Schema Integration: Solution of Naming and Data Type Conflicts

Adnan Shabbir
Comsat Institute Of Information
Technology Islamabad, Pakistan.
adnanshabbir_85m@yahoo.com

Maqbool Uddin Shaikh
Comsat Institute Of Information
Technology Islamabad, Pakistan.
maqboolshaikh@comsats.edu.pk

ABSTRACT

Schema integration has become a major field of research because of the global view of heterogeneous sources that it provides. In this paper an algorithm related to the schema integration is presented. There are many conflicts that occur during the process of schema integration. The algorithm handles naming and data type conflicts. The Web Ontology Language (OWL), which is accepted by IEEE as the standard is used to represent the source and global ontology for more affective and semantically correct matching. Attempt was made to focus on the two main characteristics Rule based and Learning based, which relates to schema integration. Authors have tried to use these characteristics in proposed algorithm. The motivation of this paper is, how to make mapping efficient and fast, which is semantically correct. The architecture of handling these conflicts along with the algorithms is explained with the help of case study.

Key Words: schema integration, mapping, semantically, rule based, learning based.

1. Introduction

The knowledge is expanding rapidly and because of the excessive use of the Internet, the semantic web has become a major field of research. Schema integration is one of the topics of semantic web, it provides the user an integrated view of multiple heterogeneous information sources. In this paper authors have tried to present an algorithm that solves naming and data type conflicts. Different techniques were studied and effort was made to use them in the proposed algorithm. How to make efficient and fast matching that is semantically correct is the benchmark for the proposed algorithm.

The rest of the paper is divided in five sections. Section 2 presents an overview of the existing and proposed work in the field of schema integration. Section 3 gives the detail description of proposed architecture. Section 4 gives a Case Study to give a proof of concept for our proposed architecture. Section 5 discusses the Architectural implementation and the output of the architecture. Section 6 gives conclusion and discuses some possible extensions.

2. Related Work

Schema Integration is used to provide a uniform access to multiple heterogeneous information sources. There are two types of integration, one is view integration and second is schema integration. View integration has nothing to do with schema. It merges the query result coming from different sources. On the other hand the schema integration deals at schema level.

Real life example is that consider a company which has different departments. The company wants to provide a global view to all of its departments. So that data could be accessed from different operational databases. Schema integration is defined as

"Schema integration is the process of identifying the components of a database which are related to one another, selecting the best representation for the global conceptual schema, and finally, integrating the components of each intermediate schema" [14].
There are many conflicts that occur during the process of schema integration, some of them are explained below.

2.1 Naming Conflict

It occurs when different names are used for the same attribute; for example homonyms and synonyms [12]. Different naming conventions are used by databases for objects. Semantic difference between elements should be recognized and mapped to common names [13].

2.2 Data Type Conflicts

These conflicts occur when different data types are used by two attributes. For example attribute ID could have data type string as well as integer [13].
There are many other conflicts like format conflict, structural conflict, missing data conflict, default value conflict, scale conflict, precision conflict and key conflict [13]. The proposed architecture is handling naming and data type conflict, which are explained above.

2.3 Rule Based Architectures

The term rule base describes whether the architecture of the schema integration process is rule based or not. A rule based system means that system will follow a set of predefined rules e.g. for every step hard coded rules were defined that will be followed by the system. Element names, data types, structures and sub elements can be used to define rules, for example two elements match if they have same name and same number of sub elements. Some rule based architectures are described below.
According to the TranScm system two elements are same, if they have same name and also have same number of sub elements. In the Dike system similarity of characteristics and similarity of related elements is considered. The Artemis and Momis system is use to measure the similarity factor, weighted sum of the similarities of name, data type and substructure is calculated. The Cupid system takes the name, data type and domain, i.e. Categorizes elements based on name, data types, and domains, and calculates a linguistic similarity coefficient to find similarity coefficient. Further details can be found in [9].

2.4 Knowledge Based Architectures

Knowledge based architectures refers to those architectures that follow the knowledge-based scheme for the schema integration process. In simple words, a knowledge base system has the ability to learn from the past experience. It is recommended to use both Rule and learner based techniques, to provide an effective matching solution. Authors have also tried to use both techniques for the proposed architecture. Some knowledge-based architectures are described below.
In the **Semint system**, two elements are matched against the attribute specification and statistics of data content. The **LSD system,** exploits the hierarchical nature of XML data, which is based on novel learning solution. It employs Naive Bayes over data instances. In the **IMAP system** elements are matched by analyzing the description of objects that are found in both schemas. The **Automatch and Autoplex** uses data instances to find similarity between schema's elements. Further details can be found in [9].

2.5 Literature Study

In the **Similarity factor Architecture,** the authors Thanh-Le Bach and Rose Dieng-Kuntz has proposed the system, which uses a similarity factor that is being calculated at the element, class and ontology level. While comparing the ontology a weight is being assigned to the each element, class and the

sum of weights is being assigned to the ontology [1].

Marcirio Silveira Chaves and Vera Lúcia Strube de Lima proposed **String Matching Based Architecture,** which uses similarity measuring technique called string matching, with two layers lexical and conceptual, to find out the similarity between two ontological structures terms, by finding out the minimum number of modifications which should be made in a string [2]. **Mediator-Wrapper Architecture** is proposed by Seksun Suwanmanee, Djamal Benslimane and Philippe Thiran. The main concepts of their architecture are *data sources:* which consists of structured data, *wrappers*: which serves as a mean of communication between local-system and *mediator:* a combination of ontology reasoning, query processor and integrated ontology [3]. **Data-Frames & Domain Snippets Architecture** proposed by David W. Embley, Li Xu, Yihong Ding. The system uses of data frames and domain ontology snippets and for the schema mapping the architecture uses a combination of matchers to improve the experimental results [4]. The **Hybrid Approach Architecture** is proposed by Ahmed Alasoud, Volker Haarslev and Nematollaah Shiri. The architecture is based on hybrid approach, which is a combination of data warehouse, and virtual approach, which inherits the advantages of both [5].

The **Ontology Bases Similarity Measure Architecture** by Farshad Hakimpour and Andreas Geppert is based on ontology for schema integration and resolving semantic heterogeneity in global schema, and also finds out all the meaningful mappings between the global schema and component schema [6]. In **Data Fusion Approach** the authors Jens Bleiholder and Felix Naumann have described the conflicts resolving strategies in an integrated information system in which their main concern is data integration process (data fusion). Data fusion resolves data inconsistencies, data contradictions and data uncertainties [7]. In **Matcher Based Approaches** the authors Erhard Rahm and Philip A. Bernstein have presented taxonomy for the automatic schema integration. Moreover they have presented a generic architecture of the match operator and have discussed different match operators that work at the schema, instance, element, language and structure level, and also some constraint base matchers [8].

In **Matching Techniques** the authors AnHai Doan and Alon Y. Halevy have surveyed the two matching techniques in schema integration. In schema matching they have discussed its rule based and learning based categories and have also discussed the architectural issues of the schema matching and incorporating the domain constraints in particular, and also different types of schema matching [9].

The **Ontology Integration Approach by** Zille Huma, resulted in finding certain similarities and differences in the schema matching and ontology mapping which they have mentioned in their paper. They have also described the ontology integration approaches. However the authors have not mentioned the conflict resolution strategies, which can encounter the conflicts that may rise using any of the integration technique [10].

3. Architecture Description

The proposed architecture follows the traditional approach in its design (input, processing and output). The inputs are Source Ontology (SO) and Global Ontology (GO). The *Concept and Attribute Matcher* does the processing. *Class and Attribute Change* files are the output of the architecture. Fig.1 describes the architecture diagrammatically. The architecture is using OWL (An ontology language, accepted by IEEE as the standard) for describing the schemas of Source and Global ontologies. There are many conflicts that occur in schema integration. Proposed architecture is handling naming and data type conflicts.

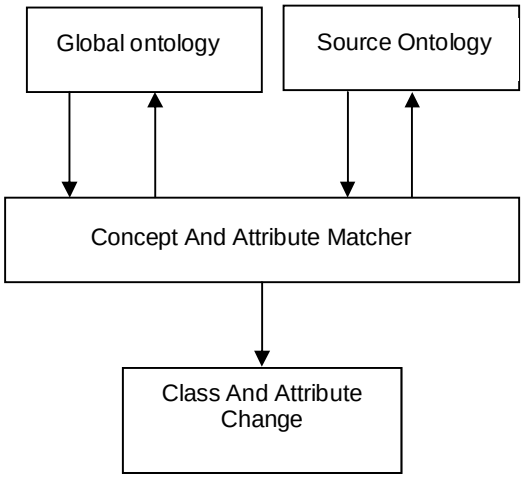


Figure 1. The Architecture Diagram

Authors have tried to use Rule based and Learning based techniques in the proposed work. Emphasize is made on, how to make mapping efficient and fast, which is semantically correct.

3.1 Global Ontology

This is the standardized ontology, which will be used to standardize the source ontology.

3.2 Source Ontology

Source ontology provides the source ontology schema, which requires to be standardized.

3.3 Concept and Attribute Matcher

It consists of two algorithms.

1. Element Match.
2. Attribute Match.

3.3.1 element match

It takes source ontology (SO) and global ontology (GO) as input. It selects the element of source ontology (SE), and matches it with first level elements of global ontology (GOL). If SE matched with the GOL then Attribute Matcher is referred. According to the **TranScm** two elements match, if they have same name and same number of sub elements. The Attribute Match is explained in the next section.

If SE does not match with the GOL then the Element Matcher will get source schema element's synonyms list (SOSL) from the built in dictionary. The dictionary also returns the parent class name of that element. This will help us to exploits the hierarical nature of ontology for efficient and semantically correct matching.

The algorithm then matches GOL with the SOSL. If any element of SOSL matches with the GOL, then Element Matcher will count the number of sub elements (classes) of both SE and element of GOL. If number of sub elements of SE and GOL element are same/equal then they are matched according to the definition of the system and Attribute Matcher is called. If SE and GOL element are not same/ equal then that element of GOL is selected from the global ontology as a root element. After that the children of selected element from GOL are selected as GOL (2^{nd} level elements). Then the step 5 is called again, to repeat the same procedure describe above. It will keep on repeating this procedure for each element of SO, till it gets the match both at naming level as well as at structural level e.g. till the elements have same name and same number of sub elements. That's how authors have used Knowledge Base technique e.g. instead of selecting all 2^{nd} level elements and compare with all of them, only those elements are selected, which belongs to SE. case study will help us to understand the functionality of Element Match in a better way, which is given in the next section.

3.3.2 attribute match

It gets the two elements, one from the source ontology (SE) and one from the Global ontology (GE). In the first step, it gets the attribute's list of both SE and GE, along with their data types. It gets one attribute from the SE Attribute list and matches it with the GE Attribute list. If attributes of SE and GE match then Attribute Matcher assigns SE Attribute the data type of the GE Attribute.

If attributes of SE and GE do not match then

Attribute Matcher gets the list of the synonyms of the SE Attribute (SOAL). Then all the attributes of SOAL are matched with the GE Attribute list one by one. The dictionary gives all the possible list of the synonyms of that attribute. Where SOAL and GE Attribute matches, then the Attribute Matcher assigns the name and data type of GE attribute to the SE attribute.

3.4 Class and Attribute Change

At the end of the algorithm the two files are generated by the architecture, one is known as Class Information File and the other is known as Attribute Information File.

3.4.1 class information

It contains the information of the class whose name was changed, e.g. old name and the new name of the class. Fig.8 shows a partial sample of Class Information File.

3.4.2 attribute information

It contains the information of the attribute whose name and data types are changed, e.g. old name, old type, new name and the new type of the attribute. Fig.8 shows a partial sample of Attribute Information File.

3.5 Proposed Algorithms for the Architecture

The following are the proposed algorithms for the architecture, which solves naming and data type conflicts. Naming conflict is solved by Element Match algorithm, which is shown in Fig.4. Data type conflict is solved by Attribute Match algorithm, which is shown in Fig.5.

General terms used in algorithm are:

SO = source ontology,

GO = global ontology.

SOSL = source ontology element's synonyms list that is obtain from built in dictionary.

GOSL = global ontology element's synonyms list that is obtain from built in dictionary.

GOL = global ontology level element list, i.e. it contains all the element of that level.

SE= Source Ontology Element

GE= Global Ontology Element

SOAL = source ontology element attribute's synonyms list that is obtain from built in dictionary.

4. Case Study

The following case study is used to explain the working of the proposed algorithms. The Global ontology (GO) and Source ontology (SO) used in this case study are shown in Figure 2 and Figure 3 respectively.

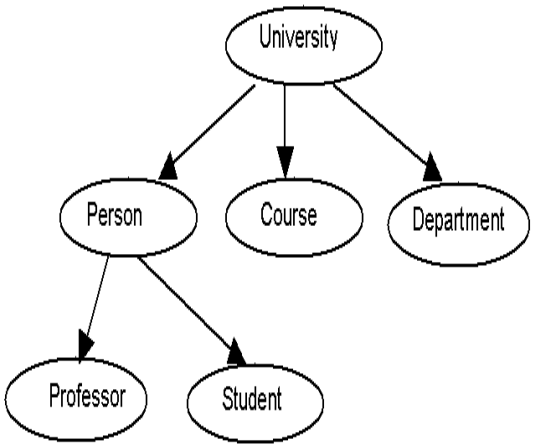


Figure 2. The Global Ontology

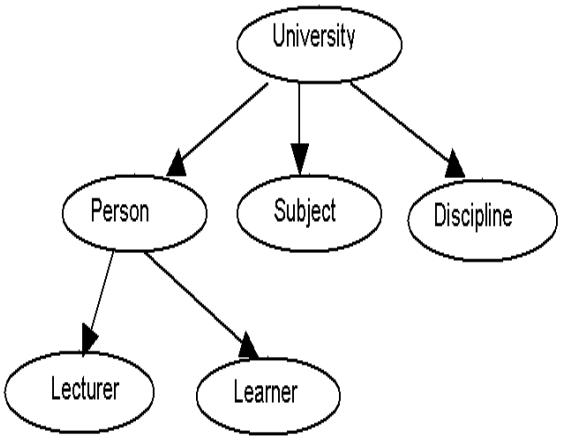


Figure 3. The Source Ontology

4.1 Working of Element Match Algorithm

The Element Match algorithm selects the element of source ontology (SE) and counts its sub elements. In this case the Learner is the SE and its sub elements are 0. The SE (Learner) is matched with first level elements of global ontology (GOL). In this case elements (Person, Course, Department) are selected in GOL as first level elements, as shown in Figure 6.

In the next step SE (Learner) is matched one by one with GOL. That is, Learner is matched with Person, then with Course and with the Department. If SE (Learner) matched with any of the GOL elements, then the number of sub elements of GOL element are counted. Then Element Match compares the sub elements of SE with GOL element. If number of sub element of SE and GOL element are equal/same, then both elements are same and Attribute Match is called. The parameters of Attribute Match are the name of the two elements. The functionality of Attribute Match is described in section 3.3.2.

But it can be seen that in this case no element from the GOL matches successfully with the Learner. So the synonyms list for source ontology element (Learner) SOSL is obtained from the Dictionary. For Learner the synonyms list SOSL will contain enrollee, student and person (generalized/parent class). After that each element of SOSL is compared with GOL element. For example enrollee is compared with Person, Course and Department, same procedure will be repeated for other two elements of (SOSL) i.e. student and person.

When the person (element of SOSL) is compared with GOL, person (element of SOSL) matches successfully with Person (element of GOL). As a result the number of sub elements of GOL element (Person) are counted. Then Element Match compares the sub elements of SE (Learner) with GOL element (Person).

But in this case, the number of sub elements of GOL element (Person) are 2 and number of sub elements of SE are 0. So SE (Learner) and GOL element (Person) are not equal in count of their sub elements. In that situation Person (GOL element) is selected as root node and it children are selected as next level element (GOL). In this way the unnecessary search by using rule base and knowledge base techniques is controlled. Instead of selecting all 2nd level elements and comparing with all of them, only those elements are selected that belongs to SE (Learner), which is Person in this case. So that it continues from Person till it find exact match as shown in Figure 7.

The GOL contains elements (Student and Professor) as second level elements. After this SE (Learner) is compared with GOL elements one by one. For example SE (Learner) is compared with Student and with Professor. No element from the GOL matches with SE (Learner). The synonyms list for source ontology element (Learner) SOSL is obtained from the Dictionary.

For Learner the synonyms list SOSL will contain enrollee, student and person (generalized/parent class). After that each element of SOSL is compared with GOL element. For example enrollee is compared with Student and with Professor, same procedure will be repeated for other two elements of (SOSL) i.e. student and person.

When the student (element of SOSL) is compared with GOL, student (element of SOSL) matches successfully with Student (element of GOL). As a result the number of sub elements of GOL element (Student) are counted, which are 0 in this case. Then Element Match compares the sub elements of SE (Learner) with GOL element (Student), which are also same, so they are same according to the **TranScm** system.

OldName Of The Element Is -->Learner
NewName Of The Element Is -->Student

This information is added in the list during the processing of the Element Match Algorithm.Once the SE and GOL elements matches, the Attribute Match algorithm is called.

```
Element Match (SO, GO)
{
1-Get the SO element.
2- Get the count of no of child/sub element of SO element.
3-Get the level1 element of the GO.
4-While (SO element! = null)
{
5-WHILE (GOL element! = null)
{
6-IF (SO element = = GOL element)
{
Get the count of children of GOL element.
IF (counts of no of child are same of GOL and SO element)
{
Attribute Matcher (SE, GE).
ADD to list.
SO element ->next.
}
ELSE
{
GOL element->next
}
IF (matches and no child exist throws exception) OR
IF (matches and in child no element matches throws exception)
}\\ Close of if
}    \\ Close of while
Get the list of synonyms of SO element. (if not found in the GOL)
7-WHILE (GOL element! = null)
{
8-WHILE (SOSL element! = null)
{

If (GOL element = = SOSL element)    \\match all one by one
{
 Get the count of children of GOL element.
 IF (counts of no of child are same of GOL and SO element)
 {
Attribute Matcher (SE, GE).
ADD to list.
SO element ->next.
 }
ELSE
{
Get the sub element list  of GOL element=GOL.
Goto step 5.
}
IF (matches and no child exist throws exception) OR
IF (matches and in child no element matches throws exception)
}
ELSE
{
SOSL -> next.
}}
GOL  -> next.
}
}
Return list.
}
```

Figure 4. Element Match Algorithm

```
Attribute Match (SE, GE)
{
Get the Attribute list and Datatype list, of both GE and SE Boolean Flag = = False.
While (SE Attribute list! = null)
{ \\source element list containing (attributes and data types)

WHILE (GE Attribute list! = null)
{ \\Global element list containing (attributes and data types)

IF (SE Attribute = = GE Attribute)
{
Flag = = True;
Get the type of SE Attribute.
Get the type of GE Attribute.
IF (SE Attribute->type = = GE Attribute->type)
{
ADD to list.
}
ELSE
{
SE Attribute->type = = GE Attribute->type;
ADD to list.
}}          \\ Close of if
ELSE
{
GE Attribute list->next;
}}                          \\ Close of while
IF (Flag = = False) \\ no match occur
{
Get the list of synonyms of SE Attribute = SOAL.
WHILE (GE Attribute list! = null)
{
WHILE (SOAL! = null)
{
IF (SOAL Attribute  = = GE Attribute)
{
Flag = = True;
Get the type of SE Attribute.
Get the type of GE Attribute.
IF (SE Attribute->type = = GE Attribute->type)
{
ADD to list.
}
ELSE
{
SE Attribute->type = = GE Attribute->type;
ADD to list.
}
}
ELSE
{
SOAL-> next;
}
}}          \\ Close of while
GE Attribute list-> next;
} \\ Close of upper while
}
SE Attribute list-> next;
}           \\ Close of up most while
Return list.
}
```

Figure 5. Attribute Match Algorithm

The parameters of Attribute Match algorithm are SE (Learner) and GOL element (Student). Figure 6 and 7 describes the Learner matching criteria.

The next source ontology element (SE) is Subject, the number of sub elements are 0. SE (Subject) is matched one by one with GOL. That is Subject is matched with Person, then with Course and with the Department. If SE (Subject) matched with any GOL element, then the number of sub elements of GOL element are counted.

Then Element Match compares the sub elements of SE with GOL element. If number of sub element of SE and GOL element are equal/same, then both elements are same and Attribute Match is called. The parameters of Attribute Match are the name of two elements. The functionality of Attribute Match is described in section 3.3.2. But it can be seen that no element from the GOL matches successfully with the Subject. So the synonyms list for source ontology element (Subject) SOSL is obtained from the Dictionary.

For Subject the synonyms list SOSL will contain chapter, course and theme. After that each element of SOSL is compared with GOL element.

For example chapter is compared with Person, Course and Department, same procedure will be repeated for other two elements of (SOSL) i.e. course and theme. When the course (element of SOSL) is compared with GOL, course (element of SOSL) matches successfully with Course (element of GOL).

As a result the number of sub elements of GOL element (Course) are counted. Then Element Match compares the sub elements of SE (Subject) with GOL element (Course), which are also same.

OldName Of The Element Is -->Subject
NewName Of The Element Is -->Course

So this information is added to list. The next source ontology element (SE) is Lecture, whose matching criterion is same as Learner.

The last element of source ontology is Discipline, same procedure will be followed for it as for Subject.

4.2 Working of Attribute Match Algorithm

Attribute Match Algorithm functionality is explained in this section. Consider the example when source ontology element SE (Learner) and global ontology element GE (Student) matches successfully and Attribute Match is called. The parameters of Attribute Match are SE (Learner) and GE (Student).

The Attribute Match gets the attribute's list of both SE (Learner) and GE (Student) along with their data types.

SE Attribute list (SEAL) and GE Attribute list (GEAL) represents the attribute's list of SE (Learner) and GE (Student) respectively. In this case SEAL contain ((name, varchar), (registration no, integer), and GEAL contain ((Name, String), (Id No, String)).

In the next step SEAL is matched one by one with GEAL. That is Learner attributes are compared with the Student attributes. The first attribute to be compared is (registration no) of SEAL, (registration no) is compared with Name and Id No of GEAL. If (registration no) of SEAL matches with any GEAL attribute, then their data types are checked. If the data types are same then they are added to the list.

If data types are different then the data type of GEAL attribute is assigned to SEAL attribute. As the queries are case sensitive, therefore the terms need to be standardized in same format and spell. So with this idea, it can be seen that no attribute of SEAL matches with GEAL.

So the synonyms list for SEAL attribute (registration no) SOAL is obtained from the Dictionary. For name the synonyms list SOAL will contain (Id No, Roll No and Serial No). After that each element of SOAL is compared with GEAL.

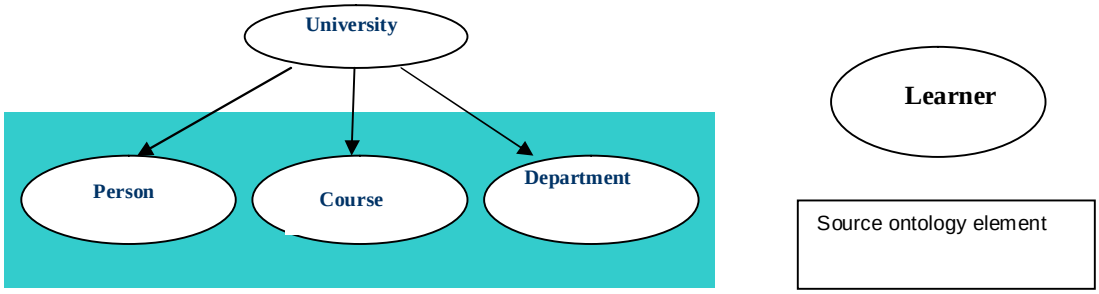


Figure 6. Global ontology first level nodes

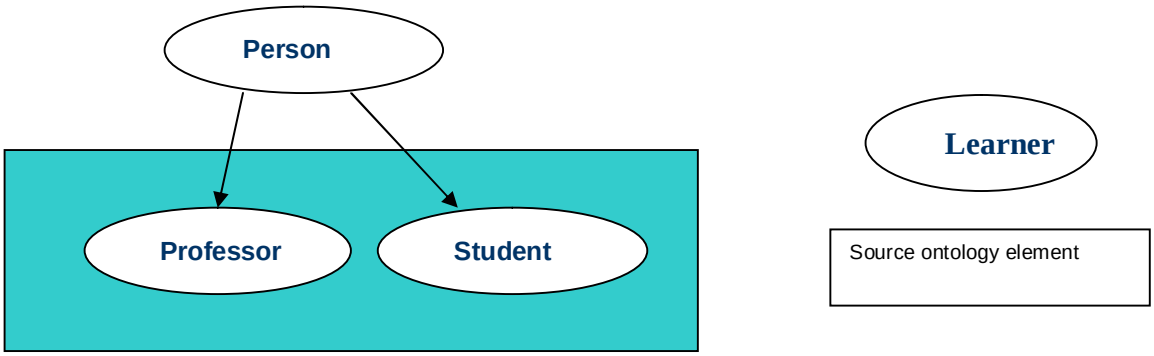


Figure 7. Global Ontology Second level nodes

For example Id No is compared with Name and Id No, same procedure will be repeated for other two elements of SOAL. When the Id No (attribute of SOAL) is compared with GEAL, Id No (attribute of SOAL) matches successfully with Id No (attribute of GEAL). As a result the data types of both SEAL and GEAL attributes are checked, if data types are same than the information is added into the list. If data types of both SEAL and GEAL attributes are not same than the data type of GEAL attribute is assigned to SEAL attribute. The information shown below is added into the list.

Attribute's OldName --> registration no
Attribute's OldType -->integer
Attribute's NewName -->Id No
Attribute's NewType -->String

Same procedure will be followed for other attributes of SEAL.

5. Architecture Implementation

This architecture has been implemented in order to evaluate its working. The architecture works successfully and produces the desirable results. Protégé is use to develop the Source and Global ontologies, which is a free, open source ontology editor. Java is used as the developing tool/language. The Jena API of java is used to load the ontologies. The *Concept and Attribute Matcher* can starts functioning immediately after loading the Source and Global ontologies.

Authors have built in their own dictionary in order to get the synonyms list of the source ontology elements. The not only returns the synonyms list, it also returns the Parent/generalized class of the concept. The information about Element name of the Source ontology is added in the list, during the processing of Element Match Algorithm. The information about attribute name and data type of the Source ontology element is

also added in the list during the processing of the Attribute Match Algorithm. This procedure is repeated for all the elements of the source ontology.

The architecture produces the hard coded files known as Class Information File and Attribute Information File. Class Information File contains the information about element of the source ontology. The attributes of Class Information File are OldName and NewName of the source ontology element. The Attribute Information File contains the information about attributes and data types of the source ontology elements. The attributes of Attribute Information File are OldName, OldType, NewName, and NewType of the source ontology element's attributes and data types.

The sample outputs that proposed architecture produces at the exit are shown in Fig.8. Due to space limitation, only some part of Class Information File and Attribute Information File is presented in Fig.8. The execution of the proposed architecture is shown in Fig.9.

The proposed architecture is supporting the wrapper/mediator framework. The functionality of wrapper/mediator framework could be define as user makes queries over global schema, mediated schema and mediator translates global schema query and reformulates it into sub-queries of local schemas [3]. So as the query reached query engine, then the mediator can reformulates it into sub queries of local schemas with the help of the document maintained (Class Information File, Attribute Information File). This is the requirement for the architecture's side, that information document is maintained that should provide help to query the system during the process of schema integration.

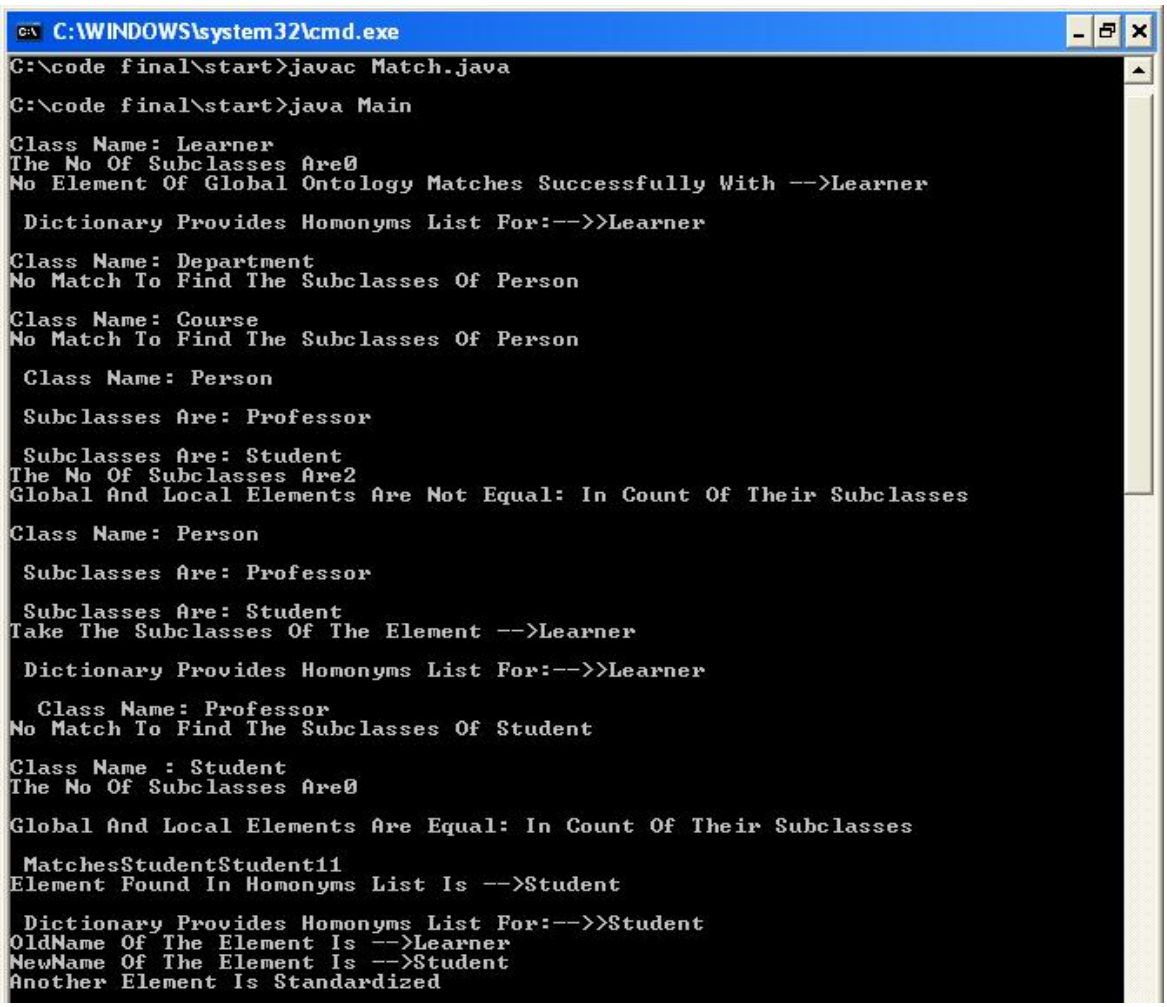
```
<?xml version= 1.0 encoding= ISO-8859-1 ?>
<ClassInformation>
<Class>
<Oldname>Discipline</Oldname>
<Newname>Department</Newname>
</Class>
<Class>
<Oldname>Subject</Oldname>
<Newname>Course</Newname>
</Class>
<Class>
<Oldname>Lecturer</Oldname>
<Newname>Professor</Newname>
</Class>
<Class>
<Oldname>Learner</Oldname>
<Newname>Student</Newname>
</Class>
</ClassInformation>
```

Class Information File

```
<?xml version= 1.0 encoding= ISO-8859-1 ?>
<AttributeInformation>
<Attribute>
<Oldname>discipline-name</Oldname>
<Oldtype>string</Oldtype>
<Newname>Departname</Newname>
<Newtype>string</Newtype>
</Attribute>
<Attribute>
<Oldname>Id</Oldname>
<Oldtype>int</Oldtype>
<Newname>DepartmentId</Newname>
<Newtype>string</Newtype>
</Attribute>
<Attribute>
<Oldname>facultyno</Oldname>
<Oldtype>int</Oldtype>
<Newname>FacultyNumber</Newname>
<Newtype>string</Newtype>
</Attribute>
<Attribute>
</AttributeInformation>
```

Attribute Information File

Figure 8. Class And Attribute Information File In XML Format.



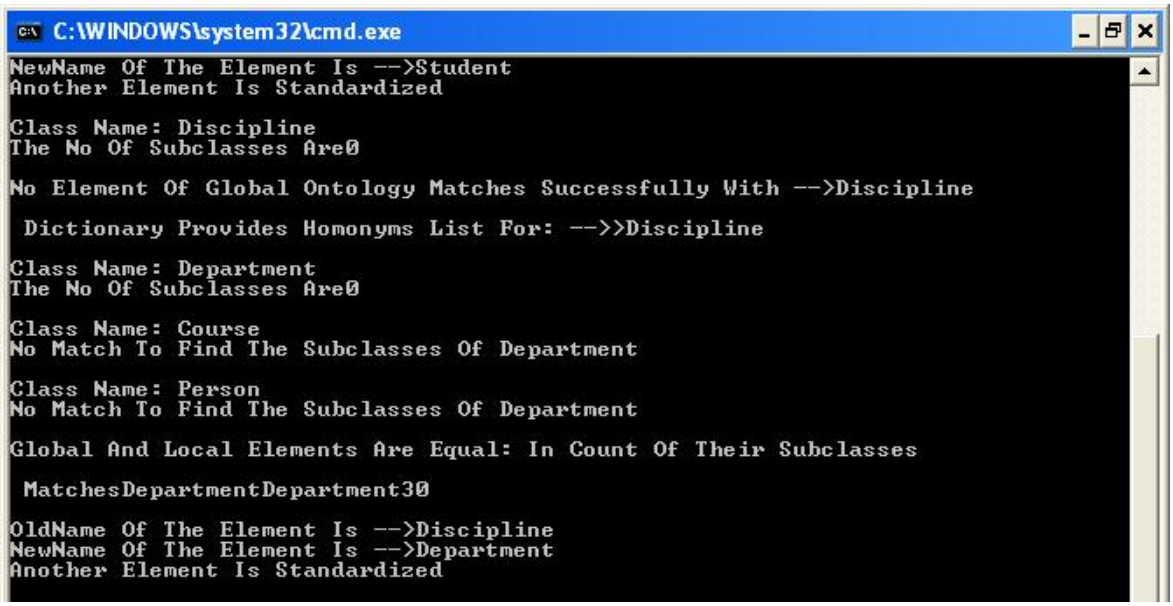


Figure 9. Execution of the Proposed Architecture.

6. Conclusions

In this paper the authors have proposed architecture for schema integration (in context of naming and data type conflict). They have tried to prove that, if source ontology is standardized then schema integration becomes an easy task. The standardization of source ontology that is semantically correct is a difficult task. Still many improvements can be made in the proposed algorithm.

Especially in the matching of two elements, more knowledge and rule base techniques could be added. Further if there is a dictionary that can return list of synonyms to any word, then it becomes more flexible and easy approach in practical life.

In future work the authors will try to suggest solution for structural conflict and add it in the proposed architecture, so that more correct matching could be made.

References

[1] Thanh-Le Bach, Rose Dieng-Kuntz **"Measuring Similarity of Elements in OWL DL Ontologies"** ACIA Project, INRIA Sophia Antipolis 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis, France.

[2] Marcirio Silveira Chaves, Vera Lúcia Strube de Lima "**Looking for Similarity among Ontological Structures**" *"in proceedings at Chaves_Lima: DIUL03, 2003"*

[3] Seksun Suwanmanee, Djamal Benslimane, Philippe Thiran "**OWL-Based Approach for Semantic Interoperability**" *"19th IEEE International Conference on Advanced Information Networking and Applications (AINA 2005)"*

[4] David W. Embley (Brigham Young University), Li Xu (University of Arizona South), Yihong Ding (Brigham Young University) "**Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned**", "Proceedings of the 3rd International Conference on Information Systems Technology and its Applications, Salt Lake City, Utah, 15-17 July 2004, 123-136"

[5] Ahmed Alasoud, Volker Haarslev, Nematollaah Shiri "**A Hybrid Approach for Ontology Integration**" *Proceedings of the 2005 VLDB Workshop on Ontologies-based techniques for DataBases and Information Systems (ODBIS-2005), Trondheim, Norway, Sept. 2, 2005, pp. 18-23*

[6] Farshad Hakimpour, Andreas Geppert "**Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach**" *In the proceedings of the International Conference on Formal Ontology in Information Systems FOIS-2001*

[7] Jens Bleiholder and Felix Naumann **"Conflict Handling Strategies in an Integrated Information System",** *"WWW Workshop in Information Integration on the Web (IIWeb) 2006, Edinburgh, UK."*

[8] Erhard Rahm, Philip A. Bernstein "**A survey of approaches to automatic schema matching**" S*pringer-Verlag 2001*.

[9] AnHai Doan, Alon Y. Halevy "**Semantic Integration Research in the Database Community: A Brief Survey**", *"In Proc. of 8th ACM SIGKDD Int."*

[10] Sven Abels, Liane Haak, Axel Hahn "**Identification of**

Common Methods Used for Ontology Integration Tasks", "*In: Proceedings of the first international ACM workshop on Interoperability of Heterogeneous Information Systems (IHIS05), CIKM conference proceedings. ACM, Sheridan publishing, 2005*"

[11] Zille Huma, Muhammad Jaffar-Ur-Rehman, Nadeem Iftikhar, **"An Ontology-Based Framework for Semi-Automatic Schema Integration"** "In Computer, Science & Technology., Nov. 2005, Vol.20. No. 6".

[12] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hubner (University of Bermen) **"Ontology Based Integration of Information, A Survey Of Existing Approaches"**, in proceedings of IJCAI-01 Workshop: ontology and information sharing, Seattle, WA, 2001,Vol.pp. 108-117.

[13] Qiang Liu, Tao Huang, Shao-Hua Liu, Hua Zhong (Chinese Academy of Sciences) **"An Ontology Based Approach for Semantic Conflict Resolution in Database Integration ", IBIS – Issue 2 (2), 2006.**

[14] Schema Integration, http://www.google.com/search?hl=en&q=Schema+Integration&btnG=Search, 2007 March 10th.