ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Ontology Evolution

Arivarasan S

Assistant Professor, Department Of Information Technology, Tagore Engineering College, Chennai

Abstract - The evolution of ontologies is an undisputed necessity in ontology-based data integration. Yet, few research efforts have focused on addressing the need to reflect the evolution of ontologies used as global schemata onto the underlying data integration systems. In most of these approaches, when ontologies change their relations with the data sources, i.e., the mappings, are recreated manually, a process which is known to be error-prone and time-consuming. In this paper, we provide a solution that allows query answering in data integration systems under evolving ontologies without mapping redefinition. This is achieved by rewriting queries among ontology versions and then forwarding them to the underlying data integration systems to be answered. To this purpose, initially, we automatically detect and describe the changes among ontology versions using a high level language of changes. Those changes are interpreted as sound global-as-view (GAV) mappings, and they are used in order to produce equivalent rewritings among ontology versions. Whenever equivalent rewritings cannot be produced we a) guide query redefinition or b) provide the best ''over-approximations'', i.e., the minimally-containing and minimally-generalized rewritings. We prove that our approach imposes only a small overhead over traditional query rewriting algorithms and it is modular and scalable. Finally, we show that it can greatly reduce human effort spent since continuous mapping redefinition is no longer necessary.

I. INTRODUCTION

The development of new scientific techniques and the emergence of new high throughput tools have led to a new information revolution. The nature and the amount of information now avail-able open directions of research that were once in the realm of science fiction. During this information revolution the data gathering capabilities have greatly surpassed the data analysis techniques, making the task to fully analyze the data at the speed at which it is collected a challenge. The amount, diversity, and heterogeneity of that information have led to the adoption of data integration systems in order to manage it and further process it. However, the integration of these disparate data sources raises several semantic heterogeneity problems.

By accepting ontology as a point of common reference, naming conflicts are eliminated and semantic conflicts are reduced. Ontologies are used to identify and resolve heterogeneity problems, usually at schema level, as a means for establishing an explicit formal vocabulary to share. During the past years, ontologies have been used as global schemata in database integration [1], obtaining promising results, for example in the fields of biomedicine and bioinformatics [2,3]. When using ontologies to integrate data, one is required to produce mappings, to link similar concepts or relationships from the ontologism to the sources by way of equivalence. This is the *mapping definition process* [4] and the output of this task is the *mapping*, i.e., a collection of mappings rules. In practice, this process is done manually with the help of graphical user interfaces and it is a *time-consuming*, *labor-intensive* and *error-prone* activity [5].

Despite the great amount of work done in ontology-based data integration, an important problem that most of the systems tend to ignore is that ontologies are living artifacts and subject to change [4]. Due to the rapid development of research, ontologies are frequently changed to depict the new knowledge that is acquired. The problem that occurs is the following: when ontologies change, the mappings may become invalid and should somehow be updated or adapted.

In this paper, we address the problem of data integration for evolving RDF/S ontologies that are used as global schemata. We address the problem for a core subset of SPARQL queries that correspond to a union of conjunctive queries. We argue that ontology change should be considered when designing ontology-based data integration systems. A typical solution would be to regenerate the mappings and then regenerate the dependent artifacts each time the ontology evolves. However, as this evolution might happen too often, the overhead of redefining the mappings each time is significant. The approach, to recreate mappings from scratch each time the ontology evolves, is widely recognized to be problematic [5–7], and instead, previously captured information should be reused. However, all current approaches that try to do that suffer from several drawbacks and are inefficient [8,9] in handling ontology evolution in a state of the art ontology-based data integration system. The lack of an ideal approach leads us to propose a new mechanism that builds on the latest theoretical advances on the areas of ontology change [10] and query rewriting [11,12] and incorporates and handles ontology evolution efficiently and effectively. More specifically:

• We present the architecture of a data integration system,

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

named *Evolving Data Integration* system, that allows the evolution of the ontology used as global schema. *Query answering* in our system proceeds in two phases: (a) *query rewriting* from the latest to the earlier ontology versions and

- The query processing in the first step consists of: (i) *queryexpansion* that considers constraints coming from the ontology, and (ii) *valid query rewriting* that uses the changes between two ontology versions to produce rewritings among them.
- In order to identify the changes between the ontology versions we adopt a high-level language of changes. We show that the proposed language possesses salient properties such as *uniqueness*, *inversibility* and *composability*. *Uniqueness* is a pre-requisite for the solution described in this paper, where the other two properties are nice to have, but they are not necessary for our solution. The sequence of changes between the latest and the other ontology versions is produced automatically at setup time and then those changes are translated into logical GAV mappings. This translation enables query rewriting by unfolding. Moreover, the *inversibility* is exploited to rewrite queries from past ontology versions to the current, and vice versa, and *composability* to avoid the reconstruction of all sequences of changes among the latest and all previous ontology versions.
- Despite the fact that query rewriting always terminates, the rewritten queries issued, using past ontology versions, might fail. We show that this problem is not inhibiting in our algorithms but a consequence of information unavailability among ontology versions. To tackle this problem, we propose two solutions: (a) either to provide best ''over-approximations'' by means of *minimally-containing* and *minimally-generalized* queries, or (b) to provide insights for the failure by means of *affecting change operations*, thus driving query redefinition.
- We show that our method is sound and complete and does not impose a significant overhead. Finally, we present our experimental analysis using two real-world ontologies. Experiments performed show the feasibility of our approach and the considerable advantages gained.

Such a mechanism that provides rewritings among data integration systems that use different ontology versions as globalschemata is *flexible*, *modular* and *scalable*. It can be used on top of any data integration system—independently of the family of the mappings that each specific data integration system uses to define mappings between one ontology version and the local schemata (GAV, LAV, GLAV [13]). New mappings or ontology versions can be easily and independently introduced without affecting other mappings or other ontology versions. Our engine takes the responsibility of assembling a coherent view of the world out of each specific setting.

This paper is an extended and revised version of a previously published conference paper [14] whereas the implemented system was demonstrated in [15]. However, only the basic ideas were described in [1], without a detailed analysis of the theoretical

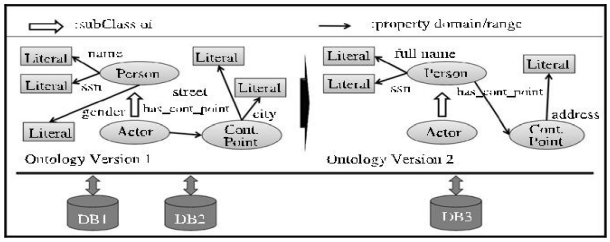

Fig. 1. The motivating example of an evolving ontology.

II. MOTIVATING EXAMPLE AND RELATED WORK

Consider the example RDF/S ontology shown on the left of Fig. 1. This ontology is used as a point of common reference,describing persons and their contact points (''*Cont.Point*''). We also have two relational databases DB1 and DB2 mapped to that version of the ontology. Assume now that the ontology designer decides to move the domain of the ''*has_cont_point*'' property from the class ''*Actor*'' to the class ''*Person*'', and to delete the property ''*gender*''. Moreover, the ''street'' and the ''*city*'' properties are merged to the ''*address*'' property. Merging is a concatenation with some special character like comma between the words. Furthermore, the ''*name*'' property is renamed to ''*fullname*'' as shown on the right of Fig. 1. Then, one new database DB3 is mapped to the new versionof the ontology leading to two data integration systems that work independently. In such a setting we would like to issue queries formulated using any ontology version available. Moreover, we would like to retrieve answers from all underlying databases.

Several approaches have been proposed so far to tackle similar problems. For example, for XML databases there

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

have been several approaches that try to preserve mapping information under changes [16] or propose guidelines for XML schema evolution in order to maintain the mapping information [17]. Moreover, augmented schemata were introduced in [18] to enable query answering over multiple schemata in a data warehouse, whereas other approaches change the underlying database systems to store versioning and temporal information such as [19–24]. However, our system differs from all the above in terms of both goals and techniques.

Other works focus on the problem of updating RDF/S [25–27] or OWL–DL [28] knowledge bases. These works mostly try to determine the effects and side-effects of elementary or complex change operations and to characterize the different class of updates with a well-defined semantics. In our work, however, we do not deal with the effects of the change operations on the ontology butwe assume that the ontology versions are directly given. Moreover, we use a language with well-defined semantics in order to identify *a-posteriori* the changes that have happened to the ontology.

The most relevant approaches that could be employed for re-solving the problem of data integration with evolving ontologies is mapping adaptation [5] and mapping composition and inver-sion [9].

In mapping adaptation [5] the main idea is that schemata often evolve in small, primitive steps; after each step the schema mappings can be incrementally adapted by applying local modifications. However, this approach is integration system-dependent, and is not specified in which way the list of changes might be discovered when two schema versions are directly provided. But even when such a list of changes can be obtained, applying the incremental algorithm for each change and for each mapping in this potentially long list will be highly inefficient. Another problem is that multiple lists of changes (by introducing redundant additions/deletions for example) may have the same effect of evolving the old schema into a new one [6]. Finally, there is no guarantee that after repeatedly applying the algorithm, the semantics of the resulting mappings will be the desired ones. This happens because complex evolution might happen, that cannot be modeled with simple additions and deletions, and dependencies might be lost. In order to tackle these problems we use a more expressive language of changes that leads to unique sequence of changes between two ontology versions with reduced size compared to the long list of low-level operators. Moreover, the initial semantics of the provided mappings are maintained since we do not change the mappings but instead we rewrite the queries.

A more general formalization of the mapping adaptation problem is through mapping composition and inversion [6, 9]. The approach would be to describe ontology evolution asmappings and to employ mapping composition/inversion to derive the adapted mappings. However, mapping composition proved to be a difficult problem and mapping inversions a more difficult one. In [29] it was shown that no first-order language is closed under composition and second-order mappings should be used instead, whereas the identification of a language closed under both inversion and composition is still an open problem [9]. An exact inverse may not exist and several notions of ''approximations'' of inverses have been lately developed such as quasi-inverses [30], maximum recoveries [31], chase-inverses [9] etc. A recent system that tries to build on composition and quasi-inverse schema mappings is PRISM [32]. However, the sequence of changes among two versions is not unique and disambiguation is needed in several places by domain experts. Moreover, the composed mappings might be too difficult for domain experts to grasp and understand (they are second-order mappings). Our approach avoids the constant involvement of domain experts since continuous mapping redefinition is no longer necessary. The changes among the ontologies are produced automatically. Moreover, instead of composing all mappings each time, in our case they are kept intact in order to be verified and updated by domain experts.

To the best of our knowledge no system today is capable of retrieving information mapped with different ontology versions.

III. EVOLVING DATA INTEGRATION

We conceive an *Evolving Data Integration* system as a collection of data integration systems, each one of them using a different ontology version as global schema. Therefore, we extend the traditional formalism from [13] and define an *EvolvingDataIntegration*system as:

Definition 3.1(*Evolving Data Integration System*)**.**An Evolving
Data Integration system I is a tuple of the form $((O_1, S_1, M_1), \ldots, (O_m, S_m, M_m))$where:

- $O_i$is a version of the ontology used as global schema,
- $S_i$is a set of local sources and

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

• $M_i$is the mapping between$S_i$andO_i($1 \leq i \leq m$).

Next we discuss how the specific components are specialized in the context of an *Evolving Data Integration* system.

A. Global and local schemata

Considering O_i we restrict ourselves to *valid RDF/S knowledgebases*, as most of the Semantic Web Schemas (85.45%) areexpressed in RDF/S [33].

The representation of knowledge in RDF [34] is based on triples of the form (*subject predicate object*). Assuming two disjoint and infinite sets *U*, *L*, denoting the URIs and literals respectively, $T = U \times U \times (U \cup L)$is the set of all triples. An RDF Graph *V* is definedas a set of triples, i.e., $V \subseteq T$. In this paper, we ignore unnamed resources, also called *blank nodes*. RDFS [35] introduces some built-in classes (class, property) which are used to determine the *type* of each resource. The typing mechanism allows us to concentrate on nodes of RDF graphs, rather than triples, which is closer to ontology curators' perception and useful for defining intuitive high-level changes.

RDFS provides also *inference semantics*, which is of two types, namely *structural inference* (provided mainly by the transitivity of subsumption relations) and *type inference* (provided by the typing system, e.g., if *p* is a property, the triple (*p*, *type*, *property*) can be inferred). The RDF Graph containing all triples that are either explicit or can be inferred from explicit triples in an RDF Graph *V* (using *both* types of inference), is called the *closure* of *V* and isdenoted by $Cl(V)$. An *RDF/S Knowledge Base (RDF/S KB) V* is an RDF Graph which is closed with respect to *type inference*, i.e., it contains all the triples that can be inferred from *V* using type inference. Moreover, we assume that the RDF/S Knowledge Bases are *valid*. The notion of validity has been described in various fragments of the RDFS language. The validity constraints that we consider in this work concern the *type uniqueness*, i.e., that each resource has a unique type, the *acyclicity* of the *subClassOf* and *subPropertyOf* relations and that the subject and object of the instance of some property should be correctly classified under the domain and range of the property, respectively. For a full list of the validity constraints we adopt see [36]. Those constraints are enforced in order to enable unique and non-ambiguous detection of the changes among the ontology versions.

Moreover, we consider as underlying data integration systems, those that integrate relational databases using an ontology as global schema. (For our experiments we used the MASTRO [12] data integration system which relates ontologies to relational schemata by global-as-view mappings.) We have to note here that the mapping from relational schemas to RDF/S is lossy since usually constraints and dependencies from the relational database such as keys and foreign keys cannot be captured. We choose such systems as the majority of information currently available is still stored on relational databases [37].

B. Modeling ontology evolution

For modeling ontology evolution we use a language of changes that describes how an ontology version was derived from another ontology version. In its simplest form, a language of changes consists of only two *low-level* operations, *Add*(*x*) and *Delete*(*x*), which determine individual constructs (e.g., triples) that were added or deleted [38,39]. Such a language is called *a low-levellanguage of changes*. However, a significant number of recentworks [10,39,40] imply that *high-level* change operations should be employed instead, which describe more complex updates, as for instance the insertion of an entire subsumption hierarchy (they group individual additions and deletions).

A high-level language is preferable than a low-level one [41], as it is more *intuitive*, *concise*, *closer to the intentions* of the ontology editors and *captures more accurately* the semantics of change. For example the high-level change operation *''Rename_Property(fullname, name)'' is more informative the following set of low level operations ''Delete(fullname, type, property), Add(name, type, property)''*. As we shall see later on, a high-level language isbeneficial for our problem for two reasons: First, because the produced change log has a smaller size and most important because such a language yields logs that contain a smaller number of individual low-level deletions (which are non-information preserving) and this affects the effectiveness of our rewriting.

IV. SEMANTICS OF AN EVOLVING DATA INTEGRATION SYSTEM

Now we will define semantics for an *Evol*ving *Data Integration* system *I*. Our approach is similar to [13] and is sketched in Fig. 3. We start by considering a *local database* for each (O_i, S_i, M_i), i.e., a database D_i that conforms to the *local sources of* S_i. For example D_1 is a local database for (O_1, S_1, M_1) that conforms to the local sources S_{11}, $S_{12}$and S_{13}.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Now, based on D_i, we shall specify the information content of the global schema O_i. We call a database for O_i a *global database*. However, since those global databases might be many, we are interested in the *legal global databases*.

Definition 4.1(*Legal Global Database*)**.**A global database$G_i$for(O_i, S_i, M_i) is said to be legal with respect to D_i, if

- $G_i$satisfies all subClass/subProperty constraints ofO_i
- $G_i$satisfies the mapping$M_i$with respect to D_i.

The notion of G_i*satisfying* the mapping M_i, with respect to D_i, is defined as it is commonly done in traditional data integration
systems (see [13] for more details). It depends on the different assumptions that can be adopted for interpreting the tuples that $D_i$assigns to relations in local sources with respect to tuplesthat actually satisfy (O_i, S_i, M_i). Since such systems have been extensively studied in the literature we abstract from the internal details and focus on the fact that for each (O_i, S_i, M_i) of our system we can obtain *several legal global databases* G_i.

Now, we can repeat the same process, i.e., to consider the legal global databases as sources and a database D which we will simply call the *global database*, the database that conforms to them. Now we can similarly define the *total databases* (databases for O_m) and the *legal total databases*. We use the term ''*total*'' only todifferentiate it from a *global* database, since we will extensively use it from now on.

Definition 4.2(*Legal Total Database*)**.**A total database$T$for$I$issaidto be legal with respect to D, if

- T satisfies all subClass/subProperty constraints ofO_m.
- T satisfies the evolution log $E = {}^{m}_{1}{}^{-i}E^{Om,Oi}$w.r.t.D.

Now we specify the notion of T satisfying the evolution log E with respect to D. In order to exploit the strength of the logical languages towards query reformulation, we convert our change operations into logical GAV mappings. So, when we refer to the notion of T satisfying E, we mean T satisfying the GAV mappings produced from E. The GAV mappings for some of the change operations used in this paper can be found in Fig. 2. In relational databases a GAV mapping associates a table from the target schema to a query over the source schemata. So, in our case a GAV mapping associates to a class/property g in T a query q_G over the other ontology versions $G_1, \ldots, G_m$, i.e., $g^T \rightarrow q_G$.

Definition 4.3**.**A database$T$satisfies the mappings$g \rightarrow q_G$withrespect to D if $g^T \supseteq q^D{}_G$ where $q^D{}_G$ is the result of evaluating the query q_G over D.

For example, the sequence of the GAV mappings that corre-sponds to our sequence of changes is:

$\boldsymbol{mu_1}$:$\forall x,y,fullname(x,y) \rightarrow name(x,y)$

$\boldsymbol{mu_2}$$x,y,address(x,y)$ $y_1, y_2, street(x, y_1)$ $city(x, y_2)$
conca
$t^{\forall}$ $(y, \{y_1, y_2\})$ $\rightarrow \exists$

$\boldsymbol{mu_3}$:$\forall x,has\_cont\_point(Person,x) \rightarrow has\_cont\_poin$
$t(Actor, x)$.

For u_4 there is no GAV mapping constructed since we do not know where to map the deleted element. Now it becomes obvious that the more individual additions and deletions in our language of changes, the more change operations will not have corresponding GAV mappings. This is why languages with ''high-level'' changes, i.e. changes that group together several individual additions and deletions are preferable.

By the careful separation between the *legal total database* T and the *legal global databases* G_i we have achieved the modular design of our *Evolving Data Integration* system and the separation between the traditional data integration semantics and the additions we have imposed in order to enable ontology evolution. Thus, our approach can be applied on top of any existing data integration system to enable ontology evolution. Moreover, we have managed to model ontology evolution in data integration as query answering over materialized views.

V. QUERY PROCESSING

Queries to I are posed in terms of the global schema O_m. For querying, we adopt a core subset of the SPARQL language corresponding to a union of conjunctive queries [43].

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

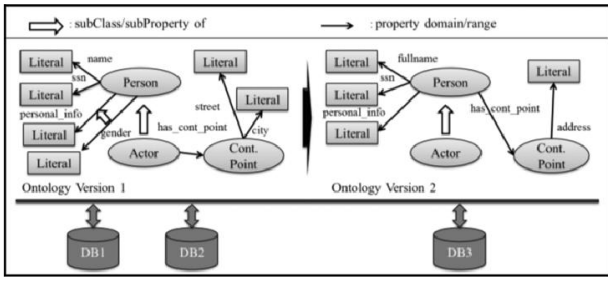

Fig. 2. An alternative ontology version 1.

In order to avoid ambiguities in parsing, we present the syntax of SPARQL graph patterns in a more traditional algebraic way,
using the binary operators *UNION* (**U**), *AND* ( ), *OPT*, and *FILTER* according to [44]. Assuming the existence of an infinite set of variables Var disjoint from *U*, *L*, a SPARQL graph pattern expression is defined recursively as follows:

- A tuple from $(U\ L\ \text{Var})x(L\ \text{Var})x(U\ L\ \text{Var})$ is a *triple pattern).*
  graph pattern *(a*
  If P_1 and P_2 are graph patterns, then expressions
  $(P_1,P2)$,
  $(P_1 OPT\ P_2)$ and $(P_1 \mathbf{U} P_2)$ are graph
- patterns.
- If *P* is a graph pattern and *R* is a SPARQL *built-in condition*, then the expression (*P FILTER R*) is a graph pattern.

A SPARQL built-in condition is constructed using elements of the
set (*U L*Var) and constants, logical connectives, inequality symbols, the equality symbol etc. (See [43] for a complete list.)

In this paper we adopt a streamlined version of the core fragment of SPARQL as presented in [44] with precise syntax and semantics. Moreover, we restrict even more the specific fragment of SPARQL since we do not consider *OPT* and *FILTER* operators which we leave for future work. The remaining SPARQL fragment corresponds to a union of conjunctive queries [44] (this will not hold if we allow *OPT* and *FILTER* operations). Moreover, the application of the *solution modifiers* and the *output* is performed after the evaluation of the query. So, without loss of generality we will not present them in this paper. Continuing our example, assume that we would like to know the ''*ssn*'' and ''*fullname*'' of all persons stored on our DBs and their corresponding ''*address*''. The SPARQL query, formulated using the latter version of our example ontology is:

q_1:*select*?*SSN*?*NAME*?*ADDRESS*
where{?*X type Person*.
?*Xssn* ?*SSN*.
?*Xfullname* ?*NAME*.
?*Xhas_cont_point*
?*Y* . ?*Y type*
Cont.Point. ?*Y*
address
?*ADDRESS*}

Using the semantics from [44] the algebraic representation of q_1 is equivalent to:

q_1 π?*SSN*, ?*NAME*, ?*ADDRESS*(
(?*X*, *type*, *Person*)
(?*X*, *ssn*, ?*SSN*)
(?*X*, *fullname*, ?*NAME*)
(?*X*, *has_cont_point*, ?*Y*)
(?Y,type,cont Point (?Y,address
?ADDRESS))

Now we define what constitutes an answer to a query over O_m. We will adopt the notion of *certain answers* [11,13].

VI. IMPLEMENTATION AND EVOLUTION

The approach described in this paper was implemented on our *exelixis*[3]platform [15]. We developed the*exelixis*platform as a webpage using PHP/JQuery/HTML for the presentation and Java/PHP for implementing the algorithms. The interface is shown on Fig. 11. Using our platform the user is able to load and visualize one version
of an RDF ontology. The visualization is provided either through the jOWL[4] API or the OWLSight[5] plug-in, or the Starlion[6] tool. Then, the user is able to search for a class or property, to visualize the corresponding description and to explore the hierarchy of the ontology. Moreover, the user can construct a SPARQL query which is issued to the system. The system expands the query using the QuOnto engine and then it computes the valid rewriting over the expanded query. Then, the query is forwarded to the underlying data integration systems, where it is answered. The results are unioned and presented to the user.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

A. Evaluation setup

In order to evaluate our system we used a workstation running Windows 7 with an Intel Core 2 Duo processor at 3.0 GHz, and 4 GB memory.

Moreover, to test our system we used two ontologies: One medium-sized ontology (CIDOC-CRM), from the cultural domain which is rarely changed and one large-size ontology (Gene Ontology) from the bioinformatics domain which is heavily updated daily. Each one of those ontologies was used as a global schema in order to query the data mapped to them (to one of their versions).
The detected change log that was produced identified 711 total changes.

Gene Ontology[8] (GO) on the other hand, is composed of about 28 000 classes. We have to note that the file containing the Gene ontology is over 100 MB and most of the ontology editors fail to load the entire file. Moreover, we used the materialized version of GO without blank nodes. This restriction is enforced by the change detection algorithm we used, which does not deal with blank nodes. GO is updated on a daily basis and for our experiments we used 4 versions dated from 16.12.2008 to 26.05.2009. The change log that was produced contained 3482 changes.

The target of our evaluation was to demonstrate the impact of our system and to show that query rewriting between ontology versions can be achieved in timely manner. To do that, we evaluated initially query rewriting between ontology versions. However, after rewriting queries between ontology versions, queries are forwarded to the underlying data integration systems in order to be answered. Since query evaluation affects user experience as well, we measured the time of the underlying data integration system (MASTRO) to evaluate the forwarded query. We have to note that our purpose here was to evaluate the feasibility of our solution and not to do a thorough evaluation[9] of the MASTRO data integration system which is not our contribution.

The evaluation we performed was based on two scenarios. One scenario with synthetic queries automatically constructed and one scenario with real queries captured from related projects and publications. For measuring query answering time, we used 10 relations in the data sources, with 10 rows each and 10 mappings between each ontology version and the local schemata.

We have to note that a comparison with other systems was not possible since there is no known implementation that allows query answering over multiple data integration systems that use different ontology versions.

B. Synthetic evaluation

In the synthetic scenario we automatically generated random queries using CIDOC-CRM v.4.2. We created 20 queries for each one of the following categories: queries with 1, 3, 7 and 20 triple patterns. The synthetic evaluation was performed only for queries formulated using CIDOC-CRM ontology since the queries using the GO ontology ask for instances of only one GO-term (GO ontology is mostly a taxonomy) and rich queries including several properties cannot be produced.

VII. DISCUSSION AND CONCLUSION

Moreover, our approach is more general than trying to use schema composition and inversion to answer queries over multi-ple ontology versions. That is, because in the latter case complex mechanisms should be employed to produce the composition of the mappings between the ontology versions and the underlying data sources, depending also on the type of the mappings used on the underlying data integration systems. In our approach however, the underlying data integration systems are seen as ''black-boxes'' and our algorithms are independent of the type of mappings used between sources and ontology versions. Finally, in our case inver-sion is always possible to be produced whereas this is not guaran-teed in mapping inversion. This is due to the fact that we consider inversion (composition) on a layer on top

where always can find the inverse (composition) of any sequence of changes efficiently.

The potential impact of our approach is witnessed by being able to successfully provide rewritings on the worst case for the 88% of the CIDOC-CRM queries (after 711 change operations) and for the 97% of the GO queries (after 3482 change operations) among ontology versions. On the other hand if our system was not used, only a small percentage of the initial queries would be successful. For most of the queries, query answering is achieved within 5 s using a simple workstation, which also shows the usability and the scalability of our approach. We have to note that in the case of GO we materialized the ontology in order to minimize further the query execution time. The great benefit

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

of our approach is the *simplicity*, *modularity* and the *short deployment* time it requires. It isonly a matter of providing a new ontology version to our system to be able to use it to formulate queries that will be answered by data integration systems independent of the ontology version used.

As future work, several challenging issues need to be further investigated. For example, local schemata may evolve as well, and the ontologies used as global schema may contain inconsistencies. An interesting topic would be to extend our approach for OWL ontologies or to handle the full expressiveness of the SPARQL language. The latter would be a difficult task, since if we allowed *OPT* and *FILTER* operations, we would no longer have the unionof conjunctive queries and the problem in some cases might be undecidable.

REFERENCES

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: the DL-lite approach, in: Reasoning Web, 2009, pp. 255–356.

[2] L. Martin, A. Anguita, V. Maojo, E. Bonsma, A.I.D. Bucur, J. Vrijnsen, M. Brochhausen, C. Cocos, H. Stenzhorn, M. Tsiknakis, M. Doerr, H. Kondylakis, Ontology based integration of distributed and heterogeneous data sources, in: ACGT, HEALTHINF, Funchal, Madeira, Portugal, 2008, pp. 301–306.

[3] M. Hartung, T. Kirsten, E. Rahm, Analyzing the evolution of life science ontologies and mappings, in: DILS, Springer-Verlag, Evry, France, 2008, pp. 11–27.

[4] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, G. Antoniou, Ontology change: classification and survey, Knowl. Eng. Rev. 23 (2008) 117–152.

[5] Y. Velegrakis, J. Miller, L. Popa, Preserving mapping consistency under schema changes, VLDB J. 13 (2004) 274–293.

[6] C. Yu, L. Popa, Semantic adaptation of schema mappings when schemas evolve, in: VLDB, VLDB Endowment, Trondheim, Norway, 2005.

[7] C.A. Curino, H.J. Moon, M. Ham, C. Zaniolo, The PRISM workwench: database schema evolution without tears, in: ICDE, 2009, pp. 1523–1526.

[8] H. Kondylakis, G. Flouris, D. Plexousakis, Ontology& schema evolution in data integration: review and assessment, in: ODBASE, OTM Conferences, VilamouraAlgarve, Portugal, 2009, pp. 932–947.

[9] A. Poggi, D. Lembo, D. Calvanese, G.D. Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. Data Semantics X (2008) 133–173.

[10] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, Madison, Wisconsin, 2002.

[11] H. Kondylakis, D. Plexousakis, Ontology evolution in data integration: query rewriting to the rescue, in: International Conference on Conceptual Modeling, ER, 2011, pp. 393–401.

[12] H. Kondylakis, P. Dimitris, Exelixis: evolving ontology-based data integration system, in: SIGMOD, 2011, pp. 1283–1286.

[13] D. Barbosa, J. Freire, A.O. Mendelzon, Designing information-preserving mapping schemes for XML, in: VLDB, VLDB Endowment, Trondheim, Norway, 2005, pp. 109–120.

[14] A. Poggi, D. Lembo, D. Calvanese, G.D. Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. Data Semantics X (2008) 133–173.

[15] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, Madison, Wisconsin, 2002.

[16] H. Kondylakis, D. Plexousakis, Ontology evolution in data integration: query rewriting to the rescue, in: International Conference on Conceptual Modeling, ER, 2011, pp. 393–401.

[17] H. Kondylakis, P. Dimitris, Exelixis: evolving ontology-based data integration system, in: SIGMOD, 2011, pp. 1283–1286.

[18] D. Barbosa, J. Freire, A.O. Mendelzon, Designing information-preserving mapping schemes for XML, in: VLDB, VLDB Endowment, Trondheim, Norway, 2005, pp. 109–120.

[19] M.M. Moro, S. Malaika, L. Lim, Preserving XML queries during schema evolution, in: WWW, ACM, Banff, Alberta, Canada, 2007, pp. 1341–1342.

[20] S. Rizzi, M. Golfarelli, *X* -time: schema versioning and cross-version querying in data warehouses, in: ICDE, 2007, pp. 1471–1472.

[21] D.N. Xuan, L. Bellatreche, G. Pierra, A versioning management model for ontology-based data warehouses, in: DaWaK, Springer, Krakow, Poland, 2006

[22] H. Bounif, Schema repository for database schema evolution, in: P. Rachel (Ed.), DEXA, 2006, pp. 647–651.

[23] N. Edelweiss, A.F. Moreira, Temporal and versioning model for schema evolution in object-oriented databases, Data Knowl. Eng. 53 (2005) 99–128.

[24] H.J. Moon, C.A. Curino, C. Zaniolo, Scalable architecture and query optimization for transaction-time DBs with evolving schemas, in: SIGMOD 2010 ACM, Indianapolis, Indiana, USA, 2010, pp. 207–218.

[25] D. Ognyanov, A. Kiryakov, Tracking changes in RDF(S) repositories, in: Pro-ceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Springer-Verlag, 2002, pp. 373–378.

[26] C. Gutierrez, C.A. Hurtado, A.A. Vaisman, The meaning of erasing in RDF under the Katsuno–Mendelzon approach, in: WebDB, 2006.

[27] G. Konstantinidis, G. Flouris, G. Antoniou, V. Christophides, A formal approach for RDF/S ontology evolution, in: Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence, IOS Press, 2008, pp. 70-74.

[28] D. Calvanese, E. Kharlamov, W. Nutt, D. Zheleznyakov, Evolution of DL-lite knowledge bases, in: Proceedings of the 9th International Semantic Web Conference on The Semantic Web—Volume Part I, Springer-Verlag, Shanghai, China, 2010, pp. 112–128.

[29] R. Fagin, P.G. Kolaitis, L. Popa, W.-C. Tan, Composing schema mappings: second-order dependencies to the rescue, ACM Trans. Database Syst. (TODS) 30 (2005) 994–1055.

[30] R. Fagin, P.G. Kolaitis, L. Popa, W.C. Tan, Quasi-inverses of schema mappings, ACM Trans. Database Syst. (TODS) 33 (2008).