Automatic RDF-ization of big data semi-structured datasets

Ronald Gualán, Renán Freire, Andrés Tello, Mauricio Espinoza, Víctor Saquicela

Departamento de Ciencias de la Computación, Universidad de Cuenca, Avenida 12 de abril, Cuenca, Ecuador.

Autores para correspondencia: {ronald.gualan, renan.freirez, andres.tello, mauricio.espinoza, victor.saquicela}@ucuenca.edu.ec

Fecha de recepción: 19 de junio del 2016 - Fecha de aceptación: 24 de julio del 2016

ABSTRACT

Linked data adoption continues to grow in many fields at a considerable pace. However, some of the most important datasets usually remain underexploited because of two main reasons: the huge volume of the datasets and the lack of methods for automatic conversion to RDF. This paper presents an automatic approach to tackle these problems by leveraging recent Big Data tools and a program for automatic conversion from a relational model to RDF. Overall, the process can be summarized in three steps: 1) bulk transfer of data from different sources to Hive/HDFS; 2) transformation of data on Hive to RDF using D2RQ; and 3) storing the resulting RDF in CumulusRDF. By using these Big Data tools, the platform will cope with the handling of big amounts of data available in different sources, which can include structured or semi-structured data. Moreover, since the RDF data are stored in CumulusRDF in the final step, users or applications can consume the resulting data by means of web services or SPARQL queries. Finally, an evaluation in the hydro-meteorological domain demonstrates the soundness of our approach.

Keywords: Automatic transformation to RDF, data integration, Semantic Web, NoSQL, RDF, semi-structured sources, big data, D2RQ, Apache Hive, CumulusRDF, Apache ServiceMix.

RESUMEN

La adopción de Linked Data sigue creciendo en muchos campos a un ritmo considerable. Sin embargo, algunos de los conjuntos de datos más importantes por lo general permanecen des-semantificados debido a dos razones principales: el enorme volumen de los conjuntos de datos y la falta de métodos para la conversión automática a RDF. Este artículo presenta un enfoque automático para hacer frente a estos problemas mediante el aprovechamiento de nuevas herramientas de Big Data y un programa para la conversión automática de un modelo relacional a RDF. En general, el proceso implementado se puede resumir en tres pasos: 1) transferencia masiva de datos desde las diferentes fuentes hacia Hive/HDFS, 2) transformación de los datos en Hive a RDF utilizando D2RQ, y 3) almacenamiento del RDF resultante en CumulusRDF. De este modo, mediante el uso de estas herramientas de Big Data garantizamos que la plataforma sea capaz de hacer frente a las grandes cantidades de datos disponibles en diferentes fuentes, ya sea que contengan datos estructuradas o semi-estructurados. Además, puesto que los datos RDF se almacenan en CumulusRDF en la etapa final, los usuarios o aplicaciones pueden consumir los datos resultantes a través de servicios web o consultas SPARQL. Finalmente, una evaluación demuestra la solidez de nuestro enfoque.

Palabras clave: Transformación automática a RDF, integración de datos, Web Semántica, NoSQL, RDF, fuentes semi-estructuradas, Big Data, D2RQ, Apache Hive, CumulosRDF, Apache ServiceMix.

1. INTRODUCTION

We have reached an era where the vast amount of data generated every day is ever increasing. Thus, the amount of data generated at daily basis is comparable to the amount of data generated from the rise of computers until 2003 (Sagiroglu and Sinanc, 2013). In addition, it was estimated that 2007 was the year when, for the first time, we could not store all the information that we were generating (Bifet, 2013). The applications which generate such data are very diverse and vary in domain and purpose. The most common applications that have contributed with this data deluge are sensor networks, bioinformatics, traffic management, GPS signals from cell phones, log records or click-streams, manufacturing processes, retail companies, email, blogging, twitter and facebook posts, and many others (Knoblock *et al*., 2012; McAfee *et al*., 2012; Sagiroglu & Sinanc, 2013). Hence, the datasets generated by those applications varies from structured, semi-structured and unstructured datasets. All what these types of datasets have in common is the three main features: volume, variety, and velocity. These so called "the three Vs" are the main characteristics of a Big Data repository (Sagiroglu & Sinanc, 2013). Big Data involves different concepts, tools, and mechanisms that allow facing the management-complexity associated to this kind of datasets; than otherwise, it would be impossible to deal with.

In addition to the complexity associated to this type of datasets, the Big Data notion of *variety* is a generalization of semantic heterogeneity (Hitzler & Janowicz, 2013). Such semantic heterogeneity poses the need to provide a semantic meaning, specially, to those semi-structured and unstructured datasets. Hence, the use of Semantic Web facilitates the creation of conceptual models to formally describe the datasets by means of ontologies[1]. One common approach to provide semantic meaning and share the data contained in those ever increasing datasets is Linked Data. Linked Data takes the main principle of the World Wide Web of global identifiers and links, and apply them to raw data of any type, not only to documents (Hitzler & Janowicz, 2013). According to Bizer the Web of Data had around 31 billion RDF triples by 2012. In addition, Erling states that Linked Data and RDF should be part of every regular data-engineering stack (Bizer *et al*., 2012). This demonstrates that Big Data, Linked Data, and RDF are concepts that go hand by hand.

One common step in several methodologies for publishing Linked Data is the transformation of the raw data from the original sources to RDF (Hyland *et al*., 2014; Marshall *et al*., 2012; Villazón-Terrazas *et al*., 2012; Villazón-Terrazas *et al*., 2011; Zengenene *et al*., 2013). In this work we call such transformation process *RDF-ization.* Particularly, we focus on the RDF-ization of structured and semi- structured data sources, e.g., shapefiles, csv files, relational databases, excel files, XML, JSON, Web services, log files. We validate our approach using a dataset of the hydro-meteorological domain. The main contributions of our work are 1) an architecture for automatically generate RDF from heterogeneous semi-structured data sources, and 2) a Big Data repository for data in RDF format; i.e., we not only stay at the data conversion to RDF, but we leverage Big Data technologies which allow dealing with the ever increasing size of the Linked Data repositories.

The remainder of this document is as follows. Section 2 presents the related work. In section 3 we present a motivating example that illustrates the advantages and the applicability our approach. In section 4 we describe the architecture of our approach. Section 5 depicts the implementation details of the proposed architecture. In section 6 we present the evaluation of our approach in the hydro-meteorological domain. In section 7 we present the discussion of the results obtained after the evaluation. Finally, in section 8 we conclude the paper and propose paths for future research.

2. RELATED WORK

Nowadays when talking about Linked Data, and the semantification process to get there, is more

[1] In computer science and information science, ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse.

common to find studies where researchers and practitioners recognize that Linked Data repositories need to be treated in a different way, i.e., as Big Data repositories. In this section we provide an overview of the studies about Linked Data generation and the management of RDF repositories as Big Data.

There is a large volume of published studies describing the process of Linked Data generation. Corcho *et al*. (2012) proposed the transformation, publishing and exploitation of the data provided by the Spanish Meteorology Public Agency. In this study the authors mention that one of the main limitations is the lack of mechanisms to tackle the accelerated increasing of the datasets size which limited to keep only the data generated in the last week. Patni *et al*. (2010) presented a framework to transform and publish the data from the Meteorology Department of the University of Utah and to make them publicly available in the LOD cloud. Later on, Patni *et al*. (2011) proposed an extension to the previous work, extracting features from the RDF data which allow to detect and describe natural phenomena (e.g., storms, hurricanes, etc.).

People working with Linked Data realized that the volume of data to be stored and processed is increasing dramatically. This has motivated researchers working on Semantic Web to combine their area of expertise with Big Data. Cudré-Mauroux *et al*. (2013) published a study of Big Data tools for massive storage of semantic data, i.e., data in RDF format. They describe four types of NoSQL databases and execute different performance tests which are compared against the results obtained from using traditional RDF triple stores. In this study they conclude that Distributed NoSQL databases may produce similar results and sometimes overcome those than using a native RDF triple store (e.g., 4Store). For instance, in a trial using a cluster of 16 nodes to manage 1 billion of triples, they obtained better results using the NoSQL databases Cassandra and Jena+HBase than those using 4Store. In addition, they observed that the loading time of triples to the server scales more naturally when using NoSQL databases running in parallel.

Khadilkar *et al.* (2012) proposed a distributed RDF triple store built on top of the Apache Jena Framework[2] and the Apache HBase[3] distributed database. They rely on the built-in support for diverse RDF processing features provided by Jena, and the scalability provided by HBase since it uses the Hadoop Distributed File System (HDFS) as its underlying storage mechanism. They state that this framework supports all the functionalities provided by traditional RDF stores, including reification, inference and support for SPARQL queries.

Papailiou *et al.* (2012) presented H2RDF, a distributed RDF store that exploits the MapReduce processing framework provided by the HBase NoSQL database. One of the main features of this approach is that it enables simple and multi-join SPARQL queries over any number of triples. Cuesta et al. propose an architecture to process semantic data combining batch and real time processing (Cuesta *et al.*, 2013). In their approach they separate the management of large data volumes from the generation and exploitation of the data in real time. The data are stored compressed using RDF/HDT[4], while the real time processing requirements are handled using NoSQL databases.

Other approaches exist that exploit the capabilities of graph NoSQL databases, pointing out that the natural form of RDF data is a graph. Zeng et al. (2013) presented Trinity.RDF, a distributed memory-based graph engine for large volumes of RDF data. They propose to store RDF data in its native form, i.e., graphs. This approach shows a better performance for SPARQL queries with respect to traditional RDF triple stores (sometimes orders of magnitude better). In addition, they state that since the data are stored as a graph, the systems can support other kind of operations, e.g., random walks, reachability, etc.

Following a similar approach, Bouhali & Laurent (2015) proposed the transformation from RDF to NoSQL graph database format. They assert that since RDF's native format is a graph, the data should be in a graph database format. In addition, they state that Graph Database solutions generally show better querying performance than those SPARQL based frameworks.

---

[2] Apache Jenna Framework - https://jena.apache.org/

[3] Apache HBase - https://hbase.apache.org/

[4] HDT (Header, Dictionary, Triples) is a compact data structure and binary serialization format for RDF that keeps big datasets compressed to save space while maintaining search and browse operations without prior decompression.

In previous studies we observed that the problem is tackled from two perspectives. Some people are devoted to generate Linked Data, while others are focused on how to handle the vast amount of data being generated. In this work we focused on both aspects at the same time. In this study we propose an architecture to automatically transform semi-structured and unstructured datasets to RDF and store it in a BigData RDF repository.

3. MOTIVATING EXAMPLE

Today, institutions are generating a vast amount of data about different domains (hydrological, meteorological, social network, geo data, etc.). Generally, those institutions have their own data repositories. A user who works on analysis of data from different institutions finds interesting to know more details about the data, for instance, the data model used in each source, how to access the data, data formats (shapefile, CVS, database, excel, etc.), the relationship between entities, entity search, available tools, querying mechanisms, etc. The current situation when working with data is that each repository is managed as a silo and each repository has its own data model and access form. Thus, syntactic and semantic heterogeneity between repositories become a real challenge to the user. In addition, when a user tries to integrate different repositories, the storage problem appears due to the large amount of data associated with this task. Currently, the process of storing and querying data from different repositories used in integration processes is performed manually. This implies that the user needs knowledge of different technologies and methodologies that allow optimal storage and processing of large volumes of data to perform searches and integration processes.

Considering the limitations in storing and accessing the data, aiming to improve the user's experience and given the conditions mentioned in this context, this work proposes the implementation of an architecture that allows storing a vast amount of data through the use of semantic and NoSQL technologies.

4. ARCHITECTURE

The main purpose of our approach is to provide an automatic tool for translating data from heterogeneous sources into RDF that can be consumed through SPARQL queries using big-data. We do not aim at merging all the possible sources together integrating the underlying semantics. Instead, each source is translated to an independent RDF dataset.

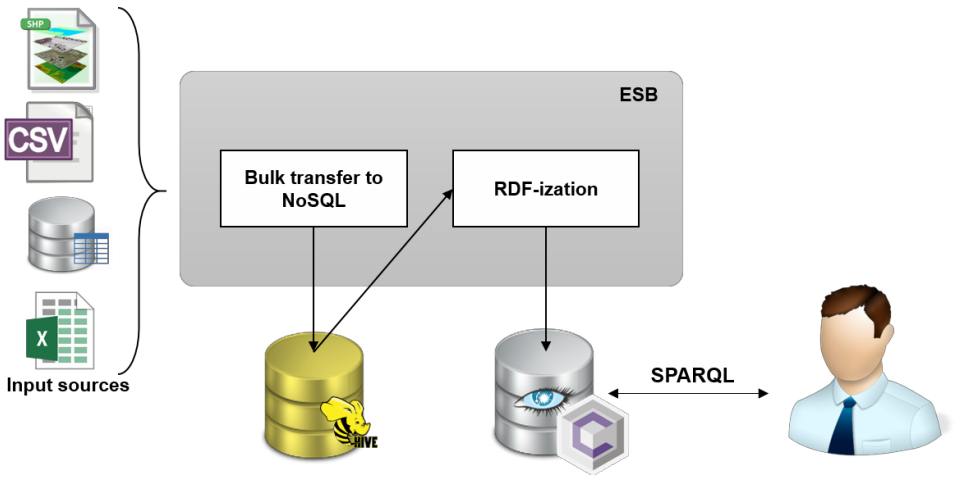

Figure 1. Architecture of the proposed automatic RDF-ization platform.

Many structured and semi-structured sources are commonly used as regular sources without the need to be considered as Big Data sources. For instance, a regular-size Shapefile is a common source of geospatial information. However, on more complex scenarios, like the one described in the previous section, where thousands of Shapefiles and many other sources (such us plain text, csv, excel files, XML, databases, etc.) store lots of information regarding the same topic (i.e. remote sensor data of a hydro-meteorological observatory) they could be considered as part of a Big Data application, because they present the 3Vs (volume, variety, and velocity) of Big Data (Zikopoulos *et al*., 2011).

The proposed architecture takes as input the aforementioned sources, processes them inside an Enterprise Service Bus platform (Chappell, 2004), and produces as a result a repository of independently stored datasets, where each source is corresponded with one dataset/graph. The final user can manipulate and query the data choosing the appropriate dataset. Figure 1 depicts the RDF-ization process which involves two main components: 1) bulk transfer data from the source file to a temporal NoSQL store, and 2) transformation of data on the NoSQL store into RDF. In these steps appropriate Big Data tools were chosen in order to guarantee scalability and distributed processing capabilities.

4.1. Bulk transfer to a temporal NoSQL database

In this component we used Apache Sqoop[5] (SQL to Hadoop) version 2 as the high-throughput transfer engine for dumping data from the source files to a temporal NoSQL database. Sqoop is an open source tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases (The Apache Software Foundation, 2016, p. 2). Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance (Prakashbhai & Pandey, 2014).

Since the main target of Sqoop are relational databases and NoSQL stores, it offers some ready-to-use connectors for those data sources. Nevertheless, the creation of a new connector to support a different kind of source is rather straightforward. As part of this work, we developed a connector for extracting data from a Shapefile source. The development process is described in the next section.

We used Apache Hive[6] as the temporal store of our approach, a popular data warehouse software which is being used as an alternative to store and process extremely large data sets on commodity hardware. Hive is an open-source data warehousing solution built on top of Hadoop, and supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop (Thusoo *et al*., 2009). These features (particularly the support of an SQL-like language) are the reason why we decided to use it as our temporal store where the data is stored before being transformed into RDF.

Development of a Sqoop2 Connector to extract data from Shapefiles

As part of this approach, we created a Sqoop 2 Connector to transfer data from Shapefiles because in the context of the use case described in Section 3 a considerable amount of geospatial information is available on these kind of files. Shapefile is a popular geospatial vector data format for geographic information systems (GIS). It is developed and regulated by ESRI as an open specification (mostly) for data interoperability between ESRI and other GIS software products (ESRI, 1998). The Shapefile format can spatially describe vector features: points, lines, and polygons, representing, for example, water wells, rivers, and lakes. Each item usually has attributes that describe it, such as name or temperature. From a simplistic perspective, data from a Shapefile are organized as records of a single table. Therefore, the idea to dump data from a Shapefile to HDFS, can be seen in an analogous manner to dump data from a table in a relational database to HDFS.

Two concerns were taken into account when developing the Shapefile Connector: 1) the Connector is of type FROM because we are interested only in extracting data from a Shapefile (The Apache Software Foundation, n.d.), and 2) it is mandatory to extract not only the extra metadata

[5] Apache Sqoop - http://sqoop.apache.org/
[6] Apache Hive - https://hive.apache.org/

attributes but also the vector data. Regarding the implementation, we decided to use the GeoTools library (GeoTools, n.d.).

According to the documentation (The Apache Software Foundation, n.d., p. 2), Sqoop2 connectors encapsulate the job lifecycle operations for extracting and/or loading data from and/or to different data sources. Each connector will primarily focus on a particular data source and its custom implementation for optimally reading and/or writing data in a distributed environment.

The development of the Shapefile Connector followed the Sqoop 2 Connector model, which captures the function of the Connector in four stages: Initialization, Partitioning, Extraction, and Destruction. Each stage is implemented in a separate class. Additionally, some classes defining the required configuration for accessing and defining runtime behavior are needed. All the functionality and structure of a Sqoop connector is deployed in a Jar file, which has to be registered in the Sqoop server before it can be used in an export job.

4.2. Automatic conversion to RDF

This section describes the second component of our approach, that is, the transformation of the relational data into RDF. As mentioned in the previous section, in the first component the bulk data transference with Sqoop2 delivers data to a temporal storage in an Apache Hive data warehouse. Following with the process, this second component ingests the temporal data stored on Apache Hive and transforms it to RDF. The core tools for accomplishing this task are D2RQ[7] platform and CumulusRDF[8] store. The former allows the conversion to RDF, while the latter is an RDF store where the generated RDF is placed.

The architecture of D2RQ consists of three main parts: 1) the D2RQ Mapping Language, a declarative mapping language for describing the relation between an ontology and an relational data model; 2) the D2RQ Engine, a plug-in for the Jena and Sesame Semantic Web toolkits, which uses the mappings to rewrite Jena and Sesame API calls to SQL queries against the database and passes query results up to the higher layers of the frameworks; and 3) D2R Server, an HTTP server that can be used to provide a Linked Data view, a HTML view for debugging and a SPARQL Protocol endpoint over the database. D2RQ allows that a relational database, which is maintained by a non-RDF legacy application, can also be accessed by a Non-RDF application. (Bizer & Seaborne, 2004; Bizer *et al*., 2009).

Although the foremost aim of D2RQ platform is to access relational databases as virtual, read-only RDF graphs, D2RQ also offers the option to create custom bulk transformation operations of relational data in RDF formats for loading it into an RDF store, which is the reason why we decided to use D2RQ.

RDF generation from a database D2RQ provides two tools to create an RDF dump of the database, the generate-mapping tool and the dump-rdf tool. The generate-mapping tool builds a D2RQ mapping file by analyzing the schema of an existing database. This mapping file, called the default mapping, maps each table to a new RDFS class that is based on the table's name, and maps each column to a property based on the column's name. The mapping relies on the D2RQ Mapping Language, a declarative language for mapping relational database schemas to RDF vocabularies and OWL ontologies. The mapping file assists the dump-rdf tool to dump the contents of the relational database into a single RDF file (C. Bizer *et al*., 2009).

Since Apache Hive is not yet supported nor has been tested by D2RQ, some adaptations on the source code of D2RQ were necessary. These changes to the source code constitute a minor contribution of this work. The first change sets that the syntax of Hive is similar to the syntax of MySQL, so it was not necessary to develop a new syntax, we just reused that of MySQL. A second change specifies that the full name of the columns when assembling a SQL query, shall not be composed by the name of the schema. And the third and final change was to include the jar files of the JDBC library of Hive inside the lib folder of D2RQ.

[7] D2RQ - http://d2rq.org/
[8] CumulusRDF - https://github.com/cumulusrdf/cumulusrdf

5. IMPLEMENTATION

Our prototype was implemented using the Apache Camel[9] integration framework, whose core is based on a routing engine that allows transfer of source-destination messages by mean of Enterprise Integration Patterns (EIPs). Camel has an architecture that facilitates interaction with any type of application, regardless of the data type or protocol used, avoiding unnecessary conversions (Ibsen & Anstey, 2010).

A number of EIPs were used to implement the approach described in this manuscript. According to Hohpe and Woolf (2004) these elements are essential for communication between applications. These kind of patterns are built and supported in most Enterprise Service Bus (ESB) as Apache ServiceMix, FUSE JBOSS, MULE, etc. (Ortiz Vivar & Segarra Flores, 2015).

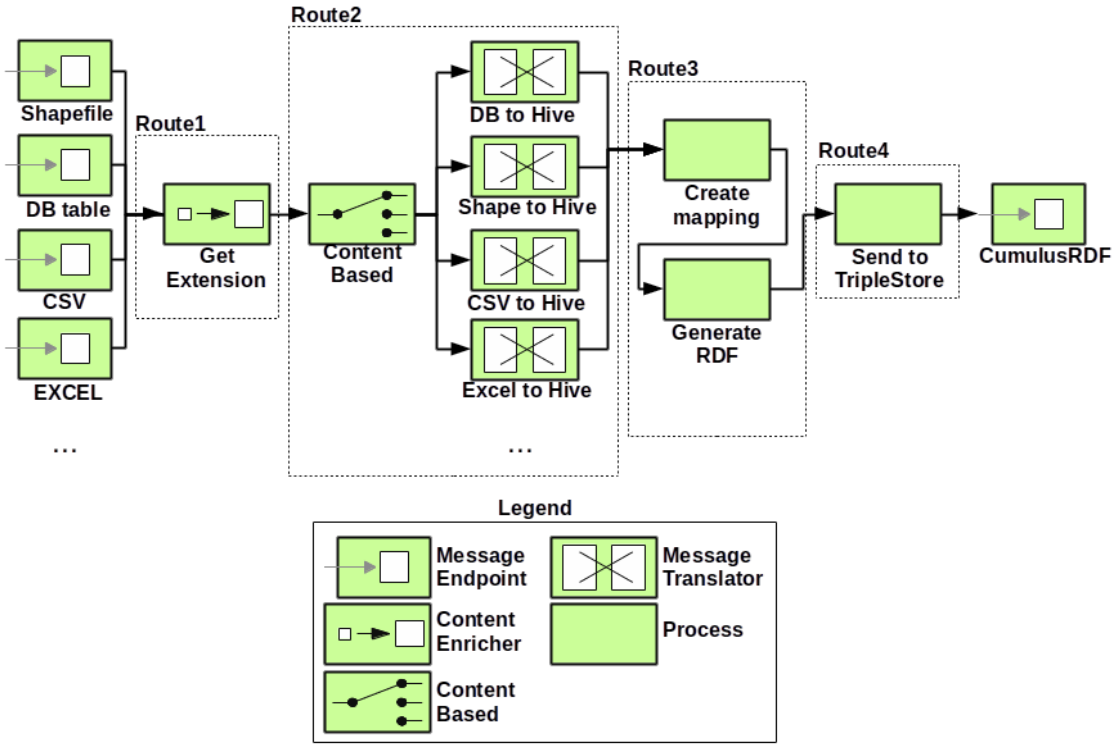

Figure 2. RDF-ization Process with Enterprise Integration Patterns.

Figure 2 depicts the proposed approach, which was implemented using four routes. Routes 1 and 2 conditionally process each source type to bulk transfer data from the source to Hive. Route 1 uses the Content Enricher EIP to add the file extension of the source as a field in the message header. The file extension is used to choose a conditional processing path for each kind of source, because for each source an appropriate Sqoop Source Connector is used. For instance, when processing a Shapefile, the Sqoop transference job is configured to use our Shapefile Connector, and accordingly when extracting data from a database the Generic JDBC Connector is used. On the other hand, since all the data is bulk transferred to Hive, the chosen connector for this destiny is Kite[10].

Route 3 performs the conversion from Hive to RDF using D2RQ in two steps. The first step is to generate a mapping file based on the structure of HIVE data, and the second step is to build an automatic RDF dump of the Hive data. Finally, Route 4 sends the RDF data on the dump file to CumulusRDF.

The prototype presented in this section was deployed on ServiceMix during the test phase. The architecture of ServiceMix is based on OSGI (Open Services Gateway Initiative) and can be

[9] Apache Camel - http://camel.apache.org/
[10] Apache Sqoop-KITE -https://cwiki.apache.org/confluence/display/SQOOP/Kite+Connector+Design

integrated with other projects such as Apache Camel, ActiveMQ, CXF, Karaf, etc. (Dirksen, n.d.).

6. EVALUATION

We evaluated our approach by RDF-izing a number of sources of different types, such as Shapefile, database, CSV, XML, among others. In this section we will describe an example using a Shapefile, which encloses information about the location of a network of remote hydro-meteorological stations in the Paute basin (See Figure 3).

(a)

	COD	CUENCA_HID	UTMX	UTMY	COTA	TIPO	ADMINISTRA
0	PRO-014-PG	Joyagshi	726012.00	9737960.00	2811	P	PROMAS
1	PRO-015-PG	Llagos	724044.00	9738880.00	2666	P	PROMAS
2	PRO-016-PG	Pacchala	723821.00	9741564.00	2130	P	PROMAS
3	PRO-017-PG	Santa Rosa	726452.00	9741199.00	2579	P	PROMAS
4	PRO-018-PG	Puay	726239.00	9744969.00	2377	P	PROMAS
5	PRO-019-PG	Chico Namza	725665.00	9746066.00	1820	P	PROMAS
6	PRO-020-PG	Namza Lise	725404.00	9744935.00	1856	P	PROMAS
7	PRO-021-PG	Totoras	760082.00	9750736.00	3640	P	PROMAS

(b)

Figure 3. Plot of the Shapefile (a) and a fragment of the atributes (b) (obtained with QGIS).

Our implementation deployed in ServiceMix automatically processes the files placed on a predefined folder, where the three components of our approach are carried out: 1) bulk transference of data to Hive/HDFS, 2) transformation from relational model of Hive to RDF using D2RQ, and 3) storing the resulting RDF to CumulusRDF.

Listing 1 shows a fragment of the mapping file used for D2RQ to generate RDF data from the Hive temporal data for the Shapefile source. Three mapping sections can be seen: the first maps the database; the second section maps the table (in this case, the Shapefile temporary table as a *d2rq:ClassMap*); and the final section maps each column or property with the table.

```
@prefix map: <#> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .

# Mapping of the database
map:database a d2rq:Database;
d2rq:jdbcDriver "org.apache.hive.jdbc.HiveDriver"; d2rq:jdbcDSN
"jdbc:hive2://localhost:10000/databasehive"; d2rq:username "ronald".
.
# Table shapefile
map:databasehive_shapefile a d2rq:ClassMap;
d2rq:dataStorage map:database; d2rq:uriPattern
"databasehive/shapefile"; d2rq:class
vocab:databasehive_shapefile;
d2rq:classDefinitionLabel "shapefile".
# Properties
map:databasehive_shapefile_featureid a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:databasehive_shapefile;
d2rq:property vocab:databasehive_shapefile_featureid;
d2rq:propertyDefinitionLabel "shapefile featureid"; d2rq:column
"shapefile.featureid".
```

Listing 1. Portion of the D2RQ mapping file for the HIVE temporal table of the Shapefile source.

Once the RDFization process has been successfully completed, we can query the generated data from Cumulus. Listing 2 shows a snippet of the data returned by a SPARQL query for the RDF data of the Shapefile.

```
@prefix dump: <www.ucuenca.edu.ec/dump#> .
@prefix vocab: <www.ucuenca.edu.ec/vocab> .

dump:databasehive/shapefile vocab:/databasehive_shapefile_administra "PROMAS" .
dump:databasehive/shapefile vocab:/databasehive_shapefile_cota "2811"^^<http://www.w3.org/2001/XMLSchema#integer> .
dump:databasehive/shapefile vocab:/databasehive_shapefile_utmy "9737960.0E0"^^<http://www.w3.org/2001/XMLSchema#double> .
dump:databasehive/shapefile vocab:/databasehive_shapefile_utmx "726012.0E0"^^<http://www.w3.org/2001/XMLSchema#double> .
dump:databasehive/shapefile vocab:/databasehive_shapefile_cuenca_hid "Joyagshi" .
dump:databasehive/shapefile vocab:/databasehive_shapefile_cod "PRO-014-PG" .
dump:databasehive/shapefile vocab:/databasehive_shapefile_the_geom "POINT (726012 9737960)" .
dump:databasehive/shapefile vocab:/databasehive_shapefile_featureid "Red_de_Estaciones.1" .
```

Listing 2. Snipped of the result of a SPARQL select for the RDF data generated from the Shapefile source. This RDF was simplified with the use of prefixes for easy of understanding.

7. CONCLUSIONS AND FUTURE WORK

We presented an approach for automatic transforming vast amounts of heterogeneous structured and semi-structured data sources to RDF by using big data tools and D2RQ. The proposed transformation process has three main components: 1) bulk transfer of data from different sources to Hive/HDFS, 2) transformation from relational model of Hive to RDF using D2RQ, and 3) storing the resulting RDF to CumulusRDF. Our approach leverages the power of Big Data tools to deal with the tremendous amounts of data faced in several domains, like the Hydro-meteorological, which was used as the motivating example of this work. Since the resulting RDF generated is placed in CumulusRDF, the final users are enabled to consume this information by means of Web Services or SPARQL queries. The soundness of our platform was demonstrated by RDFizing a Shapefile.

Our subsequent work will be focused on improving the automatic transformation to RDF by including methods that can be able to map the concepts of the data sources to the concepts of any given (parameterized) ontology, which would serve as the domain ontology. This will improve the applicability of our approach to an even wider range of domains.

ACKNOWLEDGEMENTS

This work has been funded by the DIUC (Dirección de Investigación de la Universidad de Cuenca), Research Direction of the University of Cuenca through the project “Integración, Almacenamiento y Explotación de datos Hidro Meteorológicos utilizando Big Data y Web Semántica”.

REFERENCES

Atemezing, A., O. Corcho, D. Garijo, J. Mora, M. Poveda Villalon, P. Rozas, D. Vila-Suero, B. Villazón-Terrazas, 2012. *Transforming meteorological data into linked data. Semantic Web*. Undefined, 1, 1-5, IOS Press. Available at http://www.semantic-web-journal.net/sites/default/files/swj281_0.pdf.

Bifet, A., 2013. *Mining big data in real time*. Informatica, 37, 4 pp. Available at http://ailab.ijs.si/dunja/TuringSLAIS-2012/Papers/Bifet.pdf.

Bizer, C., P. Boncz, M.L. Brodie, O. Erling, 2012. The meaningful use of big data: four perspectives–four challenges. *ACM SIGMOD Rec*., 40, 56-60.

Bizer, C., A. Seaborne, 2004. *D2RQ-treating non-RDF databases as virtual RDF graphs*. In: Proceedings of the 3rd International Semantic Web Conference (ISWC2004). Citeseer Hiroshima.

Bouhali, R., A. Laurent, 2015. *Exploiting RDF Open Data Using NoSQL Graph Databases*. In: Chbeir, R., Y. Manolopoulos, I. Maglogiannis, R. Alhajj, (Eds.). Artificial Intelligence Applications and Innovations. In: Iliadis, L., I. Maglogiannis, G. Tsoumakas, I. Vlahavas, M. Bramer (Eds.). Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI'2009), April 23-25, 2009, Thessaloniki, Greece.

Bizer, C., R. Cyganiak, J. Garbers, O. Maresch, C. Becker, 2009. The D2RQ Platform v0.7 - *Treating Non-RDF Relational Databases as Virtual RDF Graphs - User Manual and Language Specification*. Available at http://wifo5-03.informatik.uni-mannheim.de/bizer/ d2rq/spec/20090810/.

Chappell, D., 2004. *Enterprise service bus*. O'Reilly Media, Inc.

Cudré-Mauroux, P., I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F.L. Keppmann, D. Miranker, J.F. Sequeda, M. Wylot, 2013. *NoSQL databases for RDF: an empirical evaluation*. In: The Semantic Web-ISWC 2013. Springer, pp. 310-325. Available at http://ribs.csres.utexas.edu/nosqlrdf/nosqlrdf_iswc2013.pdf.

Cuesta, C.E., M.A. Martínez-Prieto, J.D. Fernández, 2013. *Towards an architecture for managing big semantic data in real-time*. In: Software Architecture. Springer, pp. 45-53. Available at http://dataweb.infor.uva.es/wp-content/uploads/2013/04/ecsa2013.pdf.

ESRI, 1998. *ESRI Shapefile Technical Description*.

GeoTools, n.d. GeoTools *The Open Source Java GIS Toolkit - GeoTools* [WWW Document]. URL http://geotools.org/ (accessed 1.19.16).

Hitzler, P., K. Janowicz, 2013. Linked Data, Big Data, and the 4th Paradigm. *Semantic Web*, 4, 233-235.

Hohpe, G., B. Woolf, 2004. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional.

Hyland, B., G. Atemezing, B. Villazón-Terrazas, 2014. *Best practices for publishing linked data*. Available at http://www.w3.org/TR/ld-bp/.

Ibsen, C., J. Anstey, 2010. *Camel in action*. Manning Publications Co.

Jos Dirksen (n.d.). *ServiceMix 4.2 - DZone - Refcardz* [WWW Document]. dzone.com. URL https://dzone.com/refcardz/servicemix (accessed 6.10.16).

Khadilkar, V., M. Kantarcioglu, B. Thuraisingham, P. Castagna, 2012. *Jena-HBase: A distributed, scalable and efficient RDF triple store*. In: Proceedings of the 11th International Semantic Web Conference Posters & Demonstrations Track, ISWC-PD. Citeseer, pp. 85-88.

Knoblock, C.A., P. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan, P. Mallick, 2012. *Semi-automatically mapping structured sources into the semantic web*. In: The Semantic Web: Research and Applications. Springer, pp. 375-390.

Ladwig, G., A. Harth, 2011. *CumulusRDF: linked data management on nested key-value stores*. In: The 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011). p. 30.

Marshall, M.S., R. Boyce, H.F. Deus, J. Zhao, E.L. Willighagen, M. Samwald, E. Pichler, J. Hajagos, E. Prud'hommeaux, S. Stephens, 2012. *Emerging practices for mapping and linking life sciences data using RDF - A case series*. Web Semant. Sci. Serv. Agents World Wide Web, 14, 2-13.

McAfee, A., E. Brynjolfsson, T.H. Davenport, D.J. Patil, D. Barton, 2012. Big data. The Management Revolution. *Harv. Bus Rev*., 90, 61-67.

Ortiz Vivar, J.E., J.L. Segarra, 2015. *Plataforma para la anotación semántica de servicio web RESTful sobre un bus de servicios*. Bachelor Thesis, Escuela de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de Cuenca, 189 pp. Available at http://dspace.ucuenca.edu.ec/handle/123456789/23105.

Papailiou, N., I. Konstantinou, D. Tsoumakos, N. Koziris, 2012. *H2RDF: adaptive query processing on RDF data in the cloud*. In: Proceedings of the 21st International Conference Companion on World Wide Web. ACM, pp. 397-400.

Patni, H., C.A. Henson, M. Cooney, A.P. Sheth, K. Thirunarayan, 2011. *Demonstration: real-time semantic analysis of sensor streams.* Proceedings of the 4th International Workshop on Semantic Sensor Networks, 119-122. Availabe at https://works.bepress.com/tk_prasad/102/.

Patni, H., C.A. Henson, A.P. Sheth, 2010. *Linked sensor data*. In: Collaborative Technologies and Systems (CTS), 2010 International Symposium on. IEEE, pp. 362-370.

Prakashbhai, P.A., H.M. Pandey, 2014. *Inference patterns from Big Data using aggregation, filtering and tagging - A survey*. In: Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-. IEEE, pp. 66-71.

Sagiroglu, S., D. Sinanc, 2013. *Big data: A review*. In: Collaboration Technologies and Systems (CTS), 2013 International Conference on. IEEE, pp. 42-47.

The Apache Software Foundation, 2016. *Apache Sqoop* [WWW Document]. URL https://sqoop.apache.org/ (accessed 6.7.16).

The Apache Software Foundation (n.d.). *Sqoop 2 Connector Development - Apache Sqoop documentation* [WWW Document]. URL http://sqoop.apache.org/docs/1.99.6/ConnectorDevelopment.html (accessed 1.19.16).

Thusoo, A., J.S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, 2009. *Hive: a warehousing solution over a map-reduce framework.* Proc. VLDB Endow. 2, 1626-1629.

Villazon-Terrazas, B., D. Vila-Suero, D. Garijo, L.M. Vilches-Blazquez, M. Poveda-Villalon, J. Mora, O. Corcho, A. Gomez-Perez, 2012. *Publishing Linked Data-There is no One-Size-Fits-All Formula*. Available at http://oa.upm.es/14465/1/2.formulaLD.pdf.

Villazón-Terrazas, B., L.M. Vilches-Blázquez, O. Corcho, A. Gómez-Pérez, 2011. *Methodological guidelines for publishing government linked data*. In: Linking Government Data. Springer, pp. 27-49.

Zeng, K., J. Yang, H. Wang, B. Shao, Z. Wang, 2013. *A distributed graph engine for web scale RDF data*. In: Proc. VLDB Endow, 6, 265-276.

Zengenene, D., V. Casarosa, C. Meghini, 2013. *Towards a Methodology for Publishing Library Linked Data*. In: Bridging Between Cultural Heritage Institutions. Springer, pp. 81-92.

Zikopoulos, P., C. Eaton, C., others, 2011. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media.