JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 9, NO. 12, DECEMBER 2006(pp. 1552-1559)

A Novel Method for Matching between RDBMS and Domain Ontology

Ki-Jung Lee[†], Taeg-Keun Whangbo[††]

ABSTRACT

In a web environment, similar information exists in many different places in diverse formats. Even duplicate information is stored in the various databases using different terminologies. Since most information serviced in the current World Wide Web however had been constructed before the advent of ontology, it is practically almost impossible to construct ontology for all those resources in the web. In this paper, we assume that most information in the web environment exist in the form of RDBMS, and propose a matching method between domain ontology and existing RDBMS tables for semantic retrieval. In the processing of extracting a local ontology, some problems such as losing domain information can occur since the correlation of domain ontology has not been considered at all. To prevent these problems, we propose an instance-based matching which uses relational information between RDBMS tables and relational information between classes in domain ontology. To verify the efficiency of the method proposed in this paper, several experiments are conducted using the digital heritage information currently serviced in the countrywide museums. Results show that the proposed method increase retrieval accuracy in terms of user relevance and satisfaction.

Keywords: Ontology Matching, Instance Matching, Semantic Matching, Semantic Searching System, RDBMS

1. INTRODUCTION

In a semantic web environment, the distributed web resources must be queried efficiently, and typically most of the web resources are in the form of relational database. In order to efficiently search the distributed data, a meta-data of the database and an ontology defining the relationships between the meta-data is needed, and through matching between local ontologies and the domain ontology, an integrated semantic search can be performed. The ontology for each database must be established in precedence, and deep knowledge on ontology building is required for each database administrator. Also, a domain ontology for an integrated search through the local ontologies must be built.

This paper presents a method which automatically performs the matching between the relational databases built in each institution and domain ontology for the integrated search. In order to perform the matching between local databases and domain ontology, the local database structure must be known, and based on it, the semantic elements are extracted. If semantic elements are extracted simply using the structure, the matching will not produce good results as a different form of ontology is made for each individual. Accordingly, this paper uses a method comparing the instances of the relational database and domain ontology and extract-

※ Corresponding Author : Taeg-Keun Whangbo, Address : (461-701) 519 Seromkwan Kyungwon University, San 65, Bokjung-Dong, Sujung-Gu, Songnam, Kyunggi-Do, Korea, TEL : +82-31-750-5748, FAX : +82-31-750-5662, E-mail : tkwhangbo@kyungwon.ac.kr

Receipt date : Oct. 31, 2006, Approval date : Dec. 22, 2006

[†] Dept. of Computer Science, Kyungwon University (E-mail : jcm5758@empas.com)

[††] Dept. of Computer Science, Kyungwon University

※ This research was supported by the Ministry of Education, Seoul, Korea, under the BK21 project.

ing the classes from them. With this method, other types of classes existing in the tables can be extracted. Then, matching using the correlation with the domain ontology is performed, increasing the matching accuracy.

The outline of this paper is as the following. Related research on this subject will be introduced in section 2, the problem definition and terms used in this paper described in section 3. Section 4 discusses the methods of extracting meta-data from the relational database and extracting class candidates, and the method of matching between the relational database and domain ontology will be described in detail in section 5. The performance of the presented algorithm will be evaluated in section 6 and the conclusion and future works will be mentioned in section 7.

2. RELATED RESEARCH

In the problem of matching between relational database and ontology, the most of the research techniques extract an ontology from the database and matched it with the domain ontology. The research on extracting ontology from the relational database is divided into two kinds of method. One method use table names and the other use the Entity-Relationship (ER) data model. The method using table names[1] employs an algorithm which compares table names with ontology class labels and matches those similarity with each other. However, the accuracy of the match drops if the names are not similar, or if a same name is used for data of different meaning.

An[2] and Gramajo[3] used the ER model and divided the entity types into strong and weak, and built the ontology by defining their relationships in functions. However, this had the disadvantage that even if a relational database was built, it was difficult to apply to cases where the ER model was not used or the database does not have the ER model currently. Upadhyaya[4] used the extended ER model and field attribute information in extracting ontology. Li[5] used key information in the ER model, that is, primary key and foreign key information, and researched on making relations between tables. Trinh[6] extracted metadata and constraints of the database without using the ER model, and generated an ontology through the relationships between the extracted information and separately defined terms.

Methods for ontology matching use the ontology structure, ontology components and machine learning. The method using the ontology structure[7,8] represents the structure in a graph, model or tree form and performs the match using the similarity of the ontology structure. Evaluating the similarity using only the ontology structure impairs the internal meaning of the ontology itself. The method using the ontology components[9,10] usually compares the similarity between the ontology class names. In this case, only the literal similarity is used regardless of the contents of the ontology. In the machine learning approach[11], a machine learns the ontology through the guidance of an expert and performs matching through this. The help of an expert is necessary in this case.

3. PROBLEM DEFINITION AND DESCRIPTION OF TERMS

In the previous research on matching between the relational database and domain ontology, the ontology was extracted first and then matched with the domain ontology. In this case, the information of the domain ontology is not utilized during the process of extracting the local ontology from the relational database and only the database structure is used, causing the loss in semantic elements. This has a disadvantage of deteriorating the accuracy.

This paper presents a method in which the information of the relational database is utilized more efficiently using the relationships between the re-

lational database tables and matching with the domain ontology instances.

Ontology is a description of the relationships between the metadata, and has elements such as class, data property and object property. Classes are the conceptualization of resources, such as temple, stupa (meaning tower or pagoda) and other terms in this context. Data properties are the attributes of the resources such as thetemple's name or founder. Object properties represent the relationship between the classes, and have a domain and range. The domain is the subject of the relationship and the range shows the applicable range of the relationship. One example can be 'temple hasStupa stupa'.

The classes extracted from the relational database in this paper are represented as CD(class name), and classes used in the domain ontology as CO(class name). For instance, if the name of the class extracted from the relational database is Stupa, then it is represented as CD(Stupa). Data properties are represented as DPD(class name, attribute name) and DPO(class name, attribute name) and object properties OPD(domain, range) and OPO(domain, range).

3.1 Description of Used Data

The ER data model used in this paper is shown in Fig. 1. A total of five tables were used -

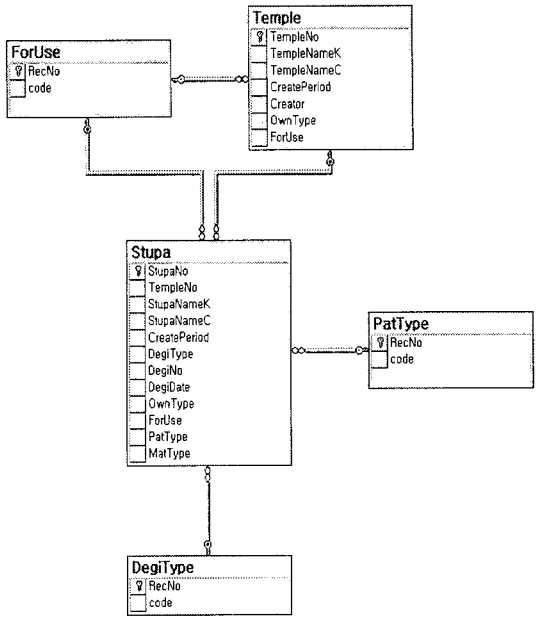

Fig. 1. ER data model which is used for this paper.

Temple, Stupa, ForUse, PatType, DegiType. The Templetable holds information of the temple and Stupa table holds information of the stupas existing in the temple. ForUse, PatType, DegiType tables are similar to each other, and hold information of the use, stupa's pattern and designation respectively.

Fig. 2 shows part of the ontology used in this paper. There are a total of seven superclasses, and each superclass has subclasses, data properties and object properties.

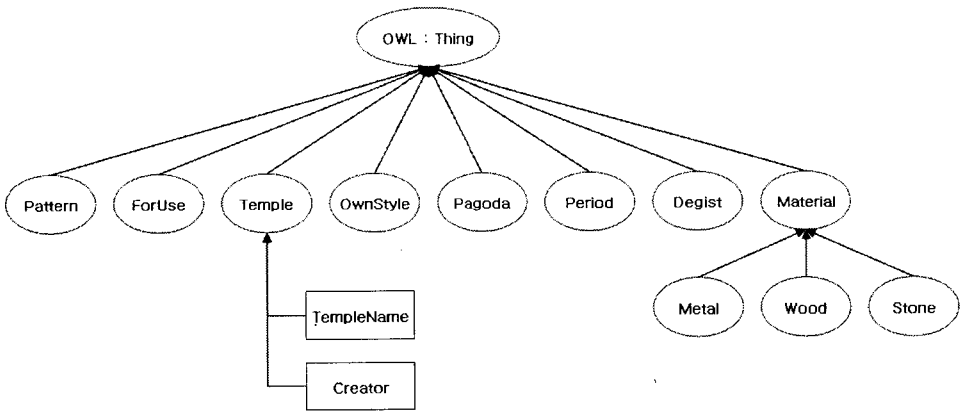

Fig. 2. Ontology structure used in this paper.

4. EXTRACTING METADATA FROM RELATIONAL DATABASE AND EXTRACTING CLASS CANDIDATES

In order to perform the matching between the relational database table and ontology, the database metadata must be extracted first. In this process, class candidates, data property candidates and relationships between the tables are extracted based on the ER data model.

Previous approaches used only the table information, and could not extract subclasses existing in the table. To complement this difficulty, the table's field information was extracted and a class matching with the domain ontology was performed. The subclasses in the tables were able to be extracted using the results of the class matching.

4.1 Process of Extracting Class Candidates Using Metadata

The metadata of a relational database holds information such as the table structure, field information and key information, and class candidates are extracted by analyzing this information. First of all, each table is selected as a class candidate. For this paper, all tables in the database were selected as class candidates. This is due to the general presumption that all the tables in the database were made for some specific need. Five classes, CD(Temple), CD(Stupa), CD(ForUse), CD(PatType), CD(DegiType), are generated.

4.2 Process of Extracting Class Candidates By Comparing Similarity

Table fields can be divided into class and data property. These fields represent the detailed attributes of the specific table or can be represented as independable classes such as code information. For instance, the TempleName field in the Temple table shown in Fig. 1 is an attribute representing the name of the Templeand can be considered as a data property. However, such characteristics cannot be distinguished simply using the database metadata, and so the distinction was made through the field instances in this paper. For this, the field instances in each table were compared with the instances of the terminal classes in the domain ontology for similarity, and the class candidates were extracted. If the similarity comparison value surpasses a certain value, it is selected as a class candidate; otherwise, it is classified as a data property.

Also, the independable fields in the table can be extracted as classes through the similarity comparison. For instance, the Stupa table has a CreatePeriod field which stores the information of the time period. Generally, time period information can be distinguished through their special code form. Accordingly, CD(CreatePeriod) can be extracted by performing matching between the domain ontology's period class and instances.

The similarity comparison is performed on all fields in every table, but this process is omitted if the field is designated as a primary key and is a reference field of another table while being in the form of a series of numbers. This is due to the fact that foreign keys refer to data of other tables while primary keys are generally used for indexing.

In measuring the similarity, Levenshtein Distance (LD) and Longest Common Prefix/Suffix(LCPS) were used. LD[12], also known as Edit Distance, is the number of insertions, deletions and substitutions of a single character needed to transform string s to t, and is represented as the following (1).

$$LD(s,t) = 1 - \frac{changeLenth}{MaxLength} \quad (1)$$

In (1), maxLength is represented as max (length(s), length(t)) and changeLength is the number of operations for the transformation. This is a function which measures the number of changes needed to make the strings identical. For instance, "Unified Shilla Period" and "Unified Shilla" can be made by six insertions.

LCPS uses the number of characters which match in the prefix or suffix for the comparison. Expression (2) shows the prefix similarity and expression (3) the suffix similarity. This paper uses (4), which uses the higher value between the prefix and suffix.

$$LCP(s,t) = \frac{prefixLength}{MinLength} \quad (2)$$

$$LCS(s,t) = \frac{suffixLength}{MinLength} \quad (3)$$

$$LCPS(s,t) = \max(LCP(s,t), LCP(s,t)) \quad (4)$$

The prefixLength in (2) is the length of matching prefix characters and minLength is min(length(s), length(t)). The suffixLength in (3) is the length of matching suffix characters.

Similarity comparison is performed for every field in the table and instances of every terminal class in the domain ontology using (5). The usage level of LD and LCPS are applied differently by the user. The final similarity measurement value is generated from (6).

$$SD(s,t) = \alpha \times LD(s,t) + (1-\alpha) \times LCPS(s,t)$$
$$where \quad 0 \le \alpha \le 1 \quad (5)$$

$$GSD_k = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} SD(s_i, t_j) \quad (6)$$

$SD(s_i, t_j)$ shows the similarity of each matched pair, and this is applied with a separate weighted value in LD and LCPS for the calculation. Elements which surpass a certain level of similarity are classified as class candidates, otherwise, data fields.

Table 1 shows the results of class matching through the similarity measurement process.

4.3 Extracting Object Properties with Key Information

Generally, relational databases use foreign keys to represent relationships withother tables. This relationship can be an ontology's Is-A relationship or object property relationship. The Is-A relationship represents the inheritance concept such as a material class and stone class shown in Fig. 2, and the object property represents the possession concept "Temple class has a stupa."

Table 1. Result of class matching

Table name	Field name	Matched class
DegiType	code	Degist
ForUse	code	Sanup
PatType	code	Anmimal
Stupa	CreatePeriod	Period
Stupa	OwnType	Government
Stupa	MatType	Wood
Temple	CreatePeriod	Period
Temple	OwnType	Government

In this paper, the object property relationship was established using the foreign key information. Foreign keys refer to information of other external tables, and in this case, the ForUse field in Stupa table referred to the code field in ForUse table as shown in Fig. 1. However, the foreign key cannot be interpreted as a one-way relationship. For example, the ForUse and Stupa tables in Fig. 1 are defined as primary keys, and the Stupa table can be represented as having the ForUse table. However, in the case of the Temple and Stupa tables, Stupa cannot possess a Temple but Templecan possess a Stupa. Accordingly, a candidate relationship setting a two-way relationship can be made in this stage, and information regarding this can be modified during the comparison for relationship similarity with the domain ontology.

The results of generating object properties Fig. 1 are shown partially in Table 2.

5. CLASS SELECTION AND RELATIONSHIP MATCHING

The similarity comparison with the domain ontology is performed withthe extracted class candidates, data properties and object properties, the classes are selected from the class candidates through this

Table 2. Generated object properties

Relation	Domain	Range
hasClass	Stupa	ForUse
		PatType
		DegiType
		Temple
		CreatePeriod
		OwnType
		MatType
	Temple	Stupa
		CreatePeriod
		OwnType
		ForUse

process. The relationships between the classes are compared through object property matching to generate the final matching information.

5.1 Class Selection

The class similarity comparison performed in the previous section extracted class candidates through comparing the database field instances with the instances of the domain ontology's terminal nodes. Since the similarity comparison with only the terminal classes were performed, a comparison with the upper nodes, that is upper classes, is required.

For instance, supposing that the MatType field in the Stupa table matched with CO(stone) of the domain ontology's CO(material), it holds the connotation for a possible match with CO(material). In this case, the instance of MatType field is compared for similarity with CO(stone)'s upper class CO(material). If the similarity passes a certain level, MatType field is changed to match with the material class and if it does not pass, it maintains its match with the stone class.

The pseudo code for this process is shown in Fig. 3, and the method for similarity measurement is the same one used in chapter 3. Renewed class matching results are shown in Table 3.

```
FOR each class of DB class list
    Set co to matched class of class
    IF co has sibling class and co is not root class THEN
        IF InstanceMatching(class, parent of co) THEN
            Update matched class to parent of co
        END IF
    END IF
End FOR
```

Fig. 3. Pseudo code : comparing parent class.

Table 3. Matching result of updated class

Table name	Field name	Matched class
DegiType		Degist
ForUse		ForUse
PatType		Pattern
Stupa	CreatePeriod	Period
Stupa	OwnType	Own
Stupa	MatType	Material
Temple	CreatePeriod	Period
Temple	OwnType	Government

5.2 Object Property Matching

The final process in the matching between the relational database and domain ontology is the matching using the class and object property.

In Fig. 2, the domain ontology's Templeclass has object property relationships with stupa, time period, possession and use classes, and the temple name and founder as data properties. Through comparing the object property of the selected class and that of the domain ontology, and finding the same type of classes which have an object property relationship including similar data property, the matching information is generated. Figure 4 shows the pseudo code of the object property comparison.

```
FOR each class of DB class list
    Set cdop to object property of DB class
    FOR each class of Ontology class list
        Set coop to object property of ontology class
        IF compare(cdop, coop) THEN
            CALL matching(cdop, coop)
        END IF
    END FOR
End FOR
```

Fig. 4. Pseudo code : comparing object properties.

6. EMPIRICAL ANALYSIS

To verify efficiency of proposed algorithm, we used three sets of databases and an domain ontology. The first database was described in section 3, and two other databases are from CHA[13] and KoreaTemple[14]. To compare matching efficiency, we have matched three databases to the ontology which was described in chapter 3.

Table 4 show the result of matching and the average accuracy is about 84%. In Table 4, No. of table represents number of tables in each databases, No. of extracted class represents number of extracted class in each databases. And No. of relation represents number of object properties. Matching accuracy is calculated by expression (7).

Matching Accuracy = right matching / number of extracted classes (7)

In case of mis-matched classes, those classes were extracted erroneously. In case of DB1, Creator field of Templetable was matched to Mud class of Material Class. But actually it had to be a data property. In our opinion, matching accuracy will be increased if we use more instances.

7. CONCLUSION

This paper presented a method for matching between the relational database and domain ontology to provide an integrated search in a semantic web environment. In order to perform the matching, the relational database metadata is used to extract table, field and relationship information of the database. Using the extracted table information, the class candidates were generated. Also, through measuring the similarity of field instances with the domain ontology, the class candidates and data property were classified, and the object property for database classes were established using the relationship information. Finally, the extracted information was compared with the ontology structure to generate the matching information.

Table 4. Experiment result

	No. of table	No. of extracted class	No. of relation	Matching accuracy (%)
DB1	5	9	12	88.8
DB2	6	12	17	83.3
DB3	10	21	30	80.9
Average Accuracy				84.3

The extracted matching information using the method proposed in this paper can be applied to an integrated search in a semantic web environment, and takes more consideration in the semantic elements compared to the previously performed simple structural matching or instance matching.

8. REFERENCES

[1] J. Kang and J. F. Naughton, "On Schema Matching with Opaque Column Names and Data Values," *In Proceedings of SIGMOD 2003*, pp.205-216, 2003.

[2] Y. An, A. Borgida, and J. Mylopoulos, "Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences," *In Proceedings of ODBASE*, LNCS3761, pp.1152-1169, 2005.

[3] Gramajo Javier and David Riano, "Meta-data and ER Model Automatic Generation from Unstructured Information Resources," *In Proceedings of 5th Joint Conference on Knowledge-Based Software Engineering*, pp. 181-186, 2002.

[4] Sujatha R. Upadhyaya and P. Screenivasa Kumar, "ERONTO: A Tool for Extracting Ontologies from Extended E/R Diagrams," *In Proceedings of SAC05*, pp.666-670, 2005.

[5] Man Li, Xiao-Yong Du, and Shan Wang, "Learning Ontology from Relational Database," *In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, pp: 3410-3415, 2005.

[6] Quang Trinh, Ken Barker, and Reda Alhajj, "RDB2ONT: A Tool for Generating OWL Ontologies from Relational Database System," *In Proceedings of AICT2006*, 2006.

[7] M. Ehrig and S. Staab, "QOM - quick ontology mapping," *In Proceedings of the Third International Semantic Web Conference*, pp. 683-696, 2004.

[8] Natalya Fridman Noy and Mark A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," *In Proceedings of AAAI*, pp.450-455, 2000.

[9] H. Do and E. Rahm, "COMA - A System for Flexible Combination of Schema Matching Approaches," *In Proceedings of VLDB*, pp. 610-621, 2002.

[10] J. Madhavan, P. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," In *Proceedings of VLDB*, pp.48-58, 2001.

[11] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy, "Learning to match ontologies on the Semantic Web," *The VLDB Journal*, pp. 303-319, 2003.

[12] J. De Bo, P. Spyns, and R. Meersman, "Assisting ontology integration with existing thesauri," *In Proceedings of ODBASE*, pp. 801-818, 2004.

[13] Cultural Heritage Administration, http://www.cha.go. kr.

[14] Korea Temple, http://eng.koreatemple.net.

Ki-Jung Lee

He received his M.S. degree in Computer Science from Kyungwon Unversity, Korea in 2003. He is currently a Ph.D. candidate in Computer Science at Kyungwon University. His research interests include Semantic Web, Semantic Matching, Ontology Matching, and Content-based Image Retrieval.

Taeg-Keun Whangbo

He received his Ph.D. degree in Computer Science from Stevens Institute of Technology, USA in 1995. He is currently a professor in the Software College of Kyungwon University. His research interests include Semantic Web, Information Visualization, 3D Graphics, and Mobile Applications.