Formalization of Direct Mapping for Relational Data to Semantic Web Representation

Kyawt Kyawt San, Khin Nweni Tun
kyawtkyawts@gmail.com, knntun@gmail.com

Abstract

The Web is one of the most popular and richest sources of information. The source of data for a Web page is generally a relational database which has been recognized as a key factor in generating huge amounts of data for semantic web applications. Relational databases and the Semantic Web, in particular RDF as its main data representation format, are two different approaches for modeling and storing data permanently. To be accessible data from these relational databases, we have to map relational data to RDF. In addition, extracting knowledge, transforming and understanding data from tables are the interesting problems in the areas of databases. The problem of mapping databases to RDF has received formal attention from the World Wide Web Consortium (W3C). When mapping relational databases into RDF, we can choose mappings which preserve relational constraints and still use the linking capabilities of RDF. Which approach is the most appropriate one depends on the kind of applications for which a resulting RDF graph is assumed to be used. In this approach, in order to transform relational data into Semantic Web representation, we apply formalization rules to complete our mapping process. We also propose a rule to handle blank node which is one of the problems of RDF data model. Our main aim is to be easily accessible relational data content on Semantic web. This is the initial approach for transforming relational data and schema items into Semantic Web representation.

1. Introduction

Most of the world's data today are still stored in relational databases. The success of the Semantic Web hinges on developing methods for making relational databases accessible to the Semantic Web. Relational databases and the Semantic Web, in particular RDF as its main data representation format, are two different approaches for modeling and storing data permanently. The idea of the semantic web is to support semantic interoperability between programs exchanging data. Resource Description Framework (RDF) is considered as the basic for building Semantic web applications. The RDF model is a graph-based model based on the concept of resources, triples and statements. A resource can be anything that can be talked about: a web document, a real world entity or an abstract concept and is identified by a URI (Uniform Resource Identifier). Thus, information on web pages can be represented by using RDF as a set of so called triples, where each triple states a subject-property-object relationship. As each such triple can be understood as a directed edge from the subject to the object, where the edge is labeled with the respective property, instead of a set of triples a corresponding RDF graph is considered. Thus, exporting data from relational databases to the semantic web using RDF basically means to map the relational data into an RDF graph.

Taking this objective into account, the content of a relational database will be made available in RDF, either in the form of results from an RDF query language like SPARQL or as RDF documents. Many different mapping algorithms are used in order to fulfill this objective. Choosing the most appropriate approach depends on the kind of applications for which a resulting RDF graph is assumed to be used. Based on the previous research works, we present a simple mapping procedure by using formalization rules in order to map relational data to RDF documents including the blank node structure handling. The structure of the paper is as follows. Section 2 presents basics of relational data model and RDF data model. Basic conversion procedure and a simple example are presented in Section 3. In section 4, we use formalization rules presented in [2] to formalize our direct mapping process and propose rule for handling blank node. Section 5 concludes the paper.

2. Relational Data Model and RDF Data Model

A relational database consists of tables, which consist of tuples or records. Each tuple consists of a set of attribute values. A tuple, therefore, is comprised of the contents of its attributes, just as an RDF node is nothing but the confluence of many property arcs. The correspondence between RDF and relational databases is:

- A tuple is an RDF subject.
- An attribute is an RDF predicate.
- An attribute value is an RDF object.

As compared to the relational data model, RDF has gained great interest in both academia and industry as an important language in describing graph data. RDF is proposed as a simple data model which allows anyone to make statements about any resource.

In RDF, each data entity has a Unique Resource Identifier (URI) and each relationship between two data entities is described via a triple within which the items take the roles of subject (S), predicate (or property)(P) and object(O). In order to express such a description, RDF uses a number of triples (n3 notation):

<Subject> <Predicate> <Object>
The above is equivalent to:
<Resource> <Property> <Property Value>

This triple is expressed in the form of an RDF statement which can easily be represented as a directed graph in which a node represents the subject, another node for the object and a connecting arc for the predicate, directed from the subject node towards the object node. Graphical representation of this triple can be represented as a graph as shown in Figure.

RDF Data Model

U: set of URIs
B: set of blank nodes
L: set of literals
$(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple
A set of RDF triples is called an RDF graph.

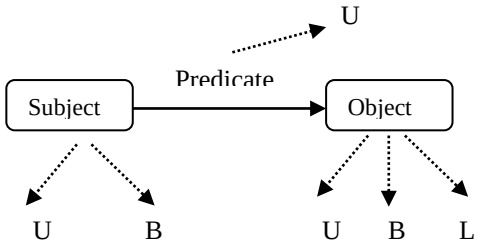


Figure2.1. Representation of RDF Graph

2.1 Mapping Approaches for RDB2RDF

Despite many efforts helping to realize Semantic Web application, the actual semantic web still lacks of enough semantic data. Most information is still modeled and stored in relational databases and thus out of reach for many Semantic Web applications. As a consequence, a technology to map relational data to the Semantic Web is required, which enables the user to create arbitrary mappings to Semantic Web ontologies, without having to adopt a novel mapping language. Additionally, this mapping technique should take the characteristics of relational databases into account, particularly data and schema evolution, i.e. enable users to access the databases transparently using Semantic Web techniques, without having to repeat the data translation process every time a modification occurs. Using such a technique, a user is rapidly able to provide Semantic Web applications with semantic rich data, which is actually stored in a relational database. Mapping database schema directly to RDF is much faster, cheaper and usually more suitable for generation of Semantic Web content.

The creation of mappings between RDB and RDF can be classified into two categories according to the survey of W3C Incubator Group:

1. Automatic Mapping Generation: In this mapping,

- A RDB record is a RDF node
- The column name of a RDB table is a RDF predicate
- A RDB table cell is a value

In this direct mapping, the database content and its SQL schema are directly translated to representations in Semantic Web languages i.e. RDF and OWL. Although these automatically generated mappings often do not capture complex domain semantics that are required by many applications, these mappings can serve as a useful starting point to create more customized, domain-specific mappings.

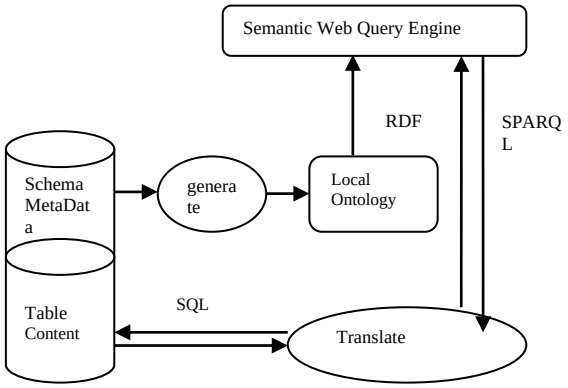


Figure2.1. Direct Mapping of a Relational Database to the Semantic Web

2. Domain Semantics-driven Mapping Generation: This approach generates mappings from RDB to RDF by incorporating domain semantics that is often implicit or not captured at all in the RDB schema. Additionally, a mapping generated by using domain semantics also reduces the creation of triples for redundant or irrelevant knowledge.

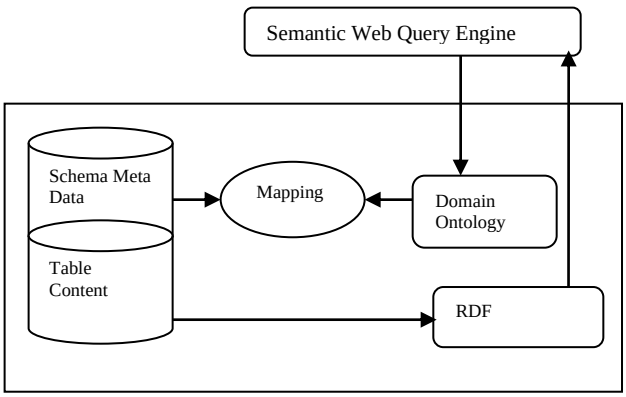


Figure2.2. Relational Database to Ontology Mapping

Undoubtedly, a default way to translate relational databases into RDF (that is, without any input from the user on how the relational data should be translated) is the use of direct mapping. In this approach, we use simple direct mapping method. The direct mapping defines an RDF graph representation of the data in any relational database. The direct mapping takes as input a relational database (data and schema) and generates an RDF graph that is called the direct graph.

3. Basic Transforming Procedure

The basic process is illustrated in Figure 3.3 which shows a much simplified version of the transformed process.
PERSON (ID, NAME): ID is the primary key

STUDENT (RNO, DEGREE, ID): RNO is the primary key; ID is a foreign key to ID in PERSON

The following instances are considered according to the schema:

Table1. PERSON

ID	NAME
111	Tin Tin
112	Yi Yi
113	Mon Mon

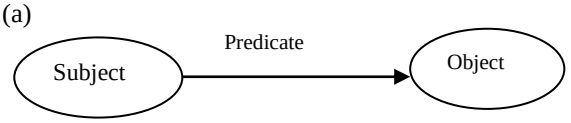
Table2. STUDENT

RNO	DEGREE	ID
1	CS	111
2	CS	112
3	CT	113

Interest in transformation of relational data into RDF reaches a peak at present, and the W3C has recently set up an Incubator Group to work on it. In this section, using database instances described above basic conversion procedure is shown as follows.

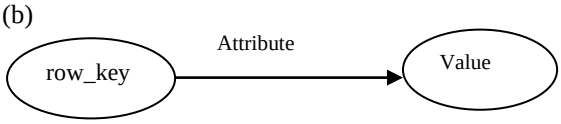
The subject and property of an RDF triple must both be resources with URIs, whilst the object may be either a URI or a literal string, If the object is a literal, as in the basic translation process described above, it automatically becomes a "leaf" node at the edge of the RDF graph, as it cannot be a subject node without a URI.

(a)



is derived from the database as follows:

(b)



where the "attribute" is a URI derived directly from the database field or column name, and the "value" is the content of the field and is represented as a literal value as a basic transformation process. Then, we have one of the triples from the database instances described above:

(c)

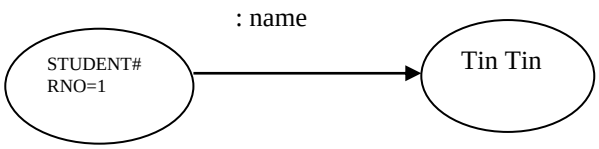


Figure3.3. Simple Transforming from Relational Tuples to RDF

The predicate URI is derived by the "**Column as Predicate**" method.

4. Formalization of Direct Mapping to Translate Relational Data into RDF

There are two main approaches of exporting relational data to Semantic web data like RDF. In the first case data are stored in a database like RDF triples and there is system, which can query these data directly in RDF way. Other approach is when data are stored in classic relational schema and there is some mapping to RDF. As far as the most of data today are stored in classic relational schema, our work is focused to the latter approach. The data used for this work is based on the University relational database schema. The schema of this database is shown below and we use this schema throughout this section.

PERSON (ID, NAME): ID is the primary key
STUDENT (RNO, DEGREE, ID): ROLLNO is the primary key; ID is a foreign key to ID in PERSON
PROFESSOR (ID, TITLE)
DEPT (CODE, NAME)
SEMESTER (SNO, YEAR, SESSION)
COURSE (CNO, TITLE, CODE)
OFFER (ONO, CNO, SNO, PID, CONO)
STUDY (ONO, RNO, GRADE): (ONO, RNO) is the primary key
REG (SID, SNO): (SID, SNO) is the primary key.

We use relational schema and databases instances of this schema as input and RDF graph is produced as output. We describe how the input is specified as shown below and these predicates are used to store a relational schema: The problem of translating relational data to RDF is solved by using formalization rules [2] as follows:

Rel(r): r is a relation name
Example: Rel(DEPT), Rel(COURSE)
Attr(a, r): a is an attribute of relation r

Example: Rel(ONO,OFFER)

PK_n (a_1, . . . , a_n, r): (a_1, . . . , a_n) ($n \geq 1$) is a primary key in r

PK_1(CODE,DEPT)

FK_n (a_1, . . . , a_n, r , b_1, . . . , b_n, s): (a_1, . . . , a_n) ($n \geq 1$) is a foreign key in relation r that references to (b_1, . . . , b_n) in relation s

FK_1(RNO, STUDENT, ID, PERSON)

This predicate is used to store the tuples in a database instance:

Value (v, a, t, r): v is the value of attribute a in a tuple with identifier t in relation r

Example: The STUDENT table is stored by using the following facts:

Value (1, Rno, t1, Student)
Value (CS, Degree, t1, Student)
Value (111,ID, t1, Student)
Value (2, Rno, t2, Student)
Value (CS, Degree, t2, Student)
Value (112, ID, t2, Student)
Value (3, Rno, t3, Student)
Value (CT, Degree, t3, Student)
Value (113, ID, t3, Student)

In the following section, we will show how to generate URIs for the RDF resources.

4.1 Generating URIs

In this section, an URI is created for RDF instances. Generating suitable URIs for the RDF "resources" is one of the key issues. An essential component of RDF graphs is IRIs. IRIs should be generated for relations, attributes and tuples. Assume given a base IRI (http://www.ucsy.edu.mm/). The following is the rules for generating IRIs for relations, attributes and tuples respectively.

(1) IRIs for Relations

RelationIRI(X, Y) ← Rel(X), $Concat_2$ (http://www.ucsy.edu.mm//, X, Y)
Example: http://www.ucsy.edu.mm/PERSON and http://www.ucsy.edu.mm/STUDENT

(2) IRIs for Attributes

IRIs for attributes (n>= 1)

$AttrIRI_n$ (X_1. . . X_n, Y, Z) ← Rel(Y),Attr(X_1, Y), . . . ,Attr(X_n, Y),
$Concat_{2+2n}$ (http://www.ucsy.edu.mm/, Y, "#", X_1, ",", X_2, ",", . . . , ",", X_n, Z)

Example:
http://www.usy.edu.mm/Student#Rno,Name, Id is generated from relation Student.

(3) IRIs for Tuples

IRIs for tuples ($n \geq 1$):

TupleID(X, Y, Z) ← Rel(Y),PK_n (X_1, . . . , X_n, Y),
Value(V_1, X_1, X, Y), . . . ,Value(V_n, X_n, X, Y),
$Concat_{2+4n}$ (http://www.ucsy.edu.mm/,Y,"#", X_1, "=", V_1, ",",X_2, "=", V_2, . . . , ",", X_n, "=", V_n, Z)

Example: http://www.ucsy.edu.mm/Student#Rno=1 is generated.

One extra case is for tuples that does not have a primary key.

HasPK(X) ← PK_n (X_1,. . ., X_n, X) ($n \geq 1$)
TupleID(X, Y, Z) ← Rel(Y), Value(V, A, X, Y), ¬HasPK(X),

$Concat_3$ (:, Y ,-, X, Z)

Example: If student does not have a primary key, then the following blank node would be the identifier of tuple t1:

-: Student-t1

Blank nodes enable indirect referencing, which is close to the human way of thinking, where not everything is precisely identified, but rather expressed by unspecified words (pronouns) as "somebody", "something" and others. The RDF graph generated in the translation process identifies each tuple in the source relational database by means of a URI. If the tuple contains a primary key, then this URI is based on the value of the primary key. If the tuple does not contain such a constraint, then a blank node is used to identify it in the generated RDF graph.

4.2 Problems with the RDF Data Model

In RDF, there are three types of nodes – URI references, blank nodes and literals. URI references identify resources, blank nodes represent anonymous resources that are not assigned a URI, and literals denote values such as numbers or dates. The subject of an RDF triple may be a URI reference or a blank node, the predicate must be a URI reference, and the object may be of all three kinds (URI references, literals, blank nodes). When combined together, RDF triples form a direct, labeled graph. Subjects and objects of RDF triples become nodes in an RDF graph, and predicates become arcs connecting them.

The RDF model is based on a simple idea, but it has problems that make it unnecessarily complicated, thus decreasing its value. These problems can be divided into three categories:

- the existence of nodes that have no name
- problems associated with the literals
- the lack of a unique concept of the node

We intend to tackle the problem of the existence of node that has no name which is known as blank node. The difficulty of using bnode is that they are blank. This means that sub-graphs cannot be straightforwardly linked using them, so each one is

separate even when they really need to be merged. We avoid a lot of error prone work by simply generating URIs so that nodes are automatically merged when appropriate. In the next section, we briefly describe about the blank node.

4.1.1 Blank Nodes

Being part of the RDF specification, blank nodes are a core aspect of Semantic Web technology. The standard semantics for blank nodes denotes the existence of some unnamed resource. Blank nodes focus on representing resources which do not have a natural URI. A blank node can perform two roles in an RDF graph; it can be the object in one RDF statement and the subject in another.

Although adoption of RDF is growing (quite) fast, one of its core features—blank nodes—has been sometimes misunderstood, sometimes misinterpreted, and sometimes ignored by implementers, other standards, and the general Semantic Web community. This lack of consistency between the standard and its actual uses calls for attention.

Due to the absence of a name (URI), manipulating data containing blank nodes is much harder – they make otherwise trivial operations far more complex. They complicate the lives of data consumers, especially if data changes in the future. Blank nodes add a lot of complexity to the standards built upon them, and the implementations consuming them. They are poorly understood and difficult for beginners. When graphs are merged, their blank nodes must be kept distinct if meaning is to be preserved; this may call for re-allocation of blank node identifiers. Such blank node identifiers are not part of the RDF abstract syntax, and the representation of triples containing blank nodes is entirely dependent on the particular concrete syntax used.

4.2.2 Our Rule for Handling Blank Nodes

The resource represented by a blank node is called an anonymous resource for which a URI or literal is not given. This leads to the difficulties in linking subgraphs when they really need to be merged. In theory the conversion of RDB to RDF process could arrange to merge blank nodes when it was certain that the data item was the same, but can avoid a lot of errorprone work by simply generating URIs instead, so that nodes are automatically merged when appropriate. We simply use formalization rules presented in [2] to generate URI for blank node in this way we avoid the problem in merging sub graphs. We simply concatenate our generated URI to blank node.

HasPK(X) ← $PK_n(X_1, \ldots, X_n, X)$ ($n \geq 1$)

TupleID(X, Y , Z) ← Rel(Y),Value(V, A, X, Y),

¬HasPK(X),

$Concat_3$ (http://www.ucsy.edu.mm/, Y, - , X, Z)

If, for example, the table student has no primary key, we will simply concatenate the base URI with the table name following with the tuple identifier.

Example: http://www.ucsy.edu.mm/Student-t1

When we have all the necessary URIs, we have to generate three triples in order to complete our mapping process in the following section**.**

4.2 Generating Triples

Through this section, we use a University relational database schema as a running example. We adopt Formalization rules [2] to complete the conversion process. Three triples are generated during the direct mapping process such as:

- Table triples: For each relation, store the tuples that belong to it.
- Literal triples: For each tuple, store the values in each of its attributes
- Reference triples: Store the references generated by foreign keys

(1) Table Triples

Triple(S, rdf:type,O) ←
Rel(X),Value(V, A, Y , X),
TupleID(Y , X, S),RelationIRI(X,O)

The following triples are generated for relation Student.

Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=1,rdf:type, http://www.ucsy.edu.mm/STUDENT)
Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=2,rdf:type, http://www.ucsy.edu.mm/STUDENT)
Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=2,rdf:type, http://www.ucsy.edu.mm/STUDENT)

(2) Literal Triples

Triple(S, P, O) ←
Rel(X), Value (O, A, Y , X),
TupleID(Y, X, S), AttrIRI(A, X, P)

The following triples are generated from facts
Value (1, RNO, t1, STUDENT) and
Value (CS, DEGREE, t1, STUDENT):

Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=1,
http://exa.org/STUDENT#RNO, 1)

Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=1,
http://exa.org/STUDENT#DEGREE, CS)

(3) Reference Triples

This family of rules is used to generate reference triples ($n \geq 1$):

Triple(S, P, O) ←

FK$_{\mathbf{n}}$ (X1, . . . , X$_{\mathbf{n}}$, X, Y$_{\mathbf{1}}$, . . . , Y$_{\mathbf{n}}$, Y),

Value (V$_{\mathbf{1}}$, X$_{\mathbf{1}}$,U, X), . . . ,Value(V$_{\mathbf{n}}$, X$_{\mathbf{n}}$,U, X),

C(V1), . . . ,C(Vn), Value(V1, Y1,W, Y), . . . ,Value(V$_{\mathbf{n}}$, Y$_{\mathbf{n}}$,W, Y), TupleID(U, X, S),

AttrIRI(X$_{\mathbf{1}}$, . . . , X$_{\mathbf{n}}$, X, P),TupleID(W, Y ,O)

Recall that attribute ID is a foreign key in relation STUDENT that references the attribute ID in relation PERSON. Then from the facts Value (1, RNO, t1, STUDENT) and Value (111, ID, t3, PERSON), the following triple is generated:

Triple
(http://www.ucsy.edu.mm/STUDENT#RNO=1,
http://www.ucsy.edu.mm/STUDENT#ID,
http://www.ucsy.edu.mm/PERSON#ID=111)

That mapping does not depend on the schema of the database: it defines a general mapping of *any* relational database structure into RDF; only a base URI has to be specified for the database, everything else is generated automatically.

5. Conclusion and Future Work

In this paper, we propose the solution to tackle the blank node problem in RDF model by assigning a URI to blank node and present the formalization rules for direct mapping of relational data to RDF data. Our main aim is to be easily accessible relational data content on Semantic web. This is the initial approach for transforming relational data and schema items into Semantic Web representation. During the mapping process, we take relational schema and database instances as input and generate an RDF document as output. We adopt this mapping as such a mapping from database schema directly to RDF is much faster, cheaper and usually more suitable for generation of Semantic Web content.

In future, we intend to extend our mapping process to handle more complex mapping.

References

[1] "RDF Primer", W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[2] A.Marcelo, "Data Exchange in the Relational and RDF Worlds", Pontificia Universidad Cat´olica de Chile.

[3] B.Kate, "Relational Database to RDF Translation in the Cultural Heritage Domain", School of Informatics, University of Edinburgh, http://www.ltg.ed.ac.uk/

[4] Cristian P´erez de Laborda and Stefan Conrad, "Database to Semantic Web Mapping using RDF Query Languages ",Institute of Computer Science, Heinrich-Heine-Universit¨at D¨usseldorf,D-40225 D¨usseldorf,Germany

[5] F.S.Juan, H.T.Syed, "Direct Mapping SQL Databases to the Semantic Web: A Survey".

[6] K.Madhab,"Retaining Semantics in Relational Databases by Mapping them to RDF", Netaji Subhas Institute of Technology, Delhi, India

[7] M.Alejandro, A.Marcelo, H.Aidan, P.Axel, " On Blank Nodes", Department of Computer Science, Pontificia Universidad Católica de Chile, Chile

[8] S. Martin, J.Ivan, "Two Layer Mapping from Database to RDF", Czech Technical University, Prague, Karlovo namesti 13, 121 35 Praha 2, Czech republic

[9] S.H.Tirmizi, J.Sequeda, D.Miranker, " Translating SQL Applications to the Semantic Web", Department of Computer Sciences, The University of Texas at Austin, USA

[10] S.Leo, "Cool URIs for the Semantic Web", February 2007

[11] T.Leon, S.Andreea, "Relational Databases and Resource Description Framework", Volume LIV, Number 2, 2009