RESEARCH ON CONSTRUCTION OF RDF WITH HBASE

Hui Hu

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

ABSTRACT

Resource Description Framework (RDF) is designed as a standard metadata model for data interchange on the Internet. Because of machine comprehensibility, it has been successfully used in many areas, such as the intelligent processing of numerous data. While the generation of RDF with relational database (RDB) receives much attention, little effort has been put into the automatic construction of RDF with HBase due to its flexible data structure. Since more data is stored in HBase, it is necessary to extract useful information from HBase. In this paper, we are devoted to construction of RDF with HBase. We put forward formal definitions of RDF and HBase and propose our strategy for generating RDF with HBase. We develop a prototype system to create RDF, and test results demonstrate the feasibility of our method.

KEYWORDS

Semantic Web, RDF, HBase, Construction

1. INTRODUCTION

The vision of the Semantic Web is to establish a semantic framework that allows sharing and reusing data across applications, businesses, and communities [1]. As the core layer of the Semantic Web, RDF is the standard for data representation and exchange recommended by the World Wide Web Consortium. RDF is a metadata model with good machine readability because it adds semantics to the data representation when describing data on the web. RDF allows web applications to share, exchange, and integrate data without losing data semantics [2]. Because of the widespread use of RDF, large amounts of RDF are proliferating. For example, Best Buy and the New York Times use RDF for data storage [3]. But the efficient generation of RDF data remains an open problem. While there have been many efforts to convert relational databases, generating RDF from HBase, a NoSQL database, currently needs more attention.

It is critical to process data in a standard way dealing with massive data with heterogeneous data formats. Because RDF provides a standard, unified semantic manner for information processing, it is necessary to study generating RDF with these data. Numerous studies have been done on RDF generation, primarily concentrating on relational databases and XML. However, as an essential player in the era of Big Data, NoSQL databases have yet to receive much research. Consequently, extracting semantic information from NoSQL databases and constructing RDF is necessary. HBase is an integral part of the NoSQL database. According to Google research, many companies are beginning to choose HBase as their data store. It is an excellent idea to generate RDF from HBase for information processing.

David C. Wyld et al. (Eds): MLSC, ITCSS, ACSTY, SOFE, NATP, BDAB - 2023
pp. 167-178, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130213

This paper concentrates on constructing RDF with HBase. We give formal representations of HBase and RDF and propose an approach to convert HBase data to RDF based on these representations. And then, We developed an RDF construction program with the proposed transformation method and proved the feasibility of our approach through experiments.
The remaining portions of this essay are structured as follows. The relevant work of RDF construction is introduced in Section 2. Section 3 provides some preliminaries of RDF and HBase. Section 4 details the mapping rules of converting HBase to RDF. The designed system and the processing consequence are shown in Section 5. Section 6 presents the conclusion of this thesis and our future work.

2. RELATED WORK

Many methods exist to build RDF by extracting information from data with another format. This section presents various RDF construction approaches with databases, XML, and JSON.

Due to the widespread use of relational databases (RDB) in many fields, numerous attempts to build RDF using relational databases have been made. In [4], the authors proposed mapping rules and created RDF from MySQL tables. After specifying the mapping relationship between the RDB and RDF models, the authors in [5] proposed the mapping rules to generate RDF with RDB. A recent review [6] specified some tools for translating RDB into RDF. The authors assessed the capabilities of a total of 17 tools. The W3C recommends two methods for creating RDF from RDB. One is direct mapping, and the other is indirect mapping using a mapping language. The construction methods of RDF from RDB have primarily followed these two methods. Some efforts also exist to construct RDF with other database models, like MongoDB [7, 8].

The second is constructing RDF with XML. In [9], the authors propose XSPARQL language, which combines XQuery and SPARQL. With XSPARQL, people can query both XML and RDF and then implement data conversions between the two formats. The authors of [10] suggest using keywords or graphical queries to convert XML into RDF. In [11], the authors suggest a declarative method based on Scala programming language for converting XML to RDF. In [12], to convert XML to RDF, the authors created a template language based on XPath, which is helpful for users unfamiliar with XPath and RDF triples. For converting XML Schema to Shape Expressions (ShEx), an RDF validation language, some mappings from XML Schema to ShEx are proposed in [13].

JSON is utilized frequently because of its benefits of quick parsing and high transmission effectiveness. Therefore, the conversion of JSON data to RDF has received some attention. In [14], to map JSON to RDF, the authors identify the metadata of JSON and align it with domain vocabulary terms. With OWL, the authors of [15] propose the mapping from JSON to RDF and develop a mapping tool. The authors of [16], who focus on the output from SPARQL queries, convert JSON to JSON-LD, a compact RDF format. Concentrating on coverage data representing spatiotemporal data, the authors of [17] convert the data with JSON format to RDF. In [18], the authors encode coverage data in many formats, such as RDF and JSON. For representing RDF, they provide a mapping between the JSON and RDF via JSON-LD.

There have been few attempts to construct RDF using the HBase database in addition to the above. This paper formally defines HBase and RDF and concentrates on RDF generation from the HBase. Furthermore, our method differs from suggestions in [19] that only support the transformation from simple HBase to RDF.

3. PRELIMINARIES

In this section, according to the characteristics of RDF and HBase, we offer formal definitions about HBase and RDF. On the basis of the definitions, it is easy to describe the process of mapping from HBase to RDF.

3.1. Data Model of RDF

RDF is a domain-independent universal description language that does not define domain semantics. The RDF data model consists of a group of RDF statements, represented by the triple denoted as (subject, predicate, object). The triple's first element denotes the resource, the second is its property, and the third is the attribute value of the resource. RDF uses Universal Resource Identifier (URI) to denote resources.

Depending on the predicate type, the object can be a literal value or resource. To overcome the defect that RDF does not define domain semantics, people use the RDF Schema to define domain semantics. The RDF Schema defines a set of modeling primitives with fixed semantics, such as rdfs: domain [20].

Following is the formal definition of the RDF data model drawn on the research of [21].
Definition 1 (RDF data model): An RDF data model is defined as a 5-tuple: $RW = (V, E, \Sigma, L, A)$ where:

$V = \{V_1, V_2, V_3, ..., V_n\}$ is a finite vertex set in the RDF graph, which can be IRI, literals, or blank nodes.

$E = \{E_1, E_2, E_3, ..., E_n\} \subset V_i \times V_j, i \neq j$ is a finite edge set in the RDF graph.
$\Sigma = \{C, DP, OP, D, T\}$, $C = \{C_1, C_2, C_3, ..., C_n\}$ is finite set of RDF class resource tags, $DP = \{DP_1, DP_2, DP_3, ..., DP_n\}$ is finite set of RDF datatype property tags, $OP = \{OP_1, OP_2, OP_3, ..., OP_n\}$ is finite set of RDF object property tags, $D = \{D_1, D_2, D_3, ..., D_n\}$ is finite set of RDF data type tags, $T = \{T_1, T_2, T_3, ..., T_n\}$ is finite set of RDF instance tags.

$L = \{L_V, L_E\}$ is used to assign labels for vertices and edges of an RDF graph. $L_V : V \to \Sigma$ is used to assign labels for vertices, and $L_E : E \to \Sigma$ is used to assign labels for edges.

A is an axiom set containing class, attribute, and instance axioms, as shown in Table 1.

Table 1. Axiom set for A.

RDF triple	RDF axiom
$(L_V(V_i) \in \Sigma.C,\ rdf{:}type, owl{:}Class)$	$Type(L_V(V_i),\ Class)$
$(L_V(V_i) \in \Sigma.C,\ L_E(V_i \times V_j) \in \Sigma.OP,\ L_V(V_j) \in \Sigma.C)$	$ObjectProperty(L_E(V_i,V_j), domain(L_v(V_i)), range(L_V(V_j)))$
$(L_V(V_i) \in \Sigma.C,\ L_E(V_i \times V_j) \in \Sigma.DP,\ L_V(V_j) \in \Sigma.C)$	$DatatypeProperty(L_E(V_i,V_j), domain(L_V(V_i)),\ range(L_V(V_j)))$
$(L_V(V_i) \in \Sigma.T,\ rdf{:}type,\ L_V(V_j) \in \Sigma.C)$	$Individual(L_V(V_i),\ L_V(V_j))$

An RDF vertex $V_i \in V$ has a corresponding label denoted as $L_V(V_i)$ to indicate the subject or object. The vertex label can be IRI, literal or empty nodes. An edge $(V_i,V_j) \in E$ indicates a directed edge between two vertexes, and its label $L_E(V_i,V_j)$ denotes the triple's predicate.

There are three types of axioms in Table 1, which are class axioms, attribute axioms, and instance axioms. Let the RDF graph vertex $V_i,V_j \in V$, and its labels are $L_V(V_i), L_V(V_j)$. The vertex with class resource label means RDF Class concept. When edge $E_m = (V_i,V_j)$ exists, its label $L_E(V_i,V_j)$ indicates the RDF predicate that can be a datatype property or object property.

3.2. HBase Data Model

As an open-source, scalable, distributed database, HBase is a column-oriented database that differs from relational databases. We can consider the HBase table a multidimensional map indexed by row key, timestamp, and column.

Following content of this section introduce the formal representation of the HBase.
Definition 2 (HBase database model): HBase database model HM is represented as tuple (B, RE, HI), where:
$B = TN \cup CF \cup CQ \cup D$ is the set of essential elements of HBase.
$TN = \{TN_1,\ TN_2,\ TN_3,\ ...,\ TN_n\}$ is the set of HBase table names.
$CF = \{CF_1,\ CF_2,\ CF_3,\ ...,\ CF_n\}$ is the set of HBase column families. $cf(tn)$ specifies the column family in table tn.
$CQ = \{CQ_1,\ CQ_2,\ CQ_3,\ ...,\ CQ_n\}$ is the set of HBase column qualifiers. $cq(tn,\ cf)$ specifies the column qualifier of column family cf in table tn.
$C = \{C_1,\ C_2,\ C_3,\ ...,\ C_n\}$ is the set of HBase table columns specified by column family and qualifier. $c(tn)$ specifies the column of table tn.
D is a finite set of distinct HBase table column data type. $D(c)$ denotes data type of HBase column.
RE is a finite set of constraint relation defined by user about HBase database model.
$RE(TN_1,C,TN_2)$ denotes reference relation from table TN_1 to table TN_2 by column C.

$RE(TN_1, CF, TN_2)$ denotes reference relation from table TN_1 to table TN_2 by column family CF.

$RE(TN_1, CF)$ denotes embed relation in table TN_1 through column family CF. Users embed entity information into column families rather than storing it in tables.

HI is a set of HBase instances. An HBase instance is a 6-tuple containing a table name, row key, column family, column qualifier, cell value, and cell timestamp. For a HBase instance $hi \in HI$, $hi.tn$ suggests its table name, $hi.rk$ means its row key, $hi.cf$ denotes its column family, $hi.cq$ implies its column qualifier, $hi.v$ suggests its cell value and $hi.ts$ means its timestamp.

4. MAPPING HBASE DATABASE MODEL TO RDF

Here Function φ is used to map the elements of HBase to RDF.

Rule 1: $\forall tn \in TN \rightarrow L_V(V_i) = \varphi(tn) \in \Sigma.C \wedge (L_V(V_i),\ rdf{:}type, owl{:}Class)$

Rule 1 maps the HBase table to the RDF vertex with the class label. For example, a table named "employee" is mapped to RDF class <ns:employee>.

Rule 2: $\forall c \in C(t) \rightarrow \varphi(c) = L_E(V_i \times V_j) \in \Sigma.DP \wedge \varphi(t) = L_V(V_i) \in \Sigma.C.$

An HBase column in the HBase database model is mapped to an RDF edge with a datatype property label. Table columns in Table 2 such as "personal:name" and "office:phone" are mapped to RDF Datatype Property in Table 3.

Table 2. A sample data with HBase table employee.

Row Key	Time Stamp	Column Family: personal	Column Family: office
00001	timestamp1	personal:name="John"	office:phone="415-212-5544"
00001	timestamp1	personal:residence_phone="415-111-1111"	office:address="1021 Market St"
00001	timestamp2	personal:residence_phone="415-111-1234"	

Table 3. Mapping Result of Rule2.

<ns:personal:name> a owl:DatatypeProperty . <ns:personal:residence:phone> a owl:DatatypeProperty . <ns:office:phone> a owl:DatatypeProperty . <ns:office:address> a owl:DatatypeProperty .

Rule 3:

$$\forall tn1, tn2 \in TN, c \in c(tn1), RE(tn1, c, tn2) \rightarrow L_E(V_i \times V_j) = \varphi(c) \in \Sigma.OP \wedge L_V(V_i) = \varphi(tn1) \in \Sigma.C$$
$$\wedge L_V(V_j) = \varphi(tn2) \in \Sigma.C.$$

Rule 3 maps an HBase relation $RE(tn1, c, tn2)$ between table $tn1$ and table $tn2$ to an RDF edge with an object property label. For example, in Table 4, a user follows another user specified by the column value, then the column is mapped to Object Property as shown in Table 5.

Table 4. A sample data with table user.

Row Key	**Column Family: follows**		
AK	follows:1=foo	follows:2=bar	follows:3=baz
foo	follows:1=bar	follows:2=AK	

Table 5. Mapping Result of Rule 3.

<ns:user> a owl:Class .
<ns:follows:1> a owl:ObjectProperty .
<ns:follows:1> rdfs:domain <ns:user> .
<ns:follows:1> rdfs:range <ns:user> .
<ns:follows:2> a owl:ObjectProperty .
<ns:follows:2> rdfs:domain <ns:user> .
<ns:follows:2> rdfs:range <ns:user> .
<ns:follows:3> a owl:ObjectProperty .
<ns:follows:3> rdfs:domain <ns:user> .
<ns:follows:3> rdfs:range <ns:user>

Rule 4:

$\forall tn1, tn2 \in TN, cf \in cf(tn1), RE(tn1, cf, tn2) \rightarrow L_E(V_i \times V_j) = \varphi(cf) \in \Sigma.OP \wedge L_V(V_i) = \varphi(tn1) \in \Sigma.C \wedge L_V(V_j) = \varphi(tn2) \in \Sigma.C.$

An HBase relation $RE(tn1, cf, tn2)$ between table $tn1$ and table $tn2$ is mapped to the RDF edge with an object property label. For example, in Table 6, one user points to another user by column family qualifier, then the column family "follows" is mapped to the object property as shown in Table 7.

Table 6. A sample data with table user.

Row Key	**Column Family: follows**		
AK	follows:foo=1	follows:bar=1	follows:baz=1
foo	follows:bar=1	follows:AK=1	

Table 7. Mapping Result of Rule 4.

<ns:user> a owl:Class .
<ns:follows> a owl:ObjectProperty .
<ns:follows> rdfs:domain <ns:user> .
<ns:follows> rdfs:range <ns:user>

Rule 5:

$\forall tn1 \in TN, cf \in cf(tn1), RE(tn1, cf) \rightarrow L_V(V_i) = \varphi(tn1) \in \Sigma.C \wedge L_V(V_j) = \varphi(cf) \in \Sigma.C \wedge L_E(V_i \times V_j) = \varphi(ref - cf) \in \Sigma.OP.$

An HBase relation $RE(tn1, cf)$ is mapped to the RDF edge with an object property label. This relation means the entity is embedded in the table column family. For example, in Table 8, entity "department" is embedded in the column family "department", and then the column family is mapped to the RDF class as shown in Table 9.

Table 8. There are sample data of table student.

Row Key	Column Family: student	Column Family: department
001	student:name= “bob”	department:dno=5001
001	student:sex=“M”	department:dname=“Computer”
001		department:header=“Alice”

Table 9. Mapping Result of Rule 5.

<ns:student> a owl:Class .
<ns:department> a owl:Class .
<ns:ref-department> a owl:ObjectProperty .
<ns:ref-department> rdfs:domain <ns:student> .
<ns:ref-department> rdfs:range <ns:department> .
<ns:dname> a owl:DatatypeProperty .
<ns:dname> rdfs:domain <ns:department> .
<ns:dname> rdfs:range <xsd:string> .
<ns:header> a owl:DatatypeProperty .
<ns:header> rdfs:domain <ns:department> .
<ns:header> rdfs:range <xsd:string> .

Rule 6:

(1) $\forall hi \in HI \rightarrow L_V(V_i) = \varphi(hi.rk) \in \Sigma.T \wedge L_V(V_j) = \varphi(hi.v) \in \Sigma.D \wedge L_E(V_i \times V_j) = \varphi(hi.cf : hi.cq) \in \Sigma.DP$

(2) $\forall hi \in HI, RE(hi.tn, hi.cf : hi.cq, tn2) \rightarrow L_V(V_i) = \varphi(hi.rk) \in \Sigma.T \wedge L_V(V_j) = \varphi(hi.v) \in \Sigma.T \wedge L_E(V_i \times V_j) = \varphi(hi.cf : hi.cq) \in \Sigma.OP$

(3) $\forall hi \in HI, RE(hi.tn, hi.cf, tn2) \rightarrow L_V(V_i) = \varphi(hi.rk) \in \Sigma.T \wedge L_V(V_j) = \varphi(hi.cq) \in \Sigma.T \wedge L_E(V_i \times V_j) = \varphi(hi.cf) \in \Sigma.OP$

(4) $\forall hi \in HI, RE(hi.tn, hi.cf) \rightarrow L_V(V_i) = \varphi(hi.rk) \in \Sigma.T \wedge L_V(V_j) = \varphi(hi.cf) \in \Sigma.T \wedge L_E(V_i \times V_j) = \varphi(ref - cf) \in \Sigma.OP$ R

Rule 6 maps a table instance to an RDF instance based on the schema about classes and properties created by the rule 1 to rule 5. The (1) maps cell values to the RDF datatype properties from rule 2. The (2), (3), and (4) add values to the properties mapped from rule 3 to 5. For example, the table “employee” instance with row key “00001” in Table 2 is mapped to RDF, as shown in Table 10.

We all know that HBase instances contain not only values but also timestamps of the cell, so how to map the timestamp of HBase instances is a problem worth investigating. Much research has been done with temporal RDF modeling, such as [22, 23]. In this paper, to be compatible with traditional RDF, our method use the reification mechanism provided by RDF to represent the time of HBase. The reification mechanism uses the three predicates (rdf:subject, rdf:predicate, and rdf:object) provided by RDF to describe the three parts of the statement. Mapping the timestamp of an HBase instance is actually mapping the timestamp to the corresponding statement, as depicted in Table 10.

Table 10. Mapping result of Rule 6.

<ns:00001> a <ns:employee> .
<ns:00001> <ns:personal:name> “John” .
<ns:st1> a <rdf:Statement> .
<ns:st1> <rdf:subject> <ns:00001> .
<ns:st1> <rdf:predicate> <ns:personal:name> .
<ns:st1> <rdf:object> “John” .
<ns:st1> <ex:timestamp> timestamp1 .
<ns:00001> <ns:personal:residence_phone> “412-111-1111” .
<ns:00001> <ns:office:phone> “412-212-5544” .
<ns:00001> <ns:office:address> “1021 Market St” .
…

5. SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1. System Architecture

To validate our construction method of RDF with HBase, we developed a prototype system to extract the HBase data and map these data to RDF. We develop this system with OpenJDK 17.0.1 and JavaFX on a PC (Intel i5-5200U (4) @ 2.700GHz, RAM 8 GB, and ArchLinux system).

The system obtains data by connecting the database and then outputs RDF data. The database parsing, RDF construction, and display modules comprise most of the system, as depicted in Figure 1. Database parsing module analyses the semantic relationships and data information of data based on the HBase database formal definition. The RDF construction module constructs RDF Schema with obtained semantic relationships from the parsing module. And then, this module constructs RDF data with RDF Schema and data information by mapping rules introduced in Section 4. Finally, the display module exhibits the RDF data created by the RDF construction module.

Figure 2 displays the system’s screen snapshot. The GUI of the system contains three display areas. The TextArea on the left exhibits information about the HBase data model, including tables and columns. Moreover, the TextArea (in the upper right) below the label “RDF Schema” shows the corresponding RDF Concept, such as Classes and Properties, in the form of the RDF Turtle. The TextArea (in the lower right) below the label “RDF Individual” exhibits the corresponding RDF individuals.

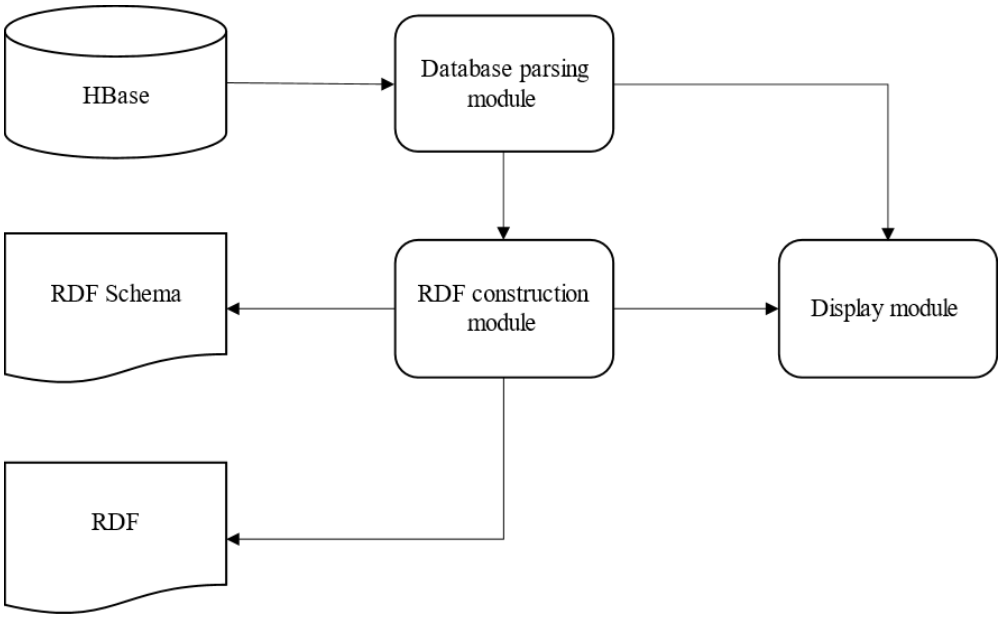

Figure 1. The architecture of System.

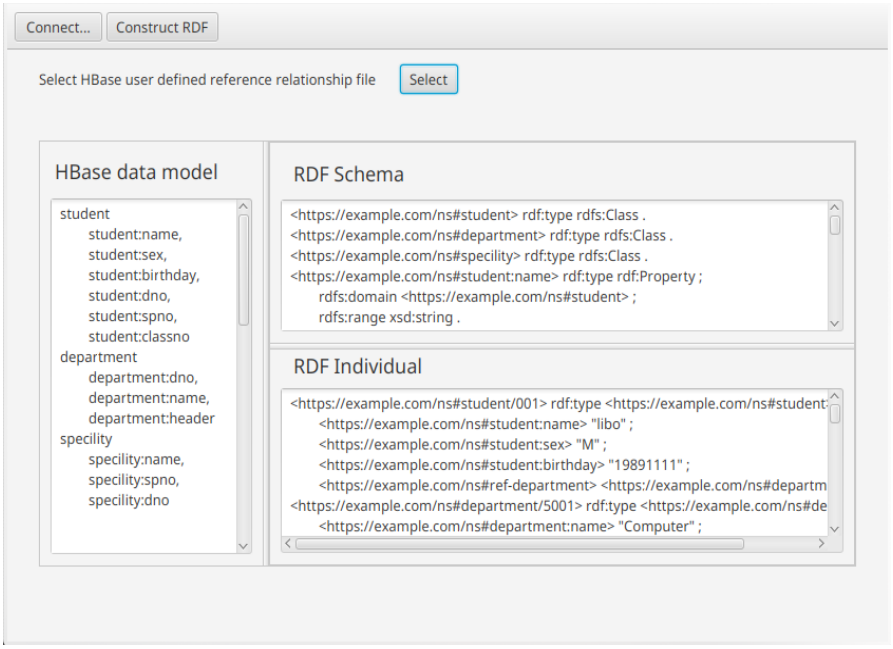

Figure 2. The screen snapshot of System.

5.2. Experimental Results and Discussions

We conducted experiments with the HBase data source from the article “Introduction to Hbase Schema Design” created by a Cloudera engineer. Since the paper [19] does not provide the source code of the system for building RDF from HBase data, this section simulates the implementation of the corresponding construction system according to the method given in the paper.

The results of the RDF construction experiments based on HBase data source are shown in Table 11. The first column of the table represents the test metrics of the RDF construction experiment, including the time consumed for construction, the number of RDF classes generated, the number of datatype properties, the number of object properties, the number of domain axioms, the number of range axioms, and the number of RDF triples generated. and the number of RDF triples generated. The second to third columns are the experimental results of the method in this paper (denoted as H2R) and the method proposed in the paper [19] (denoted as OBDI).

Table 11. Experimental results.

Test metrics	**H2R**	**OBDI**
Construction time (ms)	8934	10142
RDF classes	5	4
Datatype properties	9	16
Object properties	5	0
Domain axioms	14	0
Range axioms	14	0
RDF triples	680047	190020

Table 11 shows that our method can retain more database information for RDF creation compared to another method. And also, our method has advantages in terms of construction time. Our method constructs more RDF triples per unit time when compared with the other method. While paper [19] only considers the mapping of basic data attributes in HBase database, our approach not only considers the mapping of basic data attributes in HBase, but also pays attention to implicit reference relationships in HBase database, such as column-value based references, column-qualifier based references, and embedded references. Implicit reference relationships in the data are mapped to RDF by parsing user-defined implicit reference relationship documents. Meanwhile, our approach only traverses the HBase data source once to resolve the semantic relationships of HBase data, while the paper [10] traverses the data source twice to resolve the schema information and data information of the HBase data source. Therefore, our method not only retains more database information for constructing RDF compared with another method, but also has advantages in construction efficiency.

6. CONCLUSIONS AND FUTURE WORK

Because of the growth of the Web, more and more people select NoSQL databases to solve the data management problems that come from big data. HBase occupies a part of the market share in NoSQL databases. Hence, the study of transforming HBase data to RDF is conducive to data use and solving the problem of insufficient available RDF data. In this paper, we design a method to convert HBase semantic relationships and data to RDF and develop a system to confirm its feasibility.

In the future, optimizing the implementation system to improve efficiency is the first task. We will also pay attention to expanding the method for more column-oriented databases.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” Scientific american, vol. 284, no. 5, pp. 34–43, 2001.

[2] J. Hendler, T. Berners-Lee, and E. Miller, “Integrating applications on the semantic web,” Journal of the Institute of Electrical Engineers of Japan, vol. 122, pp. 676–680, 01 2002.

[3] Z. Ma, M. A. M. Capretz, and L. Yan, "Storing massive resource description framework (RDF) data: a survey," Knowl. Eng. Rev., vol. 31, no. 4, pp. 391–413, 2016.

[4] E. Byty ̧ci, L. Ahmedi, and G. Gashi, "RDF mapper: Easy conversion of relational databases to RDF," in Proceedings of the 14th International Conference on Web Information Systems and Technologies, WEBIST 2018, Seville, Spain, September 18-20, 2018 (M. J. Escalona, F. J. D. Mayo, T. A. Majchrzak, and V. Monfort, eds.), pp. 161–165, SciTePress, 2018.

[5] J. F. Sequeda, M. Arenas, and D. P. Miranker, "On directly mapping relational databases to RDF and OWL," in Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012 (A. Mille, F. Gandon, J. Misselis, M. Rabinovich, and S. Staab, eds.), pp. 649–658, ACM, 2012.

[6] F. Michel, J. Montagnat, and C. Faron Zucker, "A survey of RDB to RDF translation approaches and tools," research report, I3S, May 2014. ISRN I3S/RR 2013-04-FR 24 pages.

[7] H. Abbes and F. Gargouri, "Mongodb-based modular ontology building for big data integration," J. Data Semant., vol. 7, no. 1, pp. 1–27, 2018.

[8] N. Soussi and M. Bahaj, "Exploiting nosql document oriented data using semantic web tools," in International Conference on Advanced Intelligent Systems for Sustainable Development, pp. 110–117, Springer, 2018.

[9] S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres, "Mapping between RDF and XML with XSPARQL," J. Data Semant., vol. 1, no. 3, pp. 147–185, 2012.

[10] M. Kharrat, A. Jedidi, and F. Gargouri, "A semantic approach for transforming XML data to RDF triples," in 14th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2015, Las Vegas, NV, USA, June 28 - July 1, 2015 (T. Ito, Y. Kim, and N. Fukuta, eds.), pp. 285–290, IEEE Computer Society, 2015.

[11] J. P. McCrae and P. Cimiano, "LIXR: quick, succinct conversion of XML to RDF," in Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016 (T. Kawamura and H. Paulheim, eds.), vol. 1690 of CEUR Workshop Proceedings, CEUR-WS.org, 2016.

[12] J. Huang, C. Lange, and S. Auer, "Streaming transformation of XML to RDF using xpath-based mappings," in Proceedings of the 11th International Conference on Semantic Systems, SEMANTiCS 2015, Vienna, Austria, September 15-17, 2015 (A. Polleres, T. Pellegrini, S. Hellmann, and J. X. Parreira, eds.), pp. 129–136, ACM, 2015.

[13] H. Garc ́ıa-Gonz ́alez and J. E. L. Gayo, "Xmlschema2shex: Converting XML validation to RDF validation," Semantic Web, vol. 11, no. 2, pp. 235–253, 2020.

[14] F. Freire, C. Freire, and D. Souza, "Enhancing JSON to RDF data conversion with entity type recognition," in Proceedings of the 13th International Conference on Web Information Systems and Technologies, WEBIST 2017, Porto, Portugal, April 25-27, 2017 (T. A. Majchrzak, P. Traverso, K. Krempels, and V. Monfort, eds.), pp. 97–106, SciTePress, 2017.

[15] S. J. R. M ́endez, A. Haller, P. G. Omran, J. Wright, and K. Taylor, "J2RM: an ontology-based json-to-rdf mapping tool," in Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC) (K. L. Taylor, R. S. Gon ̧calves, F. L ́ecu ́e, and J. Yan, eds.), vol. 2721 of CEUR Workshop Proceedings, pp. 368–373, CEUR-WS.org, 2020.

[16] P. Lisena and R. Troncy, "Transforming the JSON output of SPARQL queries for linked data clients," in Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018 (P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, eds.), pp. 775–780, ACM, 2018.

[17] J. D. Blower and M. Riechert, "Coverages, JSON-LD and RDF data cubes," in Proceedings of the Workshop on Spatial Data on the Web (SDW 2016) co-located with The 9th International Conference on Geographic Information Science (GIScience 2016), Montreal, Canada, September 27-30, 2016 (K. Janowicz, J. Lieberman, K. Taylor, G. McKenzie, S. Gao, S. J. D. Cox, and E. Parsons, eds.), vol. 1777 of CEUR Workshop Proceedings, pp. 9–16, CEUR-WS.org, 2016.

[18] P. Baumann, E. Hirschorn, J. Mas ́o-Pau, V. Merticariu, and D. Misev, "All in one: Encoding spatio-temporal big data in xml, json, and RDF without information loss," in 2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017 (J. Nie, Z. Obradovic, T. Suzumura, R. Ghosh, R. Nambiar, C. Wang, H. Zang, R. Baeza-Yates, X. Hu, J. Kepner, A. Cuzzocrea, J. Tang, and M. Toyoda, eds.), pp. 3406–3415, IEEE Computer Society, 2017.

[19] V. Kiran and R. Vijayakumar, “Ontology based data integration of nosql datastores,” in 2014 9th International Conference on Industrial and Information Systems (ICIIS), pp. 1–6, IEEE, 2014.
[20] G. Antoniou, P. Groth, F. van Harmelen, and R. Hoekstra, A Semantic Web Primer, 3rd Edition. MIT Press, 2012.
[21] T. Fan, L. Yan, and Z. Ma, “Mapping fuzzy RDF(S) into fuzzy object-oriented databases,” Int. J. Intell. Syst., vol. 34, no. 10, pp. 2607–2632, 2019.
[22] F. Zhang, Z. Li, D. Peng, and J. Cheng, “RDF for temporal data management - a survey,” Earth Sci. Informatics, vol. 14, no. 2, pp. 563–599, 2021.
[23] D. Yang and L. Yan, “Transforming XML to RDF(S) with temporal information,” J. Comput. Inf. Technol., vol. 26, no. 2, pp. 115–129, 2018.

AUTHORS

Hui Hu is currently a master’s candidate in the College of Computer Science and Technology at the Nanjing University of Aeronautics and Astronautics, China. His research interests include RDF and Semantic Web.