TRADITIONAL RDBMS TO NOSQL DATABASE: NEW ERA OF DATABASES FOR BIG DATA

Salah A. Jowan[1*], Ramadan Faraj Swese[1] , Antisar Yousf Aldabrzi[1], Mahmood Saad Shertil[2]

[1]Computer Department, Faculty of Science, Al-Asmarya Islamic University, Zliten, Libya
[2]Computer Department, Faculty of Science, Elmergib University, Alkhoms, Libya
**Author for correspondence: salahjowan@gmail.com*

ABSTRACT

Digital world is growing very fast and become more complex in the volume (terabyte to petabyte), variety (structured and un-structured and hybrid) and velocity (high speed in growth) in nature. This refers to as 'Big Data' that is a global phenomenon. This is typically considered to be a data collection that has grown so large it can't be effectively managed or exploited using Relational Database Management Systems (RDBMS). The huge growth in the Internet market and the emerging of the new web technologies (web 2.0) and the cloud computing come with a new challenges, new applications and new concepts such as NoSQL databases which is recently become a very popular as an alternative to the relational databases specially in dealing with large data which is one of the most common features of web today, providing high availability and scalability to the distributed systems which need fast access time and can't tolerate any down time during failures and have been used heavily by the big enterprises and web companies such as Facebook, Amazon and Google. The aim of this paper is to address the concepts of NoSQL, the movement and needs behind it, and reviews the characteristics and classifications of NoSQL databases and provide a better understand of non-relational databases.

Keywords:RDBMS, NoSQL Databases, Big Data, Distributed Systems, Cloud Computing.

1. INTRODUCTION

The relational database or RDBMS has been the dominant model for database management since it was developed by Edgar Codd in 1970 [1].It is widely used in most of the application to store and retrieve data and it has been the main data storage solution for the Information Technology IT industry. However, The advent of the Internet and cloud/distributed computing has spawned a class of application (e.g. Web 2.0 applications)

where the relational databases technologies are proving to be inadequate. These new applications typically require data stores that are fast (i.e. can handle high transaction rates) and/or store large/huge quantities of data and/or can scale horizontally and/or allow easy evolving of the schemas. To overcome this problem, a new type of database called NoSQL has emerged to try to meet the challenges of this new class of application. 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, high scalability and reliability, very simple data model, very simple query language, high availability , using 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 [2,3].Therefore, The reason companies like Google, Amazon, eBay, Facebook, Yahoo and others use NoSQL as a primary datasource is not because they're chasing the latest-and-greatest technology, but rather they use NoSQL out of necessity. Their applications, use cases, and data needs have many times outgrown the legacy RDBMS model and require a different type of engine.

2. WHAT IS NOSQL?

The word "NoSQL" originated from a hashtag (#NoSQL) about a meeting where people can talk about ideas and the new types of emerging databases. NoSQL means "Not Only SQL", implying that when designing a software

solution or product, there is more than one storage mechanism that could be used based on the needs. NoSQL was never meant to knockout SQL or supplant it.

2.1 Types of NoSQL Databases

NoSQL systems store and retrieve data in different formats. There are different approaches for classification of NoSQL databases resulting in different categories and subcategories [4]. However, most generic and commonly used classification was proposed by Ben Scofield in his presentation at CodeMash conference (2010) which involved a brief comparison of different NoSQL database categories and relational databases [5]. They can be briefly described as follows.

2.1.1 Key-Value Stores

In this type of databases all the data is stored as a pair of key and value. This structure is also known as “hash table”, where data retrieval is usually performed by using key to access value in way that each possible key appears only once in the collection.

Examples include Berkeley DB, Oracle NoSQL, LevelDB, Dynamo, Memcached. Since key-value stores use primary-key access, they generally have great performance and can be easily scaled. A representation of key-value stores can be seen in Fig. 1.

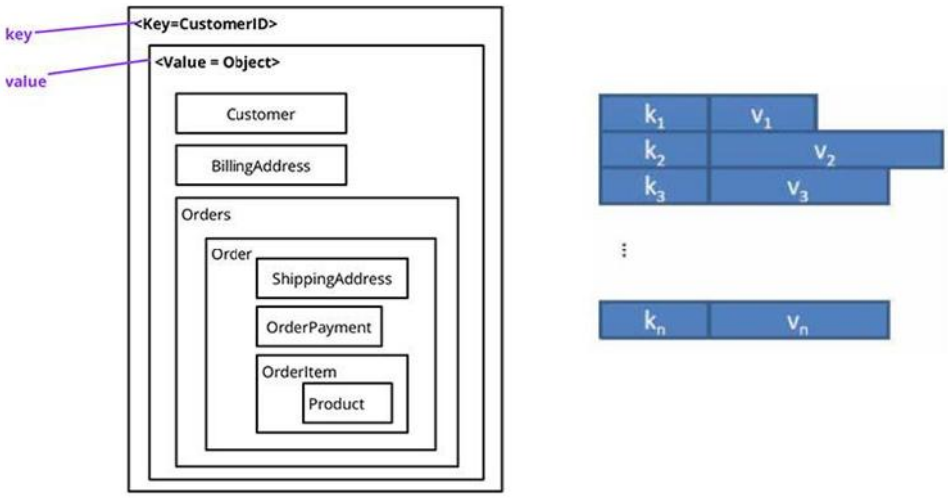

Figure 1: A representation of key-value stores

2.1.2 Column-family**Stores**

Similar to RDBMS, in this model all the data is stored as a set of rows and columns. Columns are grouped according to the relationship of data. When the data stored in some columns are often retrieved together, these columns are arranged in one group. Examples include Google Bigtable (2006), HBase, Hypertable, Cassandra. A representation of column stores can be seen in Fig. 2.

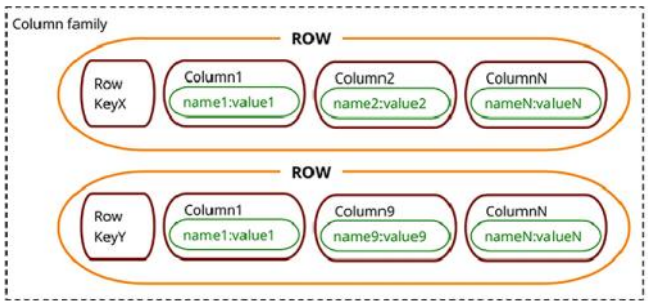

Figure 2: A representation of column stores

2.1.3 Graph databases

The best use of these databases is when stored information can be represented in the form of a graph with interlinked elements, for example, social networking, road maps or transport routes. Data is stored in graph structures with nodes, properties and lines. Nodes represent entities, properties are information about the entities and edges represent the relationship between the two.Examples include Neo4j, InfroGrid, IMS. An illustration of graph databases can be seen in Fig. 3.

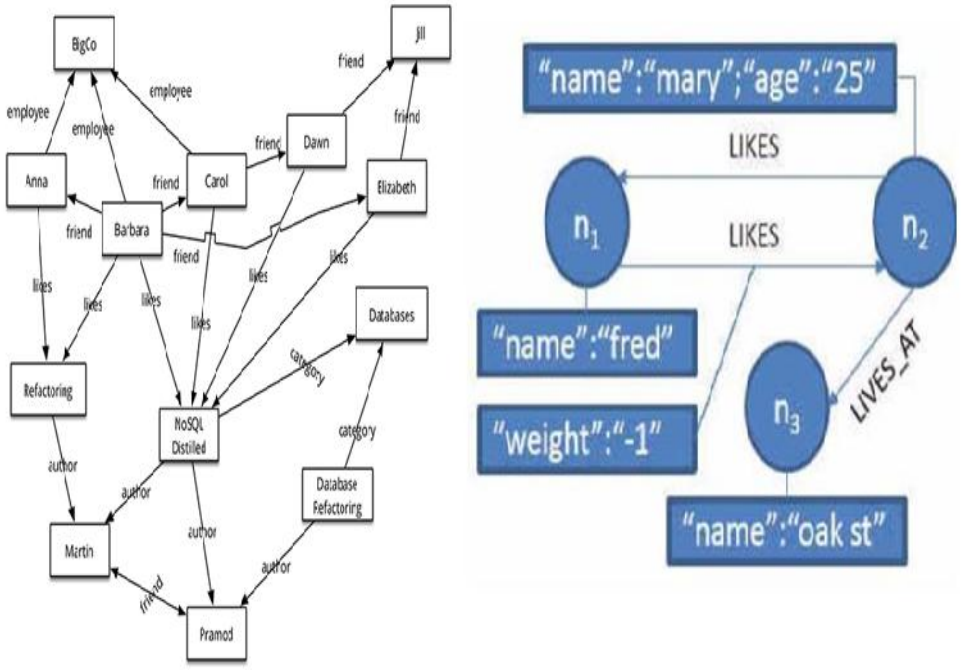

Figure 3: An illustration of graph databases

2.1.4 Document stores

Document Stores are designed around an abstract idea of a document which contain vast amount of data. Document stores accept documents in a variety of forms, and encapsulate them in a standardized internal format. Moreover, they provide support for lists, pointers and nested documents. Documents may be addressed in the database via a unique key that represents that document. This key is often a simple string, a URI, or a path. The requests can be expressed in terms of key or attribute if index exists. Examples include MongoDB, CouchDB, RaptorDB, Riak, IBM Lotus Notes. An illustration of document stores can be seen in Fig. 4; k1, k2,...kn are the keys and the corresponding information are their values.

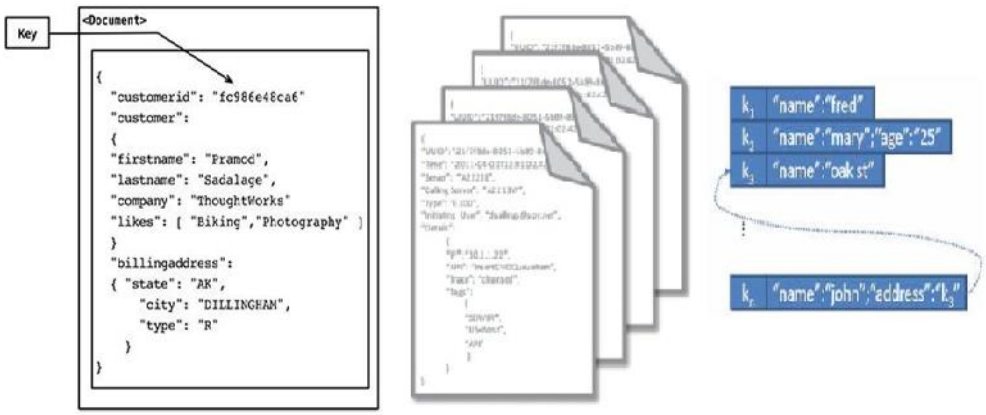

Figure 4: An illustration of document stores

Scofield's ideas were summarized in Popescu's blog [10] which can be seen in Table I.

TABLE I: CLASSIFICATION AND COMPARISON OF NOSQL DATABASES

	Performance	Scalability	Flexibility	Complexity	Functionality
Key-Value Stores	high	high	high	none	variable (none)
Column Stores	high	high	moderate	low	minimal
Document Stores	high	variable (high)	high	low	variable (low)
Graph Databases	variable	variable	high	high	graph theory
Relational Databases	variable	variable	low	moderate	relational algebra

2.2 Characteristics of NoSQL Databases

In order to guarantee the integrity of data, most of the classical database systems are based on transactions. This ensures consistency of data in all situations of data management. These transactional characteristics are also known as ACID (Atomicity, Consistency, Isolation, and Durability) [6]. However, scaling out of ACID-compliant systems has shown to be a problem. Conflicts are arising between the different aspects of high availability in distributed systems that are not fully solvable - known as the CAP- theorem [7]: **Consistency**: all clients see the same version of the data, even on updates to the dataset, **Availability**: all clients can always find at least one copy of the requested data, even if some of the machines in a cluster is down, **Partition-tolerance**: the total system keeps its characteristic even when being deployed on different servers, transparent to the client. The CAP-Theorem proposes that only two of the three different aspects of scaling out can be achieved fully at the same time (see Fig. 5).

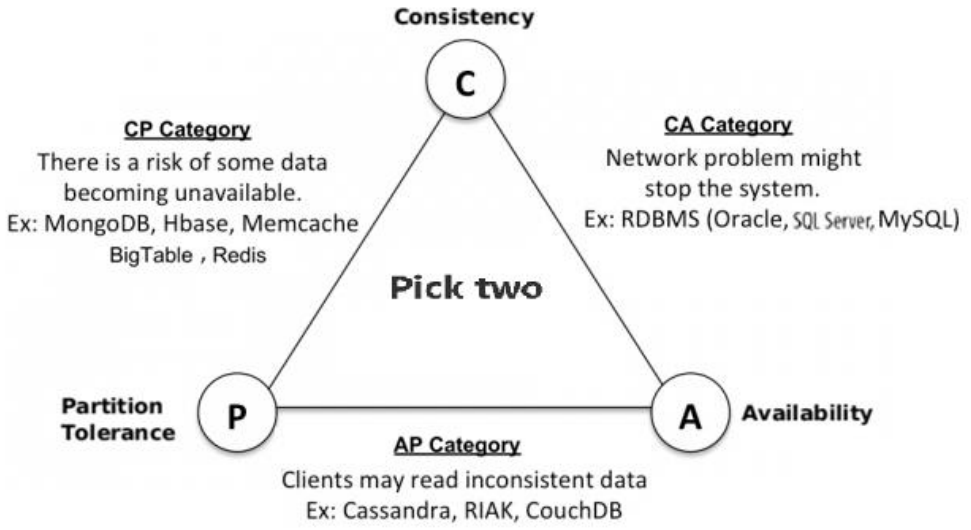

Figure 5: The CAP Theorem

Many of the NOSQL databases have loosened up the requirements on Consistency in order to achieve better Availability and Partitioning. This resulted in systems known as BASE (Basically Available, Soft-state, Eventually consistent) [8]. These have no transactions in the classical sense and introduce constraints on the data model to enable better partition schemes. Han, J., Haihong, E., Le, G., & Du, J. (2011) classifies NoSQL databases according to the CAP theorem [9].

- **Consistent, Available (CA) Systems** have trouble with partitions and typically deal with it with replication. Traditional RDBMS are normally classified as CA.
- **Consistent, Partition-Tolerant (CP) Systems** have trouble with availability while keeping data consistent across partitioned nodes. Examples of CP systems include:

- BigTable (column-oriented/tabular)
- HBase (column-oriented/tabular)
- MongoDB (document-oriented)
- Redis (key-value)
- MemcacheDB (key-value)
- Berkeley DB (key-value)

- **Available, Partition-Tolerant (AP) Systems** achieve "eventual consistency" through replication and verification. Examples of AP systems include:
  - Dynamo (key-value)
  - Tokyo Cabinet (key-value)
  - Cassandra (column-oriented/tabular)
  - CouchDB (document-oriented)
  - SimpleDB (document-oriented)
  - Riak (document-oriented)

3. WHAT'S CAUSING THE MOVE TO NOSQL?

Interactive applications have changed dramatically over the last 15 years. Big Users, the Internet, Big Data, and the Cloud are changing the way many applications are being developed. The number of concurrent users increasingly became accessible via the web (and later on mobile devices). The amount of data collected and processed soared as it became easier and

increasingly valuable to capture all kinds of data. The amount of unstructured or semi-structured data exploded and its use became integral to the value and richness of applications. Dealing with these issues was more and more difficult using relational database technology. The key reason is that relational databases are essentially architected to run a single machine and use a structured, schema-based approach to modeling data. Google, Amazon, Facebook, and LinkedIn were among the first companies to discover the serious limitations of relational database technology for supporting these new application requirements [15,19].

3.1 Big Users

Global Internet usage is growing rapidly, as is the amount of time each user spends online daily. With the invention of smart phones, people use their apps even more frequently(see Fig. 6). Increasing use of online apps means a rapidly growing number of database operations and the need for a much easier way to scale the database to meet these demands.

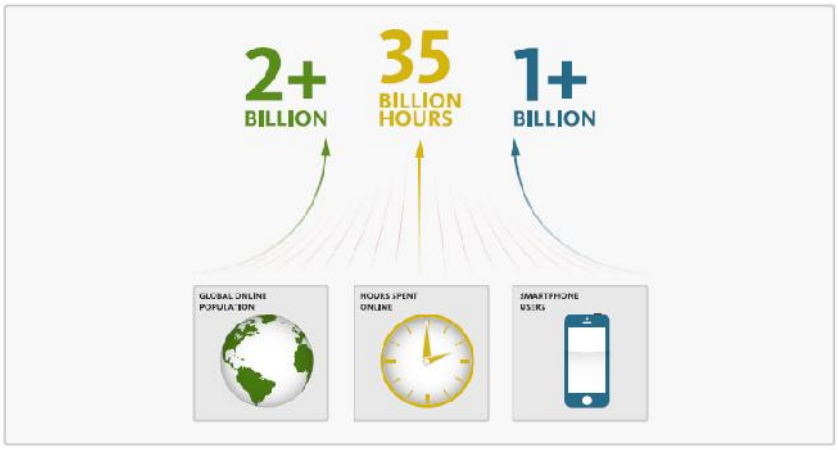

Figure 6: Large numbers of concurrent users

Not that long ago, 1,000 daily users of an application was a lot and 10,000 was an extreme case. Today, most new applications are hosted in the cloud and available over the Internet, where they must support global users 24 hours a day, 365 days a year. More than 2 billion people are connected to the Internet worldwide — and the amount of time they spend online each day is steadily growing — creating an explosion in the number of concurrent users. Today, it's not uncommon for apps to have millions of different users a day.

3.2 Structured and Unstructured Data

Structured data sets are those where the activity of processing and output is predetermined and highly organized. Payroll, Inventory control systems, point of sale systems, airline reservations are all forms of structured systems since they are using structured data- the data which is stored and displayed as a set of rows and tables. In contrast, unstructured data sets are the data that have little or no predetermined form or structured and are raw and unorganized. Ideally, all of this information would be converted into structured data. However, this would be costly and time consuming. Also, not all types of unstructured data can easily be converted into a structured model [11, 12]. A few examples of unstructured data include: Emails, Word processing files, PDF files, spreadsheets, digital Image files, video and audio files as well as social media posts. Seth Grimes reported that "80% of business-relevant information originates in unstructured form, primarily text." The following graph Fig. 7 shows the data representation of each type [11, 12]:

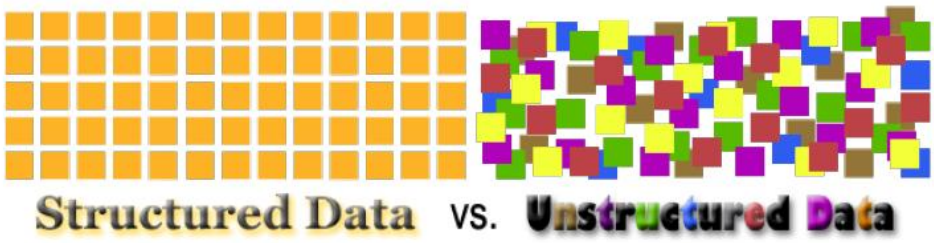

Figure 7: Structured vs. unstructured data

Relational databases are highly structured: all the data in the table are stored as rows and columns. Each column has a data type which is mostly normalized. There are always fixed number of columns although additional columns can be added later. Most of the tables are related to each other with primary and foreign keys thus providing "Referential Integrity" among the objects. The major vendors are ORACLE, SQL Server, MySQL, PostgreSQL, etc. [13, 14].

3.3 The Internet

There are billions of things connected to the Internet. They're in factories, farms, hospitals, andwarehouses. They're in homes: appliances, gaming consoles, and more. They'remobile phones and tablets. They receive environment, location, movement, temperature, weather data, and more from billions of sensors. In future more billions of things will be connected to the Internet and more data will be generated by embedded systems. The innovative enterprise is leveraging the Internet of Things to reduce time to market, reducecosts, increase efficiency, eliminate waste, and increase customer satisfaction. However, most of such data is small, semi-structured and continuous. It's a challenge for relationaldatabases that require a fixed

schema and structured data, and it's a challenge for distributed filesystems that require for discreet, unstructured data. To address this challenge, the innovativeenterprise is relying on NoSQL technology to scale concurrent data access to millions of connectedthings, store billions of data points and meet the performance requirements of mission-criticalinfrastructure and operations [9, 20].

3.4 Big Data

The amount of data is growing rapidly, and the nature of data is changing as well as developers find new datatypes – most of which are unstructured or semi-structured – that they want to incorporate into their applications (see Fig. 8).

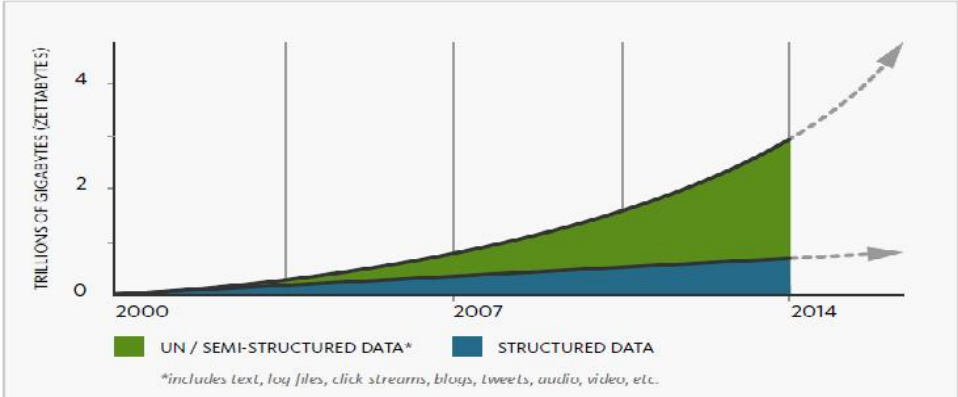

Figure 8: Big data

Data is becoming easier to capture and access through third parties such as Facebook, LinkedIn, and others. Personal user information, Geolocation data, Social graphs, user-generatedcontent, machine logging data, and sensor-generated data are just a few examples of the ever-expandingarray of

data being captured. It's not surprising that developers find increasing value inleveraging this data to enrich existing applications and create new ones made possible by it. Theuse of the data is rapidly changing the nature of communication, shopping, advertising, entertainment,and relationship management. Applications that don't find ways to leverage it quickly willquicklyfall behind.

The capture and use of the data creates the need for a very different type of database, however.Developers want a very flexible database that easily accommodates any new type of data theywant to work with and is not disrupted by content structure changes from third-party dataproviders. Much of the new data is unstructured and semi-structured, Therefore, developers also need adatabase that is capable of efficiently storing it. Unfortunately, the rigidly defined, schema-basedapproach used by relational databases makes it impossible to quickly incorporate new types ofdata, and is a poor fit for unstructured and semi-structured data.

3.5 The Cloud

Applications today are increasingly developed using a three-tier internet architecture, are cloud-based, and use a Software-as-a-Service business model that needs to support the collective needs of thousands of customers. This approach requires a horizontally scalable architecture that easily scales with the number of users and amount of data the application has (see Fig. 9).

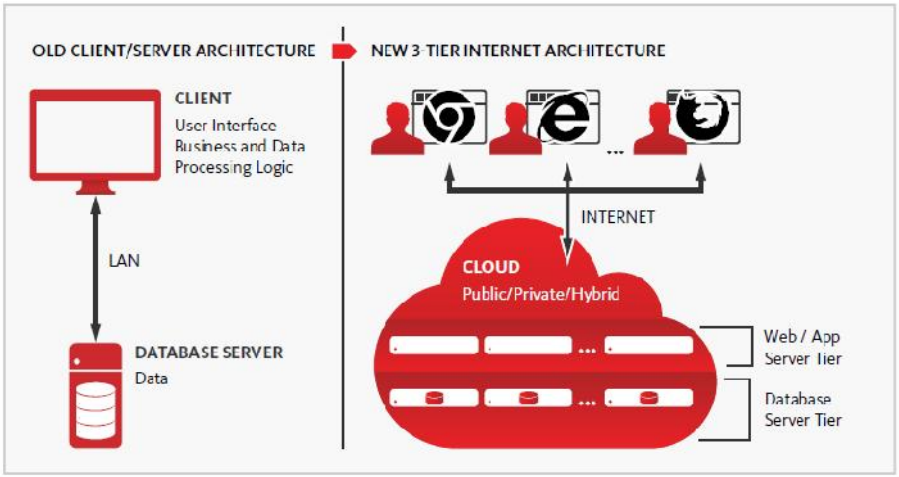

Figure 9: Three-Tier Internet Architecture

Not long ago, most consumer and many business applications were single-user applications that ran on a local PC. Most data-intensive, multi-user business applications used a two-tier, client-server architecture, run inside the firewall and supported a limited number of users. Today, most new applications (both consumer and business) use a three-tier internet architecture, run in a public or private cloud, and support large numbers of users [17, 20]. Along with this shift in software architecture, new business models like Software-as-a-Service (SaaS) and advertising-based models have become more common.At the database tier, relational databases use wasincreasingly problematic because they are a centralized, share-everything technologythat scales up rather than out. This made them a poor fit for applications that require easy anddynamic scalability. NoSQL technologies have been built from the ground up to be distributed,scale-out technologies and therefore fit better with the highly distributed nature of the three-tier internet architecture.

4. NOSQL FLEXIBLE DATA MODEL

Relational and NoSQL data models are very different. The relational model takes data and separates it into many interrelated tables (see Fig. 10). Each table contains rows and columns where a row might contain lots of information about a person and each column might contain a value for a specific attribute associated with that person, like their age. Tables reference each other through foreign keys that are stored in columns as well.

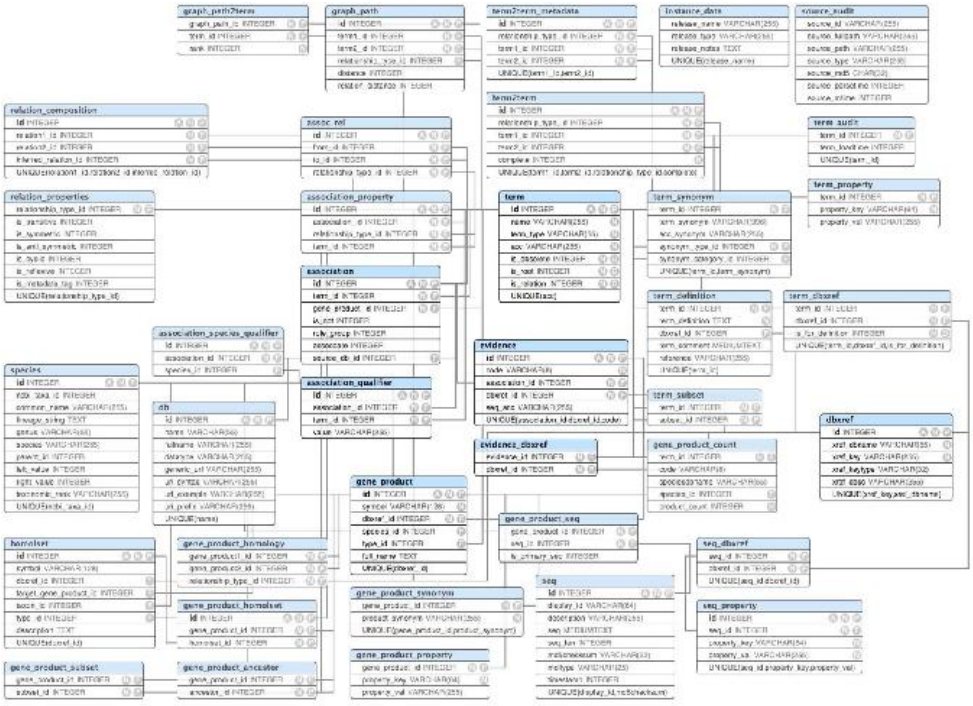

Figure 10: With relational databases, any operation requires the collection and processing of data from across tens or hundredsof interrelated tables, greatly hindering performance.

NoSQL databases have a very different model. For example, a document-oriented NoSQL database takes the data you want to store and aggregates it into documents using the JavaScript Object Notation JSON format. Each

JSON document can be thought of as an object to be used by your application. A JSON document might, for example, take all the data stored in a row that spans 20 tables of a relational database and aggregate it into a single document/object (see Fig. 11). Another major difference is that relational technologies have rigid schemas while NoSQL models are schemaless. Relational technology requires strict definition of a schema prior to storing any data into a database. Changing the schema once data is inserted is a big deal.

Figure 11: Unlike relational databases, which must store and retrieve data from scores of interrelated tables, document databases can store an entire object in a single JSON document, making it faster to retrieve.

5. NOSQL SCALABILITY AND PERFORMANCE ADVANTAGE

To deal with the increase in concurrent users (Big Users) and the amount of data (Big Data), applications and their underlying databases need to scale using one of two choices: scale up or scale out. Scaling up implies a centralized approach that relies on bigger and bigger servers. Scaling out

implies a distributed approach that leverages many standard, commodity physical or virtual servers.With relational databases, to support more users or store more data, you need a bigger server with more CPUs, more memory, and more disk storage.NoSQL databases were developed from the ground up to be distributed, scale out databases (see Fig. 12). They use a cluster of standard, physical or virtual servers to store data and support database operations. To scale, additional servers are joined to the cluster and the data and database operations are spread across the larger cluster. Since commodity servers are expected to fail from time to time, NoSQL databases are built to tolerate and recover from such failure making them highly elastic.

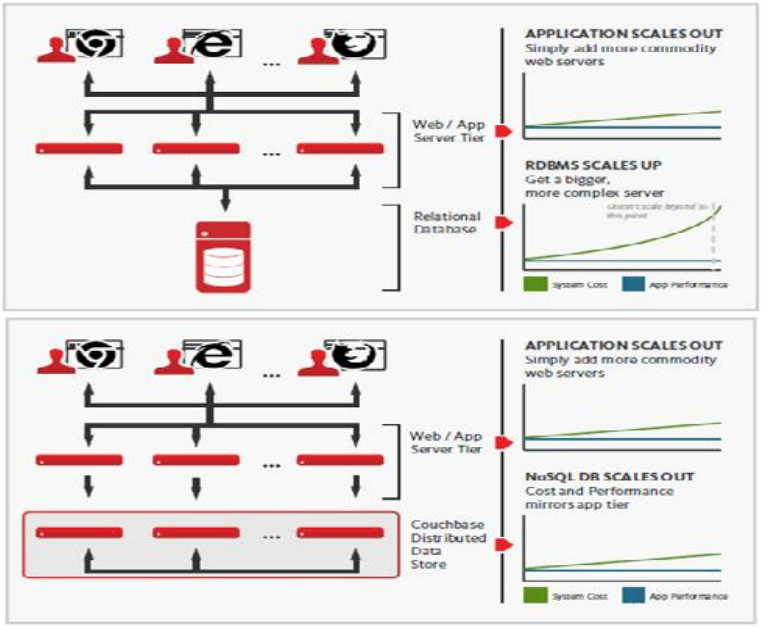

Figure 12. With relational databases, to support more users or store more data, you need a bigger server with more CPUs, more memory, and more disk storage.NoSQL databases provide a more linear, scalable approach to scaling than do relational databases.

NoSQL databases provide a much easier, linear approach to database scaling. If 10,000 new users start using your application, simply add another database server to your cluster. Add ten thousand more users and add another server. There's no need to modify the application as you scale since the application always sees a single (distributed) database.A NoSQL database automatically spreads data across servers, without requiring applications to participate (**Auto-sharding**) [17, 18]. Servers can be added or removed from the data layer without application downtime, with data (and I/O) automatically spread across the servers. Most NoSQL databases also support data replication, storing multiple copies of data across the cluster, and even across data centers, to ensure high availability and support disaster recovery.

6. CONCLUSION

RDBMS are facing major performance problems in processing exponential growth of users that applications must support, growth in the volume and variety of data being generated, stored and analyzed by modern users (user-generated data) and their applications(machine-generated data), and the rise of the cloud, which relies on a distributed three-tier internet architecture. Thus a new breed of databases, called NoSQL, has emerged to offer datamanagement capabilities that meet the needs of modern applicationthrough a more flexible data model and greater ability to scale dynamically to support more users and data. NoSQL databases are sometimes referred to as cloud databases, non-relational databases, distributed databases or Big Data databases. NoSQL is increasingly

considered a viable alternative to relational databases, and should be considered particularly for interactive web and mobile applications.

REFERENCES

[1] Codd, E. F. (1970) A relational model of data for large shared data banks, *Communications of the ACM*, 13, 6, 377-387..

[2] 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), Dec 2011.

[3] Jing Han, Haihong E, Guan Le, *"Survey on NoSQL Database"* , IEEE,2011.

[4] Shermin, M. (2013). An Access Control Model for NoSQL Databases.

[5] Scofield, Ben. (2010).NoSQL –Death to Relational Databases(?).–Presentation at the CodeMash conference in Sandusky (Ohio).

[6] ACID detail from web: http://en.wikipedia.org/wiki/ACID.

[7] Brewer's CAP Theorem, By Julian Browne on January 11, 2009 web: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

[8] Vanroose. Peter., Thillo V. Kris(2014). ACID or BASE? -the case of NoSQL. url:http://www.abis.be/resources/presentations/gsebedb220140612nosql.pdf.

[9] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on nosql database. In Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on (pp. 363-366). IEEE. [15] Tudorica, B. G., & Bucur, C.

[10] Popescu, Alex.(2010).Presentation: An Interesting NoSQL categorization. url: http://nosql.mypopescu.com/post/396337069/presentation-nosql-codemash-an-interestingnosql.

[11] Sherpa Software. "Structured and Unstructured Data: What is It?", http://www.sherpasoftware.com/blog/structured-and-unstructured-data-what-is-it/.

[12] S. Grimes, "Unstructured Data and the 80 Percent Rule", http://breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/.

[13] Relational Database Management System (RDBMS) vs noSQL. http://openproceedings.org/html/pages/2015_edbt.html.

[14] M. Ramachandran "Relational Vs Non-Relational databases", http://bigdata-madesimple.com/relational-vs-non-relational-databases.

[15] Padhy, R. P., Patra, M. R., and Satapathy, S. C. (2011) RDBMS to NoSQL: Reviewing some next-generation nonrelational database's, International Journal of Advanced Engineering Sciences and Technologies, 11, 1, 15 - 30.

[16] L. Arthur, "What is Big Data", htpp::://www.forbes.com/sites/liasaarthur/2013/08/15/what-is-big-data/.

[17] A. K. Zaki, "NoSQL Database: New Millennium Database for Big Data , Big Users, Cloud Computing and Its Security Challenges," http://esatjournals.org/Volumes/IJRET/2014V03/I15/IJRET20140315080.pdf.

[18] "Big Data: Volume, Velocity, Variability, Variety", http://nosql.mypopescu.com/post/6361838342/bigdata-volume-velocity-variability-variety.

[19] Find Source: www.couchbase.com/why-nosql/nosql- database.

[20] Pokorny, J. (2011, December). NoSQL Databases: a step to database scalability in Web environment. In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services (pp. 278-283). ACM.