Research Article **Open Access**

Reza Asgari, Milad Gholipoor Moghadam, Mehregan Mahdavi*, and Aida Erfanian

An ontology-based approach for integrating heterogeneous databases

DOI 10.1515/comp-2015-0002

Received June 19, 2011; accepted July 20, 2015

Abstract: Integrating heterogeneous data in distributed databases has been a research issue for many years. In this paper, we discuss some of these problems and propose a solution using a semantic model. This semantic model is built upon the semantic relationships between existing data. Applying these semantics enables us to take into account different dimensions of user queries and find the best possible answer for them. The proposed approach leads us to introducing "a common language that is understandable for all databases". We use such a common language in order to return an effective response to the user's query, as well as reducing the problems of integration.

Keywords: data integration; ontology; data heterogeneity; distributed databases

1 Introduction

Nowadays, everyone can realize that the amount of available information and the number of information resources are increasing. Many of these resources are in the form of databases. A group of related data sources may form a heterogeneous database. Users of such heterogeneous databases need a common framework that is understandable by all of them [1]. From this point of view, the user feels a need for bi-directed relationships among these databases and the concepts within them. Generally, design and implementation of separated and heterogeneous databases cause many problems and challenges during the communication phase of these systems [2, 3].

In the last two decades, there were many approaches to solve the semantic inconsistencies among the relational databases. Although, all of these approaches were used to solve the inconsistencies in different levels of data, schema and application, they had a common problem which is overlooking the issue of lacking of semantic relationships among the databases [2]. Since, some databases share the same fields but with different titles, in many cases, we have to find a way to define semantics for such fields, so that the required information can be retrieved.

Ontology is one way for addressing the problem. Ontologies play an important role in information exchange. They can be used for the development and enrichment of databases semantically. These ontologies are used as a common means of understanding a domain. This facilitates the communication between the users and the heterogeneous distributed systems. Ontology is a conceptual model that forms the actual entities and their relations in a specific domain, explicitly and formally [1].

Our proposed approach to the integration problem is to use ontology as a standard facility for all parts of a system. We try to reduce the heterogeneity as much as possible with keeping the costs low and the framework flexible in order to adapt it to the new circumstances it confronts.

This paper is organized as follows: In Section 2, we discuss the problems related to heterogeneous distributed databases and present existing solutions. Our proposed approach is introduced in Section 3. The proposed method is further analyzed and discussed in Section 3.4. Finally, Section 5 concludes the paper.

2 Related works

There are many theoretical obstacles in the real world which prevent the effective integration of heterogeneous distributed database [4]. This means that different varieties are used in the schemas, naming conventions, and abstract levels of the data that are semantically the same. Nevertheless, there exist other theoretical challenges such as optimization of elements in query translation and ex-

Reza Asgari, Milad Gholipoor Moghadam: Department of Computer Science and Engineering, University of Guilan, Iran, E-mail: rezaasgari.68@gmail.com

***Corresponding Author: Mehregan Mahdavi:** Department of Computer Science and Engineering, University of Guilan, Iran, E-mail: mehregan.mahdavi@gmail.com

***Corresponding Author: Mehregan Mahdavi:** School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia

Aida Erfanian: Department of Computer Engineering, Azad University-Lahijan Branch, Guilan, Iran, E-mail: erfanian@iau-lahijan.ac.ir

ecution of database servers that are managed independently [5]. Many of such problems regarding distributed databases are further discussed in [6]. The most important problems in distributed databases can be organized in four categories [5, 7]:

1. Different structures
2. Different contents
3. Different naming conventions
4. Different semantics

The structural differences origin from a strict structure that is posed on the values of one attribute. As an example the "name" attribute can have different structures in different databases such as: "firstName- lastName" or "lastName-firstName". Such databases suffer from structural heterogeneity because of the different structure of titles for essentially the same type of values.

The content differences arise when we observe one type of data in different forms. For example in an attribute like "age" one can use "birth-date" in which the "age" can be calculated by reducing the value from the current date. Another one can use "age" which shows the numeric value of the age.

The naming differences origin from the different naming conventions for the same values, like the attributes "Telephone number", "contact number" or "phone".

But the most important problem that we face with is the semantic differences. This problem arises from the various descriptions given for just one concept. There are situations in which missing some information is inevitable during the application of mapping functions. Consider the attribute "grade" which can be filled with a value between 0 and 20 in some educational systems and can be filled with "A", "B", "C", "D" or "F" in other educational systems. Both of these types of values demonstrate a common concept. If we are going to integrate these two kinds of databases we have to face with the semantic differences.

Many approaches currently exist for database integration. In [8] authors propose a method to work on Intelligent Networks based on their properties. In [9] authors propose a new pattern based on the integration of resources. This approach is somewhat different from the other framework and supports the patterns of natural programming languages.

Most of these approaches endeavor to integrate data based on structural relations between entities and do not apply semantic selections among data to solve the problems of data heterogeneity.

In [10] authors developed a model to manage data heterogeneity based on ontology and used semantic web services to extract the necessary information. Since, this approach generates the ontology automatically, it cannot model the real word relations correctly. Generally, an ontology can be made of the following components: (i) Relations among the classes (concepts), (ii) Relations among the instances, and (iii) Relations among the instances and classes [11].

In [12] the authors use ontology and algebraic methods to introduce an approach for the integration of heterogeneous databases. They assume that the ontology of database exists and it is formed and created to be used in a way they need and also it is consistent with their method. Authors in [2] define a new ontology graph that can represent all of the relations in a database and use the ontology graph for data integration and query processing. A method is proposed in [11] that uses query rewriting by creating a global query answering schema for data integration.

Some methods use ontology alignment techniques for integrating distributed ontologies that are created from databases [13–15]. These methods generally define similarity factors and then merge ontologies for creating one global integrated ontology [16–18].

Generally, there are three ways of using ontologies for database integration: (i) the methods based on a unique ontology; (ii) the methods based on multiple ontologies; and (iii) the methods based on hybrid ontologies [19].

The methods that are based on a unique ontology use a universal ontology to present common vocabularies for the description of concepts. In this approach, all of the information resources of an ontology are related together. However, if these information resources have been derived from different schemas then finding an appropriate and common ontology would be cumbersome.

In the multiple ontologies method, each information resource will be described with its own ontology. So, information resources can be changed and modified easily. However, the weakness of this architecture is the lack of common vocabularies which makes comparison of resources difficult.

Hybrid methods are used to address the weakness of the two above-mentioned methods. Similar to the multiple ontologies approach, in this method, each information resource describes its own semantics. In addition, there is a common universal ontology which can be used to compare the ontologies. The advantage of hybrid methods is that new resources can be added easily and without modifying the mappings or even the common vocabularies.

3 The proposed method

Application of ontologies in the area of distributed database integration enables us to query a distributed

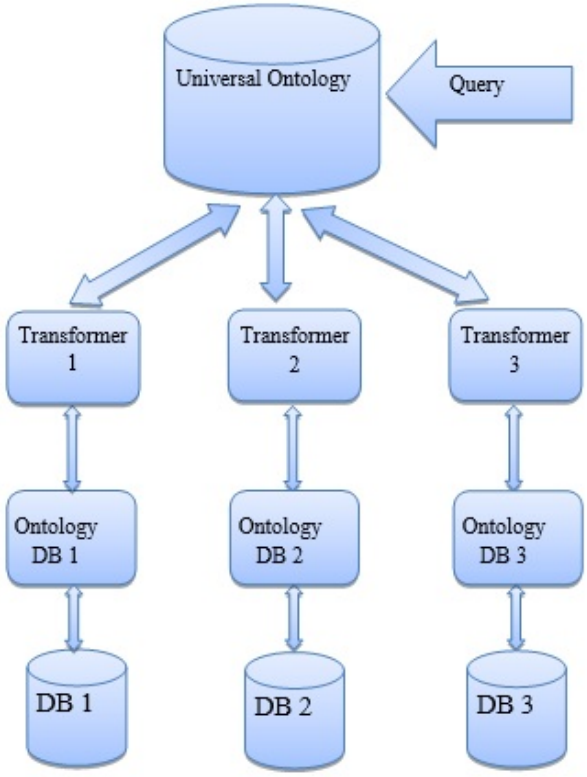

Figure 1: The combination of GAV (Global As View) and the hybrid architecture.

database in a way we are familiar with and receive the right response in the form which we expect, i.e., without any concern about the consistency problems.

In this section, we propose our ontology-based method and describe the necessary rules for the graph construction of the ontology. Then we demonstrate the way users communicate with the system.

3.1 System architecture

The architecture that is used in this research is composed of GAV (Global As View) and hybrid architectures. This architecture is shown in Figure 1. According to this architecture, the user queries the virtual (universal ontology) database (which is a view on the actual database) and receives a response.

The universal database sends a query in an acceptable standard form for each of the database transformers. These transformers transform the input ontology (the ontology that is gained from the user's queries) into a standard form of their own virtual database and resend a response to the virtual database. Then, the virtual database sends the answer to the output in a standard form. This can result in the following advantages:

- Redundancy reduction in the hybrid method: because the virtual database does not store any information and just standardizes user's queries, it sends them to the databases and receives the returned response and shows them as output. In addition, this method has all of the advantages of the hybrid method.
- Using this kind of architecture we can eliminate the wrong responses made up from the ontological graph. This issue is further discussed in the next section.

3.2 The rules for constructing the ontology

Because graphs have a good capability of expressing concepts, we use a graph to describe the ontology rather than the plain text. The graphical method that we use for expressing our ontologies is a modified version of what is presented in [2]. However, the graph used in [2] may result in some unwanted and wrong answers with some queries. For example, as shown in Figure 2, if a user queries "computer" ("pc" and "computer" in the ontology graph are defined as the same) the first tuple, that is not desirable,

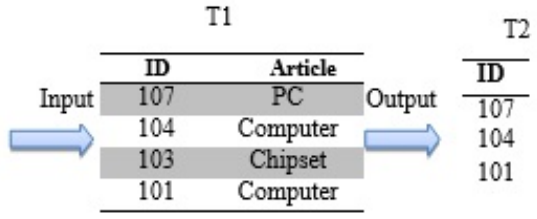

Figure 2: The result of "Select ID from T1 where Article = "computer" ("pc" and "computer" in the ontology graph are defined as the same).

will be included in the response. To address this problem, we optimized this graph according to the architecture discussed in the next section.

We formally define ontology "O" as:

$$O = \{C, A, I, R\}, \tag{1}$$

where 'C' is a set of concepts, 'A' is a set of attributes of concepts, 'I' is a set of instances, and R is a set of all relations that exist between 'C', 'A', and 'I'. The set of 'R' is defined as:

$$R = \{isA, synOf, partOf, atr, val\}, \tag{2}$$

where "isA" is used for Inheritance relations, "synOf" is used for relations between concept 'c' and all of its Synonyms, "partOf" is used for Aggregation relations, "atr" is used for showing attributes of a concept and "val" is used for accessing instances of a concept. Finally, the ontology graph in shown as:

$$OntologyGraph = \{V, L, R\}, \tag{3}$$

where 'V' is a set of nodes that contain 'C', 'L' is a set of all Linked-Lists that are represented by 'A' and 'I'. Head of the list is represented by 'A' and body of the list is represented by 'I'.

As an example, in Table 1, we denote each data represented by column "Name" as a concept in our database, and the set of A = Model, Type by the data in represented columns "Model" and "Type" as instances. In real world, concepts "Computer" and "PC" almost have the same meaning, and as a result we denote them as synonyms and represent their relation with "SynOf". On the other hand, both concepts "Computer" and "Calculator" are data processors, so we define new concept "Data Processor" and create a link between "Computer" - "Data Processor" and also "Calculator" - "Data Processor" by the label "synOf". The concept "Notebook" has many properties of concept "PC" and it can inherit them from "PC", so we represent their relation by "isA". The concepts "Desktop",

Table 1: A sample table that contains product information.

Type	Model	Name
LCD	S1227i	Desktop
Single processor	Dell	Notebook
Usb port	L11h23	Keyboard
Programmer	M19	Calculator
LED	121g2	Monitor
Scientific	N34	Calculator
Desktop pc	Intel	PC
Multi processor	IBM	Computer
Scientific	N45	Calculator

"Keyboard" and "Monitor" are part of "PC", so we can represent their relation with "PC" by "part of". On the other hand, "Keyboard" is part of "Notebook" too, so we use "part of" for the relation between "Keyboard" and "Notebook".

In Figure 3, we have shown the product ontology graph for Table 1. However, not all the instances are shown. For example, two instances of concept "Calculator" are shown. The node "Calculator" $\in$ V has relation with linked-list L1 $\in$ L by "atr". Head of L1 is represented by set 'A' that belongs to the concept "Calculator", this means that head of L1 contains two fields (Model, Type). The body of L1 is represented by set 'I' that belongs to concept "Calculator", this means that it contains three nodes (M19, Programmer), (N34, Scientific) and (N45, Scientific), that are represented in Figure 3. We use relation "val" between the nodes of of linked-list L1 $\in$ L. Also, we define the ontology of the discussed database similar to the picture shown in Figure 3.

One of the advantages of applying this graph is the possibility capability of update. If we insert a new tuple into a table or delete a tuple, what we have to do is updating the linked-list of the values of the related table.

3.3 Integration and responding the user based on the proposed method

When a user queries the universal ontology, he/she starts to build a set called F. F is a tuple composed of five attributes as:

$$F = (c, a, i, s, acc), \tag{4}$$

where "c" is the set of all concepts in an ontology, "a" demonstrates the attributes related to "c", "i" is the set of all instances related to "a", "s" is the set of all synonyms (these synonyms are defined in the universal ontology), and "acc" is the list of all verified accesses of a user to the

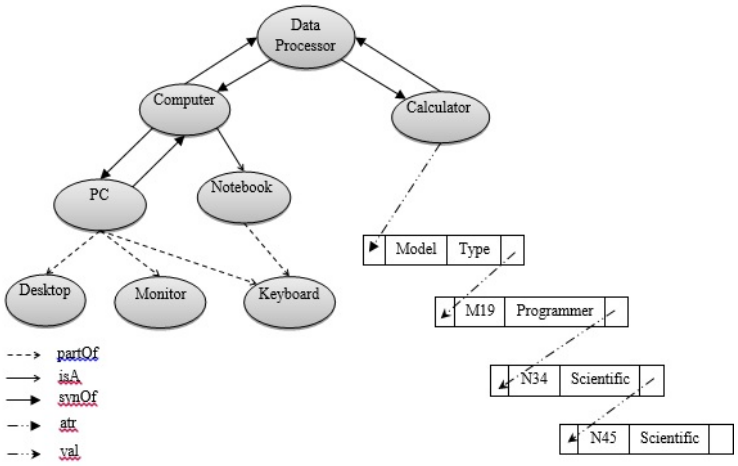

Figure 3: Part of product ontology graph for Table 1.

attributes (this set is constructed based on a predefined and constraint domain). The transformer applies the set F', on the local ontology. The local ontology traverses the existing graph and finds the best answer. Then, it returns the answer to the transformer. Therefore, the transformer sends the responses to the universal ontology in the form of F". The definitions of F' and F" are in Section 3.4. The universal ontology compares the returning responses and deletes the items that are exactly the same and then sends the response to the output.

```
Find_Mappings()
Begin
    EQ := tmpEQ := {DQ}
    For each query qi in tmpEQ Do
        For each rj in Sm Do
            if Match (qi, rj) is true
                add rj to MR
        End For
        For each rj in MR
            Add Replace(qi, rj) to EQ and tmpEQ
        Remove qi from tmpEQ
        If tmpEQ is Empty return EQ
    End For
End
```

Figure 4: The algorithm for finding all semantics of input query to create the final query used by the Local Ontology.

3.4 Integration

In the proposed method, the following steps have to be done for the integration process:

- Constructing one ontology for each database
- Constructing transformer as an intermediate language and as a communication bridge among the local ontologies and the universal ontology
- Constructing one connector between the local ontologies and the users (universal ontology)

We have discussed the first item before. Regarding the second item, that is the construction of transformers, the main idea in applying the transformers is the same as what is done in Java programming language (that is the intermediate language, byte code, is used besides a hybrid translator (language processor) for the purpose of portability).

Making the use of transformers enables us to have communications between the local ontology and the universal one. The transformers can control the returning responses and filter them, when it is necessary. Furthermore, it can add/drop a database to/from the system with fewer modifications to the system. That is because adding/dropping a database can be implemented by adding/dropping its transformer and we do not need to change the universal ontology or the common vocabulary.

The third item is the construction of a universal ontology which is a virtual environment and maintains a set of rules. It enables the user to search and query in the entire system without any problem.

3.5 Response to the user's query

In order to respond to a user, we should use F, F', and F". After sending a query to the transformers and verifying a user with acc, we can remove this parameter from F and define F'. This means that $F' = (c, a, i, s)$. We have introduced F in the previous section. Here we define F' and F".

Assuming an ontology 'K', a query q is sent to this ontology in the form of $qF'_j = (c_j, a_j, i_j, s_j)$ (where $c_j \in c$,

```
Match(q_i, r_j)
Begin
    If Number of F' in q_i is equal to Number of F' in r_j
    Then
            For each F' in q_i Do
        For each edge e_f that has relationship with F' in q_i
                                If T(e_f ∈ q_i) is not equal to T(e_f ∈ r_j)
                                        Return false
        End for
    Return true
            End if
            Return false
End
```

Figure 5: The algorithm for function $Match(q_i, r_j)$.

```
Remove_Extra()
Begin
   For each s_j in F Do
      For each i_j in F'' Do
         If c_j ∈ F'' equal with s_j ∈ F and s_j ∉ F''
            Remove F''_j = (c_j, a_j, i_j, s_j) from F''
End
```

Figure 6: The algorithm for removing extra answers after applying input query to the local transformers.

$a_j \in a$, $i_j \in i$, $s_j \in s$) and the response will be in the form of $F'' = (c, a, i, s)$ where 'c', 'a', 'i' and 's' have the same meaning as in Equation (4).

In order to generate all the responses given to the user, we need to process the user's query based on the "synOf" set and find the suitable semantic mappings that correspond to it. To find all the semantic mappings of a query, we have applied an algorithm which is a modified version of the algorithm presented in [11]. This algorithm is shown in Figure 5.

In Figure 4, EQ is the final query set that is used in the local ontologies, DQ is the set F' resulted from the user's query and Sm is set of semantic solutions using $Match(q_i, r_j)$ function (introduced in Figure 5). The function $Replace(q_i, r_j)$ replaces the left side of relation r_j with relation q_i.

The algorithm shown in Figure 5 ensures us to have the following conditions in the returned semantic mappings:

- The same number of sets in both relations
- Existence of corresponding relations for each of the variables of F' in the resulting relation from the semantic mapping

According to the conditions, only the relations that correspond to the user's query will be chosen as the semantic mappings. After finding appropriate semantic mappings, the transformer sends these mappings to the local ontologies in order to give a response. When the responses are received, the transformer deletes the wrong responses by applying the algorithm presented in Figure 6, similar to the reply that can be seen in Figure 2. Then the final response set will be sent to the universal ontology and the response given to the user will be generated as an output.

According to the algorithm presented in Figure 6, all the responses resulted from synonym concepts, that are defined in the universal ontology and returned by the local ontologies, are sent to the output if they are the members of the same synonym.

When the responses are sent to the universal ontology by the transformers, all pairs of the returned responses have to be compared in order to eliminate the repeated items. In addition, the redundant operations have to be curbed by comparing the different databases.

4 Case study

In order to demonstrate the proposed method, we have implemented a prototype system using the Java programming language. In the prototype system, we assume that the corresponding system contains two databases related to two computer utility stores. We demonstrate how these two databases can be integrated.

This system contains two databases. The first database (DB1) contains a table (TBL1), as shown in Table 2, and the second database (DB2) contains a table (TBL2), as shown in Table 3. The ontology graph of DB1 and DB2 can partially seen in Figure 7. When a user sends a query it will be applied on each of the graphs in the form of some sets that are the representative of various probable semantics.

In this approach, we break the user's query into sub-queries and find all semantics of these sub-queries by traversing the ontology graph and finding the answers of sub-queries and merging them in order to obtain the final answer. This results in increasing the capabilities of the system in generating the response to the user's queries. Moreover, by using such architecture, we can reduce the redundancy and increase the ability applying updates by the system.

Suppose that a user wants to find a list of processors that can be used in the personal computers, then he applies a query like "Select Name, Cost, Rate from UO where Application =PC". All semantics can be extracted from the query based on Figure 4, and query will be sent to each of the local databases as the standard form: "$F = (Processor, \{Application = "PC"\}, \{Name = ?, Cost = ?, Rate = ?\}, \{PC, CPU, Server, Micro\}, True)$".

Table 2: The information contained in TBL1

Rate	Cost	Type	Model	Company	Id
7.5	$22	Desktop	i3-5157u	Intel	3034
5.8	$26	Personal	i3-3210	Intel	3007
8.3	$32	Desktop	i5-5287u	Intel	2432
8.8	$47	Personal	i7-5600u	Intel	2592
8.3	$43	Personal	i7-3770k	Intel	1109
7.9	$45	Mobile	i7-4590	Intel	1074
9.1	$110	Server	E7-8850	Intel	1037
8.6	$122	Server	E7-8850	Intel	1044
6.5	$36	Desktop	Fx-4100	AMD	2737
7.6	$42	Personal	Fx-6300	AMD	3470
9.2	$55	Mobile	Fx-8370	AMD	3473
6.5	$3	Micro	ATtiny 10	AVR	4156
7.2	$4	Micro	ATtiny 25	AVR	4493
6.8	$7	Micro	ATMega 8535	AVR	6789

Table 3: The information contained in TBL2.

Rate	Price	CPU	Id
C+	£19.64	Intel i3-5157u	267
B++	£30.00	Intel i5-5300	274
B+	£28.58	Intel i5-5287u	296
B++	£41.97	Intel i7-5600u	335
C++	£37.40	Intel i7-4765	337
D+	£40.19	AMD Fx-4100	622
C	£22.00	AMD X3-460	768
A+	£51.50	AMD X6-1075	777
D+	£2.76	AVR ATtiny 10	778
C	£3.57	AVR ATtiny 25	910
D++	£6.25	AVR ATMega 8535	911
B	£18.00	ARM Cortex-A72	965
C+	£13.00	ARM Cortex-R5	970
A	£9.00	ARM Cortex-M4	971

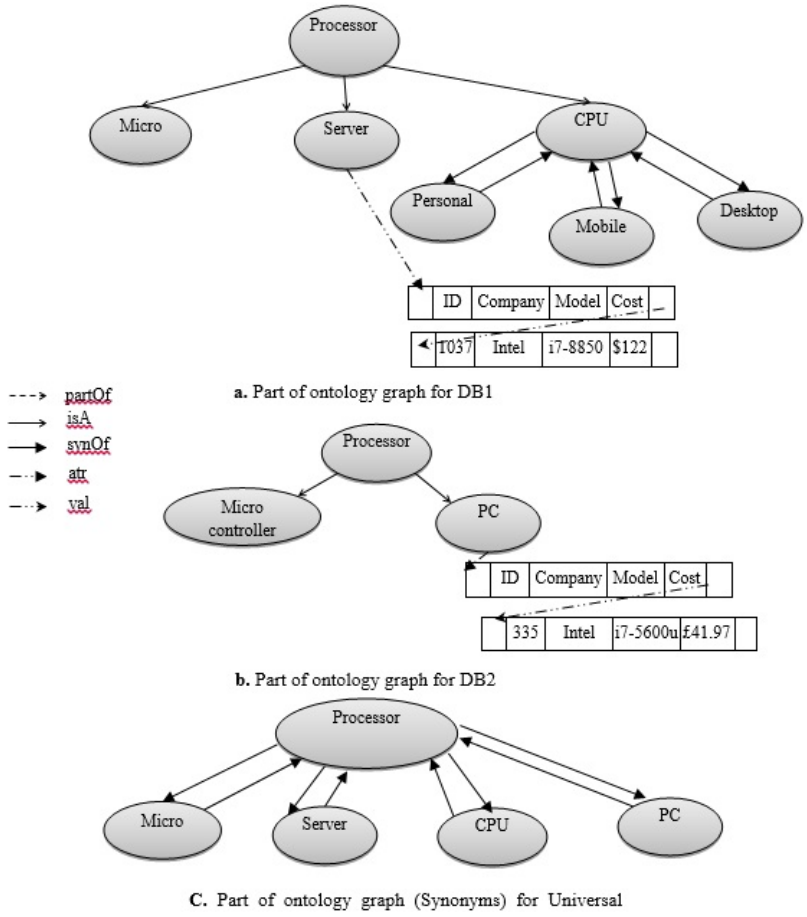

Figure 7: Part of ontology graph of DB1, DB2 and Universal Ontology.

Transformer 1:

F1'1= (Processor, CPU, {Company=?, Model=?, Cost=?, Rate=?}, Personal)

F1'2= (Processor, CPU, {Company=?, Model=?, Cost=?, Rate=?}, Mobile)

F1'3= (Processor, CPU, {Company=?, Model=?, Cost=?, Rate=?}, Personal)

Transformer 2:

F2'1 (Processor, PC, {CPU=?, Price=?, Rate=?}, PC)

Figure 8: Transformers convert input query to the local queries.

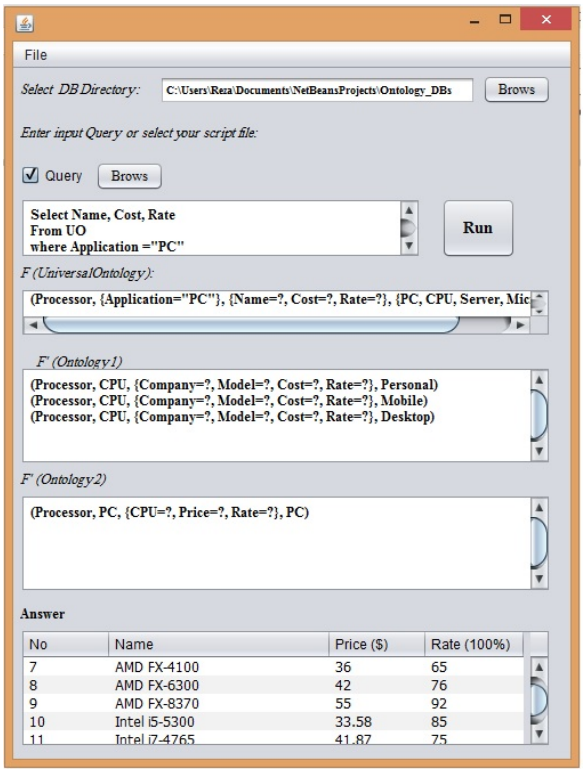

Figure 9: A sample query and extracting its semantics.

After that, transformers 1 and 2 interpret the user's request as shown in Figure 8.

Each database returns its response to the transformers based on its local queries, after refining answers based on Figure 6 (removing concepts which are considered synonymous in universal ontology, but are not synonyms in local ontology). Transformers convert answers to the standard form of universal ontology and send them. After removing similar answers, responses are displayed to the user as shown in Figure 9. In this query "U0" represents the universal ontology "=?" represents what entity the user wants to find.

5 Conclusion

In this paper, we expanded the semantic relationships in the concepts of databases using ontologies. In order to reduce the redundancy and increase the ability of the system for applying updates, we represented a new architecture by combining the methods that are based on data repository and ontology. The advantage of this method can be seen when the new database is added to the system. For this work, there is no need to change or modify the structure of the universal ontology. Moreover, we designed a transformer to coordinate the local ontology with the universal ontology. In addition, when we drop a database from the system, we need to cut the relation between the corresponding transformer and the universal ontology.

Since the databases are created and maintained locally, using the ontology graph and adding connections of synonyms and definitions of the lexemes enables us to have connections between the users and all of the databases without any knowledge of the structural form of these databases.

For increasing the capabilities of the system in generating the response to the user's queries, we made use of semantic mappings. Therefore, all the concepts that can be extracted from the user's request can be replied in an appropriate way. Additionally, we included the capability of security control issues such as access levels for the users, which is not considered in previous research. As a result, the proposed method represents a more powerful approach for integrating databases.

References

[1] S. Staab, R. Studer, Handbook on Ontologies, Springer, Berlin, 2004

[2] C.B. Necib, J. Freytag, Ontology based query processing in database management systems, Proceeding on the 6th International Conference on ODBASE, Springer, Italy, 2888, 839–859, 2003

[3] J.M. Fielding, J. Simon, W. Ceusters, B. Smith, Ontological theory for ontological engineering: biomedical systems information integration, Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR2004), Canada, 2004

[4] W. Sujansky, Heterogeneous Database Integration in Biomedicine, J. Biomed. Inform. 34, 285–298, 2001

[5] H.T. El-Khatib, M.H. Williams, L.M. MacKinnon, D.H. Marwick, A framework and test-suite for assessing approaches to resolving heterogeneity in distributed databases, Inform. Software Tech. 42, 505–515, 2000

[6] J. Euzenat, P. Shvaiko, Ontology Matching, Springer-Verlag, Berlin Heidelberg, 37, 40–42, 2007

[7] J.Y. Tao, J. Qu-Feng, W. HuiJuan, Ontology-based Research on Heterogeneous Database Semantic Integration Strategies, Second Proceedings of 2010 Second International Workshop on Education Technology and Computer Science, Seattle, USA, 477–480, 2010

[8] A. Stephanik, R. Hofestädt, M. Lange, A. Freier, Metabolic Information Control System, In Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, July 22–25, 2001

[9] J.T. McDonald, M.L. Talbert, S.A. DeLoach, Heterogeneous Database Integration Using Agent Oriented Information Systems, The International Conference on Artificial Intelligence (IC-AI'2000), Monte Carlo Resort, Las Vegas, Nevada, June 26–29,

2000

[10] L. Xia, W. Bei, A Framework for Ontology based management of Heterogeneous resources, International Joint Conference on Artificial Intelligence, Hainan Island, April 25–26, 2009

[11] J. Lee, J.H. Park, M.J. Park, C.W. Chung, J.K. Min, An intelligent query processing for distributed ontology, J. Syst. Softw. 83, 85–95, 2010

[12] J. Wang, Y. Zhang, Z. Maio, J. Lu, Query Transformation in Ontology-based Relational Data Integration, Asia-Pacific Conference on Wearable Computing Systems, Shenzhen, 17-18 April 2010

[13] L. Juanzi, J. Tang, Y. Li, Q. Luo, RiMOM: A Dynamic Multistrategy Ontology Alignment Framework, IEEE Trans. Knowl. Data 21, 1218–1232, 2009

[14] C. Xie, Semantic Similarity-Based Ontology Alignment for Enterprise Ontologies, Proceedings of 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '09), Tianjin, 386–390, 2009

[15] A. Mazak, M. Lanzenberger, B. Schandl, iweightings: Enhancing Structure-based Ontology Alignment by Enriching Models with Importance Weighting, Proceedings of 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), Poland, 992–997, 2010

[16] F. Natalya Noy, Semantic Integration: A Survey Of Ontology-Based Approaches, SIGMOD Record 33(4), 65–70, 2004

[17] Ch. Namyoun, S. Il-Yeol, H. Hyoil, A Survey on Ontology Mapping, SIGMOD Record 35(3), 34–41, 2006

[18] P. Shvaiko, J. Eenat, A Survey of Schema-Based Matching Approaches, Journal on Data Semantics IV, Springer-Verlag, Berlin Heidelberg, LNCS 3730, 146–171, 2005

[19] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hubner, Ontology-based integration of information - a survey of existing approaches, Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing, USA, 108–117, 2001