Proceedings of the International Multiconference on Computer Science and Information Technology pp. 135–139
ISBN 978-83-60810-27-9
ISSN 1896-7094

Relational database as a source of ontology creation

Zdenka Telnarova
University of Ostrava 30. dubna
22, 701 03 Ostrava, Czech
Republic
Email:zdenka.telnarova@osu.cz

***Abstract*—The article headed "Relational database as a source of an ontology creation" deals with mapping relational data into ontology, or filling ontology with data from relational databases. It describes the issue of mapping database schemas (particularly relational models) for common data models expressed in the form of ontology. Generous room is given to methods of acquiring ontology from relational databases, where rules are specified and simple example are used to demonstrate their use, mapping of individual concepts of a relational data model into ontology concepts.**

I. Methods of Creating Ontology from Database

AN ontology provides shared and repeatedly usable knowledge about a specific domain. Existing relational databases, which contain enormous amounts of data, can be used to fill ontology with knowledge. There are numerous methods how to use existing databases to fill ontology using rules or using the so-called middle model. Many works deal with the issue of adding semantic to database schemas, e.g. [2]. Also, object-oriented data models are closely linked to ontological theory, nevertheless, even these do not offer sufficient semantic properties, such as hierarchies, for example. Methods of using relational databases for acquiring ontology can be divided minimally into the following four areas.

Area 1: Creation of a new database schema based on an analysis of the current relational schemas and mapping of this schema onto an ontology using reverse engineering. The new data model is designated as a so-called middle model, which is a common model above partial relational models [3].

Area 2: A method based on filling ontology instances using external relational sources. It uses a declarative interface between the ontology and data sources in XML.

Area 3: A method which creates ontology from a conceptual database schema, in the process the aim of this method is to create a RDF(S) ontology. However, this method has an unclear semantic and does not offer an inference model which is necessary for automatic derivation.

Area 4: A method which creates ontology from a conceptual database schema, whereas the aim of this method is to create an OWL ontology.

Further, we shall deal with only the last of the above said methods.

II. Ontology over the Relational Schema

The following definitions are necessary for describing the method of mapping relational data to an OWL ontology [9].

Definition I. (ontology with relation)

The ontological structure consists of five concepts
$O = \{C, R, H^C, rel, A^0\}$, where
C is a finite set of concepts
R is a finite set of relations
H^C is a concept hierarchy or taxonomy, $H^C \subseteq C \times C$, e.g. $H^C (C_1, C_2)$ expresses that C_1 is a sub-concept of C_2
rel is a set of non-taxonomic relations, e.g. $rel(R)=(C_1, C_2)$ expresses that concepts C_1, C_2 enter into relation R.
A^0 is a set of axioms expressed in the respective logical language, e.g. first-order logic.

The ontological structure includes as its extension a set of instances which suit this structure. Most OWL ontology elements involve classes, properties and instances. Most concepts of a specified domain correspond with classes which are the roots of various taxonomic trees. Properties are used to express general facts about class members and specific facts about instances. A property is a binary relation with a specified domain and scope. OWL distinguishes two types of properties. There are property between an instance and data type and property between instances of two classes. Both properties and classes may be classified in a hierarchy.

Definition II. (relational model)

In a relational model the R_i relation corresponds to a table where columns are called attributes and are designated A. The $attr(R_i)$ function returns attributes connected with a specific R_i relation. The dom (A_i) function determines the value range of attribute A_i, $A_i \in A$. The $pkey(R_i)$ function returns the primary key of the given R_i relation, whereas it must apply that $pkey(R_i) \subseteq attr(R_i)$. The $fkey(R_i)$ function returns a foreign key of the given relation R_i, again, it must apply that $fkey(R_i) \subseteq attr(R_i)$. Relations in a relational schema may be interdependent inclusively or equivalently.

Definition III. (inclusive dependence of relations)

Let R_i and R_j be relations and it is presumed that $A_i \subseteq attr(R_i)$ and $A_j \subseteq attr(R_j)$. Let $t_i(A_i)$ be value of t_i with attributes A_i and $t_j(A_j)$ value of t_j with attributes A_j. If for each $t_i(A_i)$ in R_i exists a $t_j(A_j)$ in R_j, for which it applies that $t_i(A_i) = t_j(A_j)$, then A_i and A_j are inclusively dependent, $R_i(A_i) \subseteq R_j(A_j)$ is recorded.

Definition IV. (equivalent dependence of relations)

Let R_i and R_j be relations and it is presumed that $A_i \subseteq attr(R_i)$ and $A_j \subseteq attr(R_j)$. If exists $R_i(A_i) \subseteq R_j(A_j)$ and $R_j(A_j)$

$\subseteq R_i(A_i)$, then A_i and A_j are equivalently dependent, $R_i(A_i) = R_j(A_j)$ is recorded.

III. Acquiring Ontology from a Relational Database

The task of acquiring ontology from a relational database can be divided according to which constructs, characteristics respectively, are acquired. Further, only the following constructs and characteristics of the created ontology shall be considered: classes, properties, hierarchy, cardinality, instances.

The principle of building ontology from a relational database, as is described by the set of rules in the following chapters, presents the reverse procedure to the procedure of transforming the conceptual model to a relational model. This transformation is based on the definition of a relational data model and transforms constructs designed in the conceptual model to constructs which permit a relational data model. This involves primarily decomposition of relations in the N:M cardinality and a conceptual model using multi-value and group attributes. Further, the issue of compulsory or optional participation of entities in relation to a conceptual model entity is discussed. The building of ontology is then based on the revealing of the original conceptualization, which the transformation process rendered unnoticed and thus the original semantic of modelled reality was lost (hidden). We propose following set of rules that allows inverse process to the process of transforming conceptual model into relational model. These rules transform relational model into conceptual model or ontology.

A. Rules for creating classes

The creation of ontology concepts as "classes" is based on two rules.

Rule I:

If we have database relations R_1, R_2, …, R_i, where $P_1 = pkey(R_1)$, $P_2 = pkey(R_2)$, …$P_i = pkey(R_i)$, for which $R_1(P_1) = R_2(P_2) = \ldots = R_i(P_i)$ apply, then information from these database relations can be integrated into one ontological class C_i. These relations form one entity on a conceptual abstract level.

Rule II:

Ontological class C_i can be created from relation R_i, if no other relation exists which could be integrated with relation R_i according to Rule I. and, at the same time, one of the following conditions applies:

$|pkey(R_i)| = 1$

$|pkey(R_i)| > 1$ and, at the same time, it applies that A_i exists, where $A_i \in pkey(R_i)$ and also $A_i \notin fkey(R_i)$.

This rule expresses that if relation R_i is used to describe an identification independent entity, then it may be mapped into one ontological class. The following example shows a practical case of mapping relations into an ontological class.

Example I.

A relational database schema consists of relations, their attributes, primary and foreign keys, as is shown in the following table.

Table I.
Relational Database Schema

Relation	Primary key	Foreign key
Student(Pers_number, Name,...., Town_residence)	Pers_number	Town_residence refers to relation Town Id_town
PhDStudent(Pers_number, …, Trainer)	Pers_number	Pers_number refers to relation Student Pers_number
Town(Id_town, Name_town, …..)	Id_town	-
Department(Id_department, Name_department, …)	Id_department	-
Studies(Pers_number, Id_department, ….)	Pers_number, Id_department	Pers_number refers to relation Student Pers_number
Employee(Id_employee, ….)	Id_employee	Id_department refers to relation Department Id_department

Example II.

According to Rule I. it is possible to create ontological class `Student`, which shall integrate relations `Student` and `PhDStudent` because `Student(Pers_number) = PhD Student(Pers_number)`. In OWL this class can be created as

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

According to Rule II. it is possible to create ontological classes `Town`, `Department` and `Employee` (condition 1 of this rule is fulfilled, $|pkey(R_i)| = 1$, i.e. number of attributes of primary key is one. In OWL classes can be created:

```
<owl:Class rdf:ID = "Town" />
<owl:Class rdf:ID = "Department" />
<owl:Class rdf:ID = "Employee" />.
```

B. Rules for creating properties

In OWL there are two types of properties – object properties and data type properties. The following rule must be defined for capturing properties arising from the relation schema into ontology.

Rule III:

If R_i and R_j are relations, where $R_i(A_i) \subseteq R_j(A_j)$ and, at the same time, $A_i \not\subset pkey(R_i)$, object property of ontology P can be created, based on attributes A_i of relation R_i. Assuming that relation R_i was mapped into class C_i and relation R_j was mapped into class C_j the domain of P is C_i and range of P is C_j.

Rule IV:

If R_i and R_j are two relations, then it is possible to create two object properties "has_part" and "is_part_of", if the following two conditions are fulfilled:

$|pkey(R_i)| > 1$

$fkey(R_i) \subset pkey(R_i)$, where $fkey(R_i)$ refers to R_j

Provided that R_i is mapped on C_i and R_j on C_j, the domain, or range respectively, of the "is_part_of" C_i, C_j respectively, and the domain, or range respectively, of the "has_part" is C_j, C_i respectively.

Rule V:

Let R_i, R_j and R_k be relations and let it apply: $A_i=pkey(R_i)$, $A_j= pkey(R_j)$, $A_i \cup A_j = fkey(R_k)$ and $A_i \cap A_j = \emptyset$, the two ontology object properties can be created P_j' and P_j'' based on R_k. Provided that R_i is mapped on C_i and R_j on C_j, domain and range of P_j' is C_i and C_j and domain and range of P_j'' is C_j and C_i. Pj' and P_j'' are two inverse object ontology properties.

Example III.

According to Rule III. `Town_residence` can be created as an ontology object property. The domain of this concept is `Student`, range is `Town`. In OWL the creating of a concept can be written as:

```
  <owl:Object Property
 rdf:ID="Town_residence" />
  <rdfs:domain rdfs:source="#Student " />
  <rdfs:range rdfs:source="#Town " />
  <owl:ObjectProperty />.
```

Example IV.

According to Rule V., two object properties can be created based on the `Studies` relational relation. We'll call these properties `pertains_to` and `has_student`, which shall be associated with classes of ontology `Student` and `Department`. The corresponding record in OWL is as follows:

```
  <owl:Object Property rdf:ID = "
pertains_to " />
  <rdfs:domain rdfs:source = "#Student
" />
  <rdfs:range rdfs:source = "#Department
" />
  <owl:ObjectProperty />
  <owl:Object Property rdf:ID =
"has_student " />
  <rdfs:domain rdfs:resource =
"#Department " />
  <rdfs:range rdfs:resource =
"#Student" />
  <owl:inverseOf rdf:resource="
pertains_to " />
  <owl:ObjectProperty />.
```

RuleVI:

Rule VI. defines the method of creating data type properties and specifies that each attribute of all relations of the relational schema, which cannot be transferred to an object property, can be transferred to a data type property.

For ontology class C_i and a set of data type designated as $DP(C_i)$, if C_i correspond to database relations R_1, R_2, …R_i, then for each attribute in R_1, R_2, …R_i, for which an object property cannot be created subject to Rule III., a data type property can be created. The domain of each property P_i is C_i and the range of each property is $dom(A_i)$ for each $P_i \in DP(C_i)$ and $A_i \in attr(R_i)$.

Example V.

According to Rule VI. it is possible to create data type properties corresponding to ontology classes from attributes of relational schemas `Pers_number`, `Name`, `Trainer`, `Id_town`, `Name_town`, `Id_department`, `Name_department`, `Id_employee`. The segment in OWL could be as follows:

```
  <owl:Datatype Property rdf:ID =
"Pers_number" />
  <rdfs:domain >
    <owl:Class >
  <owl:unionOf rdf:parseType =
"Collection " />
  <owl:Class rdf:about = " #Student " />
  <owl:Class rdf:about = " #PhDStudent
" />
  <owl:Class rdf:about = " # Pers_number
" />
  </owl:unionOf >
  </owl:Class >
  </owl:domain >
  <rdfs:range rdfs:source = "&xsd;int
" />
  <owl:DatatypeProperty />
  …
```

C. *Rules for creating hierarchies*

As is well known, the conceptual model uses the so-called Isa hierarchy, which enables the application of heritability in the object model. This Isa hierarchy does not permit the relational data model and therefore this Isa hierarchy must be decomposed during transformation of the conceptual model to a relational data model. Usually the identification dependence of relations is used, i.e. the identification key of the super-type becomes the identification key of the sub-type and also its foreign key. Conversely it can be deduced that if a relation has a foreign key as a part of its primary key, then it is a relation which was created by an Isa hierarchy transformation.

RuleVII:

If R_i and R_j are relations and it applies that $P_i=pkey(R_i)$ and $P_j=pkey(R_j)$, then if Rule I. cannot be applied and $R_i(P_i) \subseteq Rj(P_j)$ is fulfilled, then class/property of the corresponding R_i is a sub-class/sub-property of the corresponding R_j.

Example VI.

According to Rule VII., ontological class `PhDStudent` is a sub-class of ontological class `Student` and can be

recorded in OWL as follows:

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

D. Rule for creating cardinalities

Attributes of relations according to the domain integral limitation designed above individual attributes of relations may acquire no, only one, at least one, maximum of one or more values. Based on a specified domain integral limitation in the relational schema the property cardinalities can be defined in the ontology.

Rule VIII:

If R_i is a relation with attributes $A_i \in attr(R_i)$, where $A_i = pkey(R_i)$, then the minimum and, at the same time, maximum cardinality corresponding to property P_i is equal to 1.

Rule IX:

If R_i is a relation with attributes $A_i \in attr(R_i)$, where A_i is declared as NOT NULL, then the minimum cardinality corresponding to property P_i is equal to 1.

Rule X:

If R_i is a relation with attributes $A_i \in attr(R_i)$, where A_i is declared as UNIQUE, then the maximum cardinality corresponding to property P_i is equal to 1.

Example VII.

According to Rule VIII. the minimum and, at the same time, maximum cardinality of attributes `Pers_number, Id_town, Id_department` is equal to 1. In OWL it is recorded as:

```
<owl:Restriction />
<owl:onProperty rdf:resource = "#Pers_number " />
<owl:minCardinality>1</owl:minCardinality>
<owl:maxCardinality>1</owl:maxCardinality>
</owl:Restriction >
<owl:Restriction />
<owl:onProperty rdf:resource = "#Id_town " />
<owl:minCardinality>1</owl:minCardinality>
<owl:maxCardinality>1</owl:maxCardinality>
</owl:Restriction >
<owl:Restriction />
<owl:onProperty rdf:resource = "#Id_ " />
<owl:minCardinality>1</owl:minCardinality>
<owl:maxCardinality>1</owl:maxCardinality>
</owl:Restriction >
```

E. Rule for creating instance

The following rule can be used for automatic creation of ontology instances from data stored in relational databases.

Rule XI:

If C_i is an ontological class of the corresponding to database relations R_1, R_2, …R_i, then each n-tuple of t_i, for which $t_i \in R_1 \times R_2 \times \ldots \times R_i$ applies, can become an instance C_i.

Note: x designates the Cartesian sum of relations.

Fig 1. Graph shows the fragment of the ontology created by describing examples using Altova SemanticWorks software.

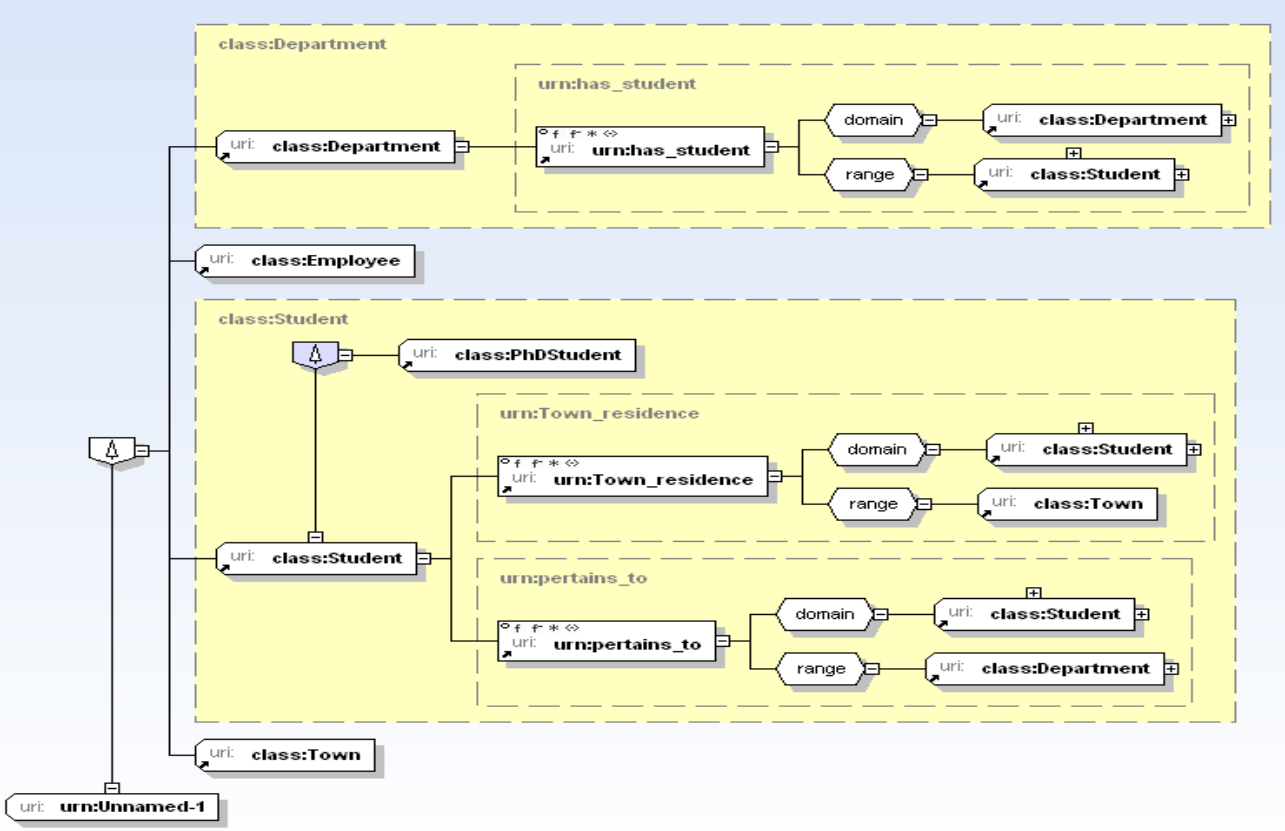

Fig. 1 Graph

IV. Conclusion

Mapping relational data into ontology, or filling ontology with data from relational databases respectively, plays an important role during the creation and updating of ontology. If we consider the quantity of data found in relational databases and the potential of their joint use across various applications, the question of finding a method for their use becomes obvious. The heterogeneity of data complicates use and therefore methods of their integration are being sought. The method based on a common semantic model – ontology is one way of efficient integration. The question is how to correctly map relational database data into ontology instances. The quantity of manual work during creation of ontology poses a serious problem; therefore, automation based on mapping subject according to the rules is a god solution. This article focuses on the principles of automatic conversion of constructs of the relational data model to constructs of OWL ontology and transfer of relational data to ontology instances. Mainly, rules are defined which map the basic concepts of the relational data model onto ontology. Individual rules are complemented by specific examples. Examples utilize a common simple data model.

References

[1] Miller, R., Haas L. M., Hernandez M. A.: Schema Mapping as Query Discovery. In VLDB'00, pages 77–88, 2000.
[2] Biskup, J.: Achivements of relational database schema design Tudory revisited. Semantics in Database. LCNS 1358, Springer Verlag, 1998
[3] Kashyap, V.: Design and creation of ontologie for environmental information retrieval. Proc. Of the 12 th Workshop on Knowledge Acquisition, Modeling and Management. Alberta, Canada. 1999.
[4] Sicilia, M.A., Lytras, M.D.: Metadata and Semantics. Springer, 2009.
[5] Rishe N. Database design: the semantic modeling approach. McGraw-Hill, 1992.
[6] Biskup, J.: Achievements of relational database schema design theory revisited. Semantics in Database, LCNS 1358, Springer Verlag, 1998.
[7] Chiang R, Barron T, Storey V. Reverse engineering of relational databases: extraction of an EER model from a relational database. Journal of data and knowledge engineering, 1994,12(2):107-142.
[8] Vermeer M, Apers P. Object-oriented views of relational databases incorporating behaviour. DASFAA, 1995.
[9] Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web Sites into Semantic Web. ACM press, 2002.