Journal of Engineering and Applied Sciences 13 (17): 7323-7328, 2018
ISSN: 1816-949X

An Approach for Big Data Interoperability

[1]Omar Hajoui, [2]Rachid Dehbi, [1]Mohamed Talea, [1]Zouhair Ibn Batouta and
[1]Abdellah Bakhouyi
[1]LTI Laboratory, Faculty of Science Ben M'sik
[2]LR2I Laboratory, Faculty of Science Ain Chock,
Hassan II University, Casablanca, Morocco

Abstract: NoSQL databases are increasingly used in big data and real-time web applications. But these databases are heterogeneous. They offer different data storage models, implementations and languages to developers and users. This wide variety of platforms makes it difficult data interoperability, data integration and even data migration from one system to another. This study proposes a literature review of study related to big data interoperability, the purpose of this study is to present and discuss the proposed solutions. Although, these solutions are ambitious, they neglect the semantic aspect of data that is considered a main axis in an effective solution to data interoperability. In this study, we present a new approach to data interoperability for NoSQL databases, in particular, it is addresses the data semantics problem.

Key words: Big data interoperability, NoSQL databases, polyglot, persistence, ontologies, OWL

INTRODUCTION

With the constant growth of stored and analyzed data, classic Relational Database Management Systems (RDBMS) exhibit a variety of limitations. Data querying loses efficiency due to the large volumes of data as well as storage and management of larger databases becomes challenging. NoSQL databases were developed to provide a set of new data management features while overcoming some limitations of currently used relational databases (Anonymous, 2016a, b). NoSQL databases are not relational and they don't require a model or structure for data storage which facilitates the storage and data search. Also, they allow horizontal scalability, it gives administrators the ability of increasing the number of server machines to minimize overall system load. The new nodes are integrated and operated in an automatic manner by the system. Horizontal scalability reduces the response time of queries with a low cost.

NoSQL databases can be used alone or as a complement to a relational database, they increase performance while bringing different advantages and new features. Currently, there are over 50 NoSQL databases with various features and optimizations. Each NoSQL database provides different mechanisms to store and retrieve data which directly affects performance. Each non-relational database has different optimizations, resulting in different data loading time and execution times for reads or updates (Anonymous, 2016a, b).

However, as the various systems have different performances and different behavior in terms of consistency and availability, it is meaningful to have applications that are not satisfied with just one of them. Also, let us assume that quantitative application use several NoSQL databases, various components of this application may use different systems. This case corresponds to what is popularly referred to as the polyglot persistence (Martin, 2011).

Unfortunately, writing the code of an application that uses multiple databases is not easy and sometimes very constraining for the developer. For example, in the case of relational databases, just use the JDBC API to access a MySQL database, Oracle and others. For cons, the lack of a standard for accessing NoSQL databases is a great problem as each NoSQL database has its own data model, query language and APIs. This kind of problems related to the heterogeneity of NoSQL databases has been evoked for the first time by Stonebraker (2011).

The problem of big data interoperability is produced when we want to exploit and analyze data stored in heterogeneous systems. Storing data in different representations makes data querying and interpretation difficult. For example, this problem arises if you want to make statistics on the data stored in several institutions of the same organization, several subsidiaries of the same company, several commercial sites of the same mall or intelligent sensors in the same city.

Corresponding Author: Omar Hajoui, LTI Laboratory, Faculty of Science Ben M'sik, Casablanca, Morocco

MATERIALS AND METHODS

Big data interoperability barriers: NoSQL databases have different data models, query languages and APIs, thus, making data interoperation a difficult task. Data model heterogeneity is caused by the use of different models or structural differences. Semantic heterogeneity is caused by different meanings or interpretations of data in various contexts.

Data model heterogeneity: The issue here is how to map data structures and operations from one DBMS into data structures and operations conforming to a different DBMS. A data model mapping step is assumed as a pre-requisite to data interoperability (Parent and Spaccapietra, 2000) and is dealt with as a separate problem. The needed mappings are those between any local data model and the common data model.

In key-value store, data is simply represented by a key/value pair, both the key and the value can be of any structure, their model can be likened to a distributed hash table. These systems have a very high performance in reading and writing but they offer simple HTTP query interface.

Document stores extend the key-value paradigm but the value is a document, the document tends to semi-structured data (XML, JSON), documents can be very heterogeneous in the database. They offer complex HTTP query interface.

In column-based store, data is stored as sections of columns of data rather than as rows of data. Data stored together with meta-data [typically including row id, attribute name and value, timestamp]. Read and write is done using columns rather than rows. The pioneer of this category is Google BigTable (Chang *et al*., 2008).

To solve the problem of data model heterogeneity in NoSQL databases context an approach leverages on the genericity of the data model to allow for a standard development practice that is not bound to a specific DBMS API but to a generic one (Bugiotti *et al*., 2013); (Cabibbo, 2013). Also, the researchers define a translation process, from the generic model constructs, to coherent system-specific structures and vice versa.

Semantic heterogeneity: Semantic heterogeneity is caused by different meanings or interpretations of data in various contexts (Fig. 1). The use of different terms and conceptual notions of a domain still poses significant challenges. The proposed solution is to create formal knowledge bases or ontologies to describe a domain and use these to create semantic agreements and resolve semantic differences.

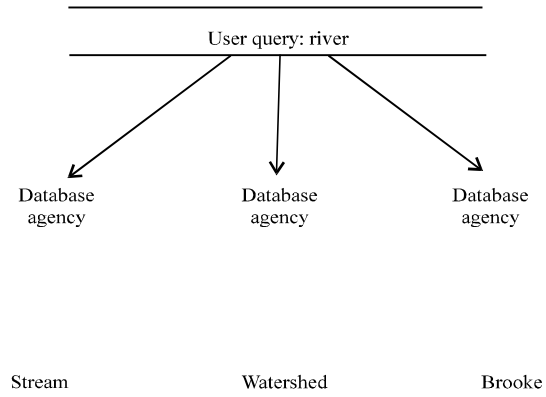

Fig. 1: Semantic interoperability example

Such knowledge bases hold terms that are generally accepted in a community. These terms can potentially be used in data sets created in the future and also be used to resolve existing heterogeneity in legacy data through mappings from ontology terms to local terms (Wiegand, 2011, 2013).

Prior solutions to avoid or resolve semantic differences were to create and enforce standards or resolve differences on an application by application basis. Although, standards can be a good solution, they may be difficult to create and if very general terms are used as standard terms, they might not fully cover the nuances and needs of local data. In any case, legacy data may not conform to the standards. And as to "one-of" solutions on an application by application basis, after enough unique "one-of" solutions are made, the need is recognized for a more universal solution. Creating and then consulting a comprehensive knowledge base for a domain, i.e., an ontology, results in a re-usable solution across many applications.

Ontologies provide a promised technology to solve the semantic heterogeneity problem because they allow to explicitly representing common semantics of a domain of discourse (Yahia *et al*., 2012). An ontology can be defined as a formal, explicit specification of a shared conceptualization (Gruber, 1993). "Conceptualization" refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. "Explicit" means that the type of concepts used and the constraints on their use are explicitly defined. "Formal" refers to the fact that the ontology should be machine readable. "Shared" reflects that ontology should capture consensual knowledge accepted by the communities. An ontology formally defines different concepts of a domain and relationships between these concepts, Fig. 2 shows an example for a small ontology.

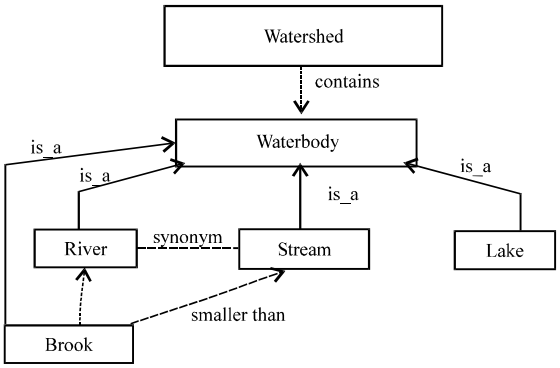

Fig. 2: Possible local hydrology ontology

In ontology-based approaches for information integration, local ontologies are used to describe the semantics of local information sources. The more recent Web Ontology Language (OWL) (Hitzler *et al*., 2009) has become a popular standard for data representation and exchange. The OWL supports the representation of domain knowledge using classes, properties and instances for the use in a distributed environment as the World Wide Web.

Query languages and APIs: In the world of relational databases, the SQL language is a standard for querying any database, it allows for complex operations such as joins aggregations and groupings, it's easy to understand, this facilitates the task of developers. On the other side, each NoSQL database has its own query language, some are poor, they implement just the CRUD operations, especially, those related to key/value store. Others are rich, particularly those related to document store and column-based store but they don't rise to the power of SQL, we can't do joins or run certain aggregate functions. Nonetheless, they implement the framework MapReduce which is a very powerful tool for making distributed calculations or statistics but the programming of MapReduce jobs needs experienced developers.

Similarly, there isn't a standard API like JDBC which allows applications to access NoSQL databases. Each system has its own API which makes access to multiple data sources quite complicated, the developer must study and test each of these APIs.

RESULTS AND DISCUSSION

The different approaches for big data interoperability: Currently, there are several works that address the big data interoperability. In this chapter, we present only implemented solutions.

ONDM; An Object NoSQL Datastore Mapper: ONDM (Cabibbo, 2013) is a framework to facilitate the storage and retrieval of persistent objects in NoSQL datastore systems. ONDM aims at supporting several challenges posed to application developers by the heterogeneity of NoSQL databases. It provides developers with a uniform application programming interface, transparent access to different NoSQL data stores and the ability to select from different data representation techniques for entity objects and relationships between objects. The highlights of ONDM are as follows: ONDM offers to application developers an ORM-like API, a variant of the popular Java Persistence API (JPA). By adopting this standard, it is easy to move existing JPA applications into the NoSQL realm and for developers it is simple to start writing NoSQL-based applications. As JPA, the ONDM data model is based on entities, relationships and embeddable objects (i.e., complex values). ONDM currently supports the access to a handful of NoSQL datastores (Apache Cassandra, Couchbase, MongoDB, Oracle NoSQL and Redis), belonging to different datastore categories. More important, it has been designed to be easily extensible. Indeed, the effort needed to implement in ONDM the access to another data store is rather limited.

ODBAPI; A unified rest API for relational and NoSQL data stores: In this study, Sellami *et al.* (2014) proposed a generic resources model to represent the different elements of heterogeneous data stores in a cloud environment. They also proposed a unique rest API that enables the management of the described resources in a uniform manner. This API is called ODBAPI and allows the execution of CRUD operations on relational and NoSQL data stores. To do so, they defined a resource model representing the different resources that they target within their API. In this version, they took into account only three data stores. These latter are a relational DBMS, a key/value data store that is Riak and a document data store which is CouchDB. In addition, they gave an overview of ODBAPI and its different operations. This API is designed to provide utmost control for the developer against heterogeneous data stores. ODBAPI eases the interaction with data stores at the same time by replacing an abundance of APIs. Moreover, it decouples cloud applications from data stores alleviating, therefore, their migration. However, the researchers do not deny that it still remains the problem of executing complex queries (e.g., group by, like, join, etc.). They did not take into account other categories of data stores such as graph data stores and column data stores that they intend to include in their future work. In addition, they did not ensure data transactions with REST architecture.

Spring data framework: The spring data framework offers some generic abstractions to handle different types of NoSQL DBMSs. These abstractions are refined for each DBMS. It provides a consistent programming model for interacting with NoSQL databases, using patterns and models from the spring framework. As a result, we get a consistent way of interacting with different NoSQL databases and we are able to leverage each one's individual strengths. However, the addition of a new data store is not so easy and the solution is strongly linked to the Java programming model.

A federated approach: The objective of this research (Dharmasiri and Goonetillake, 2013) is to address the heterogeneity of NoSQL data stores through database federation which has been around since the early 80's which has proven to be successful in integrating heterogeneous data storages. The researchers have successfully implemented a NoSQL federation with Cassandra, MongoDB and CouchDB proving that NoSQL federation is feasible with a certain degree of overhead.

A federated database management system can be considered as a virtual DBMS. With the help of data abstraction it enables users and applications to store and retrieve data from several non-contiguous databases with only a single query from its uniform interface.

With the tests and evaluation it can be concluded that NoSQL federation is feasible and it will take off a lot of time and effort taken in large scale data migrations. But you have to choose carefully on which component data stores to use as these choices have a great impact on the end performance which is quite essential in NoSQL systems.

DBMS+: In this study (Lim *et al*., 2013), the researchers make the case for developing a new breed of Database Management Systems that they term DBMS+. A DBMS+ contains multiple NoSQL systems internally. An application specifies its execution requirements on aspects like performance, availability, consistency, change and cost to the DBMS+declaratively.

For all requests (e.g., queries) made by the application, the DBMS+ will select the execution plan that meets the application's requirements best. A unique aspect of the execution plan in a DBMS+ is that the plan includes the selection of one or more NoSQL systems. The plan is then deployed and managed automatically on the selected system(s). If application requirements change beyond what was planned for originally by the DBMS+, then the application can be optimized and redeployed; usually with no additional effort required from the application developer.

A new approach to data interoperability for NOSQL databases: As stated in the second chapter, the first obstacle is the diversity of data storage models. NoSQL databases can be classified into four models (Key-value stores, document stores, column-based store and graph-based). The second problem is data semantics, for example, in two different databases the same information may have different names. This requires a domain ontology mapping. Our approach to allow data interoperability in for NoSQL databases is described by the model shown in Fig. 3. This model is original, thereafter, we will test and improve it. Our model can be divided into three main layers.

Meta layer: The processed data come from several data sources, each one has its own data storage model. The purpose of this layer is to overcome this problem, for the moment in our model, we adopt a metamodel based approach. This approach leverages on the genericity of the data model to allow for a standard development practice that is not bound to a specific DBMS API but to a generic one. The meta layer is composed of a set of Java API (Application Programming Interface). The role of these APIs is to translate the data from the original model to a generic model.

Semantic layer: It is a business representation of corporate data that helps end users access data using common business terms. The aim is to insulate users from the technical details of the data store and allow them to create queries in terms that are familiar and meaningful.

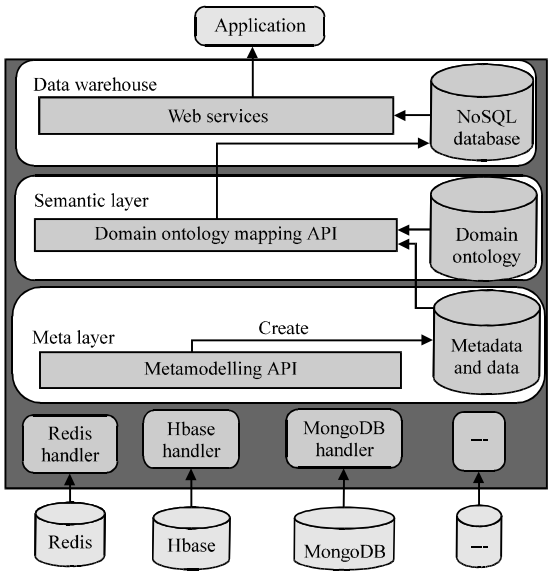

Fig. 3: Model to data interoperability for NoSQL databases

The semantic layer is configured by a person who has knowledge of both the data store and the reporting needs of the business. The person creating the semantic layer chooses to expose appropriate fields in the data store as "Business fields" and to hide any fields that are not relevant. Each business field is given a friendly, meaningful name and the business fields are organized in a way that will make sense to business users. Semantic layer delivers a number of benefits.

A semantic layer removes the dependency between queries and the data store. A change to the data store can often be handled globally, without the need to modify individual reports. So, we don't need to modify the SQL within hundreds of reports to accommodate a database design change.

Knowledge of the mappings between business fields and database fields resides in the semantic layer, not in the heads of developers. This makes it much easier for developers to understand and maintain reports that have been created by others.

An ad hoc query environment that uses a semantic layer allows developers and analysts to work interactively with business users to prototype the reports that are needed. This can be immensely valuable. The presence of a semantic layer simplifies the creation of queries.

Data Warehouse (DW): It is a set of data which is stored in a structured way to enable easy analysis, extraction or other use of data for better understanding data meaning. In our case, data is stored in a specific document store database, for example, MongoDB which its query language is similar to SQL and can be easily accessible with Java or PHP languages. This NoSQL database allows different types of data analysis as the results of data analysis should be delivered to the users, access to data can be enabled directly or through the web services.

CONCLUSION

In this study, we introduced a new approach to data interoperability for NoSQL databases. Our model is a stack of three main layers: meta layer, semantic layer and data warehouse. Each layer is for overcoming a particular obstacle to the interoperability of data. The meta layer APIs are responsible for data translation from original database to a common data model. After this step, mapping the data from common data model with the domain ontology is done by semantic layer APIs. Then, the resultant data is stored in the data warehouse which is a system used for reportingand data analysis.

RECOMMENDATIONS

In future research, we will test our model on a set of NoSQL databases (MongoDB, Redis and HBase). The choice of these systems is based on our comparative study between NoSQL systems (Hajoui *et al*., 2015). Especially, these databases belong to different categories. Finally, we will see how to improve our model to make it more efficient and operational.

REFERENCES

Anonymous, 2016. DB-Engines. Solid IT Company, USA. https://db-engines.com/en/

Anonymous, 2016. Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable. NoSQL, USA. http://nosql-database.org/

Bugiotti, F., L. Cabibbo, P. Atzeni and R. Torlone, 2013. A logical approach to NoSQL databases. Adv. Inf. Syst. Eng, 1: 1-12.

Cabibbo, L., 2013. ONDM: An object-NoSQL datastore mapper. MSc Thesis, Faculty of Engineering, Roma Tre University, Rome, Italy.

Chang, F., J. Dean, S. Ghemawat, W.C. Hsieh and D.A. Wallach *et al*., 2008. Bigtable: A distributed storage system for structured data. ACM. Trans. Comput. Syst., 26: 1-26.

Dharmasiri, H.M.L. and M.D.J.S. Goonetillake, 2013. A federated approach on heterogeneous NoSQL data stores. Proceedings of the 2013 International Conference on Advances in ICT for Emerging Regions (ICTer), December 11-15, 2013, IEEE, Colombo, Sri Lanka, ISBN:978-1-4799-1275-9, pp: 234-239.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisit., 5: 199-220.

Hajoui, O., R. Dehbi, M. Talea and Z.I. Batouta, 2015. An advanced comparative study of the most promising NoSQL and NewSQL databases with a multi-criteria analysis method. J. Theor. Appl. Inf. Technol., 81: 579-588.

Hitzler, P., M. Krotzsch, B. Parsia, P.F. Patel-Schneider and S. Rudolph, 2009. OWL 2 Web Ontology Language Primer. W3C Recommendation, Vol. 27,

Lim, H., Y. Han and S. Babu, 2013. How to fit when no one size fits. Proceedings of the Sixth Biennial Conference on Innovative Data Systems Research CIDR Vol. 4, January 6-9, 2013, Asilomar, CA, USA., pp: 1-12.

Martin, F., 2011. Polyglot persistence. Martin Flower.com, UK. https://martinfowler.com/bliki/PolyglotPersistence.html

Parent, C. and S. Spaccapietra, 2000. Database Integration: The Key to Data Interoperability. In: Advances in Object-Oriented Data Modeling, Papazoglou, M.P., S. Spaccapietra and Z. Tari (Eds.). The MIT Press, Massachusetts, USA., pp: 221-254.

Sellami, R., S. Bhiri and B. Defude, 2014. ODBAPI: A unified REST API for relational and NoSQL data stores. Proceedings of the 2014 IEEE International Congress on Big Data (BigData Congress), June 27-July 2, 2014, IEEE, France, ISBN:978-1-4799-5057-7, pp: 653-660.

Stonebraker, M., 2011. Stonebraker on NoSQL and enterprises. Commun. ACM., 54: 10-11.

Wiegand, N., 2011. INTEROP network to support geospatial data semantic interoperability. Proceedings of ASPRS 2011 Annual Conference on Ride on the Geospatial Revolution, May 1-5, 2011, Milwaukee, Wisconsin, pp: 1-5.

Wiegand, N., 2013. Semantic interoperability for data. Proceedings of the Workshop on Semantics in Geospatial Architectures: Applications and Implementation, October 28-29, 2013, University of Wisconsin-Madison, Madison, Wisconsin, pp: 1-26.

Yahia, N., S.A. Mokhtar and A. Ahmed, 2012. Automatic generation of OWL ontology from XML data source. Intl. J. Comput. Sci., Vol 9,