第33卷 第23期 计 算 机 工 程 2007年12月
Vol.33 No.23 Computer Engineering December 2007
·软件技术与数据库· 文章编号：1000—3428(2007)23—0090—04 文献标识码：A 中图分类号：N945

一种基于 Ontology 的数据集成系统

陈 遥[1]，李 珊[2]，厉 浩[3]
（1. 南京信息工程大学计算机与软件学院，南京 210044；2. 南京医科大学计算机教研室，南京 210029；
3. 江苏省招生办公室，南京 210024）

摘 要：针对异构数据源中实现基于语义的数据集成的需求，采用分层思想，在用户层与实际数据层之间增加一个中间层来屏蔽底层数据的异构，用本体作为公共语义描述工具，建立本体到各数据源的映射规则，设计实现了将基于本体的全局查询转化为基于各数据源的局部查询系统，解决了数据源之间的语义异构问题，其中用 OWL 表示本体和映射，并研究了系统中的关键技术。
关键词：语义异构；分层；本体；数据集成

Data Integration System Based on Ontology

CHEN Yao[1], LI Shan[2], LI Hao[3]
（1. Computer and Software Institute, Nanjing University of Information Science & Technology, Nanjing 210044; 2. Computer Department, Nanjing Medical University, Nanjing 210029; 3. Department of Recruit Students in Jiangsu, Nanjing 210024）

【Abstract】Due to the demand of integration of heterogeneous data sources, a semantic-based data integration system with three layers is designed, in which a middle layer between user layer and data layer is added to hide heterogeneity of various data sources. In practice, this paper introduces the ontology used as a common semantic description, and establishes the mapping between the ontology and heterogeneous data sources, and applies Web ontology language(OWL) to describe the ontology and the mapping. A heterogeneous data integration system, which can translate ontology-based global query into data-sources-based local query, is designed and realized to resolve the problem of semantic heterogeneity. Key technologies are discussed.
【Key words】semantic heterogeneity; layers; ontology; data integration

计算机技术的不断发展使许多企业和单位实现了信息的计算机管理，极大地提高了工作效率，同时累积了大量的数据。这些数据源分布于各处，它们的形成是为了满足各自的业务需要，因此，不同的数据源之间往往存在着异构，相互间难以集成和共享。随着网络技术的高速发展，越来越多的企业和单位都在推进其信息化的进程，因此，在这些信息系统中实现数据集成和共享的需求就很迫切。

数据的异构性[1]主要表现在以下 3 个方面：(1)不同数据库系统之间的数据是异构的(如Oracle与SQL Server)；(2)不同结构的数据之间是异构的(如结构化的SQL Server数据库数据和半结构化的XML数据)；(3)数据表示的语义上存在差异(如不同数据源之间存在同名异义、同义异名，以及所使用的概念的抽象)，即语义异构。

数据异构的前 2 个方面属于语法异构问题，基于 XML 的数据集成系统能很好地解决，即将各数据源都转化为 XML 数据模式。但 XML 数据模式不能表示数据源的语义，无法处理语义异构的问题。为了能实现语义层上的数据集成，引入了本体(ontology)。本文提出一种基于 ontology 构建领域信息的模型，即利用本体来获取某一领域的知识，描述该领域的概念以及概念之间的关系。

1 系统设计

本文采用分层思想设计系统模型，如图 1 所示。该模型的特点在于：对于用户(如应用程序等)，它实现数据操作(主要是查询)的透明性，即用户不用知道实际访问的数据源的位置、结构或存储方式等具体的资源信息，只须通过统一的方式(即虚拟的全局数据模式)对数据进行查询等操作。因此，在用户层和实际的数据源层之间应该有一个中间层，用以实现前 2 层的交互、屏蔽数据层中各数据源的异构情况。在分层模型中，交互操作只在相邻层之间进行。因此，中间层是本模型的设计重点。

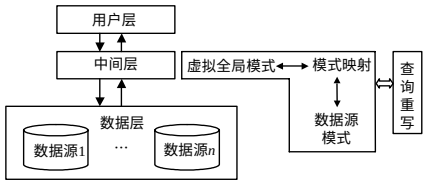

图 1 数据集成系统模型

由图 1 可以看出，该集成系统通过全局数据模式(呈现给用户一个虚拟的全局数据库)将各种数据源的数据集成起来，单实际的数据仍存储在局部数据源中，中间层通过各数据源的包装器对数据进行转换，使之符合全局模式。用户层的查询基于全局模式，不必知道每个数据源的特点，由中间层将基于全局模式的查询转换为基于各局部数据源模式的查询(即查询重写)，中间层中的查询执行引擎再通过各数据源的包装器将结果取出，最后中间层将结果集成并返回给用户。

基金项目：南京信息工程大学科研基金资助项目(y617)；南京医科大学科研基金资助项目(06nmum007)
作者简介：陈 遥(1977－)，女，硕士、讲师，主研方向：计算机应用技术；李 珊，博士、讲师；厉 浩，硕士、工程师
收稿日期：2007-07-06 **E-mail：**chenyao0077@163.com

自 20 世纪末，利用本体作为语义描述已逐渐形成一个新思想。本文采用本体作为全局模式语言，使用户层面对由本体表示的全局数据的查询，再由中间层的查询重写功能实现对各实际数据源的子查询。另外，本体的推理功能可以很好地解决可能出现的数据语义冲突问题。因此，本体是屏蔽各数据源之间的语义差异、连接各数据源、直接面对用户的枢纽。

2 系统的实现

根据上文的分析，该系统需要解决以下 3 个问题：(1)数据源描述；(2)本体的构建与表示；(3)查询重写功能，主要包括建立本体到数据源的映射规则及一个有效的查询引擎(描述自定义的查询语句如何通过本体转化为数据源中的词汇，实现对实际数据源的查询)。

2.1 数据源模式

数据源有多种形式，但都有相应的工具可以转化为XML数据源结构，因此，XML可作为异构数据源结构交互的标准[2]。数据源模式是对各异构数据源的描述。XML Schema可以和XML命名空间相结合，本文采用XML Schema作为各数据源模式的表示方式。

XML Schema 中有 2 个基本组成部分：元素和属性。元素又分为简单类型(不包含其他元素且没有属性)和复杂类型(包含其他元素或有属性)。一个结构良好的 XML 文档包括 2 种层次关系：元素和属性的关系，元素和子元素的关系。

本文以大学机构中科研成果的论文产出为例，2 个不同结构数据源 s1 和 s2 的 XML Schema 分别为 articlelist.xsd 和 journallist.xsd，表示如下：

(1)articlelist.xsd

```
<?xml version='1.0'encoding='GB2312'?>
<schema   targetNamespace="file:///d:/tt/test/articlelist.xsd#"
          xmlns="http://www.w3c.org/2001/xmlschema"
          xmlns:xmp="file:///d:/tt/test/articlelist.xsd#">
    <element name="articlelist" >
        <complextype>
        <element name="article" type="xmp:articletype"/>
         </complextype>
    </element>
    <complextype name="articletype">
    <element name="title"       type="string"/>
    <element   name="name"      type="string"/>
   <element name="volume"      type="string"/>
    <element name="pages"       type="integer"/>
    <element name="year"        type="date"/>
    <element name="price"       type="float"/>
    <element name="injournal" type="string"/>
    </complextype>
</schema>
```

(2)journallist.xsd

```
<?xml version='1.0' encoding='GB2312'?>
<schema         targetNamespace="file:///d:/tt/test/booklist.xsd#"
        xmlns="http://www.w3c.org/2001/xmlschema"
        xmlns:xmp="file:/// d:/tt/test/journallist.xsd#">
        <element name=journallist" >
            <complextype>
                <element name="book"
type="xmp:journaltype"/>
            </complextype>
        </element>
        <complextype name="journaltype">
         <element name="name"        type="string"/>
         <element name="editor"     type="string"/ >
         <element name="department" type="string"/>
         <element name="address"      type="string/">
         <element name="price"        type="float/"/>
         <element name="issn"       type="string"/>
         <element name="year"       type="date"/>
        </complexType>
</schema>
```

可以看出，数据源 s1(articlelist.xml)与 s2(journallist.xml)之间存在如下语义冲突：(1)实例层次：s1 中的 price 以美元表示，s2 中的 price 以人民币表示。(2)元素层次：s1 中的 name 表示文章的作者名，s2 中用 editor 表示文章的作者名，属异名同义；s1 中的 name 表示文章的作者名，s2 中的 name 表示文章的期刊名，属同名异义。(3)模式层次：s1 中的一个子模式 articletype/injournal 与 s3 的一个子模式 journaltype/ name 意义相同，都表示刊物名。

2.2 本体及其构建

本体[3]原是哲学上的概念，用于描述事物的本质，近几年作为信息抽象和知识描述的工具被计算机领域所采用。ontology 面向特定领域，描述特定领域的概念模型即关于该领域的一个公认的概念集，其中的概念有公认的语义，通过概念之间的关联来体现。

利用本体构建工具可以从某领域的大量资料中构建出可共享的语义资源作为本体。本体包含概念和概念之间的关系。一个专业领域中有关领域概念的本体一般可以用有向图来描述，通过节点与节点之间的层次关系，本体可让人或机器明白概念间的语义联系。

本体有 5 个基本的建模原语（或者称为本体的 5 个基本元素）：类或概念，关系，函数，公理，实例。本体图是表示本体的二元组(V, E)，其中，V 是结点集，结点对应于本体中的概念；E 是有向边的集合，有向边对应于本体中的关系。

本文构建了一个关于论文产出的部分本体：正式出版物包含书、期刊和文章；所有正式出版物都有标题、日期、作者、出版社地址和价格等属性；文章都有其所属刊物的页码；价格可以由美元、人民币等不同的度量单位表示。该本体如图 2 所示，其中，矩形表示类；椭圆表示属性；箭头表示关系；线上的词表示关系的名称。

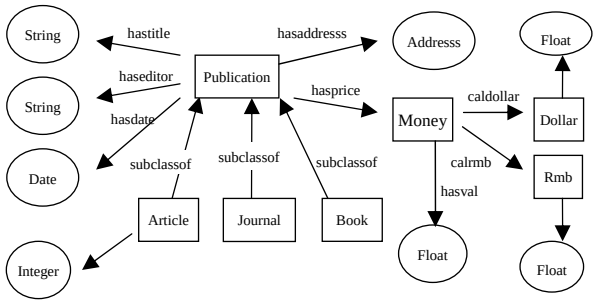

图 2 科研成果的部分本体

OWL(Web ontology language)[4]是W3C推荐的语义互联网中本体描述语言的标准。本文采用OWL表示本体，表示如下：

```
    <RDF:RDF
xmlns:RDF=http://www.w3.org/1999/02/22-RDF-syntax-ns#
xmlns:RDFs="http://www.w3.org/2000/01/RDF-schema#"
```

```
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns="file:///d:/tt/test/research.owl"
xml:base= "file:///d:/tt/test/research.owl" >
    <owl:Ontology RDF:about="research"/>
    <owl:Class RDF:ID=" Publication "/>
    <owl:Class RDF:ID="Article">
        <RDFs:subClassOf RDF:resource="#Publication"/>
    </owl:Class>
    <owl:Class RDF:ID="Book">
        <RDFs:subClassOf RDF:resource="#Publication"/>
    </owl:Class>
    <owl:Class RDF:ID="Journal">
        <RDFs:subClassOf RDF:resource="#Publication"/>
    </owl:Class>
    <RDF:Property RDF:ID="hastitle">
        <RDFs:domain RDF:resource="#Publication"/>
        <RDFs:range RDF:resource="#String"/>
    </RDF:Property>
    <RDF:Property RDF:ID=" hasdate ">
        <RDFs:domain RDF:resource="#Publication"/>
        <RDFs:range RDF:resource="#Date"/>
    </RDF:Property>
…
</RDF:RDF>
```

系统利用Jena[5]作为底层OWL本体的解析和推理引擎。Jena是惠普实验室用Java开发的一个开放源代码项目，它为处理RDF、RDFS、DAML和OWL提供了一个API编程环境，包括一个基于规则的推理引擎。系统可以通过Jena调用引擎来使用、维护本体。同时，本体到数据源的映射也由OWL表示，在处理映射的算法中可以利用引擎取得子概念、子属性，并能检查映射的一致性等。

2.3 映射规则的构建

在数据源模式和全局模式都基于树结构的情况下，全局模式到数据源模式间的映射，最好是通过 2 个模式间的XPath路径来进行[5]。本文的本体是一个有向图，可以看作是多棵树组成的森林，因此，映射的方法是将本体森林中某棵树中的路径映射到数据源模式中的XPath路径，并用OWL表示本体到数据源的映射规则。

为了解决异构数据中的语义冲突问题，应将本体中的概念或概念路径映射到数据源的元素、属性或模式路径。XML Schema 可以看成一棵树，每个元素或属性都是树中的一个结点，并确定了一条从树根到该节点的 XPath 路径，每个元素或属性都对应一条模式路径，因此，映射规则可表示为概念或概念路径到模式路径的映射。由于 OWL 推理引擎能直接解析单个概念或单种关系，因此利用本体到数据源的映射来解决语义问题的总体思路如下：若数据源 S1 中的“ X1 ”与数据源 S2 中的“ X2 ”分别是本体中的概念 C1 和概念 C2 的映射，则通过本体的推理引擎可以得到 C1 与 C2 的语义关系，从而推出 X1 和 X2 的关系。

OWL 的语言组成包括类、属性、实例、数据类型和一些注释信息。本体模型中的“ 概念 ”和“ 关系 ”分别对应 OWL 中的“ 类 ”和“ 属性 ”。

在 OWL 中，基本的映射关系是类、属性或实例间的相等和蕴含关系，具体有 equivalentClass, equivalentProperty, subClassOf, subPropertyOf, sameIndividualAs 等。分析本体的各组成部分和 XML Schema 各部分的含义后，可以建立以下基本映射规则：(1)OWL 本体中的类映射为 XML Schema 中的复杂类型；(2)OWL 本体中的数据类型映射为 XML Schema 中的简单类型；(3)若 OWL 本体中的属性代表了到其他类的关系，则映射为 XML Schema 中的子元素；(4)若 OWL 本体中的属性代表了到简单类型的关系，则映射为 XML Schema 中的属性。

现采用上述映射方法来表示本文例子中的本体到数据源映射规则集，其中，规则 R1 为: s1_articletype→Article，表示数据源 s1 的 articletype 类型映射为本体中的 Article 类。其余规则类似，用 OWL 表示如下：

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns
        #" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns="file:///d:/tt/test/mapping.owl"
      xml:base= "file:///d:/tt/test/mapping.owl">
  <owl:Ontology rdf:about="mapping"/>
  <owl:Class rdf:ID="s1_articletype">
    <owl:equivalentClass rdf:resource="#Article/>
  </owl:Class>
 <rdf:Property rdf:ID="s1_articletype.pages">
    <owl:equivalentProperty rdf:resource="#hasPages"/>
  </rdf:Property>
 <rdf:Property rdf:ID="s1_articletype.title">
    <owl:equivalentProperty rdf:resource="#hasTitle"/>
  </rdf:Property>
    …
 <rdf:Property rdf:ID="s2_journaltype ">
    <owl:equivalentProperty rdf:resource="#Journal"/>
  </rdf:Property>
  <rdf:Property rdf:ID="s2_journaltype.name">
    <owl:equivalentProperty rdf:resource="#hasTitle"/>
  </rdf:Property>
  <rdf:Property rdf:ID="s2_journaltype.editor">
    <owl:equivalentProperty rdf:resource="#hasEditor"/>
  </rdf:Property>
 </rdf:RDF>
```

用上述方法来解决 2.1 节中数据源 s1 和 s2 中引起的语义冲突：

(1)实例层次

s1 中以美元表示价格，而 s2 中以人民币表示价格，因此出现了域冲突。解决方法：将本体中的概念 Dollar 和概念 Rmb 分别映射到 s1 和 s2 中的 price，通过在本体中取得属性 caldallor 和 calrmb 的值(即换算汇率)，将价格都换算成以统一的价格尺度 Money 表示的价格。另外，s1 中的元素 name 的英文表示“ lili ”与其中文表示“ 李莉 ”之间用 sameIndividualAs 表示同义关系。

(2)元素层次

1)s1 中的作者名用 name 表示，s2 中的作者名用 editor 表示，属异名同义。将本体中关系 haseditor 映射到 s1 中的 name 以及 s2 中的 editor，这样“ name ”和“ editor ”就能被识别为同义元素。

2)s1 中的 name 表示文章的作者名，s2 中的 name 表示表示期刊名，属同名异义。将本体中的关系 haseditor 映射到 s1

中的 name，将关系 hastitle 映射到 s2 中的 name；而在本体中 haseditor 和 hastitle 并无相等或继承关系，它们是异义的，因此，s1 中的 name 和 s2 中的 name 被识别为异义元素。

(3)模式层次

s1 中的子模式 articletype/injournal 与 s2 的子模式 journaltype/name 虽是表示完全不同的 2 个模式，但通过将本体中的概念路径 Journal.hasTitle 分别映射到这 2 个子模式，可以将它们识别为意义相同的模式，都表示期刊名。

2.4 查询的实现

用户是采用基于 OWL 本体的查询语句，查询语言的设计与 SQL 类似，可参考文献[7]。例如，下面的查询语句表示“作者为李莉的正式出版物的标题”(带“？”的为变量):

```
Select ？a
From Publication=？b, ？b.hastile=？a, ？b.haseditor=？c
Where ？c="李莉"
```

先由查询引擎负责将基于本体的查询语句转化为对应数据源的子查询。该任务须调用系统的其他模块，即利用映射规则转化为相应的查询。上述查询语句除了变量外，其余部分都能直接利用映射转化为数据源的查询，因此，只需要把这些变量绑定(匹配)各数据源的相应规则即可。其具体实现主要分为：(1)查询的解析，即对查询语句作语法分析，抽取出关键信息(变量名、变量所代表的模式路径的值)，建立该语句的查询树；(2)查询变量绑定的实现，即将树中的变量匹配到各个数据源的相应规则，实现查询变量的绑定；(3)查询的分解，即根据最大绑定关系的定义，对原始查询进行重写，其中需要多次进行连接(join)和联合(union)的递归运算，形成新的查询语句；(4)查询转化，即系统把查询语句转化为对 XML 数据源的 XQuery 语句；(5)查询语句的执行并得到结果等。

3 结束语

本文采用分层的思想进行异构数据的集成系统设计，以实现数据访问的透明性。并引入本体技术解决语义冲突问题。另外，利用本体进行数据集成是为了借助本体进行推理，因为传统数据模型只能表示概念间显式存在的关系，无法从现有的已知事实中推断出其他的结论，而在异构分布的环境中，多个数据源所涉及的概念间的关系往往是隐含的，通过本体可以明确表示它们之间的关系。一旦本体定义了概念间重要的关系，推理引擎就能自动地利用这些关系进行推理。

参考文献

1 Wache H, Vogele T, Visser U, et al. Ontology-based Integration of Information — — A Survey of Existing Approaches[C]//Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing. 2001.

2 厉 浩. 基于本体的异构数据集成的研究[D]. 南京：东南大学, 2005.

3 Gruber T. A Translation Approach to Portable Ontology Specifications[J]. Knowledge Acquisition, 1993, 5 (2): 199-220.

4 Antoniou G, Harmelen F V. Web Ontology Language: OWL[M]// Handbook on Ontologies in Information Systems. [S. l.]: Springer-Verlag, 2003.

5 Jena — — A Semantic Web Framework for Java[Z]. [2006-10]. http://jena.sourceforge.net/.

6 Cluet S, Veltri P, Vodislav D. Views in a Large Scale XML Repository[C]//Proceedings of the 27th VLDB Conference, Roma, Italy. 2001.

7 王洪伟, 吴家春, 蒋 馥. 基于本体的元数据模型及其 DAML 表示[J]. 情报学报, 2004, 23(2).

（上接第 64 页）

$$T_r = \min\{T_{A1}, T_{12}, T_{23}, \cdots, T_{nB}\}$$

其中，T_{A1}表示A对第 1 个推荐者的信任值；T_{12}表示第 1 个推荐者对下一个推荐者的信任值。

(1)当存在 k 条路径，且不相关时，推荐综合信任值为

$$T_f = [\frac{1}{k}\sum_{i=1}^{k} T_{ri}]$$

其中，[]表示向下取整。

(2)当 k 条路径是相关的，存在 m 条路径是不相关的，$m < k$，推荐综合信任值为

$$T_f = [\frac{1}{m}\sum_{i=1}^{m} T_{ri}]$$

3 结束语

本文提出的普适计算信任管理与传统信任管理不同，它是基于策略的信任管理和基于声誉的信任管理的联合。最初的信任主要基于对方是否具有某些属性的信任书，同时也考虑了其他方的推荐。该信任管理将普适计算的信任分解为静态信任T_a和动态信任T_c，T_a取决于属性信任、推荐信任、经验，T_c取决于上下文信任。这种联合的信任结构能够较好地体现信任的动态性以及陌生主体之间最初信任问题。

参考文献

1 Blaze M, Feigenbaum J, Lacy J. Decentralized Trust Management[C]// Proceedings of the 17th Symposium on Security and Privacy. 1996: 164-173.

2 Ruohomaa S, Kutvonen L. Trust Management Survey[C]//Proceedings of the 3rd International Conference on Trust Management (iTrust). 2005: 77-92.

3 Mitchell J C. RT: A Role Based Trust Management Frame- work[C]// Proc. of the 3rd DARPA Information Survivability Conference and Exposition. 2003: 201-212.

4 Carbone M, Nielsen M, Sassone V. A Formal Model for Trust in Dynamic Networks[C]//Proceedings of International Conference on Software Engineering and Formal Methods. 2003: 54-61.

5 Jøsang A, Pope S. Semantic Constraints for Trust Transitivity[C]//Proc. of the 2nd Asia-pacific Conference on Conceptual Modeling. 2005: 59-68.

6 Kinateder M, Baschny E, Rothermel K. Towards a Generic Trust Model-comparison of Various Trust Update Algorithms[C]//Proc. of the 3rd International Conference on Trust Management (iTrust). 2005: 177-192.

7 Winslett M, Yu T, Seamons K, et al. Trust Negotiation on the Web[J]. IEEE Internet Computing, 2002, 6(6): 30-37.

8 Hess A, Holt J, Jacobson J. Content-triggered Trust Negotiation[J]. ACM Transactions on Information and System Security, 2004, 7(3): 428-456.

9 洪 帆, 郭亚军. 资源限制信任协商[J]. 华中科技大学学报(自然科学版), 2006, 34(5): 23-25.