IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

A Survey on NoSQL and its Terminology

Parvaneh Asghari[1], Houman Zarrabi[2]

Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran[1]

Iran ICT Research Center, Tehran, Iran[2]

Abstract: Nowadays the world needs databases to be able to store and process big data effectively and demand for very high-performance when reading and writing. These requirements effects especially in large scale and high concurrency applications such as search engines, hence the traditional database limits itself for such complex requirements. Therefore various types of non-relational databases that are commonly referred to as NoSQL databases which is abbreviation of "Not only Structured Query Language" was emerged. This paper presents a survey on NoSQL and its various data stores. NoSQL (Not Only SQL) is a database which is used to store large amount of data. NoSQL databases are distributed, non-relational, open-source and horizontally scalable. In this paper fundamental concepts like ACID, BASE and CAP theoremwill be described and present a comparison between ACID and BASE properties. Furthermore on the basis ofthe CAP theorem, various forms of data stores in NoSQL like Key/-Value data store, Column family data store, Document data store, Graph database and their characteristics are explained. In addition a taxonomy of NoSQL by various data stores and also the historical trend of popularity of themis presented. At the end, the reasons to consider a NoSQL solution for the Internet of Things data are depicted.

Keywords: NoSQL, ACID, BASE, CAP theorem, Data store, IoT.

I. INTRODUCTION

The problems with Relational database is lacked handling exponential growth of data. Many organizations collect vast amounts of customer's, scientific, sales, and other data for future analysis. Traditionally, most of these organizations have stored structured data in relational databases for subsequent access and analysis. However, a growing number of developers and users have begun turning to various types of non-relational databases, now frequently called NoSQL databases.
The term NoSQL stands for "Not Only SQL" and it is used for modern non-relational, distributed, open-source and horizontally scalable databases. Non-relational database does not organize its data in related tables unlike relational databases (i.e. data is stored in a non-normalized way). In NoSQL databases data is organized in a mixed model of structured, semi-structured and non-structured data. So, unlike relational databases, the primary advantage of NOSQL database is handling unstructured data such as documents, e-mail, multimedia's and social media's data effectively [1].
In order to store massive database a common strategy is to partition the data and store the partitions across different server nodes. Therefore distribution is one of the most important aspects of NoSQL databases. Subsequently the workload distributes across many servers, therefore multiple systems are easily added together in a linear way in order to Increase the throughput, so horizontal scalability is another positive aspect of NoSQL databases. Everyone can look into code freely, update it according to his needs and compile it, therefore the nature of being open-source makesNoSQL easy to use [2].
In this paper some important concepts about NoSQL such as ACID and BASE properties are explained. In the following the CAP theorem is described as a substantial matter which various data store types in NoSQL are defined and formed on it. Description of different types of data models such as key/-value stores, column family stores , document stores, graph databases and also a rapid comparison of NoSQL various data stores by some nonfunctional categories like performance, scalability, flexibility, complexity and functionality are presented. At the end of this paper the main reasons to consider a NoSQL Solution for Internet of Things data are pointed.

II. NOSQL TERMINOLOGY

In NoSQL there are some important concepts which are ACID free, BASE And CAP theorem.

ACID free**:** NoSQL does not support ACID properties due to consistency features of it. ACID stands for **A**tomicity, **C**oncurrency, **I**solation and **D**urability.

- **Atomicity:** Either the task (or all tasks) within a transaction are performed or none of them are. This is the all-or-none principle. If one element of a transaction fails the entire transaction fails.
- **Consistency:** The transaction must meet all protocols or rules defined by the system at all times. The transaction does not violate those protocols and the database must remain in a consistent state at the beginning and end of a transaction; there are never any half-completed transactions.
- **Isolation:** No transaction has access to any other transaction that is in an intermediate or unfinished state. Thus, each transaction is independent unto itself.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

This is required for both performance and consistency of transactions within a database.
- **Durability:** Once the transaction is complete, it will persist as complete and cannot be undone; it will survive system failure, power loss and other types of system breakdowns.

BASEstands for **B**asically **A**vailable, **S**oft state and **E**ventual consistency.
- **Basically Available:** This constraint states that the system does guarantee the availability of the data as regards CAP Theorem; there will be a response to any request. But, that response could still be 'failure' to obtain the requested data or the data may be in an inconsistent or changing state, much like waiting for a check to clear in your bank account.
- **Soft state:** The state of the system could change over time, so even during times without input there may be changes going on due to 'eventual consistency,' thus the state of the system is always 'soft.'
- **Eventual consistency:** The system will eventually become consistent once it stops receiving input. The data will propagate to everywhere it should sooner or later, but the system will continue to receive input and is not checking the consistency of every transaction before it moves onto the next one.

The BASE properties can be summarized in the following way: "an application works basically all the time (basically available), does not have to be consistent all the time (soft-state) but will be in some known-state state eventually" [3]. In TableI the contrasts of ACID and BASE are shown.

TABLE I ACID vs. BASE

ACID	BASE
Strong consistency	Weak consistency – stale data OK
Isolation	Availability first
Focus on "commit"	Best effort
Nested transactions	Approximate answers OK
Availability?	Aggressive (optimistic)
Conservative (pessimistic)	Simpler!
Difficult evolution (e. g. schema)	Faster
	Easier evolution

CAP-Theoremstands for**C**: **C**onsistency, **A**: **A**vailability, **P**: **P**artition tolerance [4]:
- **Consistency** meaning if and how a system is in a consistent state after the execution of an operation. A distributed system is typically considered to be consistent if after an update operation of some writer all readers see his updates in some shared data source.
- **Availability**and especially high availability meaning that a system is designed and implemented in a way that allows it to continue operation (i.e. allowing read and write operations) if e. g. nodes in a cluster crash or some hardware or software parts are down due to upgrades.
- **Partition Tolerance** understood as the ability of the system to continue operation in the presence of network partitions. These occur if two or more "islands" of network nodes arise which (temporarily or permanently) cannot connect to each other. Some people also understand partition tolerance as the ability of a system to cope with the dynamic addition and removal of nodes [9].

At most two of these three characteristics can be chosen in a "shared-data system" [5], as shown Fig. 1. If a system or parts of a system have to be consistent and partition-tolerant, ACID properties are required and if availability and partition-tolerance are favored over consistency, the resulting system can be characterized by the BASE properties. For example Amazon's Dynamo [6]is available and partition-tolerant but not strictly consistent, i.e. writes of one client are not seen immediately after being committed to all readers. Google's Bigtable chooses neither ACID nor BASE but the third CAP-alternative being a consistent and available system and consequently not able to fully operate in the presence of network partitions[7].

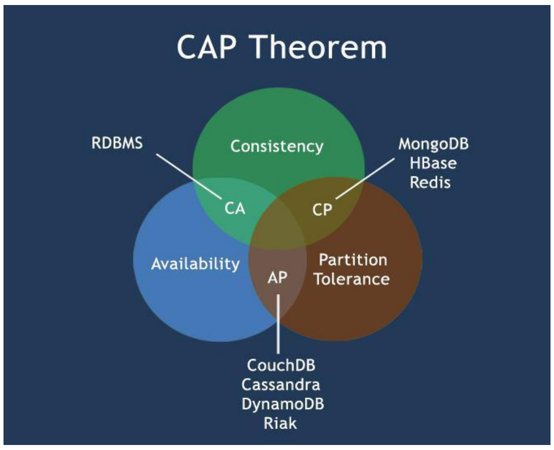


Fig. 1. CAP Theorem

III. NOSQL CHARACTERISTICS

In this section the most important characteristics of NoSQL are explained.

- NoSQL does not use the relational data model thus does not use SQL language.
- NoSQL stores large volume of data.
- In distributed environment (spread data to different machines), we use NoSQL without any inconsistency.
- If any faults or failures exist in any machine, then in this there will be no discontinuation of any work.
- NoSQL is open source database, i.e. its source code is available to everyone and is free to use it without any overheads.
- NoSQL allows data to store in any record that is it is not having any fixed schema.
- NoSQL does not use concept of ACID properties.
- NoSQL is horizontally scalable leading to high performance in a linear way.
- It is having more flexible structure [2].

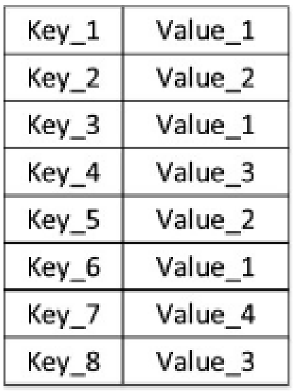
IV. NOSQL DATA STORE TYPES

On the basis of CAP theorem NoSQL databases are divided into number of data models. There are four different types of data store in NOSQL such as Key-value stores, Column family stores, Document stores, Graph databases [3, 7].

A. Key-value stores:

Key-value stores have a simple data model based on key-value pairs, which resembles an associative map or a dictionary [8]. The key uniquely identifies the value and is used to store and retrieve the value into and out of the data store. The value is opaque to the data store and can be used to store any arbitrary data, including an integer, a string, an array, or an object, providing a schema-free data model. Along with being schema-free, key-value stores are very efficient in storing distributed data, but are not suitable for scenarios requiring relations or structures. Any functionality requiring relations, structures, or both must be implemented in the client application interacting with the key-value store. Furthermore, because the values are opaque to them, these data stores cannot handle data-level querying and indexing and can perform queries only through keys. Key-value stores can be further classified as in-memory key-value stores which keep the data in memory, like Memcached [9] and Redis [10], and persistent key-value stores which maintain the data on disk, such as BerkeleyDB [11] Voldemort [12], and Riak[13]. For higher availability of data stores data objects are replicated.
In Key/-Value data stores there are two columns representing key and a value, as shown in Table II. Here key is unique and representing their values or attributes corresponding to it and data is represented in the form of ring and the partitioning of data is done on the basis of their alphabets (in sorted order) and data is also replicated in the form of ring. This is represented in next section.

TABLE II KEY/-VALUE STORE



Key_1	Value_1
Key_2	Value_2
Key_3	Value_1
Key_4	Value_3
Key_5	Value_2
Key_6	Value_1
Key_7	Value_4
Key_8	Value_3

The most important Characteristics of Key value databases are:

- Number of keys can have a dynamic set of attributes in the key value databases during storage of data.
- Data stored in the database is stored in the alphabetical order.
- All the activities can be performed on the data i.e. CRUD (Create, Read, and Update and Delete).
- All the relationships to the data are stored in the application code (not explicitly spread) [14].

The main points about Key/-Value (KV) databases are:

- It is one of the simple data model among all as it uses only key and a value.
- It handles huge data load.
- It scales to large volume of data.
- Replication of data is done using database in the form of ring. The replicated data is stored in the form of ring as well as in the alphabetical order. This is as shown in Fig. 2.

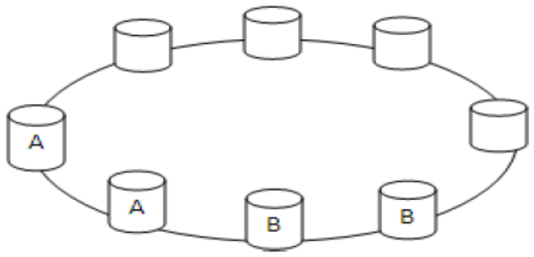


Fig. 2. Ring Partitioning and replication of data

B. Column-family stores:

Most column-family stores are derived from Google Bigtable [15], in which the data are stored in a column-oriented way. In Bigtable, the dataset consists of several rows, each of which is addressed by a unique row key, also known as a primary key. Each row is composed of a set of column families, and different rows can have different column families. Similarly to key-value stores, the row key resembles the key, and the set of column families resembles the value represented by the row key. However, each column family further acts as a key for the one or more columns that it holds, where each column consists of a name-value pair. Hadoop HBase [16] directly implements the Google Bigtable concepts, whereas Amazon SimpleDB [17] and DynamoDB [18] have a different data model than Bigtable. SimpleDB and DymanoDB contain only a set of column name-value pairs in each row, without having column families. Cassandra [19], on the other hand, provides the additional functionality of super-columns, which are formed by grouping various columns together.
In column-family stores, a column family in different rows can contain different columns. Occasionally, SimpleDB and DynamoDB are classified as key-value stores [20]; however we can consider them as column-family stores due to their table-like data model in which each row can have different columns. Typically, the data belonging to a row is stored together on the same server node. However, Cassandra offers to store a single row across multiple server nodes by using composite partition keys. In column-family stores, the configuration of column

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

families is typically performed during start-up. However, a prior definition of columns is not required, which offers huge flexibility in storing any data type.Columnar Databases are also known as column family databases because they are column-oriented databases, as shown in Fig. 3.

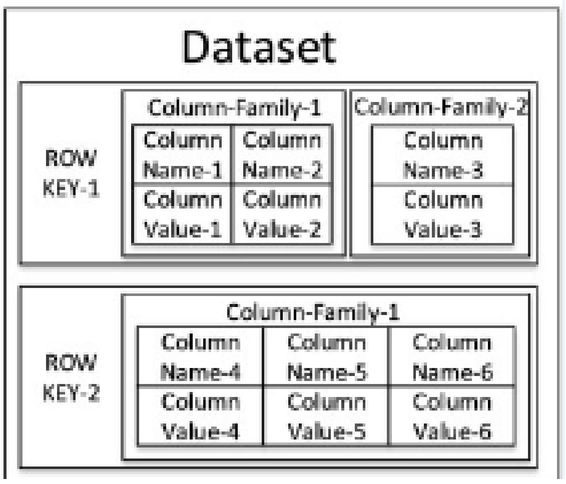


Fig. 3. Column family store

Important characteristics of columnar databases can be pointed as below:

- Columnar databases are faster than row based databases while querying.
- In columnar databases, assignment of storage unit is done to each and every column.
- In the columnar DBMS only the required columns are read, so reading is faster in this case.

In general, column-family stores provide more powerful indexing and querying than key-value stores because they are based on column families and columns in addition to row keys. Similarly to key-value stores, any logic requiring relations must be implemented in the client application [2, 3].

C. Document stores:
Document stores provide another derivative of the key/-value store data model by using keys to locate documents inside the data store. Most document stores represent documents using JSON (JavaScript Object Notation) or some format derived from it. For example, CouchDB and the Couchbase server [21] use the JSON format for data storage, whereas MongoDB [22] stores data in BSON (Binary JSON). Document stores are suitable for applications in which the input data can be represented in a document format. A document can contain complex data structures such as nested objects and does not require adherence to a fixed schema. MongoDB provides the additional functionality of grouping the documents together into collections. Therefore, inside each collection, a document should have a unique key.Unlike an RDBMS, where every row in a table follows the same schema, each document inside these document stores can have a different structure. Document stores provide the capability of indexing documents based on the primary key as well as on the contents of the documents. This indexing and querying capability based on document contents differentiates this data model from the key-value stores model, in which the values are opaque to the data store. On the other hand, document stores can store only data that can be represented as a document. Like key-value stores, they are inefficient in multiple-key transactions involving cross-document operations [4], as shown in Fig.4.

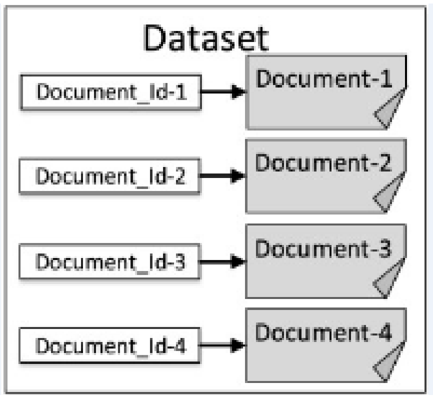


Fig. 4. Document Store

The main characteristics of Document Stores Database are:

- Documents are addressed in the database using key (unique) that represents that document.
- There are number of varieties to organize data that is collections, tags, non-visible metadata and directory hierarchies.
- In this we can use a key-value lookup to retrieve a document.

D. Graph databases:
Graph databases originated from graph theory and use graphs as their data model. A graph is a mathematical concept used to represent a set of objects, known as vertices or nodes, and the links (or edges) that interconnect these vertices. By using a completely different data model than key-value, column-family, and document stores, graph databases can efficiently store the relationships between different data nodes. In graph databases, the nodes and edges also have individual properties consisting of key-value pairs. Graph databases are specialized in handling highly interconnected data and therefore are very efficient in traversing relationships between different entities. They are suitable in scenarios such as social networking applications, pattern recognition, dependency analysis, recommendation systems and solving path finding problems raised in navigation systems [23,24].Some graph databases such as Neo4J [25] are fully ACID-compliant. However, they are not as efficient as other NoSQL data stores in scenarios other than handling graphs and relationships. Moreover, existing graph databases are not efficient at horizontal scaling because

when related nodes are stored on different servers, traversing multiple servers is not performance-efficient [3].Graph databases are based on the graph theory. In general, graph usually consists of \nodes, properties and edges [24], as shown in Fig. 5.

The main characteristics of Graph databases are:

- Graph traversals are executed with constant speed independent of total size of the graph.
- Graph databases are having high performance in context to their deep traversals.
- These are used for shortest path calculations.
- These are scalable. But its complexity increases.

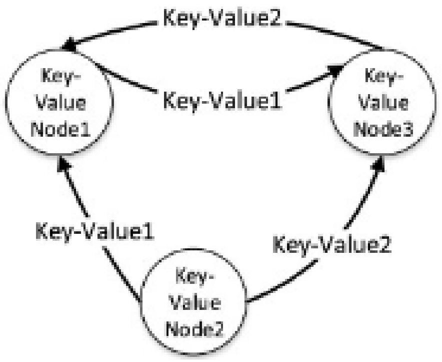


Fig. 6. Graph database

V. TAXONOMYOF NOSQL BY DATA STORES

A short comparison of classes of NoSQL databases by some nonfunctional categories like performance, scalability, flexibility, complexity and functionality is presented in Table 3 and 4.

TABLE 3 CLASSIFICATION – CATEGORIZATION and COMPARISON

	Performance	Scalability	Flexibility
Key-Value Stores	high	high	high
Column stores	high	high	moderate
Document stores	high	variable (high)	high
Graph databases	variable	variable	high
Relational databases	variable	variable	low

TABLE 4 CLASSIFICATION – CATEGORIZATION and COMPARISON

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

In the Fig. 6 the historical trend of various NoSQL databases popularityis shown.
In order to allow comparisons, the initial value is normalized to 100 [26].

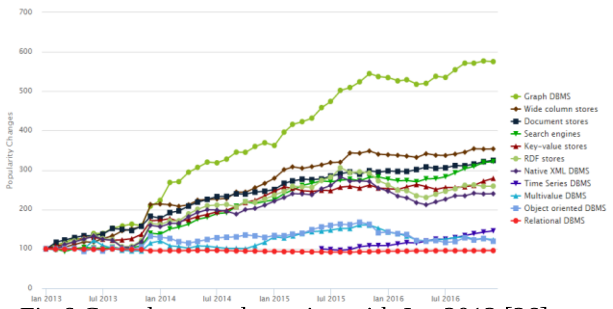


Fig.6.Complete trend, starting with Jan 2013 [26]

VI. NOSQL IN IOT

In general NoSQL databases do not enforce a strict schema and hence allow for highly flexible data modeling. The traditional relational database management systems will continue to have a role in the IoT which is stands for Internet of Things, when processing structured, highly uniform data sets, generated from a vast number of enterprise IT systems and where this data is managed in a relatively isolated manner. When it comes to managing more heterogeneous data generated by millions of sensors, devices and gateways, each with their own data structures and potentially becoming connected and integrated over the course of many years, databases will require new levels of flexibility, agility and scalability. In this environment, NoSQL databases are proving their value [27].
Data in the Internet of Things is different because it is almost by definition not completely known in advance. The market is moving so fast that its systems must be flexible, allowing the introduction of new sensors/devices and the data they emit. Data generated from an exponentially growing number of diverse sensors, devices, applications, and things will be accompanied by a growing diversity in the structure and scale of that data and more sources of additional data ranging from data sourced from corporate systems to crowd sourced data will need to be combined with this data[27].

There are several reasons that should be considered a NoSQL database in IoT [28]:

- Sensors can send huge amounts of data since they run 24/7. All of that data adds up to the need for a larger storage capacity. In this case relational databases were never really meant to deal with the kind of data that sensors generate. For one thing, sensor data does not always make sense in tabular format.
- SQL was originally designed for relatively static data structured as a table. Data from sensors can change a lot and provides a continuous stream. And the ability of adding and removing entries from anywhere is needed, which can prove difficult with relational databases.

- NoSQL databases are also more scalable, offering flexibility in data models. It is possible to have a structure similar to SQL with wide tables, or to choose to go with a document-oriented database, key-value database, or graph database. Time series databases are one of the more obvious choices for Internet of Things applications specifically.

Some businesses may join the big data revolution without knowing where they are actually going to store their data. It could been have a cluster dedicated to data and another to analytics, but that is expensive. It is preferred to have data and analytics in the same cluster. NoSQL eliminates budget waste for those with two different clusters that amount to the same thing [28].

VII. CONCLUSION

Traditional database architectures are proved to be inappropriate for many use cases because in current scenario speed and scalability are needed. Therefore nowadays applications are shifting towards In-Memory data storage which could boost the data access and the system could look forward for databases which could work according to the use cases and NoSQL is the solution for use cases.The main aim of this paper was to give an overview of NoSQL databases and its important concepts like ACID, BASE and CAP theorem. NoSQL doesn't follow ACID properties because of data consistency. On the basis of the CAP theorem, this paper presented different types of data stores, which are Key/-Value data store, Column family data store, Document data store and Graph database with comparison and taxonomy of them. In addition to all these,the historical trend of popularity of data stores was presented. At the end, the reasons to consider a NoSQL Solution for Internet of Things data wasdescribed.Further researchesare going on in the new technologies in Big Data and IoTwhich are arising the need for NoSQL that is polygon persistence.

REFERENCES

[1] BahaaldineAzarmi, Scalable big data architecture, Published by Apress, 2016.
[2] Min Chen, ShiwennMao,Yin Zhang, VicctorC.M.Leung, Big Data technologies, challenges and prospects, Published by Springer, 2014.
[3] ChristofStrauch, NoSQL Databases,Stuttgart Media University, 2014.
[4] Gray, Jonathan: CAP Theorem. August 2009. – Blog post of 2009-08-24.http://devblog.streamy.com/2009/08/24/cap-theorem/
[5] Burrows, Mike: The Chubby lock service for loosely-coupled distributed systems. In: Proceedings of the 7th symposium on Operating Systems Design and Implementation. Berkeley,CA, USA: USENIX Association, 2006 (OSDI '06), p. 335–350. – Also available online.http://labs.google.com/papers/chubby-osdi06.pdf
[6] DeCandia, Giuseppe; Hastorun, Deniz; Jampani, Madan; Kakulapati, Gunavardhan; Lakshman, Avinash; Pilchin, Alex; Sivasubramanian, Swaminathan; Vosshall, Peter; Vogels, Werner: Dynamo: Amazon's Highly Available Key-valueStore. September 2007. – http://s3.amazonaws.com/AllThingsDistributed/sosp/amazon-dynamo-sosp2007.pdf
[7] Katarina Grolinger,Wilson A Higashino, Abhinav Tiwari and Miriam AM Capretz, Data management in cloud environments: NoSQL and NewSQL data stores, ournal of Cloud Computing: Advances, Systems and ApplicationsAdvances, Systems and Applications2013
[8] Hecht R, Jablonski S: NoSQL evaluation: A use case oriented survey. Proc 2011 IntConf Cloud Serv Computing 2011, 336–341.
[9] Memcached http://memcached.org/ . Accessed 20Oct 2016
[10] Redis http://redis.io/ . Accessed 20 Oct 2016
[11] Hoff, Todd: And the winner is: MySQL or Memcached or Tokyo Tyrant?October 2009.–Blog post of 2009-10-28.http://highscalability.com/blog/2009/10/28/and-the-winner-is-mysql-or-memcached-ortokyo-tyrant.html
[12] DAMA - Philadelphia / Delaware Valley, the "Role of Data Architecture in NOSQL", Wednesday January 11th, 2012, http://www.damaphila.org/HaugheyNOSQL.pdf
[13] Klophaus R: Riak Core: building distributed applications without shared state. Proceedings of CUFP'10 - ACM SIGPLAN Commercial Users of Functional Programming. New York, NY, USA: ACM Press; 2010:1.
[14] An Oracle White Paper, "Oracle NoSQL Database", September 2011, http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf
[15] Chang F, Dean J, Ghemawat S, Hsieh W, Wallach D, Burrows M, Chandra T, Fikes A, Gruber R: Bigtable: A distributed structured data storage system. 7th OSDI 2006, 26: 305–314.
[16] Apache HBase http://hbase.apache.org/ . Accessed 20 Oct 2016
[17] Murty J: Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB. O'Reilly Media, Inc; 2008.
[18] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W: Dynamo: Amazon's highly available Key-value store. ACM SIGOPS Operating Syst Rev 2007, 41: 205.
[19] Lakshman A, Malik P: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Syst Rev 2010, 44(2):35–40.
[20] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W: Dynamo: Amazon's highly available Key-value store. ACM SIGOPS Operating Syst Rev 2007, 41: 205.
[21] Couchbase Server: The NoSQL document database. http://www.couchbase.com/couchbase-server/overview . . Accessed 20 Oct 2016
[22] MongoDB http://www.mongodb.org/ . Accessed 20 Oct 2016
[23] Lakshman A, Malik P: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Syst Rev 2010, 44(2):35–40.
[24] Buerli M: The current state of graph databases. 2012.http://www.cs.utexas.edu/~cannata/dbms/Class%20Notes/08%20Graph_Databases_Survey.pdf . Accessed 20 Oct 2016.
[25] Neo4j - What is a Graph Database? http://www.neo4j.org/. Accessed 20 Oct 2016.
[26] Complete trend, starting with January 2013. http://db-engines.com/en/ranking_categories. Accessed 20 Oct 2016.
[27] Why You Need NoSQL For The Internet Of Things. http://readwrite.com/2014/11/28/internet-of-things-nosql-data/, Accessed 31 Dec 2016.
[28] NoSQL and the Internet of Things. http://www.smartdatacollective.com/kingmesal/366893/nosql-and-internet-things, Accessed 31 Dec 2016.