Approach for Mapping Ontologies to Relational Databases

A. Rozeva

Technical University – Sofia

E-mail: arozeva@tu-sofia.bg
INTRODUCTION

Research field – mapping ontologies to databases

Research goal –

• facilitation of knowledge production and the development of semantic web applications;

• establishment of efficient storage and retrieval techniques and mechanisms for ontology;

• approach for ontology to database schema mapping.

Research environment – semantic web
MOTIVATION

To facilitate the storage of instance rich ontology and provide for its manipulation by using programmable objects.
ONTLOGIES vs. Relational Databases

Ontologies

• enable web applications to interchange and retrieve integrated information
• main processing requirement concerns inference capabilities
• lack processing standard.

Relational Databases

• tuned for performance and scale
• standardized on SQL
• main source for the generation of web content.
BACKGROUND - Ontologies to Database Mapping

Establish a model of the ontology in the relational / object-relational paradigm

• provide for the interoperation of ontology with relationally stored data
• make ontology data accessible to relational database applications.

Result:
Make semantic web really useful by enabling effective querying and reasoning on huge number of ontology instances.
Mapping Approach

Implement **mapping rules** concerning the correspondences between **OWL ontology components** and **relational database scheme elements**, i.e.:

- Class-to-Table mapping
- Property-to-Table mapping
- Property-to-Column mapping
- Property restriction-to-Column constraint mapping
Mapping Approach

• Class-to-Table mapping –
  • the differentiation between root classes and subclasses is provided by the definition of tables’ primary keys;
  • class-subclass path in the taxonomy is mapped to tables linked with foreign key relationships.

• Property-to-Table mapping (for multivalued properties)
  • object property with specified domain and range representing a class, i.e. it denotes many-to-many relationship between classes
  • datatype property, which links individuals to a literal value and no cardinality restriction is specified.
Mapping Approach

- Property-to-Column mapping–
  - single-valued datatype property - column in the table, referring to the class, specified as property domain.
  - object property with a restriction specifying its cardinality to 1 with an inverse property is - column representing a foreign key to the table of the domain class.
  - single-valued object property with an inverse multivalued property corresponds to one-to-many relationship - column in the table specified as range class.
Mapping Approach

- Property restriction-to-Column constraint mapping
  - Inverse functional property restriction - Unique constraint of the corresponding column.
  - MinCardinality (1) property restriction - Not Null constraint.
  - Restriction on the values of a datatype property - Check constraint on the corresponding column.
  - Inverse functional property with minimal cardinality of 1 restriction - primary key.
  - Inverse functional object property - foreign key.
Mapping Architecture

International Scientific Conference
"ComputerScience'2015"
Algorithm for Table Generation

Input: OWL file, Output: SQL Create Table statements.
Step1: Extract classes in a set // OWL:Class rdf: ID="Name"
Step2: Extract object properties in a set
Step3: Extract datatype properties in a set
Step4: With Class; Do // check each class from the set
    Step5: If Class \rightarrow root class Then // rdfs: subClassOf rdf: resource = "Thing"
          Step6: Generate SQL Create Table statement with ID as primary key
          Step7: If Class has subclasses Then // rdfs: subClassOf rdf: resource = "Class"

          Step8: if subclasses are disjoint Then
              Step9: For each subclass Do
                  Step10: Generate SQL Create Table statement with Class:ID as primary key
                  End Step 9
              End Step 8
          End Step 7
    End Step 4
Step11: With ObjProp; Do // check each object property from the set
    Step12: If has ObjProp: inverse and has ObjProp: no restrictions Then
        Step13: Generate SQL Create Table statement with compound primary key //
        from the foreign keys to the domain and range classes
    End Step 12
Step14: With DataTypeProp; Do // check each datatype property from the set
    Step15: If DataTypeProp: has no cardinality restriction
        Step16: Generate SQL Create Table statement with composed primary key //
        from the foreign keys to the domain class and the property value
    End step 14.
Algorithm for Column Generation

Input: Object property queue, Datatype property queue, Output: SQL Add Column statements.
Step1: With ObjPropᵢ Do // check each object property from the set
Step2: If has ObjPropᵢ inverse and has ObjPropᵢ cardinality (1) Then
Step3: Generate SQL Alter Table domainClass; Add Foreign Key ObjPropᵢ references rangeClass statement
End Step 1
Step4: With ObjPropᵢ Do // check each object property from the set
Step5: If has ObjPropᵢ inverse and has ObjPropᵢ cardinality (>1) Then
Step6: Generate SQL Alter Table rangeClass; Add Foreign Key ObjPropᵢ references domainClass statement
End Step 4
Step7: With DataTypePropᵢ Do // check each datatype property from the set
Step8: If DataTypePropᵢ has cardinality restriction with rangeClass
Step9: Generate SQL Alter Table rangeClass; Add Column DataTypePropᵢ statement
End step7.
Algorithm for Column Constraints Generation

Input: Object property queue, Datatype property queue, Output: SQL Add Constraint statements.

Step1: With ObjProp; Do // check each object property from the set
    Step2: If is ObjProp; inversefunctional Then
          Step3: Generate SQL Alter Table domainClass; Add Constraint UC Unique(rdf:ID=”name”) statement
          Step4: If has ObjProp; minCardinality(1) Then
                 Step5: Generate SQL Alter Table domainClass; Modify rdf:resource=”name” Not Null statement
          End Step1

Step6: With DatatypeProp; Do // check each datatype property from the set
    Step7: If has DatatypeProp; DataRange oneOf List Then
           Step8: Generate SQL Alter Table domainClass; Add Constraint CC Check(condition) statement
    End Step6
Sample OWL Syntax and Mapping Implementation

```xml
<owl:Class rdf:ID="Organization Unit">
  <owl:Class rdf:ID="Resource">
    <rdf:ObjectProperty rdf:ID="hasResource">
      <rdfs:domain rdf:resource="# Organization Unit"/>
      <rdfs:range rdf:resource="# Resource"/>
    </rdf:ObjectProperty>
    <rdf:ObjectProperty rdf:ID="BelongsTo">
      <owl:inverseOf rdf:resource="hasResource"/>
    </rdf:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="Rno">
      <rdfs:domain rdf:resource="# Resource"/>
      <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="HumanResource">
      <rdfs:subClassOf rdf:resource="# Resource"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="kind">
      <rdfs:domain rdf:resource="# HumanResource"/>
      <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="HumanResource">
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="kind"/>
          <owl:hasValue rdf:resource="Human"/>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>
  </owl:Class>
</owl:Class>

CREATE TABLE Resource (  
  ResourceID AUTO_INCREMENT PRIMARY KEY,  
  Rno INTEGER CHECK (Rno>0) UNIQUE)  
CREATE TABLE Organization Unit (  
  Organization UnitID AUTO_INCREMENT PRIMARY KEY)  
CREATE TABLE hasResource (  
  ResourceID INTEGER REFERENCES Resource,  
  Organization UnitID INTEGER REFERENCES Organization Unit)  
CREATE TABLE HumanResource (  
  ResourceID INTEGER PRIMARY KEY REFERENCES Resource)  
  kind VARCHAR CHECK (kind="Human")
```
CONCLUSION

Ontology to relational database mapping is **applicable** and **meaningful** besides the fact that they are based on different assumptions: databases implement the closed-world assumption while ontologies share the open world assumption.

Approaches for mapping OWL ontology to relational database schema are based on **rules**, which state the correspondences between ontology classes, object and datatype properties and restrictions to relational database tables, columns and constraints.
CONCLUSION

The proposed approach is based on architecture involving ontology parser and SQL generators and provides for the persistent storage of ontologies and for the implementation of database navigation and retrieval technologies.
FUTURE WORK

- Investigation of the generation of ontologies from relational databases.

Thank you!