NYAYA: a System Supporting the Uniform Management of Large Sets of Semantic Data

Roberto De Virgilio #1, Giorgio Orsi *2, Letizia Tanca §3, Riccardo Torlone #4

Università Roma Tre, Italy
1devirgilio@dia.uniroma3.it
4torlone@dia.uniroma3.it

* *University of Oxford, UK*
2giorsi@cs.ox.ac.uk

§ *Politecnico di Milano, Italy*
3tanca@elet.polimi.it

***Abstract*— We present NYAYA, a flexible system for the management of large-scale semantic data which couples a general-purpose storage mechanism with efficient ontological query answering. NYAYA rapidly imports semantic data expressed in different formalisms into *semantic data kiosks*. Each kiosk exposes the native ontological constraints in a uniform fashion using Datalog$^{\pm}$, a very general rule-based language for the representation of ontological constraints. A group of kiosks forms a *semantic data market* where the data in each kiosk can be uniformly accessed using conjunctive queries and where users can specify user-defined constraints over the data. NYAYA is easily extensible and robust to updates of both data and meta-data in the kiosk and can readily adapt to different logical organizations of the persistent storage. In the demonstration, we will show the capabilities of NYAYA over real-world case studies and demonstrate its efficiency over well-known benchmarks.**

I. INTRODUCTION

Ever since Tim Berners Lee presented the design principles for Linked Data,[1] the public availability of Semantic Web data has grown rapidly. Today, many organizations and practitioners are all contributing to the "Web of Data", by building RDF repositories either from scratch or by publishing in RDF data stored in traditional formats. The adoption of ontology languages such as RDF(S) or OWL supports this trend by providing the means to semantically annotate Web data with meta-data, enabling ontological querying and reasoning.

Despite the fact that storing, reasoning over, and querying large datasets of semantically annotated data in a flexible and efficient way represents a challenging area of research and a profitable opportunity for industry [1], semantic applications using RDF and linked-data repositories set performance and flexibility requirements that have not yet been satisfied by the state of the art of data-management solutions. In particular, current semantic-data management systems present some common shortcomings: (i) they usually operate within a single-language framework, implementing the reasoning and query-processing algorithms for that specific language; (ii) they hardly adapt to requirements of changing the underlying physical organization (e.g., for optimization purposes); (iii) to bring the expressiveness and performance desiderata to terms, most systems unnecessarily restrict the allowed combination of ontology-modeling constructs, although in many cases decidability and tractability of reasoning and query answering are syntactically identifiable on a per-ontology basis.

We show possible solutions to these problems by presenting NYAYA,[2] an environment for Semantic-Web data management which provides, in the same order as above:

(i) flexible and uniform ontology-reasoning and querying capabilities over semantic data sets expressed in different formalisms;
(ii) an efficient and, most importantly, general and extensible storage policy for Semantic-Web datasets, which can adapt to different query engines, exploiting their optimization strategies;
(iii) the possibility to syntactically check whether the meta-data in a given repository can be queried efficiently and to use the subset of the language that keeps the complexity of the process at the required level.

The last aspect is particularly significant. Suppose for instance that the combined use of two language constructs could potentially lead to undecidability or to a complexity increment; normally, with a Semantic Web language the two constructs would not appear in the language together, thus severely reducing expressiveness. Conversely, NYAYA does not restrict a-priori the language but checks each set of meta-data for decidability and tractability.

Reasoning and querying in NYAYA is based on Datalog$^{\pm}$[2], a family of languages that captures the most common languages for which ontological query answering is tractable, and provides efficiently-checkable, syntactic conditions for decidability and tractability.

In the rest of the paper we first provide, in Section II, an overview of NYAYA, pointing out the novel features. We then give, in Section III, an outline of our demonstration and finally, in Section IV we sketch some conclusions. Further technical details can be found in the research paper [3].

[1]http://linkeddata.org/

[2]Nyaya is the name of the school of logic in the Hindu philosophy.

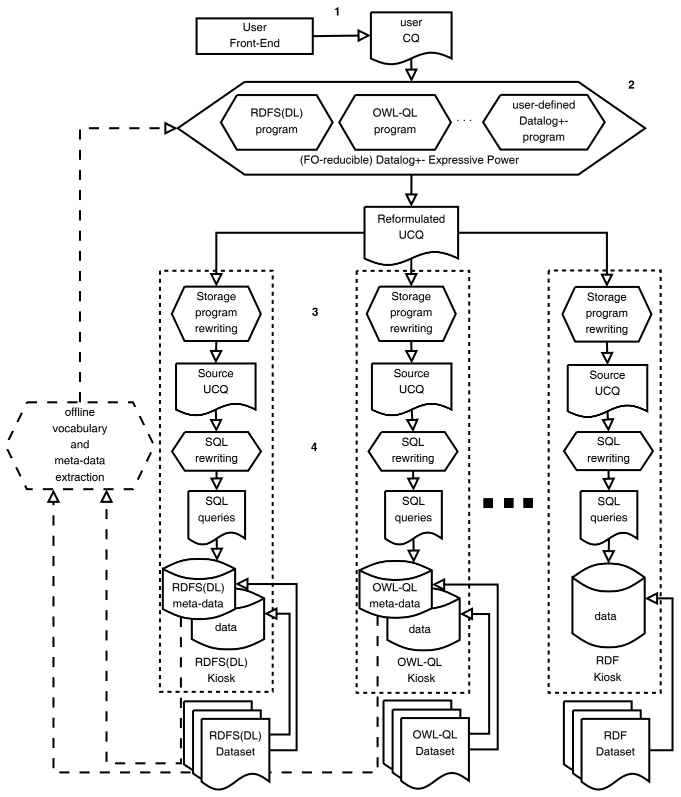


Fig. 1. A Semantic Data Market

II. System Overview

A. Semantic Data Markets

NYAYA operates over persistent repositories of Semantic-Web data that we call *Semantic Data Kiosks*, each constructed by NYAYA by importing an RDF dataset (see the bottom of Fig. 1). NYAYA separates the extensional data i.e., the RDF triples defining instances of classes and properties, from those defining intensional constraints (e.g., sub properties and sub classes). The intensional constraints (top of Fig. 1) are reformulated as a set of Datalog$^{\pm}$ [2] rules i.e., tuple-generating dependencies (TGDs), equality-generating dependencies (EGDs) and negative constraints (NCs) over a set of ontological predicates (e.g., classes and properties), in order to enable efficient reasoning and querying. The instances of the ontological predicates are mapped to their counterparts in the storage by means of a set of automatically generated non-recursive Datalog rules (essentially views) called *storage program*. This allows the instance-level organization of the storage to be described independently of the ontological constraints; as a consequence, the reasoning and querying algorithms remain efficient regardless of any modifications to the logical organization of the storage. A collection of kiosks constitutes what we call a *Semantic Data Market*, where the content of the kiosks can be accessed in a uniform way. An important aspect is the possibility to query the extensional data of a kiosk using ontological constraints that are different from those natively specified by the kiosk or to extend them with further, user-defined Datalog$^{\pm}$ constraints.

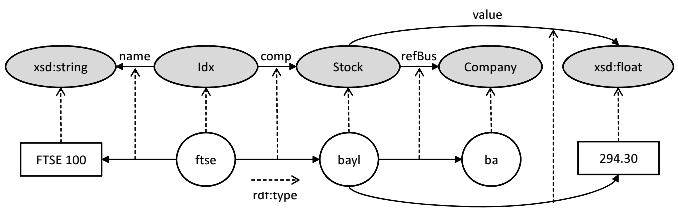
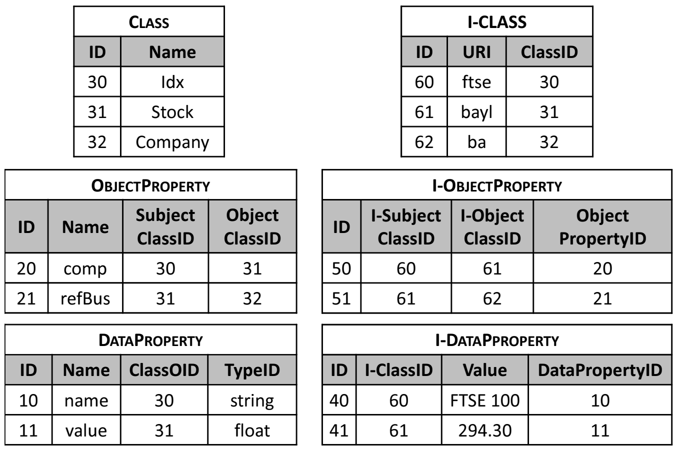


Fig. 2. A running example.

B. Representation of data and metadata

Following an approach for the uniform management of heterogeneous data models [4], extensional data and constraints are represented in NYAYA using a *metamodel* made of a set of generic *constructs*, that are used to represents the primitives of a concrete Semantic-Web language (such as RDF(S) and OWL) for describing the domain of interest. This approach has two main advantages: on the one hand, it provides a framework in which different Semantic-Web languages can be handled in a uniform way and, on the other hand, it allows the definition of *multi-language* reasoning capabilities.

As an example, the RDF(S) stock-exchange scenario in Fig. 2 is represented by the relational instance of Fig. 3 (some of the table fields are omitted for the sake of readability).



CLASS	
ID	Name
30	Idx
31	Stock
32	Company

I-CLASS		
ID	URI	ClassID
60	ftse	30
61	bayl	31
62	ba	32

OBJECTPROPERTY			
ID	Name	Subject ClassID	Object ClassID
20	comp	30	31
21	refBus	31	32

I-OBJECTPROPERTY			
ID	I-Subject ClassID	I-Object ClassID	Object PropertyID
50	60	61	20
51	61	62	21

DATAPROPERTY			
ID	Name	ClassOID	TypeID
10	name	30	string
11	value	31	float

I-DATAPPROPERTY			
ID	I-ClassID	Value	DataPropertyID
40	60	FTSE 100	10
41	61	294.30	11

Fig. 3. Representation of the running example in the metamodel used for data storage.

The approach is easily extensible. For instance, generalization hierarchies can be added to $\mathcal{M}$ by introducing constructs that capture the notions of subclass and sub-property.

C. Querying Semantic Data Kiosks

A semantic data kiosk $\mathcal{K}$ ia a triple $(\Sigma_{\mathcal{O}}, \Sigma_{\mathcal{S}}, D)$ where $\Sigma_{\mathcal{O}}$ is a set of ontological constraints, $\Sigma_{\mathcal{S}}$ is a storage program and D is the database containing the extensional data of the kiosk. In NYAYA, reasoning and querying are reduced to conjunctive-query answering over relational data under first-order reducible (hence FO-reducible) constraints. This reduction allows the delegation of both reasoning and querying to the underlying relational database engine thus obtaining the same efficiency as for traditional database queries. The semantics of query answering adopted in NYAYA is that of *certain answers* [5].

Users operate in the semantic data market using a high level front-end that generates a conjunctive query (CQ). Let Q be a user CQ over a semantic data kiosk $\mathcal{K} = (\Sigma_{\mathcal{O}}, \Sigma_{\mathcal{S}}, D)$. Proceeding from the top of Fig. 1 we proceed as follows:

(i) The user conjunctive query Q is reformulated in terms of the Datalog$^{\pm}$ FO-reducible constraints [6] using the rules in $\Sigma_{\mathcal{O}}$ by applying TGD-Rewrite [7], a backward-chaining resolution algorithm that produces a perfect rewriting $Q_{\mathcal{O}}$ of Q with respect to $\Sigma_{\mathcal{O}}$ which "embeds" the constraints. By virtue of the FO-reducibility, $Q_{\mathcal{O}}$ is a union of conjunctive queries.

(ii) Every $Q^i_{\mathcal{O}} \in Q_{\mathcal{O}}$ is rewritten in terms of the storage tables using the rules in $\Sigma_{\mathcal{S}}$. For each $Q^i_{\mathcal{O}}$, the result is another CQ $Q^i_{\mathcal{S}}$ that is perfect rewriting of $Q^i_{\mathcal{O}}$ with respect to $\Sigma_{\mathcal{S}}$. We have obtained, for each kiosk, a UCQ that takes into account the relationships between the entities occurring in the query and the structures in the persistent storage of each kiosk. The union of all the $Q^i_{\mathcal{S}}$ forms a perfect rewriting $\hat{Q}$ of Q with respect to $\Sigma_{\mathcal{O}} \cup \Sigma_{\mathcal{S}}$. NYAYA implements standard semantic-optimization techniques using key and foreign-key constraints defined by the storage metamodel to reduce the number of atoms and joins in the queries.

(iii) The query $\hat{Q}$ is rewritten in SQL and executed over the underlying DBMS.

III. Demonstration Outline

NYAYA has been implemented in Java and its inference component has been built by extending and optimizing the IRIS Datalog engine.[3] Oracle 11g R2 has been used for the persistent storage, exploiting the native referential partitioning technique of the DBMS. In particular, tables storing instance-level predicates are horizontally partitioned with respect to the foreign keys that refer to the tables storing schema-level predicates. The actual demonstration will show the reasoning and querying capabilities of NYAYA w.r.t. different ontologies and the corresponding datasets. The user can exploit a feasible graph-oriented Web front-end to define and submit queries. Meaningful use cases are reported.

LUBM - UOBM. We start with some ontological conjunctive querying examples using two widely-accepted benchmarks: LUBM[4] and UOBM [8]. For both, we consider an instance of 12.8 million triples. Since the expressiveness of the meta-data in these data sets exceeds the one of linear Datalog$^{\pm}$ (i.e., OWL-Lite for LUBM and OWL-DL for UOBM), we will use manually produced FO-reducible approximation of these ontologies. These approximations will be available on the NYAYA Web site[5] to the audience for live inspection of the set of constraints.

TABLE I
User-defined constraints for the Wikipedia3 kiosk.

article(X) $\rightarrow \exists Y$ modified(X,Y).
article(X) $\rightarrow \exists Y$ contributor(X,Y).
article(X) $\rightarrow \exists Y$ title(X,Y).
category(X) $\rightarrow \exists Y$ modified(X,Y).
category(X) $\rightarrow \exists Y$ contributor(X,Y).
category(X) $\rightarrow \exists Y$ title(X,Y).
redirectsTo(X,Y) $\rightarrow$ article(X), article(Y).
narrower(X,Y) $\rightarrow$ category(X), category(Y).
subject(X,Y) $\rightarrow$ article(X), category(Y).
category(X) $\rightarrow \exists Y$ article(Y), subject(Y,X).

SWETOdblp. We can then proceed to some querying tests over SWETODBLP, the DBLP fragment of SWETO (Semantic Web Technology Evaluation Ontology),[6] a large, high quality test ontology used to assess various ontology management tools and test their scalability. SWETO contains about 17 million triples of DBLP entries and we use it to show the optimization capabilities in Nyaya. In particular, we will execute the following query, retrieving all the 2008 doctoral thesis from the Oxford University computer science department and their ISBNs:

$$\begin{aligned} q(X,Y) \quad \leftarrow \quad & \mathit{Publication}(X), \mathit{DoctoralThesis}(X), \\ & \mathit{isbn}(X,Y), \mathit{year}(X, \text{“2008”}), \\ & \mathit{at\_organization}(X, \text{“}\mathit{OUCS}\text{”}). \end{aligned}$$

and we will show how the system optimizes the query by, e.g., removing the atom $\mathit{Publication}(X)$ because it is implied by the atoms $\mathit{DoctoralThesis}(X)$ and $\mathit{isbn}(X,Y)$.

Wikipedia. In the last run we can show how to use NYAYA to add user-defined constraints over an existing kiosk. We use WIKIPEDIA3, a conversion of Wikipedia into RDF[7] containing around 47 million triples. The expressiveness of its meta-data completely falls within that one of Datalog$^{\pm}$ so no adaptations were made to the ontology. The data set from WIKIPEDIA3 involves two main classes: Article and Category (which is a subClass of Article), four data properties (text, dc:title, dc:modified, dc:contributor) and seven object properties (skos:subject, skos:narrower, link with subProperties internalLink, externalLink, interwikiLink, and redirectsTo). No other constraints are explicitly defined in the ontology. For this reason we use Nyaya to add the user-defined constraints shown in the upper part of Table I. In particular, we force each article and category to have a title, contributor and last-modified date. Then, we define range and domains for the properties redirectsTo, subject and narrower. Finally we force each category to have at least one article referencing

[3]http://www.iris-reasoner.org/
[4]http://swat.cse.lehigh.edu/projects/lubm/
[5]http://www.nyaya.eu/.
[6]http://lsdis.cs.uga.edu/proj/sweto/.
[7]http://labs.systemone.net/wikipedia3.

TABLE II

STORAGE PROGRAM FOR THE WIKIPEDIA3 KIOSK.

redirectsTo(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'redirectsTo',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
link(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'link',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
internalLink(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'internalLink',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
interwikiLink(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'interwikiLink',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
subject(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'subject',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
narrower(X,Y)	←	I-ObjectProperty(Z_0,Z_1,Z_2,Z_3), ObjectProperty(Z_3,'narrower',Z_4,Z_5), I-Class(Z_1,X,Z_4), I-Class(Z_2,Y,Z_5).
modified(X,Y)	←	I-DataProperty(Z_0,Z_1,Y,Z_2), DataProperty(Z_2,'modified',Z_3,Z_4), I-Class(Z_1,X,Z_3).
contributor(X,Y)	←	I-DataProperty(Z_0,Z_1,Y,Z_2), DataProperty(Z_2,'contributor',Z_3,Z_4), I-Class(Z_1,X,Z_3).
title(X,Y)	←	I-DataProperty(Z_0,Z_1,Y,Z_2), DataProperty(Z_2,'title',Z_3,Z_4), I-Class(Z_1,X,Z_3).
category(X)	←	I-Class(Z_0,X,Z_1), Class(Z_1,'Category').
article(X)	←	I-Class(Z_0,X,Z_1), Class(Z_1,'Article').

them. Table II shows the storage program for WIKIPEDIA3. The demo proceeds by executing the following query:

$$q(X, Y) \leftarrow article(X), subject(X, Y)\\ broader(\text{“}Canadian\_Computer\_Scientists\text{”}, Y),\\ category(Y).$$

retrieving all the articles with the corresponding categories whose subject is (a specialization of) the $Canadian\_Computer\_Scientists$ category. NYAYA will use the user-defined constraints to perform reasoning and to optimize the query. NYAYA first eliminates atoms $article(X)$ and $category(Y)$ because they are implied by atom $subject(X, Y)$ under the constraints defined in Table I. Then, NYAYA rewrites atom $broader(\text{“}Canadian\_computer\_scientists\text{”}, Y)$ with atom $narrower(Y, \text{“}Canadian\_computer\_scientists\text{”})$ obtaining, as a result, the following two queries:

$$q(X, Y) \leftarrow subject(X, Y)\\ broader(\text{“}Canadian\_computer\_scientists\text{”}, Y).$$

$$q(X, Y) \leftarrow subject(X, Y)\\ narrower(Y, \text{“}Canadian\_computer\_scientists\text{”}).$$

The above queries are then rewritten into queries over the storage metamodel by using the mapping defined in the storage program. However, since the first query contains an atom with predicate broader which is only a “virtual” predicate introduced by an ontological constraint, only the second query leads to an actual rewriting. In addition, since in the metamodel the IDs of the instance-level tuples are determined by a known hash function, it is also possible to substitute the references between the instance-level tuples and the corresponding schema-level resource (e.g., a class or a property). Assuming that the IDs of the object properties narrower and subject were “20” and “30” respectively, and those of the classes Article and Category were “40” and “50” respectively, the output of the process would be the query:

$$q(X, Y) \leftarrow I\text{-}ObjectProperty(Z_0, Z_1, Z_2, \text{“30”})\\ I\text{-}Class(Z_1, X, \text{“40”}), I\text{-}Class(Z_2, Y, \text{“50”}),\\ I\text{-}ObjectProperty(Z_5, Z_2, Z_6, \text{“20”}),\\ I\text{-}Class(Z_6, \text{“}Canadian\_computer\_scientists\text{”}, \text{“50”}).$$

The actual SQL query corresponding to the above Datalog query and executed by NYAYA is:

```
select  C1.URI, C2.URI
from    I-ObjectProperty OP1, I-ObjectProperty OP2,
        I-Class C1, I-Class C2, I-Class C3
where   C3.URI = 'Canadian_Computer_Scientists' AND
        OP1.I-SubjectClassID = C1.ID AND
        OP1.I-ObjectClassID = C2.ID AND
        OP2.I-SubjectClassID = C2.ID AND
        OP2.I-ObjectClassID = C3.ID AND
        OP2.I-ObjectPropertyID = '20' AND
        OP1.I-ObjectPropertyID = '30' AND
        C1.ClassID = '40' AND
        C2.ClassID = '50' AND
        C3.ClassID = '50'.
```

IV. CONCLUSION

The demonstration showcases NYAYA's ability to efficiently manage different repositories of semantic data in a uniform way. NYAYA provides advanced reasoning capabilities over collections of semantic data sets and well-known benchmarks taking advantage, when available, of meta-data expressed in diverse ontology languages. In addition, it can: (i) query efficiently the sources with a uniform interface, (ii) update rapidly both data and meta-data, and (iii) easily add new sources.

Acknowledgements. This research has been partially funded by the European Commission, Programme IDEASERC, Project 227977-SMScom. Giorgio Orsi has been supported by the Oxford Martin School - Institute for the Future of Computing.

REFERENCES

[1] R. de Virgilio, F. Giunchiglia, and L. Tanca, Eds., *Semantic Web Information Management - A Model-Based Perspective.* Springer Verlag, 2010.

[2] A. Calì, G. Gottlob, and T. Lukasiewicz, “A general datalog-based framework for tractable query answering over ontologies,” in *PODS*, 2009, pp. 77–86.

[3] R. D. Virgilio, G. Orsi, L. Tanca, and R. Torlone, “Semantic data markets: a flexible environment for knowledge management,” in *CIKM*, 2011.

[4] P. Atzeni, P. Cappellari, R. Torlone, P. Bernstein, and G. Gianforme, “Model-independent schema translation,” *VLDB J.*, vol. 17, no. 6, pp. 1347–1370, 2008.

[5] S. Abiteboul and O. M. Duschka, “Complexity of answering queries using materialized views,” in *PODS*, 1998, pp. 254–263.

[6] K. Wang and L. Yuan, “First-order logic characterization of program properties,” *IEEE TKDE*, vol. 6, no. 4, pp. 518–533, 1994.

[7] G. Gottlob, G. Orsi, and A. Pieris, “Ontological queries: Rewriting and optimization,” in *ICDE*, 2011.

[8] L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, and S. Liu, “Towards a complete OWL ontology benchmark,” in *ESWC*, 2006, pp. 125–139.