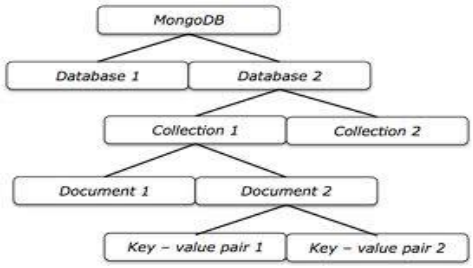

Figure 1: STRUCTURE OF MONGODB

3)*API*MongoDB [15] has its own query language namedMongo Query Language. To retrieve certain documentsfrom a db collection, a query document is createdcontaining the fields that the desired documents shouldmatch. For example,

- Insert Command

db.users.insert ({ user id:"abc123", age: 55, status:"A"})

- Drop Command

db.users.drop ()

- Select Command

db.users.find ({ status:"A", age: 55})

- Delete Command

db.users.remove ({ status:"A"})

MongoDB uses a RESTful (RepresentationalState Transfer) API. It is an architecture style for designing networkedapplications. It relies on a stateless, client-server,cacheable communications protocol (e.g., the HTTPprotocol). RESTful applications use HTTP requests topost, read data and delete data.

4) *Architecture* MongoDB cluster is built up using three main components namely Shard nodes, Configuration servers and Routing services or mongos as shown in Figure 2.

Shard nodes: 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 shards. A replicated node consists of one or more servers, where one server acts as a primary server and others are secondary servers. If the primary server fails one of the secondary servers automatically takes over as primary. All writes and consistent reads go to the primary server and all eventually consistent reads are distributed amongst all the secondary servers.

Configuration servers: A group of servers in the cluster are called configuration servers. This server are used to store metadata and routing information of the MongoDB cluster indicating which data is present on which shard.

Routing services or mongos: These servers are responsible for performing tasks requested by the client. Clients issue different types of queries and depending on the type of queries, mongos send the requests to the necessary shards and merge the result before it is send back to the client. Mongos for themselves are stateless and therefore they can run in parallel[7].

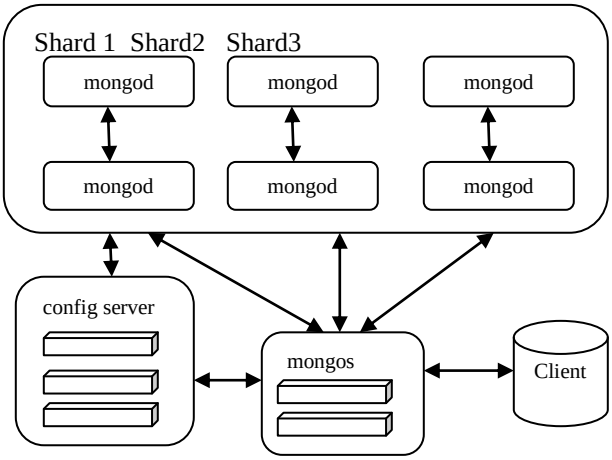

Figure 2:MONGODBARCHITECTURE

MongoDB uses memory-mapped files, in order to use as much of available memory as possible and thus increase the performance. B-trees are used to index the MongoDB database. A user specified shard key is used to partition their owning collection in a MongoDB cluster [15]. MongoDB supports auto-sharding enabling horizontal scaling across multiple nodes. Sharding is used to partition data across multiple servers by maintaining the order using the shard key. Thus, maintaining automatic rebalancing of shards.

III. COMPARISON OF MONGODB AND MYSQL

As per the detailed review of several papers, a comparative study is made between MongoDB and MySQL based on their concept and commands used for different operations.

A. Based on Terms/Concept

Table I: TERMS/CONCEPT

SQL terms/concept	MongoDB terms/concept
Table	Collection
Row, Column	documentor BSON document,field
Index	index
table joins	embedded documents and linking
primary key (explicitly)	primary key (implicitly)
fixed schema	schema less

B. Based on Schema Statements

Table II: SCHEMA STATEMENT

SQL schema	MongoDBschema
A. Create Command **CREATETABLE**teachers (t_idVarchar(30), age Number, status char(1), **PRIMARYKEY**(id))	**db.teachers.insert**({t_id:"abc123",age:55,status : "A"})
B.DROP Command **DROP TABLE teachers**	**db.teachers.drop()**
C.INSERT Command **INSERTINTO**teachers(t_id, age,status) **VALUES** ("a123",45,"A")	**db.teachers.insert**({t_id:"a123",age:45,status:"A" })
D.SELECT Command **SELECT**t_id, status,age **FROM**teachers	**db.teachers.find**({ },{t_id:1, status:"B",age:45 })
E.DELETE Command **DELETEFROM** teachers **WHERE** status ="D"	**db.teachers.remove**({ status:"D" })

C. Based on Performance

In [2], the authors have performed testing and thus have compared MongoDB with MySQL database. They have performed testing by using the textbook management system. The given graph shows the result of testing. In performance testing, the authors have inserted 100 to 50,000 textbooks information into database. The cost time of MongoDB and MySQL were recorded as shown in figure [14]. Two important factors for which MongoDB was preferred over MySQL are [14]:

- Insertion Speed

From the graph, we notice that MongoDB spends less time than MySQL, for a large amount of information as shown in figure 2. It leaves MongoDB 30×-50× faster than MySQL as sown in figure3.

Number of Parallel Clients 5 Time in seconds

Basic Insert	Total Rows	Rows / client	SQL Time	Mongo Time	Sql Ops/sec	Mongo Ops/sec
several columns	100	20	0.19	0.011	526	9,091
600 bytes per row	1,000	200	1.8	0.02	556	50,000
	5,000	1,000	9	0.25	556	20,000
	25,000	5,000	100	1.5	250	16,667
	50,000	10,000	270	2.5	185	20,000

Figure 2:INSERTION SPEED COMPARISONS [14]

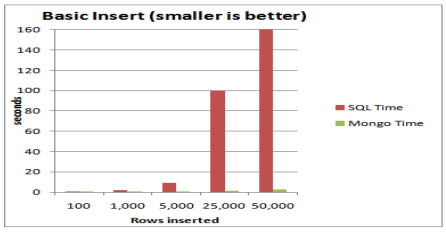

Figure 3:INSERTION TIME FOR MYSQL AND MONGODB [14]

- Query Speed

In the figure 4, it calculates the time to get the data out of the database.

Number of Parallel Clients 5 Time in seconds

	Total Rows	Rows / client	SQL Time	Mongo Time	Sql Ops/sec	Mongo Ops/sec
Basic Query	50	10	0.1	0.08	500	625
with index	500	100	0.38	0.1	1,316	5,000
	5,000	1,000	2.8	1.2	1,786	4,167
	25,000	5,000	14	4	1,786	6,250
	50,000	10,000	28	10.4	1,786	4,808

Figure 4:QUERY SPEED COMPARISONS [14]

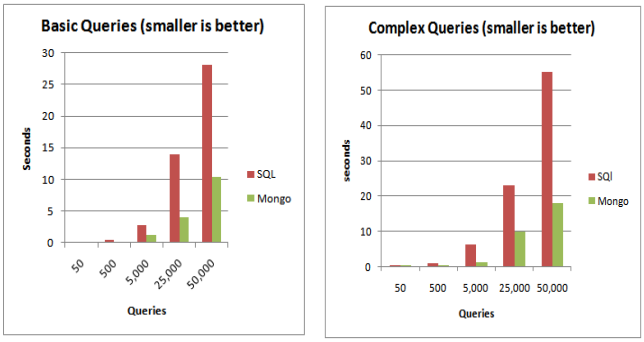

Figure 5:BASIC AND COMPLEX QUERY TIME FOR MYSQL AND MONGODB [14]

MongoDB leads MySQL with almost 3× performance as shown in figure 5. But MongoDB spends much more time on problem solving as well as the post maintenance issues and is not easier than MySQL.Thus from above comparison, it proves that for large amount of data MongoDB is preferred over MySQL.

IV. PROPOSED METHOD

A. SYSTEM DESCRIPTION THEORY

In this system, a method is proposed to integrate these two types of database by adding an interface between the Application layer and Database layer. The middleware between the two consist of **Metadata** which consist of different types of packages. This system needs to implement a package which acts as an interface between java application layer and NoSQL database (MongoDB).The system makes use of MongoDB and the reason for using this has been discussed above in detail. The system is primarily designed for embedded java based application which requires database access. The database command fired from java based application is given as input to the interface acting as a middleware(Metadata).The interface parses the input and reformats the code in the format that is been required by the back-end database (MongoDB).The reformatted code consists of functions that directs back-end database to implement and maintain the database.In form of result, the back-end databases will response to the middleware with the results. As a result, the middleware responses to the java application in terms of successor an exception in case offailure. Hence to act as middleware the system has to perform conversion from one form to another. Thus, the system has to store some information about the format conversion rules and the data structure format. In turn, the system has to store the metadata for supporting the working of the system.Figure 6depicts the architecture of our proposed system.

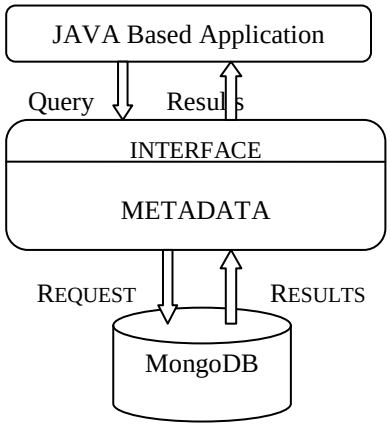

Figure 6:DATABASE INTEGRATION ARCHITECTURE

C. SYSTEM FLOWCHART

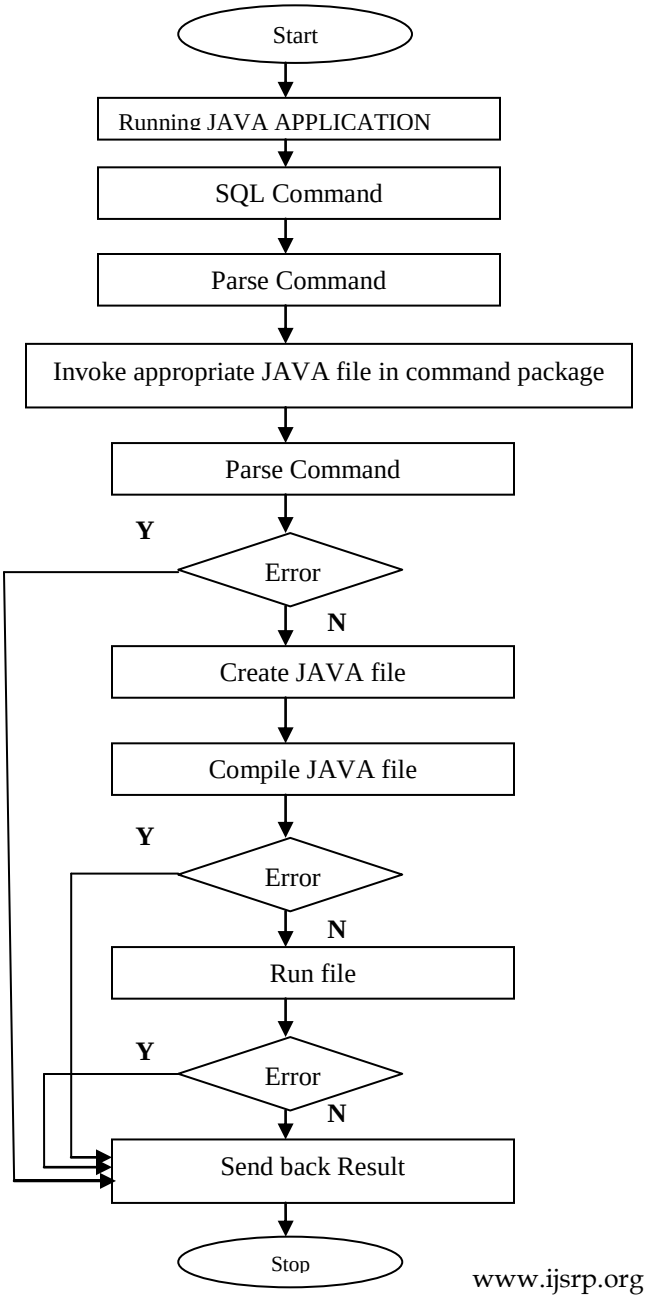

Figure 7:SYSTEM FLOWCHART

In our proposed, we make use of standard SQL query language to invoke commands to the database. Due to the need, of huge amount of data storage we make use of one of the boosting technology of NoSQL database i.e. MongoDB. MongoDB stores data in BSON structure. So in order to get result from this database we make use of Metadata structure which stores all the conversion rules as well as the data format required for the application layer and database layer to communicate. The reason for using SQL query language is that it is a most popular and standard database as well many developers throughout the world are expert in programming using SQL. Figure 9 shows the detailed flow of our proposed method.

V. CONCLUSION

RDBMS won't go away, they're still definitely needed. But the storage requirement for the new generation of applications are huge different from legacy applications.We can choose MongoDB instead of MySQL because of two factors, ease of use and performance. We conclude that if your application is data intensive and stores lots of data, queries lots of data, and generally lives and breathes by its data, then you'd better do that efficiently or have resources (i.e. money) to burn. Lastly, the report concludes by proposing a database integration method by using a middleware between the two layers. In this method, application does not have to consider about the complexity of underlying database layer there data distribution and storage. They have to use the basic SQL query language to get result from the database and all the format conversion rules will be done by the Metadata. The system was proposed because MongoDB has newly come into existence, whereas the standard SQL language has been over yearsand, therefore if we merge the two we can use the features of both the database. Although, NoSQL has the advantage of horizontal expansion, but for complex SQL requests, it cannot support them very well. For the Query based on KEY/VALUE and massive data storage requirements, NOSQL is a good choice.

REFERENCES

[1] Hailing Zhang, Yang Wang, Junhui Han, " Middleware Design for Integrating Relational Database and NoSQL Based on Data Dictionary" International Conference on Transportation ,Mechanical ,and Electrical Engineering(TMEE), 978-1-14577-1701,Dec 2011

[2] Zhu Wei-ping, Li Ming-xin, Chen Huan, "Using MongoDB to Implement Textbook Management System instead of MySQL",IEEE,978-1-61284-486,2011

[3] Bogdan George Tudorica, Cristian Bucur, "A Comparison between several NoSQL Databases with comments and notes"

[4] Jing Han, Haihong E, Guan Le, "Survey on NoSQL Database" , IEEE 978-1-4577-0208-2,2011

[5] Clarence J M Tauro, Aravindh S, Shreeharsha A.B, "Comparative Study of the New Generation, agile, Scalable, High Performance NOSQL Database.", International Journal of Computer Applications (0975 – 888) Volume 48– No.20, June 2012.

[6] Neal Levitt, "Will NoSQL Databases Live Up to Their Promise?" IEEE Computer Society, vol.43, no.2, pp.12-14, Feb.2010

[7] Rabi Prasad Padhy, Manas Ranjan Patra, Suresh Chandra Satapathy, "RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's", International Journal of Advance Engineering Sciences And Technologies, Vol. 11, Issue No. 1, 015-030,2011

[8] Dominik Bruhn, "Comparison of Distribution Technologies in Different NoSQL Database Systems":

[9] Rick Cattell, "Scalable SQL and NOSQL Data Stores" , Vol. 39, No.4, December 2010

[10] Christof Strauch, "NoSQL Database": www.christof-strauch.de/nosqldbs

[11] Nathan Hurst, "Visual Guide to NoSQL Systems": http://blog.nahurst.com/visual-guide-to-NoSQL-systems/.

[12] Kristina Chodorow, Michael Dirolf, "Orielly MongoDB, The Definitive Guide", September-2010.

[13] MongoDB, http://www.mongodb.org/display/DOCS/Home

[14] MichaelKennedy, "MongoDB vs.SQL server"http://blog.michaelckennedy.net/mongodb-vs-sql-server-2008-performance-showdow,2010

[15]Lior Okman, Nurit Gal-Oz, Jenny Abramov,"Security Issues in NoSQL Databases",IEEE, 978-0-7695-4600-1, 2011

AUTHORS

First Author – Sanobar Khan, M.E.Computer (pursuing), R.A.I.T, Email:sanobar_14@yahoo.com

Second Author –Prof. Vanita Mane, M.E. (Computer),Department ofComputer Engineering,R.A.I.T, Email:vanitamane1@gmail.com