IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 56-60

Migration: From RDBMS to NoSQL Database

Husain S. Amreliwala[1], Mohd. Shibli Qureshi[2], Taha Bikanerwala[3], Er. Zainab Mirza[4]

[1]*(IT, M. H. Saboo Siddik College of Engineering, India)*
[2]*(IT, M. H. Saboo Siddik College of Engineering, India)*
[3]*(IT, M. H. Saboo Siddik College of Engineering, India)*
[4]*(HOD IT, M. H. Saboo Siddik College of Engineering, India)*

Abstract: *Data today is increasing on tremendous scale making it a challenge for existing systems to handle it properly. Statistics state that on an average about 2.5 exabytes that is, 2.5 billion gigabytes (GB) of data is generated every day as of now. Traditional databases such as SQL databases start to fail miserably once the size of data crosses the maximum data handling threshold. With expanding the size of the data, we also observe the variation in the structure of data. NoSQL database technology was developed to handle this expansion and variety with simplicity and ease. Although one can efficiently develop and test the applications using structured data,but when unstructured and semi-structured operational data comes into the picture, NoSQL technology is the optimal choice in case of efficiency. This paper provides such examples and proofs that argue how one can migrate from RDBMS (SQL) to NoSQL and also one can enhance existing applications by opting data migration to NoSQL*

Keywords: NoSQL, SQL, MongoDB, Big Data, RDBMS

I. Introduction

Structured Query Language (SQL) databases are structured databases based on the concept of relational databases. Tables are the basic unit of SQL databases. Till 2005 - 07, almost every internet application that hosted data creation, updation, deletion and display services i.e in all data management services harnessed their data handling strengths from SQL based RDBMS (Relational DataBase Management System). SQL is the standard language that is supported by such RDBMS for the basic CRUD (Create, Read, Update, Delete) operations as well as the advance operations that consists of JOINS, TRIGGERS, Transactions, etc. Thus SQL technology was the most popular and widely used database technology at that time. But, since 2005 the size of data began to grow on an unexpected pace due to the advent of technology. Sources of data generation multiplied in numbers, playing the role of one of the major contributors to increasing size of data. This turning point in data technology, lead to the introduction of Big Data. The primary obstacle was that SQL was falling inefficient in terms of handling data that was increasing beyond its handling capacity. Although its capacity was stretched to occupy big data, but this put forth a lot of constraints on the actual functioning of SQL. One of the major issues were not only the size but also the widening of variety. Big Data constitutes the data that spreads in no one domain, but multiple domains. Data gradually began to lose structure and extensive applications endorsing image files, video files made it more difficult to store the actual files in directory filesystem and store their respective paths in the database. This proved cumbersome when retrieval operations came to light. As a counter attack to these issues, already introduced in 1998 NoSQL was reintroduced in early 2009 as a concept of database systems that are non - relational and did not expose the SQL. NoSQL functions at the cost of ACID transactions, but it provides extensible scalability, partitioning and distributed data storage. This paper presents findings and proofs that state the inefficiencies of SQL considering some examples, and efficiencies of NoSQL in that examples as a result of migration from relational algebraic systems to more Big Data-oriented NoSQL systems.

II. Literature Review

Jagdev Bhogal and Imran Choksi in their IEEE publication “HANDLING BIG DATA using NoSQL”(published in 29th International Conference on Advanced Information Networking and Application Workshops, 2015) introduced primary and quite significant comparisons between the NoSQL and SQL systems in the big data context. NoSQL (abbreviation for not only SQL) is the emerging technology in data management domain and serves as a viable alternative to SQL based systems. The fact that is highlighted in this publication is that the major problem to solve is not; how NoSQL can replace SQL, but how NoSQL and SQL can be used in a complementary fashion together. One needs to understand the requirements of data structure and accordingly decide to opt for whether SQL or NoSQL. The author demonstrates that there happens a significant trade-off between three major requirements of data management paradigm name Consistency, Availability, Infinite Scaleout. Although data stored in NoSQL systems is available anytime, anywhere (depending upon its server), and also they can be scaled on in infinite (preferably huge) scaling magnitude, but NoSQL systems trade-off with consistency and ACID transactions. On the contrary, SQL systems provide consistency and availability at its best, but they make a trade-off with scalability. Big Data cannot be handled using relational databases i.e. SQL. The author demonstrated these peculiarities by implementing a small prototype application using both SQL and NoSQL. For SQL, Oracle APEX was considered and for NoSQL, the document-oriented MongoDB was selected. The author illustrated this difference by comparing two different visual schema definitions of the same application. One was the fixed schema to be implemented in SQL and another was the Document Schema to be implemented in MongoDB. When technically analysed, the author attributed a fact why NoSQL provides better performance is that NoSQL schema can reduce the hassle of multiple joins and complex queries. Normalizing the schema definition in SQL provides clean and segregated structure, but this also requires a number of joins and complex nested queries to retrieve even small amount of relationally linked data, on other hand NoSQL being of no fixed schema, can easily retrieve data with simple function calls (for eg: in MongoDB). The author concluded by suggesting applications of NoSQL in strategic business data and IoT based sensor networks that are nothing but the real-time data. Applications of referential integrity and which possess no concrete structure are more preferred to be handled by NoSQL systems. Dynamic data model applications are most suitable for NoSQL.

Yishan Li and Sathiamoorthy Manoharan in the IEEE publication “A performance comparison of SQL and NoSQL databases” make an effective comparison between the existing SQL technology for handling big data and the rapidly emerging NoSQL technology. The author puts forth the idea that not all NoSQL base database management techniques are as efficient as SQL but also attributed to the fact that SQL lacks some major features that NoSQL provides. In this publication, the author compares the key - value implementations of both NoSQL and SQL databases. The comparison is based on four major data management operations viz. Read, write, delete, retrieval. The author provides introductory information about the initial implementations of NoSQL technology namely Google’s Big Table and the Amazon’s Dynamo. The author further provides information about the various existing implementations of NoSQL, among them are MongoDB, Hypertable, Apache CouchDB, Apache Cassandra, RavenDB and Couchbase. The author has compared above NoSQL implementations with the Microsoft’s SQL Server Express using an experimental setup to test the time taken by each of these implementations in performing operations like Instantiate, Read, Write, Delete. Also, retrieval of the data is considered. For instantiate operation, the RavenDB NoSQL implementation ranks fastest while SQL Express ranks slowest. For reading operation, the Couchbase and MongoDB perform the best while RavenDB performs the worst. SQL Express ranks on the third positions in this comparison. For a write operation, Couchbase and MongoDB again bag the top 2 positions, while RavenDB and CouchDB finish at second to last and last respectively. SQL Express makes it to the fifth position on this run. For a delete operation, the results are quite similar to that of the results of the read operation. Couchbase, MongoDB and SQL Express prove themselves the frontrunners while RavenDB and CouchDB take the last positions. The maximum number of records that are read, written and deleted as well as fetched are 100000 records. In the final test i.e fetch all records operation, SQL Express ranked the fastest while other NoSQL implementation faired well. CouchDB still was the last rank holder while Couchbase was excluded in this test as there are no APIs available for fetching all the keys. The author concludes by stating that NoSQL is peculiar because this technology is optimized for its key - value implementations. While SQL is not yet optimized. The author asserts that there are a wide variety of NoSQL implementations available and existing for applications. One can easily observe how different NoSQL implementations perform best for respective operations and worst for respective operations. The overall performance of MongoDB, a document-oriented implementation of NoSQL performed the best if the inability of Couchbase to fetch all records is considered, otherwise Couchbase is the front rank holder. The author also argues that these database comparisons and standings may not hold true when these are tested for more complex operations, although these implementations undergo various changes and revisions that may lead to degradation of existing features and introduction of new features.

III. Current Scenario

There are multiple applications and systems running on traditional relational databases in todays time, where new database technologies like NoSQL exist which are much more efficient than the traditional RDBMS in different scenarios. In this section, we will be discussing certain applications using traditional RDBMS technology:

Greenbills - Bill Management System:

Greenbills is a billing management system which allows vendors as well as sellers to manage their bills and store them locally on their mobile devices or they can store them at a centralised repository.

The system provides two ends one for seller and other one for vendor, first let's see the vendors end; Whenever a purchase is made and a bill is generated the Vendor will login to the web portal and fill in the details of the purchase along with the bill number and additional data related to bill, when the form will be submitted a unique QR code will be generated.Now at the user's end, the user will login to his/her respective account and will scan the QR code generated by the vendor after scanning the bill will be saved into the user's local storage and user will be also provided with the option to upload the bill to the central repository so that later on no additional problems will come while migrating to an another device.Users will be able to see all the bills related to their purchase and will be able to manage them accordingly, the system will also provide the users data regarding their type of purchases and will be able to sort data according to the type of purchase, date of purchase and vendor's name. A counter will be also provided in the system which will calculate the total amount spent by the user till now.

Compiler on Cloud:

The scope of this project consider the user profile customization and file management functions. This also includes the provision of an online programming suite and private cloud infrastructure based compilation and execution of user files. Also for the intent of load balancing, the scope covers the distributed architecture inclusion. The project solves the basic lacking flexibility problem in diverse range of compilers available in the community market for developers. For eg: one can consider the DEV C++ compiler which restricts the use of certain libraries and function namely:

```
#include<conio.h>
clrscr()
getch()
```

While if one consider the C language compiler TURBO C, all above libraries and functions are prominently supported. This shows there are certain constraints on use of grammar and semantics embedded in the execution and compilation scripts of these compilers. This one drawback is addressed by this project by providing a universal and uniform compiling and runtime execution solution. The Operating System requirements is Windows on both client and server side and SQL Server is the most favored option for Databases needs. The Microsoft IIS (Internet Information Services) web server is found to be an optimum choice for Server commitments.

IV. Improvements to the Current Systems/ Proof of Concept

Greenbills

The technologies used by the developers to build the system are:

Java ME for android application development

PHP and MySQL for backend and server storage

For mobile application Java is used which is the best choice as the developers were targeting only the android devices but if they would have opted for cross-platform development then hybrid app development environments like React Native or Ionic would have been a much better choice.

As the developers were trying to handle a huge repository of data PHP as a server side language is not a good choice because everytime the PHP code runs it gets converted into bytecode and recompiled again which makes it slow.

The database technology developers are using in the System is MySQL which I think is not appropriate at all because of the following reasons.

As the main motto of the system is to store bills and each bill has a different kind of format so it is very complex to store the bill data in a predefined schema because each time the when the bill needs to be stored it has to be converted into a proper format which the schema supports which in turn removes the flexibility of the system.

As there are no update and delete operations getting performed on the database using a storage technology like Column based NoSQL databases would be more efficient as they provide eventual consistency and provide partial retrieval of data which is much more efficient for analysis of data.

In the non-functional requirements the developers are promising a secure environment, but in the first stage itself the data can be breached as the vendor will be providing the user with a QR code which will let user access the content of the purchase he/she has made.

The complete project can be accomplished using the blockchain, as blockchain supports NoSQL column database technology.

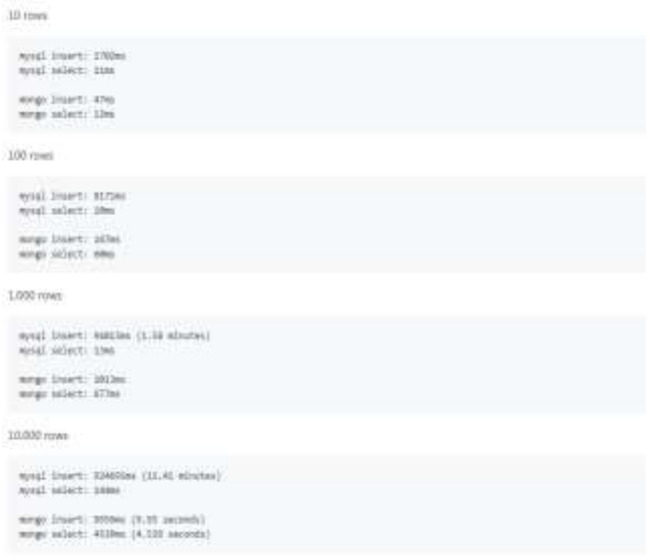
Compilers on Cloud

The aforementioned system extensively involves file creating, saving, updating, deleting i.e more of file management functions. For storage purposes the above system employs the SQL server which is a relational database management system (RDBMS) developed by Microsoft. SQL server supports single machine to large Internet facing workload applications with primary query operations done in T - SQL (Transact SQL) which is standardized for set operations in Sybase and Microsoft. Also ANSI SQL is considered for compliance purposes. In SQL , more specifically in RDBMS, the files are stored in the form of BLOB (Binary Large OBjects). This process includes first conversion of files into binary formats and then storing them as BLOBs in database. At the time of retrieval, these BLOBs are first to be reverse converted to finally gain access to the original file. Also there are certain problems associated with the use of BLOBs mainly :
Large BLOBs may slow down your server which does not only host database.
Storing files as BLOBs are hard to maintain especially if the general operation involves periodic updates on original files.
Backups of such databases are very difficult and critical in case of sensitive file data.
Migration of data (eg: exporting your database) turns out to be a task of hassle with large BLOBs.

Since it is evident from above summary that the system in consideration involves heavy amount of file storage and updation, using BLOBs is not a proper option. One of the alternatives to storing files as BLOB is storing files in filesystem (project directory structure). This process involves storing files in particular directory of the filesystem and storing the path address of the file in the database. Although this overcomes few discrepancies in case of BLOBs, but this too may vary down in case of rapid data retrieval and large amount of large files containing huge amount of data.
In this case , the files containing programming codes.

Based on above analysis we can affirm the NoSQL alternative for this project/system/web application can be MongoDB. MongoDB is a cross platform document oriented NoSQL database. It is one of the most efficient NoSQL databases existing till date. Applications of MongoDB include file handling, text files, xml files, etc. Since it is a document oriented database, it stores data in the form of documents. A GitHub based researcher Andre Caetano presented a node.js built application designed to compare SELECT and INSERT performances of MySQL and MongoDB depending upon the count of rows affected.

A snapshot of 10, 100, 1000 and 10000 affected rows is given below:



10 rows

```
mysql insert: 1702ms
mysql select: 11ms

mongo insert: 47ms
mongo select: 12ms
```

100 rows

```
mysql insert: 8172ms
mysql select: 18ms

mongo insert: 167ms
mongo select: 60ms
```

1,000 rows

```
mysql insert: 91813ms (1.53 minutes)
mysql select: 13ms

mongo insert: 1012ms
mongo select: 677ms
```

10,000 rows

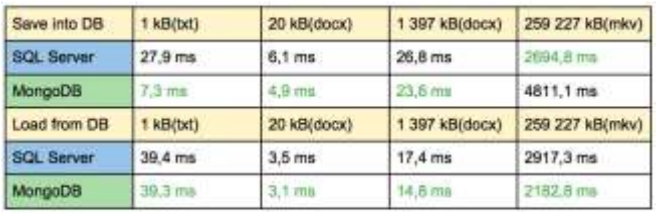
```
mysql insert: 324650ms (11.41 minutes)
mysql select: 168ms

mongo insert: 9550ms (9.55 seconds)
mongo select: 4110ms (4.110 seconds)
```

Figure 1 : Snapshot of Comparison

If we consider the maximum number of rows i.e. 10000, it is clearly evident that there is a significant execution time difference between MySQL and MongoDB insert operation (5 seconds is significant if the computation is considered). Even if we consider the minimum number of row i.e 10, one can precisely note the large timing difference (in ms) of MySQL and MongoDB insert operations.

In contrast with the idea of using the SQL for the above system, we can consider the execution times of file insertion in file retrieval operations in MySQL and MongoDB. Barna Burom, a Budapest based TypeScript and Angular developer presented a concise and practical comparison of file saving and loading query operation execution times of SQL Server and MongoDB. The three types of files considered for this test were .txt (text files), .docx (word files) and .mkv (video files). The below table shows the comparison results as provided by the developer himself:



Save into DB	1 kB(txt)	20 kB(docx)	1 397 kB(docx)	259 227 kB(mkv)
SQL Server	27,9 ms	6,1 ms	26,8 ms	2694,8 ms
MongoDB	7,3 ms	4,9 ms	23,6 ms	4811,1 ms
Load from DB	1 kB(txt)	20 kB(docx)	1 397 kB(docx)	259 227 kB(mkv)
SQL Server	39,4 ms	3,5 ms	17,4 ms	2917,3 ms
MongoDB	39,3 ms	3,1 ms	14,6 ms	2182,8 ms

Figure 2 : Mongo - SQL File Operations

From above one can simply conclude that MongoDB is quite an efficient option in case one's application involves file storing and retrieval functions.

V. Conclusion

Although from above evidences and research findings, it is implied that mysql is inefficient in multiple operations. NoSQL can be used as at places where MySQL fails in terms of performance, flexibility, data analysis etc. In some cases even NoSQL fails eg: consistency but is a success when it comes to eventual consistency. NoSQL can be used as an alternative to SQL.

References

Journal Papers:

[1]. Jagdev Bhogal, Imran Choksi, *Handling Big Data using NoSQL*, IEEE, 2015.
[2]. Yishan Li, Sathiamoorthy Manoharan, *A performance comparison of SQL and NoSQL databases*, IEEE, 2013.
[3]. Rashid Zafar, Eiad Yafi, Megat F. Zuhairi, Hassan Dao, *Big Data : The NoSQL and RDBMS review*, IEEE, 2016.
[4]. Surya Narayana Swaminathan, Ramez Elmasri, *Quantitative Analysis of Scalable NoSQL Databases*, IEEE, 2016.
[5]. Venkat N Gudivada, Dhana Rao, Vijay V. Raghavan, *NoSQL Systems for Big Data Management*, IEEE, 2014.
[6]. Bogdan George Tudorica, Cristian Bucur, *A comparison between several NoSQL databases with comments and notes*, IEEE, 2011.

Theses:

[7]. Ansari Mohd. Arshad, Arshiya Khan, Shaikh Sana,Er. Zainab Mirza, *Compilers on Cloud,* IJERT, 2013.
[8]. Er. Zainab Mirza, *Greenbills,* 2015.

Web:

[9]. https://github.com/webcaetano/mongo-mysql
[10]. https://codingsans.com/blog/nosql-vs-relational-database
[11]. https://www.microsoft.com/en-us/sql-server/sql-server-2017