D2I

Integrazione, Warehousing e Mining di sorgenti eterogenee

Programma di ricerca (cofinanziato dal MURST, esercizio 2000)

Descrizione del Linguaggio Fuzzy per l'Interrogazione di Viste Riconciliate

Silvana Castano, Danilo Montesi, Alberto Trombetta

D1.R10 **21 Gennaio 2002**

Sommario

Tema	Tema 1: Applicazioni per basi di dati su Internet e Intranet
Codice	D1.R10
Data	21 Gennaio 2002
Tipo di prodotto	Rapporto tecnico
Numero di pagine	12
Unità responsabile	MI
Unità coinvolte	MI
Autore da contattare	Danilo Montesi Dipartimento di Informatica Università di Bologna Mura Anteo Zamboni, 7, 40127 Bologna , Italia montesi@cs.unibo.it

Descrizione del Linguaggio Fuzzy per l'Interrogazione di Viste Riconciliate

Silvana Castano, Danilo Montesi, Alberto Trombetta

21 Gennaio 2002

Abstract

The Global-As-View approach in data integration system has focused on the (semi-) automatic definition of a global schema starting from a given set of known information sources. In this paper, we investigate how to employ concepts and techniques to model imprecision in generating the global schema and in specifying the mappings. We propose a language for defining fuzzy-based mappings between source and global schemas. Such mappings explicitly take into account the similarities occurring between global and source schemas, thus allowing to discard the items with low similarity and to express the relevances that different sources may have in populating the global schema. The proposed language is applied in the framework of the ARTEMIS/MOMIS integration system.

1 Introduction

Data integration systems are composed of three main components: source schemas, a global schema and a mapping between the two [1]. The automatic generation of a global, integrated schema starting from a set of independent source schemas is a well-known research problem. The approach usually employed in trying to solve this problem is called *Global-As-View* (GAV). Such approach requires that the global schema is defined in terms of the source schemas: every element of the global schema is expressed as a view over the source schemas, so that its meaning is specified in terms of the data residing at the sources [11, 1]. Thus, in addition to works dealing with global schema, other works investigate how to specify the mappings through views generation [8].

Imprecision is modelled through fuzzy sets to express similarities among source schemas and the global schema. Fuzzy sets are also used to enhance the (semi-)automatic generation of queries over the sources populating the global schema. As our starting point, we employ the ARTEMIS/MOMIS system for the analysis of source schemas and the creation of the integrated global schema [3, 2]. The fuzzy-based extensions aim at the improvement of source data integration. The same fuzzy-based extension is used to enhance the (semi-)automatic generation of queries over the sources populating the global schema. To capture imprecision in schema mappings we adopt in this work an extended relational algebra using fuzzy sets instead of crisp sets [4].

The paper is organized as follows. Section 2 shows a running example motivating the need to deal with imprecision in both global schema creation and mapping generation. Section 3 presents the notation to model imprecision with fuzzy sets within the data model and the query language. Section 4 concisely surveys the ARTEMIS/MOMIS system. Section 5 presents the fuzzy-based mapping generation algorithm. Section 6 concludes the paper.

2 Motivating Example

The example motivating the need for modeling imprecision in tools defining the queries over different sources to populate the integrated schema with data derived from the source schemas comes from the following scenario: consider a patient being hospitalized in a hospital facility, needing some clinical exams. First of all, the patient is registered at the admission desk; later on, the patient is examined by the PET laboratory and, thereafter, he is examined by the radiology lab. After the exams, the patient is dismissed. Every previous step requires to record information about the patient in different, independent databases: as the patient enters the hospital facility, his name – along with his social security number and other information – is stored in the administrative database of the the facility. The schema of this database contains a single table *AdmissionPatientRecord* storing data about patients' admissions.

`AdmissionPatientRecord`

SSN	FullName	DateAdm	AdminRef
234998491	Smith,Joe	03/03/01	23
213456776	Robson, Michael	10/04/01	35
456921320	Wilson, Martha	10/04/01	3

In particular, the attribute `SSN` (of type `integer`) stores patients' social security number and it is the primary key, the attribute `FullName` (of type `char(20)`) stores patients' full name, the attribute `DateAdm` (of type `dd/mm/yy`) stores the date of patient' admissions in the Hospital, the attribute `AdminRef` (of type `integer`) stores a code related to patients that will be processed by the hospital administration.

When the patient enters the PET laboratory, another database – storing information about patients examined by such laboratory – is modified with the insertion of data about his exams. Note that such database is totally independent from the database storing data about patients' admissions. The schema of this database contains a sigletable `PETLabRec` storing data about patients' exams in the PET laboratory.

`PetLabRec`

code	PatName	DateExam	DateResp	AdminCode	LabRef
10223	A. Watson	12/05/01	3	43	pet5
10225	A. Reese	12/04/01	3	15	pet4

The attribute `code` is of type `integer` and identifies patients who took exams in the PET lab (note that if the same patient takes two exams in this laboratory, he will be given two different codes). `PatName` (of type `char(50)`) is the patient's name, `DateExam` (of type `dd/mm/yy`) is the date of the patient's exam, `DateResp` (of type `integer`) denotes the number of the week in which the exam's response is released, `AdminCode` is a foreign key referencing to attribute `SSN` in the schema containing table `AdmissionPatientRecord` while `LabRef` (of type `char(100)`) is the URL of a web page displaying information about the PET lab. Similarly for the radiology laboratory. Whose database contains the table `RadLab`.

RadLab

CP	Name	Surname	TimeEx	Response	RadRef
rad754	Joe	Adams	14/04/04/01	07/04	45rd
rad5901	David	Foley	15/04/04/01	07/04	57rd
rad308	William	Smith	14/23/03/01	26/03	79rd

The attribute `CP` (of type `char(20)`) stores a code uniquely identifying patients taking exams in the radiology lab, attributes `Name` and `Surname` (both of type `char(20)`) store patients' name and surname, the attribute `TimeEx` is the exam's date, while `Response` is the the release day of the exam's response.

Consider a situation in which the hospital facility wants to allow the retrieval of data about patients' exams using, for example, its web site. One solution would be to make every database involved in the storage of patients' dataavailable over the Web. In order to pose relevant queries, a user would be aware of all the involved databases' schemas.

A more flexible solution is to consider a data integration system that semi-automatically builds an integrated global schema given the three different source schemas. Once the global schema has been created from the source schemas exploiting their similarities, it must be populated (semi-automatically) through the queries providing the mapping between the source schemas and the global schema. This can be done considering again the similarities occurring among source schemas and global schema – in particular, those occurring among source attribute domains and global attribute domains.

As we have already said, the generation process of the global schema takes advantage of the similarities occurring among the source schemas. In particular, we refer to the approach of the ARTEMIS/MOMIS system [2] to derive the integrated global schema. Following this approach, several kind of similarities (called affinities) occurring among different elements of the source schemas are taken into account (see Section 4.1). The global schema derived by ARTEMIS is crisp: no information about similarities is kept and consequently cannot be deployed in the definition of the global schema and the queries over the sources, whose answers populate the global schema.

Our approach exploits a fuzzy-based relational model, in order to help the definition of such queries. In this paper, we show how a global schema (obtained with ARTEMIS/MOMIS system) is extended in order to store information in the global schema about source domains similarities. This is achieved by using similarity predicates over (source and global) attribute domain types and allowing the global schema attribute domains to be fuzzy sets.

3 The Fuzzy Data Model

Aim of this section is to provide a brief introduction to a fuzzy based relational model [4], suitable for the explicit representation of similarities occurring whether at attribute level or at relation level. The choice of an extended relational model is motivated by the wide diffusion of the relational model and by the fact that it is possible to extend the operators of the corresponding relational algebra with similarity based query capabilities, as well as introduce new operators expressing very powerful queries over similarity based data. We begin introducing few fundamental concepts from fuzzy set theory, then we proceed defining our extended relational model and finally we briefly introduce the corresponding extended relational algebra, on which the query language presented in Section5 is based.

We now briefly introduce some standard notation and definitions. Given a non-empty domain $\mathcal{D}$, a *fuzzy* (sub)set $\mathbf{S}$ of $\mathcal{D}$ is denoted by its *characteristic function*

$$\mu_{\mathbf{S}} : \mathcal{D} \to [0, 1]$$

assigning to every element $d \in \mathcal{D}$ its *fuzzy membership score* $\mu \in [0, 1]$ to the fuzzy set $\mathbf{S}$. We say that d belongs to $\mathbf{S}$ with score μ and write $d \in_\mu \mathbf{S}$ (or $d : \mu$ if $\mathbf{S}$ is clear from the context). If $\mu = 0$, we say that d does not belong to $\mathbf{S}$. As we have seen in Section 1, we represent global attribute domains as fuzzy sets. The elements' scores of a global attribute domain denote how much the global attribute domain and the source attribute domains (from which the global attribute domain elements are taken) are similar. In this way, the global attribute domain

$$\mathcal{D}_{ExamResponse} = \{1, 2, \ldots, 30, 01/01, 02/01, \ldots, 31/12\}$$

contains elements coming from two source attribute domains $\mathcal{D}_{DateResp}$, and $\mathcal{D}_{Response}$. The similarities of such domains (with respect with the global attribute domain) are represented as the membership scores of the corresponding elements coming from such source domains.

Given a set of attribute names, $\mathcal{A}$, and a function *dom* which associates to each $A \in \mathcal{A}$ a value domain, $\mathcal{D} = dom(A)$, a (named) relation schema is formed by a relation name, R, a subset of attributes $X = \{A_1, \ldots, A_n\} \subseteq \mathcal{A}$, along with their corresponding domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, and it is written as $R(A_1(\mathcal{D}_1), \ldots, A_n(\mathcal{D}_n))$. A tuple $t = \langle t_1, \ldots, t_n \rangle$ over $R(X)$ is any element of $dom(X) = dom(A_1) \times \ldots \times dom(A_n)$, and $t.A_i$ denotes the value of A_i in t.

The data model presented in [4] extends the relational one by allowing both *fuzzy attributes* and *fuzzy relations*. Imprecision at the attribute level is captured by the notion of fuzzy subset previously presented.

We say that A_i is a fuzzy attribute if its values are pairs of the form $v_i : \mu_i$, where v_i is an element of a fuzzy subset of the attribute domain and $\mu_i \in [0, 1]$ is the corresponding score. The two components of A_i can be referred to as A_i^v (the "value") and A_i^μ (the "score"), respectively. Intuitively, given a tuple t, $t.A_i$ is interpreted as "t fits A_i with score μ". Imprecision can also occur at the whole tuple level, and motivates the introduction of fuzzy relations. A fuzzy relation r over $R(X)$ is defined as a fuzzy subset of $dom(X)$, characterized by a membership function μ_r (or simply μ if r is clear from the context) which assigns to each tuple t a grade $\mu(t) \in [0, 1]$. The notation $t.\mu$ will be used with the same meaning of $\mu(t)$. We say that t belongs to r ($t \in r$)

iff $t.\mu > 0$, and r is called a *crisp* instance iff $t.\mu = 1$ for each $t \in r$. A *fuzzy database schema* R is a set of *fuzzy relation schemas*, $R = \{R_1, \ldots, R_n\}$. Note that the difference between a classical relational schema and a fuzzy relational schema lies in the fact that the former has only crisp instances, while the latter can have also fuzzy instances. A *fuzzy database* R is a set of fuzzy relations $r = \{r_1, \ldots, r_n\}$, where each r_i is a fuzzy set of fuzzy tuples $t_1, \ldots, t_u$, instance of the corresponding schema R_i. The values of the attributes $A_1, \ldots, A_n$ of r_i range over fuzzy domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$.

3.1 The Query Language

Our SQL-like query language extends the well-known Web query language WebSQL [7] with fuzzy based query capabilities and answer restructuring capabilities [12]. Here we give a brief overview of the features of our query language that are employed by the extended ARTEMIS/MOMIS system for the definition of the mappings between source and global schemas presented in Section 5. Aside from classical selection conditions, the query language bases the answer's items retrieval also on their membership degrees in the fuzzy relations specified in the queries and on their relevance. Such non-classical features contribute in the definition of the answer satisfaction degrees of the items.

A query in our extended WebSQL language is written using a classical `SELECT - FROM - WHERE` construct appropriately extended in order to express non-classical features, such as membership drees and relevances. More precisely, a query in the extended WebSQL query language has the form:

`SELECT` $A_1, A_2, \ldots, A_n$
`FROM` $R_1, \ldots, R_k$
`WHERE` $Cond$
`HAVING RELEVANCE` $Threshold$

In the `SELECT` clause, A_i, $1 \leq i \leq n$ denotes either an attribute from one relation mentioned in the `FROM` clause or its fuzzy membership degree. The syntax of the `WHERE` clause is extented with the construct `HAVING RELEVANCE`, that drops from the answer the tuples having membership degree lower than the specified value $Threshold$.

As in the classical case, the results of two queries Q_1 and Q_2 can be merged using the `UNION` operator. The fuzzy membership degrees of the tuples contained both in in the answers of Q_1 and Q_2 is the maximum between their fuzzy membership degrees in the answers of Q_1 and Q_2, respectively.

Unlike the classical case, it is possible to express the different relevance that tuples satisfying different queries may have. If we consider two queries Q_1 and Q_2, the relevance of the tuples contained in their answers is denoted with weights θ_1 and θ_2. The higher the relevance of the tuples satisfying a query, the higher the corresponding weight. This relevance clearly influences the fuzzy membership degrees of the tuples contained in the answer of the weighted union. How this actually happens is explained in [5].

Given two queries Q_1, Q_2 and weights θ_1, θ_2 (both of them between 0 and 1)), the syntax of the weihted union operator is the following:

Q_1 `WITH WEIGHT` θ_1 `UNION` Q_2 `WITH WEIGHT` θ_2

4 Global Schema Definition

The definition of the global schema, starting from various, independent source schemas is based on the ARTEMIS tool environment [2]. In Section 4.1, we will give a brief overview of ARTEMIS. However, since ARTEMIS defines a crisp global schema, we extend it in order to define global schemas whose attributes take values over fuzzy domains. We show our extensions to ARTEMIS in Section 4.2.

4.1 ARTEMIS Overview

The ARTEMIS system provides a theoretical framework and an associated tool environment for the analysis of semantic heterogeneity of source schemas and for their proper reconciliation into in an integrated, unified schema. Although ARTEMIS can manage information sources defined in various data models (e.g. relational, object-oriented, XML), we limit ourselves in considering only the relational setting. The main features of the ARTEMIS approach can be summarized in the following three points:

- *Analysis of source schema elements*, to identify similar elements in different source schemas. Similarities among different schema elements are derived from the terminology and structure of elements of source schemas. Term semantic is usually provided by thesauri and taxonomies: terms are grouped according to their meaning using terminological binary relationships, such as SYN (synonym of), BT (broader term) or NT (narrower term). ARTEMIS explores a thesaurus of terminological relationships properly strenghtened to evaluate affinity coefficients for schema elements, returning values between 0 and 1. Such values are computed for various source schemas elements based on their names (Name Affinity), as well as on their structure, in terms of cardinalities (Structural Affinity). Such kind of similarities among schema elements are combined into a Global Affinity value. A detailed description of the affinity evaluation process is given in [**?**, caandeca:tkde99]
- *Clustering of source schema elements*, to group elements by affinity values so that their different source schema representations can be analyzed for unification in the global schema. The clustering procedure builds clusters by merging elements based on their affinity values.
- *Construction of the global schema*, to build a unified representation for schema elements in a similarity cluster. The global schema obtained in this way is read-only, following conventional mediator-based architectures [6], and is used for query at the global level. The global schema is crisp: no information about similarities among source schema elements is represented in the global schema.

4.2 Fuzzy Global Schema

The global schema produced by ARTEMIS/MOMIS is crisp, although its definition process is thoroughly driven by similarities among schema elements, such as similarities among source relation names, among source attribute names or among source domains. We now show how to obtain a fuzzy global schema.

The extensions proposed to this purpose are the following:

- During the global schema construction phase, the information about the similarites existing among sources' schemas – gathered during the analysis phase and employed in the clustering phase – is stored in a fuzzy relation, called `SimCat`. Every attribute in the global schema is equipped with an attribute named μ. Since an instance of the global schema is a fuzzy relation, such attribute will store the membership degree of the corresponding global attribute value in the appropriate domain.
- The definition of mappings between source schemas and the global schema is greatly enhanced by exploiting the information contained in the `SimCat` fuzzy relation. Such information drives the population process of global attributes with fuzzy sets of values coming from source attributes' domains.

As well, the process of populating queries definition will help during the answering phase of a query posed against the global schema, where the information about source schemas' similarities can be relevant in the decomposition of the query over the most appropriate sources. In the remaining of this section we show how we modify the global schema construction phase in ARTEMIS/MOMIS to make explicit the information about similarities among global and source attributes gathered in the analysis phase. We assume that the analyis phase has identified different attributes in source schemas according to their similarities and then, in the clustering and construction phase, such source attributes are grouped and then a global attribute unifying them is created.

Let us consider the following example arising from the scenario described in Section 2: the different attributes `FullName`, `PatName` and `Name`, `Surname` all denote the patients' name. In this case, the similarity among attributes is very high. In the global schema such attributes can be represented by a single global attribute named, for example, `PatientName`. As another example, consider the attributes `DateResp` and `Response`. They are very different: in the former case the answer' release time is specified as an offset from the exam's date and, while in the latter case the information is represented at a finer level, being the day and week of the answer's release explicitly specified. In this case also, the global schema groups attributes `DateResp` and `Response` in a single global attribute named, for example, `ExamResponse`. The global schema is the following:

PatientExams

integer	char(50)	hh/dd/mm/yy	dd/mm/yy	integer	char(100)
Code	Name	ExamTime	ExamResponse	AdminCode	Reference

The information about similarities among global and local attributes is stored in a fuzzy relation named `SimCat`. The name comes from the fact that such fuzzy relation acts as a catalog containing metadata (attribute similarity, in this case) on the database. A portion of an instance of `SimCat` is the following:

SimCat

GlobalAttr	SourceAttr	μ
Name	FullName	0.8
Name	PatName	0.8
ExamTime	DateAdm	0.64
ExamTime	DateExam	0.9

Where `score` is a real number between 0 and 1 denotes the similarity between `SourceAttr` and `GlobalAttr`. Such values correspond to the name affinity values computed by ARTEMIS based on the terminological relationships holding between attribute names in the sources. For example, since an ypernimy relationship $\langle NameBTFullName \rangle$ holds, we have a similarity value equal to 0.8 for these two attribute names.

Thus the output of the extended construction phase is the fuzzy global schema and the fuzzy relation `SimCat`. The information stored in such relation will be used to express queries over the sources in order to adequately populate the global schema attributes. The different similarity scores of various source attributes play a relevant role in the definition of the queries over the source schemas. Consider the global schema attribute `ExamResponse`. Assume for clarity that it is defined starting from only two source schemas' attributes, namely `DateResp` of `PetLabRec` and `Response` of `RadLab`. Without any knowledge of the similarity scores of the corresponding source schema attribute types, a typical query over the source schemas populating the global attribute is:

```
Query 1:
SELECT  PetLabRec.DateResp
FROM    PetLabRec
UNION
SELECT  RadLab.Response
FROM    RadLab
```

Clearly, this query makes no distinction whether items satisfying it are coming from `Dateresp` domain or `Response` domain. However, as we already know, such domains are very different: the former specifies exam's release dates as the number of the days passed from the exam's day, while the latter is more precise and stores the same information telling explicitly the day and the week of the exam's release. In te next section, we show how to exploit the information about domains' similarities such that the items contained in the query answer (hence populating the global schema attribute) specify how much they satisfy the query. Intuitively, satisfaction degrees measure how much the corresponding items satisfy the query. Such degrees are usually represented with real numbers between 0 and 1, and they depend on how much the corresponding domain types are similar.

5 Schema Mapping Using Similarity

Aim of this section is to define the queries over the sources to populate the global schema. Such queries will be defined using the information about similarities among source and global schemas gathered during the analysis phase and stored during the construction phase. In this way, a global schema attribute will have as domain a fuzzy set whose corresponding membership degrees' elements reflect the similarity between the global schema attribute and the source schemas attibutes unified in such global attribute.

For example, consider again the global schema attribute `ExamResponse`. This time, we take into account the similarity scores of source and global attributes, and we will define an SQL-like query able to exploit such similarities. Furthermore, the user can specify how much the source data are relevant. For example, it is possible to express that data coming from `PetLabRec` are supposed to fit as twice as `RadLab` data. The relevances are expressed as weights ranging from 0 to 1. We employ a fuzzy-based extension of the classical relational model allowing the representation of similarity both at attribute value level and at tuple level. Such data model, along with its corresponding query algebra is presented in [4]. The corresponding extension to an SQL-like language introduced in [10] allows to model such similarities.

The following query populates the global attribute `ExamResponse` with a fuzzy relation whose tuples have values coming from the domains of sources' attributes `DateResp` and `Response`, and their membership degrees are computed taking into account both similarities μ_1, μ_2 with the global schema and the associated weights 0.66, 0.33, denoting their relevances. Such fuzzy tuples are formed joining the values coming from source domains and the corresponding similarity scores, stored in the fuzzy relation `SimCat`. We employ the `WITH WEIGHT` construct for expressing weights.

```
 Query 2:
SELECT  PetLabRec.DateResp, SimCat.μ
FROM    PetLabRec, SimCat
WHERE   GlobalAttr=‘‘ExamResponse’’ AND SourceAttr=‘‘DateResp’’ WITH WEIGHT 0.66
UNION
SELECT  RadLab.Response, SimCat.μ
FROM    RadLab, SimCat
WHERE   GlobalAttr=‘‘ExamResponse’’ AND SourceAttr=‘‘Response’’ WITH WEIGHT 0.33
```

The answer to such query is a set of pairs $\langle value, score \rangle$ forming a fuzzy relation according to the definition given in Section 3.

Answer to Query 2

ExamResponse	μ
10/03	0.7
12/04	0.7
3	0.5
1	0.5
2	0.5

It may happen that the answer of Query 5 is very large and most of the items contained in it have very low membership degrees. In order to retain in the answer only items having high membership score, we introduce the `HAVING RELEVANCE` construct that specifies a threshold on membership degrees' items in the answer.

```
 Query 3:
SELECT  PetLabRec.DateResp, SimCat.μ
FROM    PetLabRec, SimCat
WHERE   GlobalAttr=‘‘ExamResponse’’ AND SourceAttr=‘‘DateResp’’ WITH WEIGHT 0.66
UNION
SELECT  RadLab.Response, SimCat.μ
FROM    RadLab, SimCat
WHERE   GlobalAttr=‘‘ExamResponse’’ AND SourceAttr=‘‘Response’’ WITH WEIGHT 0.33
HAVING RELEVANCE ≥ 0.8
```

The algorithm – whose pseudo-code is shown in Figure 1 – takes as input source and global schemas attributes, as well as their similarities, (along with a list of weights $\theta_1, \ldots, \theta_n$ and a threshold α, representing respectively the relevance of each source schema having attributes unified in the global schema and the minimum similarity such source schema attributes must have with respect to the global schema attribute they unify in. The algorithm outputs,for every global schema attribute, a query populating it with a fuzzy set of tuples whose values belongs to source attribute domains. The fuzzy membership degrees of the tuples are determined by the similarities between source and global schema attributes, already computed in the ARTEMIS/MOMIS analysis phase.

The algorithm operates in detail as follows: for every global schema attribute $\mathtt{A}_i$ and for every source schema attribute $\mathtt{A}_j$, it is checked whether $\mathtt{A}_j$ is unified in $\mathtt{A}_i$.

Being this the case, a query asking for the values source attribute $\mathtt{A}_j$ and the associate similarity score μ (with respect to global attribute $\mathtt{A}_i$) from the `SimCat` fuzzy relation is issued. The values having similarity scores less than the threshold α are discarded.

Such a query is issued over every source attribute unified in the global attribute $\mathtt{A}_i$. Then, the answers of such queries posed are put toghether by means of a weighted union. The weights $\theta_1, \ldots, \theta_u$ denote the different relevances the user associates to the sources.

6 Conclusions

In this paper we have proposed a data integration extension that permits to define similarity-based mappings between source and global schemas and we have discussed such techniques in the framework of the ARTEMIS/MOMIS system. As further work, we plan to refine the computation of similarity between sources and the global schema (e.g. taking into account also the source domains) and to employ similarity-based query processing when answering queries posed over the global schema.

References

[1] D. Calvanese, D. Lembo, M. Lenzerini. Survey on methods for query rewriting and query answering using views. *Technical Report D1.R5, D2I MURST Project*, 2001.

[2] S. Castano, V. De Antonellis, S. De Capitani di Vimercati. Global Viewing of Heterogeneous Data Sources. *IEEE Trans. on Knowledge an Data Engineering*, 13(2), 2001.

Similarity Schema Mapping Algorithm
Input: source attributes, global attributes, their similarities and relevances, a list of weights $\theta_1, \ldots, \theta_n$ and a threshold α.
Output: a a weighted union of SQL-like queries whose answers are fuzzy sets populating global attributes.
begin
  for every global attribute $\mathtt{A}_i$
  begin
    for every source attribute $\mathtt{A}_j$
    begin
      if $\mathtt{A}_j$ is unified in $\mathtt{A}_i$
      begin
        $Q_i =$ `EXEC SQL`
          `SELECT S.`$\mathtt{A}_j$`, SimCat.`μ
          `FROM S, SimCat`
          `WHERE GlobalAttr=''`$\mathtt{A}_i$`'' AND SourceAttr=''`$\mathtt{A}_j$`''`
          `HAVING RELEVANCE` α
      end
    end
  Q_1 `WITH WEIGHT` θ_1 `UNION` Q_2 `WITH WEIGHT` θ_2 `UNION` $\cdots$ `UNION` Q_n `WITH WEIGHT` θ_n
  end
end

Figure 1: The similarity-based schema mapping algorithm

[3] S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano. Semantic integration of heterogeneous information sources. *DKE*, 36(3), 2001.

[4] P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. Imprecision and user preferences in multimedia queries: a generic algebraic approach. In *Proc. of Conf. on Foundations on Information and Knowledge Systems*, 2000.

[5] R. Fagin, E.Wimmers. Incorporating user preferences in multimedia queries. in *Proc Seventh Int'l Conf on Database Theory*, 1997.

[6] R. Hull. Managing Semantic Heterogeneity in Databases: a Theoretical Perspective. In *Proc. PODS Conf.*, 1997.

[7] G. Mihaila, T. Milo, A. Mendelzon. Querying the World Wide Web. *Journal of Digital Libraries*, 1(1): 54-67, 1997.

[8] R. J. Miller, L. Haas, M. A. Hernadéz. Schema Mapping as Query Discovery. In *Proc. VLDB Conf.*, 2000.

[9] G. Modica, A. Gal, H. M. Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proc. Coopis Conf.*, 2001.

[10] D. Montesi, A. Trombetta. An imprecision-based query language for the Web. *Journal of Visual Languages and Computing* 12, 3-35, 2001.

[11] S. Ram, V. Ramesh. Schema integration: past, current and future. In *Management of Heterogeneous and Autonomous Database Systems*, 1999.

[12] A. Trombetta. Representing and querying imprecise data. Ph.D. Thesis. Dept. of Computer Science, University of Torino, 2001.