Research on the Rules of Mapping from Relational Model to OWL

Guohua Shen, Zhiqiu Huang, Xiaodong Zhu, Xiaofei Zhao

College of Information Science and Technology
Nanjing University of Aeronautics and Astronautics, 210016, Nanjing, China
{ghshen, zqhuang, zhuxd}@nuaa.edu.cn, zxf-first@163.net

Abstract. Data integration provides the user with a unified view of legacy data, and the semantic mapping from relational database (local sources) to global ontologies is one of key aspects for building data integration system. The manual mapping is time-consuming and error-prone. In this paper, groups of semantic mapping rules from relational database to global OWL ontologies are proposed in detail, including rules for the classes, properties, restriction and instances, which avoid migrating large amounts of data. The rules are demonstrated with examples. They are practical for semantic mapping or ontologies learning (semi-)automatically.

Keywords: Semantic mapping; OWL; Relational model; Ontology; Data integration

1 Introduction

Data integration is the problem of combining legacy data, and providing the user with a unified view of these data. It is one of hotspots for data management systems in the distributed computing environment. The mediator is a main approach to data integration, and the mapping between global schemas and local schemas is one of key issues.

Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related formally. It provides a way to make the data readable and understandable by the machines, whereby it improves system interoperation and knowledge share. The Web Ontology Language (OWL) is a language for defining and instantiating Web ontologies.

Large amounts of data in legacy systems are stored in the relational database, because RDBMS technology is mature and RDBMS is efficient in storing and querying data. The global ontologies is essential for semantic data integration. So how to transfer the local data to global ontologies is the key aspect, on which is focused in this paper.

Bibliographies [3] and [4] proposed methods to migrate relational data into ontologies, including ontologies definition and instances. And some rules of learning ontologies from relational database were represented [5]. The above methods are used for acquiring ontologies instances from relational data sources. That is, They are essentially data acquisition and semantic annotation. When the large amount of instance data are stored in OWL files, it is not efficient to maintain and query data. The normative OWL exchange syntax is XML, which is a meta-markup language and is fit for data description and data exchange instead of data storage. So migrating large amount of relational data into ontologies file means lots of transferring effort. The ontologies migrated from various sources are not consistent, they should be fused.

Bibliography [6] proposed a solution to extracting data from the OWL document, and then stored data in relational database. It enables users to reference ontology data directly from SQL using the semantic match operators. The relational model is good at data management, but it restricts the semantic expression.

So, only meta-data mapping is established in our solution: that is, constructing correspondence between ontologies definition and data schema, and the instance data still reside in database. It is flexible and easy to extend new data source, and avoid migrating large amount of data.

The mapping rules from relational database to OWL are focused in this paper. The rest of the paper is organized as follows: Section 2 describes the relational model, ontologies and semantic mapping relationships between them. Section 3 presents the mapping rules from relational database to OWL ontologies in detail, which are demonstrated with examples, and the rules are classified as four groups: concepts, properties, restrictions and instances. Finally, Section 5 concludes with a summary.

2 Semantic Mapping from Relational Model to OWL

2.1 Relational Model

Relational database is essentially based on relational model, which is a tuple:

RM = (NM, rel, subof), where

- NM is a set of tables, $NM = ET \cup RT \cup DT$, and $ET \cap RT = \phi$, $ET \cap DT = \phi$, $RT \cap DT = \phi$. ET is a set of entity tables, RT is a set of relationship tables, DT is a set of data types (data types are listed in Table 1 in detail).
- rel is a triple relation, $rel \subseteq ET \times RT \times ET$, indicates that one relationship table relates to two entity tables.
- subof is a binary relation, $subof \subseteq ET \times ET$, indicates sub- or super- relation between two entity tables.

Each relation R refers to a table, and each column of a table is called an attribute, denoted as A_i. The relation is denoted as $R(A_1, A_2, \ldots, A_n)$. $R.t \in R$ means that t is a tuple of R. And $t[A_i]$ is the corresponding component A_i in tuple t.

Several functions are defined as following:

- attr(R), the function gets all the attributes from relation R, obviously $A_i \in attr(R)$;
- pkey(R), the function gets the primary keys of relation R, obviously $pkey(R) \subseteq attr(R)$;

– fkey(R), the function gets the foreign keys of relation R, obviously fkey(R) $\subseteq$ attr(R) ;
– attrName(A_i), where attrName=[nam| dom | dataType], the functions get some aspects of attribute A_i. E.g. nam(A_i) gets the name of attribute A_i; dom(A_i) gets the domain of attribute A_i; dataType(A_i) gets the data type of attribute A_i, obviously dataType(A_i) $\in$ DT;

I is a set of inclusion dependencies, where each element has the form like ((R_i, A_i) (R_j, A_j)), A_i ={ A_{i1} , A_{i2} , A_{i3}…}, $A_i \subseteq$ attr (R_i) , A_j ={ A_{j1} , A_{j2} , A_{j3}…}, $A_j \subseteq$ attr(R_j). For each t[A_{ix}] in relation R_i, if there exists R_i.t[A_{ix}]= R_j.t[A_{jx}] in relation R_j, where x=1,2,3…, A_i and A_j are called inclusion dependency, denoted as $R_i(A_i) \subseteq R_j(A_j)$. I_c denotes the transitive closure of I.

Table 1. Corresponding data type between RDB and OWL

Type	RDB	OWL/XML[7,8]
Numerical	smallint	xsd:decimal
	integer/int	xsd:float
	decimal	xsd:decimal
	numeric	xsd:decimal
	float	xsd:float
Char	char	xsd:string
	varchar/vchar	xsd:string
Time	time	xsd:time
	date	xsd:date
	datetime	xsd:datetime
Bool	boolean	xsd:boolean
Byte	blob	-
	bytes	-

2.2 Ontology and OWL

An ontology is an explicit specification of a conceptualization for the purpose of enabling knowledge sharing and reuse. An ontology is a description of the concepts and relationships that can exist for an agent or a community of agents.

Ontology is a tuple, O = (C, R, func, A, I), where

– C is a finite set of concepts, e.g. student is a concept;
– R is the set of the relations between concepts;
– func is the function, a kind of special relation, e.g. motherOf(x, y) denotes x is mother of y.
– A is axioms, means tautologies.
– Instance I is an individual of the concept, e.g. Ben is a student.

The Web Ontology Language (OWL) is a language for defining and instantiating ontologies, and it can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms.. Web Ontology Language (OWL) was adopted as the recommendation by W3C in 2004. The main OWL language constructs are shown in Table 2.

Table 2. List of the main OWL language constructs

Type	Name
RDFS Features	Class
	rdfs:subClassOf
	rdf:Property
	rdfs:subPropertyOf
	rdfs:domain
	rdfs:range
	Individual
(In)Equality	equivalentClass
	equivalentProperty
	sameAs
	differentFrom
	AllDifferent
	distinctMembers
Property Characteristics	ObjectProperty
	DatatypeProperty
	TransitiveProperty
	SymmetricProperty
	FunctionalProperty
	InverseFunctionalProperty
Restricted Cardinality	minCardinality
	maxCardinality
	cardinality

2.3 Semantic Relationships between Relational Model and Ontology

We take an educational administration system as an example. Its main schema definitions are shown in Table 3.

Table 3. The example of relational schemas definition in RDB

No	Relational schemas
1	Department (deptId int, deptName vchar, deptAddr vchar)
2	Student (stuId int, stuName vchar, deptId int, sex vchar)
3	GraduateStudent (stuId int, staffId int, researchArea vchar)
4	Course (courseId int, courseName vchar, staffId int, deptId int)
5	ChooseCourse (stuId int, courseId int)
6	Staff (staffId int, staffName vchar, email vchar, memo vchar)
7	StaffEx (staffId int, MSN vchar, homePage vchar)
8	AcademicStaff (staffId int, researchArea vchar)
9	AdminStaff (staffId int, duty vchar)
10	Sex(sex vchar)

Note: the real underlines indicate the primary key, and the dashed underlines indicate the foreign key.

Suppose that the elementary global ontologies have been constructed by the domain experts. And how to map from relational database to OWL is the key issue in semantic data integration. The semantic mapping is time-consuming and error-prone, so we present dozens of mapping rules, which can be used to perform mapping semi-automatically or even automatically, avoid manual work repeatedly. The basic mapping principles are given below:

1. One relation R_i is mapped to one concept C_i ;
2. Inclusion dependency of each foreign key (in one relation R_i)on the primary key (in another relation R_j)is mapped to an ObjectProperty OP_i ;
3. Each property (exclude foreign key) of a relation R_i is mapped to a DatatypeProperty DP_i ;
4. Each tuple of a relation R_i is mapped to an individual I_i.
5. The data type corresponding relationships between relational model and OWL is given in Table 1.

All the rules are classified as four groups: concepts, properties, restrictions and instances.

3 Mapping Rules

3.1 Mapping Rules for Concepts

Rule C-1 For Relation R_i, if $|pkey(R_i)|=1$, that is, if R_i has the only primary key (R_i is an entity table), R_i can be mapped to one concept C_i.

According to the Rule C-1, The relations Department, Student, GraduateStudent can be mapped to concepts respectively.

```
<owl:Class rdf:ID="Department"/>
<owl:Class rdf:ID="Student "/>
<owl:Class rdf:ID="GraduateStudent"/>
```

Rule C-2 If $pkey(R_i)= pkey(R_j)$, and $((R_i, pkey(R_i)), (R_j, pkey(R_j))) \in I_c$, that is, if R_i and R_j have the same primary key, R_i and R_j can be mapped to the same concept Ci.

According to the Rule C-2, The relations Staff and StaffEx can be mapped to the same ontology Staff, because information is distributed in two relations.

```
<owl:Class rdf:ID="Staff"/>
```

Rule C-3 If precondition of Rule C-2 is satisfied, and concepts for both relations exist, R_i and R_j can be mapped to the concept C_i and C_j respectively, and C_i is sub concept of C_j.

According to the Rule C-3, ((GraduateStudent, {stuId}), (Student, {stuId})) $\in I_c$, so GraduateStudent is sub concept of Student.

```
<owl:Class rdf:ID="GraduateStudent">
  <rdfs:subClassOf rdf:resource="#Student" />
</owl:Class>
```

3.2 Mapping Rules for Properties

Rule P-1 For Relation R_i, if $|pkey(R_i)| \geq 1$, that is, if R_i has the primary key (R_i is an entity table), A_i ($A_i \in attr(R_i)$) is mapped to the property of concept C_i (C_i is the corresponding concept of R_i).

AppendixRule P-1.1 If Rule P-1 is satisfied, and the following conditions are satisfied:

1. $|fkey(R_i)| \geq 1$;
2. $R_i(A_i) \subseteq R_j(A_j)$, ($A_i \in fkey(R_i)$),

the foreign key (s) can be mapped to the ObjectProperty OP_i of concept C_i. The domain of OP_i is C_i, and the range of OP_i is C_j (C_i and C_j are the corresponding concepts of R_i and R_j respectively).

AppendixRule P-1.2 If Rule P-1 is satisfied, and $A = attr(R_i)-pkey(R_i)-fkey(R_i)$, if $| A | \geq 1$, each attribute in A can be mapped to the DatatypeProperty DP_i of concept C_i.

According to AppendixRule P-1.1 and AppendixRule P-1.2, we can get the ObjectProperty “deptId” of concept “Student”, and DatatypeProperties, e.g. “stuName”.

```
<owl:DatatypeProperty rdf:ID="stuId">
  <rdfs:domain rdf:resource="#Student" />
  <rdfs:range  rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty  rdf:ID="stuName">
  <rdfs:domain  rdf:resource="#Student"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty  rdf:ID=" deptId ">
  <rdfs:domain  rdf:resource="# Student"/>
  <rdfs:range rdf:resource="# Department"/>
</owl:ObjectProperty>
```

Rule P-2 For Relation R_i, R_j and R_k, if $pkey(R_i) \cup pkey(R_j) = fkey(R_k)$ and $pkey(R_i) \cap pkey(R_j) = \phi$, $|pkey(R_i)|=|pkey(R_j)|=1$, that is, if R_k is related with R_i and R_j , $pkey(R_i)$ and $pkey(R_j)$ can be mapped to the ObjectProperty OP_i and OP_j respectively. The domain of OP_i is C_i, and the range of OP_i is C_j . The domain of OP_j is C_j, and the range of OP_j is C_i . OP_i and OP_j are inverseOf each other. (C_i and C_j are the corresponding concepts of R_i and R_j respectively).

Rule P-2 is applied to the relationship table, e.g. for relation “ChooseCourse”, ObjectProperties ”chooseCourse” and ”beChosedBy” can be mapped to as below.

```
<owl:ObjectProperty rdf:ID=" chooseCourse">
  <rdfs:domain  rdf:resource="# Student"/>
  <rdfs:range  rdf:resource="# Course "/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=" beChosedBy">
  <rdfs:domain rdf:resource="# Course"/>
  <rdfs:range rdf:resource="# Student"/>
  <owl:inverseOf rdf:resource="# chooseCourse"/>
</owl:ObjectProperty>
```

Some other properties, e.g. TransitiveProperty, SymmetricProperty, FunctionalProperty may be annotated manually according to the business logic.

3.3 Mapping Rules for Restrictions

Rule R-1 If AppendixRule P-1.1 is satisfied, that is, there is foreign key in the entity table, the ObjectProperty OP_i has a restriction allValuesFrom, which refers to the corresponding inclusion dependency concept.

For example, relation “Student” has a foreign key “deptId”, then we can get the restriction “allValuesFrom” of the ObjectProperty “deptId”.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#deptId"/>
  <owl: allValuesFrom rdf:resource="# Department">
</owl:Restriction>
```

The following rules (Rule R-2, Rule R-3 and Rule R-4) are applied to cardinality constraints.

Rule R-2 For Relation R_i, if $A=pkey(R_i)\cup fkey(R_i)$ and $A\neq\phi$, the restrictions minCardinality and maxCardinality of each property P_i (P_i is the corresponding property of the attribute in A) are both assigned to 1, or restrictions cardinality of each P_i is assigned to 1.

Rule R-3 For Relation R_i, if $A_i \in attr(R_i)$ and A_i is declared UNIQUE, the restrictions maxCardinality of property P_i (P_i is the corresponding property of A_i) is assigned to 1.

Rule R-4 For Relation R_i, if $A_i \in attr(R_i)$ and A_i is declared UNLL, the restrictions minCardinality of property P_i is assigned to 0; if A_i is declared NOT UNLL, the restrictions minCardinality of property P_i is assigned to 1.

For example, According Rule R-2, we can get the cardinality constraints as below:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#stuId"/>
  <owl:cardinality rdf:datatype =
"&xsd;nonNegativeInteger"> 1 </owl:cardinality>
</owl:Restriction>
```

The attribute “memo” of the Relation “Staff” is declared NULL, so we can get restriction as below:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#memo"/>
  <owl:mincardinality rdf:datatype =
"&xsd;nonNegativeInteger">0</owl:cardinality>
</owl:Restriction>
```

Besides the above situation, many cardinality constraints are implicit, or even lacking, so they should be annotated manually according to the business logic.

3.4 Mapping Rules for Instances

According to the metadata-level mapping rules (e.g. concepts, properties, restrictions), the tuples of the relation can be transferred to the instances for data exchanging. The rules are as following:

Rule I-1 If relation R_i is mapped to the concept C_i, one tuple R_i.t can be transferred to the instance of C_i, each t[A_i] ($A_i \in$ attr(R_i)) can be transferred to the properties of the instance.

For example, the tuples of relation “Department” are (3, ”DeptOfAutomatics”, ”B11 Street.xx”), (4, ”DeptOfComputerScience”, ” B12 Street.xx”); the tuples of relation “Student” are (1, ”San Zhang”, 4), (2, ”Si Lee”, 4), and they can be transferred to the following instances:

```
<Department rdf:ID="DeptOfComputerScience">
 <deptId rdf:datatype="&xsd;integer" owl:cardinality=1>
4 </deptId>
 <deptName rdf:datatype="&xsd;string"> Dept. of
Computer Science </deptName>
 <deptAddr rdf:datatype="&xsd;string"> B12 Street.xx </
deptAddr>
</Department>
…
<Student rdf:ID="SanZhang0801" >
 <stuId rdf:datatype="&xsd;integer" owl:cardinality=1>
1 </stuId>
 <stuName rdf:datatype="&xsd;string"> San Zhang
</stuName>
 <deptId rdf:resource="# DeptOfComputerScience"/>
</Student >
…
```

Rule I-2 If all the tuples of relation R_i are mutually distinct, the instances can be asserted to be “AllDifferent”.

For example, the relation “Sex” has only two tuples: (”male”), (”female”), they can be transferred to a collection as following:

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
   <Sex rdf:about="#Male"/>
   <Sex rdf:about="#Female"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

Rule I-2 is applied to the basic table especially, e.g. a “product category” table.

4 Conclusions

In this paper, the rules of mapping relational model to OWL are proposed for the data integration, and they are classified as concepts, properties, restrictions and instances. These rules can be applied to mapping relational database to ontologies in OWL, whereby the mapping and transferring can be performed (semi-)automatically. The rules for concepts, properties and restrictions depict the correspondence at metadata level, which avoid migrating the large amount of data. The rules for instances are applied to create data for exchanging at running time. All the rules can also be applied to learning ontologies from relational database.

Because many constraints, relationships and other semantics in relational database are implicit, or even lacking, the ontologies mapped from relational model are not complete in semantics maybe. It could be annotated by experts, which depends on the domain knowledge and experiences. At the same time, some dynamical aspects in relational model, such as triggers, storage procedure cannot be mapped.

References

1. Lenzerini, M.: Data Integration: A Theoretical Perspective. ACM PODS， Madison， Wisconsin, USA, (2002)233-246
2. Mattos, N.M.: Information integration for on demand computing. In: Proc. of the 29th VLDB Conference, Berlin, Germany(2003)8-14
3. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web Sites into the Semantic Web. In: Proc. of the 17th ACM symposium on applied computing. ACM press, (2002)1100-1107
4. Rubin, D.L., Hewett, M., Oliver, D.E., et al.: Automatic data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and XML. In: Proc. of the Pacific Symposium on Biology(2002)
5. Li, M., Du, X.Y., Wang, S.: Learning ontology from relational database. In: Proc. of the 4th International Conference on Machine Learning and Cybernetics, Guangzhou, China(2005) 3410-3415
6. Das, S., Chong, E.I., Eadon, G., et al.: Supporting Ontology-based Semantic Matching in RDBMS. In: Proc. of the 30th VLDB Conference, Toronto, Canada(2004)1054-1065
7. Biron, P.V.: XML Schema Part 2: Datatypes 2nd Edition. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, (2004)
8. Smith, M.K.: OWL Web Ontology Language Guide. http://www.w3.org/ TR/2004/REC-owl-guide-20040210/, (2004)
9. McGuinness, D.L.: OWL Web Ontology Language Overview. http://www.w3.org /TR/2004/REC-owl-features-20040210/, (2004)
10. Bechhofer, S.: OWL Web Ontology Language Reference. http://www.w3.org /TR/owl-ref/, (2004)