一种基于 ORM 的 OWL 本体与关系数据库映射模型①

羊海潮

(大理学院 数学与计算机学院, 大理 671003)

摘 要：如何使现有的关系数据库具有语义特征，如何对面向对象中的对象模型进行语义描述，是语义网应用过程中的两个重要问题。提出了一种 Ontology-Object-Relational 映射模型，提出了相关的映射规则及算法，基于映射元数据实现了本本、对象以及关系数据库间的映射与相互转换。实现了基于该模型的实验系统，并对关系数据库与本体实例间的相互转换进行了实验，实验结果表明本文所提出的映射模型、映射规则及算法是正确与可行的。

关键词：语义网；本体；OWL；面向对象；语义映射

A Model of Ontology Relational Mapping Based on ORM

YANG Hai-Chao

(College of Mathematics and Computer Science, Dali University, Dali 671003, China)

Abstract：How to make relational database has the semantic character, and how to describe the semantic of object, are important problem during the application of semantic web. In this paper, a model of Ontology-Object-Relational mapping rules and algorithms was also presented. Based on ORM, the mapping and transformation between ontology, object and relational database was realized. A system for experiments had been developed, and the experiment for the transformation between relational database and ontology instances was conduct, the result show that the mapping model, the mapping rules and the algorithms are right and feasible.

Key words：semantic web; ontology; OWL; object-oriented; semantic mapping

1 引言

本体(Ontology) 作为语义网的核心,从提出语义 Web 这一概念开始,它就受到了广泛关注[1]。Ontology 的目标是捕获相关的领域的知识，提供对该领域知识的共同理解，确定该领域内共同认可的词汇，并从不同层次的形式化模式上给出这些词汇（术语）和词汇之间相互关系的明确定义[2]，在信息系统中采用本体将带来许多好处：易表达性、可伸缩性、可共享性、可推理性。

面向对象是目前软件主流的软件开发技术，面向对象贯穿于从领域建模至系统实现的各个阶段，开发者们同样期望对象模型能够具有语义特征[3]，因此越来越多的软件开发者将目光转向语义网，采用本体驱动的软件开发模式，然而由于本体、对象、关系数据库间的差别性，直接在面向对象的软件开发过程中使用本体是十分困难的。同时现有信息系统都是以关系数据库的方式进行数据存储，如何将这此数据应用于语义网中也是一个重要的问题。

本文提出一种本体、对象及关系数据库间的映射模型，解决了本体与对象、关系数据库间的转换问题，为本体与面向对象及关系数据库间的语义映射提供了一条有效途径。

2 相关研究现状

OWL 是 W3C 推荐的 WEB 本体语言，可用于描述 WEB 本体及实例，主要包括类、属性、实例、数

① 收稿时间:2011-11-17;收到修改稿时间:2011-12-26

据类型等基本要素，通过类公理的定义类间的关系，通过属性公理定义属性关系[4]。OWL 由三个版本构成：OWL Full、OWL DL、OWL Liet，其中 OWL DL 支持那些需要最强表达能力的推理系统的用户，且这个推理系统能够保证计算的安全性和可判定性，它包括了 OWL 语言的所有成分，但有一定的限制，OWL DL 旨在支持已有的描述逻辑商业处理和具有良好计算性质的推理系统[5]，本文所提出的映射模型主要是以 OWL DL 为主。

国内外很多研究者对 OWL 本体与关系数据库间的映射作了研究，朱姬凤等提出一种基于转换规则的 OWL 本体到关系数据库模式的映射方法[1]，P。Bartalos 等提出了将本体实例映射到对象的方法[2]，A。Kalyanpur 等提出 OWL 本体映射为 JAVA 的方法[6]，这些方法对本体与数据库的映射或本体与对象的映射进行了研究，但是这些方法要么是本体与数据库的直接映射，要么是本体与对象直接映射，不能有效解决针对目前广泛存在的基于 ORM 的信息系统进行语义建模的问题。

3 Ontology-Object-Relational映射

3.1 映射模式

在充分研究面向对象与 OWL 本体的基础上，本文提出一种以 ORM(object relational mapping)为基础的本体与关系数据库映射模型，借鉴 ORM 模式中对象与关系数据库映射方式，通过对 Ontology 与 Object 间的映射描述建立 Ontology 与 Object 的映射关系，Object 再通过 ORM 方式与关系数据库映射，通过 Ontology-Object-Relational 的方式实现了 OWL 本体与关系数据库间的语义映射，此方式本文称之为 OORM，相关定义如下：

定义 1 本体 OT 是一个六元组，$OT=<C_t,R,A,H,I,X>$，其中 Ct 为概念集合，R 为概念间的关系集，A 为属性集，I 为实例集合，X 为公理集合。

定义 2 对象 O 是一个三元组，O=<Co,Ro>，其中 Co 为类的集合， Ro 为类间的关系集合。

定义 3 关系模式 R 是一个四元组，R=<T,CR,P,I>，其中：T 是数据库中的表名集合; CR 是 t 的列名集合，t∈T; P 是列 c 的类型集合，c∈CR; I 是完整性约束集合。

定义 4 Ontology-Object-Relational 映射 M 可描述为：

对于任意的 ot∈OT,o∈O,r∈R,M={ot→r|ot→o∧o→r}。

1.1 映射规则

在 Ontology-Object-Relational 映射模式中，对象到关系模式的映射可通过 Hibernate 等 ORM 框架快速实现[7]，本文对此不在进行描述，在此重点描述 OWL 本体到对象的映射规则。本文采用 POJO(Plain Old Java Object)作为对象模型，考虑到 Java、C#等常用面向对象语言均不支持多重继承，将 OWL 本体的类映射为面向对象模式下的一个接口[6]，同时由于 ORM 框架无法实现接口到关系模式的映射，因此通过实现接口的 POJO 再映射到关系模式；将 OWL 中的 DataTypeProperty 映射为 POJO 的属性（set/get 方法）；将 ObjectProperty 映射为 POJO 类的关联关系。由于篇幅所限，本文列出如下主要的映射规则。

为了方便描述映射规则，首先引入相关标识符与函数，如表 1 所示。

表 1 相关标识符

OWL 标识符	说明	OO 标识符	说明
OTC	本体类集合	*class*	类
OPC	对象属性集合	*Interface*	接口
DPC	数值属性集合	*extends*	继承
range	取得值域类	*onetoone*	一对一关系
domain	取得定义域类	*onetomany*	一对多关系
functional	函数关系	*manytomany*	多对多关系
inverseOf	互逆关系	*attribute*	类的属性
subClassOf	子类关系	*type*	数据类型

规则 1 任意一个 OWL 类 ot 映射为一个 POJO 类及一个接口，即

$$\forall ot \in OTC \rightarrow class(ot) \wedge Interface(ot)$$

规则 2 若 OWL 类 ot1 与 ot2 之间存在 subClassOf 关系则相应的接口映射成继承关系，即

$$(\forall ot1, ot2 \in OTC) \wedge subClassOf\ (ot1, ot2) \rightarrow i1 = Interface\ (ot1) \wedge i2 = Interface\ (ot2) \wedge extends\ (i1, i2))$$

规则 3 任意非互逆的函数型对象属性 op，将其 range 类与 domain 类映射成一对一关系，即

$\forall op \in OPC \wedge functional(op) \wedge \neg inversof(op) \rightarrow onetoone(class(domain(op)), class(range(op)))$

规则 4 任意互逆的函数型对象属性 op1,op2，将 op1 的 range 类与 domain 类映射成一对一关系，即

$(\forall op1 \in OPC \wedge functional(op1) \wedge inversof(op2)) \wedge (\forall op2 \in OPC \wedge functional(op2) \wedge inversof(op1)) \rightarrow onetoone(class(domain(op1)), class(range(op1)))$

规则 5 op1 为对象属性，op2 为函数型对象属性，且 op1 与 op2 为互逆关系，将 op1 的 range 类与 domain 类映射成一对多关系，即

$(\forall op1 \in OPC \wedge inversof(op2)) \wedge (\forall op2 \in OPC \wedge functional(op2) \wedge inversof(op1)) \rightarrow onetomany(class(domain(op1)), class(range(op1)))$

规则 6 任意非互逆的对象属性 op，将 op 的 range 类与 domain 类映射成一对多关系，即

$\forall op \in OPC \wedge \neg inversof(op) \rightarrow manytoone(class(domain(op)), class(range(op)))$

规则 7 op1 与 op2 均为对象属性，且 op1 与 op2 为互逆关系，则将 op1 的 range 类与 domain 类映射成多对多关系，即

$(\forall op1 \in OPC \wedge inversof(op2)) \wedge (\forall op2 \in OPC \wedge inversof(op1)) \rightarrow manytomany(class(domain(op1)), class(range(op1)))$

规则 8 任意函数型数值属性 dp，将 dp 映射为 domain 指定的类属性，该属性类型为 range 指定类型，即

$\forall dp \in DPC \wedge functional(dp) \rightarrow attribute(class(domain(dp)), type(range(dp)))$

规则 9 任意数值属性 dp，将 dp 映射为类 c1，domain 指定的类与其为一对多关系，该属性类型为 range 指定类型，即

$\forall dp \in DPC \rightarrow c1 = class(dp) \wedge attribute(c1, type(range(dp))) \wedge onetomany(c1, class(domain(dp)))$

3.2 相关算法

采由 3.2 所述映射规则建立本体与对象间的映射关系后，就可实现本体实例与关系模式元组间的转换，下面给出两个主要的算法，算法 1 为本体实例向关系模式元组的转换，算法 2 为关系模式元组向本体实例的转换。

算法 1 将本体实例保存为关系模式中的元组(insert)

输入：本体 OT，OT 与对象 CO 的映射关系 map_i，本体实例 ot。

输出：元组 r，$r \subset R$，R 为关系模式

步骤：

1.找出与 OT 对应的 POJO 类 CO，并创建 CO 的对象 c1。

2.取得所有数值属性映射集合 $md \subset map_i$。

3.遍历 md 中的每个映射关系，对每个 $m_{dp} \in md$ ：

3.1 若 m_{dp} 为函数型数值属性，则将 ot 中对应属性 dp 的值附给 c1 的相应属性，即 c1。attribute(dp)= ot(dp).

3.2 若 m_{dp} 为非函数型数值属性，则创建 class(dp) 的实例 cdp，将 dp 属性的值附给 cdp，即 cdp.attribute(dp)=ot(dp),将 cdp 添加到 c1 的对应集合型属性，即

c1.attribute(range(dp)).add(cd p)。

4. 取得所有的对象属性集合 $mo \subset map_i$。

5. 遍历 mo 中的每个映射关系，对每个 $m_{op} \in mo$ ：

5.1 若 m_{op} 满足规则 3 或规则 4（即一对一关系），则将属性 op 对应的本体实例 ot_op 转为对象 c_op，将 c_op 附给 c1 的对应属性，即

c1.attribute(range(op))=c_op.

5.2 若 m(op) 满足规则 5、规则 6、规则 7 中的任意一条（即一对多），则将 op 对应的本体实例 ot_op 转为对象 c_op，将 c_op 添加给 c1 的对应集合型属性，即

c1.attribute(range(op))。add(c_op)。

6.通过 O/R Mapping 将 c1 转化为关系模式中的元组 r.

算法 2 关系模式中的元组转化为本体实例

输入：关系模式中的元组 r，本体 OT 与对象 CO 的映射关系 map_i。

输出：本体实例 ot

步骤：

1.通过 O/R Mapping 将元组 r 转换为对象 c1。

2.创建本体 OT 的实例 ot。

3.取得所有数值属性映射集合 $md \subset map_i$。

4. 遍历 md 中的每个映射关系，对每个

$m_{dp} \in md$ ：

4.1 若 m_{dp} 为函数型数值属性，则创建 ot 的函数型数值属性 dp，将 cl 中对应属性 dp 的值附给 ot 的相应属性，即 ot(dp)= c1。attribute(dp)。

4.2 若 m_{dp} 为非函数型数值属性，取得 c1 对应的集合类型属性 dpList，遍历 dpList，对每个 $dpList[i] \in dpList$ ：

4.2.1 创建 ot 的非函数型数值属性 dp。

4.2.2 将 dpList[i]的值附给 dp。

5.取得所有的对象属性集合 $mo \subset map_i$ 。

6.遍历 mo 中的每个映射关系，对每个 $m_{op} \in mo$ ：

6.1 取得 c1 对应于属性 op 的所有对象集合 opList。

6.2 遍历 opList，对每个 $opList[i] \in opList$ ：

6.2.1 若 ot 中不存在对应 opList[i]的实例，则创建对应 opList[i]的实例 ot_cp。

6.2.2 创建 ot 的对应属性 ot(op)，将 ot_cp 的 id 附给 ot(op)。

4 系统实现

本文根据上述规则及算法，基于 J2EE 多层体系架构实现了实验系统，图 1 为实验系统的系统框架，采用 POJO 作为对象模型，Hibernate 为 ORM 框架，采用 Jena 操纵 OWL 本体，Ontoloty-Object Adapter 为 OWL 本体与 POJO 类之间的映射适配器，能根据预先定义好的 Ontology-Object 映射元数据完成本体与 POJO 间的映射转换。

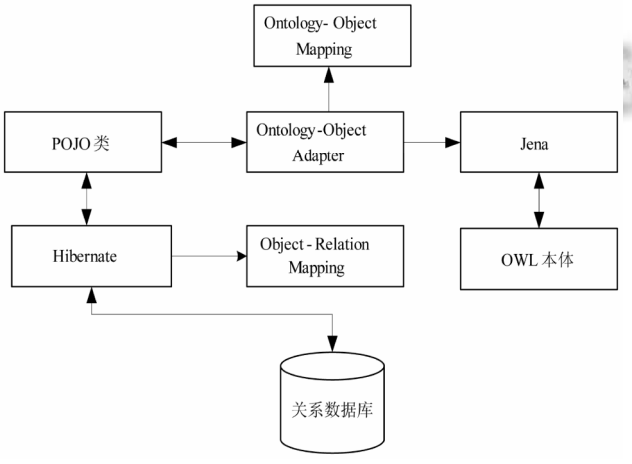

图 1　系统框架

Ontology-Object Adapter 包含如下主要的类：

① MappingManager：映射管理器，负责对本体与对象间的映射元数据进行管理，包括映射文件解析、保存、修改等方法。

② ObjectOntoConverter:对象与本体转换类，负责根据映射元数据文件完成对象与本体间的相互转换。

5 实验

本文使用已经实现的实验系统对一个面向对象的电子商务系统建立本体、对象、关系数据库间的映射与转换，实验步骤如下：

①根据关系数据库建立对象模型及基于 Hibernate 的 O/R Mapping; ②采用 protégé 建立本体概念模型; ③建立本体与对象间的映射，存储为 Customer。Onto。XML,该文件的片断如下：

```
<onto-mapping>
  <ontology ID=" Customer " class="Customer">
    <DatatypeProperty name="hasNo"
    type="FunctionalProperty">
      <field name=" custNO "/>
    </ DatatypeProperty >
    <DatatypeProperty name="hasName"
type="FunctionalProperty">
      <field name=" custName "/>
    </ DatatypeProperty >
    <ObjectProperty
name="workFor"type="FunctionalProperty">
      <field name="company"/>
    </ObjectProperty>
  </ ontology >
</onto-mapping>
```

④ 输入 HQL 语句通过 Hibernate 以 O/R Mapping 方式查询数据库，并将查询结果转换为本体实例；⑤新建一组本体实例并保存为 RDF 文件，将本体实例存储进数据库中。

实验正确的完成了本体与关系数据库间的映射与转换，系统界面如图 3 所示。

为了验证本文所提出的基于 ORM 的本体与关系数据库映射模型优越性，采用本体与关系数据库直接映射算法对同一个电子商务系统进行了进行映射转换实验，以转换正确性作为指标与本文所提方法进行对比分析，转换结果如表 2 所示。

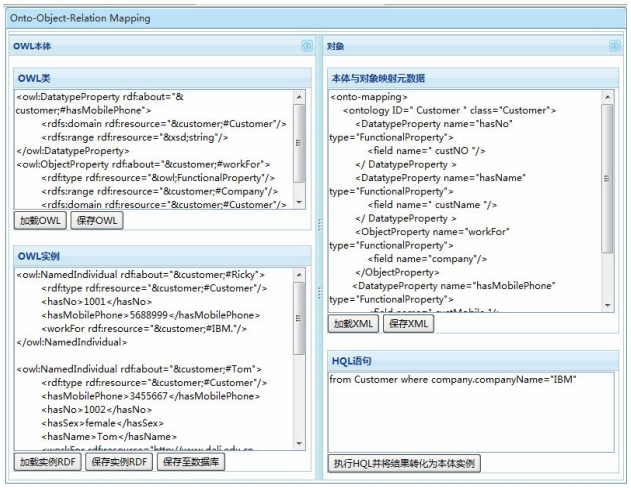

图 3 系统界面

表 2 与直接映射方法映射转换正确性对比

对象名	属性数	基于 ORM 方法正确数	直接映射方法正确数
customer	16	16	15
product	22	22	20
order	14	14	13
orderDetail	15	14	11
category	13	13	13
supplier	8	8	8

从表 2 数据可知，对于该系统的 78 个属性，基于 ORM 方法正确率为 98.7%，直接映射方法为 91%，对于面向对象的系统，本文提出的基于 ORM 的方法映射转换更加准确。

6 结语

如何使目前大量已经存在的关系数据库具有语义特征，如何对传统的面向对象中的对象模型进行语义描述，是语义网应用过程中的两个重要问题。本文的贡献在于提出了一种 Ontology-Object-Relational 映射模型，提出了相应的映射规则及算法，基于 ORM，通过建立本体与对象间的映射关系，实现了本体、对象、关系数据库的相互转换。实验结果证明本文所提出的映射模型、映射规则及其算法是正确与可行的。该模型可以在不改变已有关系数据库的表结构前提下为关系数据库增加语义特征，为已有关系数据库在语义网中的应用提供了一种有效的方式。今后还需加强映射规则的完备性，并对该模型在应用于基于语义的查询扩展进行进一步研究。

参考文献

1 朱姬凤,马宗民,吕艳辉.OWL 本体到关系数据库模式的映射.计算机科学,2008,35(8):165–169.

2 Antoniou G, van Harmelen F. A Semantic Web Primer. London:The MIT Press,2003.

3 Bartalos P, Bielikov'a M. An approach to object ontology mapping. In 2nd IFIP Central and East European Conf. on Software Engineering Techniques CEE-SET 2007,2007.

4 Athanasiadis IN. Ontologies, JavaBeans and Relational Databases for enabling semantic programming.In Proc. of the 31th IEEE Annual International Computer Software and Applications Conference(COMPSAC), Beijing, China, IEEE, July 2007.341–346.

5 Smith MK, Welty C. OWL Web Ontology Language Guide. http://www.w3.org/TR/2004/REC-owl-guide-20040210/,2004.

6 Kalyanpur A, Pastor DJ, Battle S, Padget J. Automatic mapping of owl ontologies into java.In 16th Int'l Conference on Software Engineering and Knowledge Engineering, Banff, Canada, June 2004.

7 King G, Bauer C. Hibernate-Relational Persistence for Idiomatic Java. 2010.http://docs.jboss.org/hibe-rnate/stable/core/referenc/en/html