Development of a knowledge system for Big Data: Case study to plant phenotyping data

Luyen Le Ngoc, Anne Tireau, Aravind Venkatesan, Pascal Neveu, Pierre Larmande

To cite this version:

Luyen Le Ngoc, Anne Tireau, Aravind Venkatesan, Pascal Neveu, Pierre Larmande. Development of a knowledge system for Big Data: Case study to plant phenotyping data. WIMS: Web Intelligence, Mining and Semantics, Mines Ales, Jun 2016, Nimes, France. pp.1-9, 10.1145/2912845.2912869 . hal-01411565

HAL Id: hal-01411565
https://hal.archives-ouvertes.fr/hal-01411565

Submitted on 7 Dec 2016

Development of a knowledge system for Big Data: Case study to plant phenotyping data

LE Ngoc Luyen[*†]
luyenln@dlu.edu.vn

Anne TIREAU[‡]
anne.tireau@supagro.inra.fr

Aravind VENKATESAN[†]
Aravind.Venkatesan@lirmm.fr

Pascal NEVEU[‡]
pascal.neveu@supagro.inra.fr

Pierre LARMANDE[†§◊]
pierre.larmande@ird.fr

ABSTRACT

In the recent years, the data deluge in many areas of scientific research brings challenges in the treatment and improvement of agricultural data. Research in bioinformatics field does not outside this trend. This paper presents some approaches aiming to solve the Big Data problem by combining the increase in semantic search capacity on existing data in the plant research laboratories. This helps us to strengthen user experiments on the data obtained in this research by infering new knowledge. To achieve this, there exist several approaches having different characteristics and using different platforms. Nevertheless, we can summarize it in two main directions: the query re-writing and data transformation to RDF graphs. In reality, we can solve the problem from origin of increasing capacity on semantic data with triplets. Thus, data transformation to RDF graphs direction was chosen to work on the practical part. However, the synchronization data in the same format is required before processing the triplets because our current data are heterogeneous. The data obtained for triplets are larger that regular triplestores could manage. So we evaluate some of them thus we can compare the benefits and drawbacks of each and choose the best system for our problem.

[*] Dalat University, Vietnam

[†] Institut de Biologie Computationnelle, LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier), France

[§] INRIA Zenith, Montpellier, France

[‡] INRA (Institut National de la Recherche Agronomique) - Mistea Research Laboratory, Montpellier, France

[◊] IRD (Institut de Recherche pour le Développement), UMR DIADE, Montpellier, France

Keywords

Knowledge base; Ontology; Reasoning; Inference; SPARQL; xR2RML; Benchmark; NoSQL; Big Data; Triplestore

WIMS '16, June 13-15, 2016, Nîmes, France
© 2016 ACM. ISBN 978-1-4503-4056-4/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2912845.2912869

1. INTRODUCTION

Research in agronomy aims to address challenges in improving crop productivity, resistance to diseases, minimize the impact of drastic climatic variations. With the vast amounts of data available today, there is potential for scientists to make use these data to understand important patterns and generate broad generalities. However, the data are highly distributed and diverse and with the exponential increase in the amount of data, there is an urgent need for efficient data integration and management. In this context, scientists must be aided to manage their own experimental data in the backdrop of prior knowledge to building consesus within the research community. The Phenome project[1] was conserved at the INRA computational sciences unit[2] in Montpellier, France, to efficiently manage large volumes of heterogeneous data, assessing environmental influences on plants.

This paper focuses on exploring the possibilities to, a) to store large volumes of heterogeneous data(relational and NoSQL databases); b) strengthen the ability to search different datasets by improving machine interoperability. To this end, the Semantic Web offers a solution to formally structure the data, in this way allowing the generation of new knowledge through inference rules. On the other hand, the latest big data management solutions such as MongoDB, caters to the needs of handling large amounts of data with optimized performance. Taken together, our effort explores the following aspects:

- Exploiting existing big data solutions to make semantic searches over large data.
- We propose a solution to harmonize data using RDF triple store technology.
- Benchmarking triplestores (4Store, Jena Fuseki, Virtuoso, Sesame, Stardog, GraphDB) on: data loading, data search and on inferencing.

The paper is organized as follows: in the next section, we review the state-of-the-art in big data and semantic web. Section 3 elaborates on the proposed solution with the general model and data transformation to triples. Section 4 describes our approach towards benchmarking. The final section provides the conclusion and perspectives.

[1] https://www.phenome-fppn.fr/

[2] http://www6.montpellier.inra.fr/mistea_eng

2. RELATED WORKS

2.1 Big Data context

Today we are in the age of Big Data posing challenges in data storage, analysis and visualization. There is a need to develop solutions to manage large amounts of data on a daily basis and extract new knowledge from them. The life sciences are no exception to this, further, the advances made in high throughput technologies at various levels (i.e genotypic, phenotypic, metabolomic, etc.) makes data interoperability essential, calling for structuring data to bridge the gap between the data and the knowledge it represents (common vocabulary, ontologies, rules, etc).

Phenomics aims at understanding of the complexity of interactions between plants and environments in order to accelerate the discovery of new genes and traits and optimize the use of genetic diversity under different environments. Conventional methods to quantify these interactions rely on a limited number of multi-environment evaluations of plants. Advances in genomics and high-throughput phenotyping tools provide a unique opportunity to discover new genes targeted at each interactions. Germplasm can now be sequenced at low cost to reveal genetic diversity in fine molecular detail. Missing are multi-environment and multitrait phenotypic data to rapidly discover gene-phenotype relationships.

The Phenome project handles a large amount of (observational)data (approximately 40 Tbytes in 2013, 100 Tbytes in 2014 and over 200 TBytes in 2015) at different levels (field, microplot, plant, organ, cell, etc). To this, the traditional relational databases lacks the capacity to handle such larg dynamic data. Often these systems become heavy and slow and don't provide required flexibility. To this end, the advent of the so called NoSQL database technologies offer several alternative options to organise and store data (see Table 1).

Type	Examples of this type
Key-values Store	CouchDB, Oracle NoSQL Database, Dynamo, FoundationDB, HyperDex, MemcacheDB, Redis, Riak, FairCom c-treeACE, Aerospike, OrientDB, MUMPS
Wide Column Store	Accumulo, Cassandra, Druid, HBase, Vertica
Document Store	MongoDB, Clusterpoint, Apache CouchDB, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, OrientDB, Qizx
Graph	Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog
Multi-model	OrientDB, FoundationDB, ArangoDB, Alchemy Database, CortexDB

Table 1: Database management Systems in NoSQL

2.2 Semantic Web context

An important aspect in efficient data management is making data interoperable enabling the of extract meaningful information from large datasets. To this end, the Semantic Web technologies proposed by Tim Berners Lee et al., [5] [6], offers a remedy to interlink highly distributed data and enhancing data interoperability, to share and reuse data between multiple applications and help users to extract new knowledge.

2.3 Overview of existing solutions

The first solution is based on the association between MongoDB and AllegroGraph - a document store and a graph database system. AllegroGraph provides an interface to MongoDB, called MongoGraph. This tools allows the automatic conversion of JSON objects to RDF triples and can be query by both MongoDB query language and SPARQL. With these characteristics, it is possible to develop a system that combines semantic web features with MongoDB to manage voluminous data. Figure 1 shows the architecture of such this system. Here, the experimental data remains stored in MongoDB under document format. Triples linking MongoDB documents to an ontology are imported into AllegroGraph. We use the xR2RML tool to automatically map elements of collections with the ontology[12]. This ontology allows inference by exploiting the relationship between triples. Thus the inference engine can create new relationships based on concepts defined in the ontology.

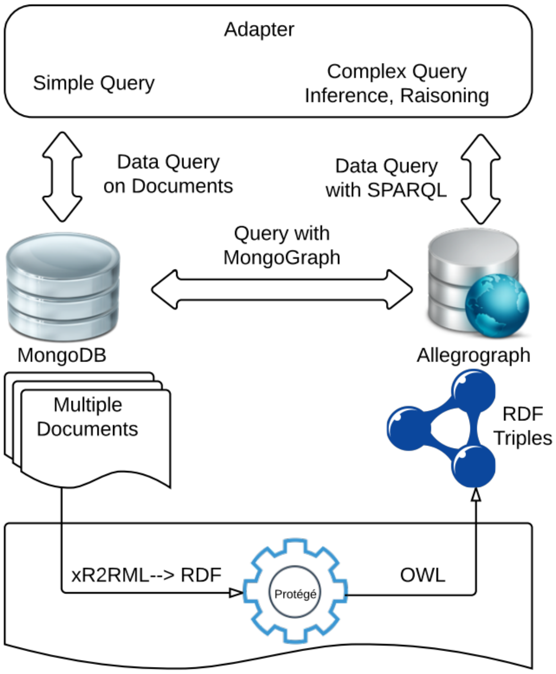

Figure 1: The components in a system MongoGraph

Advantages

- AllegroGraph allows inferences on large data
- Property based filtering to reduce the number of triples in the database.
- Massive database management with MongoDB

Disadvantages

- A more complex system with multiple stage requests
- No synchronization between the two databases.

In the second solution, we present Neo4j which is used to represent the data as graph. Neo4j has optimal components that come in kernel supplement[1]. One can structure the graph via a meta-model, getting an implementation of RDF triplestore compliant SPARQL[9]. For example, with two plugins Neo-rdf-sail [3] and Neo4j sparql-and-extension [4].

Advantages

- A graph representation of Big Data improves the performance of the system by queries based on the relationships between the objects.
- The graph representation is similar to that of ontologies and RDF data instances.

Disadvantages

- The data must be re-organized in the form of a graph, which is time consuming, depending on the complexity and size of the data.
- The data are not directly represented in RDF, thus to query with SPARQL, one needs to deploy an integrated plugin that support SPARQL.

The third solution is based on the data organization for linked data which publish and use information on the Web[3]. JSON-LD is an ideal data format for programming environments such as, REST Web Service, as MongoDB or CouchDB. JSON-LD document is both a serialization of RDF and a JSON document. The RDF data format is serialized in JSON-LD is complimentary to the JSON format used in MongoDB. So we can exploit the power of MongoDB in handling big data and the flexibility provided by RDF. Moreover, we facilitate the serialization of graphs in MongoDB, graph data can be organized and stored in memory by using APIs such as Jena or Sesame. Such APIs allow the use of SPARQL to query and infer the new knowledge. The semantic searches will be made directly on the RDF graphs that are serialized from MongoDB. In practice this setup is time consuming. Therefore, to overcome this limitation, we developed a method to organize indexes for the data and optimize of the execution time. Figure 2 shows CRUD (Create, Read, Update, Delete) operations that run on MongoDB and semantic search conducted over RDF graphs. Thus, a middle layer is necessary to synchronize the two databases.

Advantages

- JSON-LD is both an RDF serialization format and the storage model for MongoDB.
- Operations CRUD will be quickly performed on the data in MongoDB.
- SPARQL is used to query triplets in memory.

Disadvantages

- The existence of two database will increase the system complexity.
- Loading of RDF graphs in memory will be time consuming. The updated data on RDF graphs are dependent on MongoDB.

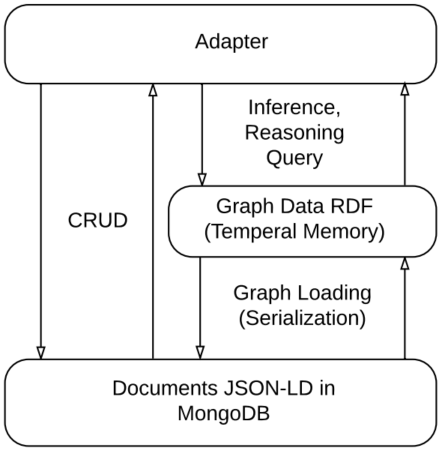

Figure 2: The components in an associated system of MongoDB and JSON-LD

- Large graphs cause memory issues, thus material capacities need large buffer memories.

The fourth solution is based the concept of the Ontology-Based Data Access (ODBA). ODBA is a paradigm for accessing and integrating data, the key idea is to resort a three-level architecture, constituted by the ontology, the data, and the mapping between two[2]. In the OBDA paradigm, an ontology defines a high-level global schema and provides a vocabulary for user queries, thus isolating the user from the details of the structure of data sources (which can be relational databases, triple stores, datalog engines, etc.). The OBDA system transforms user queries into the vocabulary of the data and then delegates the actual query evaluation to the source data.

An OBDA system is a triple [4]
$\mathcal{O} = <\mathcal{T}, \mathcal{S}, \mathcal{M}>$, Where:

- $\mathcal{T}$ is the intensional level of an ontology. We consider ontologies formalized in description logics (DLs), hence $\mathcal{T}$ is a DL TBox.
- $\mathcal{S}$ is a (federated) relational database representing the source data.
- $\mathcal{M}$ is a set of mapping assertions, each one of the form $\Phi(x) \leftarrow \Psi(x)$

  Where:

  - $\Phi(x)$ is a query over $\mathcal{S}$, returning tuples of values for x
  - $\Psi(x)$ is a query over $\mathcal{T}$ whose free variables are from x

Based on the definition of the ODBA, researchers at the University Bozen-Bolzano in Italy have developed a Framework named Ontop. It is currently used on the application Optical [5]. It aims to solve the problems of association of Big Data and semantic search capacities. The Ontop core is the query engine SPARQL QUEST that implements RDFS and OWL 2 QL by re-writing the SPARQL queries into query

[3] https://github.com/neo4j-contrib/neo4j-rdf-sail
[4] https://github.com/niclashoyer/neo4j-sparql-extension

[5] http://optique-project.eu/

SQL. Ontop is able to efficiently generate and optimized SQL queries[13].

Advantages

- The data structure is stored in the database management system (no duplication, realtime data access).
- Queries are performed on the data in SPARQL.
- The compatibility with multiple relational database systems.

Disadvantages

- System complexity will increase with the organization of ODBA models.
- The increase in time and cost to build the system.
- Applied only for relational databases, no applications avialble for NoSQL systems.

In summary, in all of the above approaches, data is organized and stored in the graph oriented database management systems (Neo4j) or document oriented database systems (MongoDB), hybrid systems (MongoDB and Triple store technology) and using ODBA approach. The solutions are broadly divided in two: query re-writing and data transformation to RDF graphs. We can see that for each approach there are advantages and disadvantages. Choosing the best solution for the data, will depend on the use and the underlying questions.

3. PROPOSED SOLUTION

3.1 General Model

We consider a solution using the RDF data model. Here, the current heterogeneous data will be converted in RDF. This approach is at this moment the optimal solution for data organization with semantic search capabilities. Most often original data that needs to be exposed as RDF already exists in relational databases. There are several methods to translate relational database to RDF but the most used is the following: D2R [6]. D2R operates with a mapping file and one or several ontologies. The mapping file is used to make the connection between the tables, attributes on one hand and classes, properties on the other. Thus, after the mapping, data will be available via semantic web applications. There are now two additional methods: R2RML[7] and Direct Mapping [8]. Both approaches are only for relational databases. So there is a need to map RDF with document-oriented databases such as CSV, XML and JSON. Franck Michel and his colleages [12] used the mapping language R2RML and Morph-RDB [9] to develop xR2RML which is apply to documents oriented databases such as MongoDB. Precisely, xR2RML is an extension of the mapping language R2RML and relies on particular properties of the mapping language and RML[7]. R2RML covers the basic mapping relational data to RDF triples. RML extends R2RML to address mappings on heterogeneous data (XML, JSON, CSV) with RDF triples. xR2RML extends its scope to a wider range of basic non-relational data.

Advantages

- Data are translate in RDF triples.
- Questions are performed with the query language SPARQL
- Reasoning capacity are supported by these triplestores.

Disadvantages

- Data transformation step is time consuming (i.e. reorganizing data in graph)
- A new system architecture needs to be implemented with these data.
- We encounter performance problems with large graphs

Current systems able to store large volumes of data, refer mostly to NoSQL systems (eg: MongoDB). With a middleware layer, we are able to organize and synchronize the existent data like an intermediate storage for interacting with the system. Subsequently, the data will be converted into RDF triples with the use of language mapping xR2RML and the tool developed by the authors [12]. Vocabularies and triples transformation rules are provided by ontologies. Ontology are important to perform advanced search on relationships and existing hierarchies. There exist varions triplestores to manage RDF. In the paper, we will focus experiments with six systems: 4Store, Virtuoso, Stardog, GraphDB , Sesame and Jena Fuseki(TDB). Considering their storage capacities and indexing are differents, we will test these systems with large RDF graphs. Thus we will test them based on the data management capabilities and data retrieval using the SPARQL query language. We will develop a search engine in order to use inference capacities. An access point will be provided to perform queries.

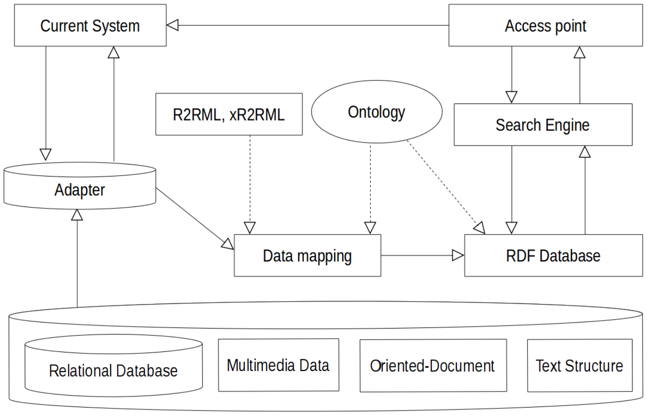

Figure 3: General model of the system

3.2 Data transformation to triples

Plant phenome research creates various heterogeneous data associated with observational process such as data imaging, watering, weighting process. Now, the majority of the data exists in the form of the documents JSON which are stored in MongoDB. This section will exploit stages of the RDF translation with xR2RML. Thus, a triples map specifies a rule

[6] http://d2rq.org/

[7] http://www.w3.org/TR/r2rml/

[8] http://www.w3.org/TR/rdb-direct-mapping/

[9] https://github.com/oeg-upm/morph-rdb is an implementation of mapping R2RML language for relational databases

for translating data elements of a logical source. The RDF triples generated from a data element in the logical source share all the same subject. An xR2RML triples map is represented by a resource that references the *xrr:logicalSource* property. It means that a logical source specifies a table mapped to triples. A subject map specifies how to generate a subject for each data element(row, document, set of XML elements, etc). A subject may be specified using the *rr:subjectMap* property, where value must be the subject map, or the constant shortcut property *rr:subject*, and zero or to many *rr:predicateObjectMap* properties, where values must be the predicate object maps. They specify pairs of predicate maps and object maps that, along with the subjects from one or many RDF triples.

```
@prefix xrr: <http://i3s.unice.fr/xr2rml#>.
@prefix rr:  <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax
    -ns#>.
@prefix f: <http://www.franz.com/> .
@prefix ia: <http://www.mistea.supagro.inra.fr/
    ontologies/2015/03/imageAnnotation#>.
 <#Image> a rr:TriplesMap;
  xrr:logicalSource [
   xrr:query """db.image.find({ 'configuration.imgid'
        :  {$exists: true} } )""";
  ];
  rr:subjectMap [
   rr:template "{$.uri}";
   rr:class ia:Image;
  ];
  rr:predicateObjectMap [
   rr:predicate ia:aboutEntity;
   rr:objectMap [ xrr:reference "$.context.plant"; ];
  ];
  rr:predicateObjectMap [
   rr:predicate ia:timeStamp;
       rr:objectMap [
               xrr:reference "$.date";
               rr:datatype xsd:date;
       ];
  ];
  rr:predicateObjectMap [
   rr:predicate ia:hasFileName;
       rr:objectMap [ xrr:reference "$.fileName"; ];
       rr:datatype xsd:string;
  ];
  rr:predicateObjectMap [
       rr:predicate ia:hasPlateau;
       rr:objectMap [ xrr:reference "$.context.
           technicalPlateau"; ];
       rr:class ia:TechnicalPlateau;
  ];
  rr:predicateObjectMap [
       rr:predicate ia:inImagingCycle;
       rr:objectMap [ xrr:reference "$.configuration.
           taskid"; ];
       rr:datatype xsd:integer;
  ];
```

Figure 4: Mapping a json data to triples

4. EXPERIMENTS AND COMPARISONS

We obtained 45 millions of triples for image annotation from about 3.5 millions images in the MongoDB system. This transformation required a lot of server execution time (about 20 hours). These data exist in the form of a graph with multiple instances. We first aim at realizing experiments with these triples and then compare performances of the following triplestores: 4Store, Sesame, Jena, Virtuoso, GraphDB, Stardog. The data are divided into five groups of triples 100.000, 1 million, 10 millions, 20 millions and 40 millions of triples. Data from the first to the fourth groups are distinct, while the last group is a collection of all data. These groups will allow us to evaluate the performance of these Triplestores. Here, we focus on three performance criteria: loading, queries and reasoning.

These experiments were performed on a server Ubuntu Server operating system. Below we can see the detailed configuration of this system:

Processor	Intel Xeon(R) CPU L5420 @ 2.50GHz
Front side bus	1333 MHz
Cache L2	12M
Memory	32GB
Storage	160GB

Table 2: Configuration of the Server

4.1 Data loading

With large RDF files generated, the import test data in triple store will give us a particular view of data loading performances. Each system has a particular way of organizing data indexing which impacts the data loading mechanism. Some triplestores allow users to set various configurations such as fields indexes, the order of priority for the index, peak memory used, etc. In addition, there are systems that cannot directly load large files (e.g. with Sesame, Virtuoso). In these cases, a system has been set up specifically to split the files in smaller chunks. Other systems like Fuseki, Stardog and GraphDB provide tools to able load large files.

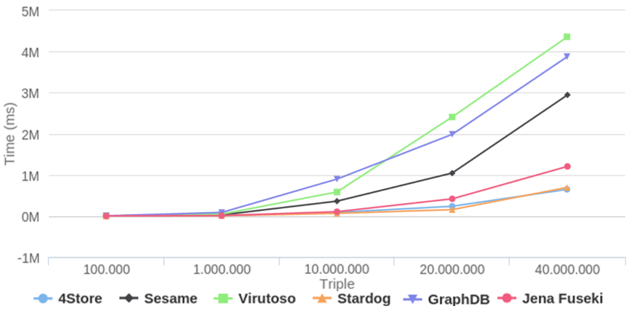

Figure 5: Comparison of loading times on Triple Stores

The results of benchmark data on load time are obtained with the best time to 4Store while the Virtuoso system is the slowest. We can explain through their differences in the structure of the index and data storage. In Virtuoso the import is performed in RDBMS tables using ODBC protocol, while in the case of 4Store import does not require processing for the storage because of its tree structure.

4.2 Data search

The most important part in a data management system is query performances. Experimentation queries allow us to evaluate in detail the system. That is why we set up the second benchmark to test the search capacity with these systems. To ensure an equality, all the queries are launched via a unique access point where several types of search have been defined.

Example query 1

In this application, we want to find the information of the image annotation with the date created, the type of shooting (beside and above), and the camera angle. For refining the results obtained, we used a filter on the camera angle with values greater than 300°or less than 100°.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ia: <http://www.mistea.supagro.inra.fr/ontologies/2015/03/imageAnnotation#>
SELECT ?Image ?Date ?ViewType ?hasCameraAngle WHERE {
 ?Image rdf:type ia:Image .
 ?Image ia:timeStamp ?Date .
 ?Image ia:hasViewType ?ViewType.
 ?Image ia:hasCameraAngle ?hasCameraAngle .
 FILTER (?hasCameraAngle < 100 || ?hasCameraAngle >300)
}
```

Figure 6: Example query 1

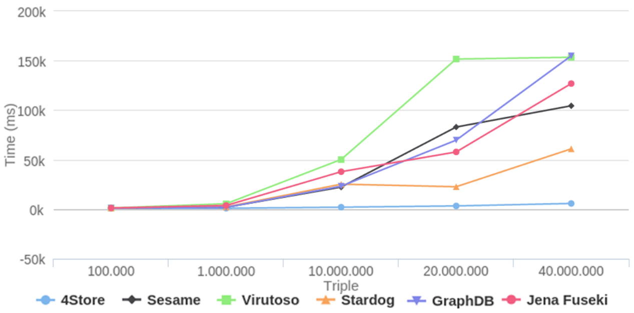

Figure 7: The evaluation of the query 1

For this query, the best result is obtained with 4Store while the worst is performed with Virtuoso. The difference of the execution time between the systems is great. In general, the systems have a linear increase over time of all data sets. In particular, for small sized data sets (100,000 and 1 million triples), the runtime system is not much different but it is significant with very large data sets.

Example query 2

The second request is constructed on the basis of the first with an additional part on the arrangement of the data obtained on the field angle of the camera and the created date of the image. This addition allows us to test the data search capacity and data grouping with the ORDER BY command.

In this case, there is no change in the ranking for Virtuoso, which still took a long time to execute this query. However, the system that gave the best result is Stardog. Like the previous query, all systems respond very well on small data sets. With 4Store, Sesame, Fuseki and GraphDB tools, the running time is close enough. This can be explained because they all have a tree form data organization while Virtuoso store it in relational tables.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ia: <http://www.mistea.supagro.inra.fr/ontologies/2015/03/imageAnnotation#>
SELECT ?Image ?Date ?ViewType ?hasCameraAngle WHERE {
 ?Image rdf:type ia:Image .
 ?Image ia:timeStamp ?Date .
 ?Image ia:hasViewType ?ViewType.
 ?Image ia:hasCameraAngle ?hasCameraAngle .
 FILTER (?hasCameraAngle < 100 || ?hasCameraAngle >300)
}
ORDER BY ?hasCameraAngle ?Date
```

Figure 8: Example query 2

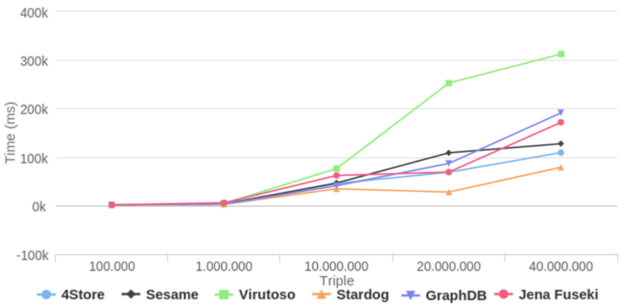

Figure 9: The evaluation of the query 2

Example query 3

This query test the search capacity of the image annotation with the date and camera angle by combining several different patterns using the UNION command. This allows us to extend the results obtained for other graphs of different values.

```
PREFIX ia: <http://www.mistea.supagro.inra.fr/ontologies/2015/03/imageAnnotation#>
SELECT ?Image ?Date ?hasCameraAngle WHERE {
{
  ?Image rdf:type ia:Image . ?Image ia:hasCameraAngle ?hasCameraAngle .
  FILTER (?hasCameraAngle < 100)
 } UNION {
  ?Image rdf:type ia:Image . ?Image ia:hasCameraAngle ?hasCameraAngle .
  FILTER (?hasCameraAngle > 200)
 }
}
```

Figure 10: Example query 3

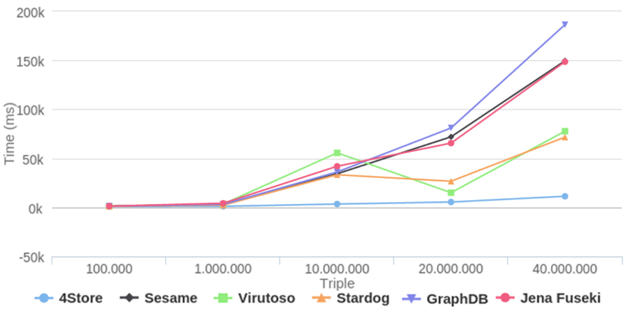

Figure 11: The evaluation of the query 3

In this query, 4Store is the best tool with the fastest query execution. On the contrary, GraphDB system gives the longest execution time, followed by Sesame and Jena. We can also see that there is an irregularity in the execution time with the two data sets 10 and 20 millions triples clearly illustrated by Virtuoso and Stardog. This difference is explained by the fact that there are two distinct sets of data in the assessment.

Example query 4

In the last query, we counted the number of triples in the systems used by the COUNT command. We used a filter on the type of view and angle of the camera to limit triples numbers in the result.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ia: <http://www.mistea.supagro.inra.fr/ontologies/2015/03/imageAnnotation#>
SELECT (count(?Image) as ?ima) WHERE{
 ?Image rdf:type ia:Image .
 ?Image ia:hasViewType ?hasViewType .
 ?Image ia:hasCameraAngle ?hasCameraAngle .
 FILTER (?hasViewType = "side" || ?hasViewType = "Side" && ?hasCameraAngle > 200 )
}
```

Figure 12: Example query 4

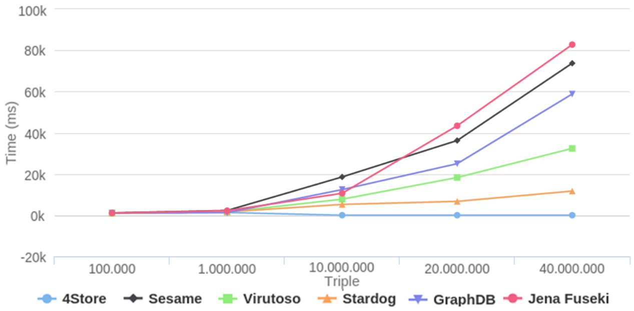

Figure 13: The evaluation of the query 4

There is an error on executing the 4Store tool with higher data sets to 10 million triples. We have included the value -1.0 to report this error. The best tool in this assessment is Stardog unlike Virtuoso who gets the longest running time.

4.3 Data inference

The first example of inference

This example was conducted to evaluate the reasoning on the relations of the properties that are defined in RDF Schema. Here, the relationship "*rdfs: subPropertyOf*" is used to presentation both "*comesFromPlateau*" and "*hasSource*" properties. So the query on the object "*Data*" can infer new information in "*Image*" and also "*TechnicalPlateau*" can be found in the object "*Source*".

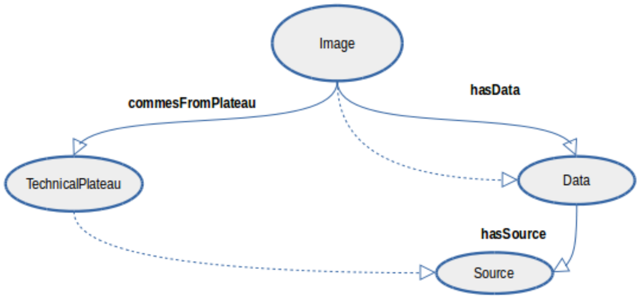

Figure 14: Inferred relationships ontology in the first example

Because the results obtained in these examples are very different between the triplestores, we use a logarithmic function to illustrate the values of the execution time. In general, we have good results with small sizes of data sets in all triplestores but different performances appear with large data sets. In this case, detailed results show that 4Store and Jena Fuseki are slower to perform inference. While GraphDB and Virutoso give the best execution time.

```
PREFIX : <http://www.mistea.supagro.inra.fr/ontologies/2015/03/imageAnnotation#>
SELECT ?data ?source ?hasViewType  WHERE {
 ?data :hasSource ?source .
 ?data :hasViewType ?hasViewType .
 FILTER regex(?hasViewType, "side","i")
}
```

Figure 15: The query SPARQL of the first example of inference

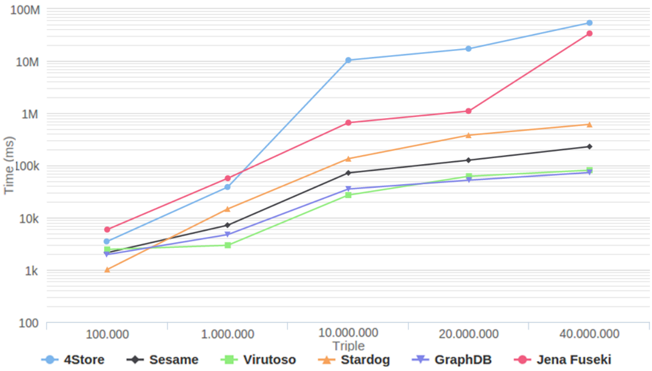

Figure 16: The execution time of the first inference

The second example of inference

In this example, we have continued to test the ability of inference in RDF Schema on the domain and rank of objects values. In fact, this inference uses the relationship we define as in the first example. However, the important point of this inference is based on specific data. We can see the detail in Figure 17

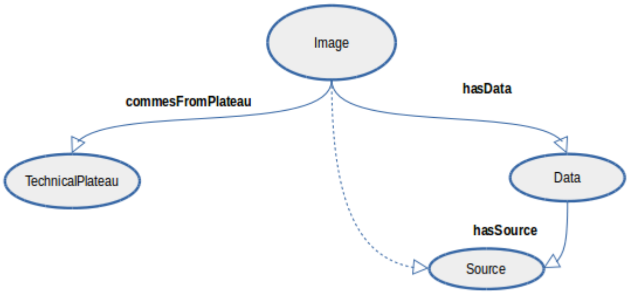

Figure 17: Inferred relationships ontology in the second example

This example allows us to confirm that the two triplestores 4Store and Jena Fuseki are very slow in executing inferences of large data sets. On the contrary, Stardog, GraphDB and Virtuoso get good execution time. Sesame in both example of inferences gets decent results for an OpenSource triplestore. In some cases, it gives better results than commercial ones.

4.4 Evaluations

This section gives an overview on implementation and per-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax
     -ns#>
PREFIX : <http://www.mistea.supagro.inra.fr/
     ontologies/2015/03/imageAnnotation#>
SELECT ?image ?Source ?hasViewType WHERE {
 ?image :hasSource ?Source .
 ?image rdf:type :Image .
 ?image :hasViewType ?hasViewType .
 FILTER (?Source="http://www.phenome-fppn.fr/m3p/
      phenoarch") .
 FILTER REGEX(?hasViewType, "top","i")
}
```

Figure 18: The query SPARQL of the second example of inference

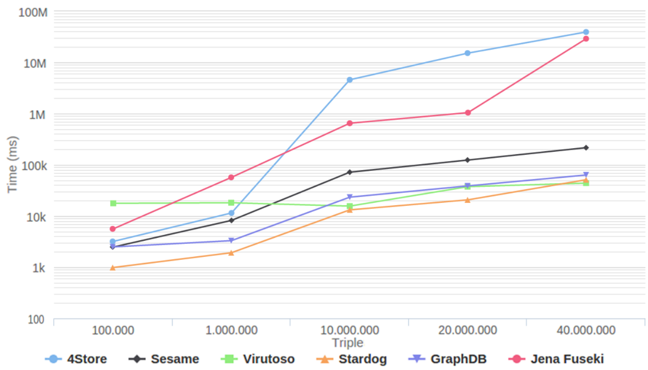

Figure 19: The execution time of the second inference

formance capacity of these systems. Each system has differences in the organization and indexation of triple data: Virtuoso uses tables as in the relational database, 4Store uses the structure of the radix tree [10], while Sesame, Jena Fuseki and GraphDB apply the structure B or B+ tree [14] [11]. These differences are important elements that impact their performances. Nevertheless, we must also consider the necessary features for a triplestore. The most important feature is the reasoning capacity with RDFS or OWL. Moreover, handle large datasets, distribute processes on multiple machines, can facilitate reasoning on large RDF graphs. In this context, the triplestores need to support management of distributed databases instead of having a network of multiple machines to communicate and exchange data [8] [15] [16] [18]. Above, we evaluate the three most important criteria for triplestores: loading, search and inference.

With 4Store, the advantages of indexation data with the radix tree provide a good system for loading and search data. It is always one of the best systems with the fastest execution time. In addition, the architecture of 4Store allows clustering of distributed data and used in several machines at the same time. However, the search function in 4Store is still perfectible, we can see errors in some cases (in the example query 4). In addition, the user interface has several limitations. But the biggest drawback is the reasoning support. In fact, the reasoning engine is an extended module named 4SR[17] which is a branch of the 4Store project, implemented for RDFS Backward Chained Reasoning. This module supports a very weak inference capacity with class and properties: *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain* et *rdfs:range* in RDFS. Choosing 4Store to build the system with large data volume will depend on the need of reasoning. If there is no need to infer the data, 4Store may be the right choice.

Sesame is one of the first systems that is used to manage RDF data. With this system, the results are average in the benchmarks on loading, search and inference data. These result are acceptable for an Opensource system. However, Sesame has disadvantages in large database management. Firstly, it can not be deployed in a cluster of machines with large distributed graphs, but rather allows to create federated databases with graphs which are completely distinct. Then the native RDF data model is optimized for medium sized graphs. Finally, the inference mechanism of Sesame creates many new triplets and increases the size of the database. This is manageable for medium sized graphs but reached its limit for large graphs.

Virtuoso is built on the architecture of relational database management system. This may explain its bad results on the data loading runtime and some examples of query search. On the contrary, Virtuoso has advantages in the data inferencing capacity. It can perform reasoning with data defined by RDFS or OWL. In this evaluation, Virtuoso is the best Opensource tool that completely supports essential components of a database management system, such as ACID transactions, security or the user interaction interface etc. In addition, it allows us to implement a system with a strong supporting reasoning. Finally, Virtuoso allows to deploy databases on multiple clusters of machines. However, this feature is only supported in the commercial version.

Jena Fuseki is developed on the basis of the Jena framework. It brings features of the first framework that is built for RDF data. Our benchmark is performed with the storage of the Jena TDB architecture and uses the indexing of the B + tree structure. In the evaluations, Jena shows good results in loading data. However, research data and inferences with Jena Fuseki took a long execution time. At this time, Jena can run on a cluster of several machines in different architectures (Some examples defined in this article [14]). Moreover, Jena provides APIs (Apache Jena Elephas) that allow to write embedded applications in Apache Hadoop. From the results we can say that Jena Fuseki suitable for RDF databases of medium size.

GraphDB Ontotext is built upon the Sesame framework to provide missing features of Sesame. The GraphDB improvements focus on reasoning ability, user interface and data cluster architecture. In nearly all evaluations, we can see that GraphDB gives a higher performance than Sesame in data search and inference. In fact, there is less difference in the indexing mechanism which explains the small difference in the execution time. With its inference engine, GraphDB supports reasoning in RDFS and OWL. Finally, it is possible to manage large volumes of RDF data.

Stagdog gives impressive results compared to the criteria: Data Loading, Data Search and Data Inference. It is always in the best tools that are the most effective. For the reasoning, it supports inferences in RDFS and OWL. Moreover it is build for cluster performances at a high level. We can say that Stardog is the best tool in the list of all the tools tested for our Benchmark. However, it is available only in commercial version.

5. CONCLUSION AND FUTURE WORKS

The efficient data management is increasingly important in conducting biological reseach. The search for optimal data management solution can help reduce time and increase the performance of database systems. The problems we encountered during this effort is the size of the data. Traditional methods have limitations in managing large amounts of data. In addition, the need to interlink and reuse data leads us to adopted methods to structure these data. There are different solutions for this problem. However, each solution described is a combination of one or more technologies. In general, we can summarize these solutions in two directions: the transformation of queries (or query rewriting) and the translation of data in RDF triples (or materialization). Each of approach has particular advantages and disadvantages. To this end we have chosen data conversion to RDF. This choice enabled us to facilitate the research on semantic data. We had to define new data models to unified and transform the experimental data in RDF. We evaluated several triplestores (4Store, Sesame, Virtuoso, Jena Fuseki, GraphDB Stardog) and carry out benchmarking based on different criterion: data loading capabilities, query search and inference on data.

The conversion of data to RDF triples is suitable to increase the semantic search capability on the data, such as performing inference tasks to extract implicit facts. However, this approach still has drawbacks including the management of large data, because until now, only few triplestores can support large volumes of data. We believe that in the future, work on query rewriting approaches (NoSQL-SPARQL) will help us to compare the advantages and disadvantages of both approaches.

6. REFERENCES

[1] R. Angles, A. Prat-Perez, and D. Dominguez-Sal. Benchmarking database systems for social network applications. *First International Workshop on Graph Data Management Experiences and Systems*, 2013.

[2] N. Antonioli, F. Castano, C. Civili, and S. Coletta. Ontology-based data access: The experience at the italian department of treasury. *Proceedings of the Industrial Track of the Conference on Advanced Information Systems Engineering 2013*, pages 9–16, 2013.

[3] C. Aswamenakul, M. Buranarach, and K. R. Saikaew. A review and design of framework for storing and querying rdf data using nosql database. *4th Joint International Semantic Technology Conference*, pages 144–147, 2014.

[4] T. Bagosi, D. Calvanese, J. Hardi, and S. Komla-Ebri. The ontop framework for ontology based data access. *The Semantic Web and Web Science - ISWC 2014*, pages 67–77, 2014.

[5] T. Berners-Lee. "the semantic web", scientific american magazine. 2001.

[6] T. Berners-Lee, Fischetti, and Mark. Weaving the web. 1999.

[7] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Proceedings of the 7th Workshop on Linked Data on the Web (LDOW2014)*, 2014.

[8] I. Filali, F. Bongiovanni, F. Huet, and F. Baude. A survey of structured p2p system for rdf data storage and retrieval. *Transactions on Large-Scale Data and Knowledge Centered Systems III*, pages 20–55, 2011.

[9] A. Flores, G. Palma, and M.-E. Vidal. Graphium: Visualizing performance of graph and rdf engines on linked data. *International Semantic Web Conference 2013*, pages 133–136, 2013.

[10] S. Harris, N. Lamb, and N. Shadbolt. 4store: The design and implementation of a clustered rdf store. *The 5th International Workshop onScalable Semantic Web Knowledge BaseSystems (SSWS2009)*, 2009.

[11] M. Hepp, P. De Leenheer, A. De Moor, and Y. Sure. *Ontology Management: Semantic Web, Semantic Web Services and Business Applications*, chapter 4: Ontology Reasoning with large data repositories. Springer, 2008.

[12] F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat. Translation of relational and non-relational databases into rdf with xr2rml. *In Proceeding of the WebIST 2015 conference*, pages 43–54, 2015.

[13] R. muro Mariano, R. Diego, S. Mindaugas, B. Timea, and D. Calvanese. Evaluating sparql-to-sql translation in ontop. *CEUR Workshop Proceedings*, pages 94–100, 2013.

[14] A. Owens, A. Seaborne, and N. Gibbins. Clustered tdb: A clustered triple store for jena. 2009.

[15] N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras, and N. Kowiris. H2rdf+: High-performance distributed joins over large-scale rdf graphs. *Big Data, 2013 IEEE International Conference on*, pages 255–263, 2013.

[16] R. Punnoose, A. Crainiceanu, and D. Rapp. Rya: A scalable rdf triple store for the clouds. *Proceedings of the 1st International Workshop on Cloud Intelligence*, 2012.

[17] M. Salvadores, G. Correndo, S. Harris, N. Gibbins, and N. Shadbolt. 4sr - scalable decentralized rdfs backward chained reasoning. *9th International Semantic Web Conference ISWC 2010*, pages 137–140, 2010.

[18] B. Wu, Y. Zhou, P. Yuan, H. Jin, and L. Liu. Semstore: A semantic-preserving distributed rdf triple store. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 509–518, 2014.