Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100307

Integration Mapping Rules: Transforming Relational Database to Semantic Web Ontology

Mohamed A. G. Hazber[1], *Ruixuan Li*[1], *Xiwu Gu*[1,*] *and Guandong Xu*[2]

[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[2] Advanced Analytics Institute, Faculty of Engineering & IT, University of Technology Sydney, Australia

Received: 7 Oct. 2015, Revised: 7 Jan. 2016, Accepted: 8 Jan. 2016
Published online: 1 May 2016

Abstract: Semantic integration became an attractive area of research in several disciplines, such as information integration, databases and ontologies. Huge amount of data is still stored in relational databases (RDBs) that can be used to build ontology, and the database cannot be used directly by the semantic web. Therefore, one of the main challenges of the semantic web is mapping relational databases to ontologies (RDF(S)-OWL). Moreover, the use of manual work in the mapping of web contents to ontologies is impractical because it contains billions of pages and the most of these contents are generated from relational databases. Hence, we propose a new approach, which enables semantic web applications to access relational databases and their contents by semantic methods. Domain ontologies can be used to formulate relational database schema and data in order to simplify the mapping (transformation) of the underlying data sources. Our method consists of two main phases: building ontology from an RDB schema and the generation of ontology instances from an RDB data automatically. In the first phase, we studied different cases of RDB schema to be mapped into ontology represented in RDF(S)-OWL, while in the second phase, the mapping rules are used to transform RDB data to ontological instances represented in RDF triples. Our approach is demonstrated with examples, validated by ontology validator and implemented using Apache Jena in Java Language and MYSQL. This approach is effective for building ontology and important for mining semantic information from huge web resources.

Keywords: Relational database, Semantic web ontology, Resource description framework (Schema) (RDF(S)), Web ontology language (OWL), Mapping rule

1 Introduction

The semantic web is one of the most important research fields that came into light recently. It is one of the ways that make the processing of web information by computers possible and to transform the web into a medium through which data can be shared, understood and processed by automated tools [1,2]. With the development of semantic web, more and more ontologies are developed for various purposes. Ontology is a key enabling technology for the semantic web. It plays a crucial role in solving the problem of semantic heterogeneity of heterogeneous data sources [3] and contributes to improve system interoperation [4]. The World Wide Web Consortium (W3C) has recommended several formats for representing web ontology, such as resource description framework (RDF) [5,6] data model, which is a standard model for data interchange on the web, RDF Schema [7], provides a data-modelling vocabulary for RDF data, and web ontology language (OWL) [8] as a formal language for authoring ontologies. All of these formats intended to provide a formal description of concepts, terms, and relationships and to enable automatic reasoning (inference) within a given domain.

The continuous explosion of RDF data opens door for new innovations in big data and semantic web initiatives, which can be shared and reused through application, enterprise, and community boundaries. Ontology data can be presented in the form of triples of data model (Subject, Predicate, Object), graph of the data model, or RDF/XML which stores RDF format in the form of XML file [5,6]. The most advantage of RDF/XML is that it can reuse the existing XML tools. Moreover, each RDF format has an internet content type [5,6], passed by the server. So, the client knows how to parse the data. In this paper, all the

* Corresponding author e-mail: guxiwu@hust.edu.cn

three forms were used to present the ontology results. The bulk of existing web content "deep web" is stored in RDBs [9], which characterized by high quality of storing and querying data but lack the ability to describe the semantics of data. Moreover, the development of the web content into the semantic web requires the inclusion of large quantities of data stored in RDBs. The RDF data generation from RDB has been the focus of research work in diverse domains. One of the challenges in real world applications is how to improve accessing and sharing knowledge residing in databases. For instance, Internet accessible databases contained up to 500 times more data compared to the static web and roughly 70% of websites are backed by relational databases [10]. Databases [11] and Wikipedia [12] are good candidates for populating the semantic web because they contain great amounts of information in a structured form. In order to utilize today's RDB to support web applications and transparently participate in the semantic web, their associated database schemas need to be converted into semantically equivalent ontologies [13]. So, it is imperative for the community to develop fully automated methods for bridging RDB content and the semantic web.

Mapping RDB to RDF is an attractive field of research. Many approaches were explored to make relational data available to semantic web enabled applications. Most of the proposed approaches are simple, equivalent matching, and neglecting the formal definition which may lead to ambiguous when applying several transformation rules [14,15,16,17,18,19]. Through these approaches there are still difficulties for domain expert to understand the meaning between these approaches, such as unclear generation approach, un-unified ontology language and other related problems. Since, the manual ontology construction is a complex, cumbersome, mistakable, time consuming, high cost process, and requires the supports of domain experts in knowledge acquisition, the main goal of our approach is to generate ontology automatically from RDB. This automatic generation allows getting flexible mapping of complex relational structures into ontology. Moreover, our approach suggests direct mapping (transformation) rules for building ontology (RDF(S)/OWL) from RDB (schema and data) and covers all possible concepts of relational model to find their best transformation into the ontology model. The major contributions of this paper are as follows.

– We propose a new approach for direct mapping RDB to semantic web ontology automatically containing:
  1. Twenty-five unambiguous and well defined sequential mapping rules that transform all well-formed schemas and instances into semantically equivalent RDF(S)-OWL ontologies.
  2. Two well-defined functions identifying binary relations and generating identifier ROWID.
– The transformation rules designed in an obvious forms, so that the rules can be extended to reverse ontology to relational tables.
– We design architecture that provides a uniform semantics between ontology mapping and information integration by transforming RDB schemas and instances to semantic web ontologies. Examples and ontology validator are used to describe how to apply and validate our approach then the proposed approach is implemented, evaluated, and compared with existing approaches.

The rest of this paper is organized as follows. Section 2 discusses the related work in two parts extract ontologies from a relational database and mapping it to existing ontology. Section 3 presents the preliminary concepts of a RDB and ontology languages, and definition mapping between RDB and semantic web ontology. Section 4 shows our proposed architecture and discuss our approach, which includes the rules for mapping RDB schema and data into semantic web (S.W) ontology schema and instance. Section 5 presents examples to describe how to apply our approach. Implementation, ontology validation, comparison, and evaluation are discussed in Section 6. Finally, Section 7 concludes this paper with the future work.

2 Related Work

This section offered an overview of some related studies in mapping between a relational database and an RDF or OWL ontologies. Different researches have been established in this area to provide methods and tools that exposed or converted data in RDB as ontological data described in RDF. The W3C RDB2RDF Incubator Group [16,20] has formed a working group to create a standard for exposing relational databases as RDF. Their efforts might solve the issue of external data access to RDBs. According to the nature of the target semantic web ontology, there are two architectural approaches to integration mapping between RDBs and ontologies. The first one used for extracting ontologies from an RDB (Fig. 1a) and the second one for mapping between relational databases and an existing ontology (Fig. 1b).

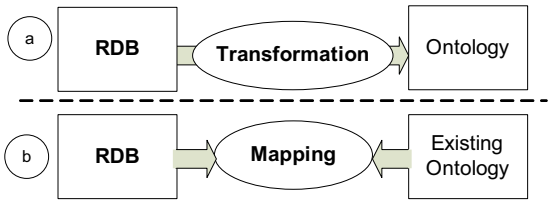


Fig. 1: Transformation Vs. Mapping

2.1 Extract ontologies from a relational database

Most of existing research in semantic extraction focused on how to directly extract ontology from specific schemata. Astrova [21] presented an approach that extracted ontologies from an RDB based on a reverse engineering method using SQL DDL as the RDB model and transformed it to RDFS ontology. This approach was composed of two processes: Analyzed information to extract a conceptual schema and transformed this schema into a semantically equivalent ontology. Finally, the data from a database were migrated into ontologies. They acknowledged that "hidden" semantics can be discovered by analyzing data disjointedness (no intersection) and data overlap (intersection). Another study carried by Buccella et al. [22], who proposed a method (semi-automatic mapping) that integrated several sources of information, based on the use of ontologies. Each data source has a source ontology built in two steps: Generating OWL initial ontology from data models represented in SQL-DDL and building the source ontology which allowed experts to add restrictions, classes and/or properties to the initial ontology. The limitation of their used rules in transformation of Datatype properties have not domain and range defined, the not-NULL constraint was translated to the number restriction, and their translation was not capable of expressing the primary keys. Moreover, in the case of SQL-DDL code did not show the minimal cardinality, to solve this problem experts need to add this cardinality after the ontology was built. Li et al. [23] proposed an approach of learning OWL ontology from data in RDB used a set of learning rules. This procedure had a disadvantage of losing the information because only the schema structure of a relational database had been used therefore, actual data was not utilized. On the other hand Shen et al. [24] were described groups of semantic mapping rules (semi-automaic) for extracting a global OWL ontology from a relational database. The mapped rules for concepts, properties and restrictions represented the correspondence at the metadata level. Stojanovic et al. [25] proposed a method, which was very closed to Astrova [21], they extracted semantics from RDB and represented it in RDFS ontology. This method defined a relational model, retrieved the model from SQL-DDL, and then mapped it to frame logic and RDF Schema. The rules in this method consist of creating classes, subclasses and properties. All these steps were realized in the semi-automatic way because some ambiguous situations can be raised when several rules were applied. This rule conflicts when information is spread across several relations, thus making this effort semi-automatic. In recent investigations Hu et al. [26] suggested a method includes three mapping rules from an RDB schema to ontology (class and property). They used these rules, to build an initial ontology of materials science that can be modified in later. While Zhou et al. [27] described a prototype tool for generating ontology from an RDB schema. The key feature of the tool can directly and automatically translate a RDB schema into ontology without translating data. Finally Zhang and Li [18] presented a method for automatic ontology building using the RDB resources to improve the efficiency, and named the ontology automatic generation system based on a relational database (OGSRD). But they ignored some tables that express association data, which could not be counted in the concepts.

2.2 Mapping a relational database to existing ontologies

In this area there are several approaches. For example Xu et al. [28] presented a practical approach for creating generic mappings between RDB schema and OWL ontology. They provided a D2OMapper tool, which automatically creates the mappings; their approach and tool can act as a gap-bridge between existing database applications and the semantic web. While R2O [29] described mappings between RDBs and ontologies implemented in RDFS or OWL. Mappings described by D2RQ [30] wraps one or more local RDB into a virtual and read-only RDF graph. Recently, Marx et al. [31] introduced an extensible Eclipse plug-in that supported the RDB2RDF conversion process used R2RML. While Hu and Qu [15] have been presented approach to discover simple mappings between RDB schema and ontology. Based on virtual document, initial simple mappings were derived and validated mapping for consistency. Their experimental results in a limited domain showed the feasibility of the approach. Other existing approaches in this area including the studies proposed by [32,33,34,35,36,37,38].

It should be mentioned that our paper focuses on extract ontology from RDB (Section 2.1) and according to the previous studies most of the approaches used in this area suffers from one of the following problems.

- –Simple, equivalent matching and neglecting the formal definition which may lead to ambiguous transformation rules.
- –May not describe the ontology from RDB directly and correctly.
- –The data integration scenario is complex and must be more flexible to make the existing approaches enabled.
- –Transformation structure in some approaches is so limited. E.g. primary key (PK) and foreign key (FK) were assumed to be a single-column. The simple relationship was assumed to be one: one.
- –Some researches ignored the constraints, and some others only assumed simple constraints. They ignoring most referential constraints and table check constraints.
- –Transform the structure and not the data.
- –Semi-automatic and require much user interaction.

–Some ambiguous situations can arise when applying several rules and making this effort semi-automatic.
–Some researches ignored implementation and others assumed complex way for implementation.
–There are still difficulties for domain expert to understand the meaning between these approaches, such as unclear generation approach, unformulated rules, and un-unified ontology language.

This paper proposes new rules for the direct mapping RDB (schema and data) to RDF(S)-OWL semantic web ontology by using a set of particular cases (rules) called mapping rules that is fully automatic. Our rules approach and adopt the methods clearly, easy, cover all concepts of relational models, no interference between transformed concepts, and very close to the software programmers. All types of relationships between tables are considered as (one: one (or zero), (one: many) and (many: many)) and other relationships such as unary relationship are created from referential constraints. Many types of foreign keys, referential constraints, and table check constraints are considered.

3 Preliminaries

In this section some aspects of a RDB (schema and data) as input of transformation and semantic web ontology (RDF triples, RDFS, and OWL) as output were briefly defined. The mapping between RDB and semantic web ontology is also considered.

3.1 Relational databases

A relational database consists of a collection of tables; every table is assigned a unique name. A row in a table represents a relationship between a set of values. The relational database is based on the relational model and uses a collection of tables to represent both data and its relationships. It also includes a DML (Data-Manipulation language) and DDL (Data-Definition language). A relational database schema is a set of tables that have the following concepts:

–Relation (table) T is a two-dimensional table.
–Each table T has a set of columns (attributes) T(Cols)={Col1, Col2,..., Coln.}.
–Both tables and columns are labeled by names and may has commented and caption name (by using SQL alias names).
–Each row in the table is called a tuple (i.e. record) Row= $Col_1 x Col_2 x \cdots x Col_{n-1} x Col_n$, where Col refer to column name which may be either single or composite (individual attributes as components).
–The intersection of a row with the columns will have data values
–Each column has a data type (i.e. string, int, float, date, etc.)
–Each table T has a primary key PK (single or composite).
–Each table T can have a foreign key FK (single or multiple FKs) which refer to another table or the same table (recursive relation).
–One or more tables have relationships (one: one (or zero), one: many, many: many, and recursive relation by using *reference constraints (FK)*.
–The PK in a table $T1$ may be at sometimes is FK which refers to PK in another table $T2$, in this case the table $T1$ is the sub-table of table $T2$ (inheritance (sub-class)).
–The different types of constraints that can be imposed on the table are Not Null, Unique, Primary Key, Foreign Key, Table Check, and On Delete Cascade.
–Duplication of rows is not allowed.

3.2 Semantic web ontology languages

Several ontology languages have been developed during the last few years, and they will surely become ontology languages in the context of the semantic web. Semantic web stack [1] includes the standard of XML, XMLS, RDF, RDFS and OWL, are used to organize, integrate and navigate the web, in addition it allowing content documents to be linked and grouped in a logical and relevant manner. Ontology languages helped to achieve a mapping from Relational Databases to Semantic web ontology and their characters are summarized below:

XML provides the syntax for writing the structural documents, but the meaning of semantic rich data is not clear. While the XML Schema enables the programmer to restrict a structure of XML documents and defines data types, we use it to mapping SQL data types.

Resource Description Framework (RDF): RDF is an XML-based language for describing information contained in a web resource through statements (or triples) and graph model for describing relationships between resources. It consists of three building blocks: **(1) Resources:** denoted by unique identifiers (URIs) for representing real world objects or abstract objects as well as statements that describes a binary relation between these objects. **(2) Properties:** they specify aspects, characteristics or attributes for describing resources. **(3) Triples (or Statements):** which include subject S, predicate P, object O take the form of T (triples) $=< S,P,O >$, all three elements are resources of an RDF model.

Resource Description Framework Schema (RDFS): RDFS extends RDF by offering additional primitives for defining RDF concepts. That is, they can be viewed as a meta-data about RDF elements. Essentially, it defines a number of classes and properties of those classes that have specific semantics [7]. A class is a set of resources, and corresponds to the notions of type or category in other representations. The most important elements of RDF and RDFS are shown in Fig. 2.

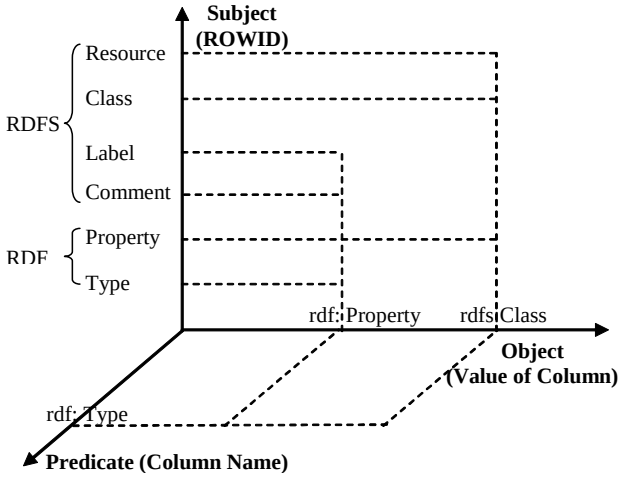


Fig. 2: Elements of RDF/RDFS

Table 1: The important elements of the RDF(S)/OWL

RDF	rdf:(type,datatype,Property,resource,parsType,ID,first,rest, List, Description,value,etc.)
RDFS	rdfs:(Class,subClassOf,subProperty,domain,range,Individual,label,commnt, etc.). The meaning of some elements represented in Figs. 3a-3c.

OWL DL and OWL Full		**OWL lite**
Class Related Constructs	**Property Characteristics**	**Property and restricted Cardinality**
owl:Class	owl:ObjectProperty	owl:cardinality
owl:ComplementOf	owl:DatatypeProperty	owl:minCardinality
owl:DeprecatedClass	owl:FunctionalProperty	owl:maxCardinality
owl:DisjointWith	owl:TransitiveProperty	owl:AllValueFrom
owl:EquivalentClass	owl:SymmetricProperty	owl:onProperty
owl:IntersectionOf	inverseFunctionalProperty	owl:Restriction
owl:one of	owl:DataRange	SomeValueFrom
owl:unionof, complex classes	owl:InverseOf	

OWL Lite Equality and Inequality	equivalentClass, equivalentProperty, sameAs, differentFrom, AllDifferent, DistinctMember, owl:equivalentProperty
OWL DL and OWL Full	owl:hasValue, minCardinality, maxCardinality, cardinality(full cardinality): While in OWL Lite, cardinalities are restricted to at least, at most or exactly 1 or 0, full OWL allows cardinality statements for arbitrary non-negative integers.

Web ontology language (OWL): Ontology is very useful for knowledge representation, which encompasses the following concepts: classes, relationships of classes, property of classes, constraints on relationships between the classes and their properties. OWL is a semantic markup language for publishing and sharing ontologies on the WWW, and it is intended to provide a language that can be used to describe the classes and their relations. When compared with RDF language, OWL has more powerful expressiveness, for example additional vocabulary such as a disjointness relation between classes, transitivity and cardinality of properties, or the creation of complex classes. Depending on the expressiveness, the W3C has split OWL vocabulary into three increasingly expressive sublanguages: OWL Lite, OWL DL and OWL Full. These three sublanguages are based on the standard [13,39], as illustrated in Table 1. Therefore, the semantic web ontology (RDF(S)/OWL Ontology) is a set of classes that have the following concepts:

- –Each class Cls(owl: class) has a set of object properties (*owl:ObjectProperty*), datatype properties (*owl:DatatypeProperty*), functional properties (*owl:FunctionalProperty*), and possible subclasses (*rdfs:subClassOf*).
- –Each object property, datatype property, and functional property has a set of domain (*rdfs:domain*) and range (*rdfs:range*) classes.
- –Each datatype property has a property refers by *rdf:ID*, type of data by XML Schema data type such as rdf:resource=(*“xsd:string”, “xsd:int” , “xsd:float”, “xsd:date”,etc.*).
- –Instances of classes and properties.
- –Datatype describe the properties of elements of classes.
- –Object properties describe the relations between elements of classes.

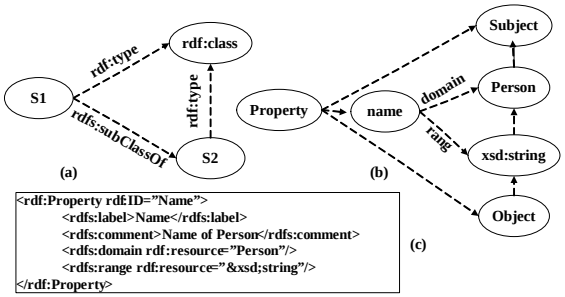


Fig. 3: The oriented graph represents an inheritance class by using rdfs:(subClassOf ,domain and range)

3.3 Definition of mapping between RDB and ontology

RDBs have a set of static structures: tables, columns, data types, relationships, primary keys, foreign keys (references), integrity constraints, and table check constraint as well as a variety of behavioral features (such as triggers, stored procedures, functions, referential actions etc.). Because of the static nature of ontologies, only the static part of relational databases can be mapped to ontologies, whereas dynamic aspects of RDBs (such as triggers) cannot be mapped [25]. When ontology is created from a relational database, the relational data model and the generated ontology are very similar. Therefore, the mapping process from the RDB schema is quite direct, and complex mapping cases do not usually appear. However, the creation of an ontology structure in this study was constructed by two ways:

1.Simple way: includes the direct transformation of each database table into an ontology class, each column into a property, column datatype into an xml schema datatype, and description of tables and column into the comment of ontology. This way is not

enough for expressing the full semantics of the database domain.
2. Complex way: includes the extraction of additional semantic relations between database elements (like the relationships, constraints, referential constraints, and table check constraints) and when we build vocabulary of an ontology it must be related to the concepts extracted from additional semantic relations.

4 Rules for mapping relational database schema and data into ontology

In this section, we introduced our approach to mapping from a relational database (schema and data) to ontologies (RDF(S)-OWL). Our method provides a new mapping rules of direct mapping RDB schema (i.e. tables, columns, relationships, integrity constraints, restriction on property of column, rows, etc.) to RDF(S)-OWL semantic web ontology (i.e. classes, datatype properties with domains and ranges, restriction on classes and properties, object property between elements of classes, inheritance between classes and properties, instances, etc.) automatically by using a set of particular rules (cases) called mapping rules. The contents of this part are represented by running example, which includes all our cases that used in this study as shown in Fig. 5. The mapping rules are divided in two parts: rules for mapping (transform) RDB schema and mapping RDB instances as shown in Fig. 4.

4.1 Proposed approach architecture of mapping rules

The conceptual data model of semantic web ontology is directly connected with the conceptual model and information resource of RDBs. Through analysis mentioned on Section 3 dealing with the concepts of RDB and ontology: A relational database consists of tables, which contains columns and rows (collection of a field's value), relationships between tables and integrity constraints on the columns, whereas an ontology (RDF(S)-OWL) consists of classes, which contains properties and instances (collection of property values), object properties are the relationships between elements of classes and restrictions on properties. The row is nothing but the content of its fields, just as an RDF node is nothing but the connections: the property values. The formal corresponding relationships between tables, columns, rows, relationships, and integrity constraints in RDB and classes, properties, instances, object properties, and restrictions in ontologies make it possible to convert one schema to another (direct and indirect mapping discussed in Section 3.3). The main objective of the proposed approach to extract an ontology from RDB automatically, rapidly and in easy ways, leading to avoid limiting manual work, reducing cost and time, and improving the efficiency for building ontology. The proposed approach architecture used to generate ontology from a relational database is shown in Fig. 4 and includes the following stages:

1. Metadata (schema) and data were extracted from RDB using JDBC driver engine in Java.
2. The schema analysis from the previous stage includes tables, columns with datatype, relationships, integrity constraints, referential constraints, and check constraints.
3. Outputs of stage 2 are used as input for mapping rules to transform RDB model to ontology model. During this stage, Apache Jena package and some functions are used to generate the output of this stage which is an OWL structure build on RDF(S).
4. The RDB data analyzed from stage 1 to rows includes data of the simple tables or related tables as the input of mapping rules of data to generate the triples of data model. During this stage, Apache Jena and sub function (generate identifier of ROWID) were used to generate the output, which are RDF triples.
5. The general ontology was generated by collecting the output of stages 3 and 4.
6. Ontology validator used to verify our generated ontology.

4.2 Rules for mapping a relational database schema to ontology

This section defines the rules that mapping RDB schema to ontology built in OWL on top of RDF(S) vocabulary using XSD datatype. Furthermore, we applied each rule separately of each case by making an example of RDB and shown in Fig. 5. Firstly, we identify the binary relation in order to transform RDB into an RDF triple with OWL vocabulary.

Identifying binary relation: A table T1 is a binary relation between two tables (T2 and T3), if:

1. T1 contains only two columns (attributes) A1 and A2, which are a primary key of T1.
2. T2=$\{B1, \cdots, Bn\}$, where Bi is a primary key, and T3=$\{E1, \cdots, En\}$, where Ei is a primary key, assuming that i $\in \{1..n\}$.
3. A1 in table T1 is a foreign key, refers to column Bi in table T2.
4. A2 in table T1 is a foreign key, refers to column Ei in table T3.

In a mapping rule, negation is represented with the symbol ! , and upper case letters are used to denote variables.

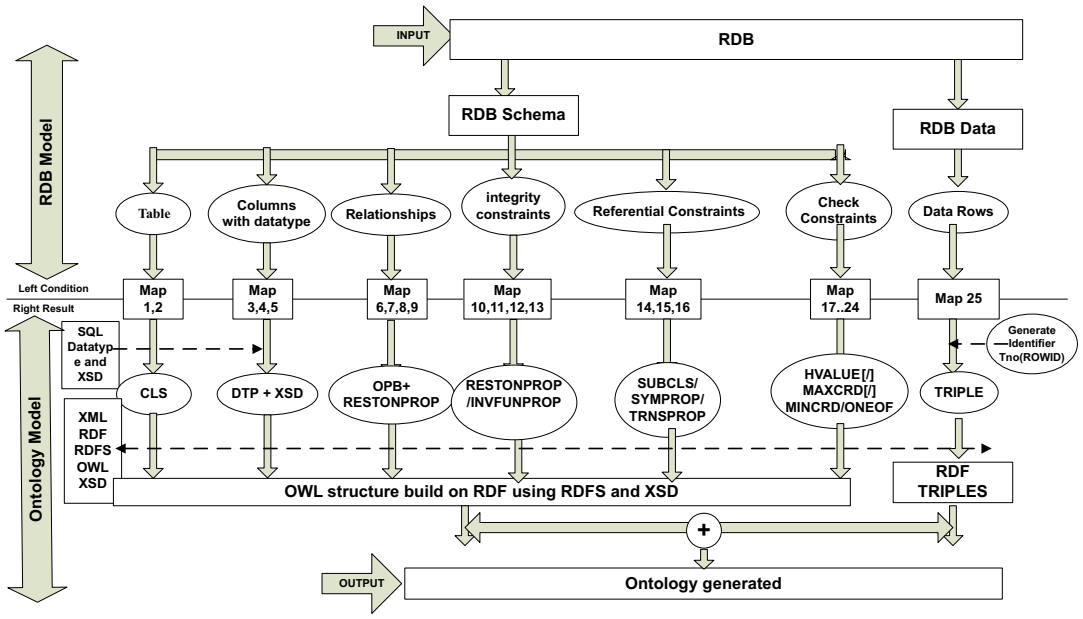


Fig. 4: System architecture for rules of mapping from RDB to Ontology

Definition 1: T1 is a binary relation between two tables(T2,T3):

T1(A1,A2) ∧ T2(B1,..,Bn) ∧ T3(E1,..,En) ∧ LENATTR[T1]==2 ∧ PKs(A1,A2,T1) ∧ [FK(A1,T1) → ATTB(B1,T2) ∧ T1≠T2 ∧ ISONEREF(A1,T1)] ∧ [FK(A2,T1) → ATTB(E1,T3) ∧ T1≠T3 ∧ ISONEREF(A2,T1)] → BinRel(T1,A1,A2,T2,B1,T3,E1) ≫ IsBinRel(T1).

LENATTR[T] is a (length) number of elements in a set T ,T={A1,A2} then LENATTR[T]=2 ISONEREF(A,T1) ← (FK(A,T1) → ATTB(B,T2)) ∧ ! (FK(A,T1) → ATTB(E,T3)) ∧ T2≠T3

In definition (1) the expression of LENATTR[T1]==2 indicates that T1 has exactly two columns. By collecting this expression with PKs (A1, A2, T1), we infer that A1, A2 are the columns of T1. ATTB(B1,T2) the expression indicates that B1 is one of attributes T2. Expression of [FK(A1,T1) ⟶ ATTR(B1,T2)] indicates that A1 is the column of a foreign key in table T1 that points to table T2 through its column B1 and predicate ISONEREF(A1,T1) means A1 is one foreign key refer to T2 and does not allow any other reference table. Then the mapping process is done progressively based on the following rules:

4.2.1 Rules for mapping relations(tables) to classes

Each table **T**n (where *n* is the name of a table) in an RDB unless the **T**n is a binary relation (**!IsBinRel**(**T**n)) and should be mapped to a class **CLS**n (where *n* is the name of the class) in the ontology. The name of **CLS**n corresponding to the name of the table **T**n, comment of table (**T**com) be transformed into the comment of the class(**CLS**com), and also optionally the caption of table as a full name(**T**cap) be transformed into the label of the class(**CLS**lab) respectively. The following rule is used for extracting ontology for class name, comment, and caption that are generated from tables.

Map 1: Each table unless a binary relation is mapped to a class (concept):

T(*n,com,cap*) ∧ !IsBinRel(T*n*)→ CLS(*n,com,lab*)
The com (comment) and cap (caption or full name of table) are optional

For instance, T(student,“All the students of the Master and PhD in the Department of C.S”,“Students”) holds in our example, after applying this rule the ontology appeared as follows:

```
<owl:Class rdf:ID=”Student”>
  <rdfs:comment>All the students of the Master and Dr in the Department of C .S</rdfs:comment> (op)
  <rdfs:label>Students</rdfs:label>  (op)
<owl:Class>
op:optional
```

Map 2: If the table **T**n is a binary relation and has three attributes **A**1,**A**2, and **A**3, where **PKs**(**A**1,**A**2,T) and **A**3 is a simple column. Therefore, in this case the table **T**(n,com,cap) mapped to **CLS**(n,com,lab).

4.2.2 Rules for mapping RDB data types to XSD data types

The datatypes of columns in the RDB are build-in SQL datatypes. Similar to columns, datatype properties in semantic web ontology unlike object properties it has a range classes and make use of RDF datatyping schema, which provides a mechanism for referring to XML Schema datatypes [13]. The RDF and OWL recommendations use the simple types of XML Schema data types [40] in semantic web ontologies. During the mapping of data type properties, the SQL data types are mapped into the matching XML Schema data types.

Table 2 showing a list of common data types that matching with XSD.

Table 2: XSD data type using for mapping from SQL data type

Type	RDB Data Type	Ontology data type
Byte	Bit Varying, Tinyint	xsd:Byte,unsignedByte
Logical	Bit, Boolean	xsd:Boolean
Char	Char,vChar,varChar,nChar, Longtext,Memo,Text, nVarChar,Bolob,TinBlob,Tintext, Mediumtext,MediumBlob, Set(v1,. . . ,vn), Enum(v1,. . . ,vn)	xsd:String, xsd:token, xsd:normalizedString
Numeric, Currency	Integer, Long	xsd:Integer,xsd:positiveInteger, xsd:ngativeInteger, xsd:nonPositiveInteger, xsd:nonNegativeInteger, xsd:unsignedInt,xsd:unsignedLong, xsd:int, xsd:long
	Smallint ,Tinyint, Mediumint	xsd:Short,xsd:unsignedShort, xsd:int
	Float,Real	xsd:Float
	Interval	xsd:Duration
	Numeric, Decimal, Mony	xsd:Decimal
	Double Precison	xsd:Double
Date/Time	Date,Time,TimeStamp, TimeStamp with Time, Time with Time Zone	xsd:Date,xsd:Datetime, xsd:Time
part of datetime	xsd:gYear,xsd:gMonth,xsd:gDay,xsd:gYearMonth, xsd:gMonthDay,xsd:duration	
XML	XML	xsd:anyType
Binary	Binary,Varbinary,Blob,Image, LongBlob,MediumBlob, TinyBlob	xsd:hexBinary,xsd:base64Binary
Link(URL)	Hyperlink to URI	xsd:anyURI

Map 3: Every SQL data type in column table **T**1(COL*Dtype*) where COL*Dtype* could be (text,int,..) is mapped into the corresponding XML Schema data types (Table 2), except if the table **T**1 is a binary relation or column COL is a foreign key in the table **T**1 that reference column in other table **T**2(COL).

T(COL*Dtype*) ∧ ! (IsBinRel(T) ∨ FK(COL*Dtype*,T)) → CLS(DTP*rng of xsd*)
i.e rng(range) of xsd ←"&xsd;string/int/double/….."

For instance, Student(Name Varchar(50)) holds in our example. The Varchar SQL datatype is mapped as follows:

```
... <rdfs:range rdf:resource="&xsd; string"/> ..,
```

4.2.3 Rules for mapping table columns to datatype properties

Every table in an RDB includes columns, which are classified into five groups:

1.Default columns.
2.Composite columns.
3.Enumeration (multi-valued) columns "Using Table Check constraints" (see Section 4.2.7).
4.Primary keys (see Section 4.2.5(map 13))
5.Foreign keys (see Section 4.2.6).

In this paper, we considered each attribute (column **COL**) belonging to groups (3, 4, and 5) as an attribute belonging to the attribute constraints (see Section 4.2.5,6,7). Therefore, each attribute belonging to the above groups (1 and 2) can be mapped into DatatypeProperty(**DTP**) in ontology. The attribute name corresponds to rdfs:label and its description corresponds to rdfs:comment. The domain is the class created from this table and the range of a datatype property is the XSD data type (see Section 4.2.2), which is equivalent to the SQL datatype of the original attribute.

Map 4: Rule for mapping default (simple) columns. Simple columns (**COL**s) in an RDB are columns that contain a data item with a determined data type, unless the column **PK(COL,T)** or **FK(COL,T)**. The column name (**COL**n) is mapped into a Datatypeproperty name (**DTP**n), the column description (**COL**com) corresponds to the rdfs:comment, and the column caption(or name) (**COL**n/cap) corresponds to the rdfs:lable. And the domain is the class(**CLS**dom) created from a table of column (**T[COL]**) and the range of a Datatypeproperty **DTP**rng of *xsd* is the xsd schema data type equivalent to the data type of its original column in the database. The following rule is used for extracting ontology for data type properties that are generated from simple columns:

COL*n,cap,com* →DTP*n*, ,*lab* ,*com* , T*n*[COL] → CLSdom(T*n*) , COL*Dtype*→ DTP*rng of xsd*
Then rule for mapping simple column is
COL(*n,cap,com*,T*n*, *Dtype*) ∧ ! (PK(COL,T) ∨ FK(COL,T)) → DTP(*n,lab,com*, dom(T*n*),*rng of xsd*)
The com (comment) and cap (caption that is created by SQL alias names) are optional

Thus by applying the rule, given that, **COL**(Name, Stud_Name,"Used to store name of student",Student, Varchar(50))∧!(**PK**(name,student)=**false**∨**FK**(name,student) =**false**)=**true**, holds in our example. That is mapped into **DTP**(*Name,Student_Name,Used to store the name of Student,Student,xsd:String*), as shown below:

```
<owl:DatatypeProperty rdf:ID="Name">
    <rdfs:label >Stud_Name</rdfs:label>      (op)
    <rdfs:comment> Used to store the name of Student </rdfs:comment> (op)
    <rdfs:domain rdf:resource ="#Student"/>
    <rdfs:range rdf:resource="&xsd; string"/>
</owl:DatatypeProperty>
```

Map 5: Rule for mapping composite columns. A composite column mainly consists of a set of values from more than one domain. For example, the address column consists of several domain names such as house number, country, phone, email etc. Assuming that we have the following table: Student (*name*, *address*); where the address is a composite column (*phone text*, *email text*). There are two ways to map composite attribute to an OWL datatype property. The first one is to map only their simple component attributes (phone, email) of a composite column (Address) to datatype properties of a corresponding OWL class, and ignores composite column (Address) itself. The second one is to map composite column to datatype property and then map its simple, component columns to sub property of corresponding datatype property.

```
<owl:FunctionalProperty rdf:ID="phone">          First Map
    <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdf:type rdf:resource="#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="email" >
    <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdf:type rdf:resource="#DatatypeProperty"/>
</owl:FunctionalProperty>
```

```
<owl:DatatypeProperty rdf:ID="address">          Second Map
    <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="phone">
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#address"/>
</owl: DatatypeProperty >
<owl:DatatypeProperty rdf:ID="email">
  <rdf:type rdf:resource="#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#address"/>
</owl: DatatypeProperty >
```

4.2.4 Rules for mapping relationship between tables to ontology relationships

Relationships in relational databases are maintained through the use of foreign keys. A foreign key is a data column(s) that appears in one table that may be part of or is coincidental with the key of another table. There are three relationships in a relational database: one: one (or zero), one: many (or many: one), and many: many, as shown in Fig. 5. Moreover, other cases of the relationships were studied in section of rules for mapping referential constraints.

Rules for mapping one: one relationship: In this relationship the maximums multiplicities is one, for example the holds relationship between Student and Position in Fig. 5a. A student holds only one position in a laboratory (Lab) and a position may be held by one student (some positions go unfilled). We can classify this relationship into two rules based on two cases, as shown in Figs. 5a and 5d.

Map 6: One:one (or zero) in which the FK∉PK. If two tables **T1**{**A1,**⋯**,An**} and **T2**{**B1,**⋯**,Bn**} are related to each other through their columns **T1.A1** and **T2.B1**, where an **FK(A1,T1)** that references **PK(B1,T2)**, therefore, the relation is one:one (or zero if **FK**=null). So, the **FK(A1,T1)** is mapped into an owl:objectproperty **OBP**(**A**1,**T1**dom,**T2**rng) that has the source table **T**1 as its domain and destination table **T**2 as its range. In the relationship one:one the **T1.A1=T2.B1**, therefore, this property is restricted to the same value from the class **T**2 **RestOnProp**(**A**1,owl:hasValue,**T**2), and because the constraint **A1≠null** mapped into a mincardinality of 1 **RestOnProp**(**A**1,owl:minCardinality,xsd^^int 1). The mapping **PK(B1,T2)** seen in the section rules of mapping primary key. Fig. 5a illustrates an example of this case. The predicate **OBP** and **RestOnProp** are defined by the following rule:

$$T1(A1,..,An)\bowtie T2(B1,..,Bn) \wedge FK(A1,T1) \wedge PK(B1,T2) \wedge A1 \neq NULL \wedge !(IsBinRel(T1) \vee IsBinRel(T2)) \rightarrow OBP(A1,T1dom,T2rng), RestOnProp(A1,owl:hasValue,T2), RestOnProp(A1,owl:minCardinality,xsd^^int1)$$

After applying this rule in the example in Fig. 5a, the ontology will be extracted as follows:

```
<owl:ObjectProperty rdf:ID="Post_No">
    <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range rdf:resource="#Postion"/>
</owl:ObjectProperty>
<owl:Class rdf:about="#Student">
 <rdfs:subClassOf> <owl:Restriction>
  <owl:onProperty rdf:resource="#Post_No"/>
  <owl:hasValue rdf:resource="#Postion"/>
  <owl:minCardinality rdf:datatype="&xsd;int"1/> [Delete it, if the relationship 1:0]
 </owl:Restriction> </rdfs:subClassOf>
</owl:Class>
```

Map 7: one:one in which FK∈PKs. The primary key of a table can be, at the same time, a foreign key of another table **(COL(A)=FK(A)=PK(A),T)**. Or, the **PKs** of a table consist of foreign key(s) of another table and some other fields (**FK∈PKs**). In our example, in Fig. 5d which holds as Prof_managerlab is a primary key in table Lab, at the same time it is a foreign key (i.e. **FK** a part of **PKs**) that refers to column Prof_no in the table Professor. However, since the **FK** is a part of the **PKs**, it is mapped to an object property **OBP**(**A**1,**T**1dom, **T**2rng) accompanied with restriction on property **A**1 **RestOnProp**(**A**1,owl:hasValue,**T**2), and a cardinality 1 **RestOnProp**(**A**1,owl:Cardinality,xsd^^int 1). The following rule is used for extracting ontology for object properties and restriction, when the foreign key represents a relationship as one: one and form a part from an **PKs**:

$$T1(A1,..,An)\bowtie T2(B1,..,Bn) \wedge PK(FK(A1,T1),T1) \wedge PK(B1,T2) \wedge A1 \neq NULL \wedge !(IsBinRel(T1) \vee IsBinRel(T2)) \rightarrow OBP(A1,T1dom,T2rng), RestOnProp(A1,owl:hasValue,T2), RestOnProp(A1,owl:Cardinality,xsd^^int1)$$

Map 8: Rule for mapping one:many or many:one relationship. This occurs when the maximum of one multiplicity is one and the other is greater than one. If two tables **T1**{**A1,**⋯**,An**} and **T2**{**B1,**⋯**,Bn**} are related to each other through their columns **T1.A1** and **T2.B1** and not similar to (one:one) relationship, where an **FK(A1,T1)** that references **PK(B1,T2)**, therefore, the relationship is one:many(or many:one) is mapped into an **OBP**(**A**1,**T1**dom,**T2**rng). In the one:many relationship all values(**T2.B1**) exist in the column value(**T1.A1**), therefore, this property is restricted to all values from the class **T**2 **RestOnProp**(**A**1,owl:allValueFrom,**T**2). If the constraint **A1≠null** hold, that is mapped into a mincardinality 1 **RestOnProp**(**A**1,owl:minCardinality, xsd^^int 1). Fig. 5b illustrates an example of this case. The following rule is used for extracting ontology for object

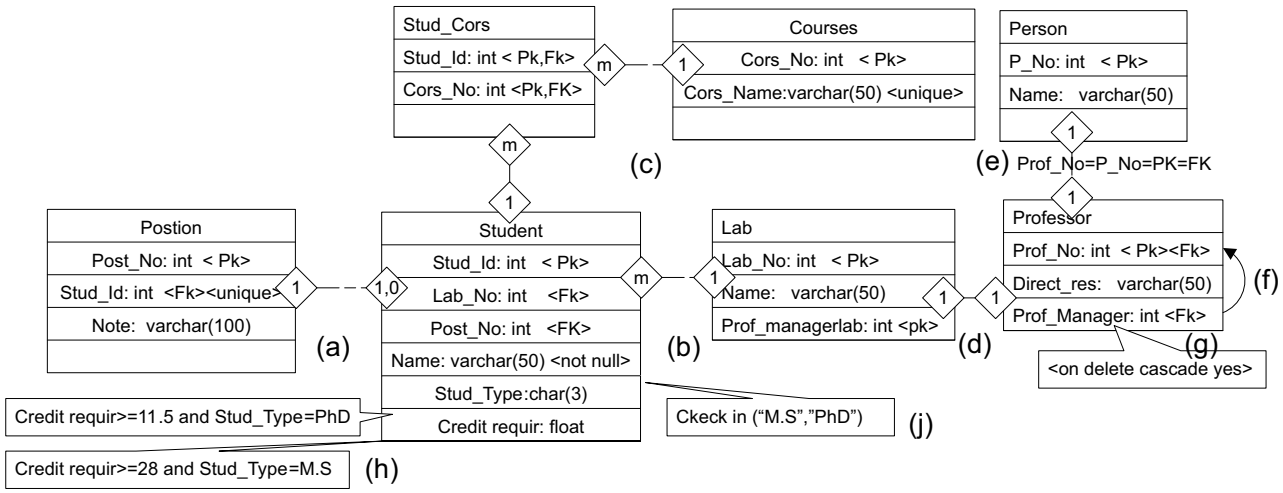


Fig. 5: Relationships and constraints in laboratory of a relational database (RDBLAB)

properties and restriction, when the foreign key represents a relationship as one:many.

$$T1(A1,...,An) \bowtie T2(B1,...,Bn) \wedge FK(A1,T1) \wedge PK(B1,T2) \wedge ValueOf(T1.A1,from,T2.B1) \wedge A1 \neq NULL \wedge !(IsBinRel(T1) \vee IsBinRel(T2)) \rightarrow OBP(A1,T1dom,T2rng), RestOnProp(A1,owl:allValueFrom,T2), RestOnProp(A1,owl:minCardinality,xsd^^int1)$$

An example the study of the relationship between student and Lab holds as Lab_No is a foreign key in the table Student that references column Lab_No in the table Lab. A student study in one Lab and any given lab has one or more students studying there.

```
<owl:ObjectProperty rdf:ID="Lab_No">
      <rdfs:domain rdf:resource="#Student"/>
      <rdfs:range rdf:resource="#Lab"/>
 </owl:ObjectProperty>
<owl:Class rdf:about="#Student">
 <rdfs:subClassOf>  <owl:Restriction>
   <owl:onProperty rdf:resource="#Lab_No"/>\
    <owl:allValueFrom  rdf:resource="#Lab"/>
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger"1/> [Delete it if the A1=null ]
  </owl:Restriction>  </rdfs:subClassOf>
</owl:Class>
```

Map 9: Rule for mapping many:many relationship. In this relationship the maximum of both multiplicities is greater than one, an example the assigned relationship between Student and Course. A student is assigned one or more courses and each course is assigned to one or more students.

If two tables **T2**{**B1,**$\cdots$**,Bn**} and **T3**{**C1,**$\cdots$**,Cn**}, are related to each other through the third table **T1**{**A1,A2**}, where **PK(A1,A2,T1)**, **FK(A1,T1)**$\longrightarrow$**PK(B1,T2)**, and **FK(A2,T1)**$\longrightarrow$**PK(C1,T3)**, then **BinRel**(**T**1,**A**1,**A**2,**T**2, **B**1,**T**3,**C**1) is holds. In such situation, only the tables **T2**{**B1,**$\cdots$**,Bn**} and **T3**{**C1,**$\cdots$**,Cn**} are represented in the ontology as classes with two objectproperty and their restrictions. Therefore, the binary relation is mapped into two an **OBP**1(**B**1,**T2**dom,**T3**rng) and **OBP**2(**C**1,**T3**dom, **T2**rng) according to the rules of one:many relationships. The following rule is used for extracting ontology for object properties and their restrictions that are generated from a binary relation:

$$(T1(A1,A2),T2(B1,...,Bn),\ T3(C1,...,Cn)) \rightarrow BinRel(T1,A1,A2,T2,B1,T3,C1)\ [definition\ 1]$$
$$BinRel(T1,A1,A2,T2,B1,T3,C1) \rightarrow OBP(B1,T2_{dom},T3_{rng}),\ RestOnProp(B1,owl{:}allValueFrom,T3), RestOnProp(B1,owl{:}minCardinality,xsd^^int\ 1).$$
$$OBP(C1,T3_{dom},T2_{rng}),\ RestOnProp(C1,owl{:}allValueFrom,T2), RestOnProp(C1,owl{:}minCardinality,xsd^^int\ 1)$$

According to the definition (1), **BinRel**(Stud_Cors, Stud_Id,Cors_No,Studen,Stud_Id,Course,Cors_No) holds in our example in Fig. 5c. Stud_Cors has two columns(Stud_Id,Cors_No) is the primary key of Stud_Cors, Stud_Id is a foreign key in Stud_Cors that references column Stud_Id in Student, and Cors_No is a foreign key in Stud_Cors that reference column Cors_No in Course. Therefore, it mapped according to the above rule.

4.2.5 Rules for mapping integrity constraints to ontology object property or (and) restriction on the property

When creating an RDB, there are some constraints, which are rules or regulations imposed on data to ensure their correctness. The SQL supports constraints: Not Null, Unique, Primary Keys, Referential constraint, Table Check constraint, etc.

Map 10: Rule for mapping not null constraints. Use the NOT NULL constraints to ensure that a column receives a value during insert or update operations. If the column is Not Null **COL**(**A**isnotnull,**T**) it is mapped to restriction such as a *minCardinality* of constraint of 1 **RestOnProp**(**A**,owl:minCardinality,xsd^^int 1).

$$COL(A_{isnotnull},T_n) \rightarrow T_n \rightarrow CLS_n, RestOnProp(A,owl{:}minCardinality,xsd^^int\ 1)$$

For instance, the **COL**(Name$_{isnotnull}$,student) holds as Name $\neq$ Null, for every row (tuple) in table Student.

```
<owl:Class rdf:about="#Student">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#Name"/>
                <owl:minCardinality rdf:datatype="&xsd;string"1/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Rule for mapping unique constraints: Use the UNIQUE constraint to ensure that a column accepts only unique data values. In this case, we can divide it into two rules (cases).

Map 11: If the column is UNIQUE **COL**(**A**$_{isunique}$,**T**) it is mapped to an inverse functional property **InvFunProp**(**A**,**T**dom,*rng of xsd*) in ontology.

Map 12: If the column is UNIQUE **COL**(**A**$_{isunique}$,**T**) and foreign key **FK**(**A**,**T**1,**B**,**T**2) indicates that **A** is the column of a foreign key in **T**1 that refers to a table **T**2, then it is mapped to object property **OBP**(**A**,**T1**dom,**T2**rng), with inverse function property **InvFunProp**(**A**,type,owl:InverseFunctionalProperty).
The following rules are used for extracting ontology for object properties and inverse functional property from RDB scheme unique constraints.

Map 11: simple column has unique constraint

COL(A$_{isunique}$,T) ∧ ! FK(A,T) → InvFunProp (A,Tdom, *rng of xsd*)

Map 12: column is foreign key and unique constraint

COL(A$_{isunique}$,T1) ∧ FK(A,T1,B,T2) →
OBP(A1,T1dom,T2rng),
InvFunProp (A,type,owl:InverseFunctionalProperty)

For instance, in **Map 11** in our relational schema, the COL(Cors_Name$_{isunique}$,courses) holds as a Cors_Name is unique in table Courses. Also in **Map 12**, the COL(Stud_Id$_{isunique}$,Postion) and **FK**(stud_id,postion,stud_id,student) holds as a Stud_Id is unique and foreign key in table Position.

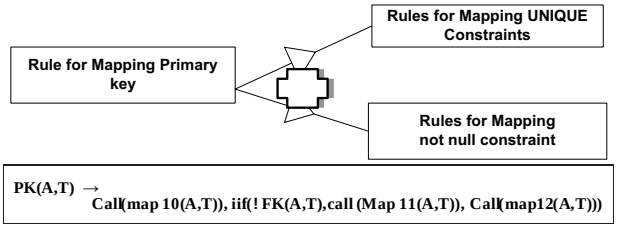
Map 11(Exp):

```
<owl:InverseFunctionalProperty rdf:ID="Cors_Namae">
    <rdfs:domain rdf:resource="#Courses"/>
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</owl:InverseFunctionalProperty>
```

Map 12(Exp):

```
<owl:ObjectProperty rdf:ID="Stud_Id">
   <rdf:type rdf:resource="&owl:InverseFunctionalProperty" />
   <rdfs:domain rdf:resource="#Position" />
   <rdfs:range  rdf:resource="#Student" />
</owl:ObjectProperty>
```

Map 13: Rule for mapping primary key constraints. A primary key is a column (or a set of columns) that contains a unique, and not null value for each row in a table. When we define a primary key constraint, it is mapped by recall two rules (rules of map unique and rule of mapping not null).



4.2.6 Rule for mapping referential constraints

Referential integrity constraint is a property of databases concept of data which needs every value of one column of a table to exist as a value of another column in a different (or the same) table. A foreign key joins and establishes dependencies between tables and creates a referential constraint. Both column constraint (reference) and table constraint are used for specifying foreign keys. For referential constraints, some rules proposed in the previous section (rules for relationships), and the rest of the rules presented in this section. More precisely, the following rules are used to transform the referential constraints such as foreign keys to reference triples for object properties with some restrictions.

Map 14: Rule for mapping referential constraints when FK=PK to ontology such as class inheritance. An inheritance relationship occurs in a relational schema, if the two (or more) tables (**T**1,**T**2,..,**T**n) share the same primary key name. The table **T**2..**T**n that inherits the properties is called the subtables(subClass). The table **T**1 whose properties are inherited is called the supertable(superClass). This case is mapped to a class inheritance, that **T**2,..,**T**n are rdfs:subclass of table **T**1 **SUBCLS**(**T**2,..,**T**n, rdfs:subClassof, **T**1). The following rules are used for extracting ontology inheritance:

SuperTable(T1),SubTables(T1..Tn) →
CLS(T1),CLS(T2..Tn),
SUBCLS(CLS(T2,..,Tn), rdfs:subClassOf,CLS(T1)).

For instance, Fig. 5e holds, where the professor is a person, the Prof_No is a primary key in table Professor at the same time is a foreign key (i.e. FK=PK) that references column P_No (Person no) in the table Person (i.e. Professor.Prof_No=Person.P_No).

```
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Professor">
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```

Map 15: Rule mapping for referential foreign key as unary (recursive) relationship such as one:many is mapped to a logical characteristic of properties "Symmetric Property". A unary (recursive) relationship **RecuRel**(COL_REF,COL,T) defined as a relationship between instances within the same table (i.e. A foreign key **FK**(COL_REF,COL,T) column is added within the same table that references the primary key **PK**(COL,T) values **Value**(COL_REF,COL,T)), this foreign key must have the same domain **Dom**(COL_REF,T) as the primary key and same range **Range**(COL_REF,T). Therefore, the

foreign key is mapped to a symmetric property **SymProp** that uses a class **CLS**(T) as both its domain and range **SymProp**(COL_REF,**T**dom,**T**rng). The following rule is used for extracting an ontology logical characteristic of properties ”Symmetric Property” from a recursive relation.

Definition 2: T is a Unary(Recursive) relationship:
! IsBinRel(T) ∧ PK(COL,T) ∧ FK(COL_REF, COL,T) ∧ Value(COL_REF,COL,T)→ RecuRel (COL_REF,COL,T).

Map 15: RecuRel (COL_REF,COL,T)→CLS(T), SymProp(COL_REF, Tdom, Trng).
SP Symprop : SP(X → y) ⇒ SP(y → x) .

Fig. 5f shows a unary relationship, which holds in our example. Note that the recursive foreign key in the table Professor is named Prof_Manager, this column has the same domain as the primary key Prof_No within the same table. Therefore, after applying the mapping rule to the database of Fig. 5f, the resulting RDF(S)-OWL vocabulary as shown:

```
<owl:class rdf:id=”Professor”/>
<owl:SymmetricProperty rdf:ID=”Prof_Manager”>
        <rdfs:domain rdf:resource=”#Professor”/>
        <rdfs:range rdf:resource=”#Professor”/>
</owl:SymmetricProperty>
```

Map 16: Rule mapping for referential foreign key with cascade delete constraints to logical characteristics of properties (Transitive Property). A foreign key with a cascade delete occurs whenever rows in the master table (referenced) **MstrTable**(COL,**T**mst) are deleted , the relevant rows in the child table(referencing) **ChldTable**(COL_REF,**T**child) with a corresponding foreign key column will get deleted as well. If **MstrTable**(COL,**T**mst) at the same time **ChldTable**(COL_REF, **T**child) means that a foreign key to the same table, indicating a unary relationship again **RecuRel**(COL_REF,COL,T) but in this case the foreign key with trigger cascade delete(on delete cascade). A foreign key **FK**(COL_REF,**T**child,COL,**T**mst) column in a **T**child that references the primary key **PK**(COL,**T**mst) (this foreign key must have the domain **Dom**(COL_REF, **T**child) and range **Range**(COL_REF,**T**mst)). Therefore, the foreign key is mapped to a transitive property **TrnsProp** that uses a class **CLS**(**T**child) as its domain and a class **CLS**(**T**mst) as its range **TrnsProp** (COL_REF,**T**child*dom*,**T**mst*rng*). The following rules are used for extracting an ontology logical characteristic of properties ”Transitive Property” from the referential foreign key with cascade delete constraint.

(1) Map 16.1:RecuRel (COL_REF,COL,Tmst=child) ∧ ONDELETECASCADE(COL_REF,T) → CLS(Tmst), TrnsProp (COL_REF, Tdom, Trng)

(2) Map 16.2:MstrTable(COL,Tmst) ∧ ChldTable(COL_REF,Tchild) ∧ FK(COL_Ref,Tchild,COL,Tmst) ∧ ! (IsBinRel(Tmst) ∨ IsBinRel(Tchild)) → CLS(Tmst), CLS(Tchild), TrnsProp (COL_REF, Tchild*dom*,Tmst*rng*).
TP TrnsProp : TP(X → Z), TP(Y → X) ⇒ TP(y → Z) .

For instance, Fig. 5g holds in our example. Note that when a row of professor is deleted, the relevant rows of the same table with a matching foreign key Prof_Manager column will get deleted as well. Therefore, after applying the mapping rule (map 16.1) to the database of Fig. 5g, the resulting RDF(S)-OWL vocabulary as shown:

```
<owl:class rdf:id=”Professor”/>
  <owl: TransitiveProperty rdf:ID=”Prof_Manager”>
   <rdfs:domain rdf:resource=”#Professor”/>
   <rdfs:range rdf:resource=”#Professor”/>
  </owl:TransitiveProperty>
```

4.2.7 Rules for mapping (table) check constraints

A table check constraints (also known as check constraints) enforce domain integrity by a condition that defines valid data when adding or updating an entry in a table of a relational database. Check constraints are similar to Foreign Key constraints in controlling the values that are put in a column. The (table) check constraints can also be defined using any of the basic constraint logic predicates (=,>,≥,<,≤) as well as BETWEEN, IN and NULL, which are considered in our papers. Furthermore, ”AND” and ”OR” can be used to string the conditions together. A check constraint is applied to each row in the table. The constraint must be a predicate. It can refer to a single or multiple columns of the table. The single and multiple columns check constraint can be represented in OWL ontology using owl:hasValue, owl:maxCardinality, and owl:minCardinality with rdf:datatype property and restrictions on property. The enumerated table check constraint can be represented in OWL ontology using owl:DataRange, owl:oneOf, and rdf:List as a range of rdf:datatype property. Our paper considered several cases of table check constraints, which corresponding to OWL ontology property restrictions are presented in Table 3.

Table 3: RDB table check constraints and ontology such as OWL property

Map No	RDB Table Check constraints		OWL restriction cardinality	Function map (axiom map)
17	Logic Check	Ck(T,Col,V,=,Vck)	Hasvalue	Cls(T),Hvalue(T,Col,V,Vck)
18		Ck(T,Col,V,>,Vckmix)	maxCardinality	Cls(T),MaxCrd(T,Col,V,Vmix)
19		Ck(T,Col,V,≥,Vckmix)	Hasvalue, maxCardinality	Cls(T),Hvalue(T,Col,V,Vck), MaxCrd(T,Col,V,Vckmix)
20		Ck(T,Col,V,<,Vckmin)	minCardinality	Cls(T),MinCrd(T,Col,V,Vmin)
21		Ck(T,Col,V,≤,Vckmin)	Hasvalue, minCardinality	Cls(T),Hvalue(T,Col,V,Vck), MinCrd(T,Col,V,Vckmin)
22	!NULL	Ck(T,Col,V,VckIsnotNull)	minCardinaly =1	MinCrd(T,Col,V,Vckmin=1)
23	Between	Ck(T,Col,V,BT,Vckmin, Vckmix)	minCardinality, maxCardinality	Cls(T),MinCrd(T,Col,V,Vckmin) ,MaxCrd(T,Col,V,Vckmix)
24	In	Ck(T,Col,V,In, {Vck1,..,Vckn})	one of list	OneOf(T,Col,V,list{V1,..Vn})

For instance, table check constraint on a single or multiple columns that where shown in Figs. 5h and 5j holds in our example. Check constraint of table student to ensure that all rows of the students’ table contain only the students who are studying MSc or PhD and they have a site in the laboratory. If he is an MSc student, he should

complete required course credits that are greater than or equal to 28 before they graduate, but if it is a PhD student, he should have total credits of at least 11.5. Therefore, a datatype property is restricted to have the same value for all instances of a class Students according to the check constraints (Credit_requir>=11.5 and Stud_Type=”PhD”), as well as, Ckeck IN(”M.S”,”PhD”) respectively, as shown below.

```
<owl:Class rdf:about="#Student">
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty rdf:resource="#Stud_Type"/>
       <owl:hasValue  rdf:datatype="&xsd;string">PhD</owl:hasValue>
      ...
      <owl:onProperty rdf:resource="#Credit_requir"/>
       <owl:minCardinality rdf:datatype="&xsd;Float">11.5</owl:minCardinality>
     ...
</owl:Class>
<owl:DatatypeProperty rdf:ID="Stud_Type">
  <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range>   <owl:DataRange>
      <owl:oneOf>
        <rdf:List>
          <rdf:first rdf:datatype="&xsd;string">M.S</rdf:first>
          <rdf:rest>  <rdf:List>
            <rdf:first rdf:datatype="&xsd;string">PhD</rdf:first>
              <rdf:rest rdf:resource="&rdf;nil"/>
            ....
      </owl:DataRange>     </rdfs:range>
</owl:DatatypeProperty>
```

4.3 Rules for mapping RDB instances into ontology instances

We now define the rules that map a relational database instance into RDF Triples. After the ontology was extracted, the process of data transformation can start according to the above mapping rules (e.g. classes, properties, restrictions, etc.) from a relational database schema. The goal of this task is the extraction of ontological instances based on the rows of the relational database tables. The values of the table rows can be transformed to the values of the corresponding property of ontological instances.

Map_Inst 1: If a table T is mapped to the class **CLS** then all rows of the table **T**(RW1,..,RWn) are transformed to the instance of a class (as an RDF graph (Triples)) **CLS**(**TRIPLES**1,..,**TRIPLES**n), also, each column in table **T**(COL1,..,COLn) can be transformed to the properties of the instance **CLS**(**DTP**1,..,**DTP**n).

In the bottom part of Fig. 7, we show how the rows of Person (**TNO**1,1,”Shady”) and (**TNO**2,2,”Mohamed”).. (**TNO**n,..,..), where **TNO**1 is identifier to generate the **ROWID**_1 for the first row such as Person_1, can be represented as an RDF graph according to the mapping rules RDB schema and data. In RDF, the subject of a triple must be an identifier, which can be represented as (**TNO**1,..,**TNO**n) in the example. A TNO is identifier of rows (**ROWID**) that generated according to the following rule.

Generate identifier Tno:
ROWID(TNO,T,ROW,COL1,..,COLn,V1, V2, . . . , Vn) ← PKn(T,COL1,COL2, . . . , COLn), VALUE (T,RW,COL1,V1), VALUE (T,RW,COL2,V2), . . . , VALUE (T,RW,COLn,Vn), COLLECT (T,”_”,V1,,V2, . . . ,Vn, TNO).

ROWID(**TNO**,T,ROW,COL1,..,COLn,V1,V2,...,Vn) generates the identifier **TNO** of a row **RW** of a relation **T**. Thus, given that the facts **PK**1(”Person”,”P_No”) and **VALUE**(”Person”,**TNO1**,”P_No”,1) hold in our example, the (**TNO**1=Person_1) is the identifier for the tuple in table Person with value 1 in the primary key. The following rule generates the RDF triples from RDB instance.

T (RW1,…,RWn) → CLS(TRIPLES1,..,TRIPLESn).
RW1(T,TNO1,COL1,V1,COL2,V2,..,COLn,Vn),…,
RWn(T,TNOn,COL1,V1,COL2,V2,..,COLn,Vn) ,V1,..,Vn≠null →

TRIPLES1{TRP1(TNO1, type, T),TRP2(TNO1, COL1,V1| TNOd) ,..,TRPn+1(TNO1,COLn,Vn| TNOd)}
,..,
TRIPLESn{ TRP1(TNOn, type, T)),TRP2(TNOn,COL1,V1| TNOd) ,..,TRPn+1(TNOn,COLn,Vn| TNOd)}

*** IF FK(COL,RW,T,TNOs) References row of table then the object of triple became resource of TNOd (rang) corresponding table of TNOs (domain) (e.g. TRP(Student_1,Lab_No,4) →TRP(Lab_4,Lab_No,4))**
*** TRP1(Tno1, type, T) Indicates that TNO1 is of type class T (i.e. it's part or belongs to the table T and connects all the values of the columns of the row one) and represents in RDF syntax as <T rdf:ID =” TNO1”/>.**

5 Case Study

In this section, we explain our approach by using some examples of mapping RDB Schema and data to ontology RDF(S)-OWL code, and validating code using W3C RDF validator to show the triples of the data model and ontology graph as resulting ontology. Therefore, to understand how to apply our rules on database to generate ontology code and graph, some examples are used as follows:

Example 1: The schema and rows of the table ”Person” represents according to the above rules as follows: Map1 (table (Person) to class(Person)), **Map3,4** (columns of table person into datatype property with xsd corresponding the original SQL datatype of columns), and **Map_ins1**(rows to triples) depending on Table 4, as follows: **RW1**(Person,**TNO**1,P_No,1,Name,”Shady”), **RW2**(Person,**TNO**1,P_No,2,Name,”Mohamed”) it is mapped to the class Person in an OWL ontology, also their table columns can be transformed to the properties of the instance. The **RW1** and **RW2** are rows of Person then the ontology instance for the rows as follows:

RW1→ TRIPLES1{ TRP1(Person_1,type,Person),
TRP2(Person_1,P_No,1),
TRP3(Person_1,Name,”Shady”)}.
RW2→ TRIPLES2{ TRP1(Person_2,type,Person),
TRP2(Person_2,P_No,2),
TRP3(Person_2,Name,”Mohamed”)}

These triples can be represented in Table 5.

Table 4: An example of a Person table

	P_No < *PK* >	Name
Row1→	1	Shady
Row2→	2	Mohamed

Table 5: Triples of the data model

	Subject	Predicate	Object	
Ontology schema	Person	Type	Class	
	P_No	Type	InvesFunctionalProperty	
	P_No	Domain	Person	
	P_No	Range	xsd; int	
	Name	Type	DatatypeProperty	
	Name	Label	Person_Name	
	Name	Comment	Store name of person	
	Name	Domain	Person	
	Name	Range	xsd;string	
Ontology instances	Person_1	type	Person	Row1→Triples 1
	Person_1	P_No	1	
	Person_1	Name	Shady	
	Person_2	type	Person	Row2→Triples 2
	Person_2	P_No	2	
	Person_2	Name	Mohamed	

```
...
 <owl:Ontology rdf:about="dbLab"/>
<!-- Person ontology Schema RDFS-OWL -->
   <owl:Class rdf:ID="Person">
        <rdfs:subClassOf rdf:resource="dbLab"/>
   </owl:Class>
   <owl:InverseFunctionalProperty rdf:ID="P_No">
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range  rdf:resource="&xsd;int"/>
  </owl:InverseFunctionalProperty>
    <owl:Class rdf:about="#Person">
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:onProperty rdf:resource="#P_No"/>
        <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality>
      </owl:Restriction>
     </rdfs:subClassOf>
    </owl:Class>
 <owl:datatypeproperty rdf:ID="Name">
        <rdfs:label> Person Name</rdfs:label>
        <rdfs:comment>Used to store the Name of Person</rdfs:comment>
         <rdfs:domain rdf:resource="#Person"/>
       <rdfs:range  rdf:resource="&xsd;string"/>
   </owl:datatypeproperty>
<!-- Person ontology Instances -RDF TRIPLES -->
    <Person rdf:ID="Person_1">
       <P_No rdf:datatype="&xsd;string">1</P_No>
       <Name rdf:datatype="&xsd;string">Shady</Name>
   </Person>
   <Person rdf:ID="Person_2">
      <P_No rdf:datatype="&xsd;string">2</P_No>
      <Name rdf:datatype="&xsd;string">Mohamd</Name>
    </Person>
...
```

Fig. 6: RDF(S)-OWL ontology extracted from an RDB Schema and instances of the Person table

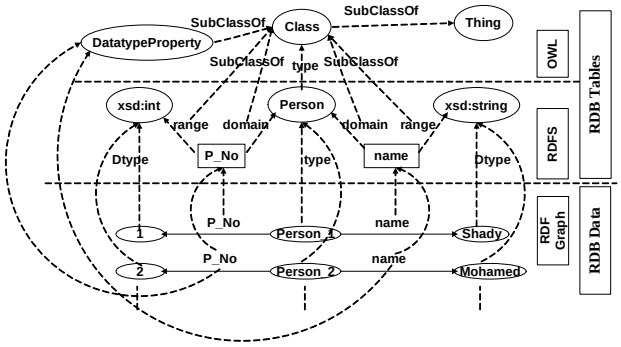


Fig. 7: Graph of ontology extracted from the Person table and its data

However, the results of applying rules return the following RDF Triple and Graph as shown in Figs. 6 and 7 respectively.

Example 2: If one of the tables refers to another table by a foreign key as shown in Figs. 9a-9c, can be mapped according to the above rules a **Map**1 (table to class), **Map**4 (column to data type property), **Map** 3(column data type to data type XML Schema data type), **Map**13 (primary key to inverseFunctionProperty), **Map** 7,8,15 (relationships to objectproperty with restriction on property), and **Map_inst** (rows to triples). Then the values of rows in the table can be transformed to the ontological instances, as shown in Fig. 9d.

From the instances mentioned (Fig. 9) we can know that the values of Lab_No in a Student represented as a property added to the class of Student because table a Student has a column Lab_No that references a table Lab through a column Lab_No. Therefore, the created class Student has one property linking the resources Lab node to represent the values of a column Lab_No in the Student. In the same way a column Prof_Managlab in table Lab that refers to Prof_No in Professor.

Fig. 8 shows the ontology created in general by applying above mapping rules on relational database example shown in Fig. 5. In this figure, oval shapes represent basic concepts in ontology, solid line rectangles represent xml schema datatype, and the other shapes that describe the rest of the vocabulary ontology are represented in the left side of the figure.

6 Implementation and Comparison

6.1 Implementation environments

Based on our system architectures, we propose to implement the method in two phases: the first phase is to transform an RDB Schema to an OWL build on RDF using RDFS and XSD. The Second phase is to transform an RDB Data to RDF Triples (Graph) with a function to generate identifier ROWID for all rows in a table. In order to implement the transformation from an RDB Schema

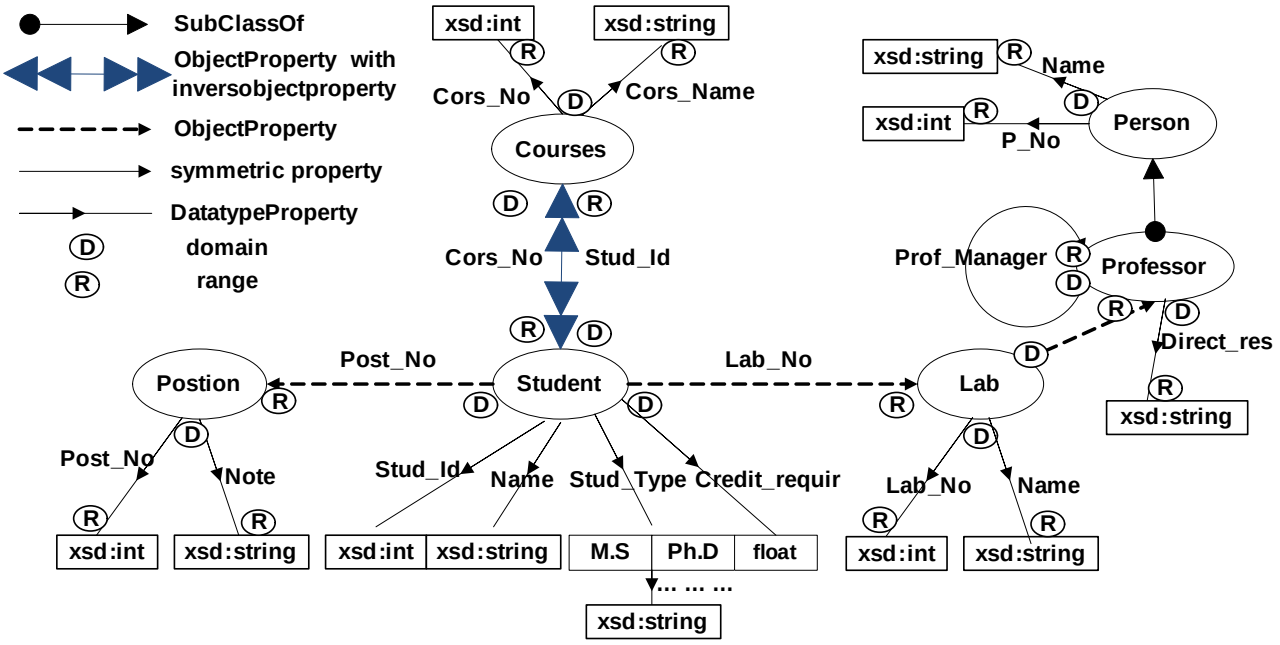
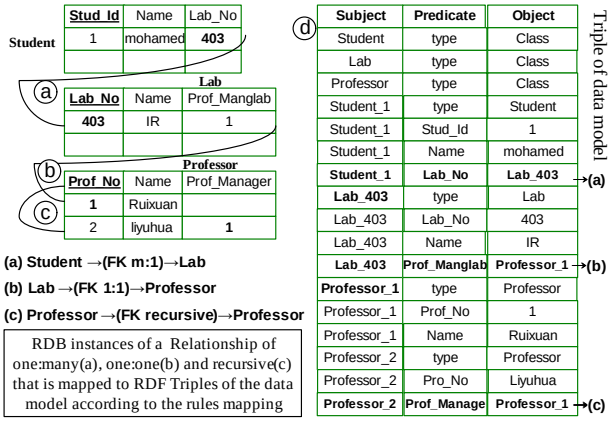


Fig. 8: Ontology generated from RDB LAB



Student

Stud_Id	Name	Lab_No
1	mohamed	403

Lab

Lab_No	Name	Prof_Manglab
403	IR	1

Professor

Prof_No	Name	Prof_Manager
1	Ruixuan	
2	liyuhua	1

(a) Student →(FK m:1)→Lab

(b) Lab →(FK 1:1)→Professor

(c) Professor →(FK recursive)→Professor

RDB instances of a Relationship of one:many(a), one:one(b) and recursive(c) that is mapped to RDF Triples of the data model according to the rules mapping

Triple of data model

Subject	Predicate	Object	
Student	type	Class	
Lab	type	Class	
Professor	type	Class	
Student_1	type	Student	
Student_1	Stud_Id	1	
Student_1	Name	mohamed	
Student_1	**Lab_No**	**Lab_403**	→(a)
Lab_403	type	Lab	
Lab_403	Lab_No	403	
Lab_403	Name	IR	
Lab_403	**Prof_Manglab**	**Professor_1**	→(b)
Professor_1	type	Professor	
Professor_1	Prof_No	1	
Professor_1	Name	Ruixuan	
Professor_2	type	Professor	
Professor_2	Pro_No	Liyuhua	
Professor_2	**Prof_Manage**	**Professor_1**	→(c)

Fig. 9: RDF triples of ontology extracted from the relationship between the Student, Lab, and Professor

The original RDF/XML document

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                 xmlns:owl="http://www.w3.org/2002/07/owl#"
                 xmlns="http://www.exp.db/mo/#"
                 xml:base ="http://www.exp.db/mo/">
<owl:Ontology rdf:about="dbLab"/>
<!-- Person ontology Schema RDFS-OWL -->
    <owl:Class rdf:ID="Person">
        <rdfs:subClassOf rdf:resource="dbLab"/>
    </owl:Class>
    <owl:InverseFunctionalProperty rdf:ID="P_No">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range  rdf:resource="&xsd;int"/>
  </owl:InverseFunctionalProperty>
    <owl:Class rdf:about="#Person">
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#P_No"/>
          <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality>
       </owl:Restriction>
     </rdfs:subClassOf>
    </owl:Class>
  <owl:datatypeproperty rdf:ID="Name">
          <rdfs:label> Person Name</rdfs:label>
          <rdfs:comment>Used to store the Name of Person</rdfs:comment>
           <rdfs:domain rdf:resource="#Person"/>
         <rdfs:range  rdf:resource="&xsd;string"/>
    </owl:datatypeproperty>
<!-- Person ontology Instances -RDF TRIPLES -->
     <Person rdf:ID="Person_1">
         <P_No rdf:datatype="&xsd;string">1</P_No>
         <Name rdf:datatype="&xsd;string">Shady</Name>
    </Person>
    <Person rdf:ID="Person_2">
         <P_No rdf:datatype="&xsd;string">2</P_No>
         <Name rdf:datatype="&xsd;string">Mohamd</Name>
      </Person>
</rdf:RDF>
```

Fig. 10: RDF(S)/OWL ontology extracted from a table Person as ontology code

and data to an ontology(RDF(S)-OWL file), we propose a method using Apache Jena 2.11.0 and using ontology validator to validate, show the triple of data model, and ontology graph. Apache Jena[1] is a Java framework for building semantic web applications. Jena provides a collection of tools and Java libraries to develop semantic web and linked-data apps, tools and servers. The proposed method is implemented using Jena 2.11.0 package (http://jena.hpl.hp.com/) in Java Language using (NetBeans IDE7.3.1 a platform framework for Java desktop applications, and an integrated development environment (IDE)). To examine it, the new method should be applied to an RDB. A sample RDB named, RDBLAB (have several cases applied by our rules), created by MYSQL5.5 and connecting with Java JDBC

[1] http://jena.apache.org

Your RDF document validated successfully. **Validation Results**

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.exp.db/mo/dbLab	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
2	http://www.exp.db/mo/#Person	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
3	http://www.exp.db/mo/#Person	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.exp.db/mo/dbLab
4	http://www.exp.db/mo/#P_No	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#InverseFunctionalProperty
5	http://www.exp.db/mo/#P_No	http://www.w3.org/2000/01/rdf-schema#domain	http://www.exp.db/mo/#Person
6	http://www.exp.db/mo/#P_No	http://www.w3.org/2000/01/rdf-schema#range	http://www.w3.org/2001/XMLSchema#int
7	http://www.exp.db/mo/#Person	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
8	genid:A212391	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Restriction
9	http://www.exp.db/mo/#Person	http://www.w3.org/2000/01/rdf-schema#subClassOf	genid:A212391
10	genid:A212391	http://www.w3.org/2002/07/owl#onProperty	http://www.exp.db/mo/#P_No
11	genid:A212391	http://www.w3.org/2002/07/owl#minCardinality	"1"^^http://www.w3.org/2001/XMLSchema#int
12	http://www.exp.db/mo/#Name	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#datatypeproperty
13	http://www.exp.db/mo/#Name	http://www.w3.org/2000/01/rdf-schema#label	"Person Name"
14	http://www.exp.db/mo/#Name	http://www.w3.org/2000/01/rdf-schema#comment	"Used to store the Name of Person"
15	http://www.exp.db/mo/#Name	http://www.w3.org/2000/01/rdf-schema#domain	http://www.exp.db/mo/#Person
16	http://www.exp.db/mo/#Name	http://www.w3.org/2000/01/rdf-schema#range	http://www.w3.org/2001/XMLSchema#string
17	http://www.exp.db/mo/#Person_1	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.exp.db/mo/#Person
18	http://www.exp.db/mo/#Person_1	http://www.exp.db/mo/#P_No	"1"^^http://www.w3.org/2001/XMLSchema#string
19	http://www.exp.db/mo/#Person_1	http://www.exp.db/mo/#Name	"Shady"^^http://www.w3.org/2001/XMLSchema#string
20	http://www.exp.db/mo/#Person_2	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.exp.db/mo/#Person
21	http://www.exp.db/mo/#Person_2	http://www.exp.db/mo/#P_No	"2"^^http://www.w3.org/2001/XMLSchema#string
22	http://www.exp.db/mo/#Person_2	http://www.exp.db/mo/#Name	"Mohamd"^^http://www.w3.org/2001/XMLSchema#string

Fig. 11: RDF(S)/OWL ontology extracted from a table Person as triples of the data model

by "com.mysql.jdbc.Driver". It consists of 7 tables and their relationships (Fig. 5). In addition to, some test data that proposed in the examples section.

6.2 Ontology validation

To understand the ontology (RDF(S)-OWL) formation, the ontology code production and the ontology graph representation. We used RDF validator[2] to validate RDF(S)-OWL code that generated after applied our rules on RDB, and show the RDF triples and Graph as resulting ontology. According to the schema and data of the Person table (Fig. 5, and Table 4), the results are shown in Figs. 10-12.

6.3 Comparison

Different aspects of approaches, such as model, lauguage support, relationship type, constraint support, complexity, validation, etc. compared with existing approaches. The solution described in this paper contains extra domain knowledge and advantages, based on important factors of mapping rules, strategy used for solution, validation, implementation and complexity degree (Table 6). Moreover, comparison of our approach to existing approaches at the rules level was shown in Table 7.

It must be considered that our approach used for the rules construction is more significant and clear through the transformation process when compared with other approaches.

6.4 Evaluation

We evaluate the proposed transforming strategies by matching a relational database with ontology (RDF(S)/OWL) to determine the true matches, and compare our results with related approaches. To approve the quality of the matching process, we use precision and recall [41] as measuring tools of relevance. Given a reference context alignment R, the precision of some matching alignment A is given by

$$Precision\,(A,R) = \frac{|R \cap A|}{|A|} \tag{1}$$

Recall specifies the share of real correspondences

$$Recall\,(A,R) = \frac{|R \cap A|}{|R|} \tag{2}$$

Where R, is the set of context with correct reference matching and A is the set of all matching retrieved by a particular approach. $|R \cap A|$ is number of correct matching found, $|A|$ is number of matching retrieved by a certain approach, and $|R|$ is number of existing context.

Since precision and recall are most widely used for comparing matching systems and one may prefer to have only a single measure, F-measure[3] is introduced to aggregate the precision and recall.

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

To obtain practical evidence, we applied our transformation to one sample database, particularly,

[2] http://www.w3.org/RDF/Validator/

[3] http://en.wikipedia.org/wiki/Precision_and_recall

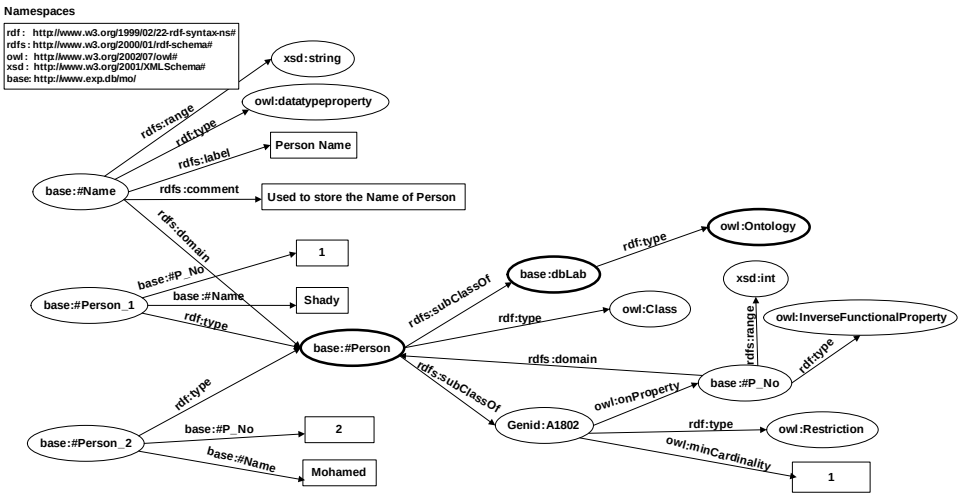


Fig. 12: Graph of Ontology extracted from a table Person and its data instance

Table 6: A comparison between our proposed approach and other existing approaches

Approache	Model	Ontology language	Data Source	Relationship	Table check constraint	Transformation of SQL data types	Data transform	Generate Identifier ROWID	Strategy used for solution	Complex	Validation	Implementation
Stojanovic et al	Semi-auto	RDFS/ F_Logic	SQL-DDL	1:1	Not described	xsd datatypes(without detail)	Yes	No	Examples	Complex	No	No
Astrova et al(2004)	Auto	RDFS/ F_Logic	SQL-DDL	1:1	Not described	xsd datatypes(without detail)	Yes	No	Examples	Normal	No	No
Astrova et al(2007)	Auto	OWL Full	SQL-DDL	M:N	Check equal, Check IN	xsd datatypes(less detail)	Yes(simple)	No	Examples	Normal	No	Yes
Buccella et al	Semi-Auto	OWL	SQL-DDL	M:N	Not described	xsd datatypes(without detail)	No	No	Expository examples	Normal	No	No
G.Shen et al	Semi-auto	OWL	RDB Schema	M:N	Not described	xsd datatypes(less detail)	Yes	No	Examples	Complex	No	No
M.Li et al	Semi-auto	OWL	OLF DB Analyzer	M:N	Not described	Not Described	Yes	No	Group of learning rules with examples	Complex	No	Ontology Learning Framework
Zhang et al (OGSRD)	Semi-auto	OWL	RDB Schema	1:1	Not described	xsd datatypes(without detail)	Yes(simple)	No	OGSRD	Normal	No	Yes
Proposed approach	Auto	RDF(S)-OWL	RDB Schema and Data	1:1,1:0,1:m ,m:m, inheritance, recursive	Check: equal, greater than, greater or equal, less than, less than or equal, Not null, Between, and Check In. "AND" and "OR" can be used to string the conditions together	xsd datatypes(more detail)	Yes	Yes	Functional mapping rules(formal) Examples and prototype framework	Easy	Yes	Yes

RDBLAB (Fig. 5). Then the precision, recall, and F-measure values are used to compare our proposed method and related work, such as Astrova et al. [33], Li. et al. [23] , Zhang and Li [18], and Juan Sequeda [38]. These approaches totally depend on SQL-DDL as a source and can automatically produce ontology. The comparing results of precision, recall and F-Measure are shown in the Fig. 13, indicating that our approach with high measuring relevance. The main reason is that our approach transforms all concepts of relational models and types of relationships between tables. As well it converts characteristics of attributes, many types of foreign keys, referential constraints, and table check constraints. Whereas other techniques transformed some relational concepts and part of them ignored some relationships, attributes properties and constraints on attribute and foreign keys. Except Astrova et al. [33], who was considered only one case of check constraints, all the study approaches ignore mapped check constraints.

Table 7: Comparison of our approach at the rules level with an existing approach

Approach/Schema	Stojanovic et al	Astrova et al (2004)	Astrova et al (2007)	Buccella et al	G. Shen et al	M. Li et al	Zhang et al(OGSRD)	Proposed approach
Defination1	No	No	No	No	No	No	No	Yes
Definition 2	No	No	No	No	No	No	No	Yes
Map 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes(developed)
Map 2	No	No	No	No	No	No	No	Yes
SQL type to XSD	No	No	Yes	No	Yes	No	No	Yes(more datatype)
Map 3	No	No	Yes	No	No	No	No	Yes(developed)
Map 4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes(developed)
Map 5	No	No	No	No	No	No	No	Yes
Map 6	1:1	1:1	No	No	No	No	1:1	1:1 and 1:0 Developed
Map 7	No	No	No	No	No	No	No	Yes
Map 8	No	No	No	No	No	No	No	Yes
Map 9	No	No	Yes	Yes(not clear)	Yes(not clear)	Yes(not clear)	No	Yes (clear and Developed)
Map 10	No	No	Yes	Yes(complex)	Yes(not clear)	Yes	Yes(not clear)	Yes
Map 11	No	No	Yes	No	Yes(not clear)	Yes(different way)	Yes(not clear)	Yes
Map 12	No	No	No	No	No	No	No	Yes
Map 13	No	No	Yes	No	No	No	Yes(not clear)	Yes(developed)
Map 14	No	No	Yes	No	Yes	Yes	No	Yes
Map 15	No	No	Yes	No	No	No	No	Yes
Map 16.1	No	No	Yes	No	No	No	No	Yes
Map 16.2	No	No	No	No	No	No	No	Yes
Map 17	No	No	Yes	No	No	No	No	Yes
Map 18	No	No	No	No	No	No	No	Yes
Map 19	No	No	No	No	No	No	No	Yes
Map 20	No	No	No	No	No	No	No	Yes
Map 21	No	No	No	No	No	No	No	Yes
Map 22	No	No	No	No	No	No	No	Yes
Map 23	No	No	No	No	No	No	No	Yes
Map 24	No	No	Yes	No	No	No	No	Yes
Map_Inst 1	Yes(simple)	No	Yes(simple)	No	Yes(simple)	Yes(simple)	Yes(simple)	Yes(in details + ROWID)

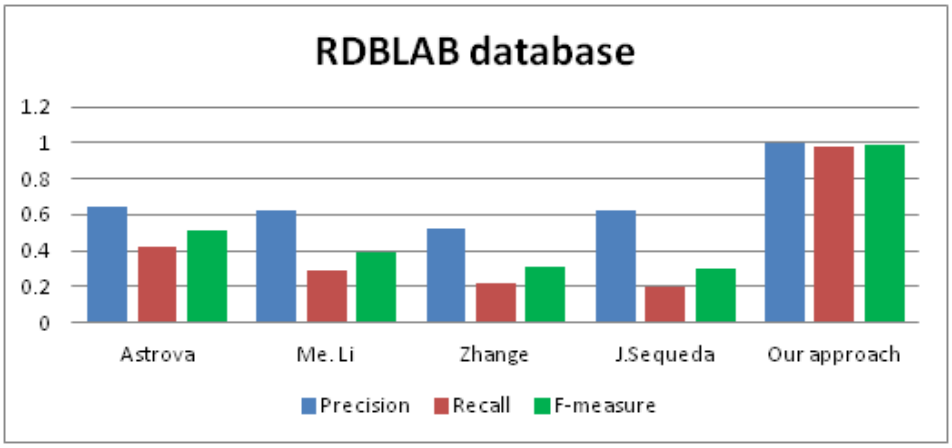


Fig. 13: Matching comparison between our method and related work on RDBLAB database

It can be concluded that one of the main feature of our approach it deals with class properties, properties of object property (property restriction), characteristics of properties and cardinality constraints in ontology building. These criteria make the constructed ontology more integrated and enable the reconstruction of RDB.

7 Conclusion and Future Work

This paper treated the area of ontology-based cooperation of information systems. We proposed a new approach for direct mapping of the relational database Schema to semantic web ontology built in OWL on top of RDF using vocabulary RDFS and Xml Schema data type. We then transformed the relational data tables to ontological instances formatted as RDF(S)-OWL. By adopting this approach, domain-related experts can be directed-automatically to engage in mapping of different sources of RDB Schema and data to RDF(S)-OWL ontologies. In order to generate semantic web ontology as per the underlying RDB, ontology database mappings are expressed as a set of correspondences that relate the vocabulary of a relational model (table, column, datatype, relationships, integrity constraints, table check constraints etc.) with the ontology model (concept, property, xsd, object property, restriction, instances etc.) automatically using mapping rules. Our approach is divided in two phases: firstly transforms an RDB schema into the ontology schema, secondly transforms an RDB data into

the ontology instances. Ontology validator is considered in this approach and implemented using Apache Jena in Java and MYSQL (not limited to MYSQL database). Our rules approach and adopt the methods clearly, easily, and closely to the software programmers. This may help software engineers to build the semantic web from relational databases rapidly and particularly, this is particularly important for mining semantic information from huge web resources. One of the main advantages of this approach is studied different cases of an RDB, which decreases the loss of information and avoided ambiguity where rules are applied.

For the future work, we confess that the domain and success frequency of direct transformation is very improbable to be achieving because the amount of domain semantics captured in SQL models are highly variable. There is a domain for extending this work by extracting new mapping rules and querying relational databases on the semantic web using an ontology generated by our rules. In this paper, static structures of relational database are studied, while the dynamic aspects such as triggers are not considered, which are still open research question.

Acknowledgement

This work is supported by National Natural Science Foundation of China under grants 61173170, 61300222, 61370230, 61433006, 61572221 and U1401258, Innovation Fund of Huazhong University of Science and Technology under grants 2015TS069 and 2015TS071, Science and Technology Support Program of Hubei Province under grant 2014BCH270 and 2015AAA013, and Science and Technology Program of Guangdong Province under grant 2014B010111007.

References

[1] T. Berners-Lee, J. Hendler, O. Lassila, *The Semantic Web*, Scientific American **284(5)**, 28-37 (2001).

[2] J.G. Breslin, D. O'Sullivan, A. Passant, L. Vasiliu, *Semantic Web Computing in Industry*, Computers in Industry **61(8)**, 729-741 (2010).

[3] O. Vasilecas, D. Bugaite, J. Trinkunas, *On Approach for Enterprise Ontology Transformation into Conceptual Model*, in: Proceedings of the International Conference on Computer Systems and Technologies, Varna, Bulgaria, IIIA.23-1-IIIA.23-6 (2006).

[4] H. Zhuge, Y. Xing, P. Shi, *Resource Space Model, OWL and Database: Mapping and Integration*, ACM Transactions on Internet Technology **8(4)**, 20-50 (2008).

[5] G. Klyne, J.J. Carroll, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation, (2004).

[6] R. Cyganiak, D. Wood, M. Lanthaler, *RDF 1.1 Concepts and Abstract Syntax*, W3C Recommendation, (2014).

[7] D. Brickley, R. Guha, B. McBride, *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Recommendation, (2004).

[8] B. Motik, B.C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, *OWL 2 Web Ontology Language: Profiles*, W3C Recommendation, (2009).

[9] K.C-C. Chang, B. He, C. Li, M. Patel, Z. Zhang, *Structured Databases On the Web: Observations and Implications*, ACM SIGMOD Record **33(3)**, 61-70 (2004).

[10] B. He, M. Patel, Z. Zhang, K.C-C. Chang, *Accessing the Deep Web*, Communications of the ACM **50(5)**, 94-101 (2007).

[11] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, D. Aumueller, *Triplify Light-Weight Linked Data Publication from Relational Databases*, in: Proceedings of the 18th International Conference on World Wide Web, Madrid, Spain, 621-630 (2009).

[12] F. Wu, D.S. Weld, *Automatically Refining the Wikipedia Infobox Ontology*, in: Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 635-644 (2008).

[13] S. Bechhofer, F. Van Harmelen, *OWL Web Ontology Language Reference*, W3C Recommendation, (2004).

[14] S. Suwanmanee, D. Benslimane, P. Champin, P. Thiran, *Wrapping and Integrating Heterogeneous Databases with OWL*, in: Proceedings of 7th International Conference on Enterprise Information Systems, Miami, USA, 11-18 (2005).

[15] W. Hu, Y. Qu, *Discovering Simple Mappings between Relational Database Schemas and Ontologies*, in: Proceedings of the 6th International the Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, Busan, Korea, 225-238 (2007).

[16] S.S. Sahoo, W. Halb, S. Hellmann, K. Idehen, J.T. Thibodeau, S. Auer, J. Sequeda, A. Ezzat, *A Survey of Current Approaches for Mapping of Relational Databases to RDF*, W3C RDB2RDF Incubator Group Report, (2009).

[17] H. Mohamed, Y. Jincai, J. Qian, *Towards Integration Rules of Mapping from Relational Databases to Semantic Web Ontology*, Web Information Systems and Mining, Sanya, China, 335-339 (2011).

[18] L. Zhang, J. Li, *Automatic Generation of Ontology Based on Database*, Journal of Computational Information Systems **7(4)**, 1148-1154 (2011).

[19] M. Arenas, A. Bertails, E. Prud'hommeaux, J.F. Sequeda, *A Direct Mapping of Relational Data to RDF*, W3C Recommendation, (2012).

[20] A. Bertails, E.G. Prud'hommeaux, *Interpreting Relational Databases in the RDF Domain*, in: Proceedings of the sixth International Conference on Knowledge Capture, Banff, Alberta, 129-136 (2011).

[21] I. Astrova, *Reverse Engineering of Relational Databases to Ontologies*, in: Proceeding of 1st European Semantic Web Symposium, Heraklion, Crete, 327-341 (2004).

[22] A. Buccella, M.R. Penabad, F. Rodriguez, A. Farina, A. Cechich, *From Relational Databases to OWL Ontologies*, in: Proceedings of the 6th National Russian Research Conference on Digital Libraries, Pushchino, Russia, (2004).

[23] M. Li, X-Y. Du, S. Wang, *Learning Ontology from Relational Database*, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 3410-3415 (2005).

[24] G. Shen, Z. Huang, X. Zhu, X. Zhao, *Research on the Rules of Mapping from Relational Model to OWL*, in: Proceedings of the Workshop on OWL: Experiences and Directions, Athens, Georgia, 548-552 (2006).
[25] L. Stojanovic, N. Stojanovic, R. Volz, *Migrating Data-Intensive Web Sites Into the Semantic Web*, in: Proceedings of the 2002 ACM Symposium on Applied Computing, Madrid, Spain, 1100-1107 (2002).
[26] C. Hu, H. Li, X. Zhang, C. Zhao, *Research and Implementation of Domain-Specific Ontology Building from Relational Database*, in: Proceedings of Third China Grid Annual Conference, Gansu, China, 289-293 (2008).
[27] S. Zhou, G. Meng, H. Ling, H. Zhang, *Tool for Translating Relational Databases Schema into Ontology for Semantic Web*, in: Proceedings of the 2nd International Workshop on Education Technology and Computer Science, Wuhan, China, 198-201 (2010).
[28] Z. Xu, S. Zhang, Y. Dong, *Mapping between Relational Database Schema and OWL Ontology for Deep Annotation*, in: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong, China, 548-552 (2006).
[29] J. Barrasa, Ó. Corcho, A. Gómez-Pérez, *R2O an Extensible and Semantically based Database-to-Ontology Mapping Language*, in: Proceedings of the 2nd Workshop on Semantic Web and Databases, Toronto, Canada, 1069-1070 (2004).
[30] C. Bizer, A. Seaborne, *D2RQ-Treating non-RDF Databases as Virtual RDF Graphs*, in: Proceedings of the 3rd International Semantic Web Conference, Hiroshima, Japan, 457-460 (2004).
[31] E. Marx, P. Salas, K. Breitman, J. Viterbo, M.A. Casanova, *RDB2RDF: A Relational to RDF Plug-in for Eclipse*, Software: Practice and Experience **43(4)**, 435-447 (2013).
[32] G. Mecca, P. Papotti, S. Raunich, *Core Schema Mappings*, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, Providence, USA, 655-668 (2009).
[33] I. Astrova, N. Korda, A. Kalja, *Rule-based Transformation of SQL Relational Databases to OWL Ontologies*, in: Proceedings of the 2nd International Conference on Metadata & Semantics Research, Corfu Island, Greece, 213-216 (2007).
[34] X. Zhou, G. Xu, L. Liu, *An Approach for Ontology Construction Based on Relational Database*, International Journal of Research and Reviews in Artificial Intelligence **1(1)**, 16-19 (2011).
[35] S. Yang, Y. Zheng, X. Yang, *Semi-automatically Building Ontologies from Relational Databases*, in: Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, Chengdu, China, 150-154 (2010).
[36] E. Dragut, R. Lawrence, *Composing Mappings between Schemas using a Reference Ontology*, in: Proceeding of International Conference on Ontologies, Databases and Applications of Semantics, Agia Napa, Cyprus, 783-800 (2004).
[37] M. Hert, G. Reif, H.C. Gall, *A Comparison of RDB-to-RDF Mapping Languages*, in: Proceedings of the 7th International Conference on Semantic Systems, Graz, Austria, 25-32 (2011).
[38] J. F. Sequeda, M. Arenas, D. P. Miranker, *On Directly Mapping Relational Databases to RDF and OWL*, in: Proceedings of the 21st International Conference on World Wide Web, Lyon, France, 649-658 (2012).
[39] D.L. McGuinness, F. Harmelen, *OWL Web Ontology Language Overview*, W3C Recommendation, (2004).
[40] P. Biron, A. Malhotra, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, (2004).
[41] M. Ehrig, J. Euzenat, *Relaxed Precision and Recall for Ontology Matching*, in: Proceeding of the K-Cap 2005 Workshop on Integrating Ontology, 25-32 (2005).

Mohamed A. G. Hazber received the B.S. degree from School of Computer Science and information systems at University of Technology-Baghdad-Iraq 2000 and his M.S. degree from School of Computer Science and Technology at Central China Normal University 2011. Now he is a Ph.D. candidate in the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include Knowledge extraction, Relational database, Semantic Web and Ontologies, Database and Ontology Integration.

Ruixuan Li received the B.S., M.S. and Ph.D. in Computer Science from Huazhong University of Science and Technology, China in 1997, 2000 and 2004 respectively. He was a Visiting Researcher in Department of Electrical and Computer Engineering at University of Toronto from 2009 to 2010. He is currently a Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include big data management, cloud computing, and distributed system security. He is a member of IEEE and ACM.

Xiwu Gu received the BS, MS, and PhD degrees in computer science from Huazhong University of Science and Technology in 1989, 1998, and 2007 respectively. He is currently a lecturer in the School of Computer Science and Technology at Huazhong University of Science and Technology, Wuhan, China. His research interests include distributed computing, data mining, social computing. He is a member of the China Computer Federation (CCF).

Guandong Xu currently is a Senior Lecturer in the Advanced Analytics Institute at University of Technology Sydney following a research fellow in Centre for Applied Informatics at Victoria University. He received MSc and BSc degree in Computer Science and Engineering from Zhejiang University, China. He gained PhD degree in Computer Science from Victoria University. After that he worked as a Postdoctoral research fellow in the Centre for Applied Informatics at Victoria University and then Postdoc in Department of Computer Science at Aalborg University, Denmark. He is an Endeavour Postdoctoral Research Fellow in the University of Tokyo in 2008. His research interests include Data mining, Machine learning, Web usage mining, Web community, Web personalization and Recommender System , Information retrieval and processing, Web search ,Social network analysis, Social media mining, Social Analytics.