

基于本体的教务数据集成的探析

陈启买, 刘 海, 李源初

(华南师范大学计算机学院, 广东 广州 510631)

[摘 要] 高校教务信息管理系统的异构数据源之间存在着各种语义冲突. 本文结合本体和中间件方法, 通过构建教务管理领域本体, 将用户对数据集成的需求利用本体词汇进行表达, 在中间层完成全局本体到局部数据源之间的模式映射, 以处理教务数据集成的语义需求, 对于推动数字化校园建设具有现实意义和实用价值.

[关键词] 本体; 数据集成; 领域本体

[中图分类号] TP39; TP392 [文献标识码] A [文章编号] 1000-9965(2009)01-0049-05

The ontology-based data integration in education field

CHEN Qi-mai, LIU Hai, LI Yuan-chu

(School of Computer, South China Normal University, Guangzhou 510631, China)

[Abstract] Heterogeneous data sources in university Information Management System exist various semantic conflict, this paper combined of ontology and middleware method, by constructing ontology in educational administration domain, using domain ontology vocabulary to express data integration needs, and utilizing a middle layer to complete mapping between ontology vocabulary and data sources for dealing with the semantic data integration needs. This method is of practical significance and practical value for the building of digital campus.

[Key words] ontology; data integration; domain ontology

高校管理信息系统中存在放多异构的数据源, 它们由于缺乏统一信息编码规范等原因, 造成数据的语义冲突或异构, 以使信息共享困难. 如何把本体技术应用到数据化校园的数据集成环境, 解决异构系统导致的各种语义问题, 具有一定的理论意义和现实需求. 本文结合数字化校园建设需求, 首先构建高校教务信息管理领域本体, 然后利用本体描述异构数据库模式语义信息, 为数据集成应用提供全局视图; 通过本体构建异构数据库集成框架, 并建立基于该框架的原型系统, 解决了传统数据集成方法中难以解决的语义冲突问题, 对促进数字校园的数据集成与交换具有现实意义和实用价值.

1 相关研究

目前主流的数据集成方法有数据仓库、基于虚拟视图的中间件集成方法、基于 XML 的中间件集成^[1-2]、和基于本体的中间件集成^[3-5]方法等. 其中数据仓库是一种紧耦合的数据集成, 是一种“数据驱动”的模型. 其优点查询响应速度快, 可以保存大量有用的历史数据; 缺点是不能满足对数据的实时需求. 中间件方法又称虚拟视图法, 该类集成系统通常拥有一个全局模式, 查询相对于全局模式进行. G. Wiederhold 最早给出了基于中间件的数据集成方法的框架, 基于中间件的数据集成系统不仅能

够集成结构化的信息,如关系数据库,还可以集成半结构化或非结构化数据源中的信息,如 Web 信息;斯坦福大学 Garcia-Molina 等人开发了 TSIMMIS^[6] 系统,是一个最典型的中间件系统。中间件数据集成注重全局查询的处理和优化,数据仓库查询速度相对较慢,但能满足数据的实时要求;随着 XML 数据的丰富,基于中间件的数据集成系统大都采用 XML 描述数据源模式^[6],作为集成和交换的主要手段,但无法从根本上解决异构数据源模式、实例等语义引起的异构问题。随着语义及本体理论在数据集成环境中的应用^[7-8],出现了基于本体的集成系统^[9-10],它们依赖本体推理机完成从映射验证到查询分解等中间件集成系统的关键功能,既简化了开发的难度,又使得信息集成具有了智能化的功能。

2 数字化校园下的本体构建及其在数据集成中的作用

2.1 本体及高校学费管理的本体构建

本体原是哲学上的概念,用于描述事物的本质,可作为信息抽象和知识描述的工具被计算机领域所采用。它面向特定领域,描述特定领域的概念模型及概念之间的语义,通过给出相关领域词汇的基本术语和关系,以及利用这些术语和关系构成的确定词汇外延的有关规则的定义,来捕获相关的领域的知识,提供对该领域知识的共同理解。一般来说本体有 5 个基本元素,分别类或概念,关系,函数,公理,实例;本体图是表示本体的二元组 $\langle N, E \rangle$,其中 N 是结点集,对应于本体中的概念; E 是有向边的集合,对应于本体中的关系。本文首先利用本体构建学校学费管理领域的语义信息模型如图 1 所示,图中的矩形表示领域中的类,如学生类,收费标准类,缴费存单类等;而椭圆代表着与这些类有关的属性或概念。

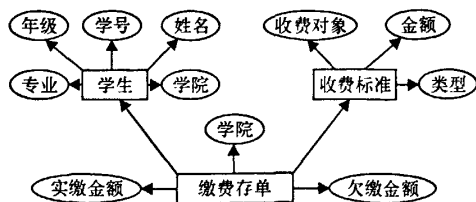


图1 利用本体构建的学费管理语义模型

2.2 本体在高校数据集成中的作用举例

(1) 本体在数据集成中的作用 利用本体,主要解决高校数据集成环境中的模式语义异构,主要是类的命名异构、属性异构和外延异构等语义冲突问题。文章采用 W3C 推荐的本体描述语言 OWL 刻

画本体和本体间关系的映射,采用各种语义异构的 OWL 解决方案如下:

类的命名异构:不同数据源使用不同术语表达同一概念,该类型异构可通过 OWL 的语义表达机制 owl:equivalentClass 解决,同一术语在不同数据源表达不同概念,可以将数据源模式信息抽象为映射局部本体。

属性异构:包括不同数据源间的属性等价、继承关系,可以使用 OWL 的语义表达机制 owl:equivalentProperty 和 rdfs:subPropertyOf 解决。

类的外延异构:类的外延由属于该类的实例或个体集合定义,由于集合间的关系有相交、包含、等价和不相交关系,于是类之间也有相应的关系,类的包含、相交以及不相交分别用 OWL 的语义表达机制 rdfs:subClassOf、owl:intersectionOf、owl:disjointWith 解决。

(2) 数字化校园数据集成系统中使用本体的方法 在数字化校园建设过程中,大都采用自顶向下的数据集成方法,先构建整个元数据,然后再建立起元数据到各个信息源的映射,但这种方法难以解决集成中的语义冲突、等价、包含等问题,而本体是解决这些问题的最好方案。

① 在应用层中,首先通过构建数据集成领域的本体,清晰描述领域本体及本体之间的联系,本体结构体系反映数据集成的需求,采用本体词汇来表达用户数据集成的语义需求,假设用户表达数据集成需求的全局查询为 Q ,它是一个三元组 $\langle S, F, W \rangle$,全局本体为 OG , S, F, W 分别代表全局查询中的 Select, from 和 Where 子句中的查询参数集合,Select 语句中的查询参数集合为 $S = \{C_i, P_j \mid i \in (1, \dots, n); j \in (1, \dots, m)\}$ 其中 C_i 是来自全局本体 OG 中的概念, P_j 是概念 N_i 的属性, m, n 属于自然数。From 子句的查询参数集合 $F = \{C_i \mid i \in (1, \dots, n)\}$ 。Where 子句的表达式集合 W 分为两类, W_{con} 是形如 $C_i, P_j \Theta Constant$ 的表达式集合,其中 $\Theta \in \{<, >, =, \leq, \geq\}$, $Constant$ 是常量表达式, W_{var} 是形如 $C_i, P_j \Theta C_u, P_u$ 的表达式集合,表示全局本体概念的属性间的约束, $W = W_{con} \cup W_{var}$ 。

② 在中间层,建立起全局本体及其属性到局部数据源模式间语义映射关系,并将用户用本体词汇表达的数据集成需求转换为由局部数据源元数据表示的标准 SQL 语句。通过对基于本体的全局查询分解和重写,将全局查询分解为多个对应于数据源局部本体的子查询,结果是对全局查询中的本体概念转化为相应的局部本体的概念。这里利用全局本体

与局部本体的映射关系,利用推理机对全局本体进行推理,得到全局本体与局部本体的映射。具体过程是分别接受来自全局查询 $Q = \langle S, F, W \rangle$ 中全局本体的概念和概念属性为参数,例如对于全局概念,通过三元组推理得出主语是全局概念,谓词是 `URI#owl:equivalentClass`,宾语是局部概念的列表;同样可推出局部本体中的概念属性,最后分别将对 S, F, W 中全局词汇的推理得到的局部词汇组装成若干个子查询。

③底层局部数据源查询的实现。将利用局部本体词汇重写后的子查询,转化成目标数据源模式的词汇,通过访问局部本体查找局部本体词汇到数据源模式词汇的映射完成转换,最后执行目标数据源的查询语句,将查询结果转换成 XML 格式传回给中间层的结果集成模块。

3 实例分析

3.1 信息源及其关系说明

高校教务处关于学生基本信息描述的关系为 `Source1.Table1` (年级,学号,姓名,学院,专业),DBMS 为 MS SQL Server;财务处关于学生缴费信息描述的关系为 `Source2.Table2` (学号,姓名,学费,住宿费,学年),DBMS 为 MySQL。

3.2 学生缴费领域的本体模型构建及 OWL 表示

结合学校教务部门与财务部门数据管理与集成的需求,分别给出其领域本体及其 OWL 表示,图 1 是学生缴费领域的本体模型,图 2~图 5 是学生缴费领域若干相关本体词汇的 OWL 表示。

```
<owl:Class rdf:about="#DepositReceipt">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="#owl:Thing">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#preference"/>
            <owl:someValuesFrom rdf:resource="#StudentCharge"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#payer"/>
            <owl:someValuesFrom rdf:resource="#Student"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#happenYear"/>
            <owl:someValuesFrom rdf:resource="#xsd:string"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#oweMoney"/>
            <owl:someValuesFrom rdf:resource="#xsd:string"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#receiveMoney"/>
            <owl:someValuesFrom rdf:resource="#xsd:string"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
</owl:Class>
<owl:Class rdf:about="#GraduateStudent">
  <rdfa:subClassOf rdf:resource="#Student"/>
</owl:Class>
```

图2 缴费存单类的 OWL 表示

```
<owl:Class rdf:about="#StandardCharge">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#moneyAmount"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#type"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#push"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <rdf:Description rdf:about="#owl:Thing"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

图3 收费标准类的 OWL 表示

```
<owl:Class rdf:about="#Student">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#admissionYear"/>
          <owl:someValuesFrom rdf:resource="#xsd:gYear"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#majorIn"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasSno"/>
          <owl:someValuesFrom rdf:resource="#xsd:int"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasName"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#isGradeOf"/>
          <owl:someValuesFrom rdf:resource="#xsd:string"/>
        </owl:Restriction>
        <rdf:Description rdf:about="#owl:Thing"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

图4 学生类的 OWL 表示

```
owl:Class rdf:about="#StudentCharge">
  <owl:equivalentClass>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <rdf:Description rdf:about="#graduateTuition"/>
        <rdf:Description rdf:about="#graduateAccommodationFee"/>
        <rdf:Description rdf:about="#undergraduateTuition"/>
        <rdf:Description rdf:about="#undergraduateAccommodationFee"/>
      </owl:oneOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfa:subClassOf rdf:resource="#owl:Thing"/>
</owl:Class>
<owl:Class rdf:about="#UnderGraduate">
  <rdfa:subClassOf rdf:resource="#Student"/>
</owl:Class>
```

图5 学生收费标准类定义

3.3 领域本体到数据库模式映射

使用 OWL 的 owl:equivalentProperty 语法得到全局本体中声明本体属性到数据库字段列表的映射,本体系统采用 GLAV 的方式定义全局视图,将数据库模式合并成一个全局模式,消除冗余字段学号和姓名,并将住宿费和学费合并成学杂费字段,其余保留,得到集成需要的全局模式(学号,姓名,学院,专业,学杂费,学年),对于全局模式的每个项都对应本体类的属性,对于本体词汇类,通过 owl:equivalentClass 映射到数据库表,领域本体到数据库模式的映射如图 6~图 7 所示。

```

<owl:Class rdf:about="#Student">
  <owl:equivalentClass rdf:resource="Source1#Table1"/>
</owl:Class>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.学号"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#name" id="name">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.姓名"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.学号"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.学号"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.学号"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source1#Table1.学号"/>
</owl:DatatypeProperty>

```

图 6 学生缴费领域本体到数据库模式的映射关系 1 的 OWL 表示

```

<owl:Class rdf:about="#DepositReceipt">
  <owl:equivalentClass rdf:resource="Source2#Table2"/>
</owl:Class>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source2#Table2.学号"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#sno" id="sno">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#xsd:string"/>
  <owl:equivalentProperty rdf:resource="Source2#Table2.学号"/>
</owl:DatatypeProperty>

```

图 7 学生缴费领域本体到数据库模式的映射关系 2 的 OWL 表示

3.4 查询处理及其重写

在基于本体的数据集成方式中,通过本体词汇来表达用户的查询需求,如 select * from Student, DepositReceipt where Student. hasSno = '030414120'; 同 SQL 一样,这种语句也是 SFW 表达式,与 SQL 查询需求不同的是,选择、投影部分中的词汇都采用本

体词汇表示,中间件需要将这种语句转化为底层各个信息源直接联系的标准 SQL 表达式。由于系统具体应用需要两个异构数据库表的字段全部信息,所以本实例仅考虑选择条件及表部分,其实现如下:

(1) 利用 Jena 推理器的 RDFS 推理引擎对全局本体(global.owl)建立推理模型

```

String globalowl = "global.owl";
Model schema = ModelLoader.loadModel("global.owl");
Reasoner reasoner = ReasonerRegistry.getRDFSReasoner();
InfModel infmodel = ModelFactory.createInfModel(reasoner, schema);

```

(2) 针对选择 From 列表中出现的全局本体词汇,利用三元组推理方法,分别推理出与学生类,缴费用单类等价的基础数据表及其字段。

```

Property predicate = infmodel.getProperty("#equivalentClass");
Resource subject = infmodel.getResource("#Student");
StmtIterator i = infmodel.listStatement(subject, predicate, null);

```

经过推理得到学生类等价的数据库模式表名集合。同样,推出与缴款存单等价的局部数据库的表名列表;同时将学生类和缴款存单类中的属性推理得到等价的表字段列表,将类或属性作为键,与之推理得出的局部数据库表名列表或字段名列表作为值存储在映射类里,作为推理结果。

(3) 利用上述推理结果,重写底层数据源能直接执行的标准 SQL 语句。其主要实现方法包括 getLocalTable() 和 getLocalCondition()。

```

public void getLocalTable(List gFromList) {
  Iterator FromItr = gFromList.iterator();
  while(FromItr.hasNext()) {
    String param = (String) FromItr.next();
    // 查找映射
    List equalClassList = mappingFind.findEqualClass(param);
    Iterator equalItr = equalClassList.iterator();
    while(equalItr.hasNext()) {
      String schemaItem = (String) equalItr.next();
      // 判断模式项属于哪个数据源
      if(schemaItem.substring(0, schemaItem.indexOf("#")).equals(
        "source1")) {
        querylocal1_from.add(schemaItem + " ");
      } else querylocal2_from.add(schemaItem + " ");
    }
  }
}

public void getLocalCondition(List gwhereList) {
  Iterator whereItr = gwhereList.iterator();
  while(whereItr.hasNext()) {
    String param = (String) whereItr.next();
    String paramsub = param.substring(0, param.indexOf("=") - 1);
  }
}

```

```

String restexpr = param. substring ( indexOf ( " = " ), param.
length );
List equalClassList = mappingFind. findEqual Property ( param-
sub );
Iterator equalItr = equal PropertyList. iterator ();
while ( equalItr. hasNext () ) {
    String propertyItem = (String) equalItr. next ();
    if ( propertyItem. substring ( 0, schemaItem. indexOf ( " #" ) ). e-
quals ( " source1 " ) ) {
        querylocal1_from. add ( propertyItem + restexpr );
    }
    else querylocal2_from. add ( propertyItem + restexpr );
}
}
}

```

4 结论

本文引入本体技术解决数据化校园建设过程中数据集成的语义问题,通过构建教务领域的本体,通过建立领域本体和底层信息源关系模式之间的映射,一方面可以方便用户从语义角度表示数据集成的需求;另一方面可以有效的发现异构数据源环境中所涉及的概念间隐含关系,对于解决高校数据集成的语义冲突、等价及其包含等问题具有重要现实意义。

[参考文献]

- [1] CHRISTINE PARENT, STEFANO SPACCAPIETRA. Issues and approaches of database integration[J]. Communications of the ACM, 1998, 41(5): 166-178.
- [2] CONG Yu, LUCIAN Popa, Constraint-based XML query rewriting for data integration [C]. Proceedings of the 2004 ACM SIGMOD international conference on Management of data, France: Paris, 2004.
- [3] CHENG HIAN GOH. Representing and reasoning about semantic conflicts in heterogeneous information sources [D]. Phd, MIT, 1997.
- [4] DOMENICO BENEVENTANO, SONIA BERGAMASCHI, FRANCESCO GUERRA, et al. Synthesizing an Integrated Ontology[J], IEEE Internet Computing, 2003, 7(5): 42-51.
- [5] 鱼 滨, 郑娅峰. 基于本体的异构数据集成方法及其实现[J]. 计算机应用与软件, 2007, (9): 30-32.
- [6] CHAWATHE S, GARCIA-MOLINA H, HAMMER J, et al. Papakonstantinou, J Ullman, and J Widom. "The TSIMMIS Project: Integration of Heterogeneous Information Sources". In Proceedings of IPSJ Conference [C]. Japan: Tokyo, 1994: 7-18.
- [7] CALVANESE D, GIACOMO G D, LENZERINI M. Description Logics for Information Integration [M]. Computational Logic: Logic Programming and Beyond, 2002: 41-60.
- [8] WACHE H, VOGEL T, VISSER U. Ontology-based integration of information - a survey of existing approaches [C]. IJCAI, 2001: 108-117.
- [9] PETER HAASE, BORIS MOTIK. A mapping system for the integration of OWL-DL ontologies [C]. In In Proceedings of the ACM-Workshop: Interoperability of Heterogeneous Information Systems (IHIS05), 2005.
- [10] FARSHAD HAKIMPOUR, ANDREAS GEPPERT. Resolving semantic heterogeneity in schema integration [C]. Proceedings of the international conference on Formal Ontology in Information Systems, USA: Ogunquit, Maine, 2001: 297-308.

[责任编辑: 黄建军]