Accessing and Reasoning with Data from Disparate Data Sources Using Modular Ontologies and OBDA

Georgios Santipantakis
Department of Digital Systems, University of Piraeus
Karaoli & Dimitriou 80
Piraeus Greece
gsant@aegean.gr

Konstantinos I. Kotis
Department of Digital Systems, University of Piraeus
Karaoli & Dimitriou 80
Piraeus Greece
kotis@aegean.gr

George A. Vouros
Department of Digital Systems, University of Piraeus
Karaoli & Dimitriou 80
Piraeus Greece
georgev@unipi.gr

ABSTRACT

This paper proposes a distributed framework for accessing, integrating and reasoning with data from heterogeneous, disparate data sources. The proposed solution combines the $E-\mathcal{SHIQ}$ modular ontology representation framework with the Ontop ontology-based data access (OBDA) technology. Distribution of knowledge allows the treatment of data from disparate sources in an autonomous manner, parallelization of operations, while it allows more efficient reasoning with the data. [1].

Keywords

Distributed ontology-based data access, modular ontology, data integration, event recognition

1. INTRODUCTION

The correlated exploitation of multiple, disparate, heterogeneous data sources is important to increasing the accuracy of computations for recognizing and predicting events about entities in surveillance systems. This is extremely important for instance to the maritime domain, where real-time tracking of trajectories and early recognition of events related to vessels are important to safety, cost, dependability, credibility and environmental-friendliness of operations at sea. Major obstacles to realize this goal, include accessing large volumes of data from disparate and heterogeneous data sources, integrating data, populating agreed models (e.g. ontologies) providing an enriched, coherent and integrated view about entities at specific times, supporting query-answering, reasoning and decision making. For example, the detection of events that may result to dangerous/disastrous situations in the maritime domain [14], [13], [4] is a crucial functionality for marine traffic monitoring systems, supporting the prevention of vessel accidents, reducing financial cost for shipping companies, and averting irrevocable damages to maritime ecosystems. Nevertheless, this is not in the general case a straightforward task due to the heterogeneity of data that need to be exploited (at the symbol/data, or the modeling levels), the large volume of data that has to be accessed, and high frequency of dynamic data updates. Furthermore, the application of reasoning tasks on raw data may require prohibitively long computational periods.

The motivation behind this work is the detection of complex critical events in the marine traffic domain: Knowledge about complex events is represented by means of ontology classes that specify patterns comprised from low-level events (e.g. low vessel speed), specific spatio-temporal conditions (e.g. vessels close to each other at a time instant) and domain-specific restrictions (e.g type of vessels). Evaluating these patterns require reasoning with expressive ontologies that are populated by means of information compiled from disparate data sources. Thus, there are a number of issues that need to be addressed: (a) Accessing data from voluminous and heterogeneous data sources, (b) processing data while they are being retrieved so as to extract semantic information, balancing between the complexity of accessing data and reasoning with data, (c) populating ontologies, and (d) reasoning with expressive ontologies. This paper focuses in the three first issues, clearly targeting to efficient reasoning.

The paradigm of using a single, monolithic ontology to represent, manage and reason with data from different data sources, fails to provide an efficient solution to handling big volumes of heterogeneous data. On the other hand, the use of multiple ontologies (or parts of a single ontology) for describing data in different sources presents many advantages, but it requires the coupling of ontologies (both at the assertional and conceptual levels) and their intertwined exploitation. These issues demonstrate that the problem of semantically exploiting and managing data from multiple sources is not a trivial problem in the general case. Furthermore, the existence of multiple, disparate data sources indicate the use of multiple ontologies, which must be coupled to enable unified data interpretations and reasoning with all data made available.

This paper proposes a distributed framework for accessing, integrating and reasoning with data from heterogeneous, disparate data sources. The proposed solution combines the $E-\mathcal{SHIQ}$ [12] modular ontology representation framework

[1]This work was carried out in the framework of the project AMINESS: "Analysis of maritime information for Environmentally Safe vessel shipping" funded by the General Secretariat for Research and Technology (GSRT), Greece (http://aminess.eu/

SEMANTiCS '15, September 15-17, 2015, Vienna, Austria
© 2015 ACM. ISBN 978-1-4503-3462-4/15/09...$15.00
DOI: http://dx.doi.org/10.1145/2814864.2814874

with the Ontop ontology-based data access (OBDA) technology [10]. Distribution of knowledge to multiple ontologies allows the treatment of data from each of the disparate sources in an autonomous manner, parallelization of OBDA operations, effective population of ontologies and distributed reasoning with the data, providing a solution to manage and reason with data in peer-to-peer systems. The paper demonstrates the application of the proposed approach in the maritime domain and evaluates its merits.

The contributions of this work can be summarized as follows: (a) We support the exploitation of distributed knowledge bases (modular ontologies) for accessing and reasoning with data from disparate data sources, (b) we incorporate solutions for processing data close to the data sources (i.e. when they are retrieved or as they are being retrieved) so as to populate ontologies with compiled information, saving reasoning time, (c) providing support for efficient data access, information integration and reasoning tasks, by means of inherent support to parallelization. Although not a contribution of this paper, we have to state that we do not replicate data neither assume permission to alter contents of data sources (i.e. sources maintain their autonomy). Finally, the proposed framework enables reasoning beyond the OWL2QL profile [1], supporting the expressive representation of event patterns.

The paper is structured as follows: Preliminaries on OBDA and the $E-\mathcal{SHIQ}$ framework are presented in section 2. The overall architecture of the framework is presented in section 3 and its instantiation to the marine traffic domain is presented in section 4. The paper provides experimental results on real data in section 5 showing the merits of our proposal. Section 6 presents related work and the paper concludes with an outlook on further work in section 7.

2. BACKGROUND KNOWLEDGE

This section provides preliminaries on the $E-\mathcal{SHIQ}$ knowledge representation framework, and on the OBDA framework Ontop.

2.1 The $E-\mathcal{SHIQ}$ Framework

The $E-\mathcal{SHIQ}$ knowledge representation framework [12] provides constructors for coupling ontology units (or simply *units*: These can be ontologies constructed independently, or modules of a single ontology) that are within the $\mathcal{SHIQ}$ fragment of Description Logics. $E-\mathcal{SHIQ}$ enables constructors for specifying correspondences between concepts in different units, for relating individuals in different units via link relations (representing domain-specific relations between individuals), as well as for specifying correspondences between individuals in different units (representing equalities between them). Link relations, as inter-unit roles, may be further restricted via value and cardinality restrictions, and they can be hierarchically related with other link relations from the same unit. Correspondences and link relations specifications are subjective, i.e. from the point-of-view of a specific unit.

Formally, given a non-empty set of indices I and a collection of units indexed by I, let N_{C_i}, N_{R_i} and N_{O_i} be the mutually disjoint sets of concept, role and individual names of the i-th unit, respectively. An i-role axiom is either a role inclusion axiom or a transitivity axiom. Let $\mathcal{R}_i$ be the set of i-role axioms. Let $\mathcal{E}_{ij}$ be the set of ij-link relations relating individuals in units i and j, $i \neq j \in I$. Link relations are not pairwise disjoint, but are disjoint with respect to the set of concept names. An $ij-relation\ box$ $\mathcal{R}_{ij}$ includes a finite number set of $ij-link\ relation$ inclusion axioms in case $i \neq j$, and transitivity axioms of the form $Trans(E,(i,j))$, where E is in $(\mathcal{E}_{ij} \cap N_{R_i})$, i.e. it is an ij-link relation and an i-role. Subsequently we use the term *property* to refer to either roles or link-relations.

C	$C^{\mathcal{I}_i} \subseteq \Delta_i$
$\top_i$	$(\top_i)^{\mathcal{I}_i} = \Delta_i$
$\bot_i$	$(\bot_i)^{\mathcal{I}_i} = \emptyset$
	$R^{\mathcal{I}_i} \subseteq \Delta_i \times \Delta_i$, $(E_{ij})^{\mathcal{I}_{ij}} \subseteq \Delta_i \times \Delta_j$
	$(Inv(R))^{\mathcal{I}_i} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}_i}\}$
$\neg C$	$(\neg C)^{\mathcal{I}_i} = \Delta_i \setminus C^{\mathcal{I}_i}$
$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}$
$(C \sqcup D)$	$(C \sqcup D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}$
$(\exists R.C)$	$(\exists R.C)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \exists y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i}, y \in C^{\mathcal{I}_i}\}$
$(\forall R.C)$	$(\forall R.C)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \forall y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i}\}$
$(\geq nR.C)$	$(\geq nR.C)^{\mathcal{I}_i} = \{x \in \Delta_i, \Vert y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i} \Vert \geq n\}$
$(\leq nR.C)$	$(\leq nR.C)^{\mathcal{I}_i} = \{x \in \Delta_i, \Vert y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i} \Vert \leq n\}$
$(\exists E_{ij}.G)$	$(\exists E_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \exists y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}}, y \in G^{\mathcal{I}_j}\}$
$(\forall E_{ij}.G)$	$(\forall E_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i \mid \forall y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \rightarrow y \in G^{\mathcal{I}_j}\}$
$(\geq nE_{ij}.G)$	$(\geq nE_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i, \Vert y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \wedge y \in G^{\mathcal{I}_j} \Vert \geq n\}$
$(\leq nE_{ij}.G)$	$(\leq nE_{ij}.G)^{\mathcal{I}_i} = \{x \in \Delta_i, \Vert y \in \Delta_j, (x,y) \in E_{ij}^{\mathcal{I}_{ij}} \wedge y \in G^{\mathcal{I}_j} \Vert \leq n\}$

Table 1: $E-\mathcal{SHIQ}$ i-concepts

The sets of $i-concepts$ (denoted using the prefix "i :") are inductively defined by the constructors shown in table 1, where R and S are i-roles, and E_{ij} are ij-link relations. Let $i : C$ and $i : D$ possibly complex concepts within the $\mathcal{SHIQ}$ fragment of DL and $i : C \sqsubseteq i : D$ (or $i : C \sqsubseteq D$) a *general concept inclusion* (GCI) axiom. A finite set of GCI's is a TBox for i and it is denoted by $\mathcal{T}_i$.

Concept correspondences may be concept *onto* concept, or concept *into* concept: Let $C \in N_{C_i}$, $D \in N_{C_j}$ with $i \neq j \in I$. A concept *onto* (*into*) concept correspondence from i to j that holds for j, is of the form $i{:}C \overset{\sqsupseteq}{\rightarrow} j{:}D$ (corresp. $i{:}C \overset{\sqsubseteq}{\rightarrow} j{:}D$)[2].

An $E-SHIQ$ *distributed knowledge base* $\Sigma = \langle \mathbf{T}, \mathbf{R}, \mathbf{C} \rangle$ is composed by the distributed TBox $\mathbf{T}$, the distributed RBox $\mathbf{R}$, and a tuple of sets of correspondences $\mathbf{C} = (\mathbf{C}_{ij})_{i \neq j \in I}$ between units. A *distributed TBox* is a tuple of TBoxes $\mathbf{T} = \langle \mathcal{T}_i \rangle_{i \in I}$, where each $\mathcal{T}_i$ is a finite set of i-concept inclusion axioms. A *distributed RBox* is a tuple of ij-property boxes $\mathbf{R} = \langle \mathcal{R}_{ij} \rangle_{i,j \in I}$, where each $\mathcal{R}_{ij}$ is a finite set of property inclusion axioms and transitivity axioms. A *distributed ABox* (DAB) includes a tuple of local ABox'es $\mathcal{A}_i$ for each unit i, and sets $\mathcal{A}_{ij}, i \neq j$ with subjective individual correspondences of the form $j{:}a \overset{=}{\mapsto} i{:}b$, from the subjective point of view of i, and property assertions of the form $(a,b) : E_{ij}$, where E_{ij} is an ij-link relation in $\mathcal{E}_{ij}, i \neq j$. A distributed knowledge base forms a network of associated (via correspondences and link relations) units.

As an example of an $E-\mathcal{SHIQ}$ distributed knowledge base $\Sigma = \langle \mathbf{T}, \mathbf{R}, \mathbf{C} \rangle$, let us consider two ($I = \{1,2\}$) units O_1 and O_2, where the distributed RBox is:
$\mathcal{R} = ((R_i)_{i \in I}, (R_{ij})_{i \neq j \in I})$, $R_i = R_{ij} = \emptyset$, $i,j \in I$, the distributed TBox $\mathbf{T}$ is:
$\mathcal{T}_1 = \{Dangerous \equiv Tanker \sqcap \exists within.ProtectedArea$,
$Dangerous \sqsubseteq \forall tuggedBy.TugBoat$,
$FishingArea \sqsubseteq ProtectedArea\}$,
$\mathcal{T}_2 = \{TugBoat \sqsubseteq Vessel, TankerVessel \sqsubseteq Vessel\}$,
and a tuple of sets of correspondences $\mathbf{C}$ is:

[2]Equivalence subjective correspondences are specified by means of *onto* and *into* subjective correspondences.

$\mathbf{C}_{21} = \{2 : TankerVessel \overset{\equiv}{\rightarrow} 1 : Tanker\}$
$\mathbf{C}_{12} = \{1 : Tanker \overset{\equiv}{\rightarrow} 2 : TankerVessel\}$,

The $E - \mathcal{SHIQ}$ distributed reasoner implements a complete and sound tableau algorithm for combining local reasoning chunks corresponding to the individual units in a peer-to-peer fashion, where each peer is responsible for a single unit, inherently supporting the propagation of subsumptions between peers.

2.2 The Ontop OBDA solution

State of the art ontology-based data access (OBDA) systems provide a unified query-interface to support ontology-mediated access to data, and do not inherently address the data integration problem. Open source OBDA frameworks that support accessing the content of relational databases and transforming them to virtual (read-only) RDF graphs are the D2RQ [3] and Ontop[11]. Relational database sources are transformed to RDF graphs by utilizing RDB to RDF mappings, using a specific mapping language (e.g. the D2RQ or R2RML mapping languages).

To specify the mappings there are three approaches: The local-as-view (LAV) approach that describes the data of $\mathcal{S}$ by means of conjunctive queries against $\mathcal{O}$, the global-as-view (GAV) approach that defines concept and role specifications in $\mathcal{O}$ by means of conjunctive queries against the source data base $\mathcal{S}$, and finally, the global-local-as-view (GLAV), which states equivalences between queries against the source database and queries against the ontology. While LAV approaches are more robust to changes in the sources, GAV allow more efficient query processing techniques and GLAV do combine these two approaches.

In this wok we are using the Ontop [10] whose performance takes advantage of optimized SPARQL-to-SQL query rewriting techniques [11], to produce as simple as possible SQL queries, following the GAV approach. Mappings in Ontop are expressed either in a custom, easy to read mapping language, or in the RDB to RDF mapping language R2RML[3]. According to the GAV approach, a mapping is a set of rules of the form $\sigma(\mathbf{x}) \rightarrow E(\mathbf{x})$, where $\sigma(\mathbf{x})$ (the *source* part of the mapping) is a conjunction of atoms with database relations or views, and a filter which is a boolean combination of built-in predicates (i.e. an SQL query over $\mathcal{S}$), and E (target part of the mapping) is a functional expression that creates new instances in the ontology.

Concerning the semantics of the virtual ABox constructed by means of data from the source, let $\Delta\Sigma = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA system and δ a database instance w.r.t. $\mathcal{S}$. We say that an interpretation $\mathcal{I}$ is a model of $\Delta\Sigma$ w.r.t. δ if it satisfies $\mathcal{O}$ and the data in δ over the mapping $\mathcal{M}$, i.e. $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models_{\mathcal{M}} \delta$, where the definition of $\models_{\mathcal{M}}$ depends on the mapping approach. Specifically, for the GAV approach we say that $\mathcal{I} \models_{\mathcal{M}} \delta$ if there is an ABox $\mathcal{A}$ s.t. $\mathcal{A}_\delta \subseteq \mathcal{A}$ and $\mathcal{I} \models \mathcal{A}$, where $\mathcal{A}_\delta$ is the ABox produced from applying $\mathcal{M}$ on δ.

An Ontop GAV mapping in the maritime domain that recognizes the tanker vessels as individuals of the concept `Tanker` $\in N_C$, assuming that vessel types are available in more than one table, i.e. in the tables `R1`, `R2` is as follows:

source:
```
SELECT imo, vessel_name as vName, typeStr as vType FROM R1
WHERE (typeStr LIKE '%TANKER%') OR (typeStr LIKE '%BUNKER-
ING%') OR (typeStr = 'CHEMICAL') OR (typeStr = 'CONTAINER')
OR (typeStr = 'CRUDE') OR (typeStr = 'LNG') OR (typeStr =
'LPG') UNION
SELECT imo_no as imo, name as vName, type_ as vType FROM R2
WHERE (type_ LIKE '%TANKER%') OR (type_ LIKE '%BUNKERING%')
OR (type_ = 'CHEMICAL') OR (type_ = 'CONTAINER') OR (type_
= 'CRUDE') OR (type_ = 'LNG') OR (type_ = 'LPG')
```
target:
```
:vessel{imo} a :Tanker ; :hasName {vName} ; :typeOf {vType}.
```

where regular expressions are applied on the values of typeStr and type_ to select the appropriate tankers from the data source.

3. SYSTEM ARCHITECTURE

This section presents the overall architecture of the proposed framework, which is sketched in Fig. 1. The key idea is to distribute knowledge in multiple and interlinked ontology units. Each ontology unit is managed by a single peer who can be connected to data sources via OBDA or via SPARQL endpoints. Peers take advantage of this distribution to inherently parallelize data access, integration and reasoning tasks. Concerning data access, peers access data by means of distinct connections to data sources (connections run in parallel for each peer). Considering data integration, peers must convert data to common formats, and couple their knowledge by means of $E - \mathcal{SHIQ}$ correspondences and link relations between instances. The computation of instances' correspondences is necessary for peers to perform reasoning tasks jointly, and is performed after the population of peers' units. This is further discussed in section 3.1. Beyond these, we aim at efficient ABox reasoning beyond the OWL2QL profile supported by Ontop-Quest, to more expressive ontologies within the $\mathcal{SHIQ}$ fragment of DL. This can be satisfied by the distributed reasoner $E - \mathcal{SHIQ}$ [12].

Thus, the proposed framework enables coupling and reasoning with distinct ontologies which are populated by concept instances and properties constructed from data accessed by multiple OBDA instances. The number of OBDA instances used may vary depending on units, sources and specific requirements to be fulfilled: We do not assume one-to-one correspondences between units and data sources. This means that a unit may use multiple OBDA instances to access data from disparate data sources, or there may be multiple OBDA instances for accessing data for the purposes of different peers from the same data source. Of course, instances of different OBDA frameworks may be used here in the most general case.

Given that data sources may provide data whose values change with time, we make the distinction between static and dynamic data. This allows us to take advantage of speeding-up the access to data at every time instant, to process static data only once - at system initiation, and also to reduce the amount of data the units' ABox's hold by updating the dynamic data when they "expire". Generally, the data access can be seen as a time dependent function $\mathbf{d} = f(q, T)$, where for a given time instant or period T and a query q, the function returns a specific set of results $\mathbf{d}$. Given a query q for static data the time variable T is infinite (i.e peers do access data only once), while for dynamic data the values of T depends on the frequency of data up-

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

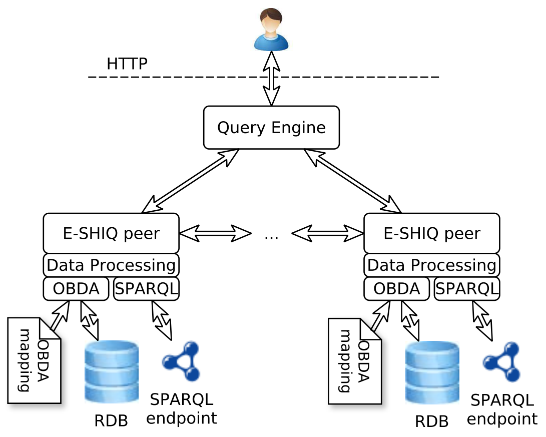


Figure 1: The system architecture

dates in the sources and may vary depending on the type of data. In this work we consider that all dynamic data are updated at specific time intervals and are accessed at the end of that interval. The duration of this interval is a parameter that must be configured appropriately. Having said this, it must be said that this work does not have the objective to process streaming data in real-time, although we do process dynamically changing data stored in a database. This is an issue that is further discussed in subsequent sections.

While distribution of knowledge and data is a key issue, a second key issue is the processing of data before populating the ontology and if possible while they are being accessed: The goal is to save processing and, most importantly, reasoning time, although the data access time may increase due to the complexity of queries to the sources. This is in contrast to populating units with raw data retrieved by the sources, which is still an option. Data processing functions for extracting information out of data can be distinguished into three main categories: a) Functions for the transformation of data to a common format, b) functions for the realization of concept instances, and c) functions for the detection of domain-specific relations between instances. To achieve a good balance between data access and data processing/reasoning, we propose the use of *triple templates*: These can be used to generate ABox assertions both for instances' types and for instances' properties. These are presented in section 3.2 and their use is evaluated in section 5.

Nevertheless, computing and asserting relations between instances may be computationally demanding: For instance, in the maritime domain we have to compute spatio-temporal relations among vessels or between vessels and sea areas. In the general case the computation of relations between entities may be demanding if multiple features, specific types of constraints and interrelations among entities must be considered. The computation of these relations is discussed in section 3.3. These computations are hard to be performed by DL reasoners, especially when numerical data are involved in the computations.

Finally, the query engine processes SPARQL-DL queries submitted by front-end clients, and directs them to the corresponding peers in the network: This part of the architecture has been implemented in a rather straightforward way and it is beyond the scope of this article. Further work concerns the implementation of a sophisticated query engine that takes full advantage of $E-\mathcal{SHIQ}$ constructors and the distribution of knowledge to units.

3.1 Data Integration

Data integration concerns (a) the transformation of data from different sources to common formats, and (b) the detection of equalities between concept instances.

OBDA mappings can use SQL functions and other operators to convert data to a common format. For example, geometries and positions may be required in the Well Known Text (WKT) format, however data sources may provide location information using the longitude (lon) and latitude (lat) fields. In these cases the WKT for the position of a vessel can be produced by the mapping:

source:

```
SELECT 'POINT(' || lon || ' ' || lat || ')' as wkt, lon,
lat FROM R1
```

target:

```
:SpObj{lat}_{lon} a :SpatialObject ;
geosparql:Geometry {wkt}
```

where the position of each `SpatialObject` instance is assigned as a WKT value to the datatype property `geosparql:Geometry`. The mapping in this example constructs the WKT using text concatenation in the source part of the mapping. As another example from the maritime domain, geometries (potentially referenced in different geodetic reference systems) can be also transformed to a specific reference system using the `st_transform(geom, srid)` SQL function.

Equalities between concept instances are distinguished to equalities between instances in a specific unit, and equalities between instances in different units. The former is necessary when a peer accesses data from different sources and has to unify entities and consolidate data about them. The latter results to instance correspondences between units and in the general case requires the collaboration of peers in the $E-\mathcal{SHIQ}$ network. To detect equalities, peers consult patterns comprising property-value pairs: These patterns are specified per ontology concept, allowing the computation of equalities between individuals of corresponding concepts (detailed below). Patterns are specified by domain experts w.r.t. the ontology units describing the data in sources, taking into account OBDA mappings for the retrieval of data. When two instances match a single pattern, then their equality is assessed. It must be noticed that peers compute subjective equalities, and thus, in the general case, they may need to reach agreements on their instance correspondences. In this work, due to the specific needs of the integration task, we consider that any instances' correspondence computed for $\mathcal{O}_i$ holds for $\mathcal{O}_j$, as well. This saves processing time for peers.

The computation of instances' equalities is restricted to corresponding concepts: Let us assume that data retrieved from a data source S by some peer i populate the ontology unit $\mathcal{O}_i$. According to the query issued to the source, the entities whose data are retrieved realize specific concepts, $i : C_k$, $k = 1, 2 ...$. Given concept correspondences of the form $j : D \overset{\#}{\to} i : C_k$ that peer i holds, $\# \in \{\equiv, \sqsubseteq, \sqsupseteq\}$ and according to the semantics of correspondences, any individual x with $C_k(x)$ may correspond to an instance of concept $j : D$ in ontology $\mathcal{O}_j$. To perform the matching, the peer i sends a matching-request message to peer j, containing the known instances, and the data patterns for the matching instances. Thus, for the individual x and pattern p_k, the peer i requests all the individuals $y \in N_{O_j}$, that satisfy p_k and $j : D(y)$. The peer j matches own instances to the pattern specified and reports back to peer i. In case of equivalence or *onto* correspondences, and in case no individual of peer j matches the pattern, peer i must inject new individuals in unit $\mathcal{O}_j$, to preserve the semantics of individual correspondences. This method for computing individual correspondences benefits from the distribution of data to peers, while preserving the subjectivity of computations.

3.2 Triple Templates

For the recognition of instances' types and relations, sufficient data about these instances must be accessed from the sources. This may increase the reasoning time for realizing these instances, and may result to complex queries that access large amounts of data, for all possible instances. To address this issue and aiming to achieve a good balance between the complexity of reasoning and the complexity of accessing data, we have implemented a wrapper on the SPARQL query engine, which allows the definition of triple templates for the automatic construction of triples from the query results. Specifically, each template is signaled by the keyword `ASSERT` and variables set in the SPARQL query. When the query is served, each variable is grounded to a specific value, forming additional triples to the query results. The `ASSERT` should not be confused with the standard `SPARQL 1.1 INSERT` and `CONSTRUCT` keywords[4]. The `INSERT` is used to manually insert data in a given graph, while `CONSTRUCT` will substitute the variables in the template with values in the result set from the `WHERE` part. Both commands affect the RDF graph with the constructed triples. The `ASSERT` on the other hand, appends the results of a query with additional triples, according to the template. Thus, it does not update the data source graph (in the general case we may neither have the permission to do so), it appends the results with triples that leverage the reasoning process.

For example, the safety regulations for vessels of International Maritime Organization[5] define that a high-speed craft is capable of a maximum speed, in metres per second (m/s), equal to or exceeding 3.7 times the one-sixth power of the volume of displacement corresponding to the design waterline (m^3). This specification can be used to realize `High-Speed` individuals in the corresponding ontology. Although this specification is hard to be realized using standard DL, it can be easily specified using a triple template as follows:

```
ASSERT ?x a :HighSpeed
SELECT ?x ?m ?d
WHERE{ ?x :maxSpeed ?m ; :draught ?d .
FILTER(?m*6>3.7*?d) .  }
```

A consequence of using templates, is that peers easily adapt to changes in ontologies and data sources. The fact that the processing of the triples is done by the peers, indicates that no modification is required on the data sources or on the OBDA mappings. On the other hand, if the specifications in an ontology unit, or the OBDA mappings for a data source are modified, queries can be tailored with the appropriate `ASSERT` statements to adapt to these modifications.

[4]http://www.w3.org/TR/sparql11-query/

[5]http://www.imo.org/OurWork/Safety/Regulations/Pages/Default.aspx

3.3 Computation of instances' relations

Although triples templates support peers to make property assertions in their Aboxes, the assertion of relations that require more elaborated computations in an n-ary space (i.e by considering n instances' features) need to be supported. An example of such relations are the spatio-temporal relations, which concern at least 3 features of the instances.

The main -and widely used idea- is to first organize the n-dimensional space into a predefined number of disjoint and adjacent segments. Each individual is "placed" in the corresponding segment of the n-dimensional space. Individuals corresponding to the same segment form a group. When all individuals are known, relations are computed only between individuals of the same segment. This can be done in parallel for each unit and for the different segments. Finally, possible relations between individuals that fall near the common edge of adjacent segments are computed. The proposed framework benefits from the concurrency of these computations, and from the fact that OBDA mappings, queries and ontology specifications are kept as simple as possible.

4. REASONING WITH MARITIME DATA

Based on the proposed framework, we have implemented a system in the maritime domain, using an $E - \mathcal{SHIQ}$ distributed knowledge base comprising 4 coupled ontology units, and 2 disperse and heterogeneous data sources for the population of the distributed ABoxes. The data sources are two relational databases, namely Hermes and AMINESS providing data concerning vessels position and trajectories at each time instant, and data concerning weather, vessels' characteristics, ports, protected areas. It must be mentioned that Hermes provides dynamic data about vessels' positions, and vessels status, which result from compiling streaming data into synopses, achieving a high degree of data compression (nearly 90%). Thus, dynamic data include the position, status and trajectories of vessels, and weather reports. The rest are static data. Ontologies, as shown in oval in Fig. 2, specify complex events' patterns and simple events concerning vessels status, provided by Hermes (ontology Events), trajectories of vessels and critical points along these trajectories (ontology Hermes), and finally, static and dynamic data from the AMINESS database concerning ports, areas, and weather (these data may also be retrieved from SPARQL endpoints or be extracted from WWW sources). Dynamic data are updated every T_s time points: This period is configured appropriately according to specific needs. An update is triggered by each $E - \mathcal{SHIQ}$ peer, requesting dynamic data from the data source(s) and updating the ABox of the ontology it owns. Each peer is responsible for an ontology unit, and consults OBDA mappings to access data from the corresponding data sources.

For instance, to demonstrate the parallel access to data sources, distinct queries for different vessel types are issued for accessing vessels' data. Similarly, regarding dynamic data, the system issues queries with triple templates for recognizing the type of simple events: Specifically, the simple events provided by the Hermes data source are identified by a "flag", which is the value of the data property `:status`

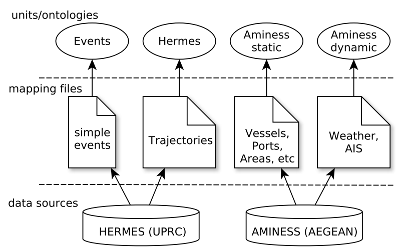


Figure 2: Data sources and coupled ontologies in the maritime domain

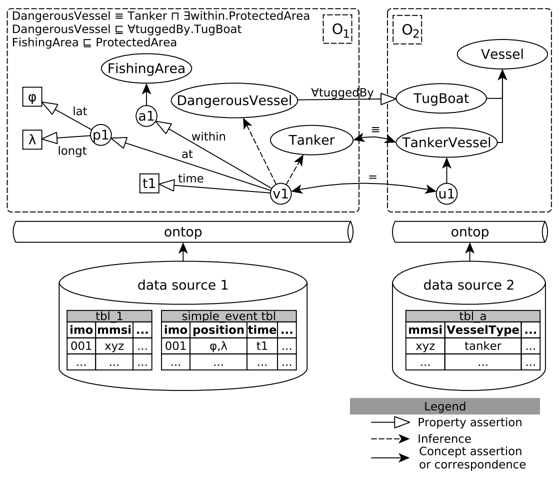
and ranges in [1,5]. Each value corresponds to one of the simple event types [NotInPort, LowSpeed, LostCommunication, Turn, InPort]. To assert the appropriate information in the corresponding unit, the following queries are issued and processed concurrently:

```
ASSERT ?e a :NotInPort
SELECT * WHERE { ?e :status ?k .  FILTER (?k=1)}
ASSERT ?e a :LowSpeed
SELECT * WHERE { ?e :status ?k .  FILTER (?k=2)}
ASSERT ?e a :LostCommunication
SELECT * WHERE { ?e :status ?k .  FILTER (?k=3)}
ASSERT ?e a :Turn
SELECT * WHERE { ?e :status ?k .  FILTER (?k=4)}
ASSERT ?e a :InPort
SELECT * WHERE { ?e :status ?k .  FILTER (?k=5)}
```

Obviously, an alternative to this is to retrieve all simple events using a single query, and let the DL reasoner compute the type of each event, but given the number of events to be processed, this should be more expensive: The next section provides experimental results concerning this issue.

To recognize events, spatio-temporal relations concerning (a) the proximity between vessels (represented by the relation `nearBy`) and (b) the inclusion of a vessel within an area (represented by the relation `within`) must be known in advance. Data are processed to identify this kind of information. The computation of these relations require three features of the objects involved: The coordinates (i.e. longitude, latitude) of their positions/polygons and the time instant/period they had that position. For example, two individuals x, y s.t. *geosparql:Geometry(x,gm1)* and *geosparql: Geometry(y,gm2)* (where *gmX* is a point or polygon in Well Known Text format) are *nearby* if the distance of $gm1, gm2$ is at most equal to a distance threshold D_{th}, and the time difference of the observed positions is no more than $Thesh$. Equalities between concept instances in different units (mostly vessels) are computed as described in section 3.1.

To demonstrate the overall process of combining data from different sources, we consider the setting of Fig. 3 of two peers, each one responsible for an $E-\mathcal{SHIQ}$ unit ($\mathcal{O}_1$, $\mathcal{O}_2$). On the system start up, peers retrieve static data concerning concepts of their own units. We assume that static data for data Source 1 in Fig. 3 concern protected areas (e.g. national parks, fishing areas, etc) with the corresponding geometries, while static data for data Source 2 in Fig. 3 concern vessels (e.g. MMSI, vessel type, vessel name etc). When the static data are retrieved and the first update is triggered, the ontology $\mathcal{O}_1$ will be updated with the infor-



tbl_1

imo	mmsi	...
001	xyz	...
...	...	...

simple_event tbl

imo	position	time	...
001	φ,λ	t1	...
...	...	...	...

tbl_a

mmsi	VesselType	...
xyz	tanker	...
...	...	...

Figure 3: Reasoning with maritime data

mation that vessel $v1$ is located at point $p1$ with latitude and longitude ϕ, λ at time $t1$. The alignment function investigates whether there are any individual correspondences between other units. Indeed, given that $v1$ in data Source 1 has MMSI value xyz a correspondence between $v1$ in $\mathcal{O}_1$ and $u1$ in $\mathcal{O}_2$ is established. Next, the spatial relations are computed and it is asserted that the position $p1$ at time $t1$ is within a fishing area $a1$ ($(v1, a1) : within$). Given that $v1$ is of type Tanker and $FishingArea \sqsubseteq ProtectedArea$ and $DangerousVessel \equiv Tanker \sqcap \exists within.ProtectedArea$, the $E-\mathcal{SHIQ}$ distributed reasoning mechanism infers that $v1$ is a dangerous vessel.

5. BALANCING BETWEEN REASONING AND ACCESSING DATA

The two key issues of the proposed approach concern the distribution of knowledge to peers, enabling parallel execution of data access, integration and reasoning tasks, and the processing of data to enable efficient reasoning in balance with efficient data access. Concerning distribution of knowledge, we have shown in [12] that the $E-\mathcal{SHIQ}$ distributed reasoner -compared to a standard centralized reasoner- can do better if knowledge is distributed effectively among peers. However, in this work we evaluate our proposal for balancing between efficient OBDA retrieval and reasoning while populating the ontology units. Specifically, we evaluate the use of triple templates on realizing simple events concepts with data retrieved from the data sources: Thus the reasoning demonstrated here concerns the realization of simple events and checking the consistency of the distributed ontology. The other techniques used in the proposed framework for the computation of relations and for data integration, although they take full advantage of the inherent parallelism allowed by the framework, can be replaced with more advanced techniques and are not novel contributions, nor do they constitute irreplaceable parts of the proposal. For these purposes we do not evaluate these here.

To demonstrate the efficiency of queries with templates we use data about simple events that associate the ID of a vessel with the type of the event (as shown in the previous section), the time, and position where this event occurred.

These are dynamic data and concern the route of numerous vessels for a large time period in the Aegean Sea. In conjunction to queries with triple templates we measure the efficiency of two alternatives: The use of "Simple" and "Conditional" queries.

Simple: This approach uses a simple OBDA mapping. Given a SPARQL query for a time period, retrieves all the simple events and populates the ontology with the raw data, inferring the type of each event, and checking the consistency of the distributed ontology. Specifically, this approach uses the following mapping:

```
source: SELECT id, time, lon, lat, status FROM R1
target: simpleEventid_time_lat_lon_status:status status;
:associatedTime :TimeObjecttime ; :associatedWithArea
:SpObjectlat_lon ; :eventAssociatedWithVessel :vesselid .
```

Conditional: This approach uses an OBDA mapping for each simple event type (represented by an ontology concept), which recognizes the type of the event using a WHERE clause. For example, for the simple event `Turn` the value of the property `status` is "3" and the OBDA mapping is:

```
source: SELECT id, time, lon, lat, status FROM R1
WHERE status = '3'
target: :eventTurn{id}_{time} a :Turn ;
:associatedTime :TimeObject{id}_{time} ;
:associatedWithArea :SpObject{lat}_{lon} ;
:eventAssociatedWithVessel :vessel{id} .
```

Template: This approach uses the Simple OBDA mapping, but the realization of instances' types is done using templates. For instance, the template for simple events of type `Turn`, is as follows:

```
ASSERT ?e a :Turn
PREFIX : <http://www.aminess.eu/ontology/events#>
SELECT * WHERE
{
?e :associatedWithArea ?x .
?e :associatedTime ?t .
?e :kind ?k .
?t :EventStartEpoch ?time .
FILTER ( (?time >= "##S##") && (?time < "##T##") ).
FILTER (?k = "3")
}
```

where the variables `##S##`, `##T##` represent the starting and terminating time instant of a period.

For each experiment we record the data access time and the time required by the reasoner. We evaluate the scalability of each alternative w.r.t. the amount of retrieved data, varying the time period in the query. Specifically, experiments concern time periods from 10 minutes to 100 minutes with a step of 10 minutes. Tables 2, 3, 4 present the retrieval, reasoning and total time, respectively, for each alternative approach. The first column for each table shows the *Update* time period in seconds, the second column indicates the number of events retrieved, and the third, fourth and fifth columns indicate the time required from each approach, in seconds.

Table 2: Retrieval time

Update	#Events	Simple	Conditional	Template
600	470	0.811	22.891	8.522
1200	1260	0.866	36.274	32.021
1800	2106	0.950	54.067	50.615
2400	2850	0.977	141.130	69.559
3000	3507	1.026	195.985	97.022
3600	4541	1.213	245.263	120.648
4200	5481	1.272	313.324	155.622
4800	6619	1.384	388.991	193.278
5400	7198	1.542	433.557	212.441
6000	8242	1.777	509.294	252.974

Table 3: Reasoning time

Update	#Events	Simple	Conditional	Template
600	470	43.477	1.241	0.499
1200	1260	407.476	0.902	0.860
1800	2106	507.258	1.032	1.363
2400	2850	965.218	1.860	2.174
3000	3507	3096.651	2.175	1.912
3600	4541	5298.594	2.926	3.218
4200	5481	7346.905	2.918	3.777
4800	6619	88071.777	3.316	4.538
5400	7198	111355.552	4.116	42.916
6000	8242	166788.949	4.313	45.668

The results for the retrieval time shown in Table 2 show that the Simple approach has a great advantage over the other two approaches, as expected. A comparison of the results for the Simple versus the Conditional approach, shows the overhead of using WHERE clauses in the OBDA mapping. Similarly, a comparison of the results for the Simple versus the Template approach, shows the overhead of processing the template query. These results indicate that the Template approach is preferable over the Conditional. This can be explained by the fact that mappings for the Conditional approach are a little more complex compared to the other two approaches, which may lead to more complex query translations. On the other hand, for Simple and Template approaches, the mappings are very simple.

The time required for reasoning are shown in Table 3. Clearly, the Simple approach is not scalable, thus it should be used only for solutions where reasoning is not applied. We observe that for an update time period 50 minutes, the reasoning time required is approximately 52 minutes (3098 seconds), thus for this amount of data, this approach has no practical use. On the other hand, the Conditional and Template approaches show similar performance, as expected, because the realization of individuals have been performed during the retrieval of the results.

Finally, the results of Table 4 provide the total time required from data access and reasoning. It is clear that for all the cases any of the Conditional or Template approaches can be used, because they require considerably less time for processing compared to the time period of the update. However, the Template approach shows the best performance in every case of the experiments.

6. RELATED WORK

Early ontology-based data integration approaches such as [15], [13] address the integration problem using a single monolithic ontology describing the available data, also without exploiting the advantages of an OBDA framework.

Table 4: Total time

Update	#Events	Simple	Conditional	Template
600	470	44.288	24.132	9.021
1200	1260	408.342	37.176	32.881
1800	2106	508.208	55.099	51.978
2400	2850	966.195	142.990	71.733
3000	3507	3097.677	198.160	98.934
3600	4541	5299.807	248.189	123.866
4200	5481	7348.177	316.242	159.399
4800	6619	88073.161	392.307	197.816
5400	7198	111357.094	437.673	255.357
6000	8242	166790.726	513.607	298.642

Furthermore, only recently the problem of integrating data using OBDA solutions in settings with dynamic and voluminous data has been given special attention. This problem requires optimization techniques and distributed solutions at the architectural or modeling level in order to be able to efficiently answer translated (sparql-to-sql) high-level ontology-mediated queries.

Optique [8] addresses the challenge of OBDA-based data integration giving emphasis to the optimization of distributed querying in order to allow scalable end-user access to big data [7].The federated query processing engine (FedX) operating on top of autonomous semantic databases enables the virtual integration of heterogeneous sources implementing efficient query evaluation strategies. Optique uses Ontop OBDA system [2] for mapping RDB data to an ontology. Although plans for exploiting ontology modularization techniques have been announced [8], to the best of our knowledge there is not a specific approach towards this.

SPEED [5] is an OBDA instantiation for a Peer Data Management System (PDMS) where queries submitted to a peer are answered with data residing at that peer and with data acquired from neighbor peers through mappings. This work put together OBDA and a visual query system to access geospatial data in a PDMS. However, the reformulation of queries between neighbor peers by taking into account semantic correspondences remains future work. In contrast to our approach, SPEED applies a non-mediated schema approach to OBDA-based data integration. In absence of evaluation results we cannot provide any further details about the scalability of this approach.

Karma system [9] aims at automating the process of mapping data sources to an ontology, emphasizing the modeling of both static and dynamic sources and the integration of new data sources. In close relation to our aims, Karma is able to automatically transform data to RDF using the R2RML model, providing access to the available sources via the Data-Fu Program. This collects data and evaluates queries over the data, while taking into account the semantics of ontology constructs. In contrast to our approach, a shared domain ontology is used to model the data sources and is being used to generate the RDF linked graphs. Current evaluation efforts report a scalable Data-Fu interpreter that scales from small sources (hundreds of triples) to sources of moderate size (hundreds of thousands of triples).

7. CONCLUSIONS

In this work we have presented a framework for accessing, integrating and reasoning with voluminous static and dynamic data from heterogeneous data sources: The key issues are the distribution of knowledge to peers, and the processing of data, before data reach the reasoner, while they are accessed. The framework uses the Ontop OBDA system, and the distributed knowledge representation framework $E-\mathcal{SHIQ}$. Experimental results show that Triple templates used for computing ABox assertions, enable a good balance between data access and reasoning.

Plans for future work include the evaluation of the framework for large networks of peers, and the investigation of methods for query processing and routing in such networks. We plan to investigate methods that will enable the detection of individual correspondences using instance matching techniques and the incorporation of link specifications. We also study the potential of extending the peers with spatio-temporal reasoning capabilities so as to take advantage of the existing inherent parallelism. Finally, we will evaluate the implemented system for the recognition of complex events and will investigate the use of the proposed framework for reasoning on expressive ontologies, combined with streaming data.

8. REFERENCES

[1] OWL 2 Web Ontology Language Profiles (2nd Ed.). http://www.w3.org/TR/owl2-profiles/#OWL_2_QL. Accessed: 2015-06-04.

[2] T. Bagosi, D. Calvanese, J. Hardi, S. Komla-Ebri, D. Lanti, M. Rezk, M. Rodriguez-Muro, M. Slusnys, and G. Xiao. The Ontop Framework for Ontology Based Data Access. In *8th CSWS*, pages 67–77, 2014.

[3] C. Bizer. D2RQ - treating non-RDF databases as virtual RDF graphs. In *3rd ISWC*, 2004.

[4] L. Chen and C. D. Nugent. Ontology-based activity recognition in intelligent pervasive environments. *IJWIS*, 5(4):410–430, 2009.

[5] R. Figueiredo, D. Pitta, A. C. Salgado, and D. Souza. Geographic data access in an ontology-based peer data management system. *JIDM*, 4(2):146–155, 2013.

[6] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *16th AAAI*, pages 67–73, 1999.

[7] M. Giese, D. Calvanese, P. Haase, I. Horrocks, Y. Ioannidis, H. Kllapi, M. Koubarakis, M. Lenzerini, R. MÃűller, M. Rodriguez-Muro, Ã. Özcep, R. Rosati, R. Schlatte, M. Schmidt, A. Soylu, and A. Waaler. Scalable end-user access to big data. In *Big Data Computing*. CRC Press, 2013.

[8] P. Haase, I. Horrocks, D. Hovland, T. Hubauer, E. Jiménez-Ruiz, E. Kharlamov, J. Klüwer, C. Pinkel, R. Rosati, V. Santarelli, A. Soylu, and D. Zheleznyakov. Optique System: Towards Ontology and Mapping Management in OBDA Solutions. In *WoDOOM*, 2013.

[9] A. Harth, C. A. Knoblock, S. Stadtmüller, R. Studer, and P. A. Szekely. On-the-fly integration of static and dynamic sources. In *COLD*, volume 1034, 2013.

[10] R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *13th ISWC*, LNCS. Springer, 2014.

[11] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Query rewriting and optimisation with database dependencies in ontop. *Proc. of DL*, 2013.

[12] G. Santipantakis and G. A. Vouros. Distributed reasoning with coupled ontologies: the $E-\mathcal{SHIQ}$ representation framework. *KAIS*, 41(3):1–44, 2014.

[13] A. Vandecasteele and A. Napoli. An enhanced spatial reasoning ontology for maritime anomaly detection. In *SoSE*, pages 1–6, 2012.

[14] A. Vandecasteele and A. Napoli. Spatial ontologies for detecting abnormal maritime behaviour. In *OCEANS'12*, pages 1–7, 2012.

[15] L. Zamboulis, A. Poulovassilis, and G. Roussos. Flexible data integration and Ontology-Based Data Access to medical records. In *8th BIBE*, pages 1–6, 2008.