ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Onto Extractor : A Tool for Ontology Extraction from ER/EER diagrams

Kurman Sangeeta
Assistant Professor
CSE Dept., AITAM
Tekkali, India

Palakollu Srinivasa Rao
Assistant Professor
CSE Dept., AITAM
Tekkali, India

Abstract: Most of the existing data based web applications uses relational data sources or XML based data sources. For these data sources to participate in semantic web, to enable semantic based data access and integration, their associated database schema must be mapped to Ontology. In order to preserve the domain information, we proposed mapping rules to transform ER/EER diagram to semantically equivalent ontology. OntoExtractor automatically converts a XML file representing an ER/EER diagram to Ontology in OWL-DL. The output of OntoExtractor is a conceptual schema in the form of Ontology. We proposed a more efficient concept capturing approach of mapping ternary relation and M:N relation with attributes. We captured key attribute consisting of more than one attribute and composite key attribute. In addition to that, we have presented a DTD for a well-formed ER/EER schema, so that any ER schema can be captured in machine-interpretable XML file.

Keywords: semantic web, Ontology, DTD, ER/EER schema, ERD(Entity Relationship Diagram)

I. INTRODUCTION

Ontology can be extracted from relational database systems through reverse engineering process. But this process induces loss of information such as inverse role relation, composite attributes, class hierarchy, multiple inheritance, multivalued attributes. Hence we need a loss less technique of semantic preservation in ontology. ER diagram captures the most of the semantics of domain of interest. Hence it is a good choice for schema mapping so that legacy data in relational databases can participate in ontology based semantic web. This work proposes mapping rules for extraction of ontology from ERD.

A. Semantic Web

The current web is full of information in terms of HTML/XML documents which can be displayed to human as a result of keyword search[1]. Machines cannot process these information to draw logical inferences. Semantic web, on the other hand is proposed to realize machine to machine interaction. Semantic web is based on metadata rich information resources which consists of structured sets of information and relationship between them. Semantic web is based on ontology. Today's WWW can be upgraded to semantic web by using either of the following approaches.

- Defining ontology over existing database sources such as relational database sources.
- Create an ontology of a domain from scratch.

Creating an ontology from scratch is a time consuming and error-prone process. In this paper, the first approach is considered.

B. Ontology

Ontology [2] is a conceptual model, describing the concepts in a domain of discourse, properties among concepts, constraints on properties. Ontology form the sources of standard vocabulary in the semantic web to describe data. There are many languages proposed to build ontology such as RDFS, OIL, DAML, OWL etc. But OWL is the W3C standard language to build ontology. OWL ontology includes machine-interpretable definitions of concepts in the domain and relationships among them. For example, the university ontology consists of concepts like Student, Staff, Courses etc. These classes represent natural concepts. Ontology is needed for the following reasons :

- For the legacy databases to participate in semantic web.
- To enable semantic based data access.
- For integration of various data sources.
- To develop a shared common understanding of domain so that interoperability issues between heterogeneous systems can be resolved.
- To facilitate reuse of domain knowledge.

II. BACKGROUND AND RELATED WORK

Several research works were carried out to use or access legacy data as a part of semantic web. Presently the available literature suggests two main approaches to the problem of extracting ontologies from relational database schemas.

- Reverse engineering of relational database system.
- Mapping ERD to ontology.

A. Reverse engineering of relational database system

DBRE(Database reverse engineering) is a process of obtaining a higher level of abstraction basically conceptual schema from relational database system. Earlier works on DBRE to extract ontology can be broadly classified under two classes. DBRE based on analysis of relational schema and DBRE based on analysis of data.

The author[3] has suggested two approaches to resolve semantic heterogeneity of various data sources. Either to create an ontology from a database or map an existing ontology to database. DB2OWL is a tool implementing the first approach. In this approach[4] the constructs of logical data model can be found from SQL-DDL and were mapped into Frame-Logic(F-logic) rules[5]. F-logic is then mapped into RDF statements thereby migrating the relational data to

ontology. Another interesting approach[6] uses DBRE based on the analysis of relational schema to extract ontology. This is followed by refinement of ontology based on user queries. This approach fails to create axioms for the ontology. The authors[7] proposed an approach for automatic extraction of ontology using DBRE techniques given the relational schema as well as constraints describing the semantics of data. Drawbacks of the above approaches is that if the database is not be normalized, normalized but may not be in 3NF form , bad-designed then the semantics of the underlying relational database can not be preserved in the ontology. In order to avoid the above drawbacks, the authors[[8] have suggested a form-driven approach of reverse engineering[9]. In this approach, ontology can be extracted by analysis of HTML forms and relational schema . User involvement is needed for analysis of HTML forms. The drawback with this approach is that the ontology gets changed every time the HTML form and data changes.

The ontology extracted from relational database schema using DBRE process does not preserve the complete domain semantics. The tool OntoExtractor is based on the second approach of extraction of ontology i.e mapping ER model to OWL ontology. OntoExtractor extracts ontology from ER/EER diagram and hence preserves most of the domain semantics.

B. Mapping ERD to Ontology

This approach is based on extraction of ontology from ERD by defining certain mapping rules. The author[10] did not specify mapping rules for EER concepts and primary key consisting of more than one attribute. The author[11] failed to give mapping rules for ternary relation and M :N binary relation containing attributes. key attribute is mapped as functional and inverse functional datatype property. Mapping Key attribute as inverse functional datatype property is not allowed in OWL-DL and OWL-lite.

We map ER model to ontology in OWL-DL. OWL-DL supports features to describe union, intersection and complement of classes unlike OWL-Lite which has feature to define intersection of classes. Also in OWL-Lite, min cardinality and max cardinality can be set to either zero or one. But OWL-DL has features to set min cardinality and max cardinality to any value ranging from zero to N. Though OntoExtractor can not map n-ary relation of order greater than 3 and advance EER concepts like aggregation, but maps most of the basic components of ER/EER diagram. Also the input to the tool OntoExtractor is XML-file that represents the ER/EER diagram.

III. MAPPING ER/EER DIAGRAM TO XML FILE

Here is a DTD for well-defined ER/EER diagram, so that any ER schema can be captured in machine-interpretable XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ERM (strongE+,weakE*,relationship*
        ,SuperClass*)>
<!ELEMENT strongE (Attribute*,key)>
<!ELEMENT weakE (Attribute*,partialKey)>
<!ELEMENT key (Attribute+)>
<!ELEMENT partialKey (Attribute+)>
<!ATTLIST strongE Name ID #REQUIRED>
<!ATTLIST weakE Name ID #REQUIRED owner IDREF
#REQUIRED>
<!ELEMENT Attribute (simple|multivalued|
                                composite|derived)>
<!ELEMENT simple (#PCDATA)>
<!ELEMENT multivalued (#PCDATA)>
<!ELEMENT composite (Attribute+)>
<!ELEMENT derived (#PCDATA) >
<!ELEMENT relationship (Attribute*,role+) >
<!ELEMENT role (card) >
<!ELEMENT card (min,max) >
<!ELEMENT min (#PCDATA) >
<!ELEMENT max (#PCDATA) >
<!ATTLIST Attribute domain CDATA #REQUIRED >
<!ATTLIST relationship relName ID #REQUIRED >
<!ATTLIST role ParticipantEntityName IDREF #REQUIRED
>
<!ATTLIST role type CDATA #REQUIRED >
<!ATTLIST composite name ID #REQUIRED >
<!ELEMENT SuperClass (SubClass+)>
<!ELEMENT SubClass (Attribute*,key*)>
<!ATTLIST SuperClass EntityName IDREF #REQUIRED
total
(yes|no) #REQUIRED>
<!ATTLIST SubClass SubEntityName ID #REQUIRED
disjoint
 (yes|no) #REQUIRED>
```

Line 2 – 15 : The root node ERM(Entity Relationship model) consists of one or more(+) strong entities *strongE*, zero or more(*) weak entities *weak*, zero or more(*) relationships *relationship* and zero or more(*) superclasses *SuperClass*. Zero relationship correspond to cases where only EER diagram is given. The strongE element consists of zero or more(*) attributes *Attribute* and one primary key *key*. The primary key may consists of one or more(+) attributes. Each strongE and weakE has a distinct name that is an ID for these elements. weakE element also contains an IDREF *owner* to the strongE element which is the owner entity. weakE element consists of zero or more(*) *Attribute* and one *partialKey* element. The *Attribute* element can be either simple, or multivalued, or composite or derived. Since a composite attribute further consists of other attributes, the *composite* element has one or more *Attribute* element. The other elements *simple*, *multivalued*, *derived* take their names as *PCDATA* value. The *composite* element has attribute *name* that captures the name of the composite attribute.

Line 16 – 20 : A relationship in ERD may have zero or more attributes and two or more participating entities. Hence, the relationship element contains zero or more(*) *Attribute* elements and one or more *role* elements. Since for each role, participation and cardinality restriction is specified in ERD, the *role* element has *card* element. Each *card* element consists of *min* and *max* elements representing cardinality ratios. Total participation of a role in a relationship in ERD is captured by assigning *PCDATA* value of 1 to *min* element.

Line 21 – 25 : The *Attribute* element has attribute *domain* that specifies attribute's XSD data type such as integer, string etc. Each relationship has a unique name that is specified by attribute *relName* of relationship element. Each *role* element has an attribute *ParticipatingEntityName* which is an IDREF to an entity participating in the relationship. The *role* element has attribute *type* which specifies whether that role is domain or range of the relationship in case of binary relationship.

Otherwise it specifies type of relation such as ternary or quaternary etc for all roles participating in the relationship.

Line 26 – 31: We consider that superclass refers to an entity and subclass as a separate concept. As a superclass in ERD has one or more subclasses, *SuperClass* element includes one or more(+) *SubClass* elements. A subclass may have an additional attribute or key other than the superclass. Hence the *SubClass* element has zero or more(*) Attribute elements or zero or more(*) key element. The *SuperClass* element has an attribute *EntityName* which is an IDREF to an entity. The *SubClass* element has an ID attribute *SubEntityName* specifying the name of the subclass. For Generalization/Specialization concepts in EER diagram, we tried to capture only basic types like (disjoint,total), (disjoint ,partial), (overlapping ,total), (overlapping,partial) through attributes *total* and *disjoint*.

If attributes belonging to two different entities have the same name then we are prefixing attribute name with entity name in the XML file. Similarly attribute of a relation and attribute of an entity or attribute of another relation have the same name then attribute name is prefixed with the corresponding relation or entity name. This is needed to ensure unique name for the attributes.

IV. MAPPING RULES FOR ER/EER TO OWL CONVERSION

ER Components	OWL-DL Components
Entity	Class
Identifying Relationship	Functional object property with range as owner entity and domain as the weak entity. Another inverse non-functional object property with domain as owner entity and range as weak entity.
Simple or derived Attribute	Functional datatype property
Multivalued Attribute	Datatype property
Composite Attribute	Class with components attributes as datatype properties. object property with domain as parent entity class and range as the new composite class.
Key	key attributes as data type properties to key class. Functional and inverse functional Object property. domain as Strong entity class and range as key class. Cardinality restriction set to (1,1) for both domain and range.
Binary Relationships	
a. Without attributes	a Pair of object properties which are inverse to each other.
one-one	Cardinality restriction on object property: Domain :(0,1) | (1,1) Range:(0,1)|(1,1) Minimum cardinality=1 for total participation.
one-many	Cardinality restriction on object property: Domain :(0,n) |(1,n)Range :(0,1) |(1,1).Minimum cardinality=1 for total participation.
many-many	Cardinality restriction on object property:Domain :(0,n)|(1,n) Range : (0,n) |(1,n) Minimum cardinality=1 for total participation.
b. With attributes	
one-one	Same as the one for without attributes. Datatype property corresponding to attribute is added to entity with total participation side if present or to any side preferably domain side.
one-many	Same as the one for without attributes.Datatype property corresponding to attribute is added to entity of many side
many-many	Class with name of relation. Datatype property corresponding to attribute is added to the above class. A pair of object properties, which are inverse to each other, between above class and each of the participating entities. Cardinality restriction same as the one without attributes.
Superclass and subclass	Superclass and subclass
Disjoint and total	Union of subclasses which are disjoint.
Disjoint and partial	Disjoint subclasses.
Overlapping and total	Union of subclasses.
Overlapping and partial	No restriction.
Ternary relationships	Class with name of ternary relation. 3 pairs of object properties, which are inverse to each other, between the above class and each of the three entities.

V. RESULTS

The ERD of Company database[13] is given below.

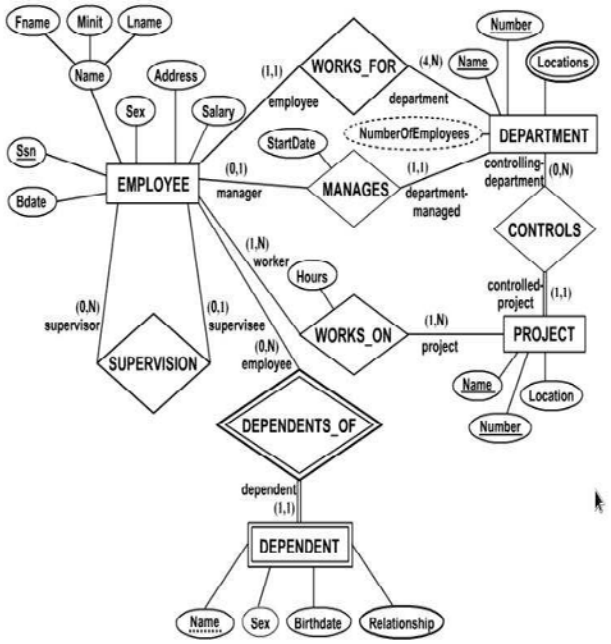

Figure. 1. ER diagram of Company database [13]

The ERD of Company database consists of

Strong entities:

- *EMPLOYEE* : The primary key is *Ssn*. *Name* is a composite attribute consisting of *Fname*, *Minit* and *Lname*
- *DEPARTMENT* : The primary keys are *Name* and *Number*. *Locations* is a multivalued attribute. *NumberOfEmployees* is a derived attribute
- PROJECT : The primary keys are *Name* and *Number*

Weak entities :

- DEPENDENT : The owner entity is EMPLOYEE entity. *Name* is a partial key.

Relationship :

- WORKS_FOR: A department has several employees but each employee works for one department.
- MANAGES: Each department has an employee who manages the department. Those employees who are managers have a start date. *StartDate* is an attribute.
- WORKS_ON : Each employee can work on several projects. *Hours* is an attribute which represents the number of hours an employee works on a project. .
- CONTROLS: Each department controls a number of projects. PROJECT is in total participation with DEPARTMENT.
- SUPERVISION : This is a recursive relationship between EMPLOYEE and EMPLOYEE entities. An employee may be a supervisor or a subordinate. Recursive relationship involves the same entity.
- DEPENDENTS_OF : This is an identifying relationship between EMPLOYEE and DEPENDENT entities.

The XML file capturing the Strong entity EMPLOYEE according to the above DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ERM SYSTEM "ERD.dtd">
<ERM>
<strongE Name="EMPLOYEE" >
<Attribute domain="string"><simple >EMPLOYEE_Sex</simple></Attribute>
<Attribute domain="integer"><simple >Salary</simple></Attribute>
<Attribute domain="date"><simple >Bdate</simple></Attribute>
<Attribute domain="string"><simple >Address</simple></Attribute>
<Attribute>
 <composite name="EMPLOYEE_Name">
   <Attribute domain="string"><simple >Fname</simple></Attribute>
   <Attribute domain="string"><simple >Minit</simple></Attribute>
   <Attribute domain="string"><simple >Lname</simple></Attribute>
  </composite>
</Attribute>
<key >
  <Attribute domain="integer"><simple >Ssn</simple></Attribute>
</key>
</strongE>
```

Figure 2 : XML file capturing Strong entity EMPLOYEE

```
<strongE Name="DEPARTMENT">
<Attribute
 domain="string"><multivalued>DEPARTMENT_Locations</multivalued></Attribute>
<Attribute  domain="integer"><derived>NumberOfEmployees</derived></Attribute>
<key >
 <Attribute domain="integer"><simple>DEPARTMENT_Number</simple></Attribute>
  <Attribute
   domain="string"><simple>DEPARTMENT_Name</simple></Attribute>
</key>
</strongE>

<strongE Name="PROJECT" >
<Attribute domain="string"><simple >PROJECT_Name</simple></Attribute>
<Attribute domain="string"><simple >PROJECT_Location</simple></Attribute>
<key >
 <Attribute domain="integer"><simple >PROJECT _Number</simple></Attribute>
</key>
</strongE>
```

Figure 3: XML file capturing Strong entities DEPARTMENT and PROJECT

The corresponding OWL-DL components in order are shown in table[1]. EMPLOYEE_Name and EMPLOYEE_KEY are two new classes generated corresponding to composite attribute *Name* and *key* attribute respectively.

	EMPLOYEE
Functional Datatype property	hasEMPLOYEE_Sex, hasSalary,hasBdate. Domain of these properties is EMPLOYEE and range is string, integer, date xsd datatype respectively
Class	EMPLOYEE_Name
Object Property	hasEMPLOYEE_Name. Domain of this property is EMPLOYEE and range is EMPLOYEE_Name Functional Datatype property & hasFname, hasMinit, hasLname. Domain is EMPLOYEE_Name for these properties and range is string, string and string xsd datatype respectively.
Functional Datatype	hasFname, hasMinit, hasLname. Domain

property	is EMPLOYEE_Name for these properties and range is string, string and string xsd datatype respectively.
Class	EMPLOYEE_KEY
Functional and inverse functional Object Property	EMPLOYEE_isIdentifiedby_key. Domain of this property is EMPLOYEE and range is EMPLOYEE_KEY. Cardinality restriction (1,1) for both domain and range
Functional Datatype property	hasSsn. Domain of this property is EMPLOYEE_KEY and range is integer xsd datatype.

Table 1: OWL-DL components for EMPLOYEE entity

The ERD for ternary relationship and its XML file are given below

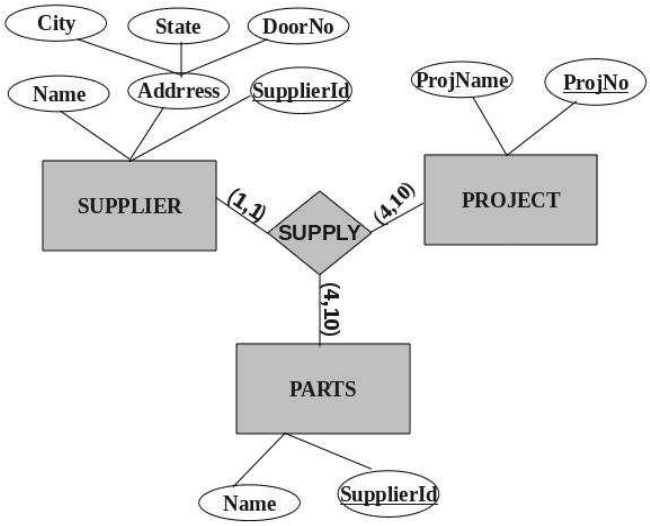

Figure. 4: ERD for ternary relation

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ERM SYSTEM "erd11.dtd">
<ERM>
<strongE Name="SUPPLIER" >

      <Attribute domain="string"><simple >SUPPLIER_Name</simple></Attribute>
      <Attribute>
          <composite name="Address">
              <Attribute domain="string"><simple >City</simple></Attribute>
              <Attribute domain="string"><simple >State</simple></Attribute>
              <Attribute domain="string"><simple >DoorNo</simple></Attribute>
          </composite></Attribute>
      <key >
          <Attribute domain="int"><simple >SupplierId</simple></Attribute>
                  </key>

  </strongE>
<strongE Name="PROJECT" >
      <Attribute domain="string"><simple >ProjName</simple></Attribute>

      <key >
          <Attribute domain="int"><simple >ProjNo</simple></Attribute>
                  </key>
```

```
  </strongE>
<strongE Name="PARTS" >

      <Attribute domain="string">
     <simple >PARTS_Name</simple></Attribute>
       <key >
          <Attribute domain="int">
          <simple >PartId</simple></Attribute>
       </key>

  </strongE>
<relationship relName="SUPPLY">
                <role ParticipantEntityName="SUPPLIER"
                    type="ternary">
          <card>
              <min>1</min>
              <max>1</max>
          </card>
       </role>

        <role ParticipantEntityName="PARTS"
             type="ternary">
             <card>
                  <min>4</min>
                  <max>10</max>
             </card>
        </role>
       <role ParticipantEntityName="PROJECT"
             type="ternary">
             <card>
                  <min>4</min>
                  <max>10</max>
             </card>
        </role>
    </relationship>
</ERM>
```

Figure 4: XML file for the above ternary relation

The snapshots of this ontology in Protege-OWL editor of Protege 4.0 version is shown below. The classes, datatype properties and object properties generated automatically from above ERD[figure 3] representing ternary relationship are shown in first, second and third column respectively.

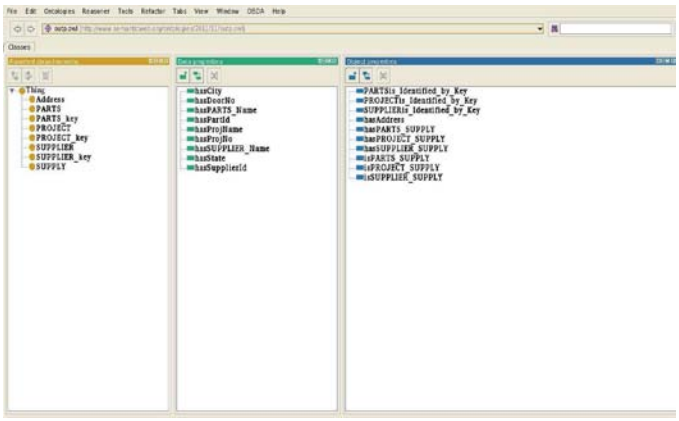

Figure 5 : Class view in Protege-OWL editor

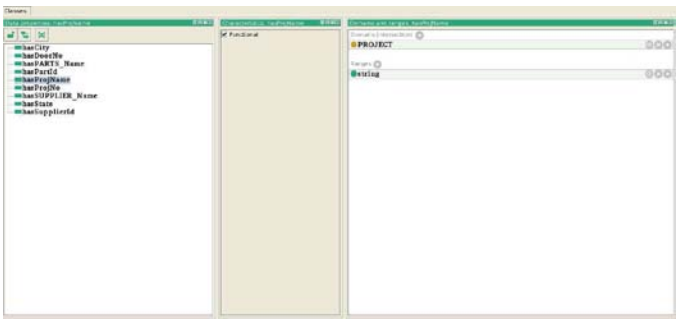

Figure 6: Data property view in Protege-OWL editor

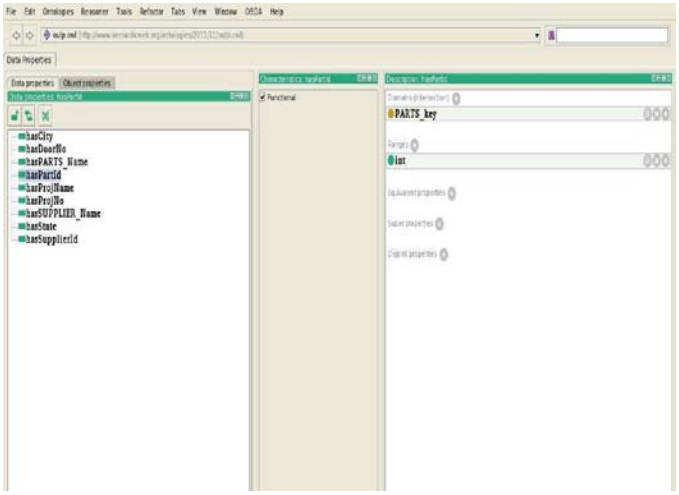

Figure 7: Data property view of key of PARTS class in Protege-OWL editor

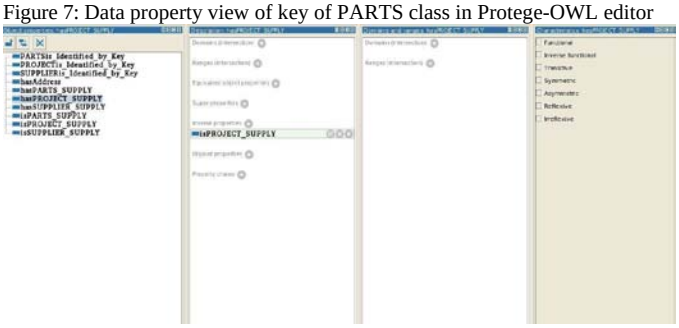

Figure 8 : The object property and its inverse between PARTS and SUPPLY

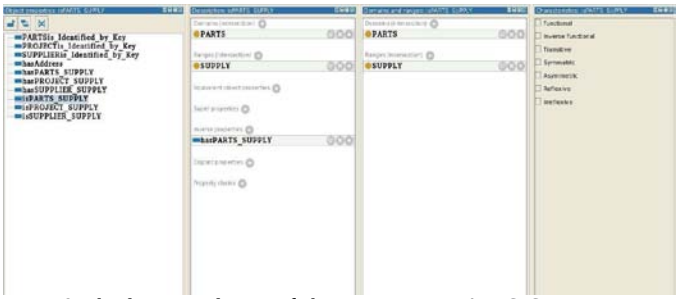

Figure 9: The domain and range of object property is PARTS_SUPPLY

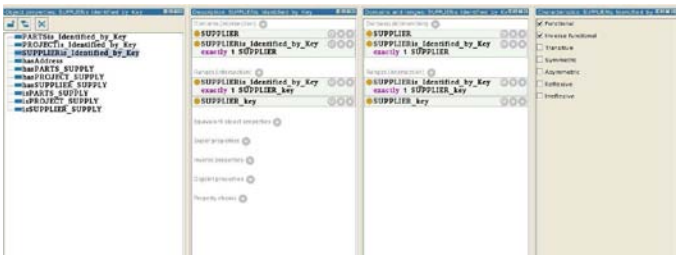

Figure 10: Object property view in Protege-OWL editor

VI. CONCLUSION

ER diagram captures most of the domain semantics and being the most widely used conceptual schema, the ontology extracted from this preserves most of the domain semantics. The ontology maps keys consisting of one or more attributes. We propose a novel mapping rules for 1:1 and 1:N relationships, possibly containing attributes thereby avoiding unnecessary classes and object properties. The ontology also maps the four different types of specialization. The names of object properties are automatically generated and hence may not be meaningful with respect to the context of their use.

VII. FUTURE WORK

Currently OntoExtractor maps only ternary and binary relationships. OntoExtractor can be extended to map n-ary relationship and complex specialization and generalization features such as multiple inheritance. Also the DTD for well-formed ER/EER diagram can be extended to support complex specialization and generalization features. Also DTD can be extended to include role name of participating entities which can be used to give meaningful names to relationships.

IX. REFERENCES

[1] Semantic Web, www.altova.com/semantic_web.html.
[2] Ontology, www.protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.
[3] N. Cullot, R. Ghawi, K. YÃl'tongnon, DB2OWL: A Tool for Automatic Database-to-Ontology Mapping, in: Proceedings of the 15th Italian Symposium on Advanced Database Systems, SEBD, 2007, pp. 491–494.
[4] L. Stojanovic, N. Stojanovic, R. Volz, Migrating data-intensive Web Sites into the Semantic web, in: Proceedings of the 17th ACM Symposium on Applied Computing, SAC, ACM New York USA, 2002, pp. 1100–1107.
[5] M. Kifer, G. Lausen, J. Wu, Logical Foundations of Object-Oriented and Frame-Based Languages, ACM 42 (1995) 741–843.
[6] V. Kashyap, Design and Creation of Ontologies for Environmental Information Retrieval, in: Proceedings of the 12th Workshop on Knowledge Acquisition,Modeling and Management, KAW'99, 1999.
[7] L. Lubyte, S. Tessaris, Automatic Extraction of Ontologies Wrapping Relational Data Sources, in: Proceedings of the 20th International Conference on Database and Expert Systems Applications, DEXA'09, Springer-Verlag Berlin, Heidelber, 2009, pp. 128 – 142.
[8] I. Astrova, Towards the Semantic Web-An Approach to Reverse Engineering of Relational Databases to Ontologies, in:Proceedings of the 9th East-European Conference on Advancesin Databases and Information Systems, ADBIS 2005, 2005, pp.111 – 122.
[9] N. Mfourga, Extracting Entity-Relationship Schemas from Relational Databases: A Form-Driven Approach, in: Proceedings of the 4th Working Conference on Reverse Engineering, WCRE 1997, 1997, pp. 184 – 193.
[10] I. Myroshnichenko, M. C. Murphy Mapping ER Schemas to OWL Ontologies, in: Proceedings of the 3rd IEEE International Conference on Semantic computing, ICSC'09, IEEE Computer Society Washington, DC, USA, 2009,pp. 324 – 329. doi:10.1109/ICSC.2009.61.
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5298643
[11] M. Fahad,ER2OWL: Generating OWL Ontology from ER Diagram, in:Proceedings of the 5th IFIP International Conference onIntelligent Information Processing, IFIP '08, Springer Boston,2008, pp. 28–37. doi:10.1007/978-0-387-87685-6_6.URL ttp://www.springerlink.com/content/9m28465576312mn7/fulltext.pdf
[12] DTD, www.w3.org/DTD/.
[13] E. Ramez, S. B.Navathe, Fundamentals of Database Systems,5th Edition, Pearson/Addison Wesley, 2007.