Standardization of Storage and Retrieval of Semi-structured Thermophysical Data in JSON-documents Associated with the Ontology

© A.O. Erkimbaev © V.Yu. Zitserman © G.A. Kobzev © A.V. Kosinov

Joint Institute for High Temperatures, Russian Academy of Sciences, Moscow, Russia

adilbek@ihed.ras.ru vz1941@mail.ru gkbz@mail.ru kosinov@gmail.com

Abstract. A new technology for data management of a complex and irregular structure is proposed. Such a data structure is typical for the representation of the thermophysical properties of substances. This technology based on storage of data in JSON files is associated with ontologies for the semantic integration of heterogeneous sources. Advantages of JSON-format – the ability to store data and metadata within a text document, accessible perceptions of a person and a computer and support for the hierarchical structures needed to represent semi-structured data. Availability of a multitude of heterogeneous data from a variety of sources justifies the use of the Apache Spark toolkit. When searching, it is supposed to combine SPARQL and SQL queries. The first one (addressed to the ontology repository) provides the user with the ability to view and search for adequate and related concepts. The second, accessed by JSON documents, retrieves the required data from the body of the document. The technology allows to overcome a variety of schemes, types and formats of data from different sources and implement a permanent adjustment of the created infrastructure to emerging objects and concepts not provided for at the stage of creation.

Keywords: thermophysical properties, semi-structured data, JSON-format, ontology.

1 Introduction

The constantly increasing volume and complexity of the data structure on the substances and materials properties imposes stringent requirements for the information environment that integrates diverse resources belonging to different organizations and states. In contrast to the earth science or medicine, here the source of data is the growing publication flow. In so doing the volume of data is determined not so much by the number of objects studied, as by the unlimited variety of conditions for synthesis, measurement, morphological and microstructural features, and so on. It can be said that of the three defining dimensions of Big Data (the so-called "3V-Volume, Velocity, Variety" [3]), it is the latter plays a decisive role, that is, an infinite variety of data types.

In this paper, we propose a set of solutions borrowed from Big Data technology, allowing to overcome with minimum expenses two main difficulties in the way of integration of resources. The first one is the variety of accepted schemes, terminologies, types and formats of data and so on, and the second is the need for permanent adaptation of the created structure to the emerging variations in the nomenclature of terms (objects, concepts etc) not provided at the design stage. The need for variation in the data structure can be associated with the expansion of the range of substances (e.g. by including nanomaterials), the range of properties (e.g. by including state diagrams), or by changing the data type, say with the transition from constants to functions.

The solutions proposed in the work are based on the joint use of previously used technologies:

- Data interchange standard in the form of text-based structured documents, each of which is treated as an atomic storage unit;
- Ontology-based data management;
- General framework **Apache Spark** for large-scale data processing.

The three main elements of the planned data infrastructure (Figure 1): a plethora of primary data sources of different structures (databases, file archives, etc.) subject to conversion to the standard JSON format; ontologies and controlled dictionaries for the semantic integration of disparate data; Big Data technology for storing, searching and analyzing data.

2 Data preparation

2.1 General scenario

The general scheme of data preparation (Figure 2) assumes as an initial material a large body of external resources, thematically related, but arbitrary in terms of volume, structure and location. Among them are sources of structured data which include factual SQL databases (DB), document-oriented DB, originally structured files in **ThermoML** [7] or **MatML** [9] standards, numerical tables in CSV or XLS formats and so on. The second group (possibly dominant in terms of volume) is formed by unstructured data: text, images, raw experimental or modeling data etc.

Proceedings of the XIX International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID/RCDL'2017), Moscow, Russia, October 10–13, 2017

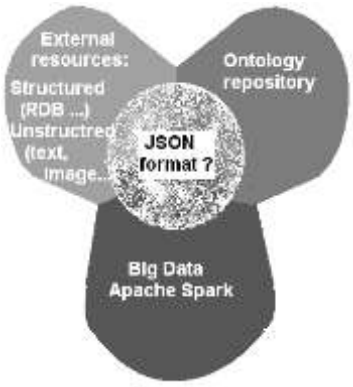

Figure 1 Key elements of the data infrastructure concept

The first stage of data preparation is the downloading of records from external sources with their subsequent conversion to the standard form of JSON documents [2]. In so doing, the conversion of structured documents can be entrusted to software whereas the unstructured part is subject to "manual" processing with the extraction of relevant information from the texts. Finally, the control element in this scheme is the repository of subject specific (domain) and auxiliary ontologies.

The distinctive characteristic of the proposed approach is that the starting data sources remain "isolated" and unchanged. Resource owners periodically download data to JSON files by templates linked with ontological models.

In so doing they determine themselves the composition, amount and relevance for the "external" world of the data being download. This type of interaction is passive, in contrast to active, when client can use the **JDBC** or **ODBC** interface to access databases.

2.2 JSON-documents

The basic unit of storage is a structured text document recorded in JSON format, one of the most convenient for data and metadata interchange propose [2]. The advantage of JSON-document – text-based language-independent format, is easy to understand and quickly mastered, a convenient form of storing and exchanging arbitrary structured information. Previously, structured text based on the XML format was proposed as a means of storing and exchange thermophysical data in the **ThermoML** project [7] and data on the properties of structural materials in the **MatML** project [9].

Here, a text document is proposed as the main storage unit, written in JSON format, which is less overloaded with details, simplifying the presentation of the data structure, reducing their size and processing time. In particular, the JSON format is shorter, faster read and written, can be parsed by a standard JavaScript function, rather than a special parser, as in the case of the XML format [https://www.w3schools.com/js/js_json_xml.asp].

Among other advantages of the format, one can note a simple syntax, compatibility with almost all programming languages, as well as the ability to record hierarchical, that is, unlimited nested structures such as "key-value". By way of **value** may be accepted object (unordered set of key-value pairs), **array** (ordered set of values), string, number (integer, float), **literal** (true, false, null). It is also important that the JSON format is a working object for some platforms, in particular for **Apache Spark**, allowing for the exchange, storage and queries for distributed data.

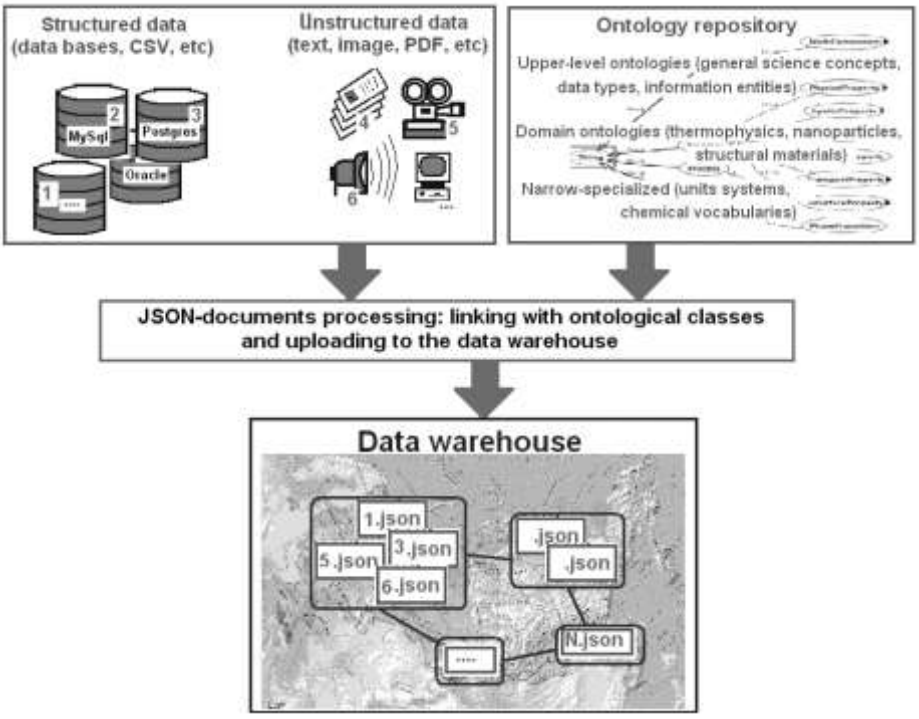

Figure 2 Schematic sketch of initial data processing

The rich possibilities of JSON-format as a means of materials properties data interchange attracted the attention of developers of the Citrination platform [10]. They proposed JSON-based hierarchical scheme **PIF** (physical information file), detailing the object, **System**, whose fields include three data groups, explaining what an object is (name, ID), how it was created/synthesized the generality of the created scheme should be sufficient for storing objects of arbitrary complexity, "from parts in a car down to a single monomer in a polymer matrix". Flexibility of the **PIF**-scheme is achieved due to additional fields and objects, as well as the introduction of the concept of **category**. This concept is nothing but a version of the scheme, oriented to a certain kind of objects, say substances with a given chemical identity.

2.3 Ontology-based data management

The second stage of data preparation is the linking of extracted metadata with concepts from ontologies and dictionaries assembled into a single repository. The management of the repository is entrusted to an **ontology-based data manager**, which allows for the search and editing of terms (class) ontologies, as well as their binding to JSON documents, Figure 3. This means that when the particular source schema is converted to a JSON format, terms from ontologies, rather than source attributes, are used as its keys. It is also possible to use additional keys for a detailed description of the data source itself, for example, indicating the type of DBMS, name and format of textual or graphical file, authorship and other official data, "sewn up" in the atomic "unit" of storage.

The role of ontologies is to introduce semantics (a common interpretation of meaning) into documents, as well as the ability to adjust the data structure of the JSON-documents by editing the ontology.

Linking documents with ontologies allows to perform semantic *(meaningful*) data search (more precisely, metadata) using SPARQL queries, which makes it possible to reveal the information of the upper and lower levels (*super and sub-classes*) and side-links (*related terms*), without knowing the schema of the source data. Thus, the user can view and retrieve information without being familiar with the conceptual schema of a particular DB or the metadata extracted from unstructured sources.

The repository should include three types of ontologies and controlled vocabularies: *upper-level*, *domain* and *narrow-specialized*. The first type is scientific top-level ontology, which introduces the basic terminology used in different fields, for example such concepts as **substance, molecule, property, state**, as well as informational entities that reflect the representation of data: **data set, data item, document, table, image,** etc. Most of these terms and links between them can be borrowed from ontologies presented on the server **Ontobee** [11], for example **SIO** (Semanticscience Integrated ontology) or **CHEMINF** (Chemical Information ontology).The second type of ontology (domain ontology) should cover the terminology of certain subject areas, for example, thermophysics, structural materials, nanostructures, etc. For each of the domains, as a rule, some ontologies previously created and presented in publications or Web are already available on the basis of which it is possible to build and further maintain its own subject-specific ontology. Finally, the third type (narrow-specialized) should include ontologies or vocabularies for systems of units (for example, **UO** on the above portal **Ontobee**) or chemical dictionaries, for example **ChemSpider** [6] and the like. Figure 3 illustrates the binding of terms from a JSON document to ontological terms.

Even at the stage of data preparation the proposed technology provides:

- consistency with accepted standards regardless of the structure and format of the original data;
- semantic integration of created JSON-documents;
- inclusion of previously not provided objects and concepts by expanding classes or introducing new ontologies and dictionaries.

The scheme of the generated data is determined by the initial data scheme with subsequent correction in the process of linking with the terms and structure of the corresponding ontological model. It should be noted that JSON-documents are objects with which one can operate using external API. In so doing, there is always the possibility of accessing keys in JSON documents not currently linked with a particular ontology term.

At the same time, it seems justified to identify or bind not only keys, but also values with ontological terms. For example, the key/value pair "Property": "Heat Capacity" is presented in Figure 3. This will allow in the future to facilitate the formation of SQL query, relying on the information received from the ontologies repository.

The experience of using ontology in the data interchange through text documents has already been implemented in a special format **CIF** (*Crystallographic Information File*) [8], intended for crystallographic information.

In other cases of using a JSON document for the storage of scientific data (for example, in the mentioned Citrination system [10]), the categorization and introduction of new concepts is carried out by a special commission without linking with the concepts of ontological models.

3 Technique of storage and access to data

Given the increasing volume and distributed nature of the data on the properties, some of the Big Data technologies would be appropriate for infrastructure design. Their advantage is due not so much to high performance in parallel computing, but rather to a pronounced orientation to work with data (storage, processing, analysis and so on) in a distributed environment (when data sources are located on remote severs). Among the available open-source means, the **Apache Spark** high-performance computing platform [5] is offered here. Along with other technological features, it is distinguished by the presence of built-in libraries for complex analytics including running SQL-queries. By means of SQL-queries one can access the contents of structured JSON documents. It is the ability of SQL-queries to data plays a key role in the task of their

integration. The efficiency of Spark in the storage and processing of data is also associated with its ability to maintain interaction with a variety of store types: from **HDFS** (Hadoop Distributed Files System) to traditional database on local servers. We should also note the built-in library **GraphX** – an application for processing graphs, which provides our project with our own tools for working with ontologies.

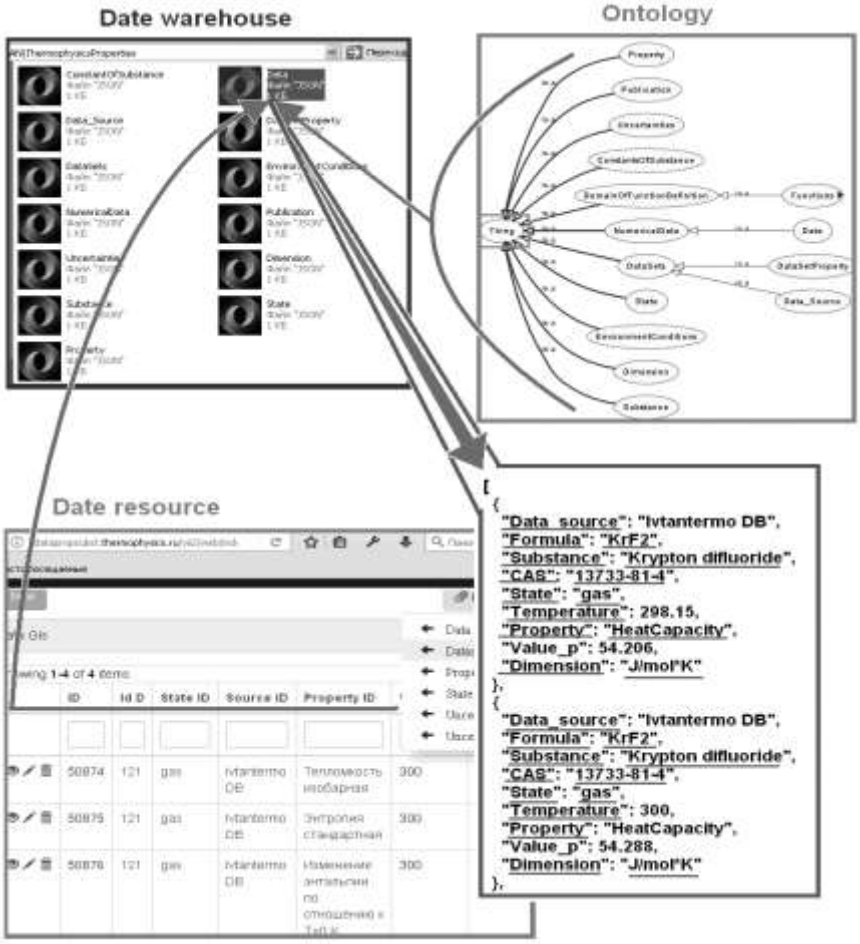

Figure 3 Linking JSON documents to ontology classes using the example of the ontology for the domain of thermophysical properties

The computing platform shown in Fig. 4 includes three basic elements:

• Management-dispatching of distributed computing under the control of Hadoop YARN, Apache Mesos or stand-alone;

• Computer interface – API for languages Scala, Java и Python;

• Powerful and diverse means of data storage.

Advantages of **Apache Sp**ark in comparison with other technologies (MapReduce) – higher computing speed and the ability to handle data of different nature (text, semi-structured and structured data, etc.) from various sources (files of different formats, DBMS and streaming data). It is also important to have APIs for Scala, Java and Python and high-level operators to facilitate code writing, integration with the **Hadoop** ecosystem [4], which unites libraries and utilities provided for in Big Data technology. The ecosystem has I/O interfaces (InputFormat, OutputFormat) that are supported by a variety of file formats and storage systems (**S3, HDFS, Cassandra, Hbase, Elasticsearch** and so on).

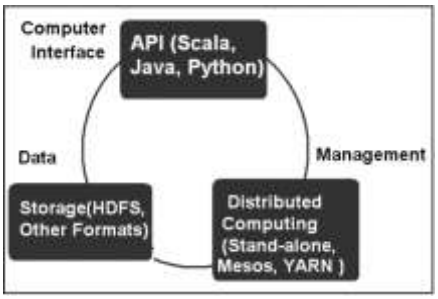

Figure 4 Spark Architecture

For storage purposes, the Apache Spark provides for interaction with three groups of data sources:

Files and file systems – local and distributed file systems (**NFS, HDFS, Amazon S3**), capable of storing data in different formats: text, JSON, SequenceFiles (binary key/value pairs) etc;

Sources of structured data available through **Spark SQL**, including JSON, **Apache Hive**;

Relational databases and key/value pairs storages (**Cassandra, HBase**), accessed by built-in and external libraries of interaction with the databases such as **JDBC/ODBC** or search engine **Elasticsearch**.

In doing so **Spark** supports loading and saving data from different formats files: unstructured (text), semistructured (JSON), structured (such as CSV or SequenceFiles). The **Apache Spark** operation reduces to the formation and transformation of **RDD** (*Resilient Distributed Dataset*) sets, which are distributed collections of elements. When parallel processing, the data is automatically distributed in sets between the computers in the cluster. **RDD** sets can be created by importing **HDFS** files using the Hadoop InputFormats tool or by converting from another **RDD**.

The main task (access and search among structured and semistructured data) is implemented by the **Spark SQL** module, which is one of the built-in Apache Spark libraries. **Spark SQL** defines a special type of **RDD** called **SchemaRDD** (in recent versions, the term **DataFrame** is used). The **SchemaRDD** class represents a set of objects row, each of which is a normal record. The **SchemaRDD** type is called a schema (a list of data fields) of records. **SchemaRDD** supports a number of operations that are not available in other sets, in particular, execution of SQL-queries. **SchemaRDD** sets can be created from external sources (JSON-documents, Hive tables), query results and from ordinary **RDD** sets.

Three main features of **Spark SQL**:

- It can download data from different sources;
- It requests data using SQL within Spark programs and from external tools related to **Spark SQL** through standard mechanisms for connecting to databases via **JDBC/ODBC**;
- It provides an interface between SQL and ordinary code (when used within Spark SQL programs), including the ability to connect sets of **RDD**s and SQL tables.

It is possible to configure **Spark SQL** to work with structured data **Apache Hive**. The **Apache Hive** store is specifically designed for querying and analyzing large data sets, both inside **HDFS** and in other storage systems. **JDBC/ODBC** standards are also supported by **Spark SQL**, which allows executing a direct SQL query to external relational databases, in the case of the above defined active type of interaction.

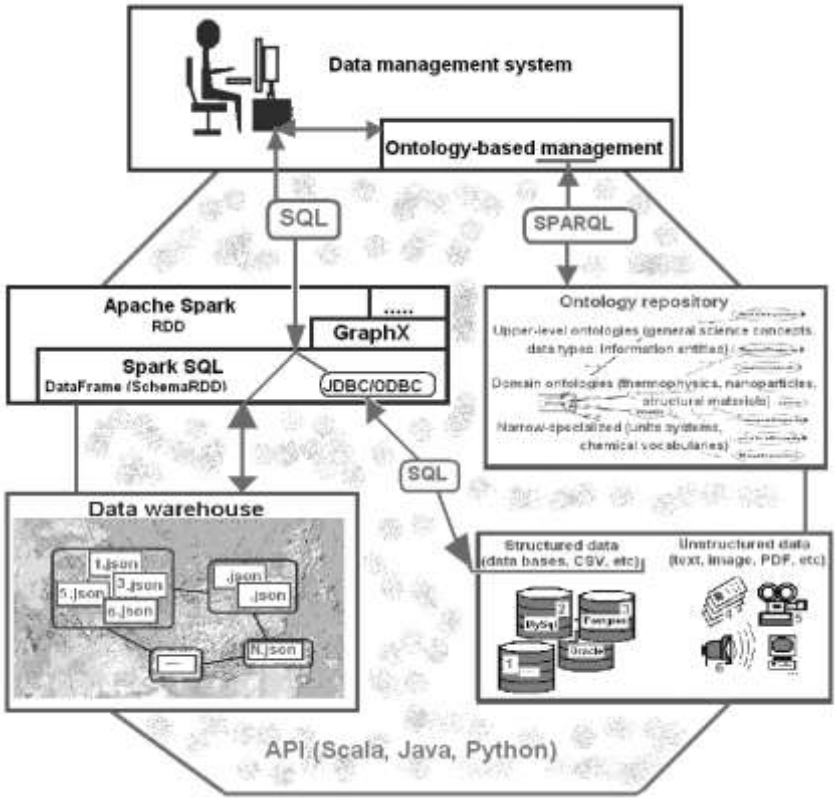

Figure 5 Web-environment for managing heterogeneous and distributed data on the substance properties (databases, unstructured and semistructured files and so on)

The main scenario that uses the **Apache Spark** features is shown in Figure 5. As a result of uploading data to JSON documents according to the above procedure, we will have data sets with a single classifying key system identical to terms from ontologies. The organization of data requests in JSON documents will always be based on definitions from this single system. Thus, one can form the SQL query of interest in the interface of the data processing system. In this case, it always remains possible to access the ontology repository to refine or supplement the terms of the query using the **SPARQL** query (Figure 5). Then the SQL-request coming from the user interface initiates the work of the Spark SQL module. As a result of the work of Spark SQL module, **RDD** or **DataFrame** sets are created, including the selected records, which can be processed by the system's service functions for further use. In fact, the user's work consists of two phases: viewing ontologies terms with the choice of adequate for the formation of SQL-query; access to the repository of

processed data with a SQL query. Thus, the main scenario involves unifying heterogeneous data by converting them to JSON documents and processing them using **Spark SQL**. Other scenarios are also justified, if we take into account the diversity of the source data. For example, the **Spark SQL** module allows direct query to relational databases without their conversion to the JSON format. On the other hand, you can provide access to JSON documents by collecting them in a file system using other Big Data tools. The first and main feature of JSON data collection based on ontological models terms – the unambiguous interpretation of the content and type of data. In this case users and external programs can freely work with data, because the ontology term, mapped to a key or value in the body of the JSON file, has available and accepted definitions and properties. For example, links to various types of files (graphics, multimedia, exe-files, etc.) can be described adequately and functionally using keys-terms from ontologies describing data formats. The second feature, as it is not strange, is the possibility of including in the exchange of such data sources that do not allow active access or changes due to various reasons. Then uploading the data to an external JSON file solves this problem, providing independent data storage and their full description via ontological models.

The listed technologies, supported by **Apache Spark**, provide unlimited productivity and variety of opportunities to handle complex data, which include data on the properties of substances, including traditional materials and nanostructures.

References

[1] Ataeva, O.M., Erkimbaev, A.O., Zitserman, V.Yu. et al.: Ontological Modeling as a Means of Integration Data on Substances Thermophysical Properties. Proc. of 15th All-Russian Science Conference "Electronic Libraries: Advanced Approaches and Technologies, Electronic Collections" – RCDL-2013, Yaroslavl, Russia, October 14–17, 2013. http://rcdl.ru/doc/2013/paper/s1_3.pdf

[2] Introduction to JSON. http://json.org/json-ru.html

[3] 3Vs (volume, variety and velocity), definition from TechTarget Network. http://whatis.techtarget.com/definition/3Vs

[4] Apache Hadoop. https://hadoop.apache.org/

[5] Apache Spark. http://spark.apache.org/docs/

[6] ChemSpider. www.chemspider.com

[7] Frenkel, M., Chirico, R.D., Diky V. et al.: XML-based IUPAC Standard for Experimental, Predicted, and Critically Evaluated Thermodynamic Property Data Storage and Capture (ThermoML) (IUPAC Recommendations 2006). Pure Appl. Chem., 78 (3), pp. 541-612 (2006)

[8] Hall, S.R, McMahon, B.: The Implementation and Evolution of STAR/CIF Ontologies: Interoperability and Preservation of Structured Data. Data Science J., 15 (3), pp. 1-15 (2016). doi: http://dx.doi.org/10.5334/dsj-2016-003

[9] Kaufman, J.G., Begley, E.F.: MatML. A Data Interchange Markup Language. Advanced Materials & Processes/November, pp. 35-36 (2003)

[10] Michel, K., Meredig, B.: Beyond Bulk Single Crystals: A Data Format for all Materials Structure-property-processing Relationships. MRS Bulletin. 41 (8), pp. 617-623 (2016)

[11] Ontobee: A Linked Data Server Designed for Ontologies. www.ontobee.org