*International Journal of Computer Applications (0975 – 8887)*
*Volume 65– No.3, March 2013*

# Direct Migration Method of RDB to Ontology while Keeping Semantics

Jamal Bakkas
University Hassan I, FSTS
Department of Mathematics and computer science

Mohamed Bahaj
University Hassan I, FST Settat
Department of Mathematics and computer science

Abderrahim Marzouk
University Hassan I, FST Settat
Department of Mathematics and computer science

## ABSTRACT

The lack of semantic richness is one of the biggest drawbacks of the data stored in the classic relational databases (RDB). This paper provides a method that gives a meaning to these data to serve the semantic web. This method allows a direct and automatic conversion of RDB to ontology, it operates on two levels: The first one is based on the principle of reverse engineering; its purpose is to extract the RDB schema and convert it directly to an ontology model (TBOX). The second level aims to populate the ontology by individuals (ABOX) using data of different records of the RDB and basing on the model of the ontology. Our approach takes into account the relationships established via foreign keys between tables, and the semantic of integrity constraints during the conversion, which allows keeping the consistency and integrity of data.

## General Terms

RDB, Ontology

## Keywords

RDB, Ontology, Mapping, Semantic web.

## 1. INTRODUCTION

The mapping between ontologies and RDBs in both directions has a major importance. Thus, a mapping of ontology to RDB can exploit the power and performance of DBMSs regarding the storage and access speed mainly for large ontologies. In this area there are several researches like that conducted by Fankam, Pierra & al who introduced the notion of Ontology-Based DataBases [1],[2].The mapping in the other direction allows conversion of existing data in RBDS, which are semantically poor to ontology for supplying the semantic web, thus Pérez & al have proposed the introduction of the notion of "Relational.OWL" which consists to represent a RDB into RDF/OWL, and then use the SPARQL query language to make queries on this representation in order to create the equivalent ontology[3]. Bizer and Seaborne have introduced D2R MAP and D2RQ which are declarative languages to describe the mapping between RDB and the semantic web [4], [5]. Other solutions propose conversion tools for mapping such as RDB2Onto [6] and DataMaster [7] and others.

This paper presents a method for automatic and direct conversion of a RDB to an ontology maintaining the consistency and integrity of data. This conversion is done in two steps one is interested in schemas and the other focuses on data. The objective of the first step is to extract the schema of the RDB and convert it into ontology model (TBOX), while distinguishing tables that represent entities of those that represent associations. And the second step focuses on the conversion of records from different tables of RDB to instances (ABOX) of the elements of TBOX. The conversion process takes into account the conservation of the RDB semantic values presented by integrity constraints, primary keys, and foreign keys.

What remains of this paper is organized as follows: Section 2 provides a general description of our method and its two levels. In Section 3, we describe the conversion of schema level and present the conversion algorithm. Section 4 discusses the conversion at the data level and provides the algorithm used at this level. In Section 5, we show - in detail - a case study with the presentation of our validation tool that implements both algorithms. A conclusion and perspectives of this work are the subject matter of section 6.

## 2. DESCRIPTION OF THE METHOD

Our method operates on two levels: schema level and data level. The first level focuses on the extraction of the RDB schema, rummaging through catalogs of DBMS to create the model of ontology which is composed of a set of classes with data type properties and linked to each other (classes) by object properties. This model constitutes the TBOX part of ontology. The second level aims to extract data (records) of the RDB and use them for assertions of the various elements of the model obtained in the first level. The set of these assertions constitutes the ABOX part of ontology.

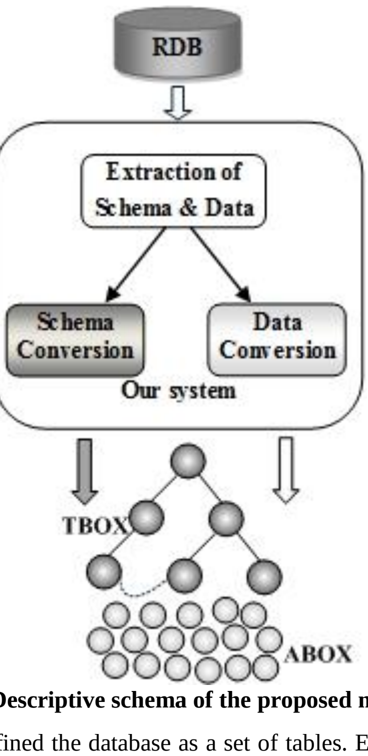


**Fig 1: Descriptive schema of the proposed method**

We have defined the database as a set of tables. Each table is characterized by its name, the list of its attributes and a list of its records. Each attribute is characterized by: its name, its type, a Boolean to test whether the field is a primary key, another Boolean to test whether the converted field is a foreign key, and the table to which it refers if it is a foreign key. A record is a set of pairs (attribute, value), which associates for each attribute a corresponding value. We can therefore define an RDB as follows:

**DB=***{Table/*
*Table=(tableName , attributeList, recordList/*
*attributeList={(attributeName,attributType,is_primaryKey,is_forenKey,tableSource)}*
*recordList={ (attribute,value)/*
*attribute ∈ attributeList*
*}*
*}*
With
***tableSource***: The name of the table referenced by the attribute if it is a foreign key, otherwise *tableSource* has no value.

**Fig 2: Database description**

## 3. CONVERSION AT THE SCHEMA LEVEL

Before starting the conversion process at this level, we have defined the model of ontology as a set of classes, a set of properties of data type, and a set of object properties, each class is characterized by name and the name of its super class. Each of the data type properties and object properties is characterized by its name, its domain, and range as indicated below:

*TBOX : { C,{dataTP},{ object} /*
*C= (className,classParent ),*
*dataTP= (dataTPDomain, dataTPName, dataTPRange) ,*
*objectP=(objectPDomain, objectPName, objectPRange)*
*}*

**Fig 3: description of ontology model**

### 3.1 Conversion process

Before starting the conversion, we have created two classes "ENTITY" and "ASSOCIATION"; these two classes are used to classify the RDB tables: the tables which represent entities are converted to classes that inherit from class "ENTITY" and the tables which represent associations (n,n) are converted to classes that inherit from "ASSOCIATION". Let's recall that an association (n,n) is represented at the physical level by a table with the primary key which is made of the foreign keys linking this table with the tables related by this association.

### 3.2 Conversion of fields

Each field is converted to a data type property whose name is the field name, its domain is the class that represents its table and its range is the type of field. And to ensure the atomicity of attributes, we declare these properties as functional properties.

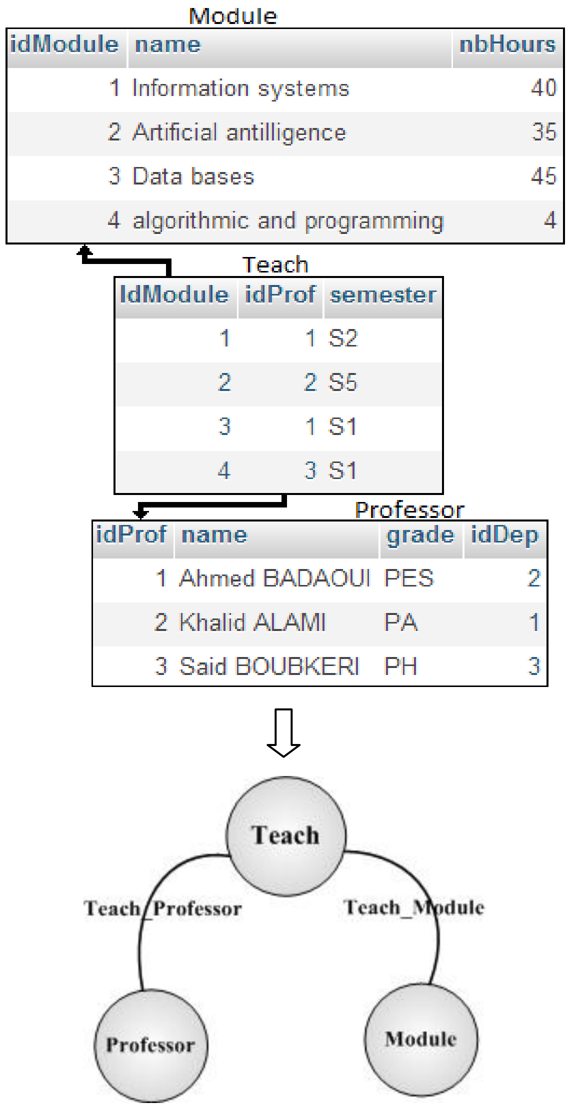
If a field is a primary key, it requires unique values for records. This implies that the values of the data type property that represent this field must be different. Therefore, these properties must be declared as InverseFunctional properties.

### 3.3 Conversion of foreign keys

The field which is a foreign key generates an object property linking the class that represents the table of field to the class that represents the table referenced by the foreign key. A field of a table can be referenced by multiple foreign keys in different tables, so to avoid creating object properties with the same name; we propose to name these properties by concatenating the names of the two classes linked.

If these foreign keys compose the primary key of the table, this table is converted into class that inherits from the class ASSOCIATION, and each of these foreign keys generates object property as described above.

```
<owl:ObjectProperty rdf:ID="Teach_Professor ">
        <rdfs:domain rdf:resource="#Teach"/>
        <rdfs:range rdf:resource="#Professor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Teach_Module ">
        <rdfs:domain rdf:resource="#Teach"/>
        <rdfs:range rdf:resource="#Module"/>
</owl:ObjectProperty>
```



Module

| idModule | name | nbHours |
|---|---|---|
| 1 | Information systems | 40 |
| 2 | Artificial antilligence | 35 |
| 3 | Data bases | 45 |
| 4 | algorithmic and programming | 4 |

Teach

| IdModule | idProf | semester |
|---|---|---|
| 1 | 1 | S2 |
| 2 | 2 | S5 |
| 3 | 1 | S1 |
| 4 | 3 | S1 |

Professor

| idProf | name | grade | idDep |
|---|---|---|---|
| 1 | Ahmed BADAOUI | PES | 2 |
| 2 | Khalid ALAMI | PA | 1 |
| 3 | Said BOUBKERI | PH | 3 |

**Fig 4: Preservation of the foreign key semantics at schema-level**

### 3.4 The conversion algorithm at the schema level

```
Creat TBOX
Creat Class "ENTITY"
Creat Class "ASSOCIATION"
For each T of DB do
    Create C of TBOX
    C.ClassName= "T.tableName"
    For each A of T.attributList do
        If  not(A.is_ forenKey)  then
            Creat dataTP of SchemaOntology
            dataTP.dataTPName  = "A.attributeName "
            dataTP.dataTPDomain = "T. tableName"
            dataTP.dataTPRange = "getType(A.attributType)"
            dataTP  is FunctionnalProperty
            If   A.is_primaryKey  then
                C.classParent =  "ENTITY "
                dataTP  is InverseFunctionalProperty
            endif
        Else
            Create objectP of SchemaOntologiy
            objectP.objectPName=
              concatenate("T.tableName","_","T. tableSource ")
            objectP.objectPDomain="T.tableName"
            objectP.objectPRange ="A.tableSource"
            objectP is FunctionnalProperty
            If  A.is_primaryKey  then
                C.classParent =  "ASSOCIATION "
            endif
        endif
    EndForEach
EndForEach
```

**Fig 5: Algorithm 1: conversion at the schema level**

## 4. CONVERSION AT THE DATA LEVEL

The result of the previous conversion is the model of the ontology which represents the diagram of the RDB. Thereafter, we make assertions of the elements of this model using the values of records from the RDB for create individuals.

Before starting the conversion at this level, we define a ABOX as a list of individuals, each individual is characterized by a name (individualName), type (individualType) which is the class instantiated by the individual, a list of assertions of data type property (assertionDTPList), and another list of assertions of object properties (assertionOPList ). The list of assertions data type property is a set of triplets (dTPname, dTPtype, dTPvalue), while an assertion of an object property is a pair (oPname, individualTarget), where individualTarget is the individual connected to the individual by the current object property. We define, therefore, ABOX as follows:

```
ABOX  : {  I /
        I = (individualName, individualType,
        assertionDTPList,assertionOPList /
        assertionDTPList ={ (dTPName,dTPtype,dTPvalue)}
        assertionOPList={(oPName, individualTarget)}
}
```

**Fig 6: description of ABOX**

### 4.1 Conversion of records

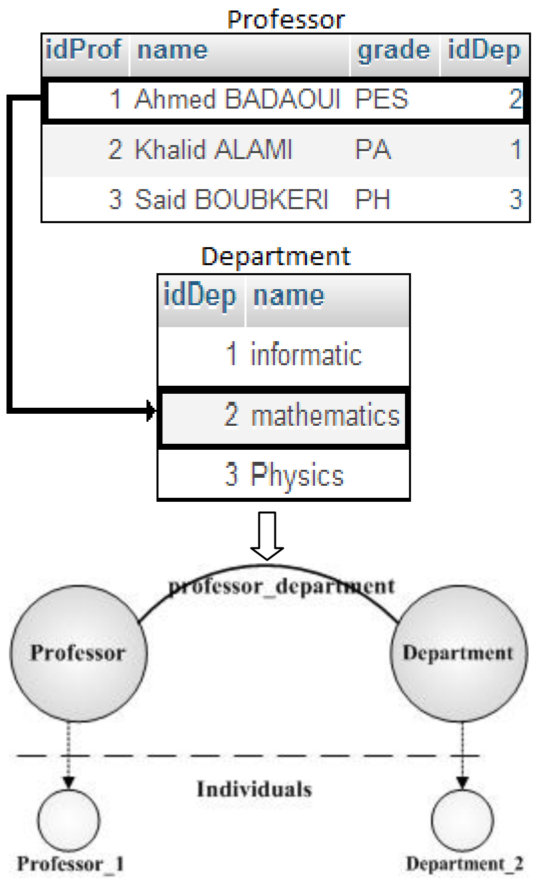
Each record of RDB is converted to an individual of ontology. This individual is the assertion of the class that represents the table of record to convert. So we generate as many instances of a class as records of the corresponding table. And to guarantee the uniqueness of these individuals, we propose to give for each of them a name obtained by concatenating the name of the table and the primary key value corresponding to the converted record as follows:

*<owl:NamedIndividual*
*rdf:about="tableName_primarykeyValue">*

Once the individual is created, it must be filled by the values of all fields of the record, including primary keys and foreign keys, these values are used for the assertion of the data type properties corresponding to these fields.

### 4.2 Semantic conservation of integrity constraints

Each record of a table with a foreign key value which connects it to another record in another table is converted into an individual containing an assertion of the object property linking the classes corresponding to the two tables [paragraph3]. This assertion connects the individual of record to convert, to the individual (target) corresponding to the record referenced by the value of the foreign key. The name of the target individual is obtained by concatenating the name of the table referenced and the value of the foreign key, since this value is the same as the value of primary key of target individual.



Professor

| idProf | name | grade | idDep |
|---|---|---|---|
| 1 | Ahmed BADAOUI | PES | 2 |
| 2 | Khalid ALAMI | PA | 1 |
| 3 | Said BOUBKERI | PH | 3 |

Department

| idDep | name |
|---|---|
| 1 | informatic |
| 2 | mathematics |
| 3 | Physics |

**Fig 7: conservation of the semantics of foreign key at data level**

The generated code is as follow:

```
<owl:NamedIndividual rdf:about="Professor_1">
    <rdf:type        rdf:resource="Professor"/>
    <idProf     rdf:datatype="&xsd;integer">1</idProf>
    <name     rdf:datatype="&xsd;string">Ahmed Badaoui
    </name>
    <grade rdf:datatype="&xsd; string ">PES</grade>
    <Professor_Faculty   rdf:resource="Department_2"/>
</owl:NamedIndividual>
```

## 4.3 The conversion algorithm at the data level

```
Creat ABOX
For each T of DB do
   For each R of T.recordList do
         Create I of ABOX
         I.individualName = "concatenate(T.tableName,"_"
                                 ,primaryKeyValue(R))"
         I. individualType ="T.tableName"
         //With the following data type properties Assertions
         For each P of R do
             Creat a of I.assertionDTPList
             a.dTPName=P.attribute.attributeName
             a.dTPType=P.attribute.attributeType
             a.dTPValue=P.value
             If (P.attribute.is_foreingKey) then
                    Creat o of assertionOPList
                   o.oPName= "P.attribute. attributeName"
               o.individualTarget="P.attribute. tableSource"
             End if
         End For each
   End For each
End For each
with:
```

***primaryKeyValue (R)**:* a function that takes as a parameter the record R and returns value corresponding to the primary key.

**Fig 8: Algorithm 2: conversion at the data level**

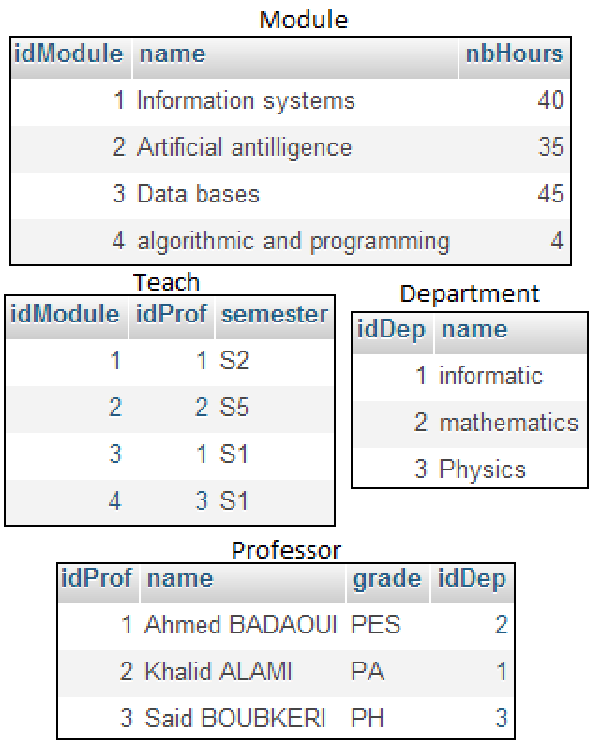
## 5. CASE STUDY AND IMPLEMENTATION OF ALGORITHMS

Consider the database below [Fig. 5] which includes various characteristics and types of relationships between tables namely, primary keys, foreign keys, and primary key composed by foreign keys.

To validate our algorithms, we have developed a tool [Fig. 10] which takes as input a RDB, then extracts its schema and applies Algorithm 1 [Fig. 2] to create the ontology model. Then browse the records to create instances by applying the second Algorithm [Fig. 8]. Both results constitute the resulting ontology which is stored in a file with the name of the RDB with the extension ". owl".

To avoid any ambiguity of interpretation of the different identifiers of our ontology, we defined a namespace which bears the name of the RDB with an URI which is the URL of the ontology. Thus, all of our ontology identifiers are prefixed by the name of the RDB.

To view our ontology and test its consistency, we loaded in Protégé. The figure below [Fig. 11] is obtained using the plugin OntoGraf protégé; it shows the hierarchy of classes and individuals of each class. This figure clearly shows that classes and individuals of resultant ontology have kept the semantic links between tables and between records of the RDB:



Module

| idModule | name | nbHours |
|---|---|---|
| 1 | Information systems | 40 |
| 2 | Artificial antilligence | 35 |
| 3 | Data bases | 45 |
| 4 | algorithmic and programming | 4 |

Teach

| idModule | idProf | semester |
|---|---|---|
| 1 | 1 | S2 |
| 2 | 2 | S5 |
| 3 | 1 | S1 |
| 4 | 3 | S1 |

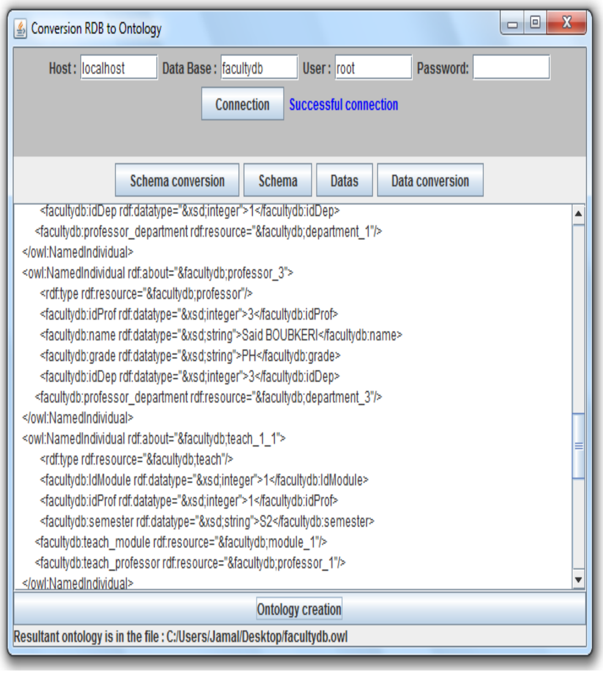
Department

| idDep | name |
|---|---|
| 1 | informatic |
| 2 | mathematics |
| 3 | Physics |

Professor

| idProf | name | grade | idDep |
|---|---|---|---|
| 1 | Ahmed BADAOUI | PES | 2 |
| 2 | Khalid ALAMI | PA | 1 |
| 3 | Said BOUBKERI | PH | 3 |

**Fig 9: Example of a RDB with different types of relationships between tables**



```
   <facultydb:idDep rdf:datatype="&xsd;integer">1</facultydb:idDep>
  <facultydb:professor_department rdf:resource="&facultydb;department_1"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&facultydb;professor_3">
   <rdf:type rdf:resource="&facultydb;professor"/>
   <facultydb:idProf rdf:datatype="&xsd;integer">3</facultydb:idProf>
   <facultydb:name rdf:datatype="&xsd;string">Said BOUBKERI</facultydb:name>
   <facultydb:grade rdf:datatype="&xsd;string">PH</facultydb:grade>
   <facultydb:idDep rdf:datatype="&xsd;integer">3</facultydb:idDep>
  <facultydb:professor_department rdf:resource="&facultydb;department_3"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&facultydb;teach_1_1">
   <rdf:type rdf:resource="&facultydb;teach"/>
   <facultydb:IdModule rdf:datatype="&xsd;integer">1</facultydb:IdModule>
   <facultydb:idProf rdf:datatype="&xsd;integer">1</facultydb:idProf>
   <facultydb:semester rdf:datatype="&xsd;string">S2</facultydb:semester>
  <facultydb:teach_module rdf:resource="&facultydb;module_1"/>
  <facultydb:teach_professor rdf:resource="&facultydb;professor_1"/>
</owl:NamedIndividual>
```

Resultant ontology is in the file : C:/Users/Jamal/Desktop/facultydb.owl

**Fig 10: Our tool for automatic creation of ontology from a RDB**

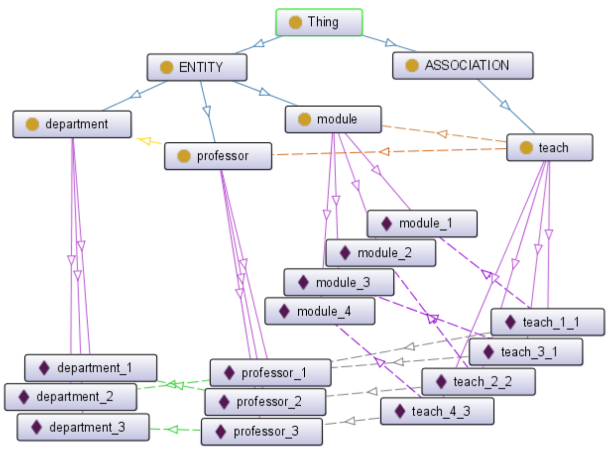


**Fig 11: OntoGraf schematic of the resulting ontology**

## 6. CONCLUSION AND PERSPECTIVES:

In this paper, we show that our conversion method of RDB to ontology keeps certain semantic characteristics of the RDB, while distinguishing the schema of the data. Our future searches will focus, first, on the interrogation of the resultant ontology from this conversion by analogy to interrogation of the RDB by requests formulated by the SQL language, and secondly, the inverse conversion of an ontology to a RDB, which will allow us to exploit the strengths of a DBMS namely the access speed, and mass storage, to store large ontologies that are beginning to appear on the net, and the size of which increasingly disquieting.

## 7. REFERENCES

[1] Chimene Fankam, Stephane Jean, and Guy Pierra: "*Numeric reasoning in the Semantic Web*" In ESWC First International Workshop on Semantic Metadata Management and Applications (SeMMA), volume 346, pages 84–103. CEURWorkshop Proceedings. 2008.

[2] Hondjack Dehainsala, Guy Pierra, and Ladjel Bellatreche: "*OntoDB: An Ontology-Based Database for Data Intensive Applications*". In Proceedings of the 12th international conference on Database systems for advanced applications, Bangkok, Thailand, April 09-12. 2007

[3] Cristian Pérez de Laborda and Stefan Conrad : "*Database to Semantic Web Mapping using RDF Query Languages* ". 25th International Conference on Conceptual Modelling, Tucson Arizona. November 2006.

[4] Christian Bizer, Andy Seaborne : "*D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs* ". In Proceedings of the 3rd International Semantic Web Conference (ISWC2004). 2004.

[5] Christian Bizer : "*D2R MAP - A Database to RDF Mapping Language* ". In WWW2003, The Twelfth International World Wide Web Conference, Budapest, HUNGARY. Poster presentation. 2003

[6] Martin Šeleng, Michal Laclavík, Zoltán Balogh, Ladislav Hluchý "*RDB2Onto: Approach for creating semantic metadata from relational database data*". In: Informatics 2007, the ninth international conference on informatics. 2007

[7] Csongor Nyulas, Martin O'Connor and Samson Tu " *DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé* ". In: 10th international Protégé conference. 2007.

[8] Chimene Fankam : "*OntoDB2 : Support of Multiple Ontology Models within Ontology Based Database* ". In ACM International Conference Proceeding Series. Volume 326, pp.21-27. 2008