Extending VoID for Expressing the Connectivity Metrics of a Semantic Warehouse

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

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

Abstract. VoID (Vocabulary of Interlinked Datasets) has been proposed by W3C as the vocabulary for expressing metadata about RDF *datasets*. Despite its important contributions, VoID cannot express metadata that concern the *connectivity* of semantic warehouses. We use the term *semantic warehouse* to refer to a read-only set of RDF triples fetched (and transformed) from different sources that aims at serving a particular set of query requirements. *Connectivity metrics* are important for evaluating the value of a semantic warehouse, since they reflect its query capabilities. Moreover they also quantify the contribution of each constituent source to the warehouse. To allow the representation, exchange, and querying of such measurements, in this paper we describe an extension of VoID that allows representing them. We demonstrate its applicability through the case of a real and operational semantic warehouse for the marine domain.

1 Introduction

An increasing number of datasets is already available as Linked Data. For exploiting this wealth of data, and building domain specific applications, in many cases there is a need for fetching and assembling pieces of information coming from more than one sources (including SPARQL endpoints). Then, these pieces can be used for constructing a *warehouse*, for offering more complete browsing and query services (in comparison to those offered by the underlying sources).

We shall use the term *Semantic Warehouse* (for short warehouse) to refer to a read-only set of RDF triples fetched (and transformed) from different sources that aims at serving a particular set of query requirements. There are various such warehouses (e.g. [13, 9, 6]), and there are various tools that can aid the construction of such warehouses, including *ODCleanStore* [8], *Sieve* [10] and *MatWare* [16]. However, putting triples together does not guarantee that they will be connected. In general, the aspect of "connectivity" concerns both schema and instances. One method to check and quantify the connectivity of a semantic warehouse is through the *connectivity metrics* proposed in [15]. These metrics provide an overview of the warehouse that reflects its query capabilities, and also quantify the contribution of each constituent source to the warehouse. In brief, the main metrics are: (a) the *matrix of percentages of the common URIs and/or literals*, (b) the *complementarity factor of the entities of interest*, (c) the *increments in the average degree of each source*, and (d) the *unique triple contribution of each source*. The values of (a),(b),(c) allow evaluating the warehouse, while (c) and (d) mainly concern each particular source.

W3C proposed VoID (Vocabulary of Interlinked Datasets), a vocabulary for describing open and linked datasets [7]. It aims at building a bridge between the publishers and the users of a dataset and applications, ranging from data discovery to cataloging and archiving of datasets. Based on Dublin Core [11] for describing generic info, it is an RDF Schema vocabulary for expressing different types of metadata such as *general metadata* (e.g. *dc:title*), *access metadata* (e.g. *void:sparqlPoint*), *structural metadata* (e.g. *void:exampleResource*) and *description of links between RDF datasets* (e.g. *void:Linkset*). The specification also provides deployment advice and discusses how *well-known URIs* can be used to locate a VoID file (which is a machine-readable description of an RDF dataset) for its discovery [7]. However, VoID cannot model the aforementioned connectivity metrics. For this reason, in this paper we describe an extension of VoID that allows the representation of such measurements, and thus enables their exchange and querying. We demonstrate its applicability through the case of a real and operational semantic warehouse for the marine domain.

In a nutshell, the key contributions of our work are: (a) we motivate (through a concrete scenario) why VoID should be extended, (b) we propose an extension of VoID that models all metrics proposed in [15], (c) we describe its applicability through the use of a real and operational Semantic Warehouse of the marine domain.

The rest of this paper is organised as follows: Section 2 describes the required background, i.e. the VoID vocabulary and the connectivity metrics. Section 3 describes the proposed extension of VoID and describes its applicability through the case of a real and operational Semantic Warehouse for the marine domain. Finally, Section 4 concludes the paper.

2 Background

Here we describe in brief VoiD (in §2.1), we synopsize the warehouse connectivity metrics (in §2.2), and discuss related works (in §2.3).

2.1 VoID

This section describes briefly the current version of VoID (Vocabulary of Interlinked Datasets) [7]. Informally, its design has been driven by representing a number of both domain-dependent features (e.g. which type of data it contains) and domain-independent ones (e.g. who published it).

Conceptually, it has been built around the notions of *void:Dataset*, *void:Linkset* and *RDF Links*. A *void:Dataset* is a set of RDF triples that are published, maintained or aggregated by a single provider. A *void:Linkset* is a collection of RDF

Links between two datasets. An RDF Link is an RDF triple whose subject and object are described in different void:Dataset.

Based on Dublin Core [11], VoID provides properties that can be attached to both *void:Dataset* and *void:Linkset* to express metadata of the type of:

General metadata helping users to decide whether the dataset is appropriate for their purpose. Based on Dublin Core model, they refer to information such as *dcterms:title*, *dcterms:description*, *dcterms:license*, *dcterms:subject*, *dcterms:creator*, *dcterms:publisher*, *dcterms:contributor*, *dcterms:created*, *dcterms:issued*, *dcterms:modified*, *void:feature*.

Access metadata used to describe methods of accessing the RDF data using various protocols. They are: *void:sparqlEndpoint*, *void:dataDump*, *void:rootResourse*, *void:uriLookupEndpoint*, *void:openSearchDescription*.

Structural metadata providing high-level information about the schema and internal structure of a dataset and can be helpful when exploring and querying the dataset. They are: *void:exampleResource*, *void:uriSpace*, *void:uriRegexPattern*, *void:vocabulary*, *void:subset*, *void:classPartition*, *void:propertyPartition*, *void:triples*, *void:entities*, *void:classes*, *void:properties*, *void:distinctSubjects*, *void:distinctObjects*, *void:documents*.

Description of links between datasets helpful for understanding how multiple datasets are related and can be used together. They are: *void:Linkset, void:target, void:linkPredicate* and all patterns for describing datasets can equally be used for *void:Linkset*.

2.2 Connectivity Metrics

One method to check and quantify the connectivity of a semantic warehouse is through the *connectivity metrics* proposed in [15]. These metrics provide an overview of the warehouse that reflect its query capabilities, and also quantify the contribution of each constituent source to the warehouse.

In brief, 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 concern each particular source.

For reasons of self-containedness, here we summarize the definition of the metrics. Table 1 introduces the symbols that are required for defining the metrics. However, for computing the metrics several policies can be followed for deciding whether two URIs or two literals should be considered equivalent. The proposed policies for equivalence are shown in Table 2. Finally, Table 3 shows how each metric is defined.

Symbol	Meaning
$S = S_1, \dots S_k$	the set of underlying sources.
$triples(S_i)$	the set of triples that each source contributes to the warehouse.
U_i	the URIs that appear in $triples(S_i)$
Lit_i	the literals that appear in $triples(S_i)$
W	the triples in the warehouse
E	the entities of interest, in the form of a set of literals and/or URIs
T	a set of triples
$deg_T(e)$	$= \lvert\{(s,p,o) \in T \mid s = e \text{ or } \ o = e\}\rvert$, i.e. the degree of an entity e in T
$deg_T(E)$	$= avg_{e \in E}(deg_T(e))$, i.e. the average degree of the entities E in T
u_i	a URI
last(u_i)	the string obtained by getting the substring after the last "/" or "#" of u_i, turning the letters of the picked substring to lowercase and deleting the underscore letters that might exist.
sameAs	the `sameAs` relationship between two URIs according to the entity matching rules that are (or will be eventually) used for the warehouse.

Table 1. Definitions

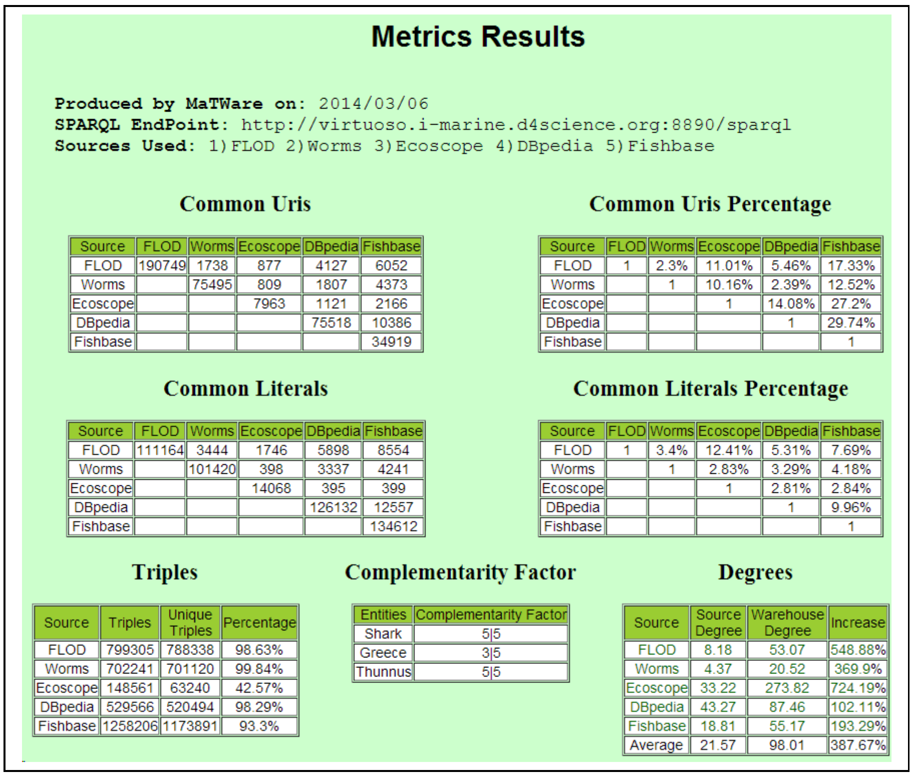
	Policy Name	Policy Description
(i)	Exact String Equality	$u_1 = u_2 \Rightarrow u_1 \equiv u_2$
(ii)	Suffix Canonicalization	$last(u_1) = last(u_2) \Rightarrow u_1 \equiv u_2$
(iii)	Entity Matching	$u_1 \ \texttt{sameAs} \ u_2 \Rightarrow u_1 \equiv u_2$

Table 2. Policies used to compare URIs coming from different sources

Metric Name	Metric Definition
Common URIs between two sources S_i and S_j	$\lvert U_i \cap U_j \rvert$
Percentage of Common URIs between S_i and S_j	$curi_{i,j} = \frac{\lvert U_i \cap U_j \rvert}{\min(\lvert U_i\rvert, \lvert U_j\rvert)}$
Common Literals between S_i and S_j	$\lvert Lit_i \cap Lit_j \rvert$
Percentage of Common Literals between S_i and S_j	$clit_{i,j} = \frac{\lvert Lit_i \cap Lit_j \rvert}{\min(\lvert Lit_i\rvert, \lvert Lit_j\rvert)}$
Increase in the average degree	$\frac{deg_W(E) - deg_S(E)}{deg_S(E)}$
Unique Triples of S_i, $triplesUnique(S_i) =$	$triples(S_i) \setminus (\cup_{1 \le j \le k, j \neq i} triples(S_j))$
Percentage of Unique Triples of a Source S_i=	$\frac{\lvert triplesUnique(S_i)\rvert}{\lvert triples(S_i)\rvert}$
Complementarity factor of an entity e, $cf(e)$=	$\lvert\{\ i \mid triples_W(e) \cap triplesUnique(S_i) \neq \emptyset\}\rvert$

Table 3. Connectivity Metrics

The metrics are currently used by the tool *MatWare* [16], and Figure 1 shows the HTML that it is produced by this tool over a warehouse that integrates information from WoRMS[1], Ecoscope[2], FishBase[3], FLOD[4] and DBpedia[5].



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. 1. Values of Metrics as computed by `MatWare`

Another metric, that was not proposed in [15], but it is useful to have, is the number of blank nodes that appear in the triples of a source S_i, i.e. in $triples(S_i)$. If these triples contain bnodes, then a blank node matching algorithm, like those proposed in [14], should be used, for improving the connectivity between the sources.

2.3 Related Work

In this section we review the main related work on modeling characteristics of semantic data sources.

[1] `http://www.marinespecies.org/`
[2] `http://www.ecoscopebc.ird.fr/EcoscopeKB/ShowWelcomePage.action`
[3] `http://www.fishbase.org/`
[4] `http://www.fao.org/figis/flod/`
[5] `http://dbpedia.org/`

Completeness. In [4] the authors introduce a formal framework for the declarative specification of completeness statements about RDF data sources and underline how the framework can complement existing initiatives like VoID. They also show how to assess completeness of query answering over plain and RDF/S data sources augmented with completeness statements, and they present an extension of the completeness framework for federated data sources.

Provenance. W3C has proposed the *PROV Family of Documents*[6] which defines a model, corresponding serializations and other supporting definitions to enable the interoperable interchange of provenance information. In addition, the work in [5] presents approaches to integrate provenance information into the Web of Data and illustrates how this information can be consumed. In particular, the authors introduce a *Provenance Vocabulary* which, by using it together with VoID, assists providers of Linked Data to describe the provenance of their data using RDF. The authors also discuss possibilities to make such provenance metadata accessible as part of the Web of Data and they describe how this metadata can be queried and consumed to identify outdated information.

Connectivity among Concepts / Fuzzy LinkSets. In [2] the authors propose extensions of VoID for (i) distinguishing datasets, and (ii) describing fuzzy linksets, i.e. links between different datasets that are not explicitly stated. As regards (i), the authors define the notion of "semantic datasets", i.e. partitions of resources that share certain semantic features. Specifically, they propose the use of two new classes (*voidgen:ConnectedDataset* and *voidgen:ConceptualDataset*) for identifying connected sets of resources or sets of conceptually similar resources. Thereby, given two such semantic datasets and respective linksets, one can, for instance, observe the connectivity among concepts. As regards (ii), they introduce the notion of k-similarity where two subjects are k-similar, if k of their predicate/object combinations are exact matches. For specifying a fuzzy linkset, the authors propose a new class *voidgen:FuzzyLinkset* and a new attribute *voidgen:kSimilarity*.

Statistics. The RDF Data Cube Vocabulary[7] [3] provides a means to publish multi-dimensional data, such as statistics, on the web in such a way that it can be linked to related datasets and concepts. The model underpinning the Data Cube vocabulary is compatible with the cube model that underlies SDMX (Statistical Data and Metadata eXchange)[8], an ISO standard for exchanging and sharing statistical data and metadata among organizations. As regards our case, we should stress that what we call *semantic warehouse* is not necessary a multi-dimensional dataset. Therefore, the Data Cube vocabulary cannot replace the need for VoID and the extension that we propose. Of course, a semantic warehouse could contain one or more multi-dimensional datasets (as for example [12]) and such datasets could be described using the Data Cube vocabulary. For instance, and for the `MarineTLO`-based warehouse [16], its part that contains *occurrences of species*, could be expressed using the Data Cube vocabulary.

[6] `http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/`
[7] `http://www.w3.org/TR/2013/PR-vocab-data-cube-20131217/`
[8] `http://sdmx.org/`

However, for the connectivity metrics per se, the adoption of a multidimensional modeling would not offer any benefit.

In [1] the authors describe LODStats, a statement-stream-based approach for gathering comprehensive statistics about RDF datasets. To represent the statistics, they use VoID and the Data Cube Vocabulary. In addition, they link a *void:Dataset* (a VoID class) to a *qb:Observation* (a Data Cube class) using a newly defined object property (*void-ext:observation*), which is a simple extension to VoID.

Difference of our approach. The main difference of our approach is that we focus on modeling *metrics* regarding the *connectivity* and the *quality* of a Semantic Warehouse, thus it can *complement* existing initiatives like VoID and it can be used together with approaches that focus on provenance, completeness, statistics, etc. These metrics reflect the query capabilities of a warehouse (so they are important for evaluating its value) and also quantify the contribution of the underlying sources, allowing evaluating the importance of each source for the warehouse at hand.

3 The Proposed Extension of VoID

At first we discuss the requirements (in §3.1), then we describe the proposed conceptual model (in §3.2), then we provide an example of using that model (in §3.3). Subsequently, we show how we can compute these metrics (in §3.4) and how we can store and use their values (in §3.5). Figure 2 gives a total view of the functionality of the metrics and the proposed extension.

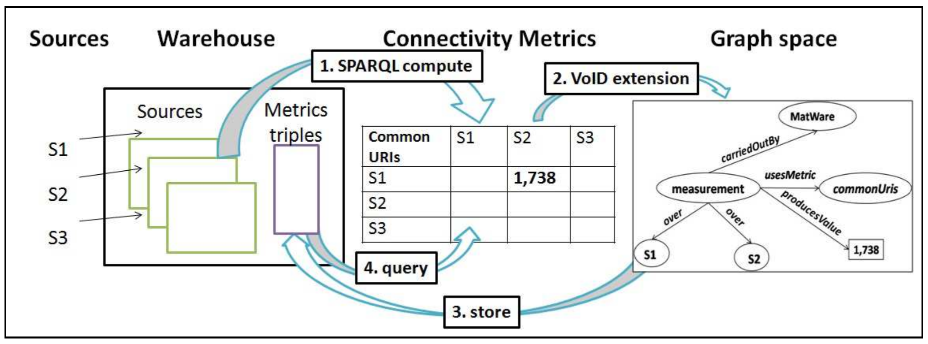


Fig. 2. The whole metrics process

3.1 Requirements

In brief, we could say that the extension should allow all information expressed in the tables of Figure 1 to be expressed in a machine processable (and query-able) manner. If such information is exposed in a machine-readable format, it could be exploited in various methods, e.g.:

– For producing *visualizations* that give an overview of the contents of a warehouse.
– For *comparing different warehouses* and producing comparative reports.
– For aiding the *automatic discovery of related data* since software services/agents based on these metrics could decide which SPARQL endpoints to query based on time/cost constraints.
– For *crediting good sources* since these metrics make evident, and quantifiable, the contribution of a source to the warehouse.

Another requirement is that the proposed extension should be compatible with the existing VoID vocabulary and the available VoID-based descriptions.

3.2 Conceptual Model

Figure 3 shows the core conceptual model as an implementation in RDF/S. As one can easily see, the implementation reuses classes and properties from *VoID, Dublin Core, RDF/S* and *XML Schema Definition*, while the new modeling elements are defined in a separate namespace, generally named with the prefix *vdw* and here depicted as the default namespace.

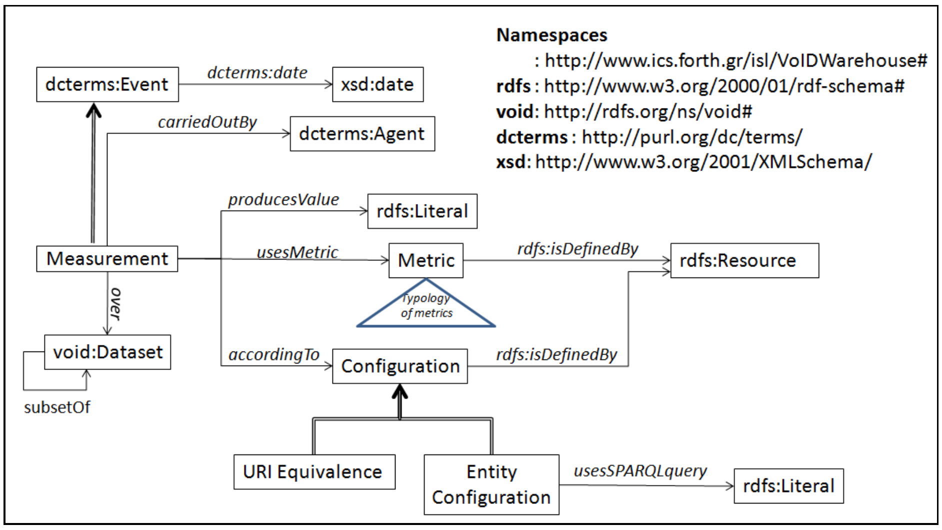


Fig. 3. Core Conceptual Model

We can see that there is the notion of *Measurement* which is actually a specialization of *Event* and therefore inherits the property *date*. A *measurement* is carried out by an *agent* using a specific *metric* according to one (or more) *configurations over* one (or more) *datasets* (atomic or composite) and produces a value (i.e. literal).

Each *metric* is an individual with a URI and is *defined by* a resource (e.g. the DOI of the scientific paper that defined that *metric*). The notion of *Configuration* concerns issues that explain how the *measurement* was done. At this point, and for the requirements at hand, we need two subclasses: the first concerns the way *URI equivalence* is defined (e.g. through the policies given in Table 2), while the second concerns how the *entities of interest* are defined. Regarding the latter the current modeling allows someone to specify the desired set of entities by providing a SPARQL query that returns them.

The extension is currently published in `http://www.ics.forth.gr/isl/VoIDWarehouse`, and apart from the vocabulary it contains URIs for the connectivity metrics.

3.3 Using VoID and the Proposed Extension to Describe the `MarineTLO`-based Warehouse

Here we show how with VoID and the proposed extension we can describe the `MarineTLO`-based warehouse [16] and the corresponding connectivity metrics (this corresponds to the process "2. VoID extension" of Figure 2). At first we discuss what we can describe using only VoID. The description is presented in a modular way, i.e. Figure 4 shows the used namespaces, and the general and administrative metadata of the warehouse, while Figure 5 shows the description of the constituent datasets and of their schema mappings with the warehouse.

```
@prefix rdf:          <http://www.w3.org/2999/02/22-rdf-syntax-ns#> .
@prefix rdfs:         <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:          <http://www.w3.org/2002/07/owl#> .
@prefix void:         <http://rdfs.org/ns/void#> .
@prefix dcterms:      <http://purl.org/dc/terms/> .
@prefix dst:          <http://www.ics.forth.gr/isl#> .
@prefix vdw:          <http://www.ics.forth.gr/isl/VoIDWarehouse#> .
# General Description of the MarineTLO-based Warehouse
dst:MarineTLOWarehouse  rdf:type  void:Dataset;
     dcterms:title        "Marine TLO Warehouse version 4";
     dcterms:publisher    <http://ics.forth.gr/isl> ;
     dcterms:description  "Warehouse for i-Marine Project version 4"> ;
     dcterms:licence      "Open Database License (ODC-ODbL)";
     void:vocabulary      <http://ics.forth.gr/Ontology/MarineTLO/core>;
     dcterms:subject      <http://dbpedia.org/resource/Marine_ecosystem>;
     dcterms:subject      <http://dbpedia.org/resource/Species>;
     dcterms:subject      <http://dbpedia.org/resource/Predator>;
     dcterms:issued       "2014-02-02T02:10:30"^^xsd:dateTime;
     void:sparqlEndpoint  <http://virtuoso.i-marine.d4science.org:8890/sparql>;
     void:feature         <http://www.w3.org/ns/formats/N-Triples>;
     void:triples         "3,500,000"
```

Fig. 4. `MarineTLO`-based warehouse VoID triples (part 1)

Now, Figure 6 illustrates how the value of the connectivity metric called *common URIs*, as computed over FLOD and Ecoscope, is represented using the proposed extension. We can see that ***MatWare*** carried out this measurement and computed 1,738 common URIs between these two sources, according to the *suffixCanonicalization*-based URI equivalence. Also notice that exact definitions of suffix Canonicalization and *commonURIs*, are given in the paper [15], whose URL is connected with the measurement at hand. The description of the measurement in triples is shown in Figure 7.

```
# Description of some of the components of the MarineTLO-based Warehouse
dst:MarineTLOWarehouse  rdf:type  void:Dataset;
   void:subset    dst:MarineTLO;
   void:subset    dst:FLODPart;
   void:subset    dst:EcoscopePart;
   void:subset    dst:Mappings
# MarineTLO
dst:MarineTLO  rdf:type  void:Dataset;
   dcterms:title       "MarineTLO ontology";
   dcterms:publisher   <www.ics.forth.gr/isl> ;
   dcterms:description "MarineTLO is a top-level ontology for the marine domain";
   dcterms:provenance  <http://www.ics.forth.gr/isl/MarineTLO/> ;
   void:triples        "4,000"
# FLODPart
dst:FLODPart  rdf:type  void:Dataset;
   dcterms:title        "Part of FLOD source";
   dcterms:publisher    <www.ics.forth.gr/isl>  ;
   dcterms:description  "Our Part from FAO data for marine domains";
   dcterms:provenance   <http://www.fao.org>;
   void:triples         "750,000"
# EcoscopePart
dst:EcoscopePart  rdf:type  void:Dataset;
   dcterms:title        "Part of Ecoscope source";
   dcterms:publisher    <www.ics.forth.gr/isl> ;
   dcterms:description  "Part of Ecoscope Data for Marine Domains";
   dcterms:provenance   <http://www.ird.fr>
   void:triples         "150,000"
# Mappings
dst:Mappings  rdf:type  void:Dataset;
  void:subset    dst:FLODPart2MarineTLO;
  void:subset    dst:EcoscopePart2MarineTLO;
  void:subset    dst:EcoscopePart2FLODPart
# Instance Matching EcoscopePart - FLODPart
dst:EcoscopePart2FLODPart  rdf:type  void:Linkset;
   void:target          dst:EcoscopePart;
   void:target          dst:FLODPart;
   void:linkPredicate  owl:sameAs
# Schema Mappings EcoscopePart - MarineTLO (SubClassOf)
dst:EcoscopePart2MarineTLO  rdf:type  void:Linkset;
   void:target          dst:EcoscopePart;
   void:target          dst:MarineTLO;
   void:linkPredicate  rdfs:subClassOf
# Schema Mappings FLODPart - MarineTLO (SubClassOf)
dst:FLODPart2MarineTLO  rdf:type  void:Linkset;
   void:target          dst:FLODPart;
   void:target          dst:MarineTLO;
   void:linkPredicate  rdfs:subClassOf
```

Fig. 5. `MarineTLO`-based warehouse VoID triples (part 2)

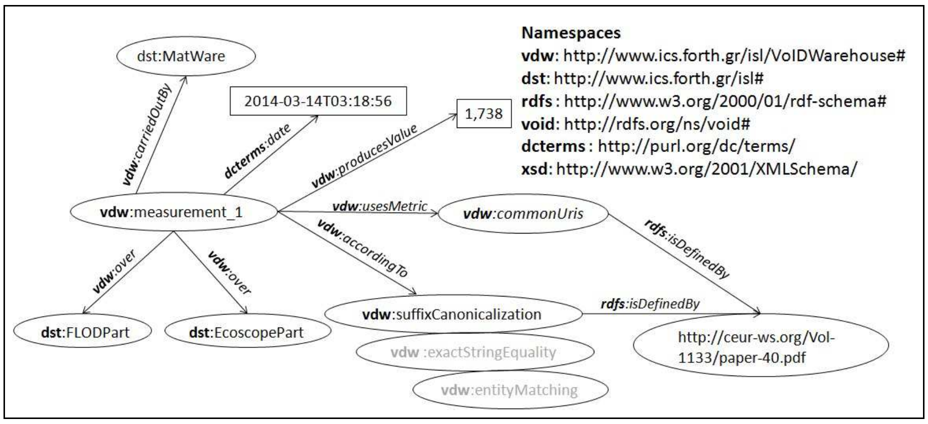

Namespaces
vdw: http://www.ics.forth.gr/isl/VoIDWarehouse#
dst: http://www.ics.forth.gr/isl#
rdfs : http://www.w3.org/2000/01/rdf-schema#
void: http://rdfs.org/ns/void#
dcterms : http://purl.org/dc/terms/
xsd: http://www.w3.org/2001/XMLSchema/
dst:MatWare
vdw:carriedOutBy
2014-03-14T03:18:56
dcterms:date
1,738
vdw:producesValue
vdw:measurement_1
vdw:usesMetric
vdw:commonUris
rdfs:isDefinedBy
vdw:accordingTo
vdw:over
vdw:over
dst:FLODPart
dst:EcoscopePart
vdw:suffixCanonicalization
rdfs:isDefinedBy
http://ceur-ws.org/Vol-1133/paper-40.pdf
vdw :exactStringEquality
vdw:entityMatching

Fig. 6. Example of using the extension for expressing a measurement

```
dst:EcoscopePart  rdf:type  void:Dataset .
dst:FLODPart      rdf:type  void:Dataset .

vdw:measurement_1  rdf:type  vdw:Measurement;
           vdw:carriedOutBy  dst:MatWare;
           dcterms:date  "2014-03-14T03:18:56"^^xsd:dateTime;
           vdw:over  dst:EcoscopePart;
           vdw:over  dst:FLODPart;
           vdw:accordingTo  vdw:suffixCanonicalization;
           vdw:producesValue  "1,738";
           vdw:usesMetric  vdw:commonUris .

vdw:commonUris  rdfs:isDefinedBy  <http://ceur-ws.org/Vol-1133/paper-40.pdf> .
vdw:suffixCanonicalization  rdfs:isDefinedBy  <http://ceur-ws.org/Vol-1133/paper-40.pdf> .
dst:MatWare  rdf:type  dcterms:Agent .
```

Fig. 7. Example (in triples) of using the extension for expressing a measurement

3.4 Computing the Connectivity Metrics using SPARQL queries

For making clear the entire life cycle, here we show how the values of the connectivity metrics can be computed using SPARQL queries (this corresponds to the process "1. SPARQL compute" of Figure 2).

Common URIs. The metric *Common URIs* over two sources S_i and S_j, can be computed with the following query:

```
SELECT COUNT (DISTINCT ?o)
WHERE { graph :Si {{?s1 ?p1a ?o} UNION {?o ?p1b ?o1}} . FILTER(isURI(?o))
        graph :Sj {{?s2 ?p2a ?o} UNION {?o ?p2b ?o2}}  }
```

In the context of the warehouse, this metric should be computed over all pairs of sources, i.e. all (S_i, S_j) such that $S_i, S_j \in S$ and $i \neq j$. Note that this metric is symmetric, i.e. the value of the pair (S_i, S_j) is equal to the value of (S_j, S_i).

Common Literals. The *Common Literals* between two sources S_i and S_j can be computed in a similar manner, i.e.:

```
SELECT COUNT DISTINCT ?o
WHERE { graph :Si { ?s ?p ?o} . FILTER(isLiteral(?o))
        graph :Sj { ?a ?b ?o} }
```

Again, this metric should also be computed over all pairs (S_i, S_j) of the warehouse.

Unique Triples Contribution. To compute the unique triple contribution of a source, say S_1, to the warehouse $S = S_1, \ldots, S_k$, we have to count the number of triples of S_1 that do not intersect with the triples of any of the other sources of S (i.e. with none of the sources in S_2 ... S_n). This can be done using the following query:

```
SELECT COUNT(*)
WHERE { graph :S1 { ?s ?p1 ?o} .
    FILTER NOT EXISTS { graph :S2 { ?s ?p2 ?o} } .
                ...
                ...
    FILTER NOT EXISTS { graph :Sn { ?s ?pn ?o} } }
```

Complementarity Factor. This metric is computed for a specific entity over all sources of the warehouse. In particular, the complementarity factor of an entity e is increased by 1 for each source $S_i \in S$ that contains at least one unique triple having the entity e. This means that if all sources in S contain unique triples for e, then its complementarity factor will be n. The query below gives the complementarity factor of an entity e over S. Notice that the WHERE clause contains n graph patterns. Each graph pattern i returns 1 if S_i contains unique triples for the entity e, or 0 otherwise.

```
SELECT  (?CF1+ .. + ?CFn) AS ?CF
WHERE { { SELECT xsd:integer(COUNT(*)>0) as ?CF1
          WHERE { { graph :S1 { ?s ?p1 ?o } }
               FILTER NOT EXISTS { graph :S2 { ?s ?p2 ?o} } .
                 ...
                 ...
               FILTER NOT EXISTS { graph :Sn { ?s ?pn ?o} }
               FILTER (regex(?s, e,'i') || (regex(?o, e,'i'))) } }
                 ...
                 ...
        { SELECT xsd:integer(COUNT(*)>0) as ?CFn
          WHERE { { graph :Sn { ?s ?pn ?o } }
               FILTER NOT EXISTS { graph :S1 { ?s ?p1 ?o } } .
                 ...
                 ...
               FILTER NOT EXISTS { graph :Sn-1 { ?s ?pn-1 ?o } }
               FILTER (regex(?s, e,'i') || (regex(?o, e,'i'))) } }
     }
```

Increase in the Average Degree. Let E be a set of entities coming from a source S_i. To compute the increase in the average degree of these entities when they "enter" into the warehouse, the following query computes both average values (before and after the entrance to the warehouse) and reports back the increase. Note that that above query considers the "entity matching" policy of Table 2.

```
SELECT ((?avgDW-?avgDS)/?avgDS) as ?IavgD
WHERE { { SELECT xsd:double((count(?in)+count(?out)))
                 /xsd:double(count (distinct ?e)) as ?avgDS
          FROM :Si
          WHERE{ ?e rdf:type :E.
               {?e ?in ?o} UNION {?o1 ?out ?e} } }
        { SELECT xsd:double((count(?in)+count(?out)))
                 /xsd:double(count (distinct ?e)) as ?avgDW
          FROM :W
          WHERE { ?e rdf:type :E .
                 { ?e ?in ?o} UNION {?o1 ?out ?e} } }
      }
```

Time efficiency. Table 4 shows the query execution times for computing the metric *Common URIs* for each of the three policies of Table 2, i.e. *Exact String Equality*, *Suffix Canonicalization* and *Entity Matching*[9]. The first row corresponds to the *pure SPARQL* approach that was presented earlier. The second row corresponds to a *hybrid* approach, where more simple queries are used for

[9] The experiments were conducted using Openlink Virtuoso V6.1, Ubuntu 12.10 64bit, Quad-Core, 4 GB RAM

getting the resources of interest (i.e. the two sets of URIs, one for each source S_i, S_j), and Java code is used for computing their intersection. We observe that the *hybrid* approach is faster than the *pure SPARQL*, as the comparisons are implemented faster in Java. In general, we have observed that the *hybrid* approach loses in time efficiency when the implemented queries return a big amount of data (as in the case of *Unique Triples Contribution*), while it is faster (than *pure SPARQL*) in comparisons.

Table 4. Times (in min) needed to compute metrics on various approaches and policies

Common URIs			
Computation Method	Policy 1	Policy 2	Policy 3
pure SPARQL	7	20	8
hybrid	3	4	4

The following query shows how SPARQL applies *Suffix Canonicalization* on URIs using some functions of Virtuoso. Regarding this policy, the *pure SPARQL* approach becomes less efficient, as the string comparisons cost more when implemented over the endpoint. Regarding the third policy, both approaches are increased by 1 minute. This uniform increase is reasonable as an additional graph that contains the triples with the *sameAs* properties is taken into account.

```
SELECT DISTINCT bif:lower(bif:regexp_substr('[^#|/]+\$',?o,0)) as ?o
FROM :Si {{ ?s ?p ?o } UNION { ?o ?p ?s } FILTER(isURI(?o)) }
```

3.5 Storing and Querying the Values of the Connectivity Metrics

The measurements computed by the aforementioned queries can be represented and exchanged using the VoID extension. They can also be stored in a graph space in the triplestore; indeed *MatWare* can compute and store these triples in a separate graph space in the same SPARQL endpoint (this corresponds to the process "3. store" of Figure 2). Figure 8 gives an example of this procedure for the metric *Common Literals* over the sources *Ecoscope* and *FishBase*. The query both computes the metric and inserts the computed values (expressed using the VoID extension) to a graphspace.

After that, one could retrieve these values from the warehouse using SPARQL queries (this corresponds to the process "4. query" of Figure 2). For example, the query in Figure 9 returns all triples of the warehouse that concern the *common URIs* metric.

4 Concluding Remarks

W3C has proposed VoID as the vocabulary for describing interlinked and open linked datasets. Motivated by a concrete scenario of modeling *connectivity metrics* in the context of an operational semantic warehouse of the marine domain, we have proposed an extension of VoID which is able to represent these metrics.

```
prefix dcterms:<http://purl.org/dc/terms/>
prefix dst:<http://www.ics.forth.gr/isl#>
prefix vdw:<http://www.ics.forth.gr/isl/VoIDWarehouse#>

INSERT INTO dst:Metrics {
    vdw:measurement_2 rdf:type vdw:Measurement ;
    vdw:usesMetric    vdw:commonLiterals;
    vdw:producesValue ?commonLiterals;
    dcterms:date      "2014-03-14T03:19:45"^^xsd:dateTime;
    vdw:carriedOutBy  dst:Matware;
    vdw:over          dst:EcoscopePart;
    vdw:over          dst:FishbasePart . }
    WHERE{{ SELECT (count(distinct ?o) as ?commonLiterals )
            WHERE { graph dst:EcoscopePart { ?s ?p ?o } . FILTER(isLiteral(?o))
                    graph dst:FishbasePart { ?a ?b ?o } } } }
```

Fig. 8. A SPARQL query which both computes the common URIs between all sources of the warehouse and inserts the computed values to a graphspace.

```
PREFIX vdw:<http://www.ics.forth.gr/isl/VoIDWarehouse#>
SELECT DISTINCT ?si  ?sj  ?commonUris
WHERE { {?measurement rdf:type vdw:Measurement .
         ?measurement vdw:usesMetric vdw:commonUris .
         ?measurement vdw:over ?si , ?sj .
         ?measurement vdw:producesValue ?commonUris} .
        FILTER (?si!=?sj)}
ORDER BY (?si)
```

Fig. 9. A SPARQL query which returns the common URIs between all sources of the warehouse

The benefit of the proposed extension is that it allows someone to publish the metrics and their associated values in a standard and machine processable way. Finally, we have shown how the metrics can be computed and we have reported the times required for computing these metrics either using solely SPARQL, or SPARQL and programming language code.

Acknowledgement

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

References

1. S. Auer, J. Demter, M. Martin, and J. Lehmann. LODStats - an Extensible Framework for High-Performance Dataset Analytics. In *Knowledge Engineering and Knowledge Management*, pages 353–362. Springer, 2012.
2. C. Böhm, J. Lorey, and F. Naumann. Creating VoID Descriptions for Web-Scale Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(3):339–345, 2011.
3. R. Cyganiak, S. Field, A. Gregory, W. Halb, and J. Tennison. Semantic Statistics: Bringing Together SDMX and SCOVO. *LDOW*, 628, 2010.
4. F. Darari, W. Fariz, 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.

5. O. Hartig and J. Zhao. Publishing and Consuming Provenance Metadata on the Web of Linked Data. In *Provenance and Annotation of Data and Processes*, pages 78–90. Springer, 2010.
6. 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.
7. M. H. Keith Alexander, Richard Cyganiak and J. Zhao. Describing linked datasets with the void vocabulary, w3c interest group note, 2011.
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. A. Powell, M. Nilsson, A. Naeve, and P. Johnston. Dublin core metadata initiative - abstract model, 2005. White Paper.
12. M. Sabou, I. Arsal, and A. M. Braşoveanu. Tourmislod: A tourism linked data set. *Semantic Web*, 4(3):271–276, 2013.
13. 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.
14. Y. Tzitzikas, C. Lantzaki, and D. Zeginis. Blank Node Matching and RDF/S Comparison Functions. In *International Semantic Web Conference (1)*, pages 591–607. Springer, 2012.
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.
16. Y. Tzitzikas, N. Minadakis, Y. Marketakis, P. Fafalios, C. Allocca, M. Mountantonakis, and I. Zidianaki. MatWare: Constructing and Exploiting Domain Specific Warehouses by Aggregating Semantic Data. In *11th Extended Semantic Web Conference (ESWC'14)*, Anissaras, Crete, Greece, May 2014.