# RDF Data Management Systems Based on NoSQL Databases: A Comparative Study

Mouad Banane[1], Abdessamad Belangour[2], El Houssine Labriji[3]

*Laboratory of Information Technology and Modeling LTIM*
*University Hassan II, Faculty of Sciences Ben M'sik, Casablanca, Morocco*

**Abstract**— *The growth of data transiting the Web has presented new challenges for RDF data management systems with respect to storage and the ability to effectively query these large quantities of RDF data. The limitations of traditional relational database systems and the development of NoSQL systems that are distributed databases that are scalable and tolerant fail. All these reasons motivated researchers to work on this topic to develop a system that can efficiently handle large RDF data based on NoSQL technology. This paper provides an overview on the voluminous RDF data management systems using NoSQL databases. It studies and evaluates these databases RDF (called triplestore). The comparison is based on some criteria of database software such as the NoSQL database, and model of database, index structure, database licence. This is a comparison of the systems proposed for efficient storage of massive RDF data.*

**Keywords** — *Semantic Web, RDF, NoSQL, Big Data, SPARQL.*

## I. INTRODUCTION

Recently, the web has experienced a quantitative explosion of digital data. This increase of information transiting on the Web complicates the modalities of search engines. In the face of this phenomenon, the members of the W3C/RDF have developed a syntax called Resource Description Framework (RDF) facilitating the description, the layout and the sharing of data constituting a Web page. In the other hand, the NoSQL technologies have emerged with their considerable performance in data management by providing the scalability and high availability prompting researchers to switch to NoSQL system dedicated to Big Data in order to manage this massive RDF data.

Resource Description Framework (RDF) developed by W3C is a graphical model for formally describing Web resources and their metadata to allow automatic processing of such descriptions. Rather, it is a data model for describing resources on the web. By resource we mean any entity that we want to describe on the web, but which is not necessarily accessible on the web. For example, one could provide information about the author of this document, even if the person described is not accessible on the web. Rather, there are resources, such as a personal page, or a photo, which can be obtained from their URL (Universal Resource Locator). These resources are related to this person, but are not that person. To designate this person, we will use a URI (Universal Resource Identifier), a unique name that syntactically resembles a URL, without the need for it to be accessible on the web. In a sense, all URLs are included in all URIs. The purpose of RDF is to allow resource information to be manipulated by applications, rather than just being displayed to web users. For this reason an XML syntax has been proposed to convey information modelled in RDF. Important features of RDF include flexibility and extensibility.

RDF is a data model in the form of a graph; it does not strictly speaking have syntax. Several syntaxes can be used to represent an RDF description. In an abstract way, the structure underlying any RDF expression is a collection of triplets, each consisting of a subject, a predicate, and an object. A set of such triplets forms an RDF graph. The subject of a triplet can be a URI or an empty node, that is to say a node that designates a resource without naming it. The predicate, which represents a property, is always a URI. Finally, the object represents the value of the property and can be a URI, an empty node, or a literal. An empty node is any node that is neither a URI nor a literal. It is a single node that can appear in several triplets, and has no intrinsic name. An empty node represents an anonymous resource. Using URIs to designate the resources described by an RDF graph has several advantages. They make it possible not to mix the designations used and to share the same vocabulary while avoiding the conflicts of names with several applications.

NoSQL databases have been designed to solve the problems of volume, multi-source and multi-format data processing in big data environments. NoSQL (Not Only SQL) databases denote a category of non-relational, horizontally scalable database management systems on machines with a standard configuration. Although this name is controversial and there are several variations of NoSQL systems, they have some or all of the following properties:

- Horizontal scalability of operations on several machines.

- Replication and partitioning of data on several machines.
- An interface or a simple call protocol (contrary to SQL).
- A weak competitive and transactional model unlike the ACID model of the majority of relational databases.
- Absence of the notion of schema which fixes the data structure, new attributes can be added dynamically to the existing structures. NoSQL systems have several differences. We find systems based on the distributed hash, distributed key-value systems like Google's BigTable [14]. Indeed, the pioneering NoSQL systems were a proof of concept validating new approaches for scalability, coherence, partitioning and fault tolerance. Using indexes in memory can be highly scalable and can distribute and replicate data across multiple nodes. There is no guarantee that data retrieved from a node is up to date, but the guarantee of propagating these updates to all nodes is ensured. On the other hand, Google has shown through BigTable that persistent storage can be scalable on thousands of nodes.

This document is organized as follows: after the first section introduction, the second section exposes some existing related works in this topic, the third section presents and exposes RDF data management systems based on the NoSQL. Section IV presents a review of RDF data management systems that use NoSQL database. Finally, section V concludes this work and suggests some future works.

## II. RELATED WORK

Recently, several researches have focused on large RDF data management. This research is based on NoSQL data management systems, after this opening of RDF data management systems on NoSQL technology. Cuder and Haque [9] compared five proposed implementations for storing massive NoSQL-based RDF data that are: Jena + Hbase, Hive + Hbase, Couchbase, 4Store, and CumulusRDF, in both distributed and single-machine deployment modes. . The results for a 16-node cluster show that: RDF data stores that use Hbase and Cassandra are not performing at the runtime level when the amount of RDF data is not large and also when running on a single machine or on a small number of nodes.

## III.DISTRIBUTED RDF DATA MANAGEMENT SYSTEMS

This section provides a detailed description of each RDF data storage management system.

### *A. AdaptRDF*

The authors of AdaptRDF[15] present a two-phase approach for designing an efficient tailored but flexible storage solution for resource description framework (RDF) data based on its query workload characteristics. AdaptRDF consists of two phases. The vertical partitioning phase which aims to reduce the number of join operations in the query evaluation process, and secondly the adjustment phase aims to maintain the efficiency of the query processing performance by adapting the sub-scheme. relate to the dynamics of queries.

### *B. Jena-HBase*

Jena-HBase[1] is a HBase backed triple store that can be used with the Jena framework along with a preliminary experimental evaluation of the prototype. This work focuses on the creation of a distributed RDF storage framework, thereby mitigating the scalability issue that exists with single-machine systems. HBase is an interesting choice only if you plan to handle a very large amount of data. Based on Hadoop, it distributes data using the Hadoop Distributed File System (HDFS) distributed file system. HBase database column-oriented. The data is organized according to row keys, then in columns-families, then in columns. The cell consists of a column name, value, and a timestamp. The value is simply stored as bytes, you do not need to define data types or pre-define columns, and only column families must be predefined.

### *C. 4store*

We know that the RDF data is stored in a triple format (Subject, predicate, object) but 4store[13] stores the RDF data in four elements of (model, subject, predicate, object), where the model is analogous to a SPARQL graph. In addition Uniform Resource Identifier (URI), literals, and empty nodes are all encoded using a cryptographic hash function. In this case, the system defines two types of compute nodes in this distributed environment: first storage nodes, which store the actual data, and second, the process nodes responsible for analyzing incoming requests and managing all distributed communications with the network storage. 4store uses the approach of partitioning data into non-overlapping segments and distributes quads based on hash partitioning of their subject.

### *D. Hive-HBase*

At the storage level, this approach[6] is based on the NoSQL HBase database as a data store and for queries it uses HiveQL as a query generator, it implements an intermediate translator prototype that will generate HiveQL language queries based on a SPARQL query then the HiveQL query will return the data to the distributed HBase storage system.

### *E. CumulusRDF*

CumulusRDF [11] is a key/value RDF data store that is nested distributed, it is used as an underlying storage component to a related data server that provides functionality for processing RDF data which are linked via HTTP Searches and also offers triple searches, for storage it offers two storage schemes. CumulusRDF implemented on Apache Cassandra which is is a distributed database that

allows storing a large amount of data thanks to its horizontal scalability. Cassandra takes the concepts of 2 existing databases. The first BigTable, created by Google, for its column-oriented data model and persistence mechanism on disk, and the second Dynamo, created by Amazon, for its distributed architecture without master node. Cassandra is very fast to handle a large volume of data. It allows to have flexible data schemas thanks to its representation in columns. Moreover, its architecture enables it to evolve without problems in a distributed environment; it integrates mechanisms of data replication and the possibility of clustering several Cassandra servers.

### *F. Rya*

Rya[3] is a scalable RDF data management system that uses a NoSQL database named Accumulo, based on Google's BigTable, Apache Accumulo is a structured and highly elastic storage written in Java. He is a father on Hadoop's Distributed File System (HDFS), which is part of the acclaimed Apache Hadoop project. Accumulo supports the recovery and storage of structured data, including interval queries, and supports the use of Accumulo tables as inputs / sorts for MapReduce processing. Apache Accumulo provides a robust and resilient data storage and retrieval system. Accumulo has some innovative features, such as cell-based access control and server-side programming mechanisms, that can modify key / value pairs at various locations in the data management process. In Rya the authors propose storage methods, and indexing schemes and use query processing techniques that can reach billions of triples on multiple nodes, more than that this approach provides quick and easy access to data via mechanisms classical query such as SPARQL.

### *G. H2RDF*

H2RDF[4] is a fully distributed RDF triplestore that combines two technologies the MapReduce processing framework with a NoSQL distributed data management system. The H2RDF system has two features that allow the efficient processing of SPARQL queries either simple or multi-join on an unlimited number of triple RDFs: join algorithms that perform joins according to the selectivity of SPARQL queries for the purpose of reducing the treatment; more than that and based on MapReduce it is an adaptive choice among centralized and distributed join execution for fast query responses.

### *H. Neo4J*

An RDF is a set of triples or instructions (subject, predicate, and object) where the subject and the predicate are resources and the object can be another resource or a literal. The only peculiarity of literals is that they cannot be the subject of other declarations. In a tree structure, we would call them leaf nodes. We also note that resources are uniquely identified by URIs. This approach consists of three rules:

Rule 1: All triples are mapped to Neo4j's nodes. A node in Neo4j representing an RDF resource will be tagged: Resource and will have a uri property with the URI of the resource.

*(S,P,O) => (:Resource {uri:S})*

Règle 2: Les prédicats de triplets sont mappés sur les propriétés de nœud dans Neo4j si l'objet du triplet est un littéral.

*(S,P,O) && isLiteral(O) => (:Resource {uri:S, P:O})*

we now come to rule 3: The *rdf: type* statements are mapped to categories in Neo4j.

(Something1, rdf:type, Category) => (:Category {uri:Something1})

### *I. SHARD*

SHARD[10], a massively scalable, robust and powerful triple-store based on Hadoop. SHARD offers a general approach to building an information system from the MapReduce software framework that responds to data queries. In this work the authors discussed the use of the MapReduce software framework to meet the challenge of building scalable and high performance distributed systems. They also proposed indexing evolutionarily, all on the basis of Hadoop.

### *J. J.-H. Um et al*

In this work [5], the authors propose a scalable distributed RDF store based on a distributed database that uses bulk loading for billions of triples to store data and respond quickly to user queries. In addition they give a mass loading algorithm using the MapReduce framework and the SPARQL query processing engine to connect to a large distributed database

### *K. Couchbase*

It is an RDF data management system based on the NoSQL Couchbase[12] database, at indexing level this approach uses the following triple patterns (? P?), (? O) and (? PO). And for querying data uses MapReduce views that are based primarily on the JSON format for Couchbase which is a column-oriented NoSQL database that relies on a Shared-Nothing architecture in which each node is independent - it contains all the services. Cluster management and data distribution on a cluster is managed from an application server that contains a cluster map that is kept up to date (Cluster Map). This architecture thus favors sizing, including another function, the support of the distribution on several data centers.

### *L. Aswamenakul et al[8]*

This Triple Store-based implementation uses the Jena TDB framework to store RDF data and an API that uses SPARQL to query Jena TDB framework

data. In this implementation which is based on a NoSQL database named MongoDB, the authors use RDF to JSON-LD Converter to convert the RDF data format to JSON-LD format, we know that the JSON format is designed for linked data, and in this work they used JSON-LD Parser to parse and import JSON-LD. For MongoDB, it uses JSON documents with schemas. MongoDB supports fields, interval queries, and regular expression searches. Fields in a MongoDB document can be indexed with primary and secondary indices. It remains expandable horizontally using sharding. MongoDB can be used as a file system with task balancing and data replication capabilities on multiple machines for file storage.

## IV. REVIEW OF NOSQL-BASED RDF SYSTEMS

This section reviews some of RDF data management systems that use NoSQL databases to store the RDF data including : Jena-HBase [1], Rainbow [2], Hive-HBase [6], 4store[13], CumulusRDF [11], Rya [3] , H2RDF [4], AdaptRDF [15], Couchbase[12], SHARD [10], Aswamenakul et al[8], Sun, J. et al[7], J.-H. Um et al[5]. The comparison is based on some criteria of database software such as :NoSQL database, NoSQL database model, database Licence, Index structure, Execution time ,Querying, Scalability, and support of SPARQL. The table 1 illustrates a review of RDF data management systems that use NoSQL database.

**Table1**: Review of RDF data management systems that use NoSQL database

| Name | NoSQL Database | Database Model | Indexation | Execution time | Scalability | Querying | SPARQL |
|---|---|---|---|---|---|---|---|
| **CumulusRDF[11]** | Cassandra | Key / Value Oriented | Medium | Medium | Yes | Sesame | Yes |
| **Neo4j** | Neo4j | Graph Oriented | Total | Low | No | SPARQL | Yes |
| **Jena+HBase[1]** | HBase | Column Oriented | Total | Medium | Yes | Jena | Yes |
| **Hive+HBase[6]** | HBase | Column Oriented | Medium | Medium | Yes | Hive | Yes |
| **H2RDF[4]** | HBase | Column Oriented | Medium | Medium | Yes | MapReduce | Yes |
| **Rainbow[2]** | HBase | Column Oriented | Medium | Medium | Yes | Sesame | Yes |
| **Rya[3]** | Accumulo | Column Oriented | Medium | Medium | Yes | SAIL API | Yes |
| **J.-H. Um et al[5]** | HDFS/Hbase | Column Oriented | Total | High | Yes | MapReduce | Yes |
| **Sun, J. et al[7]** | HBase | Column Oriented | Total | Medium | Yes | MapReduce | Yes |
| **Aswamenakul et al[8]** | MongoDB | Document Oriented | - | High | Yes | MongoDB queries | Yes |
| **Cudr et al[9]** | Cassandra /HBase /CouchDB | Column Oriented | Medium | Medium | Yes | Hive | Yes |
| **SHARD[10]** | HDFS | Split | Total | Medium | Yes | MapReduce | Yes |
| **4store[13]** | distributed RDF DBMS | Triple | Medium | Low | No | SPARQL | Yes |
| **Couchbase[12]** | Couchbase | Document Oriented | Total | Medium | Yes | MapReduce | Yes |
| **AdaptRDF[15]** | - | - | Medium | Medium | Yes | SPARQL | Yes |

We know that the standard technology that allows data access for the RDF dataset is the SPARQL recommendation of the W3C, most of the RDF storage systems presented in this work can accept SPARQL queries. Note also that SPARQL-based RDF data management systems rarely cause semantic discordance. Effective query processing is considered to be the most crucial challenge of these large RDF data management systems.

The simplicity and the dynamism of a column-oriented NoSQL databases like Accumulo HBase are the reasons why the majority of these RDF data management systems are based on column-oriented NoSQL systems.

## V. CONCLUSION

Resource Description Framework (RDF) is a standard for describing Web resources. In this paper,

we presented an overview of huge RDF data management systems based on Big Data's NoSQL technology. Many research projects have been devoted to the creation and development of distributed and scalable RDF data management systems. The NoSQL databases are systems that offer the scalability and high performance needed for efficient large data management. The paper revealed that it would be possible to further improve the efficient search process by developing web data management technologies. In our future work we propose a method to exploit a NoSQL database for the storage of massive RDF data and an algorithm for querying these data based on the HiveQL language of the Hive tool

## REFERENCES


[1] Khadilkar, V., Kantarcioglu, M., Thuraisingham, B., & Castagna, P. (n.d.). /home/vaibhav/Research/Jena-HBase/Results/LUBM/Query-Time-TDB-Comp-Q10.eps, (ii), 1–4

[2] Gu, R., Hu, W., & Huang, Y. (2015). Rainbow: A distributed and hierarchical RDF triple store with dynamic scalability. Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014, 561–566. http://doi.org/10.1109/BigData.2014.7004274

[3] Punnoose, R., Crainiceanu, A., & Rapp, D. (2012). Rya: a scalable RDF triple store for the clouds. Proceedings of the 1st International Workshop on Cloud Intelligence, 4. http://doi.org/10.1145/2347673.2347677

[4] Papailiou, N., Konstantinou, I., Tsoumakos, D., & Koziris, N. (2012). H RDF : Adaptive Query Processing on RDF Data in the. Proceedings of the 21st International Conference Companion on World Wide Web, 397–400. http://doi.org/10.1145/2187980.2188058

[5] Um, J. H., Lee, S., Kim, T. H., Jeong, C. H., Song, S. K., & Jung, H. (2016). Distributed RDF store for efficient searching billions of triples based on Hadoop. Journal of Supercomputing, 72(5), 1825–1840. http://doi.org/10.1007/s11227-016-1670-6

[6] Haque, A., & Perkins, L. (2012). Distributed RDF Triple Store Using HBase and Hive.

[7] Sun, J., & Jin, Q. (2010). Scalable RDF store based on HBase and MapReduce. ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings, 1, 633–636. http://doi.org/10.1109/ICACTE.2010.5578937

[8] Aswamenakul, C., & Buranarach, M. (n.d.). A Review and Design of Framework for Storing and Querying RDF Data using NoSQL Database, 1–4.

[9] Cudr, P., Haque, A., Harth, A., Keppmann, F. L., Miranker, D. P., Sequeda, J. F., & Wylot, M. (n.d.). NoSQL Databases for RDF : An Empirical Evaluation, 310–325.

[10] Rohloff, K., & Schantz, R. E. (n.d.). High-Performance , Massively Scalable Distributed Systems using the MapReduce Software Framework : The SHARD Triple-Store.

[11] Ladwig, G., & Harth, A. (2011). CumulusRDF: Linked data management on nested key-value stores. The 7th International Workshop on …, 30–42. Retrieved from http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/Workshops/SSWS/Ladwig-et-all-SSWS2011.pdf

[12] Zablocki, J. (n.d.). Couchbase essentials : harness the power of Couchbase to build flexible and scalable applications.

[13] Harris, S., Lamb, N., & Shadbolt, N. (2009). 4store: The design and implementation of a clustered RDF store. CEUR Workshop Proceedings, 517, 94–109.

[14] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., … Gruber, R. E. (n.d.). Bigtable: A Distributed Storage System for Structured Data. Retrieved from https://static.googleusercontent.com/media/research.google.com/fr//archive/bigtable-osdi06.pdf

[15] MahmoudiNasab, H., & Sakr, S. (2012). AdaptRDF: adaptive storage management for RDF databases. International Journal of Web Information Systems, 8(2), 234–250. http://doi.org/10.1108/17440081211241978