STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIV**, Number 2, 2009

RELATIONAL DATABASES AND RESOURCE DESCRIPTION FRAMEWORK

LEON ŢÂMBULEA AND ANDREEA SABĂU

Abstract. A large number of Web pages are generated from relational databases. Many papers appeared lately, that mention the advantages offered by Semantic Web in a possible global query of the informational sources over the Internet. It seems that one of the current challenges is to use databases (great amount of data, but with local usage) by specialized programs in the context of future development of the Internet (in Semantic Web). This paper contains a formalization of types of triplets that can appear in Resource Description Framework (RDF) documents. An automatic conversion method of information from a relational database into RDF documents in described based on the presented model.

1. Introduction

The Web is one of the most popular and richest sources of information. According to [17], in November 2009 were received about 240 million answers from world's Web sites. Some Netcraft statistics mention the fact that on every site there are on average 273 Web pages, and on the Internet there are about 65 billion Web pages, which contain a very large amount of information.

The source of data for a Web page is generally a relational database. These Web pages (static pages or dynamic pages) are built at certain moments of time or as an answer to some events that appear in dynamic pages forms. In addition, there are also many Web sites that offer a lot of documentation to be consulted (tutorials, articles, research reports, books), which represents a great quantity of static information.

The Web pages which are generated from different data sources or built by users are written in different languages and are organized in different formats. Most of them are html or xml documents. The documents that can be found

Received by the editors: December 7, 2009.

2010 *Mathematics Subject Classification.* 68P05, 68P20, 68T30.

1998 *CR Categories and Descriptors.* E.1. [**Data**]: Data Structures; H.2.1. [**Information Systems**]: Database Management – *Logical Design* H.3.5. [**Information Systems**]: Information Storage And Retrieval – *Online Information Services*.

Key words and phrases. Semantic Web, RDF, Knowledge.

on the Internet do not have the contained information structured according to a certain pattern; therefore obtaining some data from different document is not an easy task. The search engines are able to perform search operations within the Web using key words, but they are not able to perform reasoning or to interpret the results of the search (the obtained documents) [1, 2, 11]. Today, such reasoning over a page or document can be performed only by a human user. The Web documents should meet certain requirements, have certain structure, or contain also some metadata with a unique format, in order to allow a program (software agent) to perform reasoning.

Adding such metadata in Web pages or creating documents containing a unique pattern for metadata will lead to a new generation of Web (Web 3.0 or Semantic Web). There are more proposed methods for organizing this metadata. The simplest model is RDF (Resource Description Framework), which was already declared as standard by W3C [14].

2. Relational Databases and RDF

Most data that is used in generating Web pages is extracted from databases. The relational databases had a rapid evolution after their first formalization [5]. The storage of data in relational databases benefits of a lot of advantages, like:

- optimization of the access to stored data,
- control of the concurrent access to data,
- implementation of different security procedures.

Different objects are used in management of relational databases, objects which are usually included in programming languages or environments. These objects are using drivers or providers (like ODBC, JDBC, OLEDB, SQL-NCLI, MSDAORA etc.) which make possible the management of data (stored on different management systems) from different programming languages and operating systems.

The metadata of a RDF document is specified as a sequence of instructions, where such an instruction is a triplet (subject, predicate, object). This kind of instruction is similar to a simple sentence from a natural language. For example, having the sentence "The paper appears in Studia Informatica", the subject can be considered to be "the paper", "appears" can be seen as predicate (or property), and "Studia Informatica" is the correspondent for the object of an instruction. The objects from a sentence can be described by an instruction, within which the same item can be subject. For example: "Studia Informatica has the home-page at http://www.cs.ubbcluj.ro/∼studia-i/".

If a RDF instruction is wanted to be used in a wider context (ideally - a global context in Web's space), a described subject (or an object), or a predicate
(property) should be uniquely identified on Web. Therefore, we can say they are *unique resources*. This uniqueness over the Web can be guaranteed by an URI (Uniform Resource Identifier).

So far, we considered very large sources of information, which are relational databases with local usage from dedicated programs, and RDF documents, with included metadata for Web documents and with the possibility of complex queries integration. Having these sources, the raised problem is to convert a relational database (or a part of data from a relational database, a part which is considered to be useful and not confidential to be published) into a RDF document [1, 6, 8, 9, 10, 15]. For example, [12, 13] are offering very large sized RDF documents to be used.

3. Relational Databases

A relational database consists of a number of components that can be defined, modified, or deleted using SQL commands. In order to retrieve data from a relational database, only the relations (tables) and views are useful. Let's consider:

$$B = \{T_i, i := 1, m\} \cup \{V_j, j := 1, n\}$$

the relations and views of a relational database. Each table $T_i, i := 1, m$, has a schema defined by an SQL statement:

$$\text{CREATE TABLE } T_i(column\_definition[, column\_definition]... \\ [, table\_constraints])$$

where $column\_definition$ is given by:

$$column\_name\ column\_type[(length)][column\_constraints]$$

Each database view $V_j, j := 1, n$, can be defined using an SQL statement containing one or more $SELECT$ statements (retrieving data from one or more sources). Also, for a view, the column list (its structure) is known.

The definitions of a relational database's components are stored in a dictionary which can be also consulted by users with proper rights.

The constraints that are defined when the tables are created have the role of keeping the correctness of stored data. These rules are checked every time some data is added, modified, or deleted. If only a data retrieval is performed, there is no reason to check the integrity constraints (only, possibly, the user rights). When some complex queries are executed, especially those who need join operations to be performed over two or more tables, the indexes created on primary key columns and foreign key columns are usually used for optimizing the retrieval of data.

4. RDF

A RDF document [3, 6, 7, 14] is given by a set of triplets (s, p, v), where $s =$ subject, $p =$ predicate (property), and $v =$ the property's value (a literal value or an object). If the property's value is an object, it should be described in the same manner as a subject; therefore these two notions (subject, object) are identical.

Such a triplet represents information with the following meaning: "The object (subject) s has the property (predicate) p with value v". Therefore, the predicates are similar to binary operators, because they associate a subject s with a value v: $p(s, v)$.

In order to use the objects and their properties in a (as large as possible) context, eventually outside the current document, a unique identification of these is necessary. Such an identification is accomplished using an URI (for example: http://address or ftp://address), in this manner obtaining a *unique resource*. Using such identification items, some relations (links) between resources from different documents may be established. Thus, one property (defined in a RDF document) associates a value (a literal or an object, defined in the same document or in another) to an object (also defined in the same or another document).

When a new resource is defined, an ID attribute may be used, having a value which will identify that resource. Therefore, the value of this ID attribute will be used when the corresponding resource is used (for example: in retrieving the value of one of its properties).

A set of RDF triplets can be stored in different ways:

(1) Using an oriented graph, where the vertices represent objects (subjects) or literals, and the edges represent the connections between objects and values, having the name (identification) of the property as label; the vertices are labeled using the ID of the corresponding resource or the value of the literal.
(2) In an XML document.
(3) Other formats: Notation 3 (N3) [18], Turtle [19], etc.

In order to specify the value of a property which can be decomposed in more pairs (property, value), a *blank node* will be used (as an intermediary resource useful only within the current document).

The value of a property may be:

(a) a value that can be specified by a string (a literal),

(b) a resource (a complex object) having different properties and associated values; the blank nodes are used in this case.

These cases can be graphically represented as a graph (see figure 1). In figure 1.c a blank node is used.

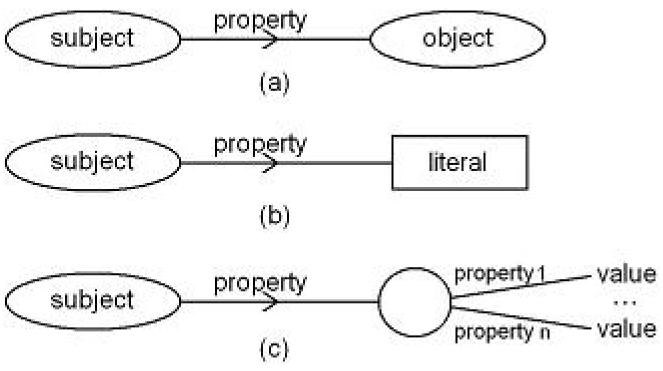

FIGURE 1. The graphical representation of the associations between properties and their values.

In order to formalize these concepts, the following notations will be used:

R = the set of URIs (resource identifiers)
V = the set of literals (the basic values which will not be described by other triplets)
P = the set of properties, $P \subset R$

Let T be the set of triplets within a RDF document, $T \subset R \times P \times (V \cup R)$. An oriented graph corresponding to set T can be built:

$G_T = (N, E, L)$,

where:

- N = the set of vertices (resources or literals), $N = \{r \mid (r,p,v) \in T\} \cup \{v \mid (r,p,v) \in T\}$. One node $n \in N$ is labeled with the ID of the corresponding resource or with the value of the corresponding literal. In the graph G_T, all nodes corresponding to resources are distinct.
- E = the set of edges, $E = \{(r,v) \mid (r,p,v) \in T\}$.
- L = the set of labels associated to edges of G_T, $L : E \to P$, $L(r,v) = p$, for $(r,p,v) \in T$.

The properties (and their values) are defined by the set of triplets from a RDF document, for a set of resources. This association is free from restrictions. In order to restrict the association of properties to a resource, a pattern should be defined for that resource. The definition of such a pattern can be done using a *class* (or a *class hierarchy*). A set of *properties* can be associated to a class once defined. Such definitions (for classes, properties, but also for sub-classes, sub-properties) can be built using the triplets (r, p, v) that were mentioned before.

The definition of classes and properties can be done using RDF schemas (noted RDFS, which is an extension of RDF). As mentioned in [6], RDF and RDFS are namespaces with the same meaning, but used with different names because of historical reasons. Properties defined in these two namespaces (rdf:*property* or $rdfs$:*property*) are used in order to define resources.

A defined resource can be a class, a property, a sub-class, a sub-property, or an instance or a defined class. Such an instance of a class can make use of the properties associated to its class (thus - a model), but also can add some new properties.

Let (r, p, v) be a triplet which defines a resource, where r represents the resource (class, sub-class, property, sub-property). We may have the following definitions:

- $p = rdf{:}type$, $v = rdfs{:}Class$, where the property is indicating that the type of the resource is defined, and the value shows the fact that this resource is a class;
- $p = rdfs{:}subClassOf$, where p is mentioning that the type of the resource is a sub-class (therefore - is inheriting another class), $v =$ reference to a resource that is a class (and which is inherited);

For the triplets $(r, rdf{:}type, rdfs{:}Class)$, $(c1, rdf{:}type, rdfs{:}Class)$, $(r, rdf{:}subClassOf, c1)$ the graph represented in figure 2 can be built.

- $p = rdf{:}type$, $v = rdf{:}Property$, indicating that the defined resource is a property;
- $p = rdfs{:}subPropertyOf$, where p is showing that the resource is a sub-property, $v =$ reference to a resource of type property;
- $p = rdfs{:}domain$, or $p = rdfs{:}range$, $v =$ reference to a resource; in this case the domain or the range of the defined resource is indicated. The domain indicates the subject (the resource) to which it is associated, and the range is the type of the value;

The following relations result from these definitions:

$rdfs{:}Class \in R$
$rdf{:}type$, $rdfs{:}subClassOf$, $rdf{:}Property$, $rdfs{:}subPropertyOf \in P$
$rdfs{:}domain$, $rdfs{:}range \in P$

The following two situations specify the type of values for resource r, for the triplet (r, p, v):

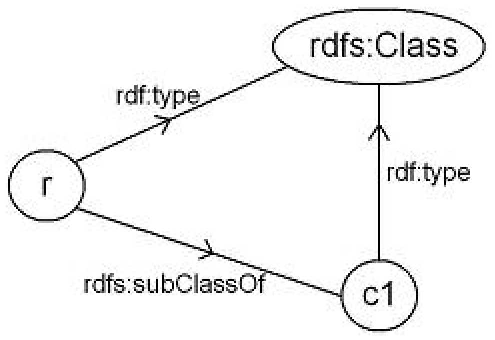

FIGURE 2. The oriented graph corresponding to triplets $(r, rdf{:}type, rdfs{:}Class)$, $(c1, rdf{:}type, rdfs{:}Class)$, $(r, rdf{:}subClassOf, c1)$.

- $p = rdf{:}type$, $v = rdfs{:}Literal$; in this case the value of the resource is a literal (a string);
- $p = rdf{:}type$, $v =$ reference to a resource which is a class or a sub-class; in this case the resource r is an instance of the class indicated by v.

A number of rules of deduction can be defined over a set of triplets, and these rules ensure obtaining new triplets [16]. Some of these rules may be:

(1) Determining the instances of a class:
(x, p, y), $(p, rdfs{:}domain, z) \Rightarrow (x, rdf{:}type, z)$
(x, p, y), $(p, rdfs{:}range, z) \Rightarrow (y, rdf{:}type, z)$
$(x, rdf{:}type, y)$, $(y, rdfs{:}subClassOf, z) \Rightarrow (x, rdf{:}type, z)$;

(2) Determining the transitive closure of a property by repeatedly using the following rule:
$(x, rdfs{:}subPropertyOf, y)$, $(y, rdfs{:}subPropertyOf, z) \Rightarrow (x, rdfs{:}subpropertyOf, z)$;

(3) Determining the transitive closure of a class by repeatedly using the following rule:
$(r, rdfs{:}subClassOf, y)$, $(y, rdfs{:}subClassOf, z) \Rightarrow (x, rdfs{:}subClassOf, z)$.

The existing definitions in a set of triplets T or the definitions that can be deduced by rules of deduction must comply with some restrictions, among which:

(a) There cannot be cycles, thus:
$\forall p, (p, rdfs{:}subPropertyOf, p) \notin T$,
$\forall c, (c, rdfs{:}subClassOf, c) \notin T$.

(b) A property may have one or more domains, but only one range:

$$(r, rdfs{:}range, x) \Rightarrow \nexists y, y \neq x, (p, rdfs{:}range, y).$$

A set of restrictions that violates at least a restriction is called *inconsistent*. Such a set may provide erroneous data to different queries.

5. Conversion of Relational Databases to RDF

There are a lot of papers that contain studies about conversion of relational databases to RDF documents. A working group was established at W3C (Incubator Group) in order to synthesize the papers from this domain, and some of its results were published in 2009 in [15].

Some results from [1, 2, 4, 6, 8, 9, 10, 15] were used for the following remarks:

1. A first proposal for converting relational databases to RDF is obtained in a natural manner. One RDF class is built corresponding to each data source $S[A_1, A_2, ..., A_n]$ (table or view) contained in a relational database. Such a class has the name of the source, and the columns of the source generate the properties of that class. Therefore, for the source S the following triplet will be obtained:

$$(S, rdf{:}type, rdfs{:}Class),$$

and for each column $A_i, i = 1, n$, the RDF document will contain the following triplets (see figure 3):

$$(S.A, rdf{:}type, rdf{:}Property),$$
$$(S.A, rdfs{:}domain, S),$$
$$(S.A, rdfs{:}range, rdfs{:}Literal).$$

Each line (record) from S will generate an instance of the class generated by S through a blank node (the type of this node is the RDF class having the name of the source S). The ID for this node may be automatically generated (as for a column of type auto-increment that exists in many relational database systems).

The structure (schema) of each of the sources from a relational database can be obtained from the database dictionary; therefore such a conversion can be automatically obtained. Some options can be used to such a conversion, like:

(a) Some source columns may change their name when they are transformed into properties. The new names may be taken from vocabularies that already exist and are specialized for certain domains. In this case, some associations between column names and the new RDF property names would be necessary.

(b) It is possible that not all tables and views, or not all data from these sources to be necessary in a conversion from relational database to

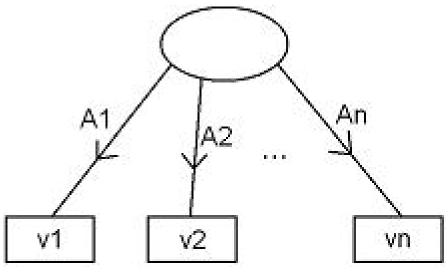

FIGURE 3. The conversion of a data source $S[A_1, A_2, ..., A_n]$ (from a relational database) to RDF document.

RDF. Therefore, in order to perform a conversion, a list of tables, views, and $SELECT$ statements can be used in order to establish the actual sources of data.

(c) We usually define a primary key for a table from a database. This primary key may consist of one or more columns and is used to uniquely identify a record within the corresponding table. In order to use the primary key's values in identifying the blank nodes from the RDF document, some strings may be generated from these values (for example: $S.v$, where S is the table's name and v is the key's value). If the primary key contains more than one column, then the values of all these columns may be concatenated. If the columns use to uniquely identify the rows within a table are not needed in the conversion to RDF, then they may not be extracted in the RDF document.

2. In a relational database, a foreign key constraint can be defined between two tables:

- table $T1$ with a primary key PK,
- table $T2$ with a foreign key FK, which references PK from $T1$.

These restrictions are usually created in the relations normalization process.

A query engine (of a relational database system) is able to optimize the queries that have to compute join between $T1$ and $T2$, using the indexes defined on the primary columns and foreign key columns, respectively. On the other side, the RDF documents that are obtained after a conversion of these tables are not efficient in performing queries on them. If the conversion method previously described is used in conversion, a join operation between the obtained documents is not efficient, because there are no indexes to be used. In order to improve the search, an auxiliary property of class $T2$ may be defined. This property would have the domain to $T2$ class, the range as $T1$ class, and the name of the foreign key constraint.

3. In order to model $m : n$ (many-to-many) relationships between two tables $T1$ and $T2$ in a relational database, an auxiliary table is created, $T3$. The third table materializes the conceptual $m : n$ relationship, and two $1 : n$ (one-to-many) relationships are obtained in database (between $T1$ and $T3$, and between $T2$ and $T3$).

As an example, the four tables represented in figure 4 are considered to exist in a relational database.

Departments

id	name
1	Math
2	Comp science

Courses

id	title
c1	course1
c2	course2
c3	course3

Students

id	name	dept
s1	stud1	1
s2	stud2	2
s3	stud3	2

CS

idc	ids
c1	s1
c2	s1
c1	s2
c3	s2
c2	s3
c3	s3

Figure 4. Example of four tables within a relational database.

The following foreign key constraints are created:

$FK\_CS\_Courses$: $CS(idc)$ references $Courses(id)$
FK_CS_Studs: $CS(ids)$ references $Students(id)$
FK_Studs_Dept: $Students(dept)$ references $Departments(id)$

In figure 5 are represented three of the fourteen blank nodes that are obtained corresponding to records from the four tables, using the method previously described and having one property for a 1:n relationship.

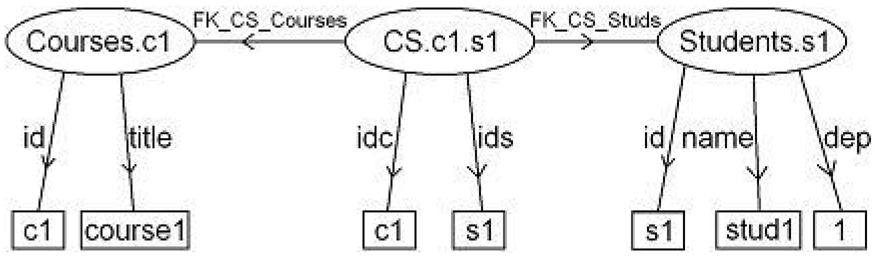

Figure 5. Three blank nodes that are obtained after the conversion of tables *Courses*, *Departments*, *Students*, and *CS*.

The properties *idc* and *ids* can be removed, because the table *CS* was created only in order to represent the $m : n$ relationship. Their values can be obtained by using the properties *FK_CS_Courses* and *FK_CS_Studs*. Once the properties *idc* and *ids* are removed, one blank node is obtained, without properties corresponding to columns of table that generated it. This is the reason why this blank node can be also removed; therefore the two existing properties (*FK_CS_Courses* and *FK_CS_Studs*) change their domain, as it is represented in figure 6.

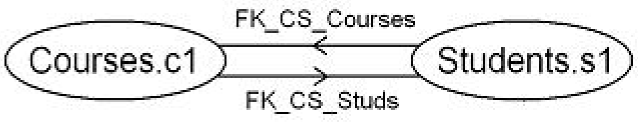

Figure 6. The two blank nodes obtained after removal of blank node corresponding to *CS* table.

6. Conclusions

The RDF documents are very large data warehouses that can be used by queries in Semantic Web. But, in order to obtain them, the conversion of data from other sources (usually from relational databases) is necessary. This paper analyzed the main characteristics of relational databases and RDF, important for such a conversion, and presented three conversion techniques.

References

[1] T. Berners-Lee, *Relational Databases and the Semantic Web (in Design Issues)*, http://www.w3.org/DesignIssues/RDB-RDF.html (1998).

[2] C. Bizer, *D2R MAP-A Database to RDF Mapping Language*, In Proceedings of the 12th International World Wide Web Conference, WWW2003, Budapest, Hungary (2003).

[3] S. C. Buraga, *Considerations Regarding the Use of Semantic Web Technologies in the Context of E-business Applications*, Informatica Economica, 3(35) (2005).

[4] K. Byrne, *Populating the Semantic Web - Combining Text and Relational Databases as RDF Graphs*, PhD Thesis, Edinburgh (2008).

[5] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, 13(6) (1970), pp. 377-387.

[6] F. Frasincar, G. J. Houben, R. Vdovjak, P. Barna, *RAL: An Algebra for Querying RDF*, World Wide Web, 7(1) (2004), pp. 83-109.

[7] I. Herman, *Introduction to the Semantic Web*, 2nd European Semantic Technology Conference, Vienna, Austria, (2008).

[8] C. Prez De Laborda, *Incorporating Relational Data into the Semantic Web*, PhD Thesis (2006).

[9] J. F. Sequeda, S. Tirmizi, D. Miranker, *SQL Databases are a Moving Target*, Position Paper for W3C Workshop on RDF Access to Relational Databases, Cambridge, USA (2007).
[10] M. Svihla, I. Jelnek, *Two Layer Mapping from Database to RDF*, In Proceedings of Electronic Computers and Informatics (ECI), Slovakia (2004).
[11] L. Ţâmbulea, A. Sabău, *From Databases to Semantic Web*, Studia Universitatis Babeş-Bolyai, Seria Informatica, Special Issue, LIV (2009), for Intl. Conf. KEPT Knowledge Engineering Principles and Techniques Cluj-Napoca (2009), pp. 85-88.
[12] *D2R Server publishing the DBLP Bibliography Database*, http://dblp.l3s.de/d2r/.
[13] *DBpedia*, http://dbpedia.org/About.
[14] *RDF/XML Syntax Specification (Revised 2004)*, W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.
[15] *A Survey of Current Approaches for Mapping of Relational Databases to RDF*, W3C RDB2RDF Incubator Group, http://esw.w3.org/topic/Rdb2RdfXG/StateOfTheArt.
[16] *RDF Semantics*, http://www.w3.org/TR/rdf-mt/.
[17] *November 2009 Web Server Survey*, http://news.netcraft.com/archives/web_server_survey.html.
[18] *Notation 3*, http://www.5.org/DesignIssues/Notation3.html.
[19] *Turtle - Terse RDF Triple Language*, http://www.dajobe.org/2004/01/turtle.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: leon@cs.ubbcluj.ro

E-mail address: deiush@cs.ubbcluj.ro