第33卷 第13期 计 算 机 工 程 2007年7月
Vol.33 No.13 Computer Engineering July 2007
·博士论文· 文章编号：1000—3428(2007)13—0010—03 文献标识码：A 中图分类号：TP311.5

基于 Ontology 映射的异构信息源查询处理

张 磊，谢 强，吴笑凡，丁秋林，黄添强
(中国矿业大学计算机学院信息科学系，徐州 221008)

摘 要：企业中存在大量语义异构数据源，语义异构阻碍了数据源的查询处理。该文针对这个问题，提出了基于 Ontology 的语义集成方法，给出了语义异构信息源的查询处理。通过实例验证了方法的可行性。某航空研究所应用表明：该方法可以有效地处理企业中存在的异构数据源查询问题。

关键词：语义异构；Ontology 映射；查询处理

Query Processing on Semantically Heterogeneous Information Sources Based on Ontology Mapping

ZHANG Lei, XIE Qiang, WU Xiaofan, DING Qiulin, HUANG Tianqiang
(Department of Science Information, School of Computer, China Mining University, Xuzhou 221008)

【Abstract】 Semantic heterogeneity baffles query processing on information sources in the enterprise where large numbers of semantically heterogeneous information sources exist. The method of semantic integration based on Ontology mapping is put forward and query processing on semantically heterogeneous information sources is specified aiming at the problem in the paper. An example is given to validate the method. Application in a certain aeronautical institute shows the method can solve the problem effectively.

【Key words】 semantic heterogeneity; Ontology mapping; query processing

某航空研究所作为我国重点型号研制单位，在长期发展过程中，积累了大量的信息，这些信息以不同的形式存在，如数据库、知识基和文档等。采用不同的词汇和概念模型，采用不同的分类标准，由不同人员单独开发而成。这些信息源使用的最大问题是语义异构，极大地阻碍了企业的应用。

多信息源语义异构现象在其他企业也普遍存在。针对这个问题，本文提出了基于 Ontology 映射的语义集成方法，给出了语义异构信息源的查询处理，应用实例验证了该方法的可行性。

1 基于 Ontology 映射的语义集成方法

1.1 Ontology

Ontology已经用于多个领域[1]，在计算机科学领域，Ontology由Gruber释义为“概念化的显式说明”[2]，提供表示和交流领域知识的词汇，在概念层次上提供包含词汇术语的关系集合。Ontology 能够描述信息源的语义，可用于语义集成，通过Ontology对信息源语义进行显式说明。现有的基于Ontology的语义集成方法基本上可分为单Ontology方法、多Ontology方法、混合方法等 3 类[3]。

1.2 基于 Ontology 映射的语义集成

本文提出的基于 Ontology 映射的语义集成方法，与 1.1 节混合方法相类似，不同之处在于：局部 Ontology 直接采用信息源中的术语进行定义，而不是采用共享术语定义；通过定义局部 Ontology 之间的映射来生成全局 Ontology 和全局-局部 Ontology 映射，通过全局-局部 Ontology 之间的映射实现信息源的语义集成。

基于 Ontology 映射的语义集成(框架见图 1)主要包括：

(1)信息源。信息源主要是指多个语义异构的信息源；

(2)局部 Ontology。每个信息源采用一个局部 Ontology 描述。局部 Ontology 根据每个信息源的术语和术语结构，单独构建；

(3)全局 Ontology。全局 Ontology 由各局部 Ontology 合并获得，通过将各局部 Ontology 合并，获得全局 Ontology 和全局-局部 Ontology 之间的映射。全局 Ontology 提供用户统一术语查询，为用户提供统一词汇。用户能够通过提交基于全局 Ontology 的查询，实现语义异构信息源的查询；

(4)Ontology 映射。Ontology 映射包括 2 部分：一部分是局部 Ontology 之间的映射，另一部分是全局-局部 Ontology 之间的映射。局部 Ontology 之间的映射是在建立局部 Ontology 之后手动建立的，全局-局部 Ontology 之间的映射是在局部 Ontology 的合并过程中自动生成的。

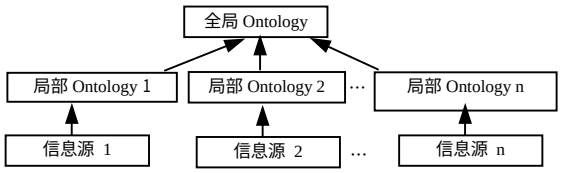

图 1 基于 Ontology 的语义集成框架

为有效说明基于 Ontology 映射的语义集成方法，将基于 Ontology 映射的异构信息源集成系统定义如下：

基金项目：国防重大基础预研基金资助项目
作者简介：张 磊(1977－)，男，博士研究生，主研方向：企业信息化，知识管理；谢 强，博士、讲师；吴笑凡，博士研究生；丁秋林，教授、博士生导师；黄添强，博士研究生
收稿日期：2006-07-21 **E-mail：**zhanglei_zyx@163.com

定义 1(基于 Ontology 映射的异构信息源集成系统描述为元组 $\Gamma=(O_g,O_l,D,M_{ll},M_{gl})$) 元组 Γ 中，$O_g$ 为全局 Ontology；O_l 为局部 Ontology 集合，$O_l=(O_1,O_2...,O_n)$；D 为多个信息源集合，$O_l=(O_1,O_2...,O_n)$。由于信息源是语义异构的，每个信息源 D_i 具有一个局部 Ontology O_i；M_{ll} 表示局部 Ontology 之间的映射；M_{gl} 表示全局-局部 Ontology 之间的映射。

基于 Ontology 映射的语义集成中，其关键是 Ontology 映射，包括局部 Ontology 之间的映射和全局-局部 Ontology 之间的映射。

文献[4]采用桥Ontology描述多Ontology之间的映射[4]，本文从Ontology在语义集成中的作用考虑定义Ontology。

定义 2(Ontology) Ontology 为元组 $O=(C,A^C,R,H)$，其中，C 为概念集合；A^C 为概念的属性集合；R 为关系集合；H 为表示概念层次。

若 $c_i\in C$，c_i 属性记为 $A^C(c_i)$；R 中的关系 r_i 表示为 $<c_p,c_q>$，表示概念 c_p,c_q 之间的二元连接关系；若 c_p 为 c_q 的父概念，则 $<c_p,c_q>\in H$。

Ontology 映射表示为两个 Ontology 概念之间的映射和属性之间的映射。

定义 3(Ontology 映射) 设 O_i,O_j 为两个 Ontology，其映射 $M_{ij}=<M_{ij}^{C},M_{ij}^{A}>$，$M_{ij}^{C}$ 为概念之间的映射集合，设 x,y 分别为 O_i,O_j 中的概念，$M_{ij}^{C}(k)$ 为第 k 个概念映射，表示为 $<x,y,\Xi>$，Ξ 取值为

$\subseteq$ 表示 O_i 的概念 x 比 O_j 的概念 y 更特殊；

$\supseteq$ 表示 O_i 的概念 x 比 O_j 的概念 y 更泛化；

$\equiv$ 表示 O_i 的概念 x 和 O_j 的概念 y 等价；

$\perp$ 表示 O_i 的概念 x 和 O_j 的概念 y 重叠。

M_{ij}^{A} 表示为属性之间的映射集合，属性映射表示为 $<x,a,y,b>$。属性映射附属于概念映射之上，即 if $<x,a,y,b>$ then $(x,y,\Xi)\in M_{ij}^{C}$

定义 4(局部 Ontology 映射和全局-局部 Ontology 映射) 局部 Ontology 映射 $M_{ll}=\{M_{ij}\mid O_i,O_j\in O_l,i\neq j\}$；全局-局部 Ontology 映射 $M_{gl}=\{M_{gj}\mid O_j\in O_l\}$。

基于Ontology映射的语义集成中，局部Ontology映射是由手工指定的，全局Ontology采用PROMPT合并算法[5]生成，并生成全局-局部Ontology映射。

2 查询处理

语义异构信息源的查询处理主要包括：全局查询形成，全局查询分解，子查询执行和查询结果集成等过程。

2.1 全局查询

全局查询为类 SQL 格式的查询，查询中的概念和属性来源于全局 Ontology。用户利用全局查询检索语义异构信息源的信息。全局查询定义如下：

定义 5(全局查询) 设 Q 为全局查询，定义为元组<S, F, W>，其中：

$S=\{gc_i.gp_{ij}\mid\forall i=1,...,m,\forall j=1,...,m\}$ 为 SELECT 子句，gc_i 为 O_g 中的概念，gp_{ij} 为 gc_i 的属性，即 $gp_{ij}\in A^C(gc_i)$；$F=\{gc_i\mid\forall i=1,...,n\}$ 为 FROM 子句，gc_i 为 O_g 中的概念；$W=W_{con}\cup W_{var}$ 为 WHERE 子句，WHERE 子句分为 W_{con}、W_{var} 两类；$W_{con}=\{w_{coni}\mid i=1,...,s\}$，$w_{coni}=gc_k.gp_{kl}\Theta Const$，表示将概念属性同常量相比较；$W_{var}=\{w_{vari}\mid i=1,...,t\}$，$w_{vari}=gc_k.gp_{kl}\Theta gc_p.gp_{pq},k\neq p$ 表示将概念属性相比较。$\Theta\in\{=,<,>,\neq,\leq,\geq\}$。

2.2 全局查询分解

通过全局-局部 Ontology 映射，将全局查询分解为多个子查询，每个子查询对应一个信息源。

定义 6(全局查询分解)全局查询Q分解为子查询q_1,…, q_n，每个子查询可采用单独的信息源 $D_1,...,D_n$ 进行回答，将各子查询的答案进行组合即可获得全局查询Q的回答。

全局查询分解包括概念属性映射和查询分组 2 个步骤。

2.2.1 查询概念属性映射

将全局查询分解中的概念、属性分别通过全局-局部 Ontology 映射映射到局部 Ontology 中，形成单个数据源对应的术语。映射包括概念的映射和属性的映射，查询概念属性映射算法如下：

算法 查询概念属性映射算法

输入 $c_g,c_g.p_g$，M_{gl},其中，c_g 全局查询中的概念；$c_g.p_g$ 为全局查询中的概念属性；M_{gl}为全局-局部 Ontology 映射。

输出 $CSet=\{CL_i\mid i=1,...p\}$，$PSet=\{CP_j\mid j=1,...,q\}$，$CL_i$ 定义为元组 $<c_g,c_l,O_l,D_i>$，其中，c_l 为局部 Ontology O_l 中的概念；D_i 为 O_l 对应的信息源；CP_j 定义为元组 $<c_g.p_g,c_l.p_l,O_i,D_i>$，其中 $c_l.p_l$ 为局部 Ontology 中概念 c_l 的属性 p_l。

```
Begin
CSet = φ;
PSet = φ;
For each O_i ∈ O_l
    For each M_gi^C(k) ∈ M_gj^C
        If M_gi^C(k).x = c_g
            CSet = CSet ∪ {(c_g, y, O_j, D_j)};
        EndIf
    For each M_gi^A(k) ∈ M_gi^A
        For each CL_i ∈ CSet
            If (M_gi^A(k).x = Cl_i.c_g) AND (M_gi^A(k).y = Cl_i.c_l)
               AND (M_gi^A(k).a = c_g.p_g)
             PSet = PSet ∪ {(c_g.p_g, c_l.p_l, O_i, D_i)}
            EndIf
        EndFor
    EndFor
EndFor
    End
```

将全局查询查询中的概念和属性通过上述映射算法映射为局部Ontology的概念和属性。对于FORM和SELECT子句可直接进行映射。对于WHERE子句，W_{con}直接进行映射；而W_{var}中，对于对应同一信息源的，进行映射；对于不对应同一信息源的，不做转换，在查询结果集成时进行处理。

2.2.2 查询分组

查询分组将全局查询概念属性映射后产生的新的查询按照所属信息源的不同分解到各个信息源上，以形成针对各个信息源的子查询。

2.3 子查询执行和结果集成

由于不同的信息源采用不同的查询语言，子查询执行必须消除查询语言异构。为消除查询语言异构，每个子查询都转换成和数据源对应的数据查询语言，该转换通过 Wrapper 完成。子查询返回的查询结果采用 Wrapper 转换为统一的数据模型。

定义 7(子查询结果集成) 给出由全局查询Q产生的子查

询 $q_1, q_2, ..., q_n$，设 $R(q_1), R(q_2), ..., R(q_n)$ 为每个子查询返回的结果，子查询结果集成描述为 $QA = \Delta(R(q_i)),\ i = 1, ..., n$，操作 Δ 表示集成过程。

子查询结果集成中，每个子查询的查询结果通过 Wrapper 转换为统一形式，将各子查询结果集成。在集成过程中需要消除名字、数据类型、缩放冲突等。

3 应用实例

3.1 语义异构实例

某航空研究所存在大量语义异构信息源，现取其中 3 个信息源作为实例。这 3 个信息源分别为关系数据库、XML 数据、OO 数据库。

(1)信息源 D_1 是以关系数据表示的员工基本信息数据库包括相互关联的两个表：基本信息表和部门信息表，如表 1、表 2 所示。

表 1 员工基本信息表

字段名	类 型
Sta_Id	String
Sta_Name	String
Salary	Double
Dep_Id	String

表 2 部门基本信息表

字段名	类 型
Dep_Id	String
Dept_Name	String
Addr	String
Par_Dep	String

(2)信息源 D_2 是以 XML-DTD 表示的工程人员信息源，该信息源的 XML-DTD 描述如下：

```
<? Xml Version=”1.0” Standalone=”yes”>
<!DOCTYP  Engi_Info [
<!ELEMENT Engi_Info (Engi_List, Pro_List)>
<!ELEMENT Engi_List (Engineer)+>
<!ELEMNET Engineer  (Engi_Name,Title,Salary,Task)>
<!ATTLIST Engi Engi_Id ID #REQUIRED>
<!ELEMENT Engi_Name (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Salary (#PCDATA)>
<!ELEMENT Task (Proj)+>
<!ELEMNET Pro (S_Date,E_Date)>
<!ATTLIST Pro Pro_Id IDREF #IMPLIED>
<!ELEMENT S_Date (#PCDATA)>
<!ELEMENT E_Date (#PCDATA)>
<!ELEMENT Pro_List (Project)+>
<!ELEMNET Project (Pro_Name,Time)>
<!ATTLIST Project Pro_Id ID #REQUIRED>
<!ELEMENT Pro_Name (#PCDATA)>
<!ELEMENT Time (#PCDATA)>
]>
```

(3)信息源 D_3 是以 OO 表示的管理人员信息源。其数据结构如图 3 所示。

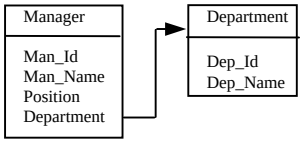

图 2 管理人员信息源结构

上述 3 个信息源存在语义异构，导致了以下语义冲突：

(1)同义冲突。3 个信息源中的 Sta_Id、Engi_Id、Man_Id 为语义等价的属性；

(2)泛化/特化冲突。概念 Staff 包含概念 Manager 和 Engineer。

3.2 局部 Ontology 和全局 Ontology

3 个数据源对应的局部 Ontology 和合并得到的全局 Ontology，如图 3 所示。

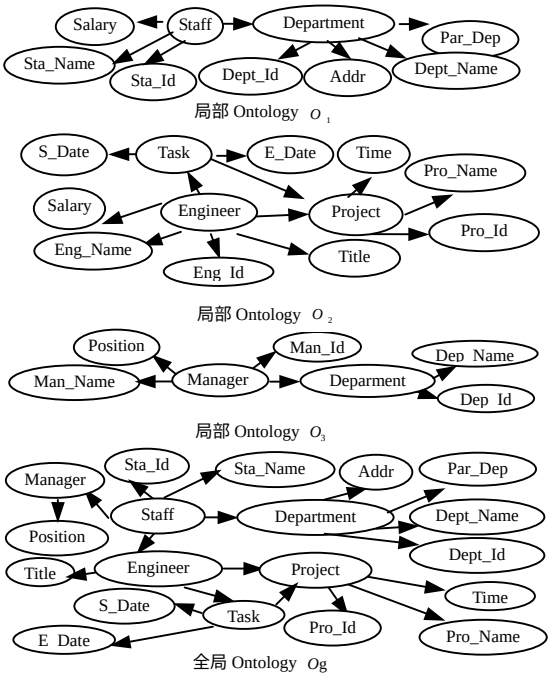

图 3 全局-局部 Ontology

全局-局部 Ontology 映射为

$M_{gl} = \{M_{g1}, M_{g2}, M_{g3}\}$;

$M_{g1} = \{(staff, staff, \equiv), (staff, staff, Sta\_Name, Sta\_Name), (staff, staff, Sta\_ID, Sta\_ID), (staff, staff, Salary, Salary), (Department, Department, \equiv), (Department, Department, Dept\_Id, Dept\_Id)(Department, Department, Dept\_Name, Depet\_Name), (Department, Department, Addr, Addr), (Department, Department, Par\_Dep, Par\_Dep)\}$

$M_{g2} = \{(Staff, Engineer, \supseteq), (Staff, Engineer, Sta\_Id, Eng\_Id), (Staff, Engineer, Sta\_Name, Eng\_Namme), (Staff, Engineer, Salary, Salary)(Engineer, Engineer, \equiv), (Engineer, Engineer, Title, Tile), (Project, Project, \equiv), (Project, Project, Pro\_Id, Pro\_Id), (Project, Project, Pro\_Name, Pro\_Name), (Project, Project, Time, Time)(Task, task, \equiv)(Task, task, S\_date, S\_Date), (Task, task, E\_Date, E\_Date)\}$

$M_{g3} = \{(staff, Manager, \perp), (staff, Manager, Sta\_Id, Man\_Id), (staff, Manager, Sta\_Name, Man\_Name), (Department, Department, \supseteq), (Department, Department, Dept\_Id, Dep\_Id), (Department, Department, Dept\_Name, Dep\_Name), (Manager, Manager, \equiv), (Manager, Manager, Position, Position)\}$

3.3 查询分解

给定一个全局查询 Q 为

```
Q=SELECT Staff.Sta_Name,Staff.Salary, Department.
Dept_Name,Engieer.title, Project.Pro_Name,Manager.
Position FROM Staff, Deparment, Engineer,Manager,
Project WHERE Staff.Salary>1500 AND
Manager.Dept_Id=Department.Dept_Id
```

利用全局-局部 Ontology 映射，全局 Ontology 可分解为

```
q1 =SELECT Staff.Sta_Name,Staff.Salary,
Dept_Name FROM Staff,
Department WHERE Staff.Salary>1500 AND
Staff.Dept_Id=Department.Dept_Id.
q2 =SELECT Engineer.Eng_Name,Engineer.
```

（下转第 21 页）