Information Technology Journal 3 (1): 57-68, 2004
ISSN 1682-6027
© 2004 Asian Network for Scientific Information

A Schema Comparison Approach to Determine Semantic Similarity among Schema Elements

Nayyer Masood
COMSATS Institute of Information Technology, Wah Campus, Wah Cantt, Pakistan

Abstract: A semantic based schema comparison approach is presented that generates semantic similarity assertions among schema elements of the component database of a federated database system. The schema comparison approach is a part of a schema analysis methodology ECCAM. The approach uses the interpretation assigned to component schemas and generates ranked semantic similarity assertions. These assertions may be used by the integrator to develop an integrated schema.

Key words: Semantic heterogeneities, schema interpretation, semantic similarity, similarity assertions

INTRODUCTION

A federated database system (FDBS) supports different federations of schemas (Fig. 1b). Each federation is represented by a federated schema that is developed on the basis of requirements of different users' groups of the FDBS. A federated schema results from the process of schema integration (SI) in which the schema elements from export schemas are merged to form the federated schema. The export schemas are the subsets of the component schemas where as each component schema belongs to a component database of the FDBS[1]. Schema integration is a crucial process for establishing an FDBS; the most critical phase in SI is schema analysis in which elements from export schemas are compared with each other to identify the semantically similar elements, that is, the elements that model the same or similar concepts. The schema elements identified as semantically similar are then candidates to be merged into a federated schema.

An effective schema analysis approach should be based on the semantics of the component database schemas rather than their structures[2]. A number of SI approaches that are based on the semantic enrichment of the component schemas[3-9]. One such approach presented in Masood[10] is ECCAM (Extended Common-Concept based Analysis Methodology). The schema comparison approach presented in this paper is a part of ECCAM. In the following, ECCAM is presented briefly that will help to understand the schema comparison approach presented in the present research.

ECCAM-A recap: ECCAM is a semantic-based schema analysis methodology in which component schemas are compared with each other to identify the semantically similar elements. A pre-requisite to ECCAM is the development of a set of concept hierarchies[5] a form of ontology[11]. Ontology is an explicit specification of a conceptualization where a conceptualization is an abstract, simplified view of the world that we wish to represent for some purposes[12]. An ontology in ECCAM consists of general concepts that may exist in the common/global Universe of Discourse (UoD) being modeled within the component databases of an FDBS. Fig. 1a presents an example set of concept hierarchies for a university library system. ECCAM consists of three phases:

Schema interpretation: In this phase component schemas are assigned an interpretation

Schema comparison: The component schemas are compared with each other to generate similarity assertions between schema elements

Similarity analysis: The assertions generated in phase 2 are analyzed to identify the ones that are more probable to be true.

This research presents the last two of the above three phases of ECCAM, that is, schema comparison and similarity analysis. The first phase has been discussed in detail in Masood and Eaglestone[11] however as help in understanding the schema comparison approach the first phase and other necessary notions are discussed briefly:

Schema interpretation is the first phase of ECCAM[10,11]. In this phase each element from a component schema is mapped to the concept(s) that it models. The mapping process results in an interpretation of the entire schema. The same process is performed on

Corresponding Author: Muhammad Zaheer Aziz, Department of Computer Science, Faculty Block 2, International Islamic University, H-10, Islamabad, Pakistan

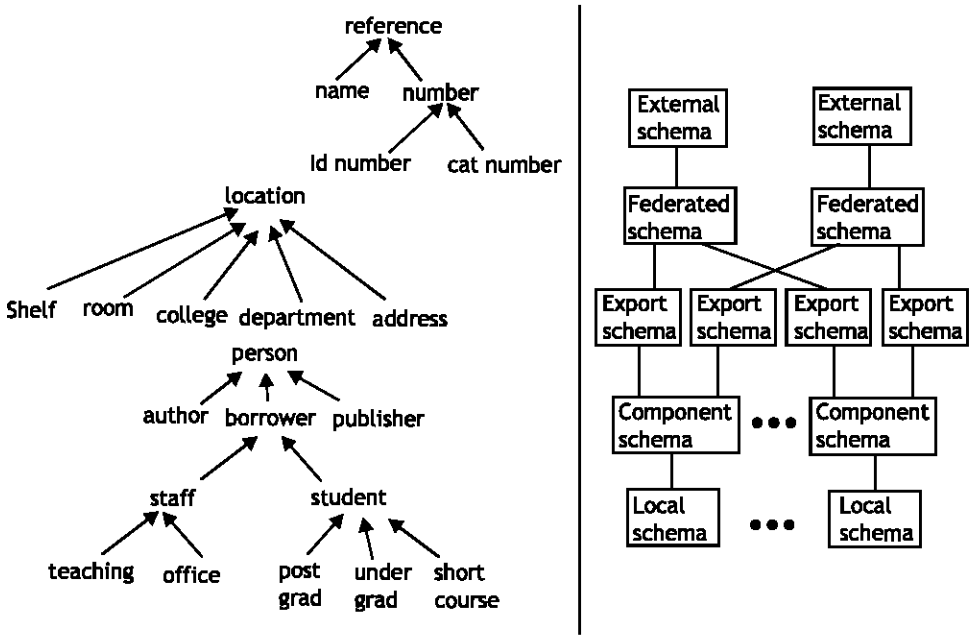

Fig. 1a: Example concept hierarchies (b) Federated database architecture

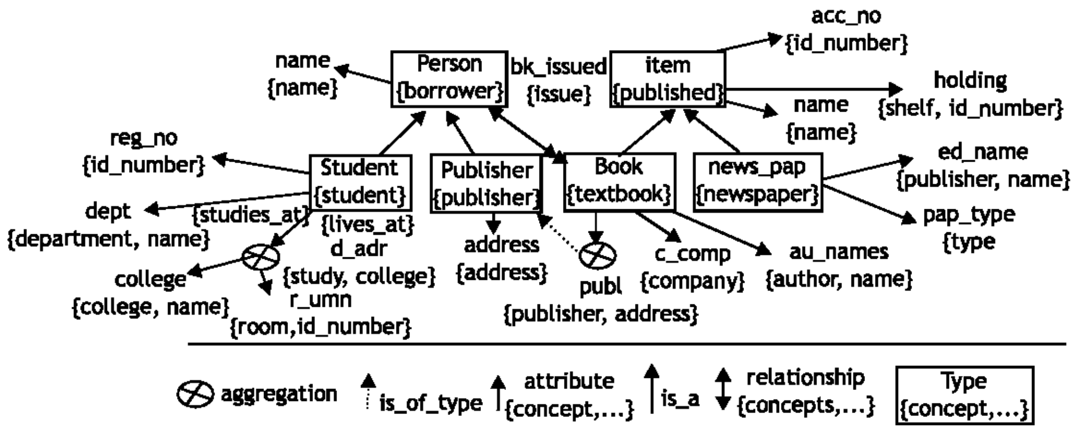

Fig. 2a: Concept model of a component schema

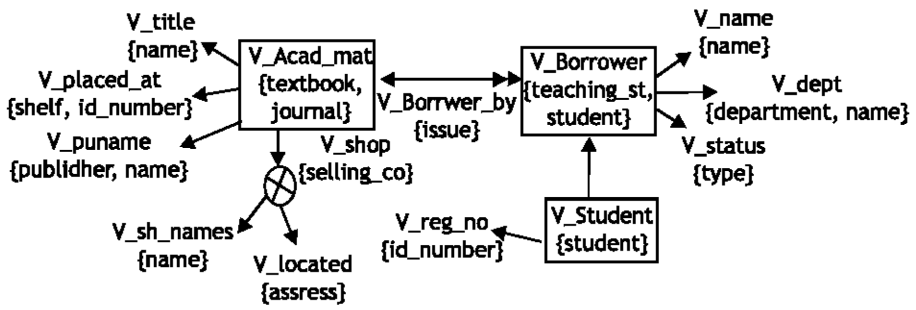

Fig. 2b: Concept model of a target federated schema

each component schema of the FDBS. The outcome of the schema interpretation phase is a concept model for each component schema of the FDBS. A concept model of a component schema assigns explicit semantics to each of the constituent elements of the schema individually as well as collectively. In addition to the component concept models, a concept model for each federated schema is also prepared called the federated concept model (FCM). An FCM consists of concepts and inter-concept relationships against the virtual schema elements that are to be included in a particular federated schema based. It is defined by the global DBA based on the requirements of a particular user group and it provides a context for identifying semantically similar elements from the component schemas to be integrated into the federated schema. The main advantages obtained by the concept models resulting from the schema interpretation phase can be listed as:

- better understanding of the schema semantics
- provide a basis for semantics oriented schema comparison
- make many of semantic heterogeneities (SHs) ineffective for the schema comparison phase
- provide a context for the schema comparison phase

Figure 2 shows concept models for example component and federated schemas.

Other important ideas discussed during schema interpretation phase and used in the later phases are given below:

Intrinsic and in-context semantics: The intrinsic semantics of a schema element is its meaning, i.e., the concepts that it denotes, independent of the context within which it is used and is represented by the function Int from the set of schema element names to the power set of concepts. Thus, the intrinsic semantic of a schema element O_i, is defined by

$\text{Int}\ (O_i)=\{c_i, I=1,m\}$ ------------ (f1)
where c_i., for i=1, ... , m are the concepts denoted by O_i

Further, the in-context semantics of an element are its more specific semantics within the contexts in which the element is defined within the schema. The in-context semantics of a schema element are determined by the concepts that it denotes, i.e., its intrinsic semantics and the contexts within which it is modeled. The contexts of a schema element are modeled within a schema by the structures in which the element is defined, which in turn denote Srs between the schema element and the structural elements to which it is related. A context (O_i, O_x) is an immediate context of O_i if O_i and O_x are linked with each other through a single SR, for example, in "Person has name" the "Person has" provides an immediate context for element "name".

The in-context semantics of a schema element Oi in context (Oi, Ox) is represented by concatenating the intrinsic meaning of the element Oi, the SR srelx,x-1 and the in-context meaning of the element O_{i+1} in context (O_{i+1}, O_x), i.e.,

$\text{ICMean}(O_i, O_x)=<\text{Int}(O_i)>\text{if } O_i=O_x$
Otherwise
$\text{ICMean}(O_i,O_x)=\text{ICMean}(O_{i+1,x})$? $\text{srel}_{x,x-1}$? $\text{Int}(O_i)$

where $O_x,O_{x-1},\ldots O_{i+1},O_i$ denotes the structural path from the in-context schema element O_x to the schema element O_i.

Shallow and deep semantic similarity: ECCAM establishes two types of semantic similarity among schema elements; shallow and deep semantic similarity[11,12]. Shallow similarity between a pair of schema elements is based on their respective intrinsic meanings only. Two schema elements (O_i, O_j) have shallow similarity between them if there are some concepts common among their intrinsic meanings. The existence of shallow similarity between two elements can be determined using the Int function (f1). Formally, I define shallow similarity as follows:

Two schema elements, O_i and O_j, are shallow similar if: $\text{Int}(O_i) \cap \text{Int}(O_j) \neq \varnothing$

Schema elements cannot be integrated on the basis of shallow similarity alone, because the same concept(s) may represent different real-world objects when viewed in different contexts. Eligibility of two schema elements to be integrated is therefore asserted on the basis of deep similarity (called semantic similarity in general) between them in ECCAM, which is defined as:

two schema elements are semantically similar if there is a correspondence between the concepts that they model when compared in a particular context.

Semantic similarity between a pair of elements O_i and O_j, with respect to the contexts, context(O_i, O_x) and context(O_j, O_y), exists if the following conditions hold:

1. $\text{Int}(O_i) \cap \text{Int}(O_j) \neq \varnothing$, i.e., O_i and O_j are shallow similar
2. if $(O_i \neq O_x)$ and $(O_j \neq O_y)$ then $\text{srel}_{i+1,i} \approx \text{srel}_{j+1,j}$ i.e., the corresponding SRs are compatible
3. O_{i+1} and O_{j+1} are semantically similar, respectively within context(O_{i+1},O_x) and context(O_{j+1},O_y)

where $O_x,O_{x-1},\ldots O_{i+1},O_i$ denotes the structural path from the context element O_x to the schema element O_i and $O_y,O_{y-1},\ldots O_{j+1},O_j$ denotes the structural path from the context element O_y to the schema element O_j.

Taxonomy of semantic similarity: Semantically Equivalent Schema Elements: Two schema elements, O_i and O_j, are said to be semantically equivalent if they model exactly the same concepts in a particular context.

Semantically related schema elements: Two schema elements (O_i, O_j) are semantically related if they have some concept(s) in common a particular context.

Semantically disjoint schema elements: According to proposed schema analysis methodology two schema elements (O_i, O_j) are semantically disjoint if there is no concept common between their intrinsic meanings

Contextually disjoint schema elements: This category includes those element pairs that are neither semantically equivalent, related nor disjoint. These are the pairs which are intrinsically shallow similar, but cannot be related because there is no correspondence between their respective contexts.

Schema comparison approach: The objective of schema comparison is to identify the schema elements from each component schema that are semantically similar to those required to be included in the target federated schema. The process is based on the comparison of the concept model of target federated schema (FCM) with the concept models for each of the component schemas one by one. The comparison process results in assertions showing the pairs of semantically similar schema elements represented within the concept models of the target federated schema and a component schema. The comparison process uses both the intrinsic and in context meanings of schema elements to compute two different types of similarities between them, called the shallow and the deep similarity. The sections to follow explain what I mean by these types of similarities and how they are computed.

Establishing shallow similarity: The strength of semantic similarity between O_i and O_j is computed on the basis of their respective intrinsic meanings. For computation purposes, the respective intrinsic meanings, $Int(O_i)$ and $Int(O_j)$ are represented as an n-component concept vectors[5] where n is the total number of concepts in ontology and CV(x), $1 \leq x \leq n$, denotes the xth component of CV, namely the xth concept. The association of a set of concepts with an attribute, say element O_i, can be represented by an n-component vector CV_{Oi}, where $CV_{Oi}(x)=0$ if concept C(x) is not associated with O_i; otherwise $CV_{Oi}(x)=1$. These concept vectors are later used to compute the semantic similarity between a pair of elements, say O_i and O_j. The concept vectors are compared using the sim function (f-2 below)[5] to compute shallow semantic similarity. This specific use of sim is defined as a function, Ssim (O_i,O_j), the value of which indicates the existence/non-existence and the strength of shallow similarity between O_i and O_j as explained below.

The detection and computation of shallow similarity consists of following two main steps:

Preparing concept vectors: n-component vectors, $CV_{Oj}(n)$, are created to represent this intrinsic semantics of each of the schema element, where n is the total number of concepts in the CMU and each component of CV represents a particular concept within CMU. The values of the vector elements are set to represent the mappings from the elements to the concepts they denote the ith element of a concept vectors ($CV_{Oj}(i)$) is set to 1 to indicate that concept ci is denoted by element Oj, otherwise it is set to 0.

Application of the sim function: After preparing the concept vectors for the both Oi and Oj, the strength of shallow similarity (Ssim(O_i,O_j)) is computed by applying the sim function, i.e.,

$$\mathrm{sim}(CV_{vol}, CV_{oj}) = \frac{CV_{voi} \bullet CV_{oj}}{\sqrt{|CV_{voi}| \, x \, |CV_{oj}|}}$$

where CV_{VOi} and CV_{Oj} are the concept vectors for the elements O_i and O_j, • is the dot product of two vectors and |Z| is the number of ones in a vector. Ssim(O_i,O_j) returns a value between 0 and 1 inclusive. A 0 denotes the absence of shallow similarity between (O_i, O_j), where as a non-zero value indicates not only the existence of the shallow similarity but also represents its strength. The greater the value of Ssim(O_i, O_j), the more the strength of shallow similarity between O_i and O_j. A 1 represents that intrinsic meanings of O_i and O_j are exactly the same, that is they model exactly the same concepts intrinsically.

The strength of shallow similarity is computed between every elements pairs (O_i, O_j), where O_i and O_j, respectively belong to the (target) federated schema and (a particular) component schema. The aim is to find component schema elements (O_j) that are shallow similar to those in the target federated schema. As an example, the process is applied on the federated and component concept models of sss 2. The process results in 53 shallow similarity assertions[11] some of them are (Fig. 3).

Shallow similarity plays two critical roles. One, it provides a basis for establishing deep similarity between schema elements and the second, it narrows the scope of search for the deep similarity between schema elements,

Ssim(O_i, O_j)	Concepts for O_i	Concepts for O_j
Ssim(V_Acad_mat, C_Item)=0.71	textbook, journal	newspaper, magazine, textbook, journal
Ssim(V_Acad_mat, C_Book)=0.71		textbook
Ssim(V_title, C_name) =1.00	name	name
Ssim(V_title, C_c_comp)=0.71		name, company
Ssim(V_title, C_name)=1.00		name
Ssim(V_placed_at, C_holding)=0.50	shelf, items	id_number, shelf
Ssim(V_puname, C_Person)=0.24	name, publisher	author, borrower, publisher, teaching_st, office_st, under_grad, short_course, research_pgs, taught_pgs
Ssim(V_puname, C_dept)=0.50		name, department
Ssim(V_puname, C_name)=0.71	name	
Ssim(V_shop, C_c_comp)=0.71	selling_co, name	name, company
Ssim(V_sh_name, C_name)=1.00		name
Ssim(V_Borrower, C_Person)=0.75	teaching_st, under_ grad, short_course,	author, borrower, publisher, teaching_st, office_st, under_grad, short_course, research_pgs, taught_pgs research_pgs, taught_pgs
Ssim(V_Borrower, C_Student)=0.89		under_grad, short_course, research_pgs, taught_pgs

Fig. 3: Ssim(O_i, O_j) for some of the elements from example FCM and the component concept model of Fig. 2 with corresponding concepts of O_i and O_j

since I define deep similarity only for elements which are shallow similar. The next section discusses the computation of deep similarity in detail.

Establishing deep similarity: This section discusses deep similarity; the type of similarity that acts as a basis for integrating schema elements within a federated schema.

Detection of deep similarity: Detection of deep similarity between schema elements is based on shallow similarity assertions. Two schema elements (O_i, O_j) can have deep similarity only if they have shallow similarity between them. Deep similarity between two shallowly similar schema elements, O_i and O_j, is asserted when a correspondence between a context of O_i and a context of O_j is established.

An exhaustive implementation of the above approach is not feasible, since it requires all the contexts of O_i to be compared with all contexts of O_j. In order to reduce the problem complexity, such that it becomes tractable, I have restricted the search space by testing only for correspondences between immediate contexts of the schema elements being compared. The procedure implemented for testing for deep similarity between two shallowly similar elements is therefore as follows:

Two shallow similar schema elements, (O_i,O_j), are deep similar if: there exists two immediate contexts, (O_x, $srel_{xi}$) and (O_y,$srel_{yj}$), such that:

O_x and O_y are shallow similar
the SRs, $srel_{xi}$ and $srel_{yj}$, are compatible.

For this purpose, each of the context element of O_i (O_x), is checked for a possible shallow similarity with all of the context elements of O_j (O_y). If the Ssim(O_x, O_y) is non-zero then this means that O_i is deep similar with O_j. The element pair (O_i, O_j) therefore qualifies for the computation of deep similarity in their respective contexts.

Formally, deep similarity between two shallow similar schema elements, O_i and O_j, holds if the following condition is true:

$$(\mathrm{Ssim}(O_i, O_j)>0) \text{ AND } (\mathrm{Ssim}(O_x, O_y)>0)$$

where O_x and O_y are the context elements of O_i and O_j, respectively.

The deep similarity is checked semi-automatically within ECCAM.

- If values of Ssim(O_i, O_j) and Ssim(O_x, O_y) are non-zero then ECCAM (automatically) asserts (O_i and O_j to be semantically similar)
- Each assertion of semantic similarity is presented to the integrator along with the SRs involved. The integrator then (manually) determines if the relevant SRs are compatible. If so, the integrator accepts the assertion as valid, otherwise it is rejected.

The deep similarity is checked for each virtual and component schema elements pair (O_i, O_j) which have a non-zero shallow similarity. Some of the elements (O_i, O_j), from the example concept models of Fig. 2, which are deep similar, their respective context schema elements, the SRs between them and the corresponding concepts (in italics)

Table 1: Some virtual and component schema elements having deep similarity

Schema elements with in-context elements, SRs between them and the concepts they model	
Target Federated Schema (Virtual) Elements	Component Schema Elements
V_Acad_mat, V_Acad_mat, self	C_Item, C_Item, self
textbook, journal	newspaper, magazine, textbook, journal
V_Acad_mat, V_Acad_mat, self	C_Book, C_Item, generalises
textbook, journal	textbook, newspaper, magazine, journal
V_title, V_Acad_mat, has	C_title, C_Item, has
name, textbook, journal	name, textbook, newspaper, magazine, journal
V_puname, V_Acad_mat, has	C_publ, C_Book, has
publisher, name, textbook, journal	publisher, address, textbook
V_Student, V_Borrower, generalises	C_Person, C_Person, self
under_grad, short_course,	author, publisher, teaching_st, office_st,
research_pgs, taught_pgs, teaching_st	under_grad, short_course,research_pgs, taught_pgs
V_Borrower, V_Borrower, self	C_Student, C_Person, generalises
under_grad, short_course,	under_grad, short_course,research_pgs, taught_pgs
research_pgs, taught_pgs, teaching_st	author, publisher, teaching_st, office_st

are shown in Table 1. The concepts denoting the intrinsic meanings of the schema elements are underlined.

Note that in the above table, SRs are described from the context element (O_x) to the element (O_i), for example, generalises (instead of is-a). So, the pairs for which deep similarity is detected are the ones for which the deep similarity is computed, as is discussed next.

Computation of deep similarity: After detection of deep similarity between a pair of schema elements (O_i, O_j), the next step is the computation of value of deep similarity and thus reveals the exact nature of semantic similarity between the pair. ECCAM uses the same function for computing the deep similarity value between elements O_i and O_j that is used for shallow similarity (f-2), however the concept vectors for the former are prepared by merging the concepts modeled by both the elements (being compared) and their respective in-context elements. Once the in-context concepts of the schema elements that have been detected as being deep similar are accumulated together, they can be used in the computation of deep similarity. For this purpose their concept vectors are prepared as for the computation of shallow similarity and the sim function (f-2) is applied.

The existence and details of the deep similarities detected and measured are presented to the integrator as a set of assertions. These assertion are defined by the function, Dsim, as follows:

$$Dsim\ (O_i, O_j)=\{<O_x, srel_{xi}, O_y, srel_{yj}, Sval, Dval, strn>\} \quad \text{(f-3)}$$

where O_i and O_j denote virtual and component schema elements;

O_x, O_y are the respective context schema elements that are related to O_i and O_j through SRs $srel_{xi}$ and $srel_{yj}$, respectively.

Sval and Dval represent the shallow and deep similarity values,

the final component "strn" denotes the strength of the assertion.

A deep similarity assertion states that a virtual element O_i in-context of O_x through $srel_{xi}$ is (deep) semantically similar to O_j in-context of O_y through $srel_{yj}$, the value of shallow similarity between the two is Sval and that of deep similarity is Dval and strn is the possibility of this assertion being valid. An example for the computation of deep similarity (Dval) follows.

Consider the elements V_puname and C_publ in the Table 1. These two elements have non_zero shallow similarity between them, that is,

Ssim(V_puname, C_publ)=0.5

So, they do qualify for the testing of deep similarity between them. Their respective in-context elements are V_Acad_mat and C_Book and

Ssim(V_Acad_mat, C_Book)=1.0

That is, the in-context elements of V_puname and C_publ also possess shallow similarity between them. To compute the deep similarity, sets of their respective in-context concepts are prepared as shown in Table 1, these concepts are used to prepare their concept vectors and finally sim function (f-2) is applied on these concept vectors and the following assertion is produced:

Dsim(V_puname, C_publ)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, strn>

The above assertion shows the value of deep similarity (Dval) between the elements V_puname and C_publ as 0.58.

Deep similarity is computed for all contexts of each of O_i and O_j, i.e., each element is compared within the context

Dsim(O_i, O_j)={<O_x, srel$_{ix}$, O_y, srel$_{jy}$, Sval, Dval, strn>}
1 Dsim (V_Acad_mat, C_Item)=<V_Acad_mat, self, C_Item, self, 0.71, 0.71, strn>
2 Dsim (V_Acad_mat, C_Book)=<V_Acad_mat, self, C_Item, generalises, 0.71, 0.71, strn>
3 Dsim (V_title, C_title)=<V_Acad_mat, has, C_Item, has, 1.00, 0.77, strn>
4 Dsim (V_title, C_au_names)=<V_Acad_mat, has, C_Book, has, 0.71, 0.67, strn>
5 Dsim (V_placed_at, C_acc_no)=<V_Acad_mat, has, C_Item, has, 0.71, 0.67, strn>
6 Dsim (V_placed_at, C_holding)=<V_Acad_mat, has, C_Item, has, 1.00, 0.82, strn>
7 Dsim (V_puname, C_Publisher)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, strn>
8 Dsim (V_puname, C_title)=<V_Acad_mat, has, C_Item, has, 0.71, 0.67, strn>
9 Dsim (V_puname, C_publ)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, strn>
10 Dsim (V_puname, C_au_names)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, strn>
11 Dsim (V_puname, C_name)=<V_Acad_mat, has, C_Book, has, 0.82, 0.75, strn>
12 Dsim (V_Borrower, C_Person)=<V_Borrower, self, C_Person, self, 0.79, 0.79, strn>
13 Dsim (V_Borrower, C_Student)=<V_Borrower, self, C_Person, generalises, 0.89, 0.79, strn>
14 Dsim (V_name, C_name)=<V_Borrower, has, C_Person, has, 1.00, 0.82, strn>
15 Dsim (V_name, C_dept)=<V_Borrower, has, C_Student, has, 0.71, 0.83, strn>
16 Dsim (V_dept, C_name)=<V_Borrower, has, C_Person, has, 0.71, 0.76, strn>
17 Dsim (V_dept, C_dept)=<V_Borrower, has, C_Student, has, 1.00, 0.93, strn>
18 Dsim (V_Student, C_Person)=<V_Borrower, generalises, C_Person, self, 0.71, 0.79, strn>
19 Dsim (V_Student, C_Student)=<V_Borrower, generalises, C_Person, generalises, 1.00, 0.79, strn>
20 Dsim (V_reg_no, C_reg_no)=<V_Student, has, C_Student, has, 1.00, 1.00, strn>

Fig. 4: Deep similarity assertions for example schemas

of each related element. The comparison process may therefore generate more than one assertion for a single virtual element, mainly due to SHs. Consequently, the integrator must identify those assertions that are valid/applicable. As a further help to the integrator, assertions are ranked according to their respective probability of validity.

Figure 4 illustrates the presentation of deep similarity information as sets of assertions. The deep similarity assertions shown in the Figure are generated for our example concept models of Fig. 2. However, note that the value of strn is still unspecified which will be explained later when I describe the process for ranking the assertions. Note also that schema elements which are not involved in an SR that establishes a context for them, e.g., V_Acad_mat and C_Item, are treated as being contexts for themselves with an arbitrary SR 'self' (as in assertion 1 in the Fig. 4).

The general format of an assertion is given at the start of Fig. 4, as a help to the reader in understanding different constituents of assertions.

The assertions given in Fig. 4 are generated by applying ECCAM to the case study concept models. Note that for some schema elements of the FCM, multiple assertions are generated, e.g., assertions 1 and 2 are about the same virtual schema element, i.e., V_Acad_mat. On the other hand for some virtual schema elements, no assertions are generated which signifies that these schema elements are not semantically similar to any of the elements in the particular component schema. However, a critical assumption behind this declaration is that all the phases of the proposed methodology have been performed properly and strictly as described. However, there is always room for human error. Specifically, the requirements that, (a) mapping is performed to the most specific concepts within a particular concept hierarchy and that (b) semantic similarity is required to be explicitly declared in the form of FCM, might cause the omission of possibly valid assertions in the following circumstances:

1) The methodology would not identify two schema elements (O_i, O_j) as semantically similar if one, say O_i, is mapped to concepts {c_i} and the other, say O_j is mapped to {c_j}, such that {c_j} is/are parent(s) of {c_i}.

For example, consider the Fig. 5 in which virtual and component schema elements in (b) and (c) are mapped to the concepts from the concept hierarchy given in (a). The attribute V_name is mapped to concept name (in b), where as the attributes initial and surname are mapped to concepts l_name and f_name (in c), respectively. The former type of mapping, that is to a non-terminal concept, is not recommended by ECCAM. So the attribute V_name will not be identified as being similar to either of the initial or surname.

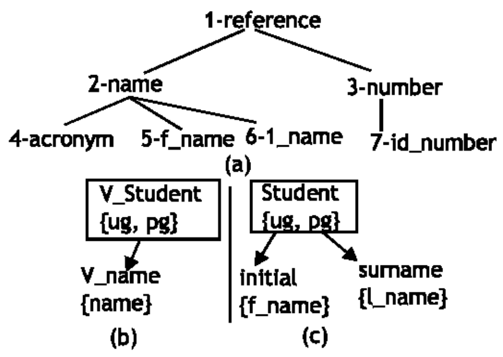

Fig. 5: Mapping to parent and child concepts (b and c) from concept hierarchy in (a)

In order to compensate for missing assertions due to such type of mapping, the detection phase is run again for the virtual schema elements for which no assertions have been generated (Fig. 6). This time the matching between concepts is searched among their respective generalizations. If such a match is found, the child concept(s) is/are set equal to the parent one(s) and similarity is computed between the schema elements in question. However, an assertion computed in this way is labeled, so that the integrator knows the basis on which it was generated.

To illustrate this generalized comparison of schema elements we again consider the example concept models and following assertion is generated:

21 Dsim(V_shop, C_c_comp)=<V_Acad_mat, has, C_Book, has, 1.00, 0.41, strn>

Fig. 6: Assertion generated by generalised mapping

2) The integrator may omit, by mistake, to map a particular schema element to one or more concepts that are actually being modelled by that element, or may forget to declare some schema elements as semantically similar in the FCM.

To illustrate this second source of assertion omission, consider Fig. 7, in which (a) contains an example concept hierarchy, (b) and (c) are example schema elements from virtual and component schemas, respectively. V_Mg (in b) denotes the concept magz (magazine), which is not represented in the component schema (in c), rather it models the concept book in Pub_mat. We can not therefore establish semantic similarity using the proposed methodology. However, in a case when there is no match found for V_Mg, or when the integrator, by mistake, may omit to mention the concepts book and magz as being semantically similar whereas they are in this particular context, it might be useful to declare (V_Mg, Pub_mat) as being ontologically similar.

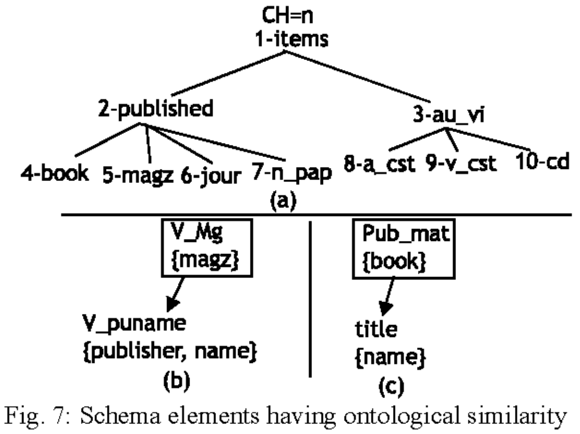

Fig. 7: Schema elements having ontological similarity

Such assertions are presented to the integrator as relatively less probable ones and they might be applicable as such or to prompt the integrator to correct his mistake or omissions.

Figure 8 contains ontological similarity assertions for the virtual schema elements from example FCM for which the proposed methodology does not generate assertions.

22 Dsim (V_sh_name, C_college)=<V_shop, has, C_d_adr, has, 0.71, 0.55, strn>
23 Dsim (V_located, C_address)=<V_shop, has, C_Publisher, has, 1.00, 0.45, strn>
24 Dsim (V_status, C_Student)=<V_Borrower, has, C_Person, generalises, 0.00, 0.80, strn>
25 Dsim (V_status, C_d_adr)=<V_Borrower, has, C_Student, has, 0.35, 0.75, strn>

Fig. 8: Ontological similarity assertions

If there are still some virtual schema elements for which assertions have not been generated, then for such elements assertions are generated based only on the shallow similarity (with 0 deep similarity). Such assertions are also useful since, (i) they inform the integrator that for these virtual elements no semantically similar component element could be found; and (ii) knowledge of shallow similarity might also be useful on its own. This situation is illustrated in the Fig. 9.

In this situation, the elements model similar concepts in exactly inverse fashion and there is no other relationship that could establish a commonality between them. So none of the above mentioned types of deep similarity could be computed in such a situation, so shallow similarity assertions are the best possible solution.

The detection and computation activities of the schema comparison process has been explained above. The comparison process is carried out with the objective of establishing semantic similarity (deep similarity) between schema elements. The methodology produces its best results if the mapping process has been performed properly[10,11]. Specially, the development of FCM is

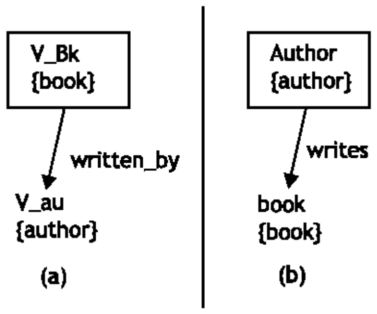

Fig. 9: Elements for which deep similarity cannot be computed

critical to the comparison process, since it establishes the context for the comparison. However, the proposed schema comparison method also include steps which compensate for some of the possible errors made during the mapping process as explained above.

Similarity analysis: The comparison process may generate many assertions for a single schema element, for different reasons, like the semantic heterogeneity among component schemas and the availability of different constructs to model the same real-world object. The final phase of the proposed schema analysis methodology is therefore designed to assist the integrator in validating the assertions generated. It does this by assigning a strength to each to indicate their likelihood of being valid. The heuristics applied for this purpose is as follows.

Ranking of assertions: The Schema Comparison phase of ECCAM may generate many assertions for individual virtual schema elements of the target federated schema. This is illustrated in the example assertions given in Fig. 8, 6 and 4. This is mainly a consequence of semantic heterogeneity between CDBSs. Multiple assertions for a single virtual schema element makes the task of the integrator more complex, since many of these will be invalid or irrelevant to the integration task. It is therefore beneficial to the integrator if these assertions are ranked according to their probability of being valid. The ranking process used in ECCAM assigns a strength to each assertion, that represents its likelihood of being valid. The strengths are presented to the integrator as part of the assertions generated by the Dsim function (f-3).

An assertion strength is a numeric value between 0 and 6:

- 0 denotes a (probably) false assertion;
- 1 to 6 represent valid assertions with decreasing possibility of validity.
- A strength of 1 (strongest) is assigned where there is a 'perfect' match, in which case other assertions involving the virtual and component schema elements in that particular assertion are overridden and assigned a 0 strength (invalid).
- Strength values 2 to 6 are assigned where the match is less strong; where as represent semantically related schema elements.

The assertion rankings are assigned by applying the following heuristics:

Perfect match heuristic: An assertion if semantic similarity between two schema elements, O_i and O_j, is

```
20   Dsim(V_reg_no, C_reg_no)=<V_Student, has, C_Student, has, 1.00,
     1.00, 1>
```

Fig. 10: An assertion showing semantically equivalent schema elements

assigned a strength of 1 (strongest), if the deep similarity value, Dsim in Dval(O_i, O_j), is equal to 1.

This happens only when Ssim(O_i, O_j) and Ssim(O_x O_y) are both equal to 1. Such an assertion indicates that the schema elements involved are semantically equivalent, i.e., they model exactly the same real-world concepts. All other deep similarity assertions involving both of O_i and O_j are then rejected, i.e., assigned a 0 strength (invalid).

Application of the perfect match heuristic has resulted following assertion in the assertions of Fig. 10 being assigned a ranking of 1, since this was the only assertion generated for V_reg_no, so there is no assertion with a zero ranking.

If the deep similarity strength between two elements, O_i and O_j is between 0 and 1, this indicates that the two schema elements are semantically related, that is, they have some concepts in common. However, because this is a weaker association than semantic equivalence, there may be many assertions generated on this basis which declare one virtual schema element to be semantically related to component schema elements. Multiple assertions are ranked by applying heuristics based upon their cause as follows:

Multiple assertions may be generated for a single schema element, O_i, as a consequence of an SH, where by the information modeled by O_i is collectively modeled by multiple component schema elements with a shared immediate context, which I call Horizontal Information Dispersion SH, which is illustrated in Fig. 11. Note that the information denoted by V_name (in (b)) is distributed (horizontally) between the initial and surname attributes (in (c)).

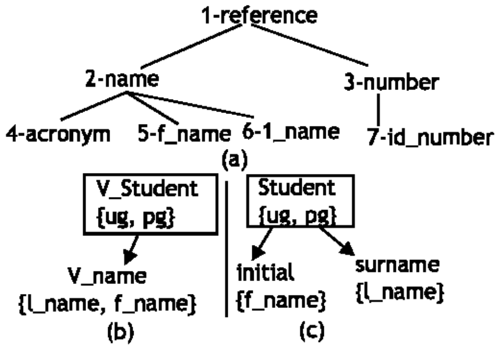

Fig. 11a: Concept hierarchy used for mapping the elements to (in b and c)

A horizontal information dispersion SH will cause assertions to be generated for each of the schema elements within the shared context. Multiple assertions for a single virtual schema element that are a consequence of an instance of this type of SH are ranked according to the following heuristic.

Horizontal information dispersion heuristic: Given a set of assertions of semantic similarity between a schema element O_i and elements which are a horizontal information dispersion of that element, O_1, O_2, ...O_m, the assertions are ranked as follows:

The assertions(s) with the maximum deep similarity value is/are assigned a strength of 2; others are assigned a weaker strength of 3. Example of assertions (Fig. 4) ranked by applying the horizontal information dispersion heuristic are given below:

1 Dsim (V_Acad_mat, C_Item)=<V_Acad_mat, self, C_Item, self, 0.71, 0.71, 2>
2 Dsim (V_Acad_mat, C_Book)=<V_Acad_mat, self, C_Item, generalises, 0.71, 0.71, 2>
3 Dsim (V_title, C_title)=<V_Acad_mat, has, C_Item, has, 1.00, 0.77, 2>
4 Dsim (V_title, C_au_names)=<V_Acad_mat, has, C_Book, has, 0.71, 0.67, 2>
5 Dsim (V_placed_at, C_acc_no)=<V_Acad_mat, has, C_Item, has, 0.71, 0.67, 3>
6 Dsim (V_placed_at, C_holding)=<V_Acad_mat, has, C_Item, has, 1.00, 0.82, 2>
7 Dsim (V_puname, C_Publisher)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, 3>
8 Dsim (V_puname, C_title)=<V_Acad_mat, has, C_Item, has, 0.71, 0.67, 2>
9 Dsim (V_puname, C_publ)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, 3>
10 Dsim (V_puname, C_au_names)=<V_Acad_mat, has, C_Book, has, 0.50, 0.58, 3>
11 Dsim (V_puname, C_name)=<V_Acad_mat, has, C_Book, has, 0.82, 0.75, 2>
12 Dsim (V_Borrower, C_Person)=<V_Borrower, self, C_Person, self, 0.79, 0.79, 2>
13 Dsim (V_Borrower, C_Student)=<V_Borrower, self, C_Person, generalises, 0.89, 0.79, 2>
14 Dsim (V_name, C_name)=<V_Borrower, has, C_Person, has, 1.00, 0.82, 2>
15 Dsim (V_name, C_dept)=<V_Borrower, has, C_Student, has, 0.71, 0.83, 2>
16 Dsim (V_dept, C_name)=<V_Borrower, has, C_Person, has, 0.71, 0.76, 2>
17 Dsim (V_dept, C_dept)=<V_Borrower, has, C_Student, has, 1.00, 0.93, 2>
18 Dsim (V_Student, C_Person)=<V_Borrower, generalises, C_Person, self, 0.71, 0.79, 3>
19 Dsim (V_Student, C_Student)=<V_Borrower, generalises, C_Person, generalises, 1.00, 0.79, 2>

Fig. 12: Assertions marked by heuristic 2

From the above example, note that:

- assertions 1 and 2 declare V_Acad_Mat to be deep similar to C_Item and C_Book and since both have the same Dval value and also have the same context schema element (C_Item), both are therefore assigned a strength 2.
- On the other hand assertions 5 and 6 have been assigned the strengths 3 and 2, respectively, since assertion 6 has the greater Dval value in the same context.
- Most of the multiple assertions for a same virtual schema element have been assigned the strength 2, since either the corresponding component schema elements have different context element or have the same Dval value in the same context.

The rankings of assertions that have been ranked 2 by the horizontal information dispersion heuristic are further refined on the basis of their shallow similarity values, since a strong shallow similarity value suggests a greater likelihood of validity. Accordingly, this third heuristic is applied:

Horizontal information distribution match heuristic: Given a set of assertions of semantic similarity between a schema element O_i and elements which are a horizontal information dispersion of that element, O_1, O_2, ...O_m and which have been ranked 2 by the horizontal information distribution heuristic:

- Assertion(s) ranking is unchanged (at 2) if the shallow similarity value is 1;
- Otherwise, assertion ranking are changed to a weaker strength of 3.

The application of horizontal information distribution match heuristic is illustrated in the following two assertions, 18 and 19, from Fig. 12:

18 Dsim (V_Student, C_Person)=<V_Borrower, generalises, C_Person, self, 0.71, 0.79, 3>
19 Dsim (V_Student, C_Student)=<V_Borrower, generalises, C_Person, generalises, 1.00, 0.79, 2>

Fig. 13: Assertions whose rankings are set by heuristic 3

Note, that these assertions declare V_Student as being semantically similar to two different component elements (C_Person and C_Student) in the same context (C_Person). However, the value of shallow similarity in assertion 19 is 1, so according to the above heuristic, 19 is maintained at ranking 2, where as assertion 18 is dropped to ranking 3.

ECCAM also asserts (weak) semantic similarity on the basis of similarity between concepts denoted by a schema element and their generalisations denoted by other elements. These assertions are ranked according to the following heuristic:

Generalization heuristic: Deep similarity assertions that are established by comparing the concepts to their generalizations are assigned a strength of 4.

Application of the generalisation heuristic is illustrated by the following assertion which has already been mentioned (without strn) in Fig. 6:

21 Dsim (V_shop, C_c_comp)=<V_Acad_mat, has, C_Book, has, 1.00, 0.41, 4>

Fig. 14: Assertion ranked according to heuristic 4

Note that, the elements V_shop and C_comp are mapped to the concepts selling_co and company, respectively. The latter concept is a generalisation of the former. This relationship among the corresponding concepts of two elements is represented by assigning a rank four to this assertion.

ECCAM also generates assertions based purely on ontological similarity, using the mapping proposed in Yu *et al.*[5]. This form of assertion is generated for (virtual) elements of the target federated schema for which no other assertions have been generated. These are given a weak ranking by the following heuristic:

Ontological similarity heuristic: Assertions based only upon ontological similarity are assigned a ranking of 5.

Application of the ontological similarity heuristic is illustrated by the following assertions:

22 Dsim (V_sh_name, C_college)=<V_shop, has, C_d_adr, has, 0.71, 0.55, 5>
23 Dsim (V_located, C_address)=<V_shop, has, C_Publisher, has, 1.00, 0.45, 5>
24 Dsim (V_status, C_Student)=<V_Borrower, has, C_Person, generalises, 0.00, 0.80, 5>
25 Dsim (V_status, C_d_adr)=<V_Borrower, has, C_Student, has, 0.35, 0.75, 5>

Fig. 15: Assertions ranked by heuristic 5

The similarity for the virtual elements in the above assertions can not be determined by adopting the direct mapping approach of ECCAM. So these assertions actually represent the ontological similarity and are ranked accordingly.

Finally, the virtual elements whose corresponding elements could still not be found are checked if they have shallow similarity with one or more component schema elements. If one or more shallow similarity assertions are found for such elements, then these elements are declared as contextually disjoint elements and such assertions are assigned a ranking 6. The example schemas do not contain any such elements so example is not quoted here.

Related research: Initially, schema comparison has been mainly considered as a part of the schema integration (SI) process, so we see different schema comparison approaches in SI approaches[3-9]. However two major factors highlighted the importance of schema comparison: (1) it proved that the most critical phase of SI is the schema comparison and success in SI mainly depends on a successful schema comparison approach and (2) the emergence and popularity of new areas in IT like E-Business, semantic query processing and data warehousing. Now we see number of projects concentrating solely on schema comparison (the term is more popular with the name "schema matching")[13-18]. Different approaches accept input schema in the form of relational, XML, ER or OO. They also differ in the level of schema elements that they match. For example, TranScm[13] matches at the element level, SKAT[15] matches between elements and structures as well. In this regard, the strength of schema comparison approach of ECCAM is that it is independent of any particular structure in establishing semantic similarity.

Another commonality among schema matching approach is that they generally assign a value between 0 and 1 value to represent the level of semantic similarity among schema elements. ECCAM also adopts the same approach but as a further help to integrator it ranks the semantic similarity assertions to rank the likelihood of the assertions of being valid.

This research presents the last two phases of ECCAM, schema comparison and similarity analysis. The former phase generates a set of semantic similarity assertions between the elements of target federated schema and a component schema. The similarity analysis phase assigns different ranks to these assertions that represent their likelihood of being valid. The utility of these ranked assertions is twofold. (i) elements identified from a particular component schema as being semantically similar to the virtual ones collectively form a 'first cut' export schema, which can then be validated by the integrator; (ii) the structure and integrity constraints of the virtual and component elements identified as semantically similar by assertions are analyzed to identify and resolve SHs, to enable integration into the federated schema.

The ECCAM further needs to be implemented on some more real life schemas to test its applicability. Moreover it can also be enhanced to perform the merging of the semantically similar elements semi-automatically.

REFERENCES

1. Sheth, A.P. and J.A. Larson, 1990. Federated Database Systems for Managing Distributed Heterogeneous and Autonomous Databases, ACM Computing Surveys, 22: 183-236.
2. Garcia-Solace, M., F. Saltor and M. Castellanos, 1996. Semantic Heterogeneity in Multidatabase Systems' Object-Oriented Multidatabase Systems A Solution for Advanced Applications, Bukhres, O.A. and A. Elmagarmid, chapter 5
3. Larson, J.A., Navathe, S.B. and R. Elmasri, 1989. A theory of Attribute Equivalence in Database with Application to Schema Integration, IEEE Transactions on Software Engineering, pp: 449-463.
4. Siegel, M. and S.E. Madnick, 1991. A Metadata Approach to Resolving Semantic Conflicts), 17th VLDB, Barcelona, pp: 133-145.
5. Yu, C., W. Sun, S. Dao and D. Keirsey, 1991. Determining Relationships Among Attributes for Interoperability of Multi-Databases Systems, Proceedings of the first International Workshop on Interoperability in Multidatabase Systems, IMS, Kyoto, Japan, pp: 251-257.
6. Fankhauser, P. and E.J. Neuhold, 1992. Knowledge based integration of heterogeneous databses, In proceddings of IFIP Conference DS-5 on Semantics of Interoperable Ddatabase Systems, Lome, Victoria, Australia.
7. Sheth, A.P., S.K. Gala and S.B. Navathe, 1993. On Automatic Reasoning for Schema Integration, Int. J. Intelligent Co-operative Info. Sys., 2: 23-50.
8. Kashyap, V. and A. Sheth, 1996. Semantic and Schematic Similarities between Database Objects: A Context Based approach, 22nd VLDB, Bombay, India.
9. Masood, N. and B. Eaglestone, 1998. Semantics Based Schema Analysis, Proceedings of 9th Int. Conf., DEXA'98, Vienna, Austria, LNCS 1460, pp: 80-89.
10. Masood, N., 1999. Semantics Based Schema Analysis, Ph.D. Dissertation, University of Bradford, UK.
11. Masood, N. and B. Eaglestone, 2002. Component and Federation Concept Models in a Federated Database System, accepted in Malaysian Journal of Computer Science, University of Malaya, Malaysia.
12. Gruber, T., 1993. A translation approach to portable ontology specification, Knowledge Acquisition, An Int. J. of Knowledge Acquisition for Knowledge-Based Systems, 5: 2.
13. Milo, T. and S. Zohar, 1999. Using schema matching to simplify heterogeneous data translation, Proc. Fusion, Sunnyvale, USA.
14. Palopoli, L., D. Sacca and D. Ursino, 1998. Semi-automatic semantic discovery of properties from database schema, In Proc Int Database Engineering and Applications Symp., IDEAS, IEEE Computer, pp: 244-253.
15. Mitra, P., G. Wiederhold and M. Kerson, 2000. A graph oriented model for articulation of ontology interdendencies, In Proc Extending database technologies, LNCS 1777, pp: 86-100.
16. Doan, A.H., P. Domingos and A. Levy, 2000. Learning source descriptions for data integration, In Proc Web-DB workshop, pp: 81-92.
17. Madahvan, J., Bernstein, P.A. and E. Rahm, 2001. Generic schema matching with Cupid, In Proc 27th Conf VLDB., pp: 49-58.
18. Yan, L., R.J. Miller, L.M. Haas and R. Fagin, 2001. Data-driven understanding and refinement of schema mappings, In Proc ACM SIGMOD conf., pp: 485-496.