JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Framework of Mongodb For Handling Big Data And Development of Mongosight For Providing Easy Operations

[1]Gangappa B Demannavar, [2]Manjunath R, [3]Sowmya Naik P T, [4]Rakesh M
[1,4]Assistant Professor, Department of CSE, City Engineering College, Bengaluru, Karnataka, India
[2]Professor, Department of CSE, R R Institute of Technology, Bengaluru, Karnataka, India
[3]Professor, Department of CSE, City Engineering College, Bengaluru, Karnataka, India

Abstract : With the emergence of Big Data, the use of NoSQL (Not only SQL) technology is growing Fast among internet companies and other enterprises. The Benefits are simplicity of design, horizontal scaling and control over availability.NoSQL databases are increasingly considered a viable alternative to relational databases, as more organizations recognize that schema less data model is a better method for handling the large volumes of structured, semi structured and unstructured data, being captured and processed today. For example NoSQL databases are often used to collect and store social media data.The aim of the project is to introduce the concept of NoSQL, which provides a review of relevant literature, highlights the different NoSQL database types, and provide arguments for and against adopting NoSQL.A small prototype has been developed to assess the stated NoSQL features and illustrate the differences between the SQL and NoSQL approaches. The last section of the paper offers some conclusions and recommendations for further research which helps to expand upon our research work.

Index Terms: NoSQL, SQL, databases, structured data, unstructured data, Big Data

I. INTRODUCTION

Big Data is a phrase which refers to a large volume and wide variety of data being captured from different sources at high speed. It is estimated that data volume is increasing 40% per year, and will grow 44 times between 2009 and 2020 [1]. Much of this data is of a textual nature and hence unstructured. With the emergence of Big Data, the use of NoSQL technology is rising rapidly among internet companies and the enterprise. Benefits include simplicity of design, horizontal scaling and finer control over availability. NoSQL databases are increasingly considered a viable alternative to relational databases, as more organizations recognize that its schema less data model is a better method for handling the large volumes of structured, semi structured and unstructured data, being captured and processed today. For example NoSQL databases are often used to collect and store social media data. The purpose is to introduce the concepts, highlight the different NoSQL database types, and provide arguments for and against adopting NoSQL. Hadoop deals with big data which is an open source Java framework. Big Data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, accurate, manage, and process data within a tolerable elapsed time. There are two core components in it namely: HDFS (Hadoop distributed file system) is the ability of a system to continue normal operation against hardware or software faults using inexpensive hardware and which stocks huge extent of data another one is MapReduce is a processing technique and programming model done in lateral and scattered manner. MapReduce provides a standardized framework for implementing large-scale distributed computation, namely, the big-data applications.

II. BACK GROUND OF NO SQL

NoSQL databases are non-relational and accommodate unstructured data. It is not a replacement for an SQL database but compliments it; both technologies can coexist. The key difference is that relational (SQL) databases have rigid schemas while NoSQL databases offer a flexible schema design that can be altered without downtime or service disruption. NoSQL was also designed for distributed data stores for large scale data needs; for example Facebook has 500 million users and Twitter accumulates terabytes of data. NoSQL takes advantage of scaling out, by spreading its load over many database servers, and this is an inexpensive solution for large datasets. However in comparison to relational databases, NoSQL databases do not have the same distinct properties or data integrity [2]. In spite of this NoSQL databases have an established track record for handling Big Data efficiently.The primary way in which NoSQL databases differ from relational databases is the data model; there are numerous NoSQL databases [3] and they primarily fall into the following four categories:Key Value Databases use a hash table where there is a unique key and a pointer to a specific set of values; data can only be queried by the key. The data can be unstructured, as it does not enforce a set schema across key value pairs. Facebook uses this database type, as the datasets are not related to each other and the data is unstructured. The simplicity of this database type makes it ideally suitable for fast highly scalable retrieval of values needed for application tasks like managing customer profiles and

retrieving product names. Amazon uses its own key value database, DynamoDB, for its shopping cart. These databases were built for quick and efficient data management in distributed systems. Figure 1 shows Key Value pairs for a Cars database [4].

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Figure1: Key Value Database (ReadWrite, 2009)

Column Oriented Databases store groups of related data into column families and rows with many columns are associated with a row key. Each record can differ in the number of columns stored, and columns nested inside other columns are called super columns. This database type is extremely scalable and works well with more complex datasets. Online Analytical Processing uses this for rapidly computing a big amount of data into one result [5]. Google uses a distributed data storage system, BigTable, for its package Google Earth. These databases were created to store and process enormous amounts of data in distributed systems, especially versioned data because of its time stamping functions. Other examples of this NoSQL category include Cassandra and Hbase. An example of a column Oriented database using Customer Information is shown in Figure 2 [6].

Row Key	Column Families	
CustomerID	CustomerInfo	AddressInfo
1	CustomerInfo:Title Mr CustomerInfo:FirstName Mark CustomerInfo:LastName Hanson	AddressInfo:StreetAddress 999 500th Ave AddressInfo:City Bellevue AddressInfo:State WA AddressInfo:ZipCode 12345
2	CustomerInfo:Title Ms CustomerInfo:FirstName Lisa CustomerInfo:LastName Andrews	AddressInfo:StreetAddress 888 W. Front St AddressInfo:City Boise AddressInfo:State ID AddressInfo:ZipCode 54321
3	CustomerInfo:Title Mr CustomerInfo:FirstName Walter CustomerInfo:LastName Harp	AddressInfo:StreetAddress 999 500th Ave AddressInfo:City Bellevue AddressInfo:State WA AddressInfo:ZipCode 12345

Figure 2: Column Oriented Database (Microsoft, 2013)

Document Databases use entire documents of structured data files, such as XML or JSON, as datasets. Each record and its associated data are usually stored together in a single document; this simplifies data access and reduces the need for joins or complex transactions. Documents stored are schema free and similar to each other; this flexibility can be particularly helpful for modelling unstructured data.

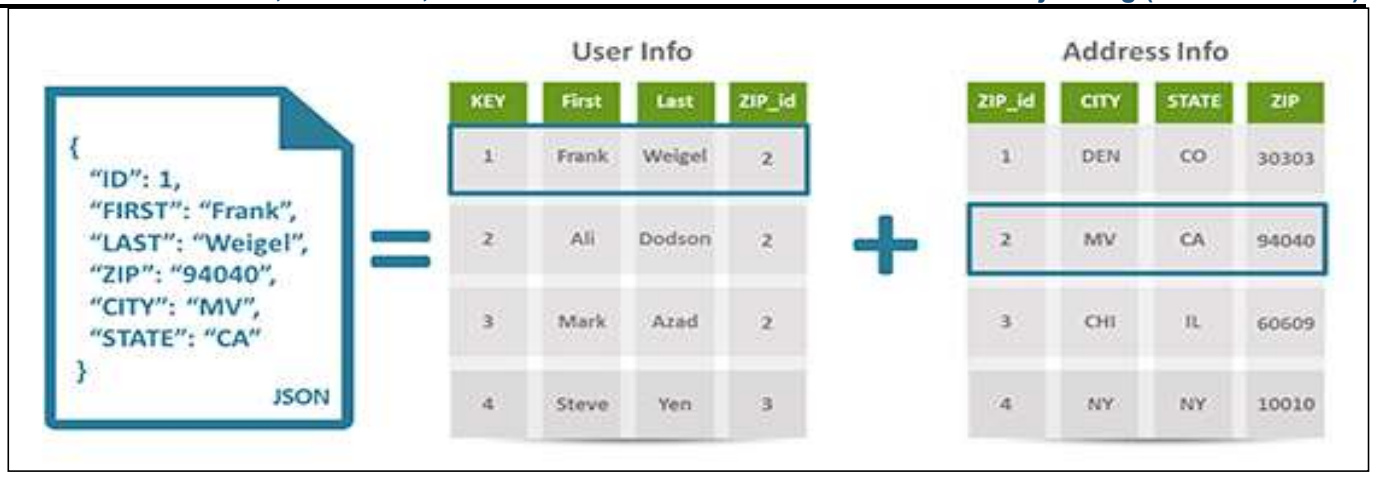

Figure3: Document Database (Couchbase, 2013)

An example of a document database is shown in Figure 3 [7]. Document databases also provide the same query robustness as relational databases; data can be queried using both the keys and values in a document. These databases are ideal for storing and managing Big Data size collections of literal documents like text documents and email messages.Examples of NoSQL Document software include CouchDB and MongoDB.

Graph Databases are built with nodes, the relationship between nodes and the properties of nodes; data is modelled as a network of relationships between specific key value pairs. Graph databases are useful when there is more interest in the relationships between data than in the data itself. Location based services use this database type, to find mutual friends on Facebook or to establish the shortest paths through everyday traffic (see Figure 4 [8]). While it takes relational databases hours to sort through a huge linked list of people, graph databases use sophisticated shortest path algorithms to make data queries more efficient. Most of these databases are schema free and easy to scale horizontally. Examples of Graph Databases include InfoGrid and Neo4J.

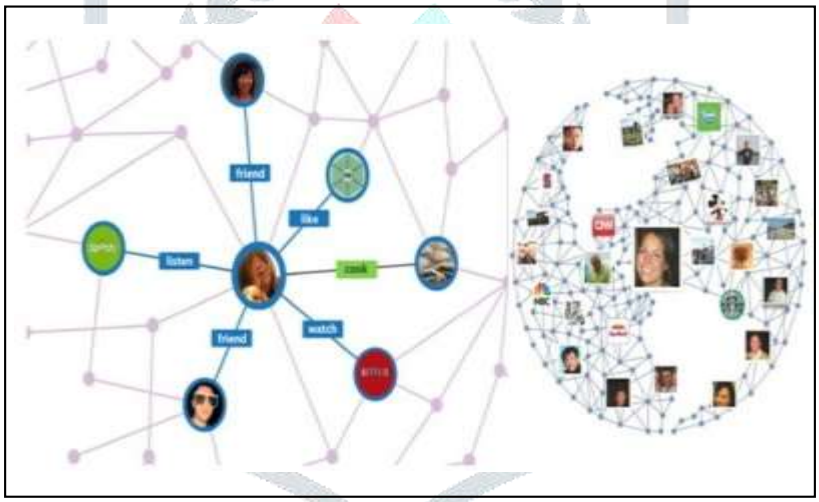

Figure4: Graph Database (Majestic Media, 2012)

NoSQL databases are reinventing data management by empowering businesses to be more agile and scalable. Organisations are using NoSQL technology to create new applications, improve customer experience and reduce costs. For example, a global insurance provider used NoSQL technology to rapidly combine customer information from over 70 existing systems and provides it in a single rapidly updated 360 degree view. As a result the provider reduces the time to resolve customer issues. Another example is that of Chicago City using NoSQL technology to reduce crime and improve public services by collecting and analysing geospatial data in real time from over 30 departments. It can evaluate the number of 911 calls and complaints in a given district, to determine if an uptake in crime is more likely than usual.Whilst NoSQL is great for dealing with many difficulties surrounding unstructured data, it is however limited in several key areas. Stonebraker believes “Those who do not understand the lessons from previous generation systems are doomed to repeat their mistakes” [9]. This statement may be justified with NoSQL having barriers that include consistency and reliable standards. Since NoSQL data is heldin partitions either availability or consistency cannot be 100% guaranteed, and this has an impact on the ability to handle complex queries.

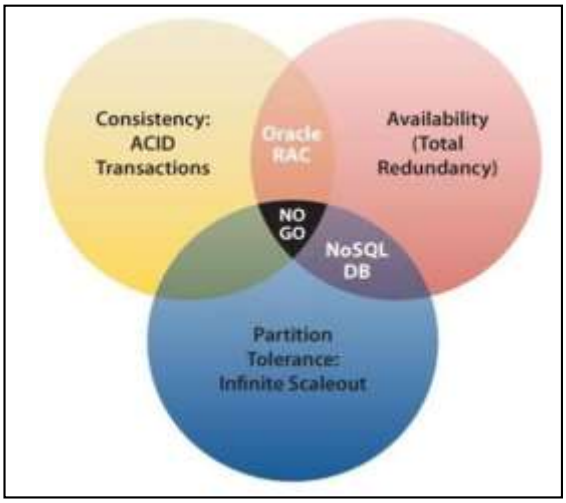

Figure5 CAP Theorem (BigDataNerd, 2012, [10])

However Big Data is creating big opportunities for NoSQL; it is an inexpensive and effici ent way to manage data [11]. For example when combined with MapReduce technology, NoSQL can perform complex operations on extremely large datasets, leaving traditional data warehouses and business intelligence applications far behind in terms of performance and cost. The CAP theorem (see Figure 5) illustrates that distributed systems can have at most two of three desirable properties: consistency, availability and partition tolerance [12].NoSQL has not reached maturity. A survey conducted by Information Week [13] highlighted that 44% of business IT experts had not heard of NoSQL and only 1% indicated that NoSQL was a part of their strategic direction. NoSQL technologies make it possible to realise value from Big Data, and this helps businesses achieve their strategic goals and generate new revenue streams.

NoSQL databases are increasingly considered a viable alternative to conventional databases, as more businesses recognise that its schema less model is a better method for handling the large volumesof semi structured and unstructured data, being captured and processed today[7]. In many businesses it is inevitable that NoSQL databases will play a significant role alongside conventional database systems. A NoSQL database is not a replacement for an SQL database but compliments it; the idea is that both technologies can coexist.NoSQL databases support numerous use cases for interactive applications; they have successfully been used by organisations across many industries for online, mobile and cloud enterprise solutions. For mobile applications, NoSQL databases can power the backend system. Facebook updates its mobile application every few months and relational databases cannot support this pace of innovation. NoSQL incorporates changes to the database when the application evolves and is flexible enough to support several smartphone types. In the case of product catalogues information is stored for product data management and ecommerce websites. NoSQL databases have the capacity to store various types of objects with different sets of attributes, all within a single database. Its dynamic schema allows businesses to add products and incorporate new features without disruption; for example customers can submit product reviews and all this data is embedded within the products database.

III. NoSQL PROTOTYPE

The product catalogue use case has been selected; a prototype system will be developed for a library inventory which will enable the benchmark testing of both SQL and NoSQL databases. The NoSQL prototype will be implemented using a document database; they are built for agile development and its dynamic data model allows the storage of unstructured data. We examine Big Data in the context of libraries. Libraries not only store an enormous amount of traditional electronic data for their product catalogues and membership information. Modern libraries can also capture Big Data from social media sites such as Twitter whereby library members can give feedback regarding the lending service and the quality of the products in the collection. Library databases store numerous products, all with various attributes and belonging to many categories ie the Variety of data from the 3 V's of Big Data. For example books come in a wide variety of subjects, while films have completely different attributes. Document databases have a flexible design that allows users to save the right product information and attributes for each single item; this eliminates the empty fields that usually occur in a relational database.

A. RELATIONAL DATABASE DESIGN – SQL

Relational databases organise data in tables, which are made up of rows and columns. Tables cannot have duplicate rows because this creates ambiguities during queries; to prevent this each table has a primary key column that uniquely identifies every record. For example Figure 6 shows that Product_ID is the primary key for the Product table. Column Author_ID in the child Product_Book table is the foreign key and this is used to reference the parent Author table. Primary and foreign keys are used for implementing referential integrity relationships between tables.ER modelling is used to design the libraries.

Relational database schema; this identifies the tables, attributes for each table and illustrates the relationships between each table. The library database has 2 one-to-many relationships, and 4 many- to many relationships. Additionally all the tables have been fully normalised; this process removes all redundant data from tables, in order to improve storage efficiency and data integrity. However normalisation can impact performance, as additional table joins may be required during data retrieval.

The libraries database employs multiple table inheritance to store common attributes in a generic Product table (see Figure 6); all distinctive attributes are stored in individual category product tables. This is more efficient than concrete table inheritance whereby a new table is created for each product category and queries are tailored for individual products. However multiple table inheritance requires many join operations to obtain all the relevant attributes of a given product.

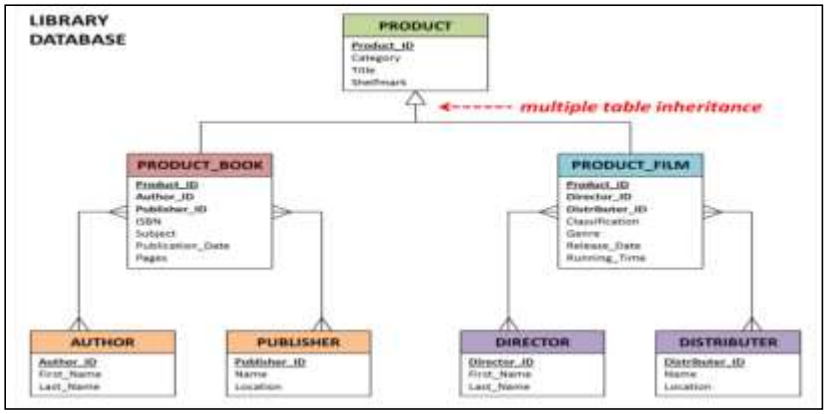

Figure6: Relational Database Schema

B. *DOCUMENT DATABASE DESIGN – NOSQL*

Document databases store data in the form of documents that are JSON like key value pairs; they are similar to rows in relational databases. Documents usually contain many different key value pairs, and keys may hold other key values. Documents are stored in collections; these are groups of related documents that have shared common indexes. Collections are similar to tables in relational databases. Figure 7 illustrates the library catalogues document database schema.

Document databases have a flexible schema; SQL databases require a tables schema to be declared before inserting data, but collections do not impose document structure. This flexibility enables documents to match the data fields of any entity, even when the data is substantially varied. Whilst designing database schemas key decisions revolve around the documents structure and how applications represent the data relationships. The relationships between Document databases can be represented using either the embedded or reference approaches. Embedded documents store related data in a single document; this denormalised data model allows systems to query and update related data in a single database operation. Figure 7 illustrates that film product documents have embedded fields that contain its director and distributor information. The embedded model is used when there is one- to-many relationships between entities; child documents always appear within the parent document. Embedding delivers better performance for read operations, but documents may grow after creation and this can impact write performance. The reference approach stores the relationships between data by providing links from one document to another; this is a normalised data model. Document databases do not support joins; therefore related data is often embedded to reduce the need for joins. But if embedding results in duplicate data, it is more efficient to store data in separate documents. Figure 7 illustrates that book product documents contain a reference to both the author and publisher documents. The reference model is used when there are numerous many-to-many relationships, as it provides more flexibility than embedding.

Figure7: Document Database Schema

A. *CHOICE OF SOFTWARE*

There are numerous relational databases that store data in tables and operate through queries. Oracle Application Express (APEX) was selected for implementing the libraries relational database. Oracle APEX is a freeware software development environment running inside the Oracle database. Its web browser interface allows inexperienced programmers to rapidly develop data orientated applications. APEX uses wizards and declarative programming to create powerful data entry applications, and contains a graphical query builder. Its SQL Workshop component is used to manage database objects and run ad hoc queries [14].There are several NoSQL databases that store data in JSON like documents and offer atomic read write operations. When deciding whether to use NoSQL, companies need to consider the business model, ACID transactions demand, cost and other requirements [15]. MongoDB was selected for implementing the libraries document database, because its flexible data model allows a prototype system to evolve during development without modifying existing data [16, 17]. Its embedding model eliminates join operations and enhances query performance. It is also horizontally scalable and can reduce workload by adding more commodity servers or cloud instances [18,19]. MongoDB provides a database system that stores rich data structures, executes complex queries and scales out. MongoDB allows the library to insert data without a predefined schema and this is useful for storing products with different attributes.

IV. IMPLEMENTATION

Firstly a relational database solution was constructed using Oracle APEX. Thereafter a document database solution was constructed using MongoDB; this design incorporates both embedding and reference data models. The purpose of this section is to illustrate the fundamental steps of each implementation and highlight the significant differences between both approaches. MongoDB employs insert operations to create data; the db.collection.insert() statement adds new documents to a collection. The _id field is automatically generated for a new document if the field is not specified. The libraries document database system employs both embedded documents and references; this allows the author to demonstrate both data models.

A. EMBEDDED DATA MODEL

Embedded documents store related data in a single document; film product documents embed the director and distributor attributes. The following statement inserts a film into the Products collection (see Figure 8):

```
> db.Products.insert({
... _id: "BCU000013",
... Category: "Film",
... Title: "Harry Potter and the Philosophers Stone",
... Shelfmark: "COL.P02/01",
... Classification: "PG",
... Genre: "Fantasy",
... Release_Date: "November 2001",
... Running_Time: 152,
... Director:{
... First_Name: "Chris",
... Last_Name: "Columbus"},
... Distributer:{
... Name: "Warner Brothers Pictures",
... Location: "California"}})
```

Figure8: Insert Embedded Document Into Products Collection

The below figure9 confirms a film was inserted into the Products.

```
> db.Products.find({_id : "BCU000013"}).pretty()
{
        "_id" : "BCU000013",
        "Category" : "Film",
        "Title" : "Harry Potter and the Philosophers Stone",
        "Shelfmark" : "COL.P02/01",
        "Classification" : "PG",
        "Genre" : "Fantasy",
        "Release_Date" : "November 2001",
        "Running_Time" : 152,
        "Director" : {
                "First_Name" : "Chris",
                "Last_Name" : "Columbus"
        },
        "Distributer" : {
                "Name" : "Warner Brothers Pictures",
                "Location" : "California"
        }
}
```

Figure9: View Embedded Document In Products Collection

A. REFERENCE DATA MODEL

References link one document to another; book product documents contain references to both author and publisher documents. The statement in Figure 10 inserts a book into the Products collection.

```
> db.Products.insert({
... _id: "BCU000001",
... Category: "Book",
... Title: "Harry Potter and the Philosophers Stone",
... Shelfmark: "ROV.B01/97",
... Author_id: "AUT0001",
... Publisher_id: "PUB0001",
... ISBN: "0-7475-3269-9",
... Subject: "Fantasy",
... Publication_Date: "June 1997",
... Pages: 223})
>
```

Figure10: Insert Document Into Products Collection

We can use similar statements to insert data into theProducts collection and the Authors collection. The implementation phase successfully provided two prototype systems; each of which uses a distinctive approach to insert data. MongoDB enables the library to dynamically incorporate new productcategories, whereas APEX would require the libraries schema to be explicitly redefined before it can insert new product types. The libraries prototype systems allow the author to benchmark test the CRUDfunctionality of both databases, during the testing phase.

VI. CONCLUSION

Both prototype systems effectively address the libraries data storage requirements, but differ in their approach. APEX requires table structure to be defined before adding data whereas MongoDB does not explicitly create collections; document structure is defined automatically during the first data insert. MongoDB enables the library to either embed product attributes into a single document or reference the data in another document. However with a normalised SQL database, product attributes might be spread over numerous tables; therefore complex SQL joins are required, to view all the attributes of a product. Also referential integrity rules specify that the libraries SQL database should only insert records with Author_ID into the Product_Book table, if the Author_ID exists in the Author table. The NoSQL database is very efficient and allows the library to obtain all film attributes using one simple query.NoSQL databases make it possible to realise great value from Big Data and empowers businesses to be more agile and scalable; this helps businesses achieve their strategic goals and generate new revenue streams. Most new data is unstructured and the rigid schema based approach adopted by relational databases, makes it is impossible to

incorporate all new data types. Sensor data can be used for exploring the concept of data variety further [20]. Future work on using NoSQL to evaluate the remaining 3 V's of Big Data is required [21]. Benchmark tests can be conducted to assess the handling of data volume and velocity with the deployment of NoSQL on Hadoop clusters and the use of real-time social media data. The social media data could be in the form of text-based product reviews and the prototype application could be extended to incorporate sentiment analysis. NoSQL databases are becoming a worthwhile alternative to relational databases, as its dynamic data model is much better for managing large quantities of unstructured data; additionally its schema can be modified without downtime or service disruption.

References

[1] Mckinsey Global Institute, "Big Data." [Report, 2011] New York: McKinsey Global Institute.

[2] S. Edlich, "NoSQL Databases." [Report 2010] Chur: University ofApplied Sciences HTW Chur.

[3] B.G. Tudorica, C. Bucur, , "A comparison between several NoSQL databases with comments andnotes," Roedunet International Conference (RoEduNet), 2011 10th , vol., no., pp.1,5, 23-25 June 2011.

[4] Readwrite,"Key Value Database." [Image Online] Available at: http://readwrite.com/2009/02/12/is-the-relational-database- doomed#awesm=~ookpfnVjGmGXHJ [Accessed: 13 Nov 2013].

[5] L. Bonnet, A. Laurent, M. Sala, B. Laurent, N. Sicard "Reduce, you say: What nosql can do for data aggregation and bi in large repositories." In Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on (pp. 483-488). IEEE.

[6] Microsoft, "Column Oriented Database", [Image Online] Available at: http://msdn.microsoft.com/en-us/library/dn313285.aspx#sec7 [Accessed: 13 Nov 2013].

[7] Couchbase, "Document Database," [Image Online] Available at: http://www.couchbase.com/why-nosql/nosql-database [Accessed: 13 Nov 2013].

[8] Majestic Media, "Graph Database." [Image Online] Available at: http://www.majesticmedia.ca/blog/2012/06/new-facebook-guidelines- for-open-graph-actions/ [Accessed: 13Nov 2013].

[9] M. Stonebraker, "Why Enterprises Are Uninterested in NoSQL." [Online] Available at: http://cacm.acm.org/blogs/blog-cacm/99512-why- enterprises-are-uninterested-in-nosql/fulltext [Accessed: 02 Oct 2010].

[10] Bigdatanerd. "CAP Theorem." [Image Online] Available at: http://bigdatanerd.wordpress.com/2011/12/08/why-nosql- part-1-cap-theorem/ [Accessed: 13 Nov 2013].

[11] R Manjunath, RK Channabasava, S Balaji, "A Big Data MapReduce Hadoop distribution architecture for processing input splits to solve the small data problem", 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), DOI: 10.1109/ICATCCT.2016.7912048, IEEE, pp. 480-487, 2016.

[12] R Manjunath, A Akshatha, S Balaji, "Reverse engineering in Big Data using Cloud computing and Open Stack virtual machine" 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), DOI: 10.1109/ICATCCT.2016.7912118, IEEE, pp. 848-853, 2016.

[13] Vinod Kumar N, Manjunath R,"A Survey of Data Aware Caching for Big-Data Applications using the MapReduce Framework', International Journal of Engineering Research & Technology (IJERT), DOI:10.17577/IJERTCONV3IS27110, Volume3, Issue27, pp. 1-10, 2018