IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2018, accepted August 7, 2018, date of publication August 24, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2867152

Mapping Object-Oriented Database Models Into RDF(S)

QIANG TONG
School of Software, Northeastern University, Shenyang 110819, China
e-mail: tongq@swc.neu.edu.cn

The work was supported by the National Natural Science Foundation of China under Grant 61672139.

ABSTRACT Resource description framework RDF and RDF schema (collectively called RDF(S)) are the normative languages to describe the Web resource information. How to construct RDF(S) from the existing data sources is becoming an important research issue. In currently, much application information is stored in the object-oriented database, and how to construct RDF(S) from the existing object-oriented database becomes an important issue to be solved in the context of the semantic Web. By comparing and analyzing the characteristics of an object-oriented database and RDF(S), this paper proposes an approach for mapping object-oriented database models into RDF(S). After giving the formal definitions of object-oriented database and RDF(S), a mapping approach from object-oriented database models to RDF(S) is proposed. Further, a mapping example is provided to show that the approach is feasible, and the analyses and discussions about the approach are done. Finally, the correspondence between object-oriented database query language and RDF query language SPARQL is given.

INDEX TERMS Object-oriented database models, RDF, RDF schema, mapping.

I. INTRODUCTION

The resource description framework (RDF) and its schema RDF Schema (collectively referred to as RDF (S)) are standard language proposed by the World Wide Web Consortium W3C to express the Web resource information and their semantics. It is the basis for knowledge processing in the Semantic Web [1], [2]. How to build RDF(S) from existing information sources becomes an important research issue in the field of Semantic Web. To this end, many researchers have come up with methods of constructing RDF(S) from different data sources (e.g., relational databases [3]–[6], UML [7], XML [8]–[10], Semi-structured documents [11], and spreadsheets [12]).

With the development of object-oriented technology, data in many fields of application is stored in object-oriented databases [13]–[15]. Currently, there have been some efforts devoted to the research of Semantic Web and object-oriented technologies [16], [17], [23]. As the normative language description information of the Semantic Web, RDF(S) is the basis of the Semantic Web to realize knowledge sharing and management. How to build RDF(S) from the object-oriented database model has become an important research issue in the Semantic Web. Mapping object-oriented database models into RDF(S) will help the Semantic Web applications to access and process domain information and reuse existing domain information. On the other hand, it can also use the Semantic Web technology to solve some problems in modeling object-oriented database (for example, the inference of object-oriented modeling process can be achieved through the existing RDF(S) tool of the Semantic Web [20]).

At present, there are few studies on building RDF(S) based on object-oriented database model. The work in [18] lists the similarities and differences between RDF and object databases, discusses and unifies different approaches for querying and navigating object-oriented and RDF graphs and builds a simple formal model of an object-oriented database on top of model-theoretic RDF semantics. The work in [19] presents Layered RDF, which is an object-oriented approach to web information representation, it provides more powerful schema mechanisms than does RDF Schema, by incorporating object-oriented features such as inheritance and polymorphism. The work in [17] proposes a transformation and storage method for RDF data into an object-oriented model. This method generates corresponding object names based on *<rdf: type>*, and treats the property that serves as a predicate in an RDF triple as an attribute in an object-oriented class.

It can be found from the existing research work that the work in [17] mainly discusses the storage of RDF in the

object-oriented database model. That is, the conversion from RDF to object-oriented database model is achieved, and the purpose is the opposite of this paper. The main purpose of this paper is to investigate how to construct RDF(S) from object-oriented database model, that is, map object-oriented database model into RDF(S), and then implement knowledge extraction and sharing of existing domain information.

For this purpose, this paper proposes a mapping method from object-oriented database models to RDF(S) by deeply comparing and analyzing the characteristics of object-oriented database model and RDF(S). Firstly, the formal definitions of object-oriented database models and RDF(S) are given. Secondly, the mapping method from object-oriented database models to RDF(S) is proposed, and the detailed mapping rules are given. Thirdly, the method is analyzed by combining the mapping examples and discussion to verify the feasibility of the method. Finally, the correspondence between object-oriented database query language and RDF query language SPARQL is given.

The remainder of this paper is organized as follows: Section 2 gives the formal definitions of object-oriented database models and RDF(S); Section 3 proposes a method for mapping object-oriented database models into RDF(S), including detailed mapping rules; Section 4 discusses and analyzes the feasibility of the method in combination with specific examples; Section 5 gives the idea of query transformation; Section 6 gives a summary and future research work.

II. FORMAL DEFINITIONS OF RDF(S) AND OBJECT-ORIENTED DATABASE MODEL

In order to establish the formal correspondence between the object-oriented database model (OODM) and the RDF(S), it is necessary to give their formalized representations. This section will give their formal definitions.

The object-oriented database model (OODM) [13]–[15] mainly includes concepts such as classes, attributes, class inheritance, and objects. Its formal representation is as follows.

Definition 1 (OODM): An OODM can be represented as a tuple (C, A, T, ISA, AGG, O), where:

- C is a collection of classes, A is a collection of properties, and T is a collection of property basic data types (e.g., string and integer). Each class $c \in C$ defines the properties of this class, namely $c = [\ldots, a : t, a' : c', \ldots]$, where $a, a' \in A$, $t \in T$, $c' \in C$.
- $ISA \subseteq c_1 \times c_2$ is the relationship defined in the classes $c_1, c_2 \in C$ and is used to represent the inheritance relationship between subclass c_1 and superclass c_2.
- $AGG \subseteq c_1 \times c_2$ is the relationship defined in the classes $c_1, c_2 \in C$ and is used to represent the aggregation relationship between an aggregate class $c_1$and a constituent class c_2.
- O is a collection of objects. The object $o \in O$ is an instance of class $c \in C$, and each object has a unique object identifier.

The basic idea of the resource description framework RDF is [1]: all information is called ''resources'', and the described resources have some ''attributes'', and each of these attributes has its ''value'', and the value can be Literal value of literal type. The description of the resource can be performed through ''Statement''. The RDF uses the triple statement <subject, predicate, object> to describe the data. Specifically, the part of the statement used to identify the resource is called the ''subject,'' and the part used to distinguish the various attributes of the resource's subject (e.g., author and name) is called the ''predicate''. The part of the statement used to distinguish the value of an attribute is called the ''object''.

RDF Schema can define a set of word set that can be clearly described on the basis of RDF resources and can be used to describe the subclass/superclass relation (rdfs:subClassOf), subattribute/ superattribute (rdfs:subPropertyOf), domain (rdfs:domain) and range (rdfs:range) of attributes, and instance constraints of the class (rdf:type).

RDF and RDF Schema are often called RDF(S). A more detailed introduction to RDF(S) can be found in [1]. The formal definition of RDF(S) is recalled as in [7].

Definition 2 (RDF(S)): An RDF(S) model can be represented as a set of tuples (L, S), where:

- L is an identifier set, which specifically includes a class resource identifier set C, an attribute resource identifier set P, a data type identifier set D, and an individual instance resource identifier set I.
- S is a triple set of statements defined on the set of identifiers L.

The formal definitions of OODM and RDF(S) described above help to map OODM into RDF(S) that will be presented later.

III. MAPPING OBJECT-ORIENTED DATABASE MODELS INTO RDF(S)

This section presents a method for mapping object-oriented database model (OODM) into RDF(S) by deeply comparing and analyzing the characteristics of OODM and RDF(S).

Both the RDF(S) data model and the object-oriented data model are built on the basis of real-world entities, attributes, and relationships, and therefore have the possibility of conversion. There are many similarities between the RDF(S) data model and the object-oriented data model, as shown in Table 1.

As can be seen from the above table, there are similarities between the RDF(S) data model and the object-oriented data model, such as the concepts of classes and inheritance relations. These common points provide the possibility of mapping object-oriented data models into RDF(S).

Given an OODM (C, A, T, ISA, AGG, O), the mapped RDF(S) $= (L, S)$ can be obtained according to the rules.

Rule 1 (Data Type Mapping): Given the basic data type T in OODM, mapped to RDF(S) data type D, where RDF(S) uses the XML Schema data type [21]. Table 2 shows the mapping

TABLE 1. Comparison between RDF(S) and OODM.

RDF(S) data model	Object-oriented data model
Class	Class
Inheritance	Inheritance
Attributes with Literal attribute value type	Attributes of a class
XSD data types	Basic data types
Attributes with Resource attribute value type	Relationship between two classes
Instance	Object of a class

TABLE 2. Mapping of OODM datatypes to RDF(S) datatypes.

OODM datatypes	RDF(S) datatypes
string	xsd:string
smallint	xsd:short
integer	xsd:integer
decimal	xsd:decimal
float	xsd:float
time	xsd:Time
date	xsd:Date
boolean	xsd:boolean
...	...

rules for some basic data types in OODM to RDF(S) data types. The other types can be similarly processed.

Rule 2 (Identifier Mapping): Given identifiers C, A, T, O in the OODM, a corresponding RDF(S) resource identifier is created for each identifier.

The RDF(S) resource identifier is uniquely processed by naming a non-unique OODM element.

Rule 3 (Class Mapping): Given a class $c \in C$ in OODM, create an RDF(S) class: <rdfs:Class rdf:ID="c"/>.

Rule 4 (Attribute Mapping of Class): Given the attribute constraint $c = [\ldots, a : t, a' : c', \ldots]$ of the class $c \in C$, where $a, a' \in A$, $t \in T$, $c' \in C$, then create corresponding RDF(S) attributes for attributes a and a' respectively <rdf:Property rdf:ID = "a"> ... </rdf:Property> and <rdf:Property rdf:ID = " a' "> ... </rdf:Property>, and restrict the domain of the attribute a to the corresponding RDF(S) class of the class c, the range to the corresponding RDF(S) data type of t, the domain of the attribute a' to the corresponding RDF(S) class of the class c, and the range is the corresponding RDF(S) class of class c'.

For example, given an OODM class and its corresponding attribute *Professor* = [*name: string, supervise: Student*], the mapped RDF(S) according to the above rules contains the following classes and attributes:

<rdfs:Class rdf:ID = "Professor"/>
<rdfs:Class rdf:ID = "Student"/>
<rdf:Property rdf:ID = "name" >
<rdfs:domain rdf:resource= "#Professor"/>
<rdfs:range rdf:resource = "&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID = "supervise">
<rdfs:domain rdf:resource= "#Professor"/>
<rdfs:range rdf:resource = "#Student"/>
</rdf:Property>

Rule 5 (Class Inheritance Mapping): Given an inheritance ISA $\subseteq c_1 \times c_2$ which is defined in classes c_1, $c_2 \in C$ in OODM, and then create the appropriate RDF(S) subclass/superclass constraint <rdfs:Class rdf:ID = "c_1"><rdfs:subClassOf rdf:resource = "#c_2"/></rdfs: Class>.

For example, given an OODM class Inheritance ISA $\subseteq$ *Undergraduate_Student* $\times$ *Student* (this inheritance class represents the Undergraduate_Student is a subclass of Student), and the RDF(S) built according to the rule 5 contains the following subclass/superclass constraints:

<rdfs:Class rdf:ID = "Student"/>
<rdfs:Class rdf:ID = "Undergraduate_Student">
<rdfs:subClassOf rdf:resource="#Student"/>
</rdfs:Class>

Rule 6 (Aggregation Mapping): Given an aggregation AGG $\subseteq c_1 \times c_2$ which is defined in classes $c_1, c_2 \in C$ in OODM, and then create a property named p_{agg} and the constrains of the property as follows:

<rdfs:Class rdf:ID = "c_1"/>
<rdfs:Class rdf:ID = "c_2"/>
<rdf:Property rdf:ID = "p_{agg}">
<rdfs:domain rdf:resource = "#c_1"/>
<rdfs:range rdf:resource = "#c_2"/>
</rdf:Property>

Rule 7 (Object Instance Mapping): Given the object instance $o \in O$ of class $c \in C$ in OODM, and create the corresponding RDF(S) constraint <rdf:Description rdf:ID = "o"><rdf:type rdf:resource = "#c"/></rdf:Description>.

Rule 8 (Attribute Value Mapping of Object Instances): Given an object instance of class $c \in C = [\ldots, a : t, a'$: $c' \ldots]$ in OODM, $o = [\ldots, a : v, a'$: $o' \ldots]$, where a, $a' \in A$, $t \in T$, $c' \in C$, v is the value of the object instance o on the attribute a, o' is the object instance of the class $c' \in C$. Based on the Rule 7, we further increase the constraint <rdf:Description rdf:ID = "o"> <rdf:type rdf:resource = "#c"/>< a rdf:datatype="t"> v </a > < a' rdf:resource = "#o'"/></rdf:Description>.

For example, given a class in OODM *Professor* = [*name*: *string*, *supervise*: *Student*] and its corresponding object instance p_1 = [*name*: *John*, *supervise*: s_1], where s_1 is an object instance of class *Student*. Then, the RDF(S) contains the following classes and attributes:

<rdf:Description rdf:ID = "s_1">
<rdf:type rdf:resource = "#*Student*"/>
</rdf:Description>

<rdf:Description rdf:ID = "p_1">
<rdf:type rdf:resource = "#*Professor*"/>
<*name* rdf:datatype="*&xsd:string*"
>*John*</*name*>
<*supervise* rdf:resource = "#s_1"/>
</rdf:Description>

To further explain the mapping rules above and illustrate the feasibility of the method, we provide a mapping example as will be shown in the next section.

IV. EXAMPLE AND DISCUSSIONS

In order to illustrate the proposed method well, Fig. 1 shows an object-oriented database model associated with a college sports department, including partial class declarations and object instance declarations, where *Staff*, *Faculty*, *Department*, *Course*, *Student* are classes; f_1, d_1, c_1, s_1 are corresponding object instances and *Faculty* is a subclass of *Staff*.

According to the proposed mapping rules, we can construct the RDF(S) as shown in Fig. 2, where the RDF(S) resource namespace is omitted.

```
Class Staff {                    Class Department {
        sid: string;                     dname: string;
        }                                }
Class Faculty ISA Staff {
        name: string;
        work_in: Department;
        teach: Course;
        supervise: Student;
        }
Class Course {                   Class Student {
        cid: string;                     sname: string;
        cname: string;                   grade: string;
        credit: integer;                 }
        }
Object d1 belong_to Department {
    dname: sports;
    }
Object c1 belong_to Course {
    cid: 20170005; cname: skating; credit: 2;
    }
Object s1 belong_to Student {
    sname: John; grade: 3;
    }
Object f1 belong_to Faculty {
    name:David; work_in:d1; teach:c1; supervise:s1;
    }
```

FIGURE 1. An OODM modeling part of the sport department in a university.

Based on the above mentioned mapping rules and building examples, we can see that Fig. 1 above shows a common object-oriented database model, OODM, which basically includes the main elements of the object-oriented database model. Further, the conversion from OODM to RDF(S) can be achieved by using the mapping rules proposed in this paper, and it can be seen from Fig. 2 that the RDF(S) obtained after mapping can represent classes, attributes, object instances, and class inheritance in OODM.

The RDF(S) mapped from the OODM in Fig. 1 is shown as follows:
L is a set of identifiers, including:
C = {Staff, Department, Faculty, Course, Student} //a set of class resource indentifier
P = {sid, dname, name, work_in, teach, supervise, cid, cname, credit, sname, grade} //a set of attribute resource identifier
D = {xsd:string, xsd:integer} //a set of data type
$I = \{f_1, d_1, c_1, s_1\}$ //a set of individual instance
S is a set of triples defined on *L*, including:

```
{<rdfs:Class rdf:ID = "Staff"/>
 <rdf:Property rdf:ID = "sid">
     <rdfs:domain rdf:resource = "#Staff"/>
     <rdfs:range rdf:resource = "&xsd:string"/>
 </rdf:Property>
 <rdfs:Class rdf:ID = "Department"/>
 <rdf:Property rdf:ID = "dname">
     <rdfs:domain rdf:resource = "#Department"/>
     <rdfs:range rdf:resource = "&xsd:string"/>
 </rdf:Property>
 <rdfs:Class rdf:ID = "Course"/>
 <rdf:Property rdf:ID = "cid">
     <rdfs:domain rdf:resource = "#Course"/>
     <rdfs:range rdf:resource = "&xsd:string"/>
 </rdf:Property>
 ……
 <rdfs:Class rdf:ID = "Student"/>
 <rdf:Property rdf:ID = "sname">
     <rdfs:domain rdf:resource = "#Student"/>
     <rdfs:range rdf:resource = "&xsd:string"/>
 </rdf:Property>
 ……
 <rdfs:Class rdf:ID = "Faculty">
    <rdfs:subClassOf rdf:resource="#Staff"/>
 </rdfs:Class>
 <rdf:Property rdf:ID = "name">
     <rdfs:domain rdf:resource = "#Faculty"/>
     <rdfs:range rdf:resource = "&xsd:string"/>
 </rdf:Property>
 <rdf:Property rdf:ID = "work_in">
     <rdfs:domain rdf:resource = "#Faculty"/>
     <rdfs:range rdf:resource = "#Department"/>
 </rdf:Property>
 ……
 <rdf:Description rdf:ID = "d1">
     <rdf:type rdf:resource = "#Department"/>
     <dname rdf:datatype="&xsd:string">sports</dname>
 </rdf:Description>
 …
 <rdf:Description rdf:ID = "f1">
     <rdf:type rdf:resource = "#Faculty"/>
     <name rdf:datatype="&xsd:string">David</name>
     <work_in rdf:resource = "#d1"/>
     <teach rdf:resource = "#c1"/>
     <supervise rdf:resource = "#s1"/>
 </rdf:Description>
```

FIGURE 2. The RDF(S) mapped from the OODM in Fig. 1.

The relationship can further illustrate the feasibility of mapping OODM into RDF(S) proposed in this paper.

It should be noted that due to the limitations of the expressiveness of RDF(S) itself, some semantic information in some

object-oriented database models cannot be directly mapped to RDF(S) during the construction process. For example, an object in an OODM can contain properties of the object as well as methods of the object. Since RDF(S) cannot represent the method invocation process between classes, the methods contained in the OODM classes cannot be represented during the mapping process. In addition, in some applications, the ''disjointness'' constraint of the class inheritance relationship in the object-oriented database model may be considered, i.e., the multiple subclasses in the defined inheritance relation are disjoint. Since RDF(S) cannot represent disjoint constraints, the disjoint constraints in OODM cannot be directly represented in the mapping process, and can only be achieved through the use of RDF(S) extension language (e.g., ontology language OWL) or RDF(S) by adding some special attributes. For example, an OODM has a disjoint inherited ISA relation (Required_course × Optional_course) × Course, where Course is a superclass and the subclasses Required_course and Optional_course are disjoint. During the mapping process, the disjoint constraint described above can be expressed by adding the special name attribute ''P_disjointness'' to RDF(S), and the following RDF(S) triples are obtained:

<rdfs:Class rdf:ID = ''*Course*''/>
    <rdfs:Class rdf:ID = ''*Required_course*''>
    <rdfs:subClassOf rdf:resource = ''#*Course*''/>
</rdfs:Class>
<rdfs:Class rdf:ID = ''*Optional_course*''>
    <rdfs:subClassOf rdf:resource = ''#*Course*''/>
</rdfs:Class>
<rdf:Property rdf:ID = ''*P_disjointness*''>
    <rdfs:domain rdf:resource=''#*Required_course*''/>
    <rdfs:range rdf:resource = ''#*Optional_course*''/>
</rdf:Property>

From the rules, examples and analysis discussed above, we can see that the proposed method can map the object-oriented database models into RDF(S). The proposed method also considers the object-oriented database model as much as possible. In the future work, in order to further verify the feasibility and effectiveness of the proposed method, the corresponding automatic tools will be further implemented.

V. QUERY CONVERSION

The query language that supports the object-oriented database db4o mainly includes QBE (Query by Example), NQ (Native Queries), and SODA (Simple Object Database Access) [13]–[15]. The standard query language for RDF(S) is SPARQL [1]. A SPARQL query generally consists of five parts, namely declaration, query form and result set, data set, graph schema, and result modification:

Declaration: PREFIX foaf: <http://xmlns.com/foaf/0.1>
Result set: SELECT ?name
Data set: FROM <http://example.org/foaf/aliceFoaf>
Graph pattern: WHERE {?x foaf:name ?name.}
Result modification: ORDER BY ?name

The SPARQL query is based on graph pattern matching. The simplest graph pattern is the basic graph pattern. It is expressed in the form of a triplet and contains the variable to be queried.

The basic graph pattern is a finite set of triple graph patterns. Table 3 below shows the correspondence between object-oriented database query language and SPARQL. The SPARQL triple pattern schema included in the WHERE constraint corresponds to the object-oriented database db4o query language, which mainly analyzes the contents of each triplet $< S, P, O >$, depending on whether the item is a constant or a variable and an item. The values give different treatment strategies. In this paper, the basic graph patterns $< S, P, O >$ are divided into the following cases according to whether the item is a variable (as shown in the following Table 3, where the apostrophe denotes that the item is a variable), and the query is expressed according to the triple map pattern. In the sense of object-oriented thinking, the correspondence between query forms based on object-oriented database and SPARQL is given.

TABLE 3. The correspondences between SPARQL triple patterns and OODB queries.

S	*P*	*O*	SPARQL Triple pattern	Object-oriented database query form
×	√	√	Query resources that have *P* attribute and value of *O*, or resources that have *P* relationship with resource *O*	Query objects that have an attribute or relationship *P* with a value of *O*
×	×	√	Query all resources related to value or resource *O*	Query all *O* objects, or *O* related objects
×	√	×	Query all resources with *P* attributes	Query all objects and attribute values with *P* attributes, or query all objects with *P* relationships
×	×	×	Query all resources	Query all objects in the database
√	×	×	Query all information about *S* resources	Query all information and relations of the object *S*
√	×	√	Query the relationship between resource *S* and resource *O*	Query the relationship between the object *S* and *O*, or the attribute of the value *O*
√	√	×	Query the value of the *P* attribute of the resource *S* or the resource of the *P* relationship	Query the value of the *P* member of the object *S*, or information about the *S* in the object attribute *P*

As a query example, e.g., querying the student whose grade is 3 in the OODM, which can be expressed by the mapped SPARQL query language:

SELECT ?x
WHERE {{?x rdf:type student.},
    {?x grade ''3''^^xsd:integer.} }

In our future work we will further investigate the query conversion in depth.

VI. CONCLUSION

This paper compares and analyzes the characteristics of the object-oriented database model OODM and RDF(S), proposes a RDF(S) construction method based on OODM, and realizes the mapping from OODM to RDF(S). Based on the formal definition of OODM and RDF(S), the mapping method from OODM to RDF(S) is proposed, and the detailed rules are given. At the same time, the feasibility of the mapping method is explained with the corresponding examples, and the method is analyzed and discussed briefly. Finally, the correspondence between object-oriented database query language and RDF query language SPARQL is given. The work of this paper will help the Semantic Web application to access and process domain information and to reuse and share existing domain information.

Further work includes: designing and implementing a corresponding automated mapping prototype system, selecting more object-oriented database models in the real-world application fields for experiments, further verifying the validity of the proposed method, and studying the mapping at query level in depth. Also, in our future work we will make an attempt to deal with the large scale object-oriented and RDF data by combining with some machine learning methods [22].

REFERENCES

[1] (2014). *RDF 1.1 Primer, W3C Working Group*. [Online]. Available: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/
[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic Web," *Sci. Amer.*, vol. 284, no. 5, pp. 34–43, 2001.
[3] J. F. Sequeda, M. Arenas, and D. P. Miranker, "On directly mapping relational databases to RDF and OWL," in *Proc. 21st World Wide Web Conf. (WWW)*, 2012, pp. 649–658.
[4] W. Y. Mallede, F. Marir, and V. T. Vassilev, "Algorithms for mapping RDB schema to RDF for facilitating access to deep Web," in *Proc. 1st Int. Conf. Building Exploring Web Based Environ. (WEB)*, 2013, pp. 32–41.
[5] M. Korotkiy and J. L. Top, "From relational data to RDFS models," in *Proc. ICWE*, 2004, pp. 430–434.
[6] M. Krishna, "Retaining semantics in relational databases by mapping them to RDF," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI)*, Dec. 2006, pp. 303–306.
[7] Q. Tong, F. Zhang, and J. Cheng, "Construction of RDF(S) from UML class diagrams," *J. Comput. Inf. Technol.*, vol. 22, no. 4, pp. 237–250, 2014.
[8] P. T. T. Thuy, Y.-K. Lee, S. Lee, and B.-S. Jeong, "Transforming valid XML documents into RDF via RDF schema," in *Proc. Int. Conf. Next Gener. Web Services Pract. (NWESP)*, Oct. 2007, pp. 35–40.
[9] M. Klein, "Interpreting XML documents via an RDF schema ontology," in *Proc. 13th Database Expert Syst. Appl. (DEXA)*, Sep. 2002, pp. 889–894.
[10] B. H. Kumar and M. S. P. Babu, "Study and constructing RDF model for a well formatted valid XML document," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 7, pp. 648–652, 2013.
[11] F. Amato, A. Mazzeo, A. Penta, and A. Picariello, "Building RDF ontologies from semi-structured legal documents," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst. (CISIS)*, 2008, pp. 997–1002.
[12] L. Han, T. Finin, C. Parr, J. Sachs, and A. Joshi, "RDF123: From spreadsheets to RDF," in *Proc. 7th Int. Semantic Web Conf. (ISWC)*, 2008, pp. 451–466.
[13] R. G. G. Cattell *et al.*, *The Object Data Standard: ODMG 3.0*. San Mateo, CA, USA: Morgan Kaufmann, 2000.
[14] S. Bagui, "Achievements and weaknesses of object-oriented databases," *J. Object Technol.*, vol. 2, no. 4, pp. 29–41, 2003.
[15] R. McHaney and C. Hagmann, "Object-oriented database," in *Encyclopedia of Library and Information Science*. 2003.
[16] H. Knublauch, D. Oberle, P. Tetlow, and E. Wallace, "A semantic Web primer for object-oriented software developers," W3C Work. Group Note, 2006.
[17] E. Oren, B. Heitmann, and S. Decker, "ActiveRDF: Embedding semantic Web data into object-oriented languages," *J. Web Semantics*, vol. 6, no. 3, pp. 191–202, 2008.
[18] J. Güttner, "Object database on top of the semantic Web," in *Proc. WI/IAT Workshop Appl., Products Services Web-Based Support Syst.*. Halifax, CA, USA, 2003, pp. 97–102.
[19] T. Vølstad, F. Liu, and S.-U. Guan, "Layered RDF: An object-oriented approach to Web information representation," *Web Intell. Agent Syst., Int. J.*, vol. 7, no. 3, pp. 281–301, 2009.
[20] V. Vidya and S. C. Punitha, "A survey on ontology tools," *Int. J. Sci. Eng. Res.*, vol. 3, no. 10, pp. 1–8, 2012.
[21] *XML Schema Part 2: Datatypes*. [Online]. Available: http://www.w3.org/TR/xmlschema-2/
[22] X. Luo *et al.*, "Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1945–1966, 2018.
[23] F. Michel, L. Djimenou, C. F. Zucker, and J. Montagnat, "xR2RML: Relational and non-relational databases to RDF mapping language," Ph.D. dissertation, CNRS, 2017.

QIANG TONG received the Ph.D. degree from Northeastern University, China. He is currently with the School of Software, Northeastern University. His research interests include RDF data management. His research work is published in several international journals such as the *Journal of Intelligent & Fuzzy Systems* and *Journal of Computing and Information Technology* (CIT).