Information Systems 63 (2017) 1–23

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Persisting big-data: The NoSQL landscape

Alejandro Corbellini*, Cristian Mateos, Alejandro Zunino, Daniela Godoy, Silvia Schiaffino

ISISTAN (CONICET-UNCPBA) Research Institute[1], UNICEN University, Campus Universitario, Tandil B7001BBO, Argentina

A R T I C L E I N F O

Article history:
Received 11 March 2014
Accepted 21 July 2016
Recommended by: G. Vossen
Available online 30 July 2016

Keywords:
NoSQL databases
Relational databases
Distributed systems
Database persistence
Database distribution
Big data

A B S T R A C T

The growing popularity of massively accessed Web applications that store and analyze large amounts of data, being Facebook, Twitter and Google Search some prominent examples of such applications, have posed new requirements that greatly challenge traditional RDBMS. In response to this reality, a new way of creating and manipulating data stores, known as NoSQL databases, has arisen. This paper reviews implementations of NoSQL databases in order to provide an understanding of current tools and their uses. First, NoSQL databases are compared with traditional RDBMS and important concepts are explained. Only databases allowing to persist data and distribute them along different computing nodes are within the scope of this review. Moreover, NoSQL databases are divided into different types: Key-Value, Wide-Column, Document-oriented and Graph-oriented. In each case, a comparison of available databases is carried out based on their most important features.

1. Introduction

Relational databases or RDBMSs (Relational Database Management Systems) have been used since the 1970s and, as such, they can certainly be considered a mature technology to store data and their relationships. However, storage problems in Web-oriented systems pushed the limits of relational databases, forcing researchers and companies to investigate non-traditional forms of storing user data [105]. Today's user data can scale to terabytes per day and they should be available to millions of users worldwide under low latency requirements.

The analysis and, in particular, the storage of that amount of information is challenging. In the context of a single-node system, increasing the storage capacity of any computational node means adding more RAM or more disk space under the constraints of the underlying hardware. Once a node reaches its storage limit, there is no alternative but to distribute the data among different nodes. Traditionally, RDBMSs systems were not designed to be easily distributed, and thus the complexity of adding new nodes to balance data is high [67]. In addition, database performance often decreases significantly since joins and transactions are costly in distributed environments [19,86]. All in all, this does not mean RDBMSs have became obsolete, but rather they have been designed with other requirements in mind and work well when extreme scalability is not required.

Precisely, NoSQL databases have arisen as storage alternatives, not based on relational models, to address the mentioned problems. The term "NoSQL" was coined by Carlo Strozzi in 1998 to refer to the open-source database called NoSQL not having an SQL interface [108]. In 2009, the term resurfaced thanks to Eric Evans in the context of

* Corresponding author. Fax: +54 249 4385681.
E-mail addresses: alejandro.corbellini@isistan.unicen.edu.ar (A. Corbellini), cristian.mateos@isistan.unicen.edu.ar (C. Mateos), alejandro.zunino@isistan.unicen.edu.ar (A. Zunino), daniela.godoy@isistan.unicen.edu.ar (D. Godoy), silvia.schiaffino@isistan.unicen.edu.ar (S. Schiaffino).

[1] Also Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).

http://dx.doi.org/10.1016/j.is.2016.07.009
0306-4379/© 2016 Elsevier Ltd. All rights reserved.

an event about distributed databases.[2] Since then, some researchers [58,57] have pointed out that new information management paradigms such as the Internet of Things would need radical changes in the way data is stored. In this context, traditional databases cannot cope with the generation of massive amounts of information by different devices, including GPS information, RFIDs, IP addresses, Unique Identifiers, data and metadata about the devices, sensor data and historical data.

In general, NoSQL databases are unstructured, i.e., they do not have a fixed schema and their usage interface is simple, allowing developers to start using them quickly. In addition, these databases generally avoid joins at the data storage level, as such operations are often expensive, leaving this task to each application. The developer must decide whether to perform joins at the application level or, alternatively, denormalize data. In the first case, the decision may involve gathering data from several physical nodes based on some criteria and then join the collected data. This approach requires more development effort but, in recent years, several frameworks such as MapReduce [31] or Pregel [74] have considerably eased this task by providing a programming model for distributed and parallel processing. In MapReduce, for example, the model prescribes two functions: a *map* function that process key-value pairs in the original dataset, producing new pairs, and a *reduce* function that merges the different results associated to each pair produced by the *map* function.

Instead, if denormalization is chosen, multiple data attributes can be replicated in different storage structures. For example, suppose a system to store user photos. To optimize those queries for photos belonging to users of a certain nationality, the Nationality field may be replicated in the User and Photo data structures. Naturally, this approach rises special considerations regarding updates of the Nationality field, since inconsistencies between the User and Photo data structures might occur.

Many NoSQL databases are designed to be distributed, which in turn allows increasing their capacity by means of just adding nodes to the infrastructure, a property also known as *horizontal scaling*. In NoSQL databases (as in most distributed database systems), a mechanism often used to achieve horizontal scaling is *sharding*, which involves splitting the data records into several independent partitions or *shards* using a given criterion, e.g. the record ID number. In other cases, the mechanism employed is *replication*, i.e. mirroring data records across several servers, which while not scaling well in terms of data storage capacity, allows increasing throughput and achieving high availability. Both *sharding* and *replication* are orthogonal concepts that can be combined in several ways to provide *horizontal scaling*.

In most implementations, the hardware requirements of individual nodes should not exceed those of a traditional personal computer, in order to reduce the costs of building such systems and also to ease the replacement of faulty nodes.

[2] NoSQL Meetup 2009, http://nosql.eventbrite.com/.

NoSQL databases can be divided into several categories according to the classification proposed in [113,19,67,49], each prescribing a certain *data layout* for the stored data:

- *Key-Value*: These databases allow storing arbitrary data under a key. They work similarly to a conventional hash table, but by distributing keys (and values) among a set of physical nodes.
- *Wide Column or Column Families*: Instead of saving data by row (as in relational databases), this type of databases store data by column. Thus, some rows may not contain part of the columns, offering flexibility in data definition and allowing to apply data compression algorithms per column. Furthermore, columns that are not often queried together can be distributed across different nodes.
- *Document-oriented*: A document is a series of fields with attributes, for example: name="John", lastname=" Smith" is a document with 2 fields. Most databases of this type store documents in semi-structured formats such as XML [16] (eXtensible Markup Language), JSON [28] (JavaScript Object Notation) or BSON [77] (Binary JSON). They work similarly to Key-Value databases, but in this case, the key is always a document's ID and the value is a document with a pre-defined, known type (e.g., JSON or XML) that allows queries on the document's fields.

Moreover, some authors also conceive Graph-oriented databases as a fourth category of NoSQL databases [49,113]:

- *Graph-oriented*: These databases aim to store data in a graph-like structure. Data is represented by arcs and vertices, each with its particular attributes. Most Graph-oriented databases enable efficient graph traversal, even when the vertices are on separate physical nodes. Moreover, this type of database has received a lot of attention lately because of its applicability to social data. This attention has brought accompanied new implementations to accommodate with the current market. However, some authors exclude Graph-oriented databases from NoSQL because they do not fully align with the relaxed model constraints normally found in NoSQL implementations [19,67]. In this work, we decided to include Graph-oriented databases because they are essentially non-relational databases and have many applications nowadays [2].

In the following sections we introduce and discuss the most prominent databases in each of the categories mentioned before. Table 1 lists the databases analyzed, grouped by category. Although there are many products with highly variable feature sets, most of the databases are immature compared to RDBMSs, so that a very thorough analysis should be done before choosing a NoSQL solution. Some factors that may guide the adoption of NoSQL as data storage systems are:

- *Data analysis*: In some situations it is necessary to extract knowledge from data stored in a database. Among the