Database Management Systems: A NoSQL Analysis

Syed Abdul Rahman

PG Scholor, Department of MCA,
Dayananda Sagar College of
Engineering,Bangalore

Rakshitha

Department of MCA,
Dayananda Sagar College of
Engineering,Bangalore

Abstract- **Addressing today's ever increasing changes in data management needs require solutions that can achieve unlimited scalability, high availability and massive parallelism while ensuring high performance levels. The new breed of applications like business intelligence, enterprise analytics, Customer Relationship Management, document processing, Social Networks, Web 2.0 and Cloud Computing require horizontal scaling of thousands of nodes as demanded when handling huge collections of structured and unstructured data sets that traditional RDBMS fail to manage. The rate with which data is being generated through interactive applications by large numbers of concurrent users in distributed processing involving very large number of servers and handling Big Data applications has outpaced the capabilities of relational databases thereby driving focus towards the NoSQL database Adoption. NoSQL database systems have addressed scaling and performance challenges inherent in traditional RDBMS by exploiting partitions, relaxing heavy strict consistency protocols and by way of distributed systems that can span data centres while handling failure scenarios without a hitch. In this paper different database management systems are discussed and their underlying design principles namely ACID, CAP and BASE theorems respectively, are evaluated.**

Keywords: Database Management Systems, Relational Databases, NoSQL Databases, ACID, CAP, BASE

I. INTRODUCTION

The advent of computer systems and the rapid changes in industrial dynamics on several fronts including research and technical knowledge increased the demand on quality and productivity of products and services. This saw the automation of real world processes and the introduction of Assembly Automation Equipment, Automated Bookkeeping and Manufacturing systems among a many others. These systems were capable of manipulating only textual and numerical data using Flat file databases as a data management system. This enabled measurement, collection, transcription, validation, organisation, storage, aggregation, update, retrieval and protection of data.

A Flat file database describes any of the various means toencode a database model (most commonly a table) as a singlefile. Flat file databases contained a logical collection of records with no structured relations which were in plain text or binary file.

Flat file databases at the time were quite useful as data management requirements were still very limited and simple. With further advances in technology, flat file databases became inadequate as they could not cater for new data types, data security and growth requirements. Also flat file databases contained no information about data and additional knowledge was required to interpret the files. There was no standard way of storing data as well as a standard of communicating to and from the database, hence it created a lot of inefficiencies.

In the 1970s cord came up with the relational theory that led to the development of the relational Database Management Systems (RDBMS) as a solution to the challenges posed by the flat file database system in the earlier years. Storage of data in RDBMS was done using Tables. Standard fields and records are represented as columns (fields) and rows (records) in a table. Their major advantage was the ability to relate and index information. Security was enhanced in RDBMS and they were also able to adapt to considerable growth of data. Structured Query Language, SQL is the programming language used for querying and updating relational databases. For a long time RDBMS has been the preferred technique for data management purposes. However, RDBMS inability to handle modern workloads has given rise to scalability, performance and availability problems with its rigid schema design. Businesses all over the world, including Amazon, Facebook, Twitter, and Google have adopted new ways to store and scale large amounts of data hence the move away from the complexity of SQL based servers to NoSQL database Systems. NoSQL is a class of database management systems that have been designed to cater for situations in which RDBMSs fall short. It is different from the traditional relational databases mainly in that it is schema-less. This makes it suitable to be used for unstructured data. These engines usually provide a query language that provides a subset of what SQL can do, plus some additional features [1]

NOSQL DATABASES

The NoSQL database approach is characterized by flexibility *iii.* in storage and manipulation of data, improvements in performance and allowing for easier scalability.Many different types of these NoSQL databases exist, each one suited for different purposes. Examples include MongoDB whose deployments are at foursquare, Disney, bit.ly, sourceforge, CERN, The New York Times, and others. Hadoop (Apache), Cassandra was primarily used by Facebook for their Inbox Search. Afterwards it was open-sourced and now it is an Apache Software Foundation top-level project, being used by Digg, Twitter, Reddit, Rackspace, Cloudkick, Cisco and others. DynamoDB is used by Amazon, Voldemort is used by Amazon, and Neo4J is used by Adobe and Cisco etc. While RDBMS is transaction oriented and based on the ACID principle, NoSQL make use of either CAP or

Among several capabilities of NoSQL databases are managing large streams of non-relational and unstructured data, fast data access speeds, availability of data even when system is operating in degraded mode due to network partitions. NoSQL databases provide near-endless scalability and great performance for data-intensive use cases. However, with so many different options around, choosing the right NoSQL database for your interactive Web application can be tricky. In general, the most important factors to keep in mind are as follows

Among several capabilities of NoSQL databases are managing large streams of non-relational and unstructured data, fast data access speeds, availability of data even when system is operating in degraded mode due to network partitions. NoSQL databases provide near-endless scalability and great performance for data-intensive use cases. However, with so many different options around, choosing the right NoSQL database for your interactive Web application can be tricky. In general, the most important factors to keep in mind are as follows:

Scalability. Adopting the Sharding technique can be useful in achieving scale regardless of the database technology in use. Sharding employs horizontal partitioning which is a database design principle in which rows of a database table are held separately .These tables may then be located on a separate database server or physical locations. Scaling quickly, on demand, and without any application changes become a determinant factor in Web traffic that has on and off surges. Resource contention between servers like disk, memory and CPU is removed. Intelligent parallel processing and maximization of CPU/Memory per database instance can be done.

i. Performance. Interactive applications require very low read and write latencies. Performance is achieved by distributing load across several servers**.** The database must deliver consistently low latencies regardless of load or the size of data. As a rule, the read and write latencies of NoSQL databases are very low because data is shared across all nodes in acluster while the application's working set is inmemory.

ii. Availability. Interactive Web applications need a highly available database. If your application is down, you are

simply losing money. To ensure high availability, your solution should be able to do online upgrades, easily remove a node for maintenance without affecting the availability of the cluster, handle online operations, such as backups, and provide disaster recovery, if the entire data centre goes down.

Ease of development. Relational databases require a rigid schema and, if your application changes, your database schema needs to change as well. In this regard, NoSQL databases offer a number of important advantages that make it possible to alter data structure without affecting your application

Supporting distributed processing of large-scale data workloads requires adequate processing frameworks likes Apache Hadoop with the MapReduce engine. The emergence of new forms of traffic profiles driven by the Social Web as well as the growing popularity of E-commerce coupled by the ever increasing interconnectedness of the World where Sites are

experiencing variations of traffic through-out the year has resulted in massive surges of writes and read traffic in Sites like Twitter, Facebook, Whatsapp in very short time frames hence the need for infrastructure that adapt quickly. Massive upswings on volumes of data movement across the Internet into storage solutions might have traffic becoming a bottleneck. The popularity of agile development methods call for techniques that offer higher scalability and performance so as to keep up with the ever changing technical environment. In-memory database for high update situations, like a website that displays everyone's "last active" time (for chat maybe). Ifusers are performing some activity once every 40 seconds, then it will push RDBMS to limits with about 5000 simultaneous users for instance, what when the numbers multiplies by 10.

a. NOSQL DATABASE CATEGORIES

A. KEY VALUE STORES

Provide a way of storing schema-less data by means of a distributed index for object storage. The key (data-type) will be displayed on the left and the corresponding value (actual data) on the right as shown in the example below.

Key	Value
Comp3_manufa	Dell
Comp20_processor	IntelCore_i5
Comp3_installedMemory	4GB
comp230_systemType	64-BitOS

Figure 1: Key Value Store

Key/Value store is best applicable where write performance is of highest priority since its schema-less structure allows for fast storage of data.

COLUMN ORIENTED DATABASES

Provide a data store that resembles relational tables but also adds a dynamic number of attributes to the model. They use keys but they point to multiple tables.

Row Key   Columns

Comp3	Brand	processor	Memory
	Dell	IntelCore_i5	4GB
Comp8	Brand	processor	Memory
	Dell	IntelCore2_duo	3GB
Printer42	Brand	Color	Type
	Hp	White	4in1

Figure 2: Column Oriented databases

DOCUMENT ORIENTED DATABASES

Data is treated as independent objects and their attributes which are stored as separate documents. Each document contains unique information pertaining to a single object. Document stores recognise the structure of the objects stored. Read and writes can be accomplished at once thus making it faster in performance. Schema-less structure gives flexibility in the wake of changing technologies. Documents are described using JSON or XML or derivatives.

Comp3	dell	IntelCore_i5	4GB	64BitOS
Comp8	dell	IntelCore2_duo	3GB	32BitOS
Comp2	Compaq	intel_celeron	2GB	64BitOS

```
Comp3
{
_id: "comp3",
Manufacturer: "dell",
Processor:" IntelCore_i5",
Memory:"4GB",
System_type:"64Bit_OS"
}
```

```
Comp8
{
_id: "comp8",
Manufacturer: "dell",
Processor:"IntelCore2_duo"
Memory:"3 GB",
Sys_type:"32 Bit_OS"
}
```
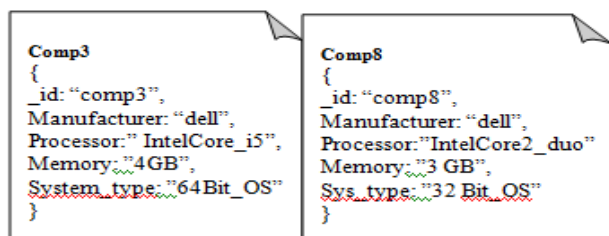
Figure 3: Document Oriented Databases

Figure 4: *Graph Databases*

A. GRAPH DATABASES

These are databases that are based on the graph theory. Graph databases store data in a graph structure with nodes, edges and properties to represent the data. The nodes represent entities in the database. Edges are connecting lines between two nodes representing their relationships.Properties are the attributes of the entities. Graph databases are more applicable in social networks and intelligent agencies as they efficiently show relationships between entities and provide a way to access data in sites with heavy workloads (predominantly reads).

A. OTHER CATEGORIES

The databases discussed above are considered to be the major ones. However, NoSQL has several other categories of databases for various applications. Other types include Multimodel Databases (eg ArangoDB, OrientDB), Object Databases (DB40, Velocity), Grid and C loud Database solutions (Gigaspace, Gemfire), XML Database (BaseX, Berkeley DB XML), Multidementional Databases (SciDB, MiniM DB).

	Key ValueStores	Column Family Databases	DocumentDatabases	Graph databases
Based on	Dynamic Hash Tables, Dynamo DB	Google's Bigtable	Lotus Notes, encoding includeJSON, XML	Euler's GraphTheory
Data Model	Key/Valuepairs	Columns	Key/ValueCollections	Graph structure-Nodes, Edges andProperties
Applicability	Handling massive load	Distributed filesystems	Web applications,full text searches and updates, information ranking	Semantic web, Social Networks, Intelligent Agencies
Advantages	Simple andeasy to implement	Fast querying of data,storage of very large quantities of data	Accepts partiallycomplete data, allows efficient querying	Easy scaling of complex data acrossdistributed systems.
Disadvantages	Inefficient inquerying/ updating part of a database	Very low-level API	No standard querylanguage	Traversal of entire graph to give correctresults
Examples	Redis,Project Voldermort	Cassandra, HBase	MongoDB, CouchDB	Neo4J, InfoGrid
Data Model	Key/Valuepairs	Columns	Key/ValueCollections	Graph structure- Nodes, Edges and Properties

FFigure 5: Summary of the four categories

ACID transactions provide 4 properties which must be guaranteed:

i. Atomicity**:** A database transaction is treated as a single unit such that all of the operations in the transactionwill complete, or none will. This property is referred to as "all or nothing" approach to execution. If one element of the transaction fails, the entire transactionis rolled back.

Consistency**:** This property ensures that there is no violation of integrity thus any transaction will transform the database state from one valid state to another**.** The transaction must adhere to rulespredefined in the system at every instance. If at one instance, a transaction that violates the rules is executed, the transaction is rolled back and the database is returned to the previous valid state. Thisproperty entails that there can never be any partially-completed transactions.The database will be in a consistent state when the transaction begins and ends. This property ensures that any transaction will bring the database from one valid state to another. In high availability environment

V. MODELS FOR STRUCTURING DATABASES

ACID transactions provide 4 properties which must be guaranteed:

i. Atomicity**:** A database transaction is treated as a single unit such that all of the operations in the transactionwill complete, or none will. This property is referred to as "all or nothing" approach to execution. If one element of the transaction fails, the entire transactionis rolled back.

ii.Consistency**:** This property ensures that there is no violation of integrity thus any transaction will transform the database state from one valid state to another**.** The transaction must adhere to rulespredefined in the system at every instance. If at one instance, a transaction that violates the rules is executed, the transaction is rolled back and the database is returned to the previous valid state. Thisproperty entails that there can never be any partially-completed transactions.The database will be in a consistent state when the transaction begins and ends. This property ensures that any transaction will bring the database from one valid state to another. In high availability environment this rule must be satisfied for all nodes in a cluster.

	Flat File Database	RDBMS	NoSQL
Data Model	Flat File	Tables	Columns, Graph, Document, Key/Value
Schema	Schema-less	Fixed Schema	Schema-less
Query Languages	CQL	SQL	API calls, JavaScript and REST
Integrity Model	None	ACID	CAP, BASE
Applicability	Any	Relational and transactional data	Non-relational data
Security	No security	Limited security mechanisms, vulnerable to SQL injection	Authorisation and authentication weaknesses, no encryption, Multiple interfaces increase attack surface.
Advantages	Simpler to use, Less expensive, suited for small scale use	Ensures data integrity between transactions, better security, supports medium to larger sized organisations, provides backup and recovery controls	Can cater for Big Data, unstructured data and distributed systems
Disadvantages	No support for multi-user access, redundancy and integrity problems	Expensive and difficult to manage in distributed systems, Complex and difficult to learn, not suitable for unstructured data	Security is a concern (no encryption), lack of standard query language, Too many varied databases thus no single solution for different purposes
Examples	MsDOS	Oracle, Postgres, MySQL, Microsoft SQL Server	MongoDB, Cassandra, Neo4J

IV.

Figure 6: Summary of flatfile database, RDBMS and NoSQL

usage, characterized by Big Data, large number of users and unstructured data in distributed environments which has called for NoSQL databases .

Isolation: Every transaction's execution is independent another and thus will behave as if it is the only operation being performed upon the database. Each transaction has to execute in a "black box" and thus should be transparent to any

other concurrent transaction. No transaction should ever see the intermediate product of another transaction until it is completed

Durability: After a transaction is committed, the effects thereof

are permanent. Any subsequent disturbances or system failure will not result in a change in the current database state.

At every given database operation, all the data undergoes checks to make sure they adhere to constraints imposed by ACID properties. This has worked well for over three decades in normalized, small data environments with less concurrent users in the relational database age. However with new trends in technology and burgeoning internet

V. CONCLUSIONS

The underlying features of the main database management systems namely the Flat File Database, RDBMS and NoSQL were reviewed. The main problems found on the Flat file and RDBMS that were common to both database systems include security vulnerabilities, scalability limitations, and availability of data regardless of network partition, timely propagation of changes to ensure consistency, performance bottlenecks and existence of a single point of failure. Owing to the rigid schema of the RDBMS, not all data structures can be represented and stored. These challenges manifest as a result of the architectural constraints inherent in the databases. It was observed that these DBMS have some aspects that are still desirable for instance to achieve reliability and integrity. Completely doing away with the traditional databases in favour of total adoption of the NoSQL also poses great challenges in our data management quest. NoSQL has challenges of not adequately catering for relational and transactional data. While giving cognisance to mission critical data, transactional data and a varied more cases where we seek to ensure reliability as a key aspect, NoSQL may not be ideal, calling for a revisit to the good old mature, tried and tested RDBMS. Owing to this scenario, both RDBMS and NoSQL are suited for different purposes and therefore cannot be absolute substitutes for each other.

REFERENCES

[1] Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin, "MongoDB vsOracle - database comparison", IEEE 2012

[2] Kris Zyp http://www.sitepen.com/blog/2010/05/11/ nosql-architecture/,
May 2010

[3] http://www.rackspace.com/blog/nosql-ecosystem/

[4] Ruxandra Burtica, Eleonora Maria Mocanu, Mugurel Ionuţ Andreica, Nicolae Ţăpuş, "Practical application and evaluation of no-SQL databases in Cloud Computing", IEEE 2012

[5] Jim Gray, "The Transaction Concept: Virtues and Limitations", Proceedings of Seventh International Conference on Very Large Databases, June 1981

[6] Vibneiro,http://ivoroshilin.com/2012/12/13/brewers-cap-theorem-explained-base-versus-acid/,December 2012

[7] Anders Karlsson, http://karlssonondatabases.blogspot.com/ August 2013

[8] http://datastax.com/docs/1.0/ddl/column_family

[9] http://www.infoq.com/news/2011/08/UnQL

[10] http://www.wikipedia.org/wiki/SPARQL

[11] W3C,http://www.w3.org/TR/rdf-sparql-query,March 2013

[12] Vibneiro,http://ivoroshilin.com/2012/12/13/brewers-cap-theoremexplained-base-versus-acid/,December 2012

[13] Dmitriy kalyada,http://blog.altoros.com/four-things-to-consider-when-choosing-a-db-for-your-interactive-application.html,June 11,2013

[14] Charles Roe,
http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/ March 2013

[15] Mike Chapple, http://databases.about.com/od/ other databases/a/Abandoning-Acid-In-Favor-Of-Base.htm, August 2013Sones GmbH http://en.wikipedia.org/wiki/Sones_GraphDB May 2011