FORUM: A Flexible Data Integration System Based on Data Semantics*

Zohra Bellahsene
LIRMM
Montpellier – France
bella@lirmm.fr

Salima Benbernou †
LIPADE
Paris – France
salima.benbernou@parisdescartes.fr

Hélène Jaudoin
IRISA-ENSSAT
Lannion – France
jaudoin@enssat.fr

Francois Pinet
CEMAGREF
Clermont-Ferrand – France
francois.pinet@cemagref.fr

Olivier Pivert
IRISA-ENSSAT
Lannion – France
pivert@enssat.fr

Farouk Toumani
LIMOS
Clermont-Ferrand – France
ftoumani@isima.fr

ABSTRACT

The FORUM project aims at extending existing data integration techniques in order to facilitate the development of mediation systems in large and dynamic environments. It is well known from the literature that a crucial point that hampers the development and wide adoption of mediation systems lies in the high entry and maintenance costs of such systems. To overcome these barriers, the FORUM project investigates three main research issues: (i) automatic discovery of semantic correspondences (ii) consistency maintenance of mappings, and (iii) tolerant rewriting of queries in the presence of approximate mappings.

1. INTRODUCTION

Integrating and sharing information across disparate data sources entails several challenges: relevant data objects are split across multiple sources, often owned by different organizations. The data sources represent, maintain, and export their data using a variety of formats, interfaces and semantics. In the last two decades, much research work in the database community has been devoted to developing approaches and systems that enable information integration in heterogeneous and distributed environments. The FORUM project aims at leveraging integration technology in order to facilitate flexible information integration over a large number of sources. More precisely, we focused on three issues:

- flexibility of integration, mainly by investigating approaches that enable automatic schema mapping discovery,
- flexibility at a semantic level, by investigating the mapping maintenance and restructuring according to the evolution of the data sources, and
- flexibility and scalability of query processing: our aim is the design and development of flexible query rewriting algorithms that scale up in the number of available information sources.

In order to alleviate the integration task, we developed techniques that enable automatic discovery of semantic mappings between schemas of heterogeneous information sources. We implemented a tool that combines existing matching discovery algorithms. The system exploits machine learning techniques in order to combine the most appropriate algorithms with application domains.

The maintenance of schema mappings is an issue that deserves specific attention, more particularly in the context of dynamic environments where source contents and schemas may evolve very frequently. In the FORUM project, we addressed the problem of maintaining consistency of mappings between XML data when changes in source schemas occur. We proposed a two-step approach that exploits regular grammar's tree to first identify schema changes that may affect existing mappings and then suggest adequate incremental adaptation of the affected mappings.

In open environments, it is unrealistic to assume that the mappings may always be precisely defined. Thus, we studied the problem of answering queries in the context of a mediation system equipped with approximate mappings obtained from a relaxation

*Supported by ANR Research Grant ANR-05-MMSA-0007

†The work has been conducted while she was at LIRIS-université de Lyon

of the constraints present in user queries. We developed a flexible query rewriting technique that exploits approximate mappings in order to compute tolerant rewritings. From a formal point of view, we moved from the classical query semantics based on the notion of *certain answers* to a new semantics based on a notion of *probable answers*. Moreover, since the number of approximate rewritings may be very high, we devised a rewriting algorithm that generates only the top-k rewritings of a given query.

A prototype implementing the approach has been developed and experimented using hundreds of real-world data sources from the agricultural domain. A mediator schema was produced as well as correspondences between the sources and the mediator schema in order to facilitate data exchange as well as the querying process.

The remainder of the paper is structured as follows. Section 2 presents an automatic approach to the discovery of dependences, based on the use of a decision tree. Section 3 deals with schema mapping maintenance and describes an incremental approach that decomposes complex changes into atomic ones. Section 4 describes the flexible rewriting technique which relies on an approximate matching of interval constraints. Section 5 presents an application of these works to the traceability of agricultural activities. Section 6 recalls the main contributions and concludes the paper.

2. AUTOMATIC DISCOVERY OF SEMANTIC CORRESPONDENCES

Schema matching is the task of discovering correspondences between semantically similar elements of two schemas or ontologies [5, 11, 12, 13]. While mappings between a source schema and target schema specifies how the data in each of the source schemas is to be transformed to targed schema. In this subsection, our emphasis is on schema matching as it has been discussed in the survey by Rahm and Bernstein [16], and extended by Shvaiko and Euzenat in [21] with respect to semantic aspects. We have designed several algorithms for schema matching particularly in a large scale context [18]. Due to space limitation, we only describe the most recent of these approaches.

The kernel of traditional matching tools is the aggregation function, which combines the similarity values computed by different similarity measures. However, the aggregation function entails several major drawbacks: running all similarity measures between all elements from input schemas is expensive in terms of time performance and it can negatively influence the quality of matches. Furthermore, adding new similarity measures mainly implies to update the aggregation function. Finally, a threshold is applied on the aggregated value; yet, each similarity measure has its own value distribution, thus each should have its own threshold. The novel approach we propose has been implemented as a protoype named MatchPlanner [6], avoids the aforementioned drawbacks. Its principle consists in replacing the aggregation function by a decision tree. Let us recall that decision trees are predictive models in which leaves represent classes and branches stand for conjunction of features leading to one class. In our context, a decision tree is a tree whose internal nodes represent the similarity measures, and the edges stand for conditions on the result of the similarity measure. Thus, the decision tree contains plans (i.e., ordered sequences) of similarity measures. All leaf nodes in the tree are either *true* or *false*, indicating if there is a correspondence or not. We use well-known similarity measures from Second String [20], i.e., Levenshtein, trigrams, Jaro-Winkler, etc. We also added the neighbor context from [7], an annotation-based similarity measure, a restriction similarity measure and some dictionary-based techniques [23].

A first consequence of using a decision tree is that the *performance* is improved since the complexity is bounded by the height of the tree. Thus, only a subset of the similarity measures it involves is used for an actual matching task. The second advantage lies in the improvement of the *quality* of matches. Indeed, for a given domain, only the most suitable similarity measures are used. Moreover, the decision tree is flexible since new similarity measures can be added, whatever their output (discrete or continuous values).

Let us now outline the different steps of the algorithm. The similarity value computed by a similarity measure must satisfy the condition (continuous or discrete) on the edges to access a next node. Thus, when matching two schema elements with the decision tree, the first similarity measure — that at the root node — is used and returns a similarity value. According to this value, the edge for which its condition is satisfied leads to the next tree node. This process iterates until a leaf node is reached, indicating whether the two elements match or not. The final similarity value between two elements is the last one which has been computed, since we consider that the previous similarity values have only been computed to find the most appropriate similarity measure. Figure (1) illustrates an example of a decision tree. Now, let us illustrate how the

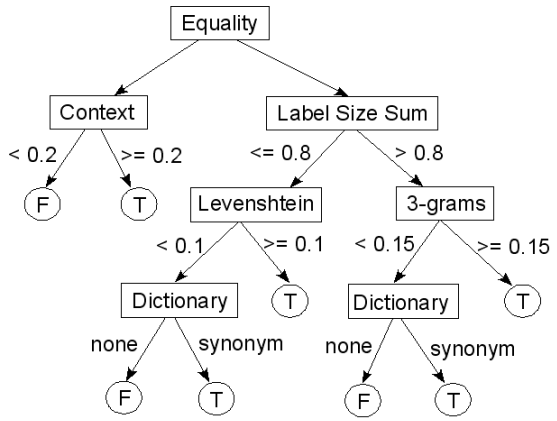

Figure 1: Example of decision tree

matching works using this tree and three pairs of labels to be compared.

- ***(quantity, amount)*** is first matched by *equality* which returns 0, then the *label sum size* is computed (value of 14), followed by the *3-grams* similarity measure. The similarity value obtained with 3-grams is low (0.1), implying the *dictionary* technique to be finally used to discover a synonym relationship.

- on the contrary, ***(type_of_culture, cultureType)*** is matched using *equality*, then *label sum size*, and finally *3-grams* which provides a sufficient similarity value (0.17) to stop the process.

- finally, the pair ***(analysis_date, analysis_date)*** involves identical labels, implying the *equality* measure to return 1. The *neighbor context* must then be computed to determine if there is a match or not. Indeed, both labels may refer to different analysis, for example one could represent a water analysis and the other stand for a report date.

Thus, only nine similarity measures have been used to discover the matches: four for *(quantity, amount)*, three for *(type_of_culture, cultureType)* and two for the last pair, instead of eighteen if all distinct similarity measures from the tree had been used. Indeed, with a matching tool based on an aggregation function, all of the similarity measures would have been applied for each pair of schema elements.

This approach has been implemented as a matching tool. Several default decision trees are provided and the user can select his/her configuration so as to emphasize e.g. the performance aspect or the quality of matches.

3. SCHEMA MAPPING MAINTENANCE

In dynamic networks such as peer to peer architectures, the nodes may change not only their data but also their schemas and their query domain. Facing this situation, schema mappings can become obsolete, a phase of maintenance is thus needed to maintain their consistency. Several solutions have been proposed to automate the adaptation of mappings when data schemes evolve. These solutions can be classified into two categories: (1) *incremental method* is implementing changes separately for each type of change occurs in the source and target schema [22]; (2) *Composition approach*, is a mapping-based representation of schema, and it is more flexible and expressive than the change-based representation [3].

Based on those categories, the proposed approach is not limited to the adaptation of affected matchings, but it also enables automatic detection of schema changes and decomposition of modifications into basic changes. It is two-phase approach:

1. processing on the schema to detect and automatically identify the changes,

2. processing on the mappings to provide an adaptation and keep the mappings consistent and compliant with the peer schemas.

Phase 1. We focused on handling the matchings for XML schemas. Then, the query languages for XML data model such as XQuery or XSLT are restricted to providing mechanisms for querying data sources, and the results are files in XML format. In this case, no device can be used to detect real time changes in the data schema. Facing situations that are most likely to cause changes, a first step is to determine if these treatments affect the structure and/or the semantics of the schema. Detecting a change in the schema level may then be interpreted as comparing two schemas (the old one and the new one), i.e., comparing two simple regular grammars' trees. This is possible since an *XML Schema* document can be represented by a regular grammar tree of a simple type (GARS) [14]. However, the existing GARS algorithms are similarity-metric-based approaches. In our case, the comparison between two GARS will provide elementary changes, and then should verify (1) syntactic equivalence (2) structural equivalence i.e., the tree generated using the first grammar must exhibit the same structure as the one generated by the second grammar. Due to space limitation, we just give an overview of the description.

The proposed algorithm will process in two steps, first it compares the terminal sets of both gram-

mars G_1 ,G_2 (associated with sources S_1 and S_2 resp), and the result is two sets t^1 and t^2 (terminals belonging to one grammar and not to the other), which is not enough to detect the changes. The second step will give more information by comparing each production rule with its juxtaposed rule, tackling the suppression, the renaming and the moving of an element. For instance, if an entity is moved, it will appear neither in t^1 nor in t^2, such change can only be detected by the production rules. The algorithm complexity is polynomial.

Phase 2. Once the detection is achieved, the next step is to report the modifications on the mapping in order to keep its semantics compliant. There exists different mapping representations based on a query language such as s-t-tgds[17]or [10], or very recent work in MapMerge prototype[2]. We used a mapping representation in a general way — a fragment of first order like language — that can be translated to any other existing representation. It including variables, constraints on the variables and the correspondences. Let us illustrate the mapping over our case study in the project — the agriculture area —, and consider the source schema S1 (*parcel 1* including an entity Info) and the target schema S2 (*parcel 2* including an entity Personal), one of the mappings is:

$$m ::= (\forall x \in S1.Info) \wedge (\exists y \in S2.Personal)) \wedge (y.Title = Director) \rightarrow [(x.NumFarmhand = y.SocialNum) \wedge (x.NameFarmHand = y.Name)]$$

It is represented by four parts and is interpreted as follows: (1) for each element x in the entity Info from the source $S1$ belonging to the mapping m, (2) it will exist elements y in the entity Personal (3) with a constraint on y (the title of the personal is Director — used to represent a restriction on the entities of the schemas) (4) there exist correspondences between variables defined by "=" function; *NumFarmhand* from the source $S1$ is equal to *SocialNum* from $S2$ and *NameFarmHand* ($S1$) equal to *Name* ($S2$).

An incremental approach is used in order to decompose complex changes into atomic ones. We classify them into *structural* (add, delete, modify an element of the tree), and *semantic* changes (add or delete, a correspondence constraint between two elements in the same schema or a key). Adding an element will not affect the mapping m, because it is a new information and we need to discover the matching between sources by means the algorithm presented in previous section. When deleting a complex element e (with many attributes), the adaptation in the mapping is limited to deleting the mapping assigned to e. If the deleted entity is atomic (one attribute) or belonging to a complex structure assigned to a mapping, the adaptation of m will be done over the substitution of clauses that refer to e in m (variable, conditions and correspondences). While displacing an entity (subtree), two adjustments are required:

- adaptation of the schema's internal constraints, thus, a redefinition of referential constraints must automatically be issued. Let e be the moving entity, o the origin of the reference and d its destination. There are three possible cases of adaptation of the reference during the displacement. In the first case, o and d belong to the same displaced structure; the reference remains unchanged. The second case describes a reference source $o \in e$ and its target $d \notin e$, the redefinition is achieved by taking into account the new location of o. Finally, if o does not belong to e, a new reference is set between o and d in its new location.
- adaptation of the mappings related to the displaced entity e. If e is complex, the adaptation is limited only to update the location, i.e., the variable access. If e is atomic, one needs to split the mapping and assign the element to a new mapping with constraints and correspondences. Renaming an element implies renaming all occurrences of e in the mapping. The referential changes (correspondences) will affect only the destination schema.

A prototype of the adaptation part has been developed towards the use case in agriculture.

4. FLEXIBLE QUERY REWRITING

In this section, we deal with flexible query answering in the FORUM data integration system. A Local-As-View type of approach is used, i.e., data sources are defined as views over the global schema. We consider the case where views and queries involve simple interval constraints $X \in [a, b]$ where X is an attribute name and a, b are two constants. The problem of rewriting queries using views in the presence of interval constraints is well known [15, 1]. However, in this section, we investigate this problem by means of a *tolerant* method. As an example, let us consider a query Q that aims to retrieve *codes* of *OMSParcels* (*organic matter spreading parcels*, i.e., agricultural plots receiving organic matter) whose *surface* is in [22, 35] hectares, and two views V_1 and V_2 such that:

V_1 provides *codes* of *OMSParcels* whose *surface* $\in [20, 40]$
V_2 provides *codes* of *OMSParcels* whose *surface* $\in [30, 45]$.

Both V_1 and V_2 have an interval constraint on attribute *surface*. However, none of these intervals is included in that of the query, thus the mappings between the two intervals and that of the query are only partial (imperfect). Moreover, since V_1 and V_2 only provide codes of parcels, selection on attribute surface is impossible, hence V_1 and V_2 cannot be used to get certain answers to Q. In such a context, rewriting algorithms based on the *certain answer* semantics fail to reformulate the query, thus to provide the user with any answer. The idea we advocate is to exploit approximate mappings between interval constraints involved in views and queries in order to compute *tolerant* query rewritings. Any such rewriting Q' is associated with a score between 0 and 1 which reflects the probability for a tuple returned by Q' to satisfy the initial query Q.

One considers any candidate rewriting, i.e., a rewriting which is contained in Q when interval constraints are ignored. Then one computes an *inclusion degree* between each pair of intervals I'_Q and I_Q restricting a same attribute X as follows:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \frac{\mid I_{Q'} \cap I_Q \mid}{\mid I_{Q'} \mid} \qquad (1)$$

This degree corresponds to the proportion of elements from $I_{Q'}$ which are in $I_{Q'} \cap I_Q$ when the distribution of the values over the domain is continuous and uniform.

Let us come back to the previous example and consider the intervals $I_{V_1} = [20, 40]$ and $I_Q = [22, 35]$ respectively involved in view V_1 and query Q. The inclusion degree between I_{V_1} and I_Q equals $\alpha = \frac{|[22,35]|}{|[20,40]|} = \frac{13}{20} = 0.65$. We can then attach this degree to the rewriting V_1 of Q, which is denoted by $V_1 \sqsubseteq_{0.65} Q$. According to the semantics of Equation 1, it means that an answer to V_1 has the probability 0.65 to satisfy the constraints of Q.

When the query involves several constraints, the degrees obtained are aggregated in order to calculate the global score associated with Q'. Notice that the inclusion degree associated with an attribute X denotes the probability that an answer to Q' satisfy the constraint over X in Q. Then the global score expresses the *probability* for an answer returned by the rewriting Q' to be an answer to Q. When constraints are attached with *existential* variables in views, i.e., attributes that do not appear in the head of views, the assumption of independence between constraints is guaranteed and the degrees can be aggregated with the product operation. Indeed, if a view covers a query subgoal as well as an interval constraint over one of its existential variables, it must also cover any query subgoal involving this variable [15]. Consequently, in a same rewriting, it is not possible to have more than one view that restricts the value domain of an existential variable. The only case where the independence assumption may be violated concerns *distinguished* variables in views, i.e., attributes that appear in the head of views. Indeed, several views occurring in a given rewriting may involve interval constraints on a same distinguished variable. In such a case, the overall degree cannot be computed as an aggregation of partial degrees, but must be based on the intersection of the interval constraints. The degree attached to a distinguished variable is either 0 or 1, depending on whether the new interval is disjoint or not from that of Q. If it is 0, the rewriting is dismissed, whereas if it is 1, the constraint from Q is added to the rewriting.

As an example, let us now consider a query Q which aims at retrieving *codes* of *OMSParcels* whose *surface* is in [22, 35] hectares and located in a city with a wastewater treatment plant ($STEP$) whose *capacity* is in [1200, 2300], and two views:

$V_1(code,\ citycode) :-$
$\quad OMSParcels(code,\ surface,\ citycode),\ surface \in [18,\ 38]$
$V_2(citycode) :- STEP(citycode,\ capacity),\ capacity \in [1000, 2400].$

Let $Q_1(code) :- V_1(code, citycode), V_2(citycode)$ be a tolerant rewriting of Q. The degree α_1 attached to the tolerant rewriting V_1 of the first subgoal of Q is computed from the interval constraints on surface and it equals $\frac{|[22,35]|}{|[18,38]|} = \frac{13}{20}$ while the degree α_2 attached to the tolerant rewriting V_2 of the second subgoal of Q is computed from the interval constraints on capacity and it equals $\frac{|[1200,2300]|}{|[1000,2400]|} = \frac{11}{14}$. Therefore, $Q_1(code)$ gets the degree $\alpha_1 * \alpha_2 = 0.51$ and $Q_1 \sqsubseteq_{0.51} Q$. The tuples issued from Q_1 have a probability over 50% to satisfy the constraints involved in Q.

As can be seen, the rewriting mechanism proposed is not based on the notion of *certain answers* anymore, but rather on that of *probable answers*. The idea of computing probable answers is not new since it has been used for open integration contexts [4, 19]. The nature of the probability degree associated with the answers returned in our approach is different from that considered in these works, though. Indeed, in [4] it indicates the probability of *existence* of the answers produced, while in [19] it ensues from the uncertainty pervading the mappings between several data sources.

Since approximate mappings between queries and views are obviously more frequent than strict mappings, the number of possible tolerant rewritings may be huge. To cope with this situation, we propose an algorithm which generates only the top-k rewritings to a given query. This algorithm rests on the principle of a well-known regular query rewriting algorithm, namely `Minicon` [15]. It slightly modifies the first step of the `Minicon` by associating a degree with any view able to rewrite a subgoal of the query. Subsequently, as the second step of the `Minicon` amounts to the computation of the exact covers of a hypergraph (i.e., those of the subgoals of a query), our algorithm exploits an efficient structure to implement it, called *dancing links* [9], in order to generate only the k best rewritings.

5. EXPERIMENTS

In a decision-making context, the different protagonists of the agricultural domain need to exchange data. Each protagonist should be able to obtain information according to his specific requirements: a farmer may need information for choosing the place of his crops, a public authority needs to monitor the environmental impacts of agricultural activities, etc. Due to the generalization of the traceability of the agricultural practices, each protagonist produces more and more data. Indeed, the amount of data involved is huge. For example, in France one counts more than 500,000 farms and thousands of agencies or services that take part in the agricultural activity. The need for exchanging and querying this information is crucial in this application. Experts in the field of agriculture in France participated in the project. They helped propose a coherent exchange scenario by identifying the possible protagonists and their data needs. Even if we did not deploy our system at a national level in France, the goal was to provide a scenario sufficiently realistic to be implemented. Numerous data sources have been collected for the project. We have conducted experiments for integrating 300 data sources issued from different French institutions and farms. More precisely, data sources are extractions of databases or Excel files. Each extraction is in fact a view (e.g., a table). The first challenging issue is to build the mediated schema over the different data sources. Many concepts are common to different sources (environmental objects, crops, farms, etc) but the expertise of each source is specific. Designing the mediated schema required finding the common concepts between the sources. Manually finding the correspondences between sources is an hard task for experts. We used MatchPlanner, the automatic schema matching tool presented in Section 2. The list of correspondences that it provided helped highlight the concepts repeated in several sources i.e. the potential relations of the mediated schema. The experts (in)validated this list manually by removing the false positives and by adding new correspondences. This process allowed to discover forty common concepts (each one corresponds to a table of the mediated schema). Experts highlighted that one can save a significant amount of time by using the schema matching tool.

Then, the mappings between the mediated schema and the data sources were described manually following an LAV approach. Each mapping was modelled as a Datalog clause; several hundreds of Datalog rules were thus written. Here an example of such a mapping:

```
wastewaterPlantRegion1(wwpIdSandre,wwpName,
     managerName,managerId):-
     manager(_,managerId,_,managerName,_,_)
     wastewaterPlant(_,wwpIdSandre,
                     wwpName,_,_,_,
                     managerIdSiret).
```

`wastewaterPlantRegion1` is a source gathering the list of wastewater plants that belong to the geographical area 1. In the body of the rule, `manager` and `wastewaterPlant` are tables in the mediated schema. Five variables are involved in this mapping:

1. wwpIdSandre: id of the wastewater plant expressed in format named Sandre (French format),
2. wwpName: tname of the wastewater plant,
3. managerName: name of the main manager of a wastewater plant,
4. managerId: id of the manager of a wastewater plant,
5. managerIdSiret: id of the managers expressed in a specific format called Siret (French format).

So as to test the flexible rewriting technique, the experts expressed certain attributes values as intervals of values. Note that for many queries there are only rewritings with a degree less than 1. Therefore, in many cases a classical approach does not compute any rewriting.

The approach has been implemented in a prototype using Java SE 5, and PostgreSQL 8.3 databases. Twenty queries provided by real users were processed. The prototype computes the best rewritings

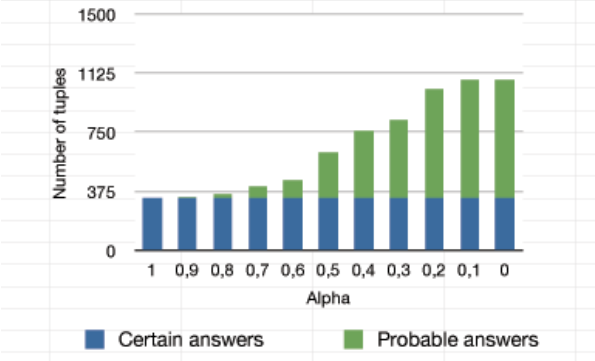

Figure 2: Impact of α_{min}

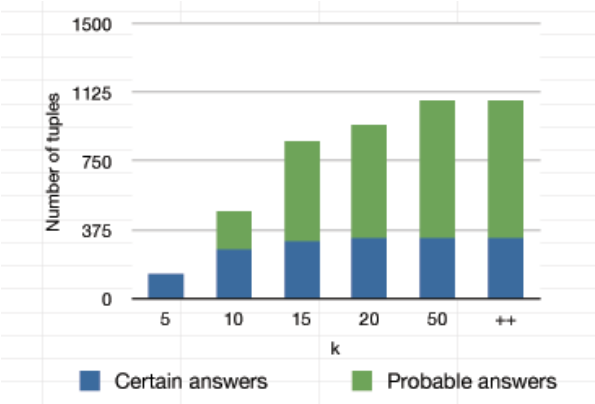

Figure 3: Impact of k

associated with a given query: either the top-k ones (k being an integer) or those whose associated degree is over a given threshold $\alpha_{min} \in]0, 1]$. The first experimentation (cf. Fig. 2) shows the evolution of the *average* number of answers, w.r.t. the number of certain anwers computed by the `MiniCon` algorithm, when the threshold α_{min} changes. Here the parameter k is fixed to infinity. Notice that the number of certain answers remains constant. Indeed, all exact rewritings are computed for any value of α_{min}. On the other hand, the number of probable answers decreases when the threshold increases. When α_{min} equals 1, only certain answers are provided. In the second experimentation (cf. Fig. 3), the threshold is set to 0 but k, the number of best rewritings sought for, must be specified. When k decreases, the average number of probable answers decreases too, which shows that only the best rewritings are generated. When k is less or equal than 10, the number of certain answers decreases too. Indeed, for some queries, a number of 10 rewritings is not enough to compute all the certain answers. The prototype shows very good performances: the execution time is about 2 seconds in the worst case, and the performances of the approach are comparable to that of the classical `MiniCon` algorithm (in the worst case, the overhead is only 10%). See [8] for more detail.

6. CONCLUSION

The main objective of the FORUM project was to develop new methods and techniques for flexible data integration. This project has led to significant advances in the field. The results were published in international journals and conferences. Original and efficient tools have been developed and experimented in an environmental application.

Finally, the project has enabled the development of collaborations between partners. For example, a collaboration between LIMOS and IRISA teams led to the development of a novel approach for flexible query rewriting. A collaboration between LIRMM and IRISA will begin soon on schema integration using fuzzy set and possibility theories to capture uncertainty resulting from the use of similarity measures.

7. ACKNOWLEDGMENTS

Work partially funded by the ANR Research Grant ANR-05-MMSA-0007. We would like to thank the reviewers for their very fruitful comments to improve the paper.

8. ADDITIONAL AUTHORS

Stephan Bernard (CEMAGREF, stephan.bernard@cemagref.fr), Pierre Colomb (LIMOS, colomb@isima.fr), Remi Coletta (LIRMM, coletta@lirmm.fr), Emmanuel Coquery (LIRIS, emmanuel.coquery@liris.cnrs.fr), Fabien de Marchi (LIRIS, fabien.demarchi@liris.cnrs.fr), Fabien Duchateau (LIRMM, duchatea@lirmm.fr), Mohand-Saïd Hacid (LIRIS, mohand-said.hacid@liris.cnrs.fr), Allel Hadjali (IRISA, hadjali@enssat.fr) and Mathieu Roche (LIRMM, mroche@lirmm.fr)

9. REFERENCES

[1] F. N. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, pages 209–220, 2002.

[2] B.Alexe, M. Hernández, L.Popa, and W.C.Tan. Mapmerge: Correlating independent schema mappings. *PVLDB*, 3(1):81–92, 2010.

[3] P. A. Bernstein, T. J. Green, S. Melnik, and A. Nash. Implementing mapping composition. *VLDB J.*, 17(2):333–353, 2008.

[4] N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *Proc. of VLDB 2005*, pages 805–816. VLDB Endowment, 2005.

[5] H. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *IWWD*, 2003.

[6] F. Duchateau, Z. Bellahsene, and R. Coletta. A flexible approach for planning schema matching algorithms. In *OTM Conferences*, pages 249–264, 2008.

[7] F. Duchateau, Z. Bellahsene, and M. Roche. A context-based measure for discovering approximate semantic matching between schema elements. In *RCIS*, 2007.

[8] H. Jaudoin, P. Colomb, and O. Pivert. Ranking approximate query rewritings based on views. In *Proc. of the 8th International Conference on Flexible Query Answering Systems (FQAS'09)*, pages 13–24, Roskilde, Denmark, 2009.

[9] D. Knuth. Dancing links. *Arxiv preprint cs.DS/0011047*, 2000.

[10] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 61–75, New York, NY, USA, 2005. ACM.

[11] C. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *VLDB*, 1994.

[12] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, pages 49–58, 2001.

[13] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *VLDB*, pages 122–133, 1998.

[14] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4):660–704, 2005.

[15] R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB*, pages 484–495, San Francisco, CA, USA, 2000.

[16] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[17] R.Fagin, P. Kolaitis, R.Miller, and L.Popa. Data exchange:semantic and query answering. In *Proc. ICDT 2003*, 2003.

[18] K. Saleem, Z. Bellahsene, and E. Hunt. PORSCHE: Performance ORiented SCHEma mediation. *Information Systems - Elsevier*, 33(7-8):637–657, 2008.

[19] A. D. Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proc. of SIGMOD 2008*, pages 861–874. ACM, 2008.

[20] Secondstring. http://secondstring.sourceforge.net/, 2007.

[21] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, pages 146–171, 2005.

[22] Y. Velegrakis, R. Miller, and L. Popa. Preserving mapping consistency under schema changes. *The VLDB Journal*, 13(3):274–293, 2004.

[23] Wordnet. http://wordnet.princeton.edu, 2007.