Emerald Insight

International Journal of Web Information Systems

A semantic integration approach to publish and retrieve ecological data
Ana Maria de Carvalho Moura Fabio Porto Vania Vidal Regis Pires Magalhães Macedo Maia Maira Poltosi Daniele Palazzi

Article information:

To cite this document:
Ana Maria de Carvalho Moura Fabio Porto Vania Vidal Regis Pires Magalhães Macedo Maia Maira Poltosi Daniele Palazzi , (2015),"A semantic integration approach to publish and retrieve ecological data", International Journal of Web Information Systems, Vol. 11 Iss 1 pp. 87 - 119
Permanent link to this document:
http://dx.doi.org/10.1108/IJWIS-08-2014-0028

Downloaded on: 01 November 2016, At: 23:03 (PT)
References: this document contains references to 41 other documents.
To copy this document: permissions@emeraldinsight.com
The fulltext of this document has been downloaded 297 times since 2015*

Users who downloaded this article also downloaded:

(2008),"Dragon wine: developments in the Chinese wine industry", International Journal of Wine Business Research, Vol. 20 Iss 3 pp. 244-259 http://dx.doi.org/10.1108/17511060810901055

(2014),"Highly Adaptable but Not Invulnerable: Necessary and Facilitating Conditions for Research in Evolutionary Developmental Biology", Research in the Sociology of Organizations, Vol. 42 pp. 235-265 http://dx.doi.org/10.1108/S0733-558X20140000042008

Access to this document was granted through an Emerald subscription provided by emerald-srm:563821 []

For Authors

If you would like to write for this, or any other Emerald publication, then please use our Emerald for Authors service information about how to choose which publication to write for and submission guidelines are available for all. Please visit www.emeraldinsight.com/authors for more information.

About Emerald www.emeraldinsight.com

Emerald is a global publisher linking research and practice to the benefit of society. The company manages a portfolio of more than 290 journals and over 2,350 books and book series volumes, as well as providing an extensive range of online products and additional customer resources and services.

Emerald is both COUNTER 4 and TRANSFER compliant. The organization is a partner of the Committee on Publication Ethics (COPE) and also works with Portico and the LOCKSS initiative for digital archive preservation.

*Related content and download information correct at time of download.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

The current issue and full text archive of this journal is available on Emerald Insight at:
www.emeraldinsight.com/1744-0084.htm

A semantic integration approach to publish and retrieve ecological data

Ana Maria de Carvalho Moura and Fabio Porto
Extreme Data Laboratory (DEXL), National Laboratory of Scientific Computing (LNCC), Petrópolis, Brazil
Vania Vidal, Regis Pires Magalhães and Macedo Maia
Department of Computing, Federal University of Ceará, Fortaleza, Brazil, and
Maira Poltosi and Daniele Palazzi
Extreme Data Laboratory (DEXL), National Laboratory of Scientific Computing (LNCC), Petrópolis, Brazil

Received 2 August 2014
Revised 12 November 2014
Accepted 20 November 2014

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Abstract

Purpose – The purpose of this paper is to present a four-level architecture that aims at integrating, publishing and retrieving ecological data making use of linked data (LD). It allows scientists to explore taxonomical, spatial and temporal ecological information, access trophic chain relations between species and complement this information with other data sets published on the Web of data. The development of ecological information repositories is a crucial step to organize and catalog natural reserves. However, they present some challenges regarding their effectiveness to provide a shared and global view of biodiversity data, such as data heterogeneity, lack of metadata standardization and data interoperability. LD rose as an interesting technology to solve some of these challenges.

Design/methodology/approach – Ecological data, which is produced and collected from different media resources, is stored in distinct relational databases and published as RDF triples, using a relational-Resource Description Format mapping language. An application ontology reflects a global view of these datasets and share with them the same vocabulary. Scientists specify their data views by selecting their objects of interest in a friendly way. A data view is internally represented as an algebraic scientific workflow that applies data transformation operations to integrate data sources.

Findings – Despite of years of investment, data integration continues offering scientists challenges in obtaining consolidated data views of a large number of heterogeneous scientific data sources. The semantic integration approach presented in this paper simplifies this process both in terms of mappings and query answering through data views.

Social implications – This work provides knowledge about the Guanabara Bay ecosystem, as well as to be a source of answers to the anthropic and climatic impacts on the bay ecosystem. Additionally, this work will enable evaluating the adequacy of actions that are being taken to clean up Guanabara Bay, regarding the marine ecology.

Originality/value – Mapping complexity is traded by the process of generating the exported ontology. The approach reduces the problem of integration to that of mappings between homogeneous ontologies. As a byproduct, data views are easily rewritten into queries over data sources. The

This work has been partially supported by CNPq through its Institutional Capacity Program (Proc. 382.489/09-8) and Productivity Research Fellowship (Proc. 309502/2009-8) and FAPERJ through a research project (E-26/111.147/2011).

International Journal of Web Information Systems
Vol. 11 No. 1, 2015
pp. 87-119
© Emerald Group Publishing Limited
1744-0084
DOI 10.1108/IJWIS-08-2014-0028

architecture is general and although applied to the ecological context, it can be extended to other domains.

Keywords Web semantics architectures, Applications and standards, Metadata and ontologies, Web data integration

Paper type Research paper

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

1. Introduction

Organizing and cataloguing the existing ecological (biotic and abiotic) resources became crucial to enable a more accurate control and efficient management of natural reserves on the planet.

There are currently a number of initiatives, such as Global Biodiversity Information Facility (GBIF)[1], associated with the development of ecological information systems in support of some of these challenges. Some of them provide software artifacts to support scientists in different tasks such as: powerful interfaces for describing and publishing ecological metadata, and for defining domain terminology making use of ontologies (Gruber, 1995), so that they can be shared and used by the scientific community (Wieczorek *et al.*, 2012; Leinfelder *et al.*, 2010). Ontologies can be defined as a hierarchy of concepts using subsumption relationships (as in taxonomies) and axioms that are added to express relationships among concepts and to limit their intentional interpretations (Guarino, 1998). Additionally, axioms make ontologies more expressive by allowing the use of inference mechanisms.

Despite of years of research on data integration, interoperability is still considered a major challenge, as it is essential to support queries providing a consolidated data view from different heterogeneous repositories. In this context, the use of taxonomies and conceptual abstractions (hierarchies, aggregations, constraints, etc.) provided by ontologies, together with a new technology approach, known as linked data (LD), have helped to overcome this challenge. When used together, ontologies and LD rise as a good strategy to integrate data. LD has been proposed to enable data sharing and reuse on a massive scale (Berners-Lee, 2006; Heath and Bizer, 2011). It permits publishing and interlinking structured data on the Web, by the use of a standard language, the RDF (Manola and Miller, 2004). Among the many LD forums, this technology has gained significant uptake in life sciences, as mentioned by Heath and Bizer (2011). It has enabled the connection of a large number of datasets, in a diverse range of science domains such as: geography, biology, etc.

Although being generic, this architecture has been applied in the context of ecology to integrate ecological resources of an important Brazilian project, the Brazilian Long-Term Ecological Research Program (PELD/Brazil)[2], which is currently in development. Some of the main goals of this project are: to leverage ecological knowledge, so that important data can be provided to help and to reinforce government decision-making; to support research related to the management of natural resources; and to share this information among different sectors of society. The PELD project currently includes 29 collection sites, which are distributed along different Brazilian biomes, including fauna, flora, hydrology, forests, fish species, etc. Its main purpose is to collect and organize biodiversity resources from many regions in the country, and to learn about the functioning of ecosystems. Additionally, this initiative aims to provide researchers and society with an integrated view of the Brazilian ecological position.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

In the context of the ideas above, we use the Guanabara PELD[3] as a proof of concept. It aims to provide knowledge about the Guanabara Bay ecosystem, as well as to be a source of answers to the anthropic and climatic impacts on the bay ecosystem. Additionally, it will enable the evaluation of the adequacy of actions that are being taken to clean up Guanabara Bay, regarding the marine ecology.

However, integrating and publishing all the data produced by these groups is crucial not only to provide a homogeneous view of this data but also to make it available to other groups working in other PELDs throughout the country. When analyzing other projects in the literature, such as Daltio and Bauzer (2008), Patton *et al.* (2014) and GBIF, we observe different strategies for integrating ecological data. Even though some of them are specifically oriented to the application domain, while others have more generic purposes, they all share the same approach: the use of ontologies and/or metadata standards and domain vocabularies to describe the data resources since their origins.

This scenario is quite different from what happens in PELD. For example, Guanabara PELD is developed by a large group of biologists, responsible for distinct domains (hydrology, plankton, fishes, etc.). Moreover, in this project, data are produced independently, in different formats, collected according to domain-specific methodologies, and not applying metadata and/or ontologies standards for describing them.

The architecture proposed in this work contributes to the linked-data initiative by providing a data integration framework that inherits from mediator systems (Wiederhold, 1992). It enables the publication of public and private data sources as LD endpoints in RDF. Data from multiple sources are integrated as high-level data views, and may be complemented with additional information published on the Web of data. Data views allow non-specialist users to specify their views of interest over an integrated conceptual schema defined as an ontology. Data views are automatically transformed into queries, which are therein produced and executed as workflows by a query processor engine.

However, having inter LD on the Web by applying LD technology is not enough to construct integrated data views. The major challenges are related to the following factors (Heath and Bizer, 2011):

- the discovery of the relevant LD sources;
- the heterogeneity of the LD sources and their vocabularies;
- the quality of the data, which can be fragmented, incomplete, incorrect or inconsistent; and
- the use of multiple unique resource identifiers (URIs) to denote the same resource, which requires URI conflict resolution.

In this work, we solve some of these challenges with respect to the integration of heterogeneous data sources. We provide a mediated view over these sources by describing it as an ontology in RDF. Ontologies are adopted as the formalism for representing exported data and the integrated view, and mappings between ontology elements on different integration levels solve ambiguities and define rules for composing integrated concepts.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

The great contribution of this work is focused on:

- fostering linked-data initiative in science by providing an adapted data mediation architecture based on linked data principles;
- providing a semantic integration approach that simplifies the integration process both in terms of mappings and query answering through data views; and
- offering scientists (i.e. non-computer science specialists) a tool that enables them to specify high-level data views that are automatically rewritten as an integrated workflow for RDF queries.

This architecture has been validated for real applications. Applied as an exploration mechanism, queries executed over this architecture enabled biologists to discover interesting facts concerning marine species. Some useful examples of queries involve:

- exploring trophic chain hierarchies within a certain region and under specific conditions; or
- getting information about species predators in different levels of a hierarchy within a taxonomy.

This latter query turned into a complex query, which was enriched with additional information available in DBpedia.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 describes in detail the four-level architecture proposed to integrate data resources. Section 4 details the process of generating data integration workflows (DTWs). Section 5 presents some real PELD application scenarios that will be used as case study for integration. It also shows the PELD ontologies generated at each level of the architecture. Section 6 describes the environment created for the semantic integration process through data views, in the context of some real queries applied over PELD. Additionally, this section presents a set of experiments that compares the query executor engine (QEF-LD), used to process data views, with other federated query engines. Finally, Section 7 concludes the paper with suggestions for future work.

2. Related work

Data integration refers to the ability to access and manipulate data transparently across multiple data sources. This issue has been a hard problem to handle since the 1980s, when database scientists needed to integrate many different database schema models. In this section, we focus on two main points of data integration: semantic technologies; and a survey of the domain of biodiversity, from a data integration perspective.

Basically, two main strategies may be considered for data integration: materialized and virtual approaches. The materialized approach represents a classic data integration scenario: it collects, stores and accesses data in a central database, based on a schema mediation (Wiederhold, 1992), where each data source depends on some code or on some DTW definition in an extract–transform–load environment. On the other hand, the virtual approach enables the execution of federated queries over a fixed set of data sources but it requires query rewriting when a data source schema changes, which may be time-consuming.

Although the latter is the current approach used to integrate data in the Web of data scenario (Heath and Bizer, 2011), it requires a significant effort, as there is often no explicit semantic description of the contents published there.

For many years, adding semantics to the data integration process has been a concern (Wache *et al.*, 2001; Noy, 2004; Cruz and Xiao, 2005; Barret *et al.*, 2005; Goble and Stevens, 2008; Cruz and Xiao, 2009). They all call attention to the need of using ontologies as an important mechanism to achieve this goal, and a diversified number of tools have been developed to cope with the complexity of semantic data integration. Dark (Quilitz and Leser, 2008), SemWIQ (Langegger *et al.*, 2008) and FedX (Schwarte *et al.*, 2011) are some examples of semantic Web engines for distributed query processing. They make use of the SPARQL language to execute federated queries (Prud′hommeaux and Seaborne, 2008), providing transparent access to RDF data sources. Dark extends Jena[4] ArQ[5] to enable federated queries with transparent access to multiple SPARQL endpoints. SemWIQ[6] is based on a mediator–wrapper architecture, while FedX is a framework that extends Sesame[7] with a federation layer for transparent access to data sources. In this federated query scenario, the same strategy is used for accessing data sources: a SPARQL query is decomposed into subqueries, which are then sent to distributed data sources for execution. The results of the subqueries are returned from the respective sources and integrated, before providing the final query results.

Additionally, it is also worth mentioning the LD integration framework (LDIF) (Schultz *et al.*, 2012). It can be used as a component within LD applications. It gathers LD from the Web and translates them into a clean local target representation in RDF, allowing for SPARQL endpoints access. As the data is materialized, the LD crawler may keep track of these modifications.

Despite the many advantages of all these tools, most of them are not able to execute queries over a domain ontology, with mappings for specific application ontologies.

In this context, many system architectures have been developed in this direction. Mastro Studio (Civili *et al.*, 2013) is an ontology-based management system, where ontologies are specified in description logics (Calvanese *et al.*, 2007). Data sources are seen as a single relational database (RDB), and mapping assertions are used to associate the ontologies and the database. A graphical user interface allows users to navigate through the ontology and to express queries, which are computed through a query rewriting process to standard structured query language (SQL). Another important work has been developed by Angele and Gesman (2006). Similar to our work, they also use a four-level architecture (data sources, source schemas, mediated view – expressed by an ontology – and views on top level) to integrate data in the business domain. Mappings between data sources and business ontologies are specified by F-Logic rules, and views are manually defined queries.

More aligned with life sciences and the Web of data scenarios, which represent the main focus of our work, it is worth mentioning an advance in bioinformatics. The work proposed by Knoblock *et al.* (2012) semi-automatically maps contents of bioinformatics databases in terms of a given ontology. The source model therein generated is refined through a graphical tool, and converted into RDF triples.

Additionally, in the biodiversity domain, the literature points out several system architectures for managing and sharing ecological data. A very important initiative is the GBIF[1], which aims to make the world’s biodiversity data freely and universally available via the Internet. As a mega-science initiative, GBIF’s goal is to offer an

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

essential global informatics infrastructure for biodiversity research and applications worldwide. It provides functionalities for publishing, discovering, indexing, integrating, retrieving and analyzing data. Data and their associated metadata from different repositories are made available using ecological metadata language) (Fegraus *et al.*, 2005) and Darwin Core (Wieczorek *et al.*, 2012), which are metadata standards for, respectively, describing and sharing data about biodiversity.

In Brazil, there exist many initiatives to manage and provide access to valuable ecological information across the country. They take into account its ecological diversity, which is considered to be one of the richest on the planet, responsible for 10 per cent of the whole terrestrial biota (Mittermeier *et al.* 1997). CRIA[8] is the acronym of a Reference Center on Environmental Information in Brazil, and it encompasses many biological repositories, such as the SpeciesLink[9], the INCT-Virtual Herbarium of Flora and Fungus[10] and SinBiota[11]. More recently, the SiBBr (System for Brazilian Biodiversity)[12], a very important project in the context of biodiversity is under development. It represents the Brazilian node of GBIF, and it provides an entry to integrate data collected and published by Brazilian institutions, such as academic, research and governmental agencies.

A deep analysis of these systems, from a computer scientist point of view, lead us to conclude that integration for them means having data available from many different sources, in a centralized way, using a Web portal. Species are indexed according to well defined domain taxonomies, but information retrieval in these portals is usually done through keywords that are compared with metadata descriptors, without taking into account semantic and/or data constraints provided by the use of ontologies.

However, the literature also mentions some important advances in the biodiversity domain, where ontologies are used as a very important step to integrate data. Aondê (Daltio and Bauzer, 2008) and SemantEco (Patton *et al.*, 2014) are initiatives developed in the context of biodiversity. They both use ontologies as a means to integrate and interoperate data, making it easier for resource managers to discover new sources of data to support more complex domain applications. The first work is based on an ontology Web service. It gives support to manage and store ontologies, besides providing for service operations over integrated sets of ontologies. Applications can thus enhance their semantics and interoperate by becoming clients of this service, thereby exchanging, reusing, integrating and adopting concepts from ontologies published on the Web.

The second work, the SemantEco, is a semantically enabled environmental monitoring framework. It integrates various ecological and environmental data (such as water quality, fish and wildlife species), using a family of domain ontologies. Interoperability is achieved by mapping SemantEco's schema with applications compatible with Extensible Observation Ontology (Madin *et al.*, 2007). By representing these ontologies in OWL-2 ontology language[13], data querying is provided with inference, supported by good reasoning capability.

In the context of PELD, data sources are very heterogeneous and captured through different formats (datasheets, images, text, csv files and relational tables). Differently from the systems studied so far, and probably due to the schema heterogeneity and lack of consensus of using standard metadata vocabularies, each PELD coordinator follows his/her own methodology to collect data. Only very few coordinators do follow any ontology standard to describe data and metadata. To overcome this situation, we apply a strategy for semantic integration, different from the strategies applied in the works studied so far. In our architecture, data from distinct data sources are transformed into an ontological representation based on mappings, and joined into a single-mediated schema. Thus, the mapping complexity across the architecture layers is traded by the process of transforming the data sources into an exported ontology in RDF. The exported ontology is based on the same vocabulary of the architecture upper level, the application ontology (AO), thus reducing the problem of integration to that of mappings between homogeneous ontologies.

Once PELD data is published as LD, it is possible to enrich it with new information. This is captured on the fly from other sources in the Web of data. This would not be possible, for example, if all data was materialized previously in a data warehouse because in such systems, data dynamics is not the main goal.

In the architecture proposed in this work, we adopted an hybrid approach, where data can be materialized (mainly those that do not change so often) or virtually accessed, according to the situation. Hence, for us, data integration is considered as the result of a query, expressed as a data view, which is submitted over a mediated schema and executed over the data sources as LD. A submitted query is processed transparently as a workflow, whose goal is to apply data transformation operations to integrate data sources. This represents the main differential of our work.

3. Ontology-based architecture for building data views

In this section, we describe a four-level ontology-based architecture that facilitates the creation of data views from multiple data sources published as LD. Data interpretation between these levels is ensured by an ontological representation based on mappings, according to a mediated approach, extended from (Wiederhold, 1992), LDIF (Schultz *et al.*, 2012) and Vidal *et al.* (2011). Each level of the architecture illustrated in Figure 1 is described as follows:

- *Level (4)*: Each data source S_i represents an information resource described by a schema (Sch_i) that will be published on the Web according to the LD principle.
- *Level (3)*: Each data source S_i may export one or more exported views. Each such exported view E_i is described by an ontology EO_i. The specification of EO_i views is obtained by mapping concepts of Sch_i to concepts of the upper level, the AO, using a set of mapping rules μ_E. The subset of AO mapped to a given Sch_i exported view forms the corresponding EO_i. Each EO_i becomes available as SPARQL endpoints, i.e. it provides a RDF view of the underlying data source S_i, which ensures its publication as LD. Making use of inter-ontology links (e.g. those created by *owl:sameas*, or linked by the EO_i namespace), instances of one EO_i can be directly associated with instances of another EO_i, or with a Web data set published as LD. Hence, an E_i can be specified by a quintuple (*E, S, Sch, EO* and μ_E), where: *E* is the *name* of the view; *S* is a data source; *Sch* is the schema of *S; EO* is the exported view ontology*; and* μ_E is a set of mapping rules from *Sch* to *EO.*
- *Level (2)*: This level represents the mediated view, i.e. an integrated view of EO_i*s*. The mediated schema is described by an ontology named as AO. Given that the EO_i*s* have been constructed according to mappings to the AO, the correspondences between the AO and EO ontologies are simplified. This means that the set of mediated mappings that specify how to transform EO_i instances

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

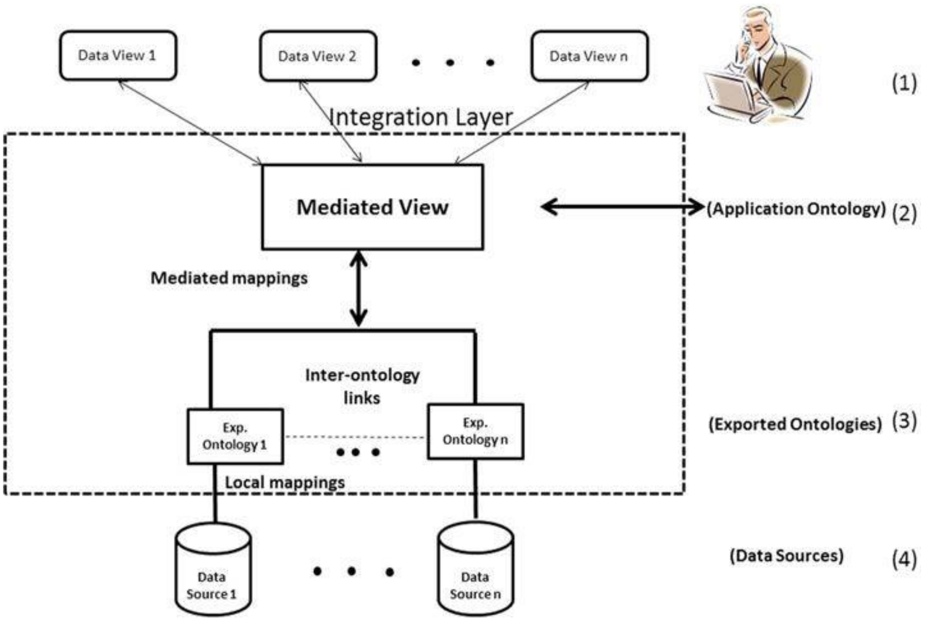


Figure 1.
A four-level architecture for LD view integration

(triples) into AO instances become trivial. Table I shows an example of these transformations. A *mediated view* can be specified as an *n-tuple* (AO, E_1 [...], E_n), where: AO is the application ontology; and E_1 [...], E_n are *exported view specifications* with their respective ontologies EO_1 [...] EO_n. Finally, an AO is correspondently published in RDF.

- *Level (1)*: The user data requirements are specified in this level through data views. Conceptually, a data view DV_i defines the data objects users want to extract from the data sources, according to the vocabulary *V* specified in the AO. Thus, a DV_i reflects on some fragment of the AO, expressed as a query. A DV_i can be specified as a *n-tuple* (*Q,AO,C,P* and *F*), where: *Q* is a query identifier; C is a set

Table I.

AO	Plankton EO
(a)	
a: Plankton_Sample	epl: Plankton_Sample
a: has_pl_analysis	epl: has_pl_analysis
a: collected_in	epl: collected_in
a: weight_pl	epl: weight_pl
...	...
(b)	
a: Sample(p) ⇐ epl: Sample (pl) U ecf: Sample(cf) U eco: Sample(co)	

Notes: (a) Vocabulary matching between the AO and the plankton EO; (b) a sample instance in the AO is expressed as the union of the EO instances: (*pl*) for plankton; (*cf*) for CatFish; and (*co*) for Comm_fish

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

of classes in AO; P is a set of properties in AO; and F is a set of constraints (filters) applied over the terms of AO. A data view is specified through a friendly interface that offers the user the ability to select the terms that will compose it. Figure 7 presents the interface provided for data view creation, with a query example concerning PELD application.

This architecture follows a similar idea of the three-level ANSI-SPARC database architecture (Elmasri and Navathe, 2010), where a data view corresponds to an external view in the ANSI-SPARC architecture. It describes the database part a particular group of users is interested in.

4. Data integration workflows

As presented in Section 3, a data view DV defines a query on an AO. Executing a DV requires generating a workflow that integrates the corresponding $EO_i s$ and fulfills the data requirements in DV. In this section, the process of generating DTWs is detailed.

4.1 DTWs as union and join

A DTW is defined as adequate with respect to the data view, if it is a composition of *union* and *join* operations. Unions are used whenever the same query object, class or property, appears in more than one EO. Retrieving instances corresponding to such a query object requires the application of the *union* operator over the EO_is (such as sample in Figure 6). *Joins*, in their turn, are used whenever a query object refers to a property in the AO, whose range and domain are mapped into classes in different EO_is.

4.2 DTW generation

The process of generating a DTW transforms the data view over the AO into a workflow composed of: queries to the participating $EO_i s$; joins and unions. Queries select objects from the data sources according to the constraints in the data view.

Hence, a DTW can be specified as a triple (C_{dv}, R and F), where: C_{dv} *205* C is the set of classes selected in the data view; R are the object properties (relationships) associated with these classes. Each relationship is associated respectively with a *domain* and with a *range* class; and F is the set of constraints applied over R. The DTW, also known as a *query execution plan*, represents the output generated by Algorithm 1, as described next. This DTW will contain an ordered list of operators Op_k, $1 \leq k \leq n$, which can be *union* or *join*.

4.2.1 Query execution plan. Given a data view, a graph corresponding to a query execution plan (i.e. a DTW) for that data view is generated, according to Algorithm 1. Initially, this algorithm identifies the relationships that are of type "is-a" (Line 3) and those that are not (*nIsA*) (Line 4), and processes-specific functions to group classes taking part of *Unions* (Line 5) and *Joins* (Line 6). These two functions are respectively executed by Algorithms 2 and 3. Finally, the final DTW is generated by Algorithm 4, which is presented in Appendix 1.

In Algorithm 2, for each relationship of type "is-a" it groups into a *Union* list the classes that have the same range and creates a union list (Lines 2-10). Each union list corresponds to a different union node, as there may exist classes in the DV taking part of different unions. This algorithm returns the Union list containing all the unions found in the DV (Line 11).

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Algorithm 1. Data Transformation Workflow (DTW)

Input: (DV, AO) /* Data View and application ontology */
Output: DTW/* list of operations that will be performed over the classes, as a workflow, by QEF */

```
   /* **Initialization** */
1  r ← getRelevantRelationships (DV,AO) /* r contains all relationships in AO
   relevant to DV */
2  c ← getRelevantClasses (DV, AO) /* c contains all classes in AO relevant to DV */
3  isA ← r.getRelationships_type_IsA() /* isA contains all relationships in AO of type
   is_A */
4  nIsA ← r.getRelationships_type_Not_IsA() /* nIsA contains all relationships in AO
   of type ≠ is_A */
5  unionList ← Create_UnionList (isA) /* Create unions */
6  joinList ← Create_Join_list (nIsA,unionList) /* Create joins */
7  end-for
8  buildDTW (JoinList, UnionList) /* Build the DTW output */
```

Algorithm 2. Create_Union_List (List isA)

/* Create union nodes: group classes in a domain role in "is-a" relationships that share the same range and adds them as a unionNode */

```
 1  unionList ← Ø
 2  for each rel in isA
 3      unionNode ← null
 4      obtain classRoot from rel such that classRoot is rel.Range /* get isA range class*/
 5      obtain unionNode from unionList such that classRoot = unionNode.root
 6        if (unionNode ≠ null)
 7            unionNode ← createUnion(classRoot)
 8        end-dif
 9        unionNode.add(rel.Domain);
10      end-for
11      return unionList
```

Algorithm 3 is responsible for creating *Join* lists whenever one of the following situations arises:

- a join occurs between classes that do not take part of the *Union* list (Lines 1-5). A join node is then created between these two classes (Line 2-5);
- one of the classes in the *join* relationship takes part of the *Union* list as range class (Lines 7-10) or as domain class (Lines 11-13), after which a join list is created between the root of the union node and the range (Line 10) or the domain (Line 12), respectively;
- a join occurs between classes of the same *Union* list (Lines 17-25). In this case, both classes are removed from their union nodes (Lines 21-22), their schemas are merged into a union node and added into a join list (Lines 23-24); and
- when a join occurs between classes from different union lists.

From the domain of one class and the range of the other, the union nodes are obtained and a join between these two nodes is created (Lines 26-34). Finally, the algorithm returns the join list concerning the DV (Line 35).

Algorithm 3. Create_Join_List (List nIsA, List unionList)

```
/* Create join nodes: Identify situations where a join can occur */
/* i- Relationships with classes not taking part of any UnionNode */
1   obtain x from nIsA with no domain or range class in Union
2   for each rel_x in x
3       joinNode←createJoin(rel_x.Domain,rel_x.Range) /* Create join list with classes
      not taking part of a Union */
4       JoinList.add(JoinNode)
5   end-for
/* ii- Relationships with a class, domain or range, taking part of a UnionNode */
6   obtain y from nIsA with only one class, domain or range in unionList
7   for each rel_y in y /* Create a join list whose range of a class takes part of the
    Union */
8       if (exists c such that c in unionList.classes ∩ rel_y.Domain) /* if c is the
      domain */
9           obtain unionNode from unionList such that unionNode contains c /*
            unionNode          is the Union operator */
10          joinNode←createJoin(rel_y.Range, unionNode.root) /* a joinNode is created
      between the range of this relationship and the union */
11      else /* a joinNode is created between the domain of this relationship and the
      union */
12          joinNode←createJoin(rel_y.Domain, unionNode.root)
13      end-if
14  JoinList.add(JoinNode)
15  end-for
/* iii- Relationships between classes of a same UnionNode */
16  obtain z from nIsA with both classes(domain and range) in a UnionNode
17  for each rel_z in z
18      mergeNode←createMerge(rel_z.domain, rel_z.range) /*merge into
     a single class */
19      c ← merge.getOutputClass /*obtain the schema of the merged classes */
20      obtain unionNode from unionList such that unionNode contains rel_z.domain
21      unionNode.remove(rel_z.domain)
22      unionNode.remove(rel_z.range)
23      unionNode.add(c)
24      joinList.add(mergeNode)
25  end-for
/* iv- Relationships with classes in different unionNodes */
26  obtain k from nIsA with classes in different UnionNodes
27  for each rel_k in k
28      obtain c from rel_k.domain
29      obtain d from rel_k.range
30      obtain unionNode from unionList such that unionNode contains c
31      obtain unionNode2 from unionList such that unionNode2 contains d
32      joinNode ←createJoin(unionNode.root,unionNode2.root)
33      joinList.add(joinNode)
34  end-for
35  return joinList
```

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

4.2.2 Example of DTW. Figure 2 illustrates an example of DTW generated by Algorithm 1, based on PELD application. It represents a query specified through a data view. Observe that the DTW representation takes the format of a left deep tree, usually adopted as the representation of query execution plans (Elmasri and Navathe, 2010). In this representation, each leaf node of the tree corresponds to a query to be sent to an endpoint. Intermediate nodes represent data transformation using one of the operators, *Join* or *Union*.

For each EO that is mapped to the AO fragment of interest, a SPARQL query is internally generated. SPARQL EO queries return the instances of interest from each source, which are input to the DTW. The set-*bindjoin* operator (Magalhães *et al.*, 2013) obtains the set of input values from the EO queries, and computes the node on its left-hand side ($Q_{plankton}$ 200 $Q_{catfish}$ 200 Q_{comm_fish}) (Figure 2), which will be joined with the data in the data source represented on its right-hand side (Q_{region}) in the tree. These bound values substitute placeholders in an EO query.

The ordering of the workflow operations in a DTW may influence query results. Ordering Unions has no effect on either the result values or the computation efficiency. They can be placed in the workflow in any order. The same is not observed between joins. Thus, during the data view design (which consequently generates a DTW preserving the processing order of the data view elements), the user may try different orderings, by running experimental workflows and choosing the most efficient one. Hence, with these very simple heuristics, users are able to produce a simple and efficient DTW. Moreover, given that the semantics of the data transformation operations are very familiar (i.e. as in relational model operation semantics), in the future, an optimizer may automatically improve the initial user workflow, applying automatic workflow transformations (Ogasawara *et al.*, 2013). A URI is assigned to each DTW specification.

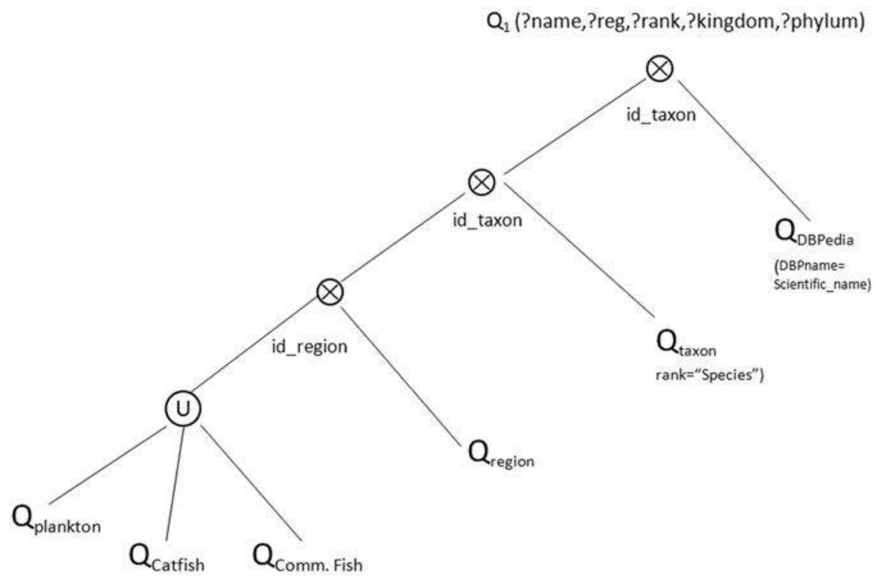


Figure 2.
A DTW representing a query

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

This will allow the user to reexecute the same workflow or simply alter some of its parameters, according to his/her application needs.

4.3 DTW execution environment

Executing a data view represents the final step of the semantic integration process: the result sets obtained from the data sources are processed by a DTW, integrated and computed by QEF (Porto *et al.*, 2007), an extensible workflow query engine, together with QEF-LD, an additional component to process LD (Magalhães *et al.*, 2013). The semantic integration can be expressed as tuple (*DTW* and *RS*), where: *DTW* represents the ordered list in which the operators will be executed over the classes; and *RS* is the result set containing the query results.

Therefore, results obtained after executing a DTW contain the scientist's integrated and materialized data, corresponding to his/her data views, as described in Section 3. QEF-LD is a Web service application that processes SPARQL queries over federated LD sources, i.e. through their endpoints, making use of a scan operator. It includes LD algebraic operators (such as *Union* and *set-bind-join*) and wrappers that submit a SPARQL query to a D2R endpoint or to any other LD Web resource. During the query processing, join ordering between distributed endpoints are defined, while local joins remain specified in SPARQL subqueries to be run by the endpoints themselves. Inter-site joins are implemented by the *Set-BindJoin* operator, whose algorithm description is described in detail in (Magalhães *et al.*, 2013).

Figure 3 illustrates a generic architecture for processing DTWs. The execution process depicted in this architecture works as follows: a user data view processing request is submitted to the system referring to a URI that points to the corresponding

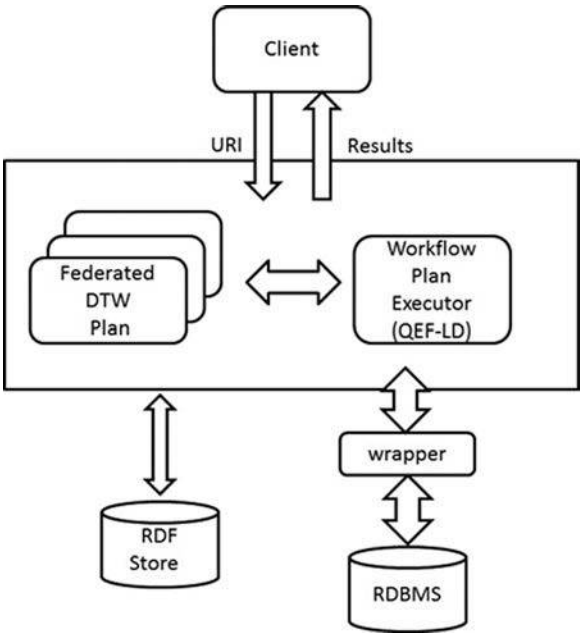


Figure 3. System architecture for computing DTWs in LD

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

DTW. Then it is processed by QEF, the workflow plan executor. To access EO elements, the DTW is segmented into distinct SPARQL queries. Section 6 (Item 3) illustrates this segmentation process in more detail, during a data view execution applied to the PELD scenario. It is worth observing that this architecture also considers data already stored in RDF repositories.

5. Application scenarios

This section describes the Guanabara PELD scenarios that will be used for integration, according to the architecture depicted in Figure 1. The main goal of this integration is to provide knowledge about the Guanabara Bay ecosystem, as well as to be a source of answers to the anthropic and climatic impacts on the bay ecosystem. Additionally, it will enable evaluating the adequacy of actions that are being taken to clean up Guanabara Bay, regarding the marine ecology.

Guanabara PELD aims to get biotic and abiotic data from samples extracted from the bay water, from planktonic and benthonic communities and from fishing resources, from which a few features related to the plankton biomass are studied.

A life science taxonomy classifies organisms according to a hierarchy. The first level corresponds to the *Kingdom*, which is decomposed into *Phylums* and successively into *classes*, *orders*, *families*, *genus* and *species*. Each level has respectively its own subdivisions. Any level within this classification is called a taxon. There exist differences in the levels concerning each organism. Some of them have been reclassified, and in this case, both classifications are kept and a synonymous relationship is established between them. Furthermore, each taxon scientific name is associated with its corresponding instance or class name in DBpedia[14].

Some important features deserve some attention in ecological data analysis. Geographical region information is used for selecting and classifying events according to their location. On the other hand, trophic relations are fundamental for the ecosystem study. Finally, the taxonomy enables a hierarchical analysis of the species. The analysis of these aspects may be explored by the use of inference in an integrated way. The main characteristics of each scenario are described next.

- *Plankton*: In the plankton scenario, a sample data takes into account temporal (data and time) and spatial (latitude, longitude and profundity) information, as well as methods used for sample collection and preservation, atmospheric and maritime conditions during each collection. For each analysis performed, information about its taxonomy classification and the applied method of collection are registered. Biomass measurements of organisms found in the samples can be done at species level or at the taxonomy highest level.
- *Community fish*: Besides temporal and spatial information, this application stores the fishing method used to catch fish, taking into account two different depths (initial and final). It is worth observing that fish collections are divided into three samples, from which the total weight and number of individuals are analyzed for each taxon found in the collection process.
- *Catfish genidens*: Differently from the previous scenarios, this application analyzes each specific species individually, considering not only spatial and temporal references but also the fishing method used in the collection process, the specie weight, length and gender (male/female).

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

It is worth observing that each scenario represents the study of a different research group. Hence, the importance of this work is to create an infrastructure to allow biologists of different groups to explore all these ecological data by integrating and filtering them according to specific constraints, to enrich their researches.

Figure 4 applies PELD scenarios to the architecture presented in Section 3, which will be used as our case study.

5.1 PELD ontologies

Based on the application scenarios described before, this section presents the ontologies generated at each level of the architecture depicted in Figure 4. These ontologies will be relevant to describe the semantic integration process that will be presented in Section 6.

5.1.1 PELD exported ontologies. The full potential use of LD depends on how easy it is to transform data from relational databases (RDBs) into RDF triples. According to Malhotra (2005), there exist two main ways of exposing data:

(1) translating relational data into RDF and loading them into an RDF store; and

(2) generating a RDB mapping that can be queried using SPARQL, which is then translated into SQL.

Additionally, we consider a third strategy, also called hybrid, which combines Strategies 1 and 2. In the hybrid strategy, part of the data is materialized and directly stored in RDF, whereas another part requires a RDB mapping as in (ii). Considering that Strategy 1 is harder to maintain consistency as a result of frequent DB update, and that part of PELD data is stored as materialized data (due to the low update frequency), we opted for implementing the hybrid strategy, according to the proposed integration architecture.

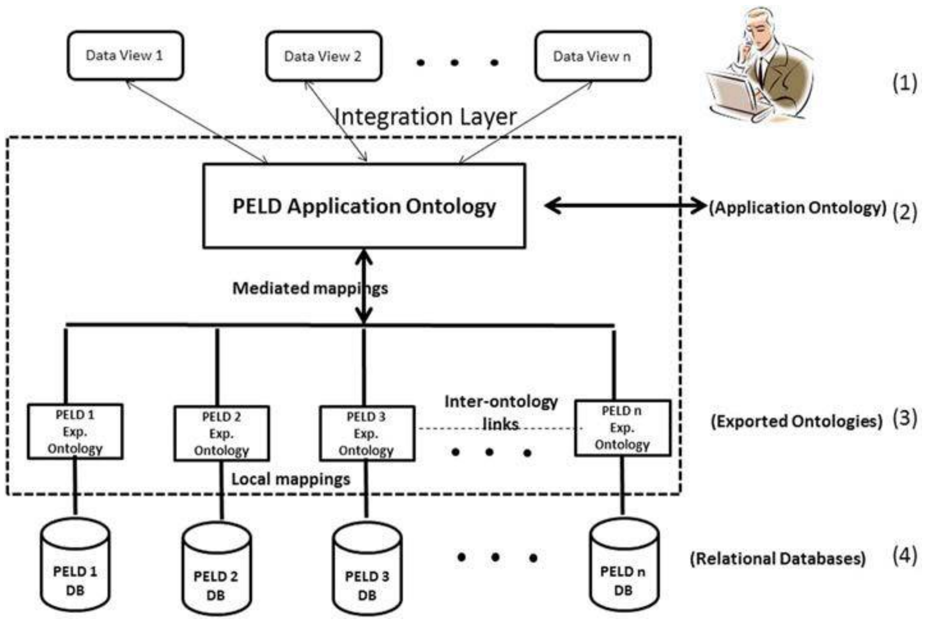


Figure 4. The architecture applied for PELD scenarios

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Since most of PELD data are stored in RDBs, we decided to use technologies developed in the context of the RDB2RDF group[15]. Its main goal is to standardize a language for mapping relational data and schemas into RDF and OWL[13] to enable publishing, and integrating vast amounts of information stored in RDBs on the Web.

The generation of RDB mappings is the strategy adopted in this work to make our PELD relational data sources available as endpoints. PELD databases are stored in the PostgreSQL[16] RDB, and are mapped automatically into a virtual RDF graph with the help of the D2RQ tool[17], a platform developed in Java that enables executing SPARQL queries and accessing data in the database as LD. This platform is constituted of the following components: the D2RQ mapping language, a declarative language that describes the mappings; the D2R server (Bizer *et al.*, 2006), a HTTP server that provides a LD view and SPARQL queries; and the D2RQ engine, a plugin that accepts reasoners such as Jena[5] and Sesame[6].

Hence, an EO describes each respective view schema published in RDF. Figure 5 presents the *Plankton*, the *Community Fish* and the *Catfish Gen* EOs, according to the application scenarios described before. The first EO is composed of three main classes: *Sample*, *Plankton_Sample* and *Pl_analysis;* the second contains the *Sample*, *Catfish_Sample* and *Cf_analysis* classes, while the third EO comprises the *Sample*, *Comm_Fish_Sample* and *Comm_analysis* classes.

These classes are virtually associated with other ones by specific object properties such as *has_taxon* and *collected_in*. *Taxon, Trophic_Chain* and *Taxonomy* classes are responsible for describing each marine sample (plankton, community fish and catfish) in

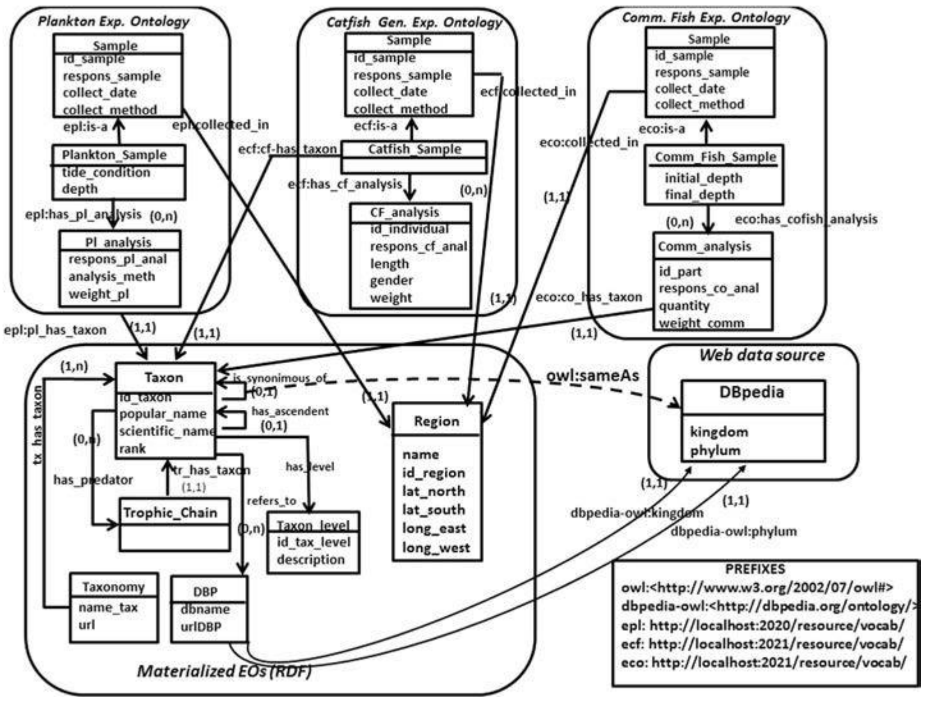


Figure 5.
Marine EOs, materialized EOs, and a Web data source (DBpedia library)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

terms of its taxonomical classification; *Region* gives the geographical localization where the sample collection took place; and the *DBP* class contains the link (*url*) and the corresponding taxon in the DBpedia library.

As the classes *Region, DBP, Taxon, Taxonomy* and *Throphic_Chain* are shared by most of PELD views, and their instances do not change very frequently, we decided to materialize them. This means that their materialized EO instances are stored in RDF, together with the RDF schema to ensure a better performance during the workflow execution. In other words, the access to these instances will not be virtually processed during a workflow execution, as it normally occurs with the other classes, although this strategy is completely transparent to the final user. It is worth mentioning that the *DBP* class, which is instantiated during the EO generation, has been added to the EO as a means to provide the user with additional information extracted from DBpedia library.

Each *id_taxon* in the *Taxon* class is associated with the *DBP* class by the object property *refers_to,* whose goal is to establish a link between a taxon species (through its *scientific_name*) and the DBpedia library. *DBP* instance values (*DBPname* and *url*) are extracted from DBpedia by LIMES (Ngomo and Auer, 2011), a program that discovers links in metric spaces. This approach uses mathematical characteristics of metric spaces during the mapping process to filter out those instance pairs that do not meet the mapping conditions.

The following shows some examples of owl:*sameAs* instances that link a Taxon scientific name to its equivalent species in DBpedia library, as shown in Figure 5.

http://localhost:2020/resource/peld_taxon/taxon/111
www.w3.org/2002/07/owl#sameAs http://dbpedia.org/resource/Acartia.

http://localhost:2020/resource/peld_taxon/taxon/60
www.w3.org/2002/07/owl#sameAs http://dbpedia.org/resource/Coscinodiscophycidae

These links are made possible by the *url* stored in the *DBP* relation.

EO elements are distinguished through namespace prefixes "*epl*:", "*ecf*:", "*eco*:" and refer to the *Plankton, Catfish Geniden and Comm. Fish* vocabularies within PELD ontologies, respectively. For example, the instances of the property *epl:collected_in* has *Plankton* as domain class and *Region* as range.

5.1.2 Application ontology. Figure 6 shows the AO conceptual model that represents the mediated view of the three EOs, over which data views will be built. The design specification from EO to AO is another important step in this architecture, but it will not be explored in the scope of this paper.

The namespace prefix "a" is used to refer to the vocabulary of the AO. Because most of the class properties are self-described, we just give a few examples of the class properties. *Sample* is the superclass that encompasses all the marine species within Guanabara PELD. It specializes into *Plankton_Sample, Catfish_Sample* and *Comm_Fish_Sample* classes; *a:collect_method* property is defined as a datatype property with domain *a:Sample* and range string; *a:has_predator* is an object type property with domain *a:Taxon* and range *a:Trophic_Chain*; and *a:has_pl_analysis* is defined as an object property with domain *a:Plankton_Sample* and range *a:Pl_analysis*.

The mapping procedure is important to ensure an efficient rewriting process. It is achieved by using a *global-as-view* mapping (GAV) (Lenzerini, 2002) between the AO and EO concepts. The GAV strategy considers that the AO schema is described in terms of its EO schemas. However, since the AO and EO ontologies share the same vocabulary, the mediated mappings that define the correspondences between the concepts and

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

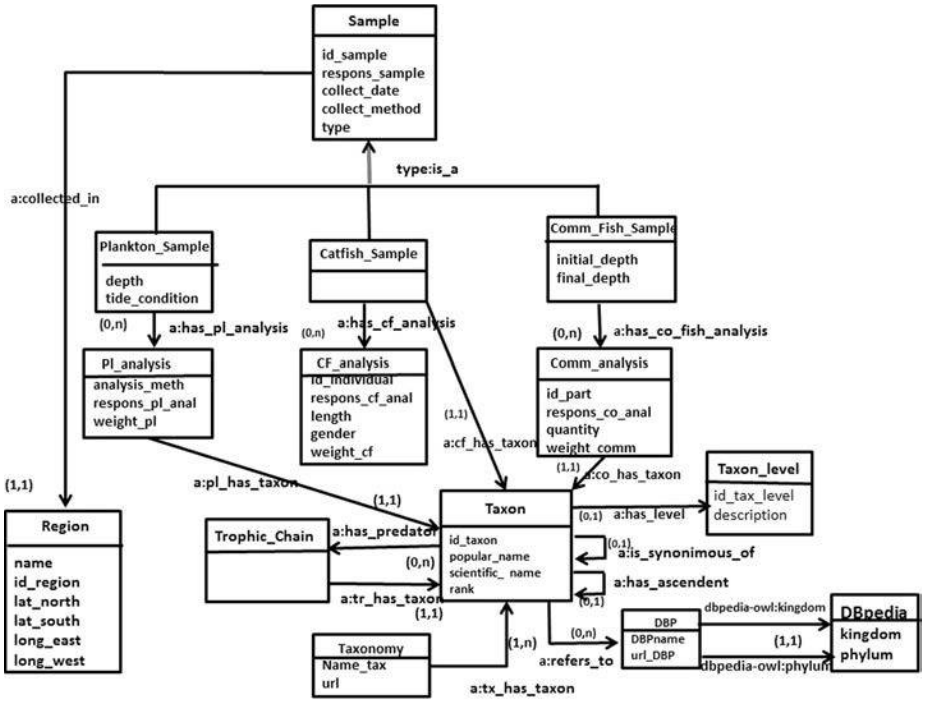


Figure 6.
PELD AO

properties of these ontologies become simplified. Furthermore, as the AO is a conceptual integrated view of the EOs, it is worth observing that in our application, a species in the *Sample* class can be seen as the union of the different species of each EO. Table I shows an excerpt of the vocabulary matching between the AO and EO levels, focused on the *Plankton* ontology. The same matching procedure applies to the other EOs.

6. User environment for creating and processing data views

This section aims at describing step by step the integration process performed over PELD resources, using data views. To exemplify how DTWs are defined and executed using this architecture, consider an application where a biologist wants to have an overview of some taxonomy properties of all the species living in the Guanabara Bay, followed by their geographical localization. This query can be expressed in natural language as:

Q$_1$: "List the names, rank, region, and DBPedia library kingdom and phylum properties, considering all samples at their lowest level in the taxonomy hierarchy".

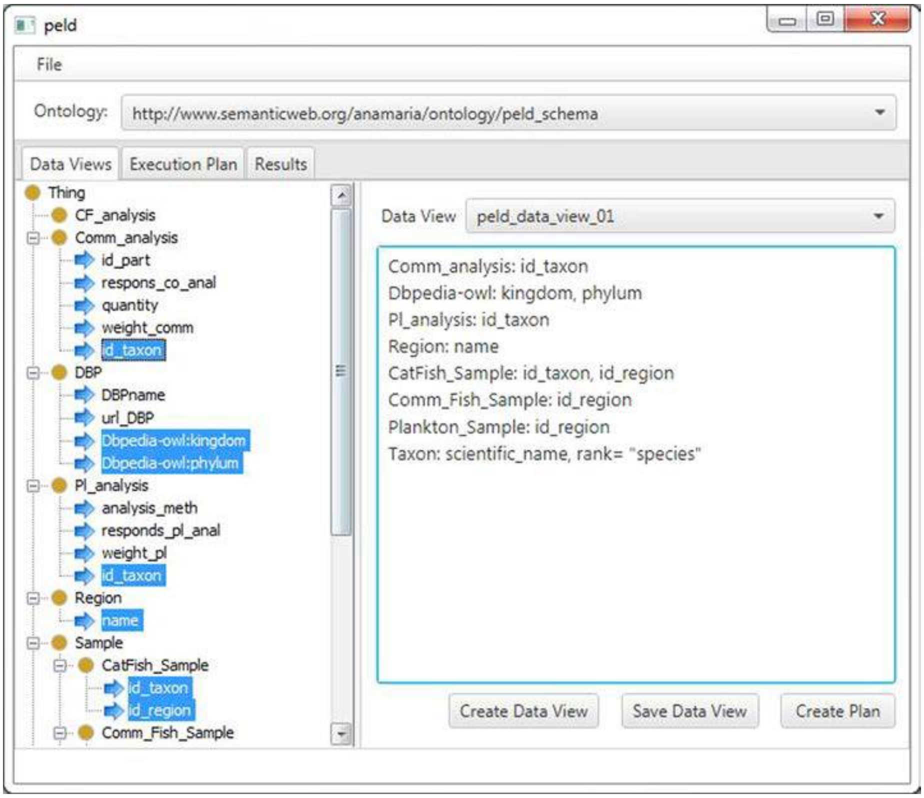
The term *sample* here is used to refer to all the three kinds of species included in our application scenario, represented by the class *Sample* in the AO (Table I). It is worth observing that as not all samples are cataloged at their lowest level (*species*), the query will retrieve all samples, no matter the taxonomy level they have been cataloged in.

Data integration according to our proposal involves the following steps:

- *Defining data views*: The user initially identifies the elements of the AO that represent the expected view of a data source, considered as data views. A hierarchical representation of the AO elements is presented to allow the user to select his/her objects of interest.

According to the AO vocabulary, for answering Q_1, it is necessary to select the *Plankton_Sample, Catfish_Sample* and *Comm.Fish_Sample* classes that compose the class *Sample*. Figure 7 presents the interface from which the user interacts with the system. Data views are composed by the elements shown on the left window of the interface. The user selects the classes and the properties whose values the user wants to retrieve, and specifies filters that will be applied (such as *rank*= "species").

- *Generating DTW*: From the data view created in the previous step, the user can save it and create a query execution plan to execute this data view. In this case, existing mappings between AO and EO levels are applied over the data view to generate a query execution plan (Figure 8), according to Algorithm 1 presented in Section 4.2.1. This query plan, seen as a workflow (DTW), corresponds to a list of operations that will be performed over the EOs to extract objects from the data sources. Furthermore, it is responsible for indicating to the query processor the order in which EO data sources will be accessed. Observe that the DTW receives as inputs the results of queries over the mapped EOs endpoints (exposed as RDF



Ontology: http://www.semanticweb.org/anamaria/ontology/peld_schema

Data View: peld_data_view_01

```
Comm_analysis: id_taxon
Dbpedia-owl: kingdom, phylum
Pl_analysis: id_taxon
Region: name
CatFish_Sample: id_taxon, id_region
Comm_Fish_Sample: id_region
Plankton_Sample: id_region
Taxon: scientific_name, rank= "species"
```

Figure 7.
Creating data views

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

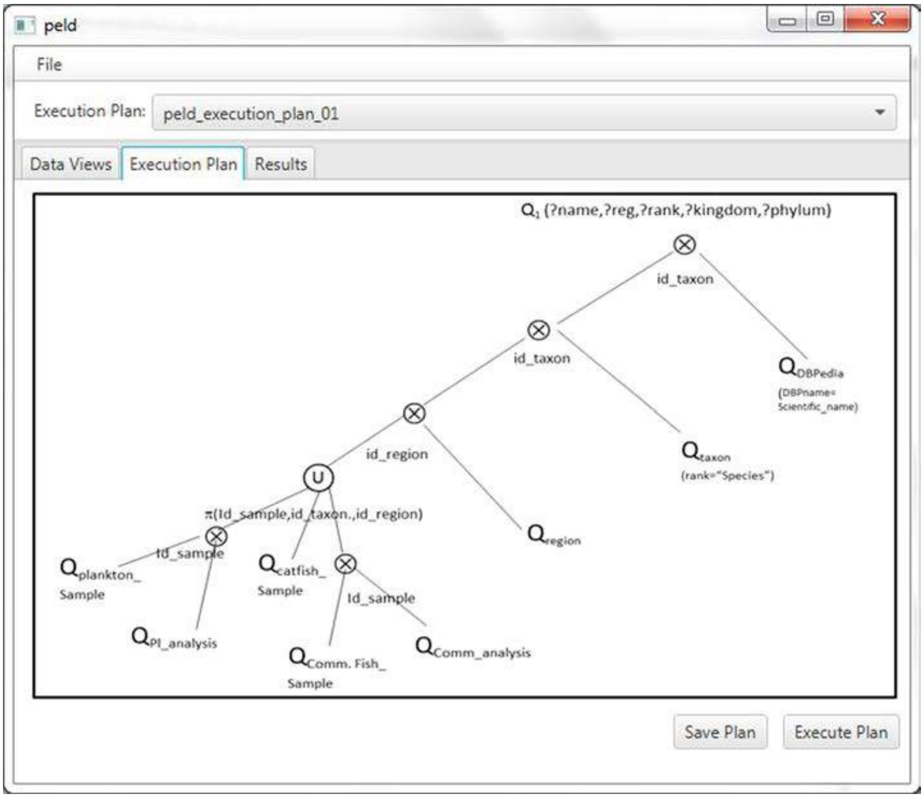
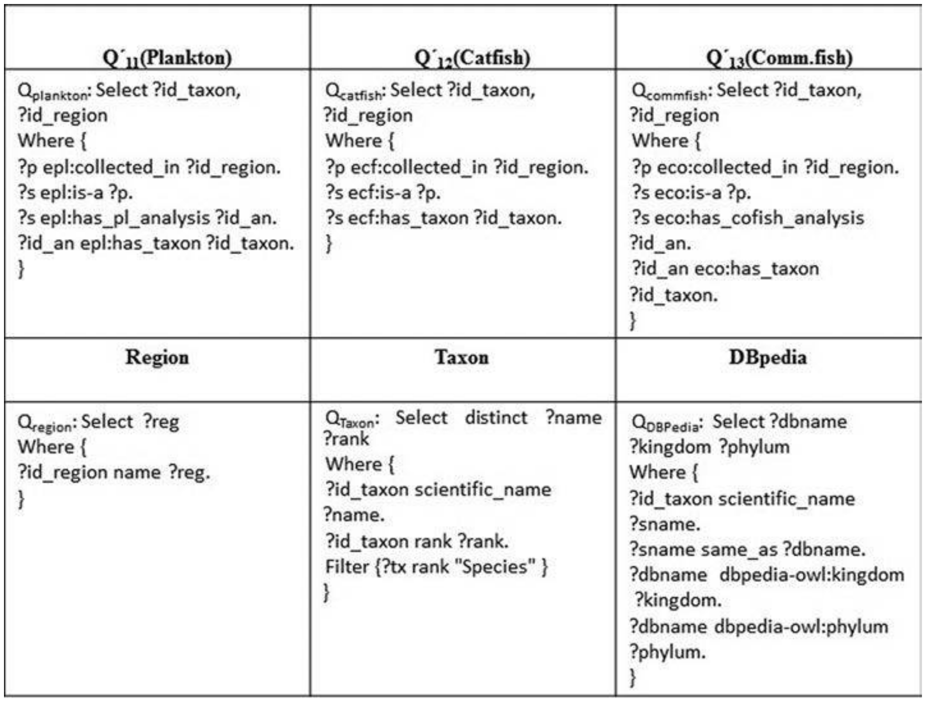


Figure 8.
Query execution plan for Q_1

triples from the source databases, as described in Section 3) and integrates them. The "*Execute plan*" button on the right (Figure 8) allows the user to see the query results, whose execution process is described next.

- *Executing data views*: As described in Section 4.3, a user data view is processed by QEF-LD, the workflow plan executor. To access EO elements, the corresponding DTW of Q_1 is segmented into distinct SPARQL queries, as shown in Figure 9. Observe that each subquery Q'_{1i} represents an access to an EO endpoint. Through the "collected-in" predicate, a *set-bindjoin* retrieves the region from the materialized EO, identifying the place where the samples have been collected. Next, the following *set-bindjoin* operator uses the link *has_taxon* to select in the materialized Taxon EO, the samples classified in rank "Species". Finally, the third *set-bindjoin* complements the data with information from the *DBpedia*. The resulting tuples are fed into the DTW that applies the algebraic operators, computes the final data view and returns it to the client. Because part of the data in PELD is materialized and stored as RDF triples, and because they share the same vocabulary of the AO, a wrapper in this case is not required. Appendix 2 presents the whole DTW XML script submitted to QEF-LD. Finally, Figure 10 shows Q_1 final results that can be saved for later use.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)



Q´$_{11}$(Plankton)	Q´$_{12}$(Catfish)	Q´$_{13}$(Comm.fish)
Q$_{plankton}$: Select ?id_taxon, ?id_region Where { ?p epl:collected_in ?id_region. ?s epl:is-a ?p. ?s epl:has_pl_analysis ?id_an. ?id_an epl:has_taxon ?id_taxon. }	Q$_{catfish}$: Select ?id_taxon, ?id_region Where { ?p ecf:collected_in ?id_region. ?s ecf:is-a ?p. ?s ecf:has_taxon ?id_taxon. }	Q$_{commfish}$: Select ?id_taxon, ?id_region Where { ?p eco:collected_in ?id_region. ?s eco:is-a ?p. ?s eco:has_cofish_analysis ?id_an. ?id_an eco:has_taxon ?id_taxon. }
Region	**Taxon**	**DBpedia**
Q$_{region}$: Select ?reg Where { ?id_region name ?reg. }	Q$_{Taxon}$: Select distinct ?name ?rank Where { ?id_taxon scientific_name ?name. ?id_taxon rank ?rank. Filter {?tx rank "Species" } }	Q$_{DBPedia}$: Select ?dbname ?kingdom ?phylum Where { ?id_taxon scientific_name ?sname. ?sname same_as ?dbname. ?dbname dbpedia-owl:kingdom ?kingdom. ?dbname dbpedia-owl:phylum ?phylum. }

Figure 9.
SPARQL sub-queries expressed in terms of Eos

Other interesting DTWs have been submitted to our architecture to explore trophic chain hierarchies and temporal space characteristics (Moura *et al.*, 2012). The query Q_2: "Get the sample species found at Paqueta Island in 2004, their synonyms and predators", is such an example.

Figure 11 presents the DTW produced for Q_2. It shows that the tuples retrieved from each endpoint by QEF-LD (i.e. the samples collected in 2004) are unified (*union* operator), but only those of the "Paqueta" region are of the user's interest. The relationship *has_predator* (Figure 6) is recursively exploited along the trophic chain hierarchy of each taxon retrieved, together with its synonym through the relationship (*synonymous_ of*). Final results are depicted in Figure 12.

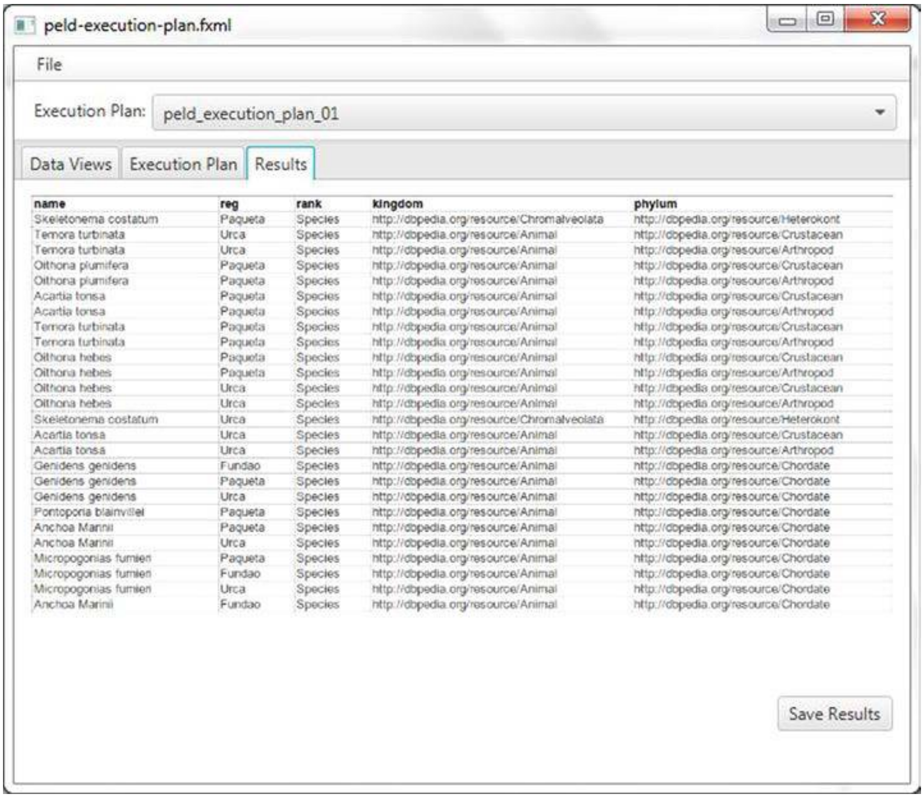
6.1 Experiments

Experiments carried out in this work consisted of evaluating our query engine (QEF-LD) concerning its ability to execute data views expressed as DTW. To quantitatively evaluate our approach, we performed experiments using QEF-LD and the most widely used tools to run federated SPARQL queries: Jena, Sesame and FedX. This section discusses the results obtained from the experiments that have been carried out. We used efficiency as the metric related to query processing time, and the memory footprint to evaluate each SPARQL query processor.

To carry out the tests, we used the following datasets available as local SPARQL endpoints: *Plankton_sample, Pl_analysis, Catfish_sample, Comm.Fish_Sample* e

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)



name	reg	rank	kingdom	phylum
Skeletonema costatum	Paqueta	Species	http://dbpedia.org/resource/Chromalveolata	http://dbpedia.org/resource/Heterokont
Temora turbinata	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Temora turbinata	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Oithona plumifera	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Oithona plumifera	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Acartia tonsa	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Acartia tonsa	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Temora turbinata	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Temora turbinata	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Oithona hebes	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Oithona hebes	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Oithona hebes	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Oithona hebes	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Skeletonema costatum	Urca	Species	http://dbpedia.org/resource/Chromalveolata	http://dbpedia.org/resource/Heterokont
Acartia tonsa	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Crustacean
Acartia tonsa	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Arthropod
Genidens genidens	Fundao	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Genidens genidens	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Genidens genidens	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Pontoporia blainvillei	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Anchoa Marinii	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Anchoa Marinii	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Micropogonias fumieri	Paqueta	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Micropogonias fumieri	Fundao	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Micropogonias fumieri	Urca	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate
Anchoa Marinii	Fundao	Species	http://dbpedia.org/resource/Animal	http://dbpedia.org/resource/Chordate

Figure 10.
Final query results for Q_1

Comm_analysis. DBPedia SPARQL endpoint was the only remote dataset used in our experiments. OpenLink Virtuoso[18] was used to store the RDF data and to provide the endpoint services to local datasets. The workload comprised two queries Q_1 and Q_2, as presented before. The queries evaluate the *join, union* and *project* algebraic operators. Query Q_1 makes use of local and remote datasets, while query Q_2 uses only local datasets. To measure efficiency, we submitted ten executions for each one of the two queries in the workload.

Three nodes comprised the test environment: a local server, a remote server and a client machine, and a local network connected the client and the local server. The local server hosted the OpenLink Virtuoso, which stored the RDF data and provided a SPARQL endpoint service to each local dataset used in the workload. The remote server hosted the DBPedia dataset. The client machine hosted the evaluated SPARQL query engines: Jena 2.12.1, Sesame 2.7.1, FedX 3.1 and QEF-LD 1.1. The local server machine used in the experiments was an Intel Core i7 2.93GHz with 16 GB RAM DDR3 1333 MHz. The client machine used during the tests was an Intel Core2 Quad 2.40GHz with 8GB RAM 667 MHz.

To evaluate the efficiency of the SPARQL query engines, we used two metrics:

(1) the query response time; and

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

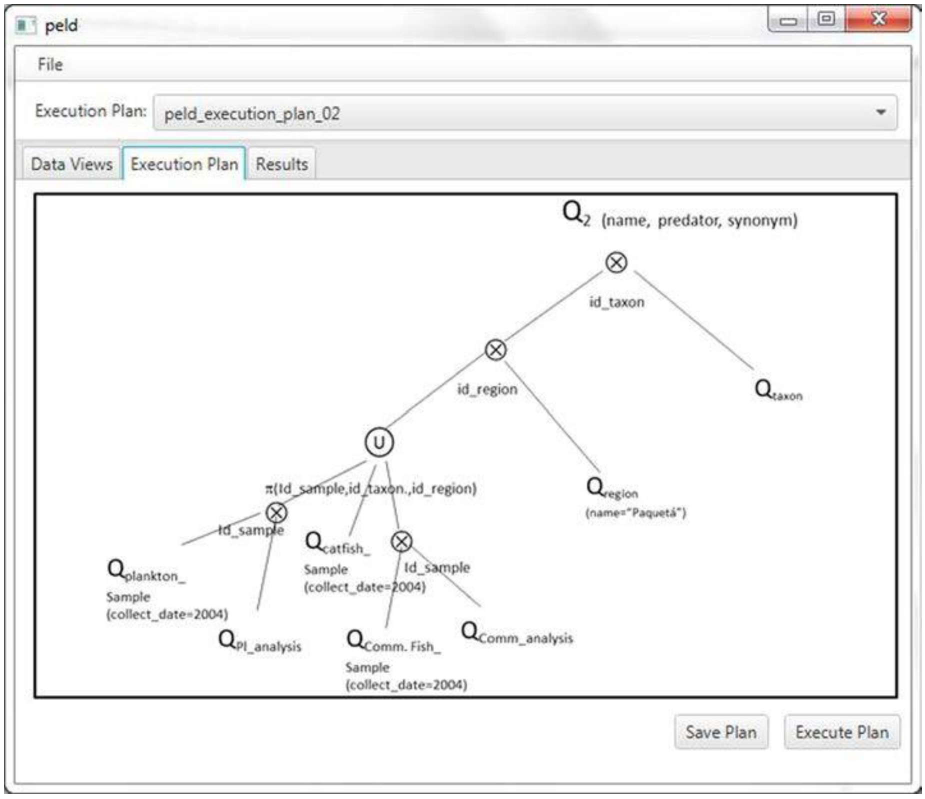

peld
File
Execution Plan: peld_execution_plan_02
Data Views
Execution Plan
Results
Q2 (name, predator, synonym)
id_taxon
id_region
Qtaxon
π(Id_sample,id_taxon.,id_region)
Qregion (name="Paquetá")
Id_sample
Qplankton_Sample (collect_date=2004)
Qcatfish_Sample (collect_date=2004)
Id_sample
QPl_analysis
QComm. Fish_Sample (collect_date=2004)
QComm_analysis
Save Plan
Execute Plan

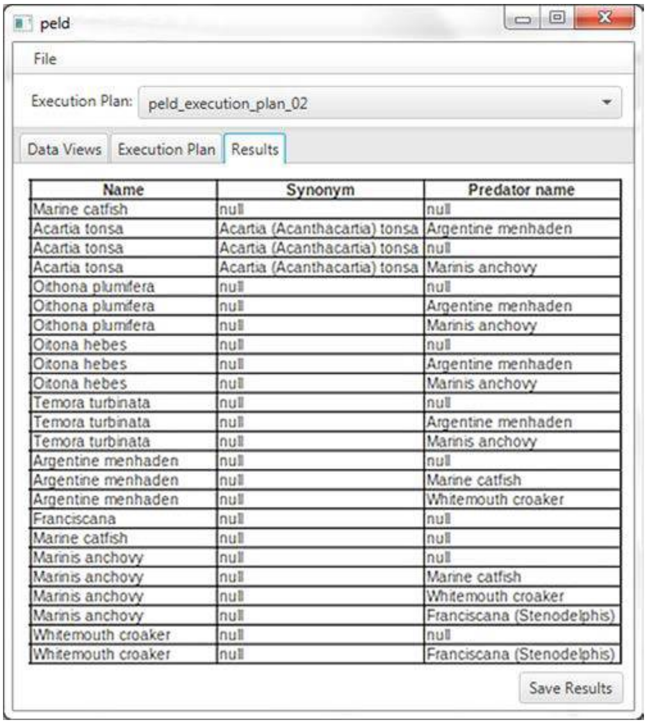
Figure 11.
Query execution plan for Q_2

(2) the maximum amount of memory used by the Java virtual machine during each query run.

Figure 13 shows the average query response times obtained for running ten executions of queries Q_1 and Q_2. For query Q_1 QEF-LD obtained considerably smaller query response times than the other evaluated SPARQL query engines, although QEF-LD demanded more time than FedX to run query Q_2. FedX and Sesame were substantially slower than Jena and QEF-LD in query Q_1 due to the ordering of query plan operations defined by them. While Jena, Sesame, and QEF-LD returned 27 results from the execution of query Q_1, FedX returned an incomplete result with only 11 tuples. All engines achieved complete results (17 tuples) from query Q_2. FedX got the smaller query time for query Q_2.

Figure 14 illustrates the average amount of memory consumed in ten executions for queries Q_1 and Q_2. QEF-LD consumed slightly more memory than the other engines in queries Q_1 and Q_2. It happened due to the multiple threads used by the *join* and *union* operators performed by QEF-LD (Magalhães *et al.*, 2013), which increases memory consumption. FedX stood out by the highest use of memory in query Q_1. FedX makes some modifications in the execution plan that can substantially improve or even worsen memory usage and execution time of queries.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)



Execution Plan: peld_execution_plan_02

Name	Synonym	Predator name
Marine catfish	null	null
Acartia tonsa	Acartia (Acanthacartia) tonsa	Argentine menhaden
Acartia tonsa	Acartia (Acanthacartia) tonsa	null
Acartia tonsa	Acartia (Acanthacartia) tonsa	Marinis anchovy
Oithona plumifera	null	null
Oithona plumifera	null	Argentine menhaden
Oithona plumifera	null	Marinis anchovy
Oitona hebes	null	null
Oitona hebes	null	Argentine menhaden
Oitona hebes	null	Marinis anchovy
Temora turbinata	null	null
Temora turbinata	null	Argentine menhaden
Temora turbinata	null	Marinis anchovy
Argentine menhaden	null	null
Argentine menhaden	null	Marine catfish
Argentine menhaden	null	Whitemouth croaker
Franciscana	null	null
Marine catfish	null	null
Marinis anchovy	null	null
Marinis anchovy	null	Marine catfish
Marinis anchovy	null	Whitemouth croaker
Marinis anchovy	null	Franciscana (Stenodelphis)
Whitemouth croaker	null	null
Whitemouth croaker	null	Franciscana (Stenodelphis)

Figure 12.
Final integrated results for Q_2

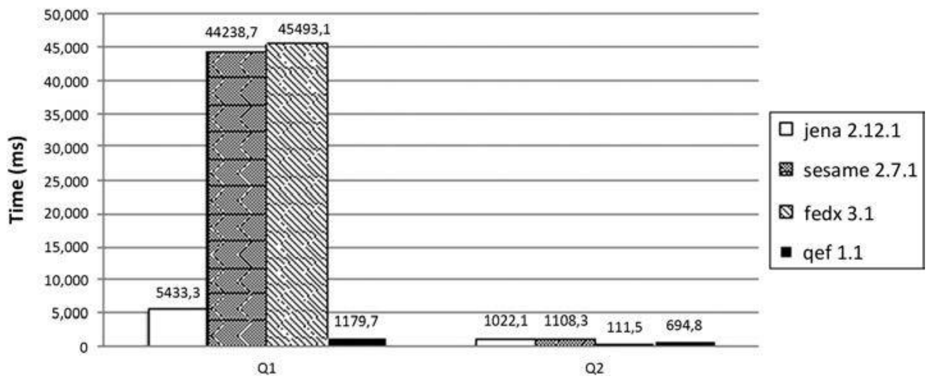


Figure 13.
Execution times for queries Q_1 and Q_2

7. Conclusion

Especially in life sciences (Knoblock *et al.*, 2012; Heath and Bizer, 2011), the world-wide exchange of research data between scientists becomes crucial. The strength and diversity of the ecosystems that have evolved in these cases demonstrate a previously unrecognized, and certainly unfulfilled, demand for access to data.

Data integration is, however, a complex endeavor, in part, due to the huge volume of heterogeneity of autonomous defined data. The lack of standards implies an adapted

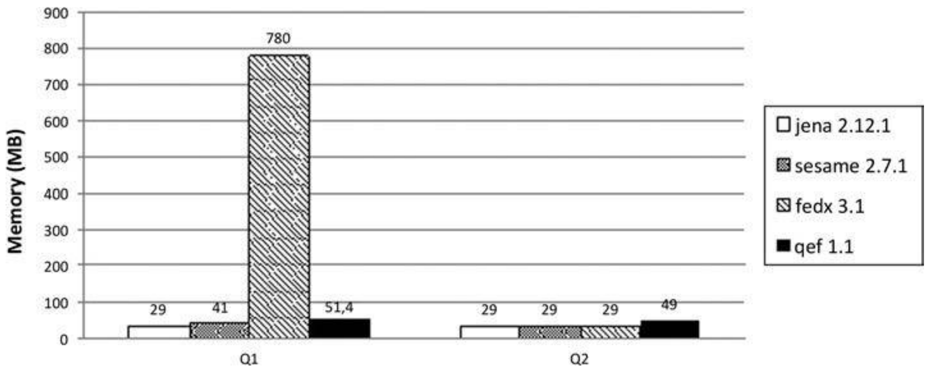


Figure 14. Memory usage for queries Q_1 and Q_2

solution for each scientific data integration initiative. In this context, LD rises as an interesting technique to apply the general architecture of the Web to the task of sharing structured data on a global scale. Publishing data according to the LD best practices solves part of the integration problem, which is to make data available in a common format (Auer *et al.*, 2014). Further effort is needed to place data into a common understandable model. Ontologies come as a rescue ground from which integration becomes possible, as they provide a common vocabulary to be shared among the different data sources.

This paper reports on a data integration framework and on its ability to answer queries expressed as data views over heterogeneous resources, represented as LD. Queries are processed as workflows by an extensible workflow query engine (QEF-LD). The major contribution of this work is focused on the semantic integration approach that simplifies the integration process both in terms of mappings and query answering through data views. The mapping complexity across the architecture layers is traded by the process of transforming the data sources into an exported ontology in RDF, thus reducing the problem of integration to that of mappings between homogeneous ontologies. A complete data integration scenario is discussed based on the challenges involved in publishing ecological data produced by the PELD Guanabara project, in Brazil. to evaluate QEF performance in the integration process, some queries have been executed using other query engines, with good results.

The experiment results carried out during this work indicate that the proposed data integration architecture is promising, and can be adopted as a standard for more complex ecological database integration scenarios. For future work, we intend to apply the customized RDB-RDF mappings (Vidal *et al.*, 2013) to generate EOs more efficiently in our architecture, in a virtual and/or materialized approach, according to the user's application needs.

Notes

1. GBIF: Web site for the Global Biodiversity Information Facility (GBIF): www.gbif.org/ (accessed May 2014).
2. PELD Brasil: www.icb.ufmg.br/peld (accessed May 2014).
3. Guanabara PELD: www.lncc.br/peldguanabara/index.php (accessed May 2014).

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

4. Jena and Sesame are SPARQL query processors.
5. ARQ Jena: http://jena.apache.org/documentation/query/ (accessed April 2014).
6. SemWIQ is no longer maintained and its last update was in 2010.
7. Sesame: www.openrdf.org/ (accessed April 2014).
8. CRIA – Reference Center of Environmental Information (in Portuguese): www.cria.org.br/ (accessed May 2014).
9. SpeciesLink: http://splink.cria.org.br (accessed April 2014).
10. INCT – Virtual herbarium of Flora and Fungus (in Portuguese): inct.florabrasil.net/herbario-virtual/ (accessed May 23 2014).
11. Sinbiota: sinbiota.biota.org.br/ (accessed May 2014).
12. SiBBr – System for Brazilian Biodiversity: www.sibbr.gov.br/ (accessed May 2014).
13. OWL: Web Ontology Language Overview, 2004, www.w3.org/TR/owl-features/ (accessed May 2014).
14. DBpedia: dbpedia.org/ (accessed April 2014).
15. RDB2RDF: RDB2RDF Working Group Charter, 2011, www.w3.org/2011/10/rdb2rdf-charter.html (accessed May 2014).
16. Postgresql: PostgreSQL beta release, www.postgresql.org/ (accessed April 2013).
17. D2RQ: The D2RQ Mapping Language, http://d2rq.org/d2rq-language (accessed May 2014).
18. OpenLink Virtuosohttp://virtuoso.openlinksw.com/

References

Angele, J. and Gesman, M. (2006), "Data integration using semantic technology: a use case", *Proceedings of the 2nd International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML'06), Athens, GA*, pp. 58-66.

Auer, S., Bryl, V. and Tramp, S. (Eds) (2014), "Linked open data – creating knowledge out of interlinked data – results of the LOD2 project", *Lecture Notes on Computer Science (LNCS)*, Springer.

Barret, T., Jones, D., Yuan, J. and Uschold, M. (2005), "Applying semantic web technology to the integration of corporate information", *International Journal of Web Engineering and Technology*, Vol. 2, Nos 2/3.

Berners-Lee, T. (2006), "Linked data - design issues", available at: www.w3.org/DesignIssues/LinkedData.html (accessed April 2014).

Bizer, C., Health, T. and Berners-Lee, T. (2006), "D2R Server – publishing relational databases on the Web as SPARQL endpoints", *Proceedings of the 15th International World Wide Web Conference*, *Edinburgh*.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M. and Rosati, R. (2007), "Tractable reasoning and efficient query answering in description logics: the DL-Lite family", *Journal of Automated Reasoning*, Vol. 39 No. 3, pp. 385-429.

Civili, C., Console, M., De Giacomo, G., Lembo, D., Lenzerini, M., Lepore, L., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Santarelli, V. and Savo, D.F. (2013), "MASTRO STUDIO: managing ontology-based data access applications", *Proceedings of the VLDB Endowment (PVLDB)*, Vol. 6 No. 12, pp. 1314-1317.

Cruz, I.F. and Xiao, H. (2005), "The role of ontologies in data integration", *Journal of Engineering Intelligent Systems*, Vol. 13 No. 4, pp. 245-252.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Cruz, I.F. and Xiao, H. (2009), "Ontology driven data integration in heterogeneous networks", *Complex Systems in Knowledge-based Environments: Theory, Models and Applications Studies in Computational Intelligence*, Vol. 168, pp. 75-98.

Daltio, J. and Bauzer, M.C. (2008), "Aondê: an ontology web service for interoperability across biodiversity applications", *Information Systems*, Vol. 33, pp. 724-753.

Elmasri, R. and Navathe, S.B. (2010), *Fundamentals of Database Systems*, 6th ed., Pearson Benjamin-Cummings.

Fegraus, E.H., Andelman, S., Jones, M.B. and Schildhauer, M. (2005), "Maximizing the value of ecological data with structured metadata: an introduction to Ecological Metadata Language (EML) and principles for metadata creation", *Bulletin of the Ecological Society of America*, Vol. 86 No. 3.

Goble, C. and Stevens, R. (2008), "The state of the nation in data integration", *Journal of Biomedical Informatics*, Vol. 41 No. 5, pp. 687-693.

Gruber, T. (1995), "Towards principles for the design of ontologies used for knowledge sharing", *International Journal of Human-Computer Studies*, Vol. 43 Nos 5/6, pp. 907-928.

Guarino, N. (1998), "Formal ontology and information systems", *Proceedings of Formal Ontology in Information Systems (FOIS)*, Trento.

Heath, T. and Bizer, C. (2011), "Linked data: evolving the Web into a global data space", *Synthesis Lectures on the Semantic Web: Theory and Technology*, Vol. 1 No. 1, pp. 1-136.

Knoblock, A., Szekely, C.A., Ambite, P., Goel, J.L., Gupta, A., Lerman, S., Muslea, K., Taheriyan, M. and Mallick, P. (2012), " Semi-automatically mapping structured sources into the semantic web", *The Semantic Web: Research and Applications, Lecture Notes in Computer Science*, Vol. 7295, pp. 375-390.

Langegger, A., Wöß, W. and Blöchl, M. (2008), " Semantic web middleware for virtual data integration on the web", *Proceedings of the 5th European Semantic Web Conference (ESWC), Springer Verlag*, pp. 493-507.

Leinfelder, B., Tao, J., Costa, D., Jones, M.B., Servilla, M., O'Brien, M. and Bur, T.C. (2010), "A metadata-driven approach to loading and querying heterogeneous scientific data", *Ecological Informatics*, Vol. 5, pp. 3-8.

Lenzerini, M. (2002), "Data integration: a theoretical perspective", *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pp. 233-246.

Madin, J., Bowers, S., Schildhauer, M., Krivov, S., Pennington, D. and Villa, F. (2007), "An ontology for describing and synthesizing ecological observation data", *Ecological Informatics*, Vol. 2, pp. 279-296.

Magalhães, R.P., Monteiro, J.M., Vidal, V.M.P., Macêdo, J.A.F., Maia, M., Porto, F. and Casanova, M.A. (2013), "QEF-LD – a query engine for distributed query processing on linked data", *15th International Conference on Enterprise Information Systems (ICEIS)*, Vol. 1, pp. 185-192.

Malhotra, A. (2005), "W3C RDB2RDF incubator group report", available at: www.w3.org/2005/Incubator/rdb2rdf/XGR-rdb2rdf-20090126/ (accessed April 2013).

Manola, F. and Miller, E. (2004), "RDF primer", *W3C Recommendation*, available at: www.w3.org/TR/2004/REC-rdf-primer-20040210/ (accessed May 2014).

Mittermeier, R.A., Gil, P.R. and Mittermeier, C.G. (1997), "Megadiversity: earth's biologically wealthiest nations", *Cemex*, 1st ed., Mexico (in Spanish).

Moura, A.M.C., Porto, F., Poltosi, M., Palazzi, D., Magalhães, R.P. and Vidal, V.M.P. (2012), "Integrating ecological data using linked data principles", *Proceedings of Joint V Seminar on Ontology*

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Research in Brazil and VII International Workshop on Metamodels, Ontologies and Semantic Technologies (ONTOBRAS-MOST), Recife, pp. 156-167.

Ngomo, A.-C. and Auer, S. (2011), “LIMES – a time-efficient approach for large-scale link discovery on the web of data”, *Proceeding of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2312-2317.

Noy, N.F. (2004), “Semantic integration: a survey of ontology-based approaches”, *SIGMOD Record*, Vol. 33 No. 4.

Ogasawara, E., Dias, J.S., Chirigati, V., Oliveira, D., Porto, F., Valduriez, P. and Mattoso, M. (2013), “Chiron: a parallel engine for algebraic scientific workflows”, *Concurrency and Computation: Practice and Experience*, Vol. 25 No. 16, pp. 2327-2341.

Patton, E.W., Seyed, P., Wang, P., Fu, L., Dein, F.J., Bristol, R.S. and McGuiness, D.L. (2014), “SemantEco: a semantically powered modular architecture for integrating distributed environmental and ecological data”, *Future Generation Computing Systems*, pp. 36430-36440.

Porto, F., Tajmouati, O., Silva, V.F.V., Schulze, B. and Ayres, F.V.M. (2007), “QEF – supporting complex query applications”, *7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid), Brazil*, pp. 846-851.

Prud’hommeaux, E. and Seaborne, A. (2008), “Sparql query language for RDF”, *W3C Recommendation*, available at: www.w3.org/TR/rdf-sparql-query/ (accessed April 2014).

Quilitz, B. and Leser, U. (2008), “Querying distributed RDF data sources with SPARQL”, *Proceedings of the 5th European Semantic Web Conference (ESWC)*, *Springer Verlag*, pp. 524-538.

Schultz, A., Matteini, A., Isele, R., Mendes, P., Bizer, C. and Becker, C. (2012), “LDIF – a framework for large-scale linked data integration”, *21st International World Wide Web Conference WWW2012*.

Schwarte, A., Haase, P., Hose, K., Schenkel, R. and Schmidt, M. (2011), “Fedx: optimization techniques for federated query processing on linked data”, *Proceedings of the 10th International Conference on the Semantic Web – Vol. Part I. ISWC’11*, *Springer-Verlag, Berlin, Heidelberg*, pp. 601-616.

Vidal, V.M.P., Casanova, M.A. and Neto, L.E. (2013), “Towards automatic generation of R2ML Mappings”, *European Semantic Web Symposium (ESWS)*, Montpellier.

Vidal, V.M.P., Macedo, J.A.F., Pinheiro, J.C., Casanova, M.A. and Porto, F. (2011), “Query processing in a mediator based framework for linked data integration”, *Intenational Journal of Business Data Communications and Networking (IJBDCN)*, Vol. 7 No. 2, pp. 29-47.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S. (2001), “Ontology-based integration of information – a survey of existing approaches”, *International Joint Conferences on Artificial Intelligence (IJCAI) Workshop on Ontologies and Information Sharing*.

Wieczorek, J., Bloom, D., Guralnick, R., Blum, S., Döring, M., Giovanni, R., Robertson, T. and Vieglais, D. (2012), “Darwin core: an evolving community-developed biodiversity data standard”, *PLoS ONE*, Vol. 7 No. 1, p. e29715.

Wiederhold, G. (1992), “Mediators in the architecture of future systems”, *Computer*, Vol. 25 No. 3, pp. 38-49.

Further reading

Higgins, D., Berkley, C. and Jones, M.B. (2002), “Managing heterogeneous ecological data using morpho”, *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM’02)*, *IEEE*, pp. 69-76.

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Appendix 1

Based on the Union and Join lists created by Algorithm 1, the goal of this one is to create the query plan (DTW) for the user's data view. The strategy adopted in this algorithm is the following: traverser the join list to mount subexpressions for building the final DTW. It first identifies the type of each operator (*join* or *merge*), and keeps it in a variable "join" (Lines 1-5). The construction of the query plan is based on the analysis of the left and right operands of the "join" operator node. The algorithm initially obtains from the Union list the root node that corresponds to the left branch of the current item (j) in the join list. If it is a Union, a function is executed to get the union operand from the Union list (Algorithm 5) and to create the corresponding subexpression (Lines 8-10). Otherwise, the current item (j) corresponds to a class, and a function is executed to obtain this class in the Union list (Algorithm 6). Then a subexpression is created and it is assigned as a left operand of the join operator node (Line 12). The same procedure is performed to mount subexpressions for the right join operand (Lines 15-20). From the subexpressions created from the left and right operands for each item (j) traversed in the join list (Lines 21-24), the algorithm returns to Algorithm 1 the final DTW (Line 27).

Algorithm 4. BuildDTW (unionList, joinList)

```
1   for each j in joinList /* traverses the joinList mounting subexpressions of the DTW */
2       if (j.operationType = "join")
3           join ← createJoinOperator(j) /* creates the physical join operator */
4       else
5           join ← createMergeOperator(j) /* creates the physical merge operator
    (of type join) */
6       end-if
7       unionNode ← null
8       obtain unionNode from unionList such that unionNode.root= j.left /* check whether
    left operand is union */
9       if (unionNode ≠ null)
10          leftOperand ← obtainUnionOperand (j.left, subExpressionList, unionList)
11      else
12          leftOperand ← obtainClassOperand (j.left, subExpressionList) /* left operand is a
    class */
13      end-if
14  unionNode ← null
15  obtain unionNode from unionList such that unionNode.root= j.right /* check whether
    right operand is union */
16      if (unionNode ≠ null)
17          rightOperand ← obtainUnionOperand (j.right, subExpressionList, unionList)
18      else
19          rightOperand← obtainClassOperand (j.right, subExpressionList) /* right operand
    is class */
20      end-if
21      join.leftOperator(leftOperand) /*add left operand to join operator */
22      join.rightOperator(rightOperand) /*add right operand to join operator */
23      subExpression ← createSubExpression(join) /* create a subExpression with the new
    join operator */
24      subExpressionList.add(subExpression) /* add the subExpression to the list */
25  end-for
26  DTW ← subExpressionList.pop() /* obtain the final DTW */
27  return DTW
```

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Algorithm 5. ObtainUnionOperand (Class c, List subExpressionList, List unionList)

```
/* Build a subexpression where the left operand of the plan is a union node */
 1  obtain unionNode from unionList such that unionNode.root= c
 2  subExpression← verifyUnioninSubExpression(unionNode) /* checks if union is in
    expessionList */
 3  if (subExpression = null) /* verifies whether unionNode is already in subexpression */
 4     union←createUnionOperator(unionNode) /* if it is not in subexpression add the
    unionNode to the join operand */
 5     expression ← union
 6  else
 7     expression ← subExpression /* associates the subexpression containing union to
       the
    return expression */
 8     subExpressionList.remove(subExpression) /* the previous expression is eliminated */
 9  end-if
10  return expression
```

Algorithm 6. ObtainClassOperand(Class c, list subExpresionList)

```
/* Build a subexpression where the left branch is a class */
 1  obtain subExpresion from subExpressionList such that subExpression contains c;
 2     if (subExpression ≠ null) /* checks if the class is in a subexpression */
 3        expression ← subExpression
 4        subExpressionList.remove(subExpression)
 5     else
 6        expression ← c /* if class is not in a subexpression it is returned as an operand */
 7     end-if
 8     return expression
```

Appendix 2

The code below presents an excerpt of a DTW XML script submitted to QEF-LD according to the sequence order of operations produced by the algorithm above.

```
    <?xml version="1.0" encoding="UTF-8" ?>
    - <!--
      ****************************************************************************************************
    The local Initial Query Execution Plan (QEP), ie.no remote operations
      ****************************************************************************************************
      helpdesk@linea.gov.br
     -->
-<QEPTemplatexmlns="http://giga03.lncc.br/DIP/WP4/CoDIMS-D"xmlns:op="http://giga03.lncc.br/DIP/WP4/CoDIMS-
      D/Operator"xmlns:qep="http://giga03.lncc.br/DIP/WP4/CoDIMS-D/QEP">
-<qep:QEP type="Initial">
-<op:Operator id="1" prod="2" type="">
 <Name>Project</Name>
-<ParameterList>
 <Variables>name,reg,rank,kingdom,phylum</Variables>
     </ParameterList>
     </op:Operator>
-<op:Operator id="2" prod="4,3" type="" parallelizable="true">
 <Name>SetBindJoin</Name>
-<ParameterList>
 <maxActiveThreads>0</maxActiveThreads>
 <blockSize>10</blockSize>
     </ParameterList>
     </op:Operator>
-<op:Operator id="3" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://dbpedia.org/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
                 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
                   select distinct * where{
                   ?id_dbpediadbpedia-owl:kingdom ?kingdom .
                   ?id_dbpediadbpedia-owl:phylum ?phylum .
                                                          }
     ]]>
     </SPARQLQuery>
     </ParameterList>
     </op:Operator>
-<op:Operator id="4" prod="6,5" type="" parallelizable="true">
 <Name>SetBindJoin</Name>
-<ParameterList>
 <maxActiveThreads>0</maxActiveThreads>
 <blockSize>10</blockSize>
     </ParameterList>
     </op:Operator>
-<op:Operator id="5" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://localhost:3030/Peld-Taxon/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
        PREFIX reg: <http://localhost:2024/resource/vocab/>
          PREFIX owl:<http://www.w3.org/2002/07/owl#>
                   select distinct * where{
                   ?id_taxonreg:peld_taxon_scientific_name  ?name .
                   ?id_taxonreg:peld_taxon_rank  ?rank.
                   ?id_taxonowl:sameAs ?id_dbpedia .
                   }
     ]]>
     </SPARQLQuery>
```

(*continued*)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

```
     </ParameterList>
     </op:Operator>
-<op:Operator id="6" prod="8,7" type="" parallelizable="true">
 <Name>SetBindJoin</Name>
-<ParameterList>
 <maxActiveThreads>0</maxActiveThreads>
 <blockSize>10</blockSize>
     </ParameterList>
     </op:Operator>
-<op:Operator id="7" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://localhost:3030/Peld-Region/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
        PREFIX rg: <http://localhost:2023/resource/vocab/>
                  select distinct * where{
                  ?id_regrg:peld_region_name_reg ?reg .}
     ]]>
     </SPARQLQuery>
     </ParameterList>
     </op:Operator>
-<op:Operator id="8" prod="9,10,11" type="">
 <Name>Union</Name>
-<ParameterList>
 <useThreads>true</useThreads>
     </ParameterList>
     </op:Operator>
-<op:Operator id="9" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://localhost:2020/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
        PREFIX epl: <http://localhost:2020/resource/vocab/>
                  select distinct ?id_taxon ?id_reg where{
                  ?sepl:peld_analysis_id_taxon ?id_taxon.
                  ?sepl:peld_analysis_id_collect ?id_an.
                  ?id_anepl:peld_collect_local ?id_reg.
                  FILTER EXISTS { ?id_taxonepl:peld_taxon_rank  "Species" } .
                  }
        ]]>
     </SPARQLQuery>
     </ParameterList>
     </op:Operator>
-<op:Operator id="10" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://localhost:2021/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
        PREFIX ecf: <http://localhost:2021/resource/vocab/>
                  select distinct ?id_taxon ?id_reg where {
                  ?secf:peld_id_taxon ?id_taxon.
                  ?secf:peld_collect_local ?id_reg.
                  FILTER EXISTS { ?id_taxonecf:peld_taxon_rank  "Species" } .
                  }
```

(*continued*)

```
        ]]>
        </SPARQLQuery>
        </ParameterList>
        </op:Operator>
-<op:Operator id="11" prod="0" type="Scan"numberTuples="?">
 <Name>Service</Name>
-<ParameterList>
 <DataSourceName>SparqlEndpoint</DataSourceName>
 <ServiceURI>http://localhost:2022/sparql</ServiceURI>
    -<SPARQLQuery>
    - <![CDATA[
           PREFIX eco: <http://localhost:2022/resource/vocab/>

     select distinct ?id_taxon ?id_reg where {
     ?seco:peld_fish_analysis_id_taxon ?id_taxon.
     ?seco:peld_analysis_id_collect ?id_an .
     ?id_aneco:peld_fish_local_collect ?id_reg.
                  FILTER EXISTS { ?id_taxoneco:peld_taxon_rank  "Species" } .
                  }
        ]]>
        </SPARQLQuery>
        </ParameterList>
        </op:Operator>
        </qep:QEP>
        </QEPTemplate>
```

About the authors

Ana Maria de Carvalho Moura is a Professor at Extreme Data Lab (DEXL), National Laboratory of Scientific Computing (LNCC). Ana Maria de Carvalho Moura is the corresponding author and can be contacted at: anamaria.moura@gmail.com

Fabio Porto is a Professor at Extreme Data Lab (DEXL), National Laboratory of Scientific Computing (LNCC).

Vania Vidal is a Professor at Department of Computing, Federal University of Ceará (UFC).

Regis Pires Magalhães is a Professor and Graduate student at Department of Computing, Federal University of Ceará (UFC).

Macedo Maia is a graduate student at Department of Computing, Federal University of Ceará (UFC).

Maira Poltosi is a Researcher at Extreme Data Lab (DEXL), National Laboratory of Scientific Computing (LNCC).

Daniele Palazzi is a Researcher at Extreme Data Lab (DEXL), National Laboratory of Scientific Computing (LNCC).

For instructions on how to order reprints of this article, please visit our website:
www.emeraldgrouppublishing.com/licensing/reprints.htm
Or contact us for further details: **permissions@emeraldinsight.com**

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)

This article has been cited by:

1. Dhomas Hatta Fudholi, Wenny Rahayu, Eric PardedeOntology-Based Information Extraction for Knowledge Enrichment and Validation 1116-1123. [CrossRef]

Downloaded by TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES At 23:03 01 November 2016 (PT)