Dynamic Transform Method for Ontology DB from Semi-structured Datasets

Gui-hyun Baek, Kee-hong Ahn, Su-kyoung Kim
Hanbat University, Republic of Korea

Abstract

This study is a framework to transform optimized ontology database from semi-structured datasets (such as spreadsheet, JSON, XML, CSV, TSV etc.) based RDM (Relational Data Model). The most important thing in big data analysis ability, is a classification and association between domain according to a context or meaning of sentence. In IoT environment, to increase a semantic interaction between virtual or physical objects, between an objects in different domain has an ability to associate by context or meaning. This ability is impossible using ontology technology based ontology schema model. Most semi-structured dataset has a problem to make a connection or interaction between objects (data) because homonym or synonym of linguistic feature. Therefore, studies to implement an ontology from existing RDB is in progress, as a result, typically, R2RML was published by W3C RDB2RDF Working Group. But, existing a mapping language can be limited, when RDB is not normalized (such as, 1NF), or when a semi-structured data provided (e.g. spreadsheet, text of key-value type (such as JSON), text of XML type etc.). Therefore, we need a transform method that construct an ontology database from semi-structured datasets for more accurate and speedy. So, we suggest a method that consist of three function block, first a Cell-Value Importer (CVI) and second, a Transformation Table Generator (TTG), and last a Property Expression (PropertyExp) for dynamically importing value at semi-structured data and describing mapping information by intuitive and concise sentence. Lastly, we tested that our design satisfied 10 of all 14 possible features of mapping languages [1] through testing by that transform ontology database from research equipment semi-structured dataset, after we implement prototype of a framework according to JDK Environment.

1. Introduction

Many platforms of Web 2.0 supply an Open API to use a data in those platforms. But, most platform has been designing data model that dependent on specific service through silo domain, and then they has been providing disconnected information based on RDM for which it has been specified relationship between individuals by indirectly connecting primary key and foreign key of entity. Therefore, these information has deficient connectivity. Also, complexity of relationship between objects increase as growing complexity of information, and then query of RDB has been performing complicated task because of this has been performing lot of JOIN operation, in the domain. This being so, development of service required query that is implicit enhanced function has been difficult, but it had been generating various data within various domain such as social media, IoT environment and big data. Additional, exiting platform environment demand lot of cost for reconstructing normalized schema, when service or data model had changed (e.g. change of many-to-many relationship of between entities.)

In order to solve this problems, it continually had been progressing studies about reconstruction existing RDB into metadata of ontologies. If concept hierarchy and relation hierarchy in ontologies exploit to design data model, then reconstructed metadata will redefine standardized naming based on integrated domain, and sharing and reusing of data will facilitate through Linked Data paradigm.

We propose methods that consist of a TTG and CVI, for which the method dynamic transform optimized ontology database from a non-normalized RDB or a semi-structured dataset, and more studied that describe mapping information by intuitive and concise sentence named a Property Expression (PropertyExp). And to prove of a accuracy in our method and algorithm, tested it and we can get a accurate result. Finally, in JDK Environment, we implement prototype of this framework using our method.

2. Related Work

All Many researcher suggested method of converting normalized RDB to RDF recommended R2RML [2] and Direct Mapping [3] about RDB-to-RDF translation in W3C RDB2RDF Working Group. Accordingly, Tools based R2RML or based non-R2RML was surveyed by [1] that motivated to
obtain information in deep web by Linked Data or to integrate heterogeneous information.

On the other hand, method of transforming many kinds of semi-structured dataset to RDF is non-standard, but its tools provide open library, and then, these tools listed “W3C Wiki ConverterToRDF” page. These activities showed that representation RDF from specific semi-structured dataset, yet these tools wasn’t been standard level. We research that method transform optimized ontology database according to semantic web from much used operating data among various semi-structured data. This paper use definition of ontology model defined in [4], and use relation algebra for operating TT (Transformation Table) and mapping information. But most method not satisfied above problem, nowadays.

3. Proposed Method

We can see Fig. 1 that process of broad flow which is dynamic transformation ontology database in four part divided. Exceeding semi-structured data -spreadsheet, delimiter-separated values(DSV) - such as, CSV, TSV-, result set of executing SQL from non-normalized RDB as well as normalized RDB - and a type using Open API (such as, JSON, XML) to RDF triple, we aim optimized constructing ontology KB by following process.

1) Extracted target column set and row set which was columns set(name or index set), name set of target object, sub-element name of target element, header name’s set and Attributes of ResultSet from semi-structured dataset, in order to construct transformation table.
2) Construct TT (Transformation Table symbol is T) from semi-structured dataset of each other format by Transformation Table Generator.
3) Describe to mapping information in mapping document by PropertyExp, in order to map a extracted column to properties.
4) Transform ontology database from row set of transformation table by using mapping information in Mapping document that describe in Mapping document.

3.1. Modeling of Figure 1’s Process

In order to perform the processes in section 3, this section define model and operators used each process. First, transformation dataset D was defined by following (1).

$$D = (T, M)$$  \hspace{1cm} (1)

As above (1), D is transformation dataset for constructing ontology KB from semi-structured dataset and construct set of TT and the M (mapping document) about this set. M is set of mapping information. Mapping information contain pair of column and mapping description. T in (1) was defined by following (2)-(4).

$$T(\text{column}_1, \text{column}_2, \ldots, \text{column}_n) = \{t_i\} \hspace{1cm} (0 \leq i < |t|)$$  \hspace{1cm} (2)

$$t_i = \{\text{row}_1, \text{row}_2, \ldots, \text{row}_m\} \hspace{1cm} (3)$$

$$\text{row} = (\text{value}_1, \text{value}_2, \ldots, \text{value}_n) \hspace{1cm} (4)$$

In (2) and (3), T is set of TT $t_i$ that had element of columns. Also, T is same concept from relation of RDB, use relation algebra. As above (4), rows of $t_i$ corresponds to columns that belong to itself of $t_i$. Actually, this rows contains cell that had value.

We define following (5)-(8) that cited by [5] about ontology model definition.

$$\alpha_c : \forall a \forall c (a_c \in R \land c \in C) \land t_a(a_c) \subseteq t_C(c) \times c$$  \hspace{1cm} (5)

$$p : p \in (R \cup A)$$  \hspace{1cm} (6)
In (5), \( \alpha_c \) means relation identifier between instance of class and this class. In the \( \alpha_c \), \( c \) is this class of element in set \( C \), \( C \) is set of class that defined in ontology, and this called concept identifiers. In (6), \( p \) is element within set of union of \( R \) and \( A \). \( R \) is set of object property, and this called relation identifiers. \( C \) is set of class that defined in ontology, and this called attribute identifiers.

For constructing optimized ontology KB, following (7), (8) extends definition of ontology KB in [4]. The (7) defines graph of class that called class graph. Class graph is unit that stored instances of specific class and relations of these instance. This regard single space as KB defined in [4]. And then the (8) redefines the pairs of the classes in ontology and the class graphs defined (7). Notations of (7) was described by following.

\[
cg = (I, t_R, t_A) \quad (7)
\]

\[
KB(c, cg) = ((c_i, cg_i)), 0 \leq i \leq |C| \quad (8)
\]

- \( I \) : set of instance of class of specific class graph
- A function \( t_R \) : for all \( r \in R \), this function get a set of ordered pair of instance within subset of set I and other instance, which these can connect \( r \), and call a relation instantiation.
- A function \( t_A \) : for all \( a \in A \), this function get a set of ordered pair of instance within subset of set I and specific datatype, which these can connect \( a \), and call an attribute instantiation.

### 3.2. Transforming Algorithm

The Figure 2 describes algorithm of function TRANSFORM for which construct ontology DB using those model and definition defined in section 4.

```sql
TRANSFORM(D) -> KB
function TRANSFORM(D)
KB := create empty KB;
T := D.T;
M := D.M;
for (int t := 0; t < SIZE(T); t++)
    t := T.t;
for (int i := 0; i < SIZE(T); i++)
    row := t[i];
for (int k := 0; k < SIZE(M); k++)
    m := M[k];
column := m.column;
c := m.c;
p := m.p;
cg := \alpha_{cg} (\pi_{cg} (KB))
if (cg == null) {
    cg := create empty cg;
    add cg to KB(c, cg)
} else {
    a := \pi_{TYPE(m,c)} (row);
    if (p \in A) {
        add (a, \pi_{column}(row)) to cg.t_A(p);
    } else {
        add (a, \pi_{column}(row)) to cg.t_R(p);
    }
} output KB
end
```

**Figure 1. Definition of Transform Algorithm**

In order to perform the processes in section 3, this section define function TYPE in Figure 2 is algorithm for which get column of TT that corresponded relation between class of ontology and generated instance in specific set of mapping information, show following Figure 3.

```sql
function TYPE(M, c) -> column about a_c
output \pi_{column} (\sigma_{c=c_p \land p=a_p} (M))
end
```

**Figure 2. Definition of TYPE Algorithm**

### 4. Design and Implement of Method

#### 4.1. TTG (Transformation Table Generator)

The TTG generates TT for constructing ontology DB from semi-structured datasets of each other format. As Fig. 1, transformation table construct column set and row set. Each row of row set construct cell that corresponds to each column of column set. Cell exist value that corresponds to specified column from the row. The following described the methods that are exploited for generating TT to semi-structured datasets of each other format.

- Exploiting XPath can generate transformation table to XML format.
- Exploiting JSON Path can generate transformation table to JSON format.
- Exploiting names or index of header can generate separated value format, such as CSV, TSV.
- Exploiting name or index of column can generate transformation table to spreadsheet format.
Consequentially, after defining the schema of TT for semi-structured dataset, this algorithm is defined in Figure 2, which can perform when describe mapping information by PropertyExp that will be described in section 5.

4.2. CVI (Cell-Value Importer)

Some values in specific row within TT for non-normalized RDB and semi-structured dataset are non-atomic data or raw data that required processing data through to find rule of value, immediately TT was generated by TTG, this TT can’t is impossible to construct an ontology KB. Therefore, in order to solve above it, we design that CVI is in charge of importing some value of cell in specified row of transformation table from programing level, and it can customize a procedure for processing non-atomic data or raw data that come up above. Thus, application based this framework enable to convert RDF triple from this data that had finite rule, by extending CVI. Following Figure 4 and Table 1 is shown to implement a Java interface CVImporter for CVI.

The Java interface CVImporter basically implement abstract method for function of importing value - such as, string, integer, long, double, Boolean, date string, time string, date-time string, etc. - is contained by specific cell within TT or for function of determining what value in cell is blank. In particular, the abstract method customizedImport enable that it is stored to the ontology KB to directly implement procedure of converting a RDF triples from various data by developer according to the need.

4.3. Mapping CVI (Cell-Value Importer)

The M (Mapping document) describes a column of TT generated by TTG and Mapping information, as (1). TT generated by TTG has common columns. In (5), we detailed prototype of M that used JSON format. The ‘datasetPath’ item is defined location of semi-structured dataset. The ‘numberingColumn’ item is assigned column can be ID of TT. Also, this item is in charge of importing value while value in cell within sequent row that corresponded to assigned column is null, in order to generate TT from semi-structured dataset, “mapping” item is defined these mapping information has “column”, “required”, and “property” item, “column” item is column of TT. The “required” item determine that can be allowed null in cell within rows corresponding to column. If a “required” item in specific this mapping information has “true”, cell within rows corresponding to column doesn’t allowed to contain null. The “property” item is defined PropertyExp that can define element of ontology corresponding to cell.

![Figure 3. Realization of CVI by Java Interface](image)

![Figure 4. Prototype of M using JSON Format](image)

![Table 1. Description of Abstract Method in CVI](image)
Mapper construct ontology KB using M. Mapper request to generate TT about these file from TTG, after this mapper is loaded files within path of semi-structured dataset in “datasetPath” item in M as Figure 5. Subsequently, this mapper imports value of cell within row in TT generated by TTG, corresponding to column in mapping information in M. And then, mapper interprets expression within “property” item in M, according PropertyExp description forms, such as Figure 7. Finally, this mapper converts RDF triple from TT, and then constructs ontology KB. This constructed ontology KB combined other ontology KB that was constructed by other mapper.

4.4. Syntax of a PropertyExp

PropertyExp is Non-R2RML, is intuitive and concise expression for mapping description that correspond column of transformation table to properties of ontology schema. This is design for which one of column correspond to many properties of ontology. Following Figure 6 describe syntax and its elements for PropertyExp. Among elements of PropertyExp, {qname_of_s_class}, {property}, {qname_of_o_class} and {datatype} is described QName (Qualified Name).

Items of Figure 7 are PropertyExp description forms using converted method that convert a value in cell within row in TT to RDF triples.

5. Optimization of Ontology DB

5.1. Configuration of Ontology DB Environment

This section describes method for distributing graph that is storing a converted triples and is contained by ontology KB. Ontology KB construct instance that is materialized class defined on ontology model. At this moment, a named graph of triples about instance that is materialized class is graph of selection by ontology engineer. This method exists drawback that selecting named graph when create a SPARQL statement by developer is ambiguous and non-unitive. In order to solve above it, triples of instance must be regularly and univestore specified a named graph when construct an ontology KB. We were able to solve this problem by using that following simple rules of domains defined by Time Berners-Lee and a PropertyExp:

- **row-to-resource**: each row of a table is a resource, i.e. an individual which class is represented by the table. The resource URI is formed using the primary key: "namespace/database/table/primaryKey" or "namespace/database/table#primaryKey"

Above this rule is that all resources of RDF concatenate in front of namespace when convert data contained RDB to RDF i.e.:

```
"namespace/database/table"
"namespace/database/table#"
```
First, we define namespace of class which a string is concatenated \{qname_of_s_class\} or \{qname_of_o_class\} in PropertyExp described in all of M and "/" or "#". And then, if the namespace of class concatenate all instance of class, all instance have prefix as namespace of class. Also, such instances can be concatenated prefix corresponding to this namespace of class. We define prefix of class such as this prefix, and define graph of class which IRI of named graph assigned this prefix. Consequently, when TT convert RDF triples corresponding to M, it can be distinguished a local name of all instance from other classes.

5.2. Ontology DB Rule Processing and Searching

This section describes two algorithms, one is for optimization structure of Ontology DB repository, another is searching instance quickly. The Figure 8 and Figure 9 introduce the algorithm.

Figures 10 and 11 are algorithms that composed RDF repository to satisfy sub-section 3.1. This algorithm loads the schema graph, and then composes the datasets, and then loading instance data files, or resort instance data.

6. Test and Conclusion

We defined model, algorithm and major component in order to construct ontology KB from semi-structured dataset. And then, we described how construct optimized ontology KB.

Table II. Result for Test Using the Semi-Structured Dataset within Equipment Domain at NFEC

<table>
<thead>
<tr>
<th>Mapping Property</th>
<th>Count of TT</th>
<th>Count of Original Cell in Row</th>
<th>Rate of Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>rdfs:label</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasImageURL</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasName</td>
<td>109,590</td>
<td>109,590</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasTakeDate</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasTakePrice</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasWGS_X</td>
<td>54,707</td>
<td>54,707</td>
<td>100.0%</td>
</tr>
<tr>
<td>hasWGS_Y</td>
<td>54,707</td>
<td>54,707</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedForm</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedModel</td>
<td>54,861</td>
<td>54,861</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedProject</td>
<td>43,952</td>
<td>43,952</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedStatus</td>
<td>54,868</td>
<td>54,868</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedUseScope</td>
<td>54,852</td>
<td>54,852</td>
<td>100.0%</td>
</tr>
<tr>
<td>relatedUseType</td>
<td>54,847</td>
<td>54,847</td>
<td>100.0%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>810,592</td>
<td>810,592</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
Finally, in order to construct ontology KB from semi-structured dataset, we propose structural method more than existing method. This framework becomes flexible because CVI support implementation of method for converts various data. Also, Optimization for ontology KB grows inference performance, because of possible to finding triples of instance in a named graph.

In the Future, our framework required research for method of inserting real-time data generated by IoT and big data environment from ontology KB, and required what is achieved growing performance for architecture of existing Triple store using our method.

7. References


