MatWare: Constructing and Exploiting Domain Specific Warehouses by Aggregating Semantic Data

Yannis Tzitzikas[1,2], Nikos Minadakis[1], Yannis Marketakis[1], Pavlos Fafalios[1,2], Carlo Allocca[1], Michalis Mountantonakis[1,2], Ioanna Zidianaki[1,2]

[1] Institute of Computer Science, FORTH-ICS, Greece
[2] Computer Science Department, University of Crete, Greece
{tzitzik,minadakn,marketak,fafalios,carlo,mountant,izidian}@ics.forth.gr

Abstract. In many applications one has to fetch and assemble pieces of information coming from more than one web sources such as SPARQL endpoints. In this paper we describe the corresponding requirements and challenges, based on our experience, and then we present a process and a tool that we have developed, called *MatWare*, for constructing such semantic warehouses. We focus on domain-specific warehouses, where the focus is given on the aspects of *scope control*, *connectivity assessment*, *provenance*, and *freshness*. *MatWare* (Materialized Warehouse) is a tool that automates the construction (and reconstruction) of such warehouses, and offers methods for tackling the aforementioned requirements. Finally we report our experiences from using it for building, maintaining and evolving an operational semantic warehouse for the marine domain, that is currently in use by several applications ranging from e-infrastructure services to smart phone applications.

Keywords: #eswc2014Tzitzikas

1 Introduction

An increasing number of datasets are publicly available in various formats (including Linked Data and SPARQL endpoints). For exploiting this wealth of data, and for building domain specific applications, one has to fetch and assemble pieces of information coming from more than one sources. These pieces can then be used for constructing a warehouse that offers more complete browsing and query services (in comparison to those offered by the underlying sources). For instance, in the marine domain, there is not any individual source that can answer queries of the form: *"Given the scientific name of a species, find the ecosystems, water areas and countries that this species is native to, and the common names that are used for this species in each of the countries".*

There are *domain independent* warehouses, like the Sindice RDF search engine [12], or the Semantic Web Search Engine (SWSE) [4], but also *domain specific*, like [14, 9, 5]. We focus on the requirements for building domain specific

warehouses. Such warehouses aim to serve particular needs, particular communities of users, consequently their "quality" requirements are higher. It is therefore worth elaborating on the process that can be used for building such warehouses, and on the related difficulties and challenges. In brief, and from our experience from running an operational semantic warehouse, the main questions and challenges include:

- How to define the objectives and the scope of such a warehouse and how to test that its contents meet the objectives?
- How to *connect* the fetched pieces of information? Common schemas, URIs or literals are not always there.
- How to measure the value of the warehouse as well as the quality of the warehouse (this is important for e-science)?
- How to keep such a warehouse fresh, i.e. how to automate its construction, and how to monitor its quality (as the underlying source change)?
- How to tackle the various issues of provenance that arise?

In this paper we present our experience on defining such a process and the tool that we have developed (*MatWare*) for realizing the process and running an operational semantic warehouse for marine resources. The rest of the paper is organized as follows: Section 2 describes the context, the main requirements, and related works. Section 3 describes the adopted integration approach for tackling the corresponding functional and non-functional requirements. Section 4 describes the tool *MatWare* and Section 5 describes how the *MatWare*-constructed warehouse is currently being used in various applications. Finally, Section 6 concludes the paper. More details are available in the web[1].

2 Context, Requirements and Related Work

2.1 Context and Requirements

Below we list the main functional and non functional requirements. The source of these requirements is the *iMarine project*[2] that offers an operational distributed infrastructure that serves hundreds of scientists from the marine domain. As regards semantic technologies, the objective is to integrate information from various marine sources, specifically from WoRMS[3], Ecoscope[4], FishBase[5], FLOD[6] and DBpedia[7].

Functional Requirements

F1 *Multiplicity of Sources.* Ability to access multiple sources (including SPARQL endpoints), get data from these sources, and ingest them to the warehouse.

[1] `http://www.ics.forth.gr/isl/MarineTLO` and `http://www.ics.forth.gr/isl/MatWare`

[2] FP7, Research Infrastructures, `http://www.i-marine.eu/`

[3] `http://www.marinespecies.org/`

[4] `http://www.ecoscopebc.ird.fr/EcoscopeKB/ShowWelcomePage.action`

[5] `http://www.fishbase.org/`

[6] `http://www.fao.org/figis/flod/`

[7] `http://dbpedia.org/`

F2 *Mappings, Transformations and Equivalences.* Ability to accommodate mappings, perform transformations and create `sameAs` relationships between the fetched content for connecting the corresponding schema elements and entities.

F3 *Reconstructibility.* Ability to reconstruct the warehouse periodically (from scratch or incrementally) for keeping it fresh.

Non Functional Requirements

N1 *Scope control.* Make concrete and testable the scope of the information that should be stored in the warehouse. Since we live in the same universe, everything is directly or indirectly connected, therefore without stating concrete objectives there is the risk of continuous expansion without concrete objectives regarding its contents, quality and purpose.

N2 *Connectivity assessment.* Ability to check and assess the connectivity of the information in the warehouse. Putting triples together does not guarantee that they will be connected. In general, connectivity concerns both schema and instances and it is achieved through common URIs, commons literals and `sameAs` relationships.

N3 *Provenance.* More than one levels of provenance can be identified and would be desired, e.g. provenance at triple level (from what source that triple was fetched), at URIs and values level (from what source we get a URI or value), or at query level (what sources are being used to answer a query).

N4 *Consistency and Conflicts.* Ability to specify the desired consistency, e.g. regarding the acceptance or rejection of different objects for a particular subject-predicate pair in a triple, or ability to accommodate different objects while making evident their provenance.

2.2 Related Approaches

Below we refer and discuss in brief the more related systems, namely *ODCleanStore* and *Sieve*.

ODCleanStore [11, 8, 7] is a tool that can download content (RDF graphs) and offers various transformations for cleaning it (deduplication, conflict resolution), and linking it to existing resources, plus assessing the quality of the outcome. It names *conflicts* the cases where two different quads (e.g. sources) have different object values for a certain subject s and predicate p. To such cases conflict resolution rules are offered that either select one or more of these conflicting values (e.g. ANY, MAX, ALL), or compute a new value (e.g. AVG). [7] describes various quality metrics (for scoring each source based on conflicts), as well for assessing the overall outcome.

Another related system is *Sieve* [10] which is part of the Linked Data Integration Framework (LDIF)[8]. That work also proposes metrics like *schema completeness* and *conciseness*. However, such metrics are not useful for the case of domain specific warehouses that have a top-level ontology, in the sense that the schema mappings and the transformation rules can tackle these problems. This is true

[8] `http://www4.wiwiss.fu-berlin.de/bizer/ldif/`

in our warehouse (it is also assumed in the scenarios of *ODCleanStore*). Finally, [1] contains an interesting discussion about completeness in query answering.

3 The Adopted Integration Approach

This section describes how we tackled the Functional and Non-Functional Requirements, as well the entire construction process.

3.1 Tackling the Functional Requirements (F1-F3)

To tackle the need for *multiplicity of sources* (F1) and *reconstructability* (F3), we developed a tool that can automate the construction (or reconstruction) of such warehouses (it is called `MatWare` and it is described in more detail in Section 4). Regarding *Mappings, Transformations and Equivalences* (F2), it is always a good practice to have (select or define) a top-level ontology as it alleviates the schema mapping effort (avoids the combinatorial explosion of pair-wise mappings). Moreover, since the entities (instances of schemas, URIs) have to be mapped too, there is a need for approaches that can automate this as much as possible. In general there is a need for rules that can create `sameAs` relationships. For this reason, `MatWare` exploits SILK[9] which is a tool for discovering relationships between data items within different Linked Data sources.

3.2 Scope Control

As regards *scope* (N1), we use the notion of *competency queries*. A competency query is a query that is useful for the community at hand, e.g. for a human member (e.g. a scientist), or for building applications for that domain. Therefore, a list of such queries can sketch the desired scope and the desired structuring of the information. Figure 1 displays the textual description of some competency queries as they were supplied by the marine community.

#Query	For a **scientific name** of a **species** (e.g. **Thunnus Albacares** or **Poromitra Crassiceps**), find/give me:
Q_1	the biological environments (e.g. **ecosystems**) in which the **species** has been **introduced** and more general descriptive information of it (such as the **country**)
Q_2	its **common names** and their complementary info (e.g. **languages** and **countries** where they are used)
Q_3	the **water areas** and their **FAO codes** in which the **species** is **native**
Q_4	the **countries** in which the **species lives**
Q_5	the **water areas** and the **FAO** portioning **code** associated with a country
Q_6	the presentation w.r.t **Country, Ecosystem, Water Area** and **Exclusive Economical Zone** (of the water area)
Q_7	the projection w.r.t. **Ecosystem** and **Competitor**, providing for each competitor the **identification information** (e.g. several codes provided by different organizations)
Q_8	a map w.r.t. **Country** and **Predator**, providing for each predator both the **identification information** and the **biological classification**
Q_9	**who** discovered it, in which **year**, the **biological classification**, the **identification information**, the **common names** - providing for each common name the **language** and the **countries** where it is **used in**.

Fig. 1: Some indicative competency queries.

These queries specify the required structuring of the ontology. It is a good practice to have a *top-level schema/ontology* not only for alleviating the schema

[9] `http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/`

mapping effort, but also for formulating the competency queries using that ontology (instead of using elements coming from the underlying sources, which change over time). For the iMarine project, and since there was not any ontology that allowed formulating the desired queries, we defined the top-level ontology called *MarineTLO*[10] [14].

After deciding the schema level, the next step is to specify the contents of the underling sources that should be fetched and stored (as they are, or transformed) to the warehouse. In many cases, this requires using various access methods (SPARQL endpoints, HTTP accessible files, JDBC) and specifying what exactly to get from each source (all contents, or a specific part). For instance, and for the case of the iMarine warehouse, we fetch all triples from FLOD through its SPARQL endpoint, all triples from Ecoscope obtained by fetching OWL files from its web page, information about species (ranks, scientific names and common names) from WoRMS accessed through the Species Discovery Service of the gCube infrastructure[11], information about species only from DBpedia's SPARQL endpoint, and finally information about species, water areas, ecosystems and countries from the relational tables of FishBase.

3.3 Connectivity Assessment

Connectivity (N2) has two main aspects: *schema* connectivity and *instance* connectivity. For the first, we use the top level ontology and *schema mappings* for associating the fetched data with the schema of the top level ontology[12]. Based on these we can *transform and ingest* the fetched data. Some data can be stored as they are fetched, while others have to be transformed, i.e. apply a *format* transformation and/or a *logical* transformation for being compatible with the top-level ontology.

As regards the connectivity of instances, one has to *inspect and test the connectivity* of the "draft" warehouse, i.e. the warehouse produced by ingesting the fetched information. This is done through the competency queries as well as through a number of *connectivity metrics* that we have defined. The main metrics are: (a) the *matrix of percentages of the common URIs and/or literals* (it shows the percentage of common URIs/literals between every pair of sources), (b) the *complementarity factor of the entities of interest* (it is the number of sources that provided unique triples for each entity of interest), (c) the table with the *increments in the average degree of each source* (it measures the increment of the graph-theoretic degree of each entity when it becomes part of the warehouse graph), and (d) the unique triple contribution of each source (the number of triples provided by a source which are not provided by any other source). The values of (a),(b),(c) allow valuating the warehouse, while (c) and (d) mainly

[10] Documentation and examples are available in `http://www.ics.forth.gr/isl/MarineTLO`.

[11] `https://i-marine.d4science.org/web/guest/about-gcube`

[12] In our case we use `rdfs:subClassOf`, `rdfs:subPropertyOf` and `owl:equivalentClass` properties.

concern each particular source. The metrics are elaborated in detail in [15], while an example is given in Figure 10. In comparison to the quality metrics introduced by other works (like those mentioned in Section 2.2) which focus more on conflicts, we focus on connectivity, which is important for a warehouse that has wide scope, rich structured conceptual model, and requirements for answering queries which contain "long" (not trivial) path expressions.

Based also on the results of the previous step, the next step is to *formulate rules for instance matching*, i.e. rules that can produce `sameAs` relationships for obtaining the desired connections. These rules are formulated by the curator by manually inspecting the contents of the sources. This is done only the first time; the formulated rules are automatically applied in the subsequent warehouse reconstructions. Specifically we *apply the instance matching rules* (SILK rules in our case) for producing (and then ingesting to the warehouse) `sameAs` relationships. Moreover we apply some transformation rules to further improve the connectivity of the warehouse, specifically for changing or enhancing the data which are fetched from different sources in order to comply to the ontology or to overcome limitations of inference.

Finally we have to test the produced repository and evaluate it. This is done again through the competency queries and the metrics. Specifically, by inspecting the proposed metric-based matrixes one can very quickly get an overview of the contribution of each source and the tangible benefits of the warehouse.

3.4 Provenance

As regards provenance (N3), we support four levels of provenance: (a) at *conceptual modeling* level, (b) at *URIs and values* level, (c) at *triple* level, and (d) at *query* level.

As regards conceptual level (level a), for the case of iMarine, part of the provenance requirements are covered by the ontology `MarineTLO`. Specifically `MarineTLO` models the provenance of species names, codes, etc. (who and when assigned them). Therefore there is no need for adopting any other conceptual model for modeling provenance (e.g. OPM[13]), also because the data fetched from the sources do not have any kind of provenance information, and the warehouse construction does not have any complex workflow that need special documentation through a provenance model.

As regards the level of *URIs and values* (level b), we adopt the namespace mechanism for reflecting the source of origin of an individual[14]. In addition, during the construction of a warehouse, there is the option of applying a uniform notation "@source" (where source can be FLOD, Ecoscope, WoRMS, FishBase or DBpedia) to literals. This notation is useful in querying for indicating how

[13] `http://openprovenance.org/`

[14] E.g. for the cases of Ecoscope, FLOD and DBpedia we use the namespaces, that these sources already provide, while for WoRMS we use `http://www.worms.org/entity#` as a namespace and for FishBase we use `http://www.fishbase.org/entity#`.

each result derived by the warehouse was produced. An example of this functionality in shown in Figure 2 for the case of the scientific name and authorship of a species. This notation allows asking source-centric queries in a relative simple way by using the SPARQL filter as follows: `FILTER(langMatches (lang(?literal), ‘‘source’’))`. The shortcoming of this approach is that it is not possible to store both the source and the language of a literal, and that before storing the data to the warehouse a data transformation step (i.e. the attachment of @source) is required. Although this approach has the above limitations and "misuses"the semantics of an RDF feature (the language tag), this level can be a reasonable choice in cases where all data should be stored in a single graph space and the storage of language is not required (e.g. if all literals are in one language). If these pre-conditions are not met, then one should use the *triple* level provenance, that is described below, since it overcomes these limitations.

scientificName	year	authority
"Thunnus albacares"@worms	"1788"@worms	"Bonnaterre"@worms
"Thunnus albacares"@dbpedia	"1788"@dbpedia	"Bonnaterre"@dbpedia

Fig. 2: Scientific Name and Authorship information of *Yellowfin Tuna*.

As regards the *triple* level (level c), we store the fetched (or fetched and transformed) triples from each source in a *separate graph space*. This is useful not only for provenance reasons, but also for refreshing parts of the warehouse, as well as for computing the connectivity metrics that were described earlier. Furthermore, and compared to level b, it leaves the data intact since there is no need to add any extra information about their provenance. Finally, the separate graph spaces are also useful for the fourth level described below.

As regards the *query* level (level d), ***MatWare*** offers a query rewriting functionality that exploits the contents of the graph spaces for returning the sources that contributed to the query results (including those that contributed to an intermediate step). Let q be a SPARQL query that has n parameters in the select clause and contains k triple patterns of the form ($?s_i$ $?p_i$ $?o_i$), e.g.:

```
SELECT ?o_1 ?o_2 ... ?o_n
WHERE {
    ?s_1 ?p_1 ?o_1. ?s_2 ?p_2 ?o_2. ... . ?s_k ?p_k ?o_k }
```

The rewriting produces a query q' that has $n + k$ parameters in the select clause: the original n variables plus one variable for each of the k triple patterns in the query. Specifically, for each triple pattern, say ($?s_i$ $?p_i$ $?o_i$), of the original query, we introduce a variable $?g_i$ (for getting the source of the triple) and in q' the triple pattern is replaced by the graph pattern $?g_i\{?s_i$ $?p_i$ $?o_i\}$. Eventually, the rewritten query will be:

```
SELECT ?o_1 ?o_2 ... ?o_n   ?g_1 ?g_2 ... ?g_k
WHERE {
   graph ?g_1 {?s_1 ?p_1 ?o_1}.
   graph ?g_2 {?s_2 ?p_2 ?o_2}.
```

...
graph ?g_k {?s_k ?p_k ?o_k}}

A real example follows. Consider the query *"For a scientific name of a species (e.g. Thunnus Albacares) find the FAO codes of the water areas in which the species is native"*. Its evaluation will return the corresponding FAO codes, information that obviously comes from FLOD. However the fact that *Thunnus Albacares is native in a specific Water Area* comes from Fishbase, which is a fact that the end user will not be aware of, if the corresponding graph space will not be returned. The upper part of Figure 3 shows the initial SPARQL query (which has 2 triple patterns), while the lower part shows the query as it has been derived after applying the rewriting described above. The answer of the last query is shown in Figure 4(left) which shows the sources that contributed to the result.

```
SELECT ?faocode WHERE {
 ?ecoscope:thunnus_albacares    marineTLO:isNativeAt                    ?waterarea.
 ?waterarea                     marineTLO:LXrelatedIdentifierAssignment ?faocode }

SELECT ?faocode  ?waterarea_graphspace  ?faocode_graphspace WHERE {
    graph ?waterarea_graphspace
        {ecoscope:thunnus_albacares marineTLO:isNativeAt                    ?waterarea }
    graph ?faocode_graphspace
        {?waterarea                 marineTLO:LXrelatedIdentifierAssignment ?faocode } }
```

Fig. 3: Rewriting a query for keeping the provenance of the intermediate results.

As another example, consider the query *"Find the Scientific Name of a Species"*. This query will return the scientific names of the species according to the various sources as shown in Figure 4(right).

faocode	waterarea_graphspace	faocode_graphspace
41	http://www.ics.forth.gr/isl/Fishbase	http://www.ics.forth.gr/isl/FLOD
47	http://www.ics.forth.gr/isl/Fishbase	http://www.ics.forth.gr/isl/FLOD
31	http://www.ics.forth.gr/isl/Fishbase	http://www.ics.forth.gr/isl/FLOD
34	http://www.ics.forth.gr/isl/Fishbase	http://www.ics.forth.gr/isl/FLOD

scientific_name	scName_graphspace
thunnus albacares	http://www.ics.forth.gr/isl/Fishbase
Thunnus albacares	http://www.ics.forth.gr/isl/DBpedia
Thunnus albacares	http://www.ics.forth.gr/isl/Worms
Thunnus albacares	http://www.ics.forth.gr/isl/Ecoscope

Fig. 4: Left: The results of the enhanced query (of Fig. 3), Right: Scientific Names (enriched with source provenance)

Intuitively, one can conceive the query evaluation process as a pipeline defined by the triple patterns, and the values shown in the additional (due to the rewriting) columns of the answer are the names of the graph spaces (in our setting this corresponds to sources) that contributed to each step of that pipeline.

3.5 Consistency and Conflicts

Instead of specifying or deciding how to cope with the different values coming from the sources (as in [8]), we instead (as shown in the provenance section) show to the user the information that is provided by each source. This is more transparent, and allows the involved authorities to spot (and hopefully fix) the various errors.

3.6 The Entire Process

Figure 5 sketches the construction process. The main effort has to be dedicated for setting up the warehouse the first time, since in that time one has to select/define the schema, the schema mappings, the instance matching rules, etc. Afterwards the warehouse is reconstructed periodically for getting refreshed content, without requiring human intervention. For *monitoring* the warehouse after reconstructing it, ***MatWare*** computes the connectivity metrics after each reconstruction. By comparing their values in the previous and new warehouse, one can understand whether a change in the underlying sources affected negatively the quality (e.g. connectivity) of the warehouse.

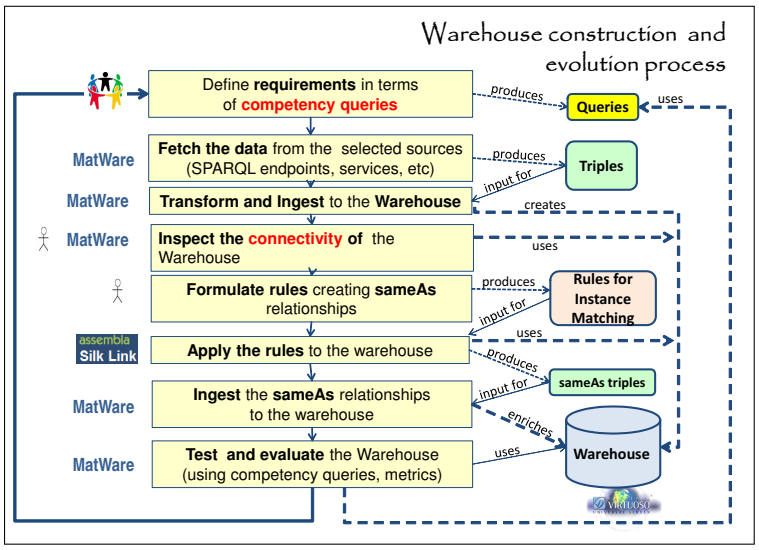


Fig. 5: The process for constructing and evolving the warehouse

For example, consider that we want to refresh the warehouse because the data coming from WoRMS have been changed and suppose that the schema of that source has not been changed. It is evident that we do not have to re-construct the warehouse from scratch since all the other sources will be the same. Instead we remove all the triples about WoRMS from the warehouse by removing them from the corresponding graph space. We also remove all same-as triples between WoRMS and any other source in the warehouse. In the sequel, we get the new contents for that source, ingest them to the warehouse and run again the steps for applying the transformation rules and the production of the same-as triples between the (new) contents of WoRMS and other sources. Finally we test and evaluate the warehouse as before. It is clear that if the schema of the source has been changed then we should also modify the mappings between that source and the top-level ontology.

4 The Warehouse Construction Tool *MatWare*

The main functionality of *MatWare* is the automatic creation and maintenance of a semantic warehouse. In brief, it is capable to: (a) download data from remote sources (e.g. FLOD, Ecoscope, WoRMS, FishBase, DBpedia), (b) create Virtuoso repositories, (c) ingest the data to the warehouse, (d) apply the necessary transformation rules to the data, (e) create `sameAs` links between the entities of the different sources, (f) create the inference ruleset, (g) refresh the repository, (h) run the competency queries, and (j) compute the connectivity metrics. It has been implemented in Java and it uses the Sesame/Virtuoso APIs. It has a modular architecture which is illustrated in Figure 6.

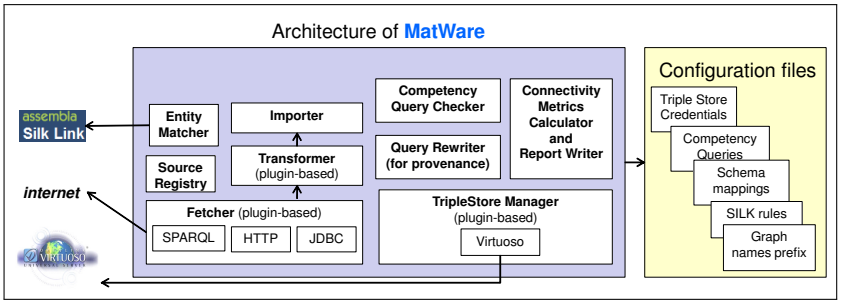


Fig. 6: The architecture of *MatWare*

Configurability is very important and in *MatWare* it is achieved by changing the context of an xml file (`config.xml`). To create a warehouse from scratch, one has to specify the type of the repository, the names of the graphs that correspond to the different sources, and the URL, username and password for connecting to the repository. These options are enough for creating the warehouse and importing the data from the sources. In addition, one can specify the next actions to be performed which include: downloading the data from each source (by providing the fetcher classes as plugins), execution of the transformation rules (by providing the transformer classes as plugins), creation of `sameAs` links between the entities of the various sources (by providing the SILK rules as xml files), calculation of the connectivity metrics, refreshing of the warehouse for the case of a source that changes, creation of the virtuoso ruleset, querying the repository, deletion of graphs. In addition one can specify the folder containing the locally stored content, and whether an existed graph should be overwritten or not.

In case one wants to add a new source, the following actions are needed: (a) include the fetcher class for the specific source as plug in, (b) provide the mapping files, (c) include the transformer class for the specific source as a plug in and (d) provide the SILK rules as xml files.

5 The Resulting Warehouse and its Current Exploitation

5.1 The resulted MarineTLO-based Warehouse

Here we discuss the *MarineTLO-based warehouse*[15], which is outcome of the above process carried out using ***MatWare***. Its first version is described in [14]. Now it is operational[16] and it is exploited in various applications. The objective of the warehouse is to provide a coherent set of facts about marine species. Just indicatively, Figure 7 illustrates some information about the species `Thunnus albacares` which are stored in different sources (here FLOD, Ecoscope, WoRMS, FishBase and DBpedia). These pieces of information are complementary and are assembled for enabling advanced browsing, querying and reasoning. This is also evident from Figure 8 which shows the underlying sources that contribute information regarding the main concepts of the MarineTLO ontology.

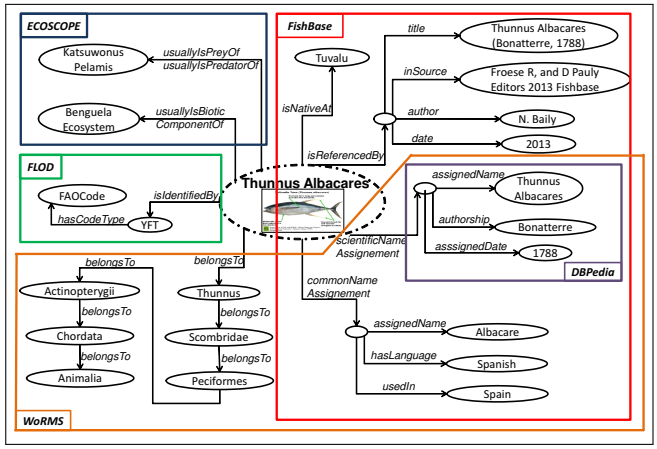


Fig. 7: Integrated information about `Thunnus albacares` from different sources

Concepts	Ecoscope	FLOD	WoRMS	DBpedia	FishBase
Species	✓	✓	✓	✓	✓
Scientific Names	✓	✓	✓	✓	✓
Authorships			✓	✓	✓
Common Names	✓	✓	✓	✓	✓
Predators	✓				✓
Ecosystems	✓				✓
Countries					✓
Water Areas		✓			✓
Vessels	✓	✓			✓
Gears	✓	✓			✓
EEZ		✓			

Fig. 8: Concept coverage by the sources in the MarineTLO-based warehouse

[15] Complete documentation, competency queries, SILK rules and examples are available in `http://www.ics.forth.gr/isl/MatWare/#products`

[16] URL of the warehouse (restricted access): `http://virtuoso.i-marine.d4science.org:8890/sparql`

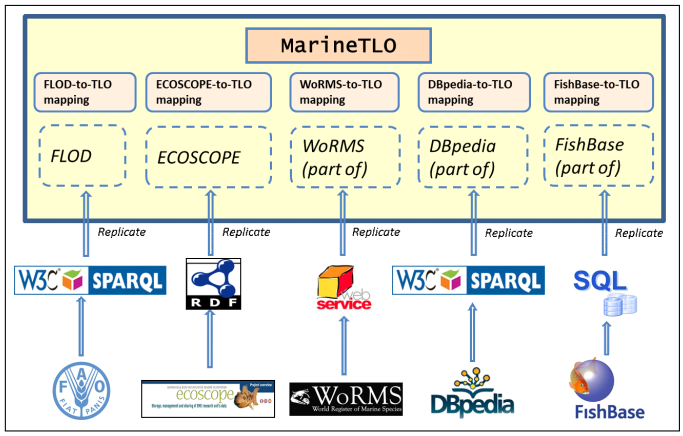
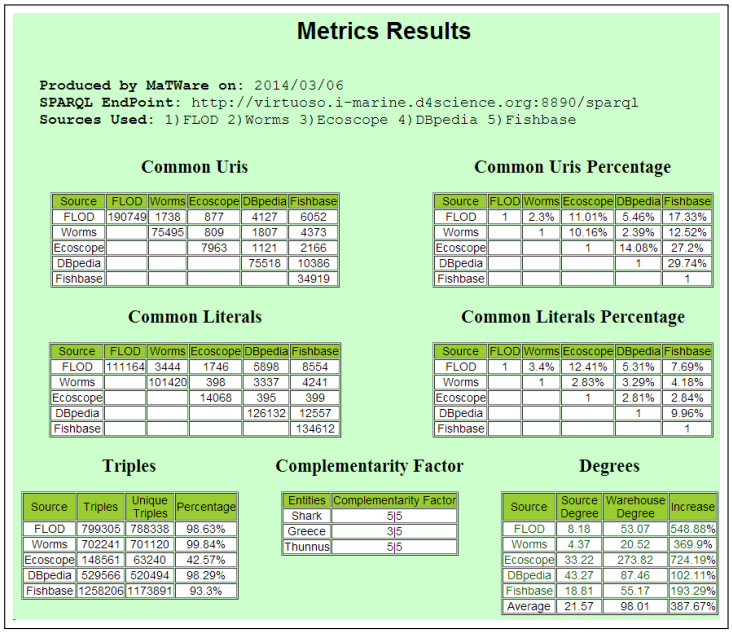


Fig. 9: Overview of the MarineTLO-based warehouse



Metrics Results

Produced by MaTWare on: 2014/03/06
SPARQL EndPoint: http://virtuoso.i-marine.d4science.org:8890/sparql
Sources Used: 1)FLOD 2)Worms 3)Ecoscope 4)DBpedia 5)Fishbase

Common Uris

Source	FLOD	Worms	Ecoscope	DBpedia	Fishbase
FLOD	190749	1738	877	4127	6052
Worms		75495	809	1807	4373
Ecoscope			7963	1121	2166
DBpedia				75518	10386
Fishbase					34919

Common Uris Percentage

Source	FLOD	Worms	Ecoscope	DBpedia	Fishbase
FLOD	1	2.3%	11.01%	5.46%	17.33%
Worms		1	10.16%	2.39%	12.52%
Ecoscope			1	14.08%	27.2%
DBpedia				1	29.74%
Fishbase					1

Common Literals

Source	FLOD	Worms	Ecoscope	DBpedia	Fishbase
FLOD	111164	3444	1746	5898	8554
Worms		101420	398	3337	4241
Ecoscope			14068	395	399
DBpedia				126132	12557
Fishbase					134612

Common Literals Percentage

Source	FLOD	Worms	Ecoscope	DBpedia	Fishbase
FLOD	1	3.4%	12.41%	5.31%	7.69%
Worms		1	2.83%	3.29%	4.18%
Ecoscope			1	2.81%	2.84%
DBpedia				1	9.96%
Fishbase					1

Triples

Source	Triples	Unique Triples	Percentage
FLOD	799305	788338	98.63%
Worms	702241	701120	99.84%
Ecoscope	148561	63240	42.57%
DBpedia	529566	520494	98.29%
Fishbase	1258206	1173891	93.3%

Complementarity Factor

Entities	Complementarity Factor
Shark	5|5
Greece	3|5
Thunnus	5|5

Degrees

Source	Source Degree	Warehouse Degree	Increase
FLOD	8.18	53.07	548.88%
Worms	4.37	20.52	369.9%
Ecoscope	33.22	273.82	724.19%
DBpedia	43.27	87.46	102.11%
Fishbase	18.81	55.17	193.29%
Average	21.57	98.01	387.67%

Fig. 10: Metric's results displayed in HTML.

Figure 9 shows an overview of the warehouse's contents, as fetched and transformed from the various sources. As regards its evolution, a new release of the warehouse is published every two months. The current warehouse contains information for about 37,000 marine species. In total, it contains 3,772,919 triples. The current warehouse takes about 7 hours[17] to reconstruct from scratch. This

[17] Virtuoso and machine specs: OpenLink Virtuoso V6.1, Windows 8.1, 64-bit, Intel i3 dual-core, 4 GB RAM

process includes: downloading the sources (60 min), importing the data in the repository (230 min), applying the transformation rules (40 min), producing `sameAs` links using SILK (30 min), and computing the mertics (100 min). As regards query evaluation, the time required to answer a competency query ranges from 31 ms to 3.4 seconds. The query rewriting for provenance (which is done automatically by *MatWare*) does not increase the query evaluation time. After each reconstruction, *MatWare* computes the various connectivity metrics and exports them in the form of an HTML page, as shown in Figure 10. This not only enables monitoring the quality of the warehouse, but it is also a kind of quantitative documentation of the warehouse. For instance, and for the warehouse at hand, by considering the values of the complementarity factors and the increment of the average degrees (recall Section 3.5) we can understand that the resulting warehouse not only contains concrete information for each entity *from all* sources, but we can also see *how much* the average degree of these entities has been increased in the warehouse.

5.2 Applications over the Warehouse

Here we describe three applications that exploit the current warehouse

A. Fusion of Structured and Unstructured Data at Search Time. One big challenge nowadays is how to integrate structured data with unstructured data (documents and text). The availability of harmonized structured knowledge about the marine domain is currently exploited for a *semantic post-processing* of the search results. Specifically the work done in the context of iMarine so far, described in [2, 3], proposed a method to enrich the classical (mainly keyword based) searching with *entity mining* that is performed at *query time*. The left part of Figure 11 illustrates the process, while the right part depicts a screen shot from a prototype search system.

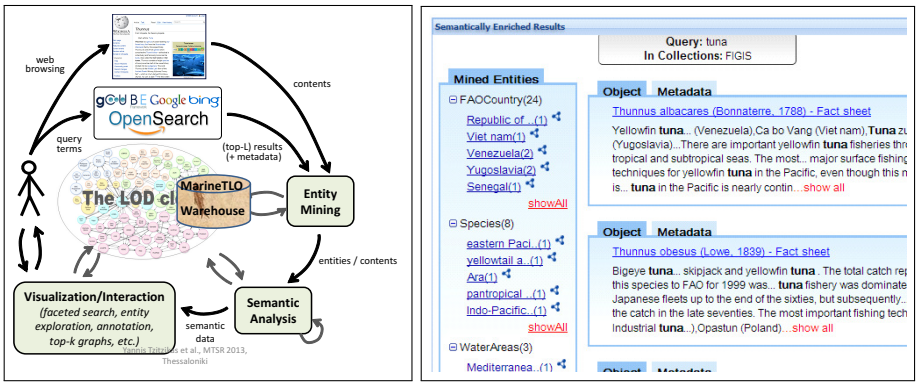


Fig. 11: Semantic post-processing of search results

In particular, the results of entity mining (entities grouped in categories) complement the query answers with information which can be further exploited by

the user in a faceted and session-based interaction scheme [13]. This means that instead of annotating and building indexes for the documents (or web pages), the annotation can be done at *query time* and using the desired entities of interest. These works show that the application of entity mining over the *snippets* of the top hits of the answers can be performed at real-time, and indicated how semantic repositories can be exploited for specifying the entities of interest and for providing further information about the identified entities. For applying these methods over the full-contents it is worth exploiting multiple machines. A MapReduce-based decomposition is described in [6].

B. Fact Sheet Generator. FactSheetGenerator[18] is an application provided by IRD aiming at providing factual knowledge about the marine domain by mashing-up relevant knowledge distributed across several data sources. Figure 12(left) shows the results of the current FactSheetGenerator when searching for the species `Thunnus albacares`.

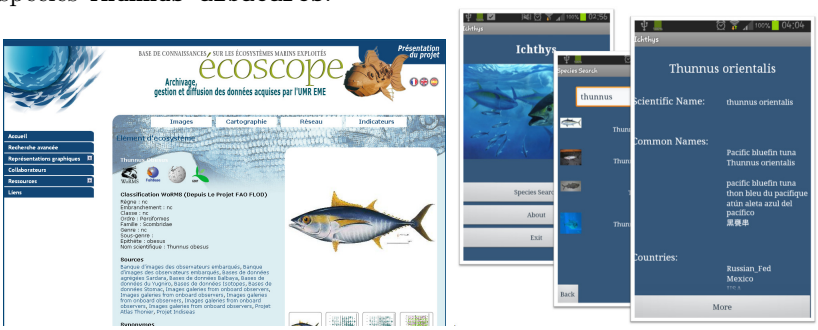


Fig. 12: Left: `Thunnus albacares` in FactSheetGenerator, Right: Screens from the `Ichthys` Android application

C. Android application. We have developed (and currently improve) an Android application, called `Ichthys` that exploits the contents of the warehouse for providing to end users information about marine species in a user friendly manner. A few screens are shown in Figure 12(right).

6 Concluding Remarks

We have described the main requirements and challenges, stemming from our experience in designing, building, maintaining and evolving an operational semantic warehouse for marine resources. We have presented the process and the tools that we have developed for supporting this process with emphasis on *scope control*, *connectivity assessment*, *provenance*, and *freshness*. To tackle these requirements and automate the warehouse construction process we have developed, *`MatWare`*, an extensible tool for supporting and automating the above process. In future we plan to elaborate on improving the scalability as the volume of data grows (e.g. using a single graph space that materializes all triples for offering efficient query answering, while keeping also the separate graph spaces for

[18] `http://www.ecoscopebc.ird.fr/`

provenance reasons). Furthermore, we plan to further work on the evaluation of the quality of the warehouse's contents.

Acknowledgement This work was partially supported by the ongoing project *iMarine* (FP7 Research Infrastructures, 2011-2014).

References

1. F. Darari, W. Nutt, G. Pirro, and S. Razniewski. Completeness Statements about RDF Data Sources and their Use for Query Answering. In *The Semantic Web–ISWC 2013*, pages 66–83. Springer, 2013.
2. P. Fafalios, I. Kitsos, Y. Marketakis, C. Baldassarre, M. Salampasis, and Y. Tzitzikas. Web Searching with Entity Mining at Query Time. In *Proceedings of the 5th Information Retrieval Facility Conference*, 2012.
3. P. Fafalios and Y. Tzitzikas. X-ENS: Semantic Enrichment of Web Search Results at Real-Time. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 28 - August 01 2013.
4. A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and Browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4), 2011.
5. Y. Hu, K. Janowicz, G. McKenzie, K. Sengupta, and P. Hitzler. A Linked-Data-driven and Semantically-enabled Journal Portal for Scientometrics. In *The Semantic Web–ISWC 2013*, pages 114–129. Springer, 2013.
6. I. Kitsos, K. Magoutis, and Y. Tzitzikas. Scalable Entity-based Summarization of Web Search Results Using MapReduce. *Distributed and Parallel Databases*, 2013 (accepted, online first).
7. T. Knap and J. Michelfeit. Linked Data Aggregation Algorithm: Increasing Completeness and Consistency of Data.
8. T. Knap, J. Michelfeit, J. Daniel, P. Jerman, D. Rychnovskỳ, T. Soukup, and M. Nečaskỳ. ODCleanStore: a framework for managing and providing integrated linked data on the web. In *Web Information Systems Engineering-WISE 2012*, pages 815–816. Springer, 2012.
9. K. Makris, G. Skevakis, V. Kalokyri, P. Arapi, S. Christodoulakis, J. Stoitsis, N. Manolis, and S. L. Rojas. Federating Natural History Museums in Natural Europe. In *Metadata and Semantics Research*, pages 361–372. Springer, 2013.
10. P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123. ACM, 2012.
11. J. Michelfeit and T. Knap. Linked Data Fusion in ODCleanStore. In *International Semantic Web Conference (Posters & Demos)*, 2012.
12. E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a Document-Oriented Lookup Index for Open Linked Data. *Int. J. Metadata Semant. Ontologies*, 3(1):37–52, 2008.
13. G. Sacco and Y. Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, volume 25. Springer-Verlag New York Inc, 2009.
14. Y. Tzitzikas, C. Alloca, C. Bekiari, Y. Marketakis, P. Fafalios, M. Doerr, N. Minadakis, T. Patkos, and L. Candela. Integrating Heterogeneous and Distributed Information about Marine Species through a Top Level Ontology. In *Proceedings of the 7th Metadata and Semantic Research Conference (MTSR'13)*, Thessaloniki, Greece, November 2013.

15. Y. Tzitzikas, N. Minadakis, Y. Marketakis, P. Fafalios, C. Alloca, and M. Mountantonakis. Quantifying the Connectivity of a Semantic Warehouse. In *Proceedings of the 4th International Workshop on Linked Web Data Management (LWDM 2014) in conjunction with the 17th International Conference on Extending Database Technology (EDBT 2014)*, 2014.