ICIC Express Letters
Part B: Applications
Volume **7**, Number **2**, February **2016**

ICIC International ©2016 ISSN 2185-2766
pp. 435–441

RESEARCH ON PARSE AND STORAGE OF COMPLEX LINKED DATA: TAKING DOI DATA AS AN EXAMPLE

Hui Liu and Yao Liu*

Information Technology Support Center
Institute of Scientific and Technical Information of China
No. 15, Fuxing Road, Haidian District, Beijing 100038, P. R. China
*Corresponding author: liuy@istic.ac.cn

Received July 2015; accepted September 2015

Abstract. *A large amount of complex linked data exists in fields of social network, e-commerce and resource retrieval. Complex linked data has large data size, includes multiple types of complex relationships and contains a wealth of information in data and relationships. On the one hand, storing complex linked data by relational database causes amount of data redundancy and difficulties in dynamic update. On the other hand, information of structures and relationships of data are hard to be stored and utilized. Based on tree structure and graph database, a method of parse and storage of complex linked data is proposed to realize the efficient storage and utilization of complex linked data. The experimental result shows that this method can store DOI data, process complex links of data effectively and solve problems which relational database causes.*
Keywords: Complex linked data, DOI, Graph database, Neo4j

1. **Introduction.** Complex linked data exists in various fields. The value of data can be utilized with efficient parse and storage. The feasible way is to store data in a relational database by converting unstructured data into structured form [1]. Because of restriction of data pattern, mapping schema should be provided to data conversion [2]. Two main methods are structure mapping and model mapping [3]. Schema of files like DTD or schema of XML in structure mapping should be mapped into relational schema which is used to reveal logical structures. Data was mapped into relational schema based on DTD of files in [4]. The graph of DTD and element was generated from the simplified DTD. Relational schema was generated by graphs. XML files which conform to the schema were converted into relational database. [5] proposed STORED method which realizes the effective management of complex linked data by combining technologies of database and semi-structure data. Model mapping makes models of XML files map into relational schema. [6] used directed graphs to represent XML files and provided Edge, Binary and Universal methods to store information edges. [7] proposed the XREL method which showed model structures of XML files by interval number. These methods need strict format of files and generate too many tables. Besides, data in the hierarchical form is not supported well by relational database and part of information is dropped in the mapping process.

There are also some problems in reality when we use relational database to store complex linked data. Institute of Scientific and Technical Information of China and Wanfang Data are responsible for development and operation work of Chinese DOI as registration agency [8]. There are three main problems in storing metadata of DOI by using relational database. Firstly, the cost of relational database is high. Secondly, huge amount of data is not processed well by relational database [9,10]. With the increasing register users and amount of data, the number of record in each table is multiplied. Records in relational database have reached 19 million. At present, all DOI numbers are stored in

a single table and resources which DOI numbers belong to are recognized by the field of DOI_TYPE. Due to huge tables, it is too inefficient to execute SQL queries. Thirdly, relational database cannot process complex links efficiently. Moreover, it is easy to create data redundancy and hard to update dynamically [11]. There are too many connections between tables. A query of a complete journal register record involves 12 tables. Adding some fields which realize the connections between tables causes data redundancy.

In this research we use the emerging graph database to replace relational databases. According to the characteristic of graph database, large-scale data is parsed by using tree structure as a mapping layer. Data in XML files and relational database is converted into the suitable form for the storage of graph database. The mapping process is simplified and structure information is reserved.

2. **Parse of Complex Linked Data.**

2.1. **Converting data into a tree.**

On the one hand, the format of records in the relational database should be converted into a right format. On the other hand, the format of data in XML files should be converted into a right format so that data of XML registration files which users submit can be stored in the graph database. Graph database adopts a graph model, so data can be converted into a tree structure which can store data and structure information of XML registration files. Different types of data have different tree structures. For example, science data and journal form different tree structures according to their characteristic and XML schema. In addition, the same type of data has different nodes, for example, journals have different styles of volume, issue, and numbers of authors.

2.1.1. *Converting records in relational database into a tree structure.* The relational database is used to store DOI register data. Data is written in the corresponding table by recognizing each tag in XML files. Field ID is created in every table to improve the efficiency of retrieval and realize efficient connection between tables such as Field ID in table DOI, field ARTICLE_ID in table ARTICLE, field ISSUE_ID in table issue, and field JOURNAL_ID in table JOURNAL. For example, the DOI registration data of journal involves 12 tables like table Article which enumerates all the metadata on the level of article such as primary key field ARTICLE_ID, foreign key field ISSUE_ID, key word field KEYWORD, abstract field ABSTRACT, table DOI which stores correlation information of DOI number such as primary key field ID, subordinate resource field PARENT_ID and DOI number field DOI. Field ISSUE_ID is used as connection between table ARTICLE and table ISSUE, field JOURNAL_ID is used as the connection between table DOI and table JOURNAL, and field PARENT_ID is used as the connection between table DOI and other tables. On the other hand, fields DOI_TYPE and PARENT_ID form a composite index and identify the resource which DOI numbers belong to.

When records are converted into tree structures, fields in each table which realize connections between tables can be omitted. Branch nodes of tree form structure and leaf nodes store values. Records in the relational database are traversed and stored into leaf nodes according to the structure of trees. Every complete record forms a tree. The structure information of tree refers to the XML schema.

2.1.2. *Converting XML files into a tree structure.* Journals, books, scientific data, components of journals and scientific data can be registered in Chinese DOI system [12]. XML files are created by users according to the XML schema and registered on Chinese DOI system. The schema defines the standard of XML files including required field and structure of data. For example, the DOI registration data of journal can be divided into two parts. One is head part which includes unique identification number of batch processing

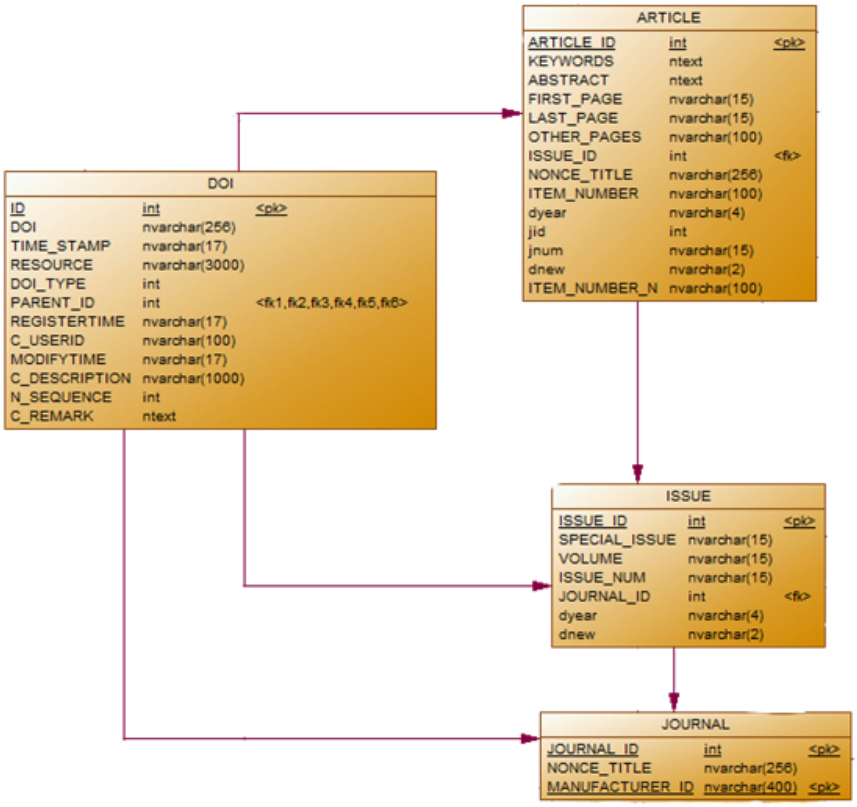

FIGURE 1. A partial entity relationship diagram of article DOI

```
<journal_metadata>
    <manufacturer_id>zhhljy</manufacturer_id>
    <full_title>Chinese Journal of Nursing Education</full_title>
    <issn>1672-9234</issn>
    <cn>11-5289/R</cn>
</journal_metadata>
<journal_issue>
    <journal_volume>
        <volume>08</volume>
    </journal_volume>
    <issue>12</issue>
</journal_issue>
<journal_article>
        <title>the level of professional attitude of male nursing
    <publication_date>
        <year>2011</year>
    </publication_date>
    <publisher_item>
        <item_number>zhhljy201112002</item_number>
    </publisher_item>
    <doi_data>
        <doi>10.3761/j.issn.1672-9234.2011.12.002</doi>
        <resource><![CDATA[http://doi.med.wanfangdata.com.cn/10.37
    </doi_data>
    <pages>
        <first_page>534</first_page>
        <last_page>536</last_page>
    </pages>
</journal_article>
```

FIGURE 2. A partial XML file of journal register data

file produced by publisher, timestamp of batch processing data and registrant etc. The other is body part which includes metadata of journal, issue, article and DOI data etc.

XML files can be converted into a tree structure easily. Labels of data elements are converted into names of nodes and properties of labels are converted into properties of nodes. Texts in the XML files can form leaf nodes as children nodes of labels, or as what we do in this paper, texts are regarded as properties of label nodes according to the character of graph database Neo4j. For example, node ISSN of journal metadata is stored as ISSN: { name: "ISSN", value:"1009-3419"}.

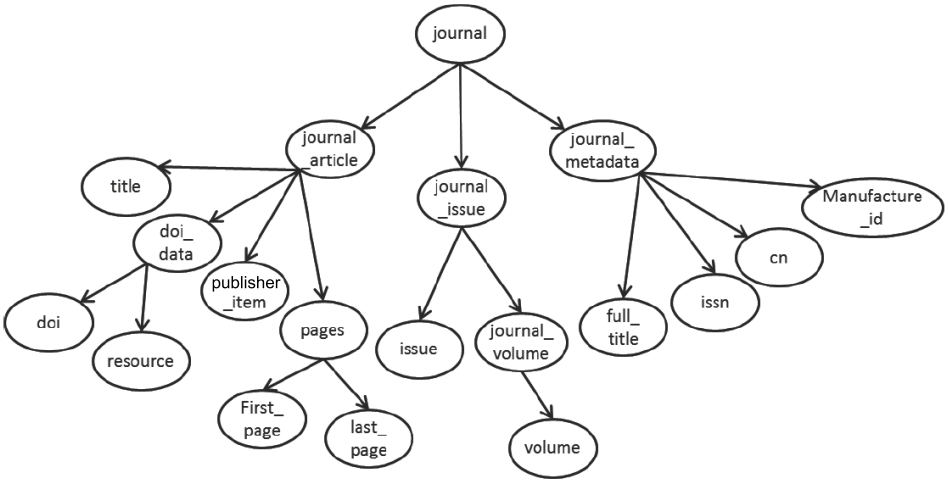

FIGURE 3. A partial tree structure of journal register data

2.2. **Converting tree structure into format GRAPHML and CSV.** Data can be imported into Neo4j by using its query language Cypher or different kinds of tools and plugins. We import data into Neo4j in format GRAPHML and CSV.

2.2.1. *Converting data into format CSV.* CSV files are imported into Neo4j by Cypher that is a declarative graph query language. Cypher is a relatively simple but powerful language. Very complicated database queries can be easily expressed through Cypher. We can put parent nodes into the first row, children nodes into the second row and relations into the third row.

2.2.2. *Converting data into format GRAPHML.* GRAPHML is an XML-based file format for graphs. It supports the entire range of possible graph structure constellations including directed, undirected, mixed graphs and application-specific attributes. Neo4j-shell-tools can be used to import and export GRAPHML files. GRAPHML of partial journal register data follows.

```
<graph id="1" edge default="undirected">
        <Node id="n1">
        <Data key="name">JOURNAL_METADATA<Data>
        </Node>
        <Node id="n2">
        <Data key="name">JOURNAL_ID<Data>
        <Data key="value"> 303<Data>
        </Node>
        <Node id="n3">
        <Data key="name"> FULL_TITLE<Data>
        <Data key="value"> Chinese Journal of Nursing Education <Data>
        </Node>
        <Edge id="e1" source="n1" target="n2">
        <Edge id="e1" source="n1" target="n3">
</graph>
```

3. Storage of Complex Linked Data.

3.1. **Storage and retrieval in graph database.** Relation model cannot satisfy the need of some fields with expansion of data size and augment of data complexity [13]. Using relational database brings about redundancy. Relation database cannot satisfy dynamism or support multilayer complex queries. Graph database can store data and execute multilayer complex queries efficiently. Neo4j can be regarded as assemblage of nodes and relations. Meanwhile, data is stored in nodes which are organized by relations. Properties of nodes and relationship are saved by a doubly linked-list of key-value and relationship of nodes is saved by a doubly linked-list. Node records contain only a pointer to their first property and their first relationship. Files of properties and relationship use arrays as the core storage structure.

In Neo4j, searching in graph can be done with traversal algorithm. The speed of graph traversal which has nothing to do with the size of graph is a constant. Indexes of nodes, relationship and properties can be created for searching a certain node or relationship. Nodes can be found rapidly by creating indexes rather than traversing the graph.

Every complete record forms a tree. The tree is formed by single branch of journal data and multiple branches of article data. Article data can be retrieved by traversing branches of article and journal data can be retrieved by traversing branches of journal. In addition, nodes in tree have different types such as DOI, and title. Searching some records like ten article DOI numbers of Chinese Journal of Nursing Education can be achieved easily by traversing node DOI.


```
{"value":"10.3761/j.issn.1672.9234.2014.03.001"}
{"value":"10.3761/j.issn.1672.9234.2014.03.002"}
{"value":"10.3761/j.issn.1672.9234.2014.03.003"}
{"value":"10.3761/j.issn.1672.9234.2014.03.004"}
{"value":"10.3761/j.issn.1672.9234.2014.03.005"}
{"value":"10.3761/j.issn.1672.9234.2014.03.006"}
{"value":"10.3761/j.issn.1672.9234.2014.03.007"}
{"value":"10.3761/j.issn.1672.9234.2014.03.008"}
{"value":"10.3761/j.issn.1672.9234.2014.03.009"}
{"value":"10.3761/j.issn.1672.9234.2014.03.010"}
```


FIGURE 4. Ten article DOI numbers of Chinese Journal of Nursing Education

3.2. **Query performance tests.** Query performance is tested on both relational database and graph database. We have chosen some common queries in DOI service such as getting some records in a single table, getting an integrated record from tables, getting all articles in the same issue, getting references of articles and getting citing literature of articles. We have set five levels including very slow, slow, medium, fast and very fast. These levels are comparative degree based on the gap between two databases.

The result of tests shows that relation database performs better than graph database Neo4j when it queries in a single table. Neo4j performs remarkably when data conforms to the structure of graph traversals. Queries are executed according to existing nodes and relations. Compared to the inefficient query that gets references and citing literature of articles on relational database, Neo4j just finds nodes of articles and traverses adjacent nodes along relations.

According to this method, ten million records of DOI data are parsed and stored. Data in relational database and XML files is saved completely. Complex links and structure information of data are stored efficiently.

TABLE 1. Query performance tests between relational database and graph database

	Slow	Very slow	Medium	Fast	Very fast
Records in a single table				Neo4j	SQL SERVER
Integrated records in tables				SQL SERVER	Neo4j
All articles in a same issue				SQL SERVER	Neo4j
References of articles			SQL SERVER		Neo4j
Citing literature of articles		SQL SERVER			Neo4j
References of articles' references	SQL SERVER				Neo4j

4. **Conclusions.** Efficient storage of DOI data and complex link between data are achieved by converting DOI data in relational database and XML files which users provide in tree structures and storing in a graph database. The problem of data redundancy and nonsupport of dynamic update is solved. DOI system is improved greatly and data is well utilized. In addition, it is proved that complex linked data and its internal relations can be stored, managed and updated efficiently by graph database. So structure and correlation information of data can be utilized efficiently and the value of complex linked data is reflected. Based on the current work, we will improve the performance of graph database according to characteristics of DOI data. In addition, applications of citation data will be developed based on graph database.

Acknowledgment. This work is partially supported by National Key Project of Scientific and Technical Supporting Programs No. 2013BAH21B02; the authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] T. Peng, L. Sun and H. Bao, Research of unstructured data transformation based on XML, *International Conference on Internet Technology and Applications*, pp.1-4, 2010.
[2] J. Freire and M. Benedikt, Managing XML data: An abridged overview, *Computing in Science & Engineering*, vol.6, no.4, pp.12-19, 2004.
[3] B. Yao, H. Zhu and L. Wang, Strategy of mapping XML document to relational database with Dewey encoding, *Computer Engineering and Applications*, vol.47, no.27, pp.128-131, 2012.
[4] J. Shanmugasundaram and K. Tufte, Relational databases for querying XML documents: Limitations and opportunities, *Proc. of the 25th Conference on Very Large Databases*, pp.302-314, 1999.
[5] A. Deutsch, M. Fernandez and D. Suciu, Storing semi-structured data with STORED, *Proc. of ACM SIGMOD Conference*, pp.431-442, 1999.
[6] D. Florescu and D. Kossmann, Storing and querying XML data using an RDBMS, *IEEE Data Engineering Bulletin*, vol.22, no.3, pp.27-34, 1999.
[7] M. Yoshikawa and T. Amagasa, XRel: A path-based approach to storage and retrieval of XML documents using relational databases, *ACM Trans. Interact Technology*, vol.1, no.1, pp.110-141, 2001.
[8] Y. Li, Y. Zhao, C. Yao and X. Guo, Review of projects R&D relevant to Chinese DOI – From plan to domestic & international cooperation, *Digital Library Forum*, no.8, pp.32-38, 2009.
[9] M. Rys, Scalable SQL, *Communications of the ACM*, vol.54, no.6, pp.48-53, 2011.
[10] Z. Sui, W. Kang and Y. Tian, Synchronously extracting instances and attributes for the concepts from the Web, *International Journal of Knowledge and Language Processing*, vol.3, no.3, pp.1-17, 2012.

[11] R. Angles and C. Gutierrez, Survey of graph database models, *ACM Computing Surveys*, vol.40, no.1, pp.1-6, 2008.
[12] Y. Zhao, Research on DOI of digital periodical resource in China, *Information Science*, vol.25, no.7, pp.1018-1021, 2007.
[13] Y. Liu, Z. Sui, Q. Zhao, Y. Hu and R. Wang, On automatic construction of medical ontology concept's description architecture, *International Journal of Innovative Computing, Information and Control*, vol.8, no.5(B), pp.3601-3616, 2012.