www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 6 June 2018, Page No. 23990-23996
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i6.05

Semantic Access of Relational Databases by mapping Ontology to Relational tables

K.Sangeeta

Assistant Professor, Department of CSE, Aditya Institute of Technology and Management,Tekkali.

Abstract:

Ontologies are used as the mediator for different data sources to participate in semantic web. Legacy databases such as the relational and XML based data sources are mapped to Ontologies and are accessed through the SPARQL queries. The proposed method includes the extraction of Ontology from ER/EER diagrams. The conversion of ER/EER diagrams to semantically equivalent Ontologies preserves most of the domain semantics. But in order to make them participate in the semantic web, an efficient method to access these data sources through SPARQL queries should be provided. The proposed method allows to access the underlying data sources (relational tables) by executing SPARQL queries on the Ontology extracted from ER/EER schema**.**

Keywords: SPARQL,SQL,RDF.

1. Introduction

Most of the existing data based web applications use either relational data sources or XML data sources. For these data sources to participate in semantic web and enable semantic based data access and integration, their associated database schema must be mapped to Ontology. In order to preserve most of the domain semantics of relational database systems, mapping rules are proposed to transform conceptual data models(ER/EER diagrams) into semantically equivalent Ontologies[1]. Assuming that the underlying relational database schema is derived from the conceptual model ER/EER schema, the framework proposed in the project enables the translation of SPARQL queries into corresponding SQL queries thereby accessing the underlying relational data sources. It Maps the Ontology to the relational tables efficiently. It converts almost all basic SPARQL queries on the Ontology created from ER diagrams to SQL queries.

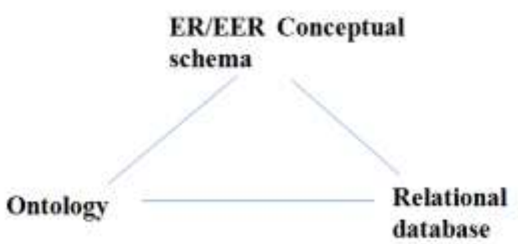


Figure 1: Mapping relational databases to semantic web

2. 2. Background and Related Works

2.1 R2O

A R2O mapping create instances in the Ontology from the data stored in the DB. It automatically populate an Ontology with information extracted from the content in the DB[2] .It does not allow Ontology instances to be converted to Database. Direct Mapping. A DB table directly maps a concept in the Ontology. Every record of the table will correspond to an instance of an ontology concept. Join/Union. A set of DB tables map a concept in the ontology when they are joined. Every join record of the joined tables correspond to an instance of an ontology concept. Projection. It appears when a subset of the columns of a DB table are needed to map a concept in the ontology.
Selection. A subset of the rows of a DB table map a concept in the ontology. Any combination of them are also possible.

2.2 Relational.OWL

An OWL-based representation format for relational data and schema components[3].The schema and data items originally stored in relational database systems are described using OWL ontology. It defines a OWL Full ontology to describe the schema and data of a RDB.

2.3 Virtuoso DBMS

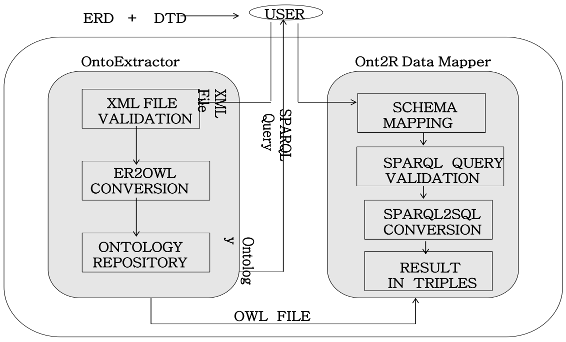
Virtuoso is a multi-protocol server providing ODBC/JDBC access to relational data stored either within Virtuoso itself or any combination of external relational databases[4]. Virtuoso's data storage consists of a single table of four columns holds one quad, i.e. triple plus graph per row. The columns are G for graph, P for predicate,S for subject and O for object. P, G and S are IRI IDs. Virtuoso offers SPARQL inside SQL.Thus SPARQL inherits all the aggregation and grouping functions of SQL, as well as any built-in or user defined functions. SPARQL is converted into SQL at the time of parsing the query. If all triples are in one table, the translation is, union becoming a SQL union and optional becoming a left outer join. The toplevel of the data mapping metadata are quad storages. A quad storage is a named list of RDF views. A SPARQL query will be executed using only quad patterns of views of the specified quad storage.

2.4 D2RQ

D2RQ is a declarative language to describe mappings between application-specific relational database schema and RDF-S/OWL Ontologies[5],[6]. D2RQ allows RDF applications to treat non-RDF relational databases as virtual RDF graphs, which can be queried using RDQL. The central concept in D2RQ is the ClassMap. A ClassMap represents a class or a group of similar classes from the ontology. It specifies whether instances are identified by using URI column values from the database, by using an URI pattern together with the primary key values or by using blank nodes. Each ClassMap has a set of property bridges, which specify how instance properties are created and how given URIs or literals are reversed into database values. There are two types of property bridges: DatatypePropertyBridges for literals and ObjectPropertyBridges for URIs and for referring to instances created by other class maps. Property values can be created directly from database values or by using patterns and translation tables.

3. Architecture of the System

The overall system architecture consists of two parts. OntoExtractor [1] which automatically converts ER/EER model into Ontology and Ont2R Data Mapper that provides the Ontology based access of relational database.The OntoExtractor[1] extracts the Ontology from ER/EER diagram which is input to Ont2R Data Mapper.



Ont2R The Ont2R Mapper maps the ontology with the underlying relational database and allows to execute SPARQL queries on the ontology created. The SPARQL query gets converted into SQL queries and gets executed on the relational database and the results are returned as RDF triples. The entire work of the Ont2R can be divided into 3 modules

A) Mapping ontology to relational schema

B) Converting SPARQL to SQL & Execution of SQL

C)Converting Results back into RDF

3.1 Mapping Ontology To Relational Schema

The standard rules for conversion of ER model to Relational model is based on [7]. The conversion of SPARQL to SQL is mainly based on how OntoExtractor[1] converts ER diagrams to Ontology. Some of the OntoExtractor[1] mappings that are relevant while converting SPARQL to SQL are given below :

Table 1: OntoExtractor[1] Mapping Rules

ER Component	OWL-DL Component
Strong Entity,Weak Entity	Class
Identifying Relation	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 and multivalued attributes	datatype property
Composite	Class with components attributes as datatype properties. Functional 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.
Binary Relation	
a. Without attributes	Pair of inverse object properties.
b. Binary Relation With attributes	
many-many	Class with name of relation. Datatype property corresponding to attribute is added to the above class. Two pairs of object property between above class and two participating entities.
Superclass and subclass	Superclass and subclass.
Ternary relation	Class with name of ternary relation. Three pairs of inverse object property between the above class and three entities.

Consider the following ER diagram for Company Schema as shown in figure 2

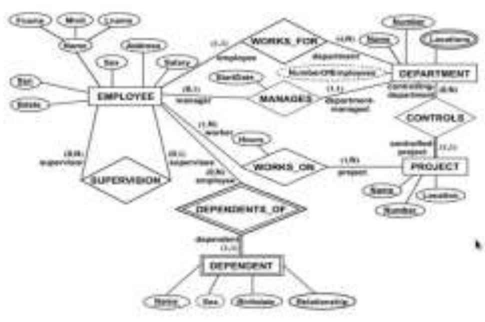
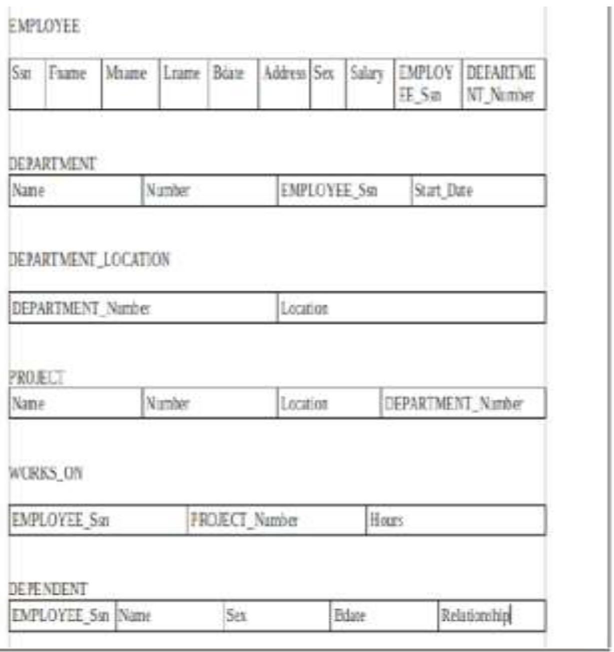


Figure 2 : ER diagram of Company Schema[7]

SPARQL	SQL
SELECT ?subject ?object WHERE{ ?subject rdfs:property ?object } rdfs:property-datatype property(if the domain is an object)	SELECT <attribute > FROM < table > Select from the table with name as the domain of the property select column with object name
subject - variable	Refers to one tuple
Object - variable	Refers to the value of the property
Object - constant	Value of column taking assertions from where
FILTER	WHERE
Property-object property(Range is an entity in ER Diagram)	Do join of the two tables with names as that of object
Property object property(Range not in ERD)(key or composite attribute)	From domain table select the attribute
OPTIONAL	NULL values are covered in the result

The Relational database of this Company schema according to the standard rules for conversion[7] consists of the following tables :



EMPLOYEE

Ssn	Fname	Mname	Lname	Bdate	Address	Sex	Salary	EMPLOYEE_Ssn	DEPARTMENT_Number

DEPARTMENT

Name	Number	EMPLOYEE_Ssn	Start_Date

DEPARTMENT_LOCATION

DEPARTMENT_Number	Location

PROJECT

Name	Number	Location	DEPARTMENT_Number

WORKS_ON

EMPLOYEE_Ssn	PROJECT_Number	Hours

DEPENDENT

EMPLOYEE_Ssn	Name	Sex	Bdate	Relationship

Figure 4: Relational schema generated from the

company
ER diagram
The Ontology created by the OntoExtractor [1] consists of Classes, DatatypeProperties, Object Properties as shown in figure 5.

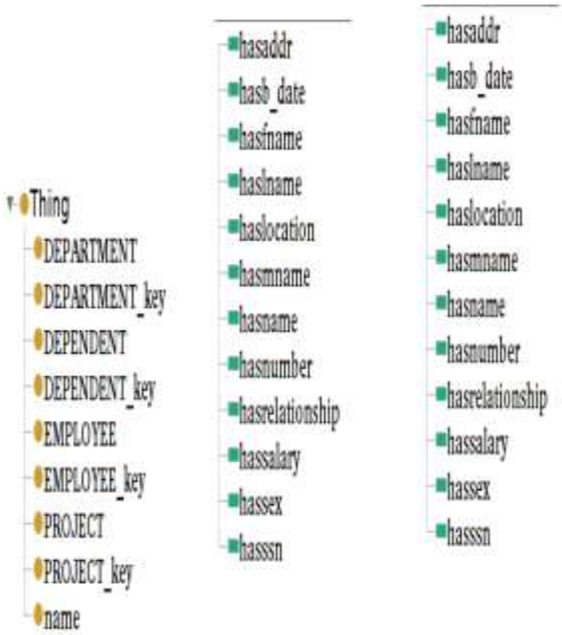


Figure 5a:Classes Figure 5b:DatatypeProperties Figure 5c:Object Properties

3.2 RULES FOR CONVERTING SPARQL TO SQL

The following rules are used for conversion of SPARQL Query to SQL query as shown in given table 2 conversion rules

Table 2: Conversion Rules

Algorithm 1: SPARQL to SQL Algorithm
Input: SPARQL query
Output: SQL Query

1: repeat
2: if i ← containsdatatypeproperty then
3: datatype(i)
4: else if i ← containsobjecttypeproperty then
5: objecttype(i)
6: else if i ← contains filter then
7: filter(i)
8: end if
9: until end of the query

Algorithm 2: datatype(i)

X = property name without ”has”;
if X ← multivaluedattribute **then**
FROM+= (X,Domain)
WHERE+= ”X.key(Domain)=Domain.key”
end if
if Domain ← Entity **then**
if Domain ← subclass **then**
FROM+= (subclass, superclass)
WHERE+=” subclass.key=superclass.key)”
else
FROM+= Domain
end if
if object ← variable **then**
if object is in the select _temp **then**
SELECT+ =X
else
temp _var= X
end if
else
WHERE+= ”Domain.X = object ”
end if
else
if Domain contains ”Key” **then**
Domain = remove Key from Domain
if object ← variable **then**
if object is in the select_temp **then**
SELECT+ =X
else
temp var= X
end if
else
WHERE+= ”Domain.X = object ”
end if
else
object_property_name = has +Domain
if object ← variable **then**
if object is in the select_ temp **then**
SELECT+ = X
FROM+ = domain of object property name
end if
else
WHERE+= ”domain of object property name.X = object”
end if
end if
end if

Algorithm 3 :objecttype(i)

X = property name;
if X ← relationship **then**

if Domain/Range ← weakentity **then**
FROM +=(domain, range)
WHERE +=weakentity.key(owner)=owner .key
else
if Xis1 : NorN : 1 **then**
FROM +=(Nside, 1side)
WHERE += Nside.key (1 side)=1side.key
else if Xis1 : 1 **then**
FROM+=(Entitywithtotalparticipation(E1), otherEntity(E2)
WHERE +=E1.key(E2)=E2.key
else
FROM +=(Domain, Range, X)
WHERE+=Domain.key1(domain)=X.key1and Range.key2(range)=X.key2
end if
end if
if object is in the select_temp **then**
FROM+ = Domain
SELECT = Domain.key
end if
else if X contains substring is_ Identified_ by_ **then**
if object is in the select temp **then**
FROM+ = Domain
SELECT = Domain.key
end if
else if X is an attribute of Domain **then**
if object is in the select_temp **then**
FROM+ = Domain
SELECT = components of Domain
end if
end if
if Range ← relationship **then**
FROM =(Domain, range)
end if

Algorithm 4*: filter(i)*
WHERE+ =”Domain.property”+filter condition+ filtervalue.

An example of procedure of converting a SPARQL query to SQL is given below.

The **SPARQL Query** is:
prefix abc:<http://www.owl ontologies.com/Ontology1328199218.owl#>

SELECT ?bdate
WHERE {?a abc:isDEPENDENTS OF ?z.
?z abc:hasName ?ename.
?ename abc:hasFname ”cs”.
?a abc:hasBdate ?bdate.
}

The procedure of conversion is given in detail for the above query

step1: select temp=bdate
isDEPENDENTS_OF is object property and X=DEPENDENTS_OF .X is relation name and its domain is weak entity.

step2: FROM=DEPENDENT,EMPLOYEE

step3: WHERE=DEPENDENT.Ssn=EMPLOYEE.Ssn

step4: hasName is object property.
X=Name. X is not a relation name and doesn’t contain is_identified_by. Name is composite attribute of domain EMPLOYEE.

step5: hasFname is a datatype property.
X=Fname.Its domain Name is class corresponding to composite attribute. object property= hasName.The domain of this object property is EMPLOYEE.
WHERE={DEPENDENT.Ssn=EMPLOYEE.Ssn and EMPLOYEE .Fname=cs}

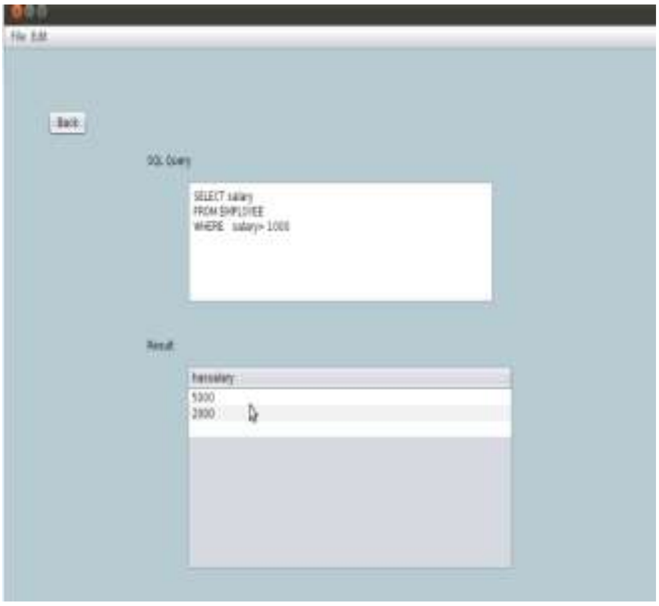
step6: hasBdate is a datatype property. Its domain is EMPLOYEE.
SELECT=Bdate

From all these steps the final **SQL query** obtained is
SELECT DEPENDENT.Bdate
FROM DEPENDENT,EMPLOYEE
WHERE DEPENDENT.Ssn=EMPLOYEE.Ssn and EMPLOYEE .Fname=cs

3.3 Results in triples

Based on the mappings to the relational database, the result of the SQL query is converted to RDF triples and displayed.

4. Experimental Results



The queries are executed on the ontology given below.

Query1 : Bdate of dependents who depend on employee with Fname=cs

SPARQL:
prefix abc:<http://localhost/defaultBase# >
SELECT ?bdate
WHERE {?a abc:isDEPENDENTS OF ?z.
?z abc:hasName ?ename.
?ename abc:hasFname ”cs”.
?a abc:hasBdate ?bdate.
}
SQL: SELECT DEPENDENT.Bdate
FROM DEPENDENT,EMPLOYEE
WHERE DEPENDENT.Ssn=EMPLOYEE.Ssnand EMPLOYEE .Fname=cs
Query2: Get the ssn,salary of all employees who works for department no=21

SPARQL:
prefix abc:<http://localhost/defaultBase# >
SELECT ?ssn ?salary
WHERE { ?x abc:hasSalary ?salary.
?x abc:isWORKS FOR ?y.
?y abc:DEPARTMENTis Identified by key ?s.
?s abc:hasDnumber ?z.
filter(?z=21).
?x abc:EMPLOYEEis Identified by key ?t.
?t abc:hasSsn ?ssn }

SQL: SELECT Salary,Ssn
FROM EMPLOYEE ,DEPARTMENT
WHERE
EMPLOYEE.Dnumber=DEPARTMENT.Dnumber
and DEPARTMENT.Dnumber=21
Query3: Select department locations for all departments which controls project number =30
SPARQL:
prefix abc:<http://localhost/defaultBase# >
SELECT ?dlocation
WHERE { ?x abc:isCONTROLS ?y.
?y abc:hasDlocation ?dloc
?y abc: PROJECTis identified by key ?s
?s abc:hasPnumber ?z
filter(?z=30) }
SQL: SELECT Location
FROM DLOCATION,DEPARTMENT,PROJECT
WHERE
DEPARTMENT.Number=DLOCATION.Dnumber
AND
PROJECT.Number=DEPARTMENT.PNUMBER
AND PROJECT.Number=30
Query4: Select the salary of employees whose salary is greater than 1000

SPARQL:
prefix abc:<http://localhost/defaultBase# >
SELECT ?salary
WHERE { ?x abc:hassalary ?y.
filter(?y> 1000)
}
SQL: SELECT Salary FROM EMPLOYEE
WHERE salary>1000

5. Conclusion

The proposed method converts almost all basic SPARQL queries into SQL. The data from Relational database is accessed without any loss. Currently it maps only basic queries. As a future work it can be extended to work for complex queries such as nested queries and superclass, subclasses.

References

[1] Barrasa Rodríguez, Jesús and Corcho, Oscar and Gómez-Pérez, A. (2004). *R2O, an extensible and semantically based database-to-ontology mapping language*. In: "Proceedings of the Second Workshop on Semantic Web and Databases, SWDB 2004". Springer-Verlag, Berlín, Alemania, pp. 1069-1070. ISBN 978-3-540-24576-6.

[2] PÃ©rez de Laborda, Cristian & Conrad, Stefan. (2005). Relational.OWL - A data and schema representation format based on OWL. Conferences in Research and Practice in Information Technology Series. 43. 89-96.

[3] [Erling, Orri & Mikhailov, Ivan. (2007). RDF support in the virtuoso DBMS. Studies in Computational Intelligence. 221. 59-68. 10.1007/978-3-642-02184-8_2.

[4] Andy Seaborne Christian Bizer. D2rq treating non-rdf databases as virtual rdf graphs. Poster at ISWC2004.

[5] Hert, Matthias & Reif, Gerald & C. Gall, Harald. (2011). A comparison of RDB-to-RDF mapping languages. 25-32. 10.1145/2063518.2063522.

[6] Shamkant B Navathe Ramez Elmasri. Relational Database Design by ER- Mapping, chapter 7. Pearson, 2007.