Journal of Southeast University (English Edition) Vol. 24, No. 3, pp. 385 - 388 Sept. 2008 ISSN 1003—7985

Ontology-based schema matching method in web query interface integration

Yan Zhongmin Li Qingzhong Cao Luhui Kong Lanju Dong Yongquan Ding Yanhui

(School of Computer Science and Technology, Shandong University, Jinan 250101, China)

Abstract: Deep web data integration needs to do schema matching on web query interfaces and obtain the mapping table. By introducing semantic conflicts into web query interface integration and discussing the origins and categories of the semantic conflicts, an ontology-based schema matching method is proposed. The process of the method is explained in detail using the example of web query interface integration in house domain. Conflicts can be detected automatically by checking semantic relevance degree, then the categories of the conflicts are identified and messages are sent to the conflict solver, which eliminates the conflicts and obtains the mapping table using conflict solving rules. The proposed method is simple, easy to implement and can be flexibly reused by extending the ontology to different domains.

Key words: deep web; web query interface integration; semantic conflict; schema matching; ontology

In recent years, the web has been rapidly “deepened” by more and more networked databases on the Internet, and it is believed that a significant amount of information is “hidden” in the deep web, behind the query forms of searchable databases. Since the information in these databases cannot be accessed directly through static URL links, they are only available as responses to dynamic queries submitted through the query interface of a database. Because current crawlers cannot effectively query databases, such data are invisible to traditional search engines, and thus remain largely hidden from users[1]. It is in high demand to provide integrated search systems over web databases, and integrating query interfaces is the first and the most important step in implementing such kinds of systems, because the quality of the integrated query interface will directly influence the final results of integrated systems. We have proposed a frame named IQIBO to integrate query interfaces, in which we obtain a UQI based on domain ontology and obtain the mapping of query interfaces and the UQI[2]. In the IQIBO, the main bottleneck is the semantic heterogeneity existing in query interfaces and the UQI. In other words, we must eliminate semantic conflicts and resolve semantic heterogeneity.

In this paper, we propose a method to do schema matching, which includes three modules: a coordinator, a conflict detector and a conflict solver. In the method, the concepts in different web query interfaces are described as ontologies and sent to the conflict detector. The conflict detector first computes the semantic relevance degree of the UQI and every query interfaces; if they are relevant, the conflict detector detects if there are any semantic conflicts, and then reports the conflicts to the conflict solver to eliminate the conflicts and obtain the mapping table.

1 Semantic Conflict

Semantic conflicts refer to the conflicts caused by using different ways in different web databases to express the same entity in reality[3]. They can be classified as data-level conflicts or as schema-level conflicts[4]. Data-level conflicts are due to different perceptions of the same concepts, and schema-level conflicts are due to differences in logical structure of the same concept. We give the concepts of every type of conflicts in Tab. 1. For example, dates can be represented as a 6-character string (e. g. “160505”) or as a Julian date (e. g. “16-May-2005”), which causes a data representation conflict. Another example, in a concept “house” is represented as House (floor number, size, price), and in another data resource it is represented as House (location, floor size, price), which causes a schema isomorphism conflict.

In web query interface integration, we will meet all kinds of conflicts except ontology-caused conflict.

2 Schema Matching Method

In the IQIBO, there is a processor to dispose query interfaces, which has two parts, semantic matching and tentative queries detection. The processor can discover web sites in special domains on the web efficiently and accurately by using ontology and match UQI and query interfaces to generate a mapping table. In this chapter we will give explanation concerning semantic matching, which concludes with the semantic conflicts solver as the main component.

2.1 Semantic conflicts solver

The semantic conflicts solver is the key in semantic matching, and it consists of three parts: they are the coordinator, the conflict detector and the conflict solver. Fig. 1 shows its structure and its relationship with other parts in IQIBO.

In the method, web query interfaces are described using ontology and sent to the conflict detector. The conflict detector finds semantic conflicts and reports them to the conflict solver. After eliminating the conflicts, the conflict solver gets the mapping table. The coordinator is responsible for coordinating and monitoring all kinds of activities, and mainly interacts with the conflict detector and the conflict solver.

Received 2008-04-15.

Biographies: Yan Zhongmin (1977—), female, graduate; Li Qingzhong (corresponding author), male, doctor, professor, lqz@ sdu. edu. cn.

Foundation items: The National Natural Science Foundation of China (No. 60673130), the Natural Science Foundation of Shandong Province (No. Y2006G29, Y2007G24, Y2007G38), the Encouragement Fund for Young Scholars of Shandong Province (No. 2005BS01002).

Citation: Yan Zhongmin, Li Qingzhong, Cao Luhui, et al. Ontology-based schema matching method in web query interface integration[J]. Journal of Southeast University (English Edition), 2008, 24(3): 385 - 388.

Tab. 1 Concepts of all types of semantic conflicts

Class	Conflict type	Description
Data-level conflict	Data value conflict	Different interpretations of the meaning of data instance values.
	Data representation conflict	Similar objects are described by different data types or data format representations.
	Data unit conflict	Use of different measurement units.
	Data precision conflict	Implementation of different scales, different domain precisions, or different data granularities.
	Data reliability conflict	Data present in different databases may be subject to data reliability.
	Data range conflict	Use different ranges.
Schema-level conflict	Naming conflict	Labels of schema elements are somewhat arbitrarily assigned by different database designers.
	Entity identifier conflict	Assignment of different identifiers to the same concept in different databases.
	Schema isomorphism conflict	The same concept is described by a dissimilar set of attributes or is not set-operation compatible.
	Generalization conflict	Different design choices for modeling related entity classes.
	Aggregation conflict	When an aggregation is used in one database to identify a set of entities in another database.
	Relation conflict	The relationships of the same concept with other concepts are different in different databases.
	Restriction conflict	The restrictions on the same concept are different in different databases.
	Ontology caused conflict	Use different ontology languages or use different versions of the same ontology language.

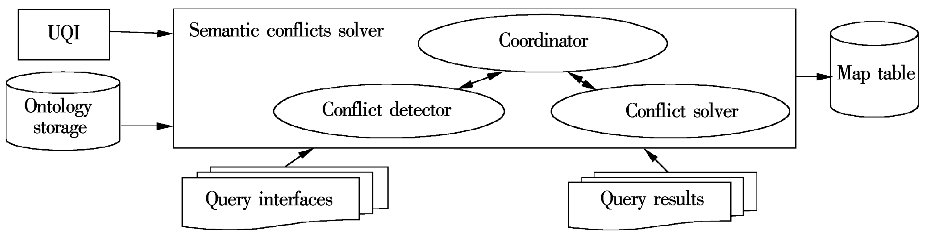


Fig. 1 The structure of semantic conflicts solver

2.2 Ontology description of concepts

In the conflicts detection algorithm, concepts in different web databases are described in OWL[5] as follows:

〈owl: Class rdf: ID = "concept"/〉
〈owl: Class rdf: ID = "attribute"/〉
〈owl: Class rdf: ID = "restriction"/〉
〈owl: Class rdf: ID = "identifier"〉
  〈rdfs: subClassOf rdf: resource = "#restriction"/〉
  〈/owl: Class〉
〈owl: Class rdf: ID = "reference"〉
  〈rdfs: subClassOf rdf: resource = "#restriction"/〉
  〈/owl: Class〉
〈owl: Class rdf: ID = "constraint"〉
  〈rdfs: subClassOf rdf: resource = "#restriction"/〉
  〈/owl: Class〉
〈owl: ObjectProperty rdf: ID = "Hasattribute"〉
  〈rdfs: domain rdf: resource = "#concept"/〉
  〈rdfs: range rdf: resource = "#attribute"/〉
  〈/owl: ObjectProperty〉
〈owl: ObjectProperty rdf: ID = "Hasrestriction"〉
  〈rdfs: domain rdf: resource = "#concept"/〉
  〈rdfs: range rdf: resource = "#restriction"/〉
  〈/owl: ObjectProperty〉
〈owl: ObjectProperty rdf: ID = "Hasidentifier"〉
  〈rdfs: subPropertyOf rdf: resource = "# Hasrestriction"/〉
  〈rdfs: range rdf: resource = "#identifier"/〉
  〈/owl: ObjectProperty〉
〈owl: ObjectProperty rdf: ID = "Hasreference"〉
  〈rdfs: subPropertyOf rdf: resource = "# Hasrestriction"/〉
  〈rdfs: range rdf: resource = "#reference"/〉
  〈/owl: ObjectProperty〉
〈owl: ObjectProperty rdf: ID = "Hasconstraint"〉
  〈rdfs: subPropertyOf rdf: resource = "# Hasrestriction"/〉
  〈rdfs: range rdf: resource = "#constraint"/〉
  〈/owl: ObjectProperty〉

We use a tree structure (see Fig. 2) to preserve the concepts. For example, Fig. 3 is a housing web query interface, and there are eight labels such as information category (sale or rent), location, price, house kind, size, floor number, fitment and keyword. We can obtain all the labels, we can describe it using a tree structure (see Fig. 4).

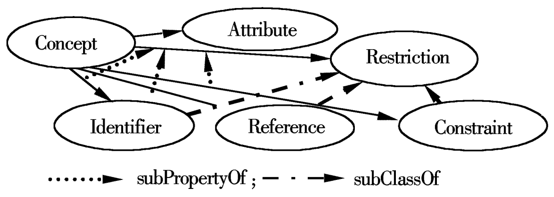
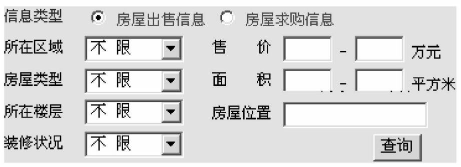


Fig. 2 A part of tree structure for concept



信息类型	[✓ 房屋出售信息] 房屋求购信息		
所在区域	不 限	售　　价	- 万元
房屋类型	不 限	面　　积	- 平方米
所在楼层	不 限	房屋位置	
装修状况	不 限		查询

Fig. 3 A house web query interface

It is a part of description and there is more information that needs to be written down. For example, class attribute has several characteristics to record, which are data-type, data-unit, data-precision, data-reliability, and data-range. In a similar manner, other classes also have their appropriate

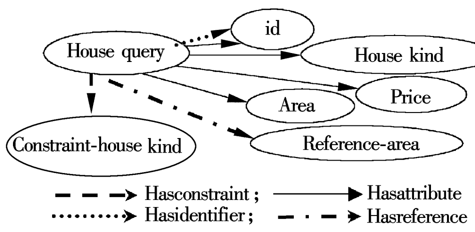


Fig. 4 A part of house

characteristics to be recorded. We are not going to dwell on them.

2.3 Conflict detector

The conflict detector is responsible for detecting various types of semantic conflicts, which is the emphasis of this paper. The algorithm is described as follows:

Input: The concepts in the UQI and query interfaces. The query results from different web query interfaces.

Step 1 Compute semantic relevance degree.

Step 2 Detect schema-level conflicts. When a conflict is found, write it into the conflict list and repeat step 2.

Step 3 Detect data-level conflicts. When a conflict is found, write it into the conflict list and repeat step 3.

Output: The conflict list.

In the algorithm, detecting schema-level conflicts is prior to detecting data-level conflicts, because schema-level conflicts have higher priority than data-level conflicts.

2.3.1 Computing semantic relevance degree

Irrelevance degree[6] is introduced before semantic relevance degree is defined.

Definition 1 The irrelevance degree of p and q is expressed using $I(p, q)$, the semi-irrelevance degree of p and q is expressed using $S(p, q)$ and the upper bound set of p is expressed using $U(p)$, These definitions satisfy the following conditions:

1) $\mathrm{Max}(I(p, q), I(q, r)) \in U(I(p, r))$;

2) Replace $I(p, q)$ in 1) with $U(I(p, q))$, and/or replace $I(q, r)$ with $U(I(q, r))$, 1) is still tenable;

3) $\forall p \subset q, I(p, q) = \min\{u \mid u \in U(I(p, r))\}$;

4) $\mathrm{Max}(S(p, q), S(q, r) \in U(S(p, r))$;

5) Replace $S(p, q)$ in 4) with $U(S(p, q))$, and/or replace $S(q, r)$ with $U(S(q, r))$, 4) is still tenable;

6) $S(p, q) = \min\{u \mid u \in U(S(p, r))\}$;

7) $I(p, q)$ is a finite value in the conditions of 1) to 6), other conditions, $I(p, q) = \infty$;

8) If q is an ancestor of p, $I(p, q) = 0$.

Next is the definition of the relevance degree[7] of two ontologies.

Definition 2 Given any two ontologies OP and OQ in a specific field model DM, if the irrelevance degree of OP and OQ is n, $n \in N$ and $0 \leqslant n \leqslant \infty$, the relevance degree is expressed using $R^W_{\mathrm{DM}}(p, q)$, $0 \leqslant R^W_{\mathrm{DM}}(p, q) \leqslant 1$, and

1) $R^W_{\mathrm{DM}}(p, q) = 0$, when $n = \infty$;

2) $R^W_{\mathrm{DM}}(p, q) = 1$, when $n = 0$;

3) $R^W_{\mathrm{DM}}(p, q) = 1/n$, when $0 < n < \infty$;

4) $R^W_{\mathrm{DM}}(p, q) = 1$, when $p > q$, that is p is an ancestor of q;

5) $R^W_{\mathrm{DM}}(a, c) = \min(m, n)$, when $R^W_{\mathrm{DM}}(a, b) = n$ and $R^W_{\mathrm{DM}}(b, c) = m$.

Under these definitions, the semantic relevance degree of two concepts can be computed. By setting a threshold, we can know whether the two concepts are semantically relevant, then a conflict detection algorithm is applied to analyze whether they are conflictive and which type the conflict is.

2.3.2 Detecting schema-level conflicts

Given two semantically relevant concepts C1 and C2, we detect schema-level conflicts as below.

Step 1 Build trees for C1 and C2.

Step 2 Compare the names of C1 and C2. If their names are different, there is a naming conflict.

Step 3 Compare every attribute of C1 with all attributes of C2 to determine whether they are semantically relevant. If there are semantically relevant attributes, compare their names. If their names are different, there is a naming conflict. Compare semantically relevant attributes to find generalization conflicts and aggregation conflicts.

Step 4 Compare attribute sets of C1 and C2 except semantically relevant attributes. If they are different, there is a schema isomorphism conflict.

Step 5 Compare identifier attribute sets of C1 and C2. If they are not semantically relevant attributes, there is an entity identifier conflict.

Step 6 Compare reference sets of C1 and C2. If they are not semantically relevant, there is a relation conflict.

Step 7 Compare constraint sets of C1 and C2. If they are not semantically relevant, there is a restriction conflict.

2.3.3 Detecting data-level conflicts

Given two attributes A1 and A2, we detect data-level conflicts using the following operations.

Step 1 Compute their relevance degree.

Step 2 If A1 is semantically relevant to A2, build trees for them. If not, go to step 4.

Step 3 Compare all the characteristics of A1 and A2, such as data-type, data-unit, data-precision, data-reliability, and data-range. If they are different, we can obtain a conflict of the corresponding type.

Step 4 If A1 and A2 have the same or similar names but they are not semantically relevant, there is a data value conflict.

2.3.4 Reporting conflicts

After detecting semantic conflicts, the last thing that the conflict detector needs to do is to report the conflict list to the coordinator. The conflict list includes all conflicts detected by conflict detector with all their types and concepts in it.

2.4 Conflict solver

The conflict solver has a rule base and several processors. In the rule base, there are rules for conflict solving, for example, {rule: price→rent charge | information category = "rent"} means if "rent charge" in a query interface is a naming conflict with regards to "price" when the information category is "rent", then "rent charge" matches "price". Every type of conflict has an appropriative processor, and by using the rules in the rule base, the processors can obtain a mapping table easily and quickly.

3 Conclusion

There are several advantages in using ontology and semantic conflict to resolve schema matching problems in web query interface integration, which can improve the efficiency of information retrieval, facilitate the sharing and reuse, and provide a flexible and convenient interactive platform for information users. Through the analysis of various types of semantic conflicts, an ontology-based schema matching method is introduced, thus the semantic conflicts in web query interfaces can be identified and given to a conflict solver to be resolved, then a mapping table is gained. The main task in the future is to enhance the reliability and to improve its efficiency.

References

[1] Chang Kevin Chen-Chuan, He Bin, Li Chengkai, et al. Structured databases on the web: observations and implications [J]. *SIGMOD Record*, 2004, **33**(3): 61 - 70.

[2] Yan Zhongmin, Li Qingzhong, Dong Yongquan, et al. An ontology-based integration of web query interfaces for house search [C]//*Proceedings of the* 2008 *IEEE International Conference on Information and Automation*. Zhangjiajie, China, 2008: 190 - 194.

[3] Wu Shengli, Wang Nengbin. A method to support semantic discrepant data in heterogeneous distributed database systems [J]. *Chinese J Computers*, 1996, **19**(5): 363 - 368.

[4] Ram S, Park Jinsoo. Semantic conflict resolution ontology (SCROL): an ontology for detecting and resolving data and schema-level semantic conflicts [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2004, **16**(2): 189 - 202.

[5] W3C. OWL web ontology language [EB/OL]. (2004-02-10) [2008-04-10]. http://www. w3. org/TR/owl-features.

[6] Lu R Q, Jin Z. *Domain-modeling-based software engineering* [M]. Kluwer Academic Publishers, 2000: 1 - 347.

[7] Chen Gang, Lu Ruqian, Jin Zhim. Constructing virtual domain ontologies based on domain knowledge reuse [J]. *Journal of Software*, 2003, **14**(3): 350 - 355.

Web 查询接口集成中基于本体的模式匹配方法

闫中敏　李庆忠　曹鲁慧　孔兰菊　董永权　丁艳辉

(山东大学计算机科学与技术学院,济南 250101)

摘要: Deep web 数据集成需要对 web 查询接口进行模式匹配并获得映射关系. 在 web 查询接口集成中引入语义冲突的概念,通过分析语义冲突的起源和分类,提出了一种基于本体的模式匹配方法. 以房产领域的 web 查询接口集成为实例,详细阐述了这种方法的具体过程:通过比较语义相似度自动检测不同查询接口之间存在的语义冲突,识别冲突类别并且给冲突解决器发送消息,冲突解决器借助领域专家定义推理规则来消除冲突获得映射表. 使用检测和解决语义冲突的方法来进行模式匹配,算法简单易于实现,扩充本体定义就可以使用于不同领域,灵活性和重用性较好.

关键词:deep web;web 查询接口集成;语义冲突;模式匹配;本体

中图分类号:TP31