

Dynamerger: A Merging Algorithm for Structured Data Integration on the Web

François Scharffe

Digital Enterprise Research Institute

Leopold-Franzens-Universität

Innsbruck, Tyrol, Austria

Email: francois.scharffe@deri.org

http://www.deri.org

Integrating various data sources is a major problem in knowledge management. Integration systems proposed in the last two decades as a solution start showing some limitations for various reasons. First the environment scaled from a few data sources in the same location to an unknown number of possible data sources on the Web. Second the sources to be integrated may change quickly over time. We propose in this article a method allowing for the dynamic integration of data sources on the web. Based on a network of schemas (database schemas, ontologies) related via mappings our algorithm generates a global view over a set of resources. Our approach presents the advantage to minimize the necessary human intervention in order to integrate sources.

Keywords: Ontology Merging, Structured Data Integration, Web

1. Introduction

The integration of heterogeneous data sources has a long research history following the different evolutions of information systems. Different approaches have been studied from whose two general classes can be extracted. Federated systems [1–3] using a global schema and mappings between the source schemas and the global schema, and distributed systems [4,5] where peers are integrated using one-to-one mappings. The second approach, more recent, follows the emergence of the web and its extension towards a machine-processable, structured, semantic web [6]. While the federated approaches were adapted for the integration of a limited number of data sources in a close environment, they have started to show limitations in the open environment the web constitutes. Distributed approaches overcome these limitations by offering a scalable architecture.

There would be no problem if data sources were obeying a common schema. It is obviously not the case. Many different ontologies are describing various domains with possible intersections. The answer to the schema heterogeneity problem came early in the database integration field via the use of *mappings* and is still a hot topic in ontology research, particularly on automatic mapping, see [7,8] for recent surveys. The problem of automatically relating overlapping two ontologies is still open [9].

A potentially huge number of ontologies can be related via mappings. Multiple questions are raised at this level about the scalability of an ontology network or about the way to give a consistent and unified view over the data.

Scalability - In the case where we want to exchange data between a large number of sources each described by their own schema, those need to be related together. Integrated approaches where data sources were related to a hand-made global schema are no longer applicable in a scenario where thousands of schemas have to be related. A more *collaborative* approach is more likely to appear where schemas are related one-to-one or one-to-a-few by diverse entities, and the mapping published. In that perspective the complexity of the mapping network grows in $\Theta(n^2)$ of the number of schemas in it. Such a complexity is obviously not acceptable in an environment like the web.

Redundancy - As an information space integrates more and more previously mapped schemas, redundancy between the mappings begin to appear. The transitivity between mappings should be maximally exploited in order to reduce, or better suppress this redundancy. It only has for an effect to augment the length of paths between schemas in the mapping network, and therefore the number of data transformations and query rewriting steps to achieve proper data integration.

Mediator systems were developed as a way to integrate heterogeneous data source by providing a unified view over the data. They are as already argued not scalable in a wide, open environment. The main evolutions of this environment resides in the distributivity or resource repartition, as well as the increased number of possible integrated resources. Next section presents existing solutions for such an environment.

Inconcistencies - An important question is about solving the inconsistencies created by the alignment of many logical theories. [10] propose a

framework to reason with inconsistent ontologies. Then come problems close to the ones that arisen in database integration, the fundamental question being how to provide an unified view of the information to a particular application or service. The obvious solution is to create a global ontology and relate it to domain specific schemas. This requires first to build a global schema capturing the concepts of the local schemas and second to draw the mappings between each local schemas to the global one. There is an extensive amount of work since 20 years on this topic [1–3,11,12]. The bottleneck has always been the necessary human intervention. It is still a problem as it is a time consuming task. Creating a new global schema and relating it to the local schema each time you want to define or redefine an application is not realistic in the context of the semantic web where services are expected to be dynamically created given some requirements.

Section 3 study the evolution of information systems, showing fundamental differences in the problem shape. We present the algorithms to construct the unified view in Section 3. Finally future work and conclusions are given Section 5.

2. Idea: Providing a Unified View

The classical information integration problem consists in finding a way to relate databases in order to provide a unified view over the data. Solutions emerged proposing the use of *mediators* [13] making the link between local schemas and the global hand-made schema.

The shape of the data integration problem however changed. The rising up of the web makes it nowadays possible to easily publish and access data across the world, resulting in a large number of available data sources. New formalisms and systems to describe and manage structured data arose as well. We can already identify two changes:

- (1) The number of potentially available sources is much larger
- (2) Different formalisms are used to describe data

Ontology mediation means to solve this problem by allowing to create mappings between ontologies. Two different approaches are proposed. The first approach consists in applying database integration research to the ontology work by using global ontologies which are related to domain ontologies or databases. This approach is however not likely as the mappings and the global schema must be designed from scratch for each application.

Also the necessary human intervention considerably limit the dynamics of the process.

The second approach consists in linking ontologies one to one. Issues appear when trying to reason over this network of ontologies as first, inconsistencies appear and second the complexity of the network raises scalability problems. Besides this theoretical limitations, this approach doesn't provide a unified view over the data, which make querying more complex for the user. We believe a better solution to information integration can be brought. In order to get closer to this solution we detail in the following issues arising when trying to integrate data on the web.

From this analysis we conclude the need to somehow fill the gap existing between centralized, "old school" data integration systems paradigm and to adapt it to the distributed, large scale environment of the web. This adaptation require some modifications in the way mappings and global schemas are designed.

We propose to dynamically generate a global ontology out of a set of ontologies related together with mappings. We study the feasibility of automatic generation of this global ontology and show the advantages of having a unified view over the data, reusing the mappings between ontologies, strongly reduce the complexity of the network and therefore augment the scalability of semantic web applications. Generating a global ontology from a set of ontologies is related to ontology merging and database schema integration. However, merging ontologies is always a semi automatic task as the mappings have first to be provided. Our approach concentrates on the automatic generation of the global ontology, *given* the mappings. We need a dynamic process where ontologies can be added or removed from network, thus modifying the global ontology.

We integrate structured data sources designed using various meta-languages. In order to achieve this we provide a way to create mappings in an abstract way from the meta-language used to describe these sources. We use the abstract mapping language defined in [14]. Based on this mappings we are able to automatically generate a unified view over a set of related ontologies.

3. Method: Generating the Unified View

Based on a set of ontologies and mappings between them we are able to generate the unified view automatically. We first define a merge operator

able to merge two ontologies related via a mapping. The process incrementally adding ontologies to the unified view using the merge operator is then described. This process allows for dynamically adding sources. For space reasons we do not detail the algorithms here.

3.1. A merge operator

The basic element of the merge process is a merge operator, taking in input two schemas and a mapping between them, and returning a merged schema. The merged schema G get the result of merge that take as a parameter a source schema O_s , a target schema O_t and a mapping between the two schemas $Map_{s,t}$.

The merge operator has to merge classes, attributes and relations found in ontologies. It must preserve the hierarchy relation and must solve possible conflicts. This operation has also for task to name the nodes of the resulting schemas, this is done by giving a preferred schema whose nodes labels will be kept. Merge takes into account the complex correspondences expressed in the mapping such as one-to-many correspondences or conditions on attributes values.

Merge can at this state be decomposed in three procedures $mergeClasses(root(O_s))$, $mergeAttributes$ and $mergeRelations$ which merge respectively classes, attributes and relations of the given schemas. Merging classes^a) is done by traversing the preferred schema concept tree from the root node using a breadth-first algorithm. This permits the proper creation of the merged schema concept tree in a top-down fashion.

The procedure $getMappingAndMerge(Class c)$ ^b is called in the process of merging classes in order to retrieve the mapping rule corresponding to the class it currently processes and creates a proper merged node. If no corresponding rule is found, the node is simply added as a child of the node in the class hierarchy of the merged schema corresponding to its parent in the source (preferred) schema.

The $getMappingAndMerge(Class c)$ procedure finds rules in which the current class appears and adds a new class to the unified view together with the mappings between the unified view and the two merged schemas (lines 4,5,6). We can note the new node is only attached to the preferred

^aWe did not include the algorithm here for space reasons. We provide it on internet at <http://www.scharffe.fr/pub/swiis2007/algorithm1.jpg>

^bWe did not include the algorithm here for space reasons. We provide it on internet at <http://www.scharffe.fr/pub/swiis2007/algorithm2.jpg>

schema parent class. We deliberately don't specify the semantics of $c \in rule$, meaning that a class is part of a mapping rule, as variations exist that are not considered for the moment. If the processed node appear in no mapping rule it is added at the right place in the class hierarchy together with the corresponding mapping rule.

This merge operator allows the construction of the merging process between an evolving set of ontologies. Next section details this process.

3.2. Maintaining the unified view

Creating the unified view consists in incrementally merging the set of given schemas. This set is organized in a graph whose nodes are schemas and edges mappings relating schemas. More precisely, a unified view will be created over a subgraph of such a graph using the nodes corresponding schemas to be integrated.

Definition 3.1. Let the *context graph* be a graph $C = \langle S_c, M_c \rangle$ whose nodes S_c represent the schemas to be merged and there is an edge $m(a, b) \in M_c$ if there is a mapping between the schemas corresponding to the nodes a and b .

Definition 3.2. Let the *merge candidate graph* be a graph $G = \langle S_g, M_g / S_g \sqsubset S_c, M_g \sqsubset M_c \rangle$

Integrating the schemas raises the following obvious requirement on the topology of the context graph.

Proposition 3.1. *Either there is a path in G between every two nodes $a, b \in G$ or there is a path between a and b in C .*

Proposition 3.1 indicates that if merge candidate graph is not connected, then context graph must be so. The proof is obvious as two schemas cannot be integrated if there is no mapping (even indirect) between them. This proposition leads us to the problem of composing mappings. When there is no direct mapping between two schemas but there is a path between their corresponding nodes in the context graph, it is necessary to compose mappings in order to create a direct edge between the two nodes. Composing mappings is studied in databases [15], we plan to study the applicability of such methods to ontological schemas in future work (see Section 5).

Supposing the existence of a *Compose* operator, we are able to preprocess the merge candidate graph in order to add missing mappings using

the composition operator. We obtain an extended merge candidate graph which is now connected.

Definition 3.3. Let *extended merge candidate graph* be a graph $G^{ext} = \langle S_{g_{ext}}, Mg_{ext}/S_{g_{ext}} = S_g, Mg_{ext} = M_g \cup M_{compose} \rangle$ where $M_{compose}$ represents the set of mappings resulting of the *Compose* operator applied on disconnected components of the merge candidate graph.

The merge operator defined above take a preferred schema in order to solve naming and possible conflicts. We choose this preferred schema with the goal of minimizing the need of composing mappings. This leads us to give an order in which the unified view is created using merge. We make the assumption that the most *central* nodes in the merge candidate graph should be merged first, the process iterating with lesser central nodes, the growing unified view being always the preferred schema. We measure the *betweenness* of each nodes (see for example the definition in [16]) in order to determine its centrality in the ontologies network^c.

We suppose that there is always a mapping between the unified view and the schema to be merged. In practice this mapping has to be deduced from the schema already in the unified view whose corresponding node in the extended merge candidate graph is a neighbor of the node representing the schema to be merged. It can be easily proven that using a centrality ordering and having a connected graph, less central nodes will always be linked to a node which schema has previously been integrated in the central view.

A few work are dealing with automatically merging schemas. We review them in the next section.

4. Related Work

Most of the work on schema merging has been realized on semi-automatic merging. The first significant work in this area has been realized by C. Batini and M. Lenzerini *et al* in [11]. They present a comparative study on methods to merge database schemas and thus identify a number of criterion categorizing these methods. They particularly identify the inputs and outputs of the merging process and present the integration process cycle as:

^cWe did not include the algorithm here for space reasons. We provide it on internet at <http://www.scharffe.fr/pub/swiis2007/algorithm3.jpg>

Comparison of the schemas Corresponds to the process of mapping schemas.

Conforming the schemas Possible conflicts are solved during this phase

Merging and restructuring The proper merge of the schemas. Quality of the merge operation can be measured on the criteria of *completeness*, *correctness*, *minimality* and *understandability*.

This works give as well a taxonomy of the different integration strategies encountered. This review, giving the bases for database schema integration doesn't deal with automation of the process. It also considers database schemas and thus doesn't take into account hierarchical structures.

The mapping of the mappings to logical rules is an important part of the implementation studied in the data integration field. Local-as-view, Global-as-view approaches and their derivatives [1,12] provide techniques for writing the mapping rules by considering the global schema in terms of the local schemas or the other way around. More recently such techniques have been adapted for ontology integration, considering more expressive formalisms like description logics [17].

More recent research involving the merging of ontologies is presented by N. Noy *et al* and M. Musen in [18]. They propose a tool named PROMPT that helps to iteratively merge two ontologies based on suggestions from an algorithm. Most of the work is concentrated on the suggestion algorithm using linguistic and graph matching techniques to propose merge candidates. Based on the user choices, PROMPT proposes a set of operations which outcome is a new set of suggestions. The process guide the user in completely merging two ontologies. It also include some conflict resolution functionalities if entities are duplicated or dangling. Despite the popularity of this tool as being performant on merging ontologies, it is of medium relevance for our algorithm focuses on an automatic process, where mappings are already given.

In the RONDO system [19] S. Melnik and E.Rahm and P. Bernstein present an actual merge algorithm which input is as our algorithm taking two schemas and a mapping between them. The algorithm consists in three steps: node renaming, graph union and conflict resolution. The last step is let to be solved to the human engineer. In order to facilitate the conflict resolution procedure, the algorithm tags the merged nodes with a priority measure relating a preference over them. This algorithm only deals with equivalence, one-top-one mapping rules.

A more theoretical work is presented by P. Buneman, S. Davidson and A. Kosky in [20]. They study the merge operation and present a general

technique to merge data models. Data models are first translated in a direct graph which nodes are classes and arcs relations or attributes of our model^d. They consider merge as being a *commutative* and *associative* operation, so that schemas can be iteratively merge in any order. They distinguish between upper merge and lower merge operations which respectively return the least upper bound and the greatest lower bound of a collection of schemas in terms of information ordering.

Finally, in her PhD thesis [21], R. Pottinger considers a generic merge operator, as well as an merging algorithm. Her approach deals with complex mapping representation as a third party schema. She introduce the notion of *preferred model* for conflicts resolution. The merge operator defined is commutative and associative, as defined in [11]. This work is the closest we found to ours. The dynamic aspects of the merge operation are however not being taken into account^e.

5. Conclusions and Future Work

We have presented the evolution of the the information integration problem for structured data sources. From a closed world with a known, mostly static number of sources it evolved with the apparition of the web towards a dynamic environment. From a network of interrelated ontologies we propose the automatic generation and maintenance of a unified view. This approach presents the advantages for the user to only have to query one schema. It also reduces the complexity of the necessary process of translating queries between the sources in the network. Our future work will concentrate on exploiting more complex mappings between the ontology entities. We will also study how mappings transitivity can be used to reach new sources.

References

1. M. R. Genesereth, A. M. Keller and O. Duschka, Infomaster: An information integration system, in *proceedings of 1997 ACM SIGMOD Conference*, 1997.
2. D. Beneventano and S. Bergamaschi, The momis methodology for integrating heterogeneous data sources, in *IFIP World Computer Congress*, (Toulouse, France, 2004).
3. G. Wiederhold, *Computer* **25**, 38 (1992).
4. A. Loeser, W. Siberski, M. Wolpers and W. Nejdl, Information integration in schema-based peer-to-peer networks, in *Proceedings of the Conference on Advanced Information Systems Engineering*, June 2003.

^dIncluding the meta relations Sub- and Super-

^eIn the related experiment it takes 20 hours to merge two big medical taxonomies.

5. L. Predoiu, F. Martin-Recuerda, A. Polleres, C. Feier, A. Mocan, J. de Bruijn, F. Porto, D. Foxvog and K. Zimmermann, *Framework for Representing Ontology Networks with Mappings that Deal with Conflicting and Complementary Concept Definitions*, tech. rep., DIP EU project, FP6 - 507483 (2004).
6. B.-L. T., J. Hendler and O. Lassila, *Scientific American* (2001).
7. P. Shvaiko and J. Euzenat, *A Survey of Schema-based Matching Approaches*, Tech. Rep. DIT-04-087, University of Trento (2004).
8. E. Rahm and P. A. Bernstein, *VLDB Journal: Very Large Data Bases* **10**, 334 (2001).
9. A. Halevy, *ACM Queue* **3**, 50 (2005).
10. Z. Huang, F. van Harmelen and A. ten Teije, Reasoning with inconsistent ontologies, in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, (Edinburgh, Scotland, 2005).
11. C. Batini, M. Lenzerini and S. B. Navathe, *ACM Comput. Surv.* **18**, 323 (1986).
12. M. Lenzerini, Data integration: a theoretical perspective, in *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (ACM Press, New York, NY, USA, 2002).
13. G. Wiederhold and M. Genesereth, *IEEE Expert: Intelligent Systems and Their Applications* **12**, 38 (1997).
14. J. Euzenat, F. Scharffe and L. Serafini, *Specification of the Alignment Format*, tech. rep., Knowledge Web European Project (2006).
15. R. Fagin, P. G. Kolaitis, L. Popa and W.-C. Tan, *ACM Trans. Database Syst.* **30**, 994 (2005).
16. Betweenness definition in wikipedia
<http://en.wikipedia.org/wiki/Betweenness>.
17. D. Calvanese, G. D. Giacomo and M. Lenzerini, A framework for ontology integration, in *In Proc. of the First Semantic Web Working Symposium*, 2001.
18. N. F. Noy and M. A. Musen, Prompt: Algorithm and tool for automated ontology merging and alignment, in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, (AAAI Press / The MIT Press, 2000).
19. S. Melnik, E. Rahm and P. A. Bernstein, Rondo: A programming platform for generic model, in *In Proceedings of SIGMOD 03*, 2003.
20. P. Buneman, S. B. Davidson and A. Kosky, Theoretical aspects of schema merging, in *EDBT '92: Proceedings of the 3rd International Conference on Extending Database Technology*, (Springer-Verlag, London, UK, 1992).
21. R. Pottinger, Merging schemas and processing queries in support of data integration, PhD thesis, University of Washington 2004.