International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 14931-14937
© Research India Publications. http://www.ripublication.com

An Implementation of LOD Instance Development System using Schema-Instance Layer Separation

Heekyung Moon*, Zhanfang Zhao, Jintak Choi*** and Sungkook Han***

** Department of Computer Engimeering, College of Engineering, Wonkwang University, 460 Iksandae-ro, Iksan, Korea.*

*(*Corresponding Author)*

*** Department of Computer Engimeering, College of Engineering, Hebei GEO University, Huai An Road No. 136, Shijiazhuang, Hebei Provincei, China.*

**** Department of Computer Science & Engineering, College of Information Technology, Incheon National University, 119 Academy-ro, Yeonsu-gu, Incheon, Korea.*

Abstract

The research and development of Web of Data to realize the opening and sharing of diverse, heterogonous data on the Web has been actively accomplished. As a standard data model for this effort, Linked Open Data (LOD) based on ontology has been proposed. In general, LOD datasets consist of two elements: the ontology that defines the common vocabularies and the instances that represents the actual data of information resources. For the effective development of LOD datasets, it is required that the instance layer should be separated from ontology layer.

This paper proposes a new method to separate the instance layer from the ontology layer. A Schema-Instance Interface (SII) between ontology layer and instance layer is devised to separate and interconnect two layers. This papers also describes a LOD instance development system based on SII. This system can generate the application-specified instance development system automatically. The proposed system can be applied for the practical LOD datasets development in the diverse application areas since even novices without knowledge about domain ontology can develop LOD datasets.

Keywords: Linked Open Data (LOD), triple, instance, Resource Description Framework (RDF), RDF Schema

INTRODUCTION

According to the rising of the new information technologies such as Big Data, Internet of Things (IoT) and Deep Learning, the data becomes the crucial resource for the development of new information applications. At the same time, Web of Data to open and share the diverse data of heterogeneous types in the Internet has become the hot research topic. Ontology-based Linked Open Data (LOD) that allows computers to understand and process the data semantics has emerged to realize the goals of Web of Data. LOD becomes the powerful enabler by using the standard data model for both structured and unstructured information resources and a shared semantic representation on the Web [1, 2, 3].

The high qualitative LOD datasets are required to develop the diverse intelligent services on the Web. In the last decade the numerous best practices of LOD datasets, including DBpedia and YAGO, have been published by an increasing number of researchers, governments, public organizations and data providers [4, 5]. Many LOD development approaches have been also proposed such as ontology-based LOD generation, the translation of relational databases to LOD and the population of LOD from various information resources [6, 7, 8]. However, the conventional approaches suffer from their complexities and difficulties, and so the development of LOD datasets has made little progress due to the lack of appropriate, specialized methodologies and tools [8].

To copes with complexity in system design and achieve the required engineering quality factors, the separation of constituent elements into layers that each component has distinct functionality with as little overlap as possible is the common, basic principle in system design [8, 9, 10, 11]. In general, LOD datasets consist of two elements: ontologies that define the common vocabularies represented by the standard data model Resource Description Framework (RDF) and the instances that represent the actual data of information resources [1, 2, 12]. For the effective development of LOD datasets, it is required that the instance layer should be separated from ontology layer. This paper proposed a new layer separation method using Schema-Instance Interface (SII) that separates and mediate two layers.

This papers also describes an application-specific instance development system based on SII. The proposed system can generate an instance development system that can be customizable for the domain applications. This system can be applied for the practical LOD datasets development in the diverse application areas since even novices without knowledge about ontology can develop LOD datasets.

The rest of the paper is organized as follows. Section 2 reviews related work on the development and population of LOD datasets. Section 3 presents the principle and approach for separation of concerns. The details of Schema-Instance Interface (SII) that separate the instance layer from the upper ontology layer is described in the section. Section 5 shows the architecture of the instance development system based on SII and the typical use-case of the generated system. Section 5 concludes the paper and discusses possible future work.

RELATED WORK

The methods and approaches used in the development of LOD datasets can be categorized into three approaches [13]. 1) Since LOD is based on ontology, ontology development methods are generally applied for LOD development as it is. As these methods mix ontologies and instances together, it is difficult to focus on LOD instances. 2) Since the vast amount of useful data are still stored in relational databases (RDB), one of the most efficient ways to develop LOD datasets is to map data in RDB into RDF which is the standard data model of LOD. Due to the importance of mapping RDB to RDF (RDB2RDF), the multifold mapping approaches have been proposed. The most important step towards RDB2RDF is two standard recommendations by the W3C RDB2RDF Working Group: Direct Mapping and R2RML mapping language [14]. However, mapping description like R2RML is complicate to realize the complex table relations. 3) LOD population from the various information resources have been attempted. These methods using natural language processing are difficult, and the quality of data is poor to use in practical applications. Considering the particularity of LOD datasets, more effective and practical method is required to foster LOD applications.

LAYER STRUCTURE OF LOD INSTANCE DEVELOPMENT

In general, LOD datasets development usually focuses on the generation of the instances. So the development methodology specialized for the instance generation is significant. This section describes the principle and the approach to implement LOD instances development system.

Separation of Concerns and Layer Structure of LOD

The separation of concerns (SoC) is a system design principle for separating constituent components into layers that each component has the distinct functionality with as little overlap as possible [8]. SoC as the one of the foundational principle of system design is to reduce the complex problems into the independent core components. SoC can cope with complexity in system design, and achieve the required engineering quality factors.

The concept of SoC is widely applied to the diverse areas for the effective system design, especially, to software engineering. The core concepts of object-oriented design such as modularization, encapsulation and composition are originally based on SoC. In addition, software design pattern of Model–View–Controller (MVC) is also the typical example of SoC. SoC is the well-known approach for the effective system design and development.

In LOD datasets development, the instance generation is based on two standards: RDF that provides standard modeling framework to represent the data of heterogeneous information resources over the Web and RDF Schema to define the common shared vocabularies. RDF and RDF Schema defines domain ontology that represents the concepts and relationships in the domain knowledge. LOD instances are the actual individual of the concept classes defined by RDF and RDF Schema. Hence it is reasonable to separate LOD instance layer from ontology layer as shown Fig. 1. While the ontology layer model concepts and their relationships in the domain knowledge, the instance layer is a set of the actual RDF triples based on ontology layer.

In Semantic Web using the ontology language OWL, the separation of TBOX and ABOX is fundamentally important [15]. While TBOX represents the schema or taxonomy of the domain, ABOX describes the attributes of instances, the relationships between instances, and other assertions about instances according to the TBOX concepts. OWL also uses the concepts of layer separation. So the separation of the ontology layer and the instance layer is the primitive feature inherent to the development of LOD datasets.

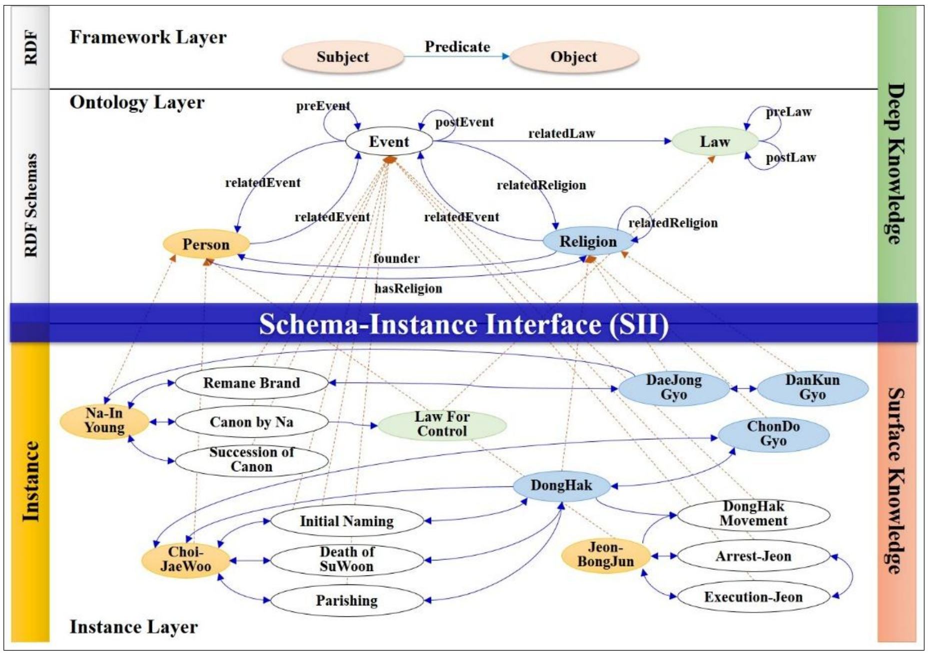


Figure 1: Layer structure of LOD datasets development.

3.2 Implementation of Layer Interface

The separated layers should be seamlessly interlinked to build the complete operational system structure. The interlink of the separated layers are usually achieved by means of defining an interface associating two layers. This paper proposes a Schema-Instance Interface (SII) to associate the instances with the ontologies defined by RDF Schema. SII is a specification to coordinate the semantic property of the instances with the domain ontology and it also defines the instance development environment. So SII plays two important roles in LOD datasets development. SII should provide the links to the class or property vocabularies in ontology layer for the instances. In addition, SII should be able to define LOD instance development environment to create the instances without knowledge about ontology layer.

As the interface specification of two separated layers, SII consists of the entries that define the link to the properties of ontology classes and the screen layout of the instance entry. Each entry has the following format:

$$Entry := entryElement, entryType, entryAttribute+ \qquad (3.1)$$

where Entry is the property of the class and becomes the predicate of the triple, entryType is the screen layout for the instance input and entryAttribute is the attributes of entryType, respectively.

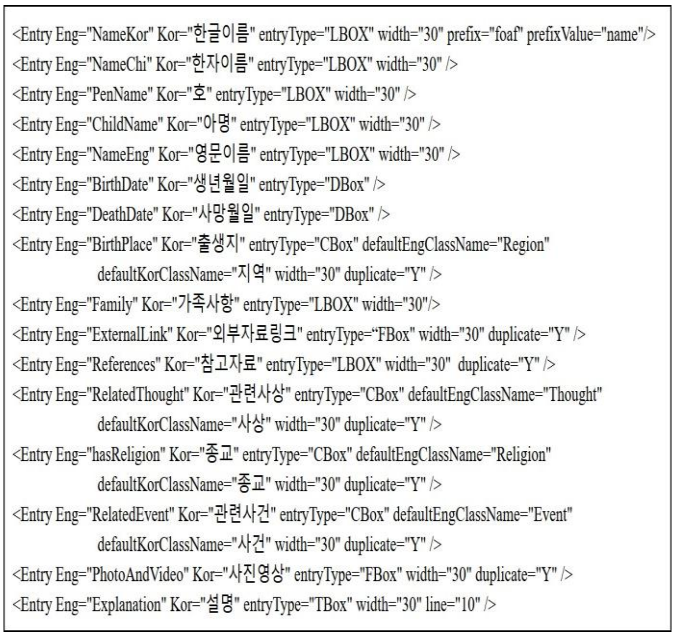
Table 1: Entry types define in SII

entryType	contents	entryAttr
LiteralBox (LBox)	Input multipurpose string	width
TextBox (TBox)	Input text, explanation, etc	width, line
ClassBox (CBox)	Set class of instance	defaultClassName, width
FileBox (FBox)	URI for the relevant information resources, documents, photos, images, videos, etc.	defaultFolder, width
NumberBox (NBox)	Input numeric data	width
DataBox (VBox)	Input date information	dateFormat
URIBox (UBox)	URI location of information resource	width
DateBox (DBox)	General data value information	datatype, width
FolderBox (GBox)	Folder location where information resources are stored	defaultFolder, width
SelectorBox (SBox)	Select possible data values	width, item

According to the value type of the property, entryType and entryAttribute is specified. Table 1 shows entryType and entryAttribute defined in SII by means of the datatypes in RDF and OWL. SII is customizable to create more specified development environment. Fig. 2 shows a typical example of SII document.

Some entries that can generate a customizable development environment shown in Table 1 is defined in SII.

SII can be constructed automatically or semi-automatically by the definition of the classes and their properties. The prominent feature of SII is to create an application-specific LOD instance development environment with domain ontology. SII also provides user-friendly tool similar to the traditional data entry system.



```
<Entry Eng="NameKor" Kor="한글이름" entryType="LBOX" width="30" prefix="foaf" prefixValue="name"/>
<Entry Eng="NameChi" Kor="한자이름" entryType="LBOX" width="30" />
<Entry Eng="PenName" Kor="호" entryType="LBOX" width="30" />
<Entry Eng="ChildName" Kor="아명" entryType="LBOX" width="30" />
<Entry Eng="NameEng" Kor="영문이름" entryType="LBOX" width="30" />
<Entry Eng="BirthDate" Kor="생년월일" entryType="DBox" />
<Entry Eng="DeathDate" Kor="사망월일" entryType="DBox" />
<Entry Eng="BirthPlace" Kor="출생지" entryType="CBox" defaultEngClassName="Region"
          defaultKorClassName="지역" width="30" duplicate="Y" />
<Entry Eng="Family" Kor="가족사항" entryType="LBOX" width="30"/>
<Entry Eng="ExternalLink" Kor="외부자료링크" entryType="FBox" width="30" duplicate="Y" />
<Entry Eng="References" Kor="참고자료" entryType="LBOX" width="30"  duplicate="Y" />
<Entry Eng="RelatedThought" Kor="관련사상" entryType="CBox" defaultEngClassName="Thought"
          defaultKorClassName="사상" width="30" duplicate="Y" />
<Entry Eng="hasReligion" Kor="종교" entryType="CBox" defaultEngClassName="Religion"
          defaultKorClassName="종교" width="30" duplicate="Y" />
<Entry Eng="RelatedEvent" Kor="관련사건" entryType="CBox" defaultEngClassName="Event"
          defaultKorClassName="사건" width="30" duplicate="Y" />
<Entry Eng="PhotoAndVideo" Kor="사진영상" entryType="FBox" width="30" duplicate="Y" />
<Entry Eng="Explanation" Kor="설명" entryType="TBox" width="30" line="10" />
```

Figure 2: Example of the generated SII document.

LOD INSTANCE DEVELOPMENT SYSTEM WITH LAYER SEPARATION

The schema-instance layer separation proposes a new approach to develop the LOD instances. This section describes the architecture of LOD instance development system using SII and some algorithms for the instance management. A typical use-case is also presented to show the effectiveness of the layer separation approach.

Architecture of LOD Instance Development

The domain ontology is usually developed by using the conventional ontology editors. SII is generated from the domain ontology represented by RDF and RDF Schema. The architecture of LOD instance development system i-Manager has three main functionalities as shown in Fig. 3. i-Manager creates the instance editing environment as the front end of the instance development. As the back end of the system, i-Manager provides many core functions such as instance editing and store, validation, visualization and navigation. Since the back end functions are implemented in add-on modules, additional subsystems such as SPARQL endpoint can be easily integrated.

i-Manager is a kind of system generator. It generates a specified instance development environment based on the domain ontology automatically or semi-automatically. Moreover, the instance developers can create LOD instances without concerns of ontologies.

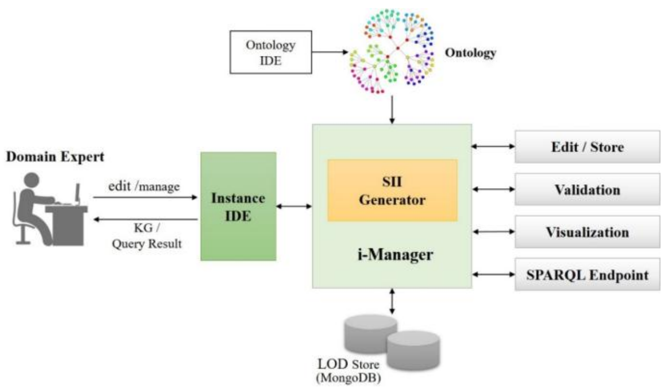


Figure 3: Architecture of LOD Instance Development System.

4.2 Algorithms for Instance Management

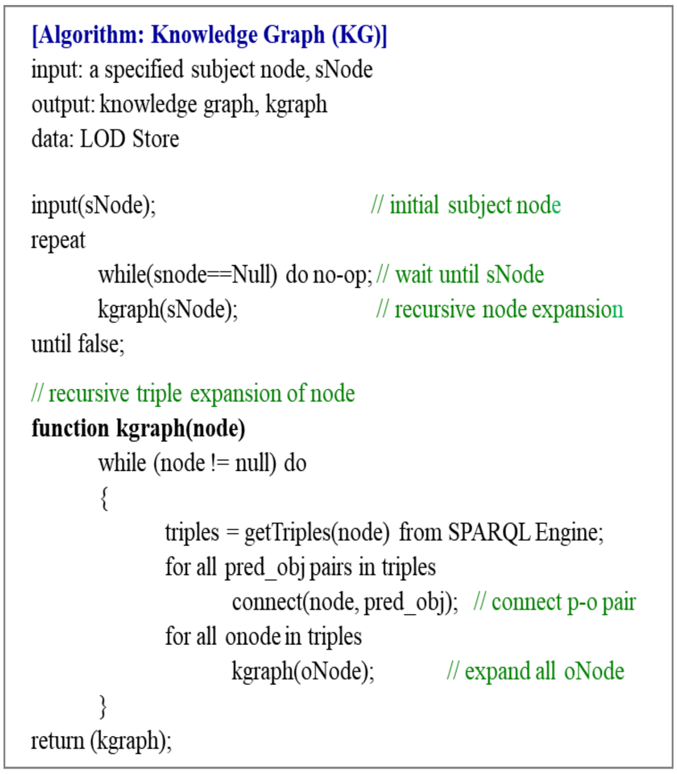
The visualization of LOD organization is important for the effective management of the large amount of the instances. The common representation of LOD visualization is Knowledge Graphs (KGs) that shows the relationships among the triples. KG displays structured information graphically and provides crucial functions for information retrieval, search and question answering. Since SPARQL quires are accomplished on the graph patterns of LOD datasets, KG becomes the primary function that LOD instance development systems should be supported.

i-Manager implements KG generation using SII and SPARQL query processing. When a specific subject node (snode) is given to KG, the following SPARQL query is execute to access the predicate-object pairs related to snode. And then for all object nodes in query result, the same SPARQL query is recursively executed until the whole picture of KG is obtained.

$$SELECT\ \ ?p,\ ?o\ \ WHERE\ \{\ sNode\ \ ?p\ \ ?o\ \} \qquad (4.1)$$

The algorithm of KG generation is shown in Fig. 4.

Other instance management functions can be implemented using SPARQL query like KG generation. The approach used in i-Manager shows more reasonable way to implement LOD development system.



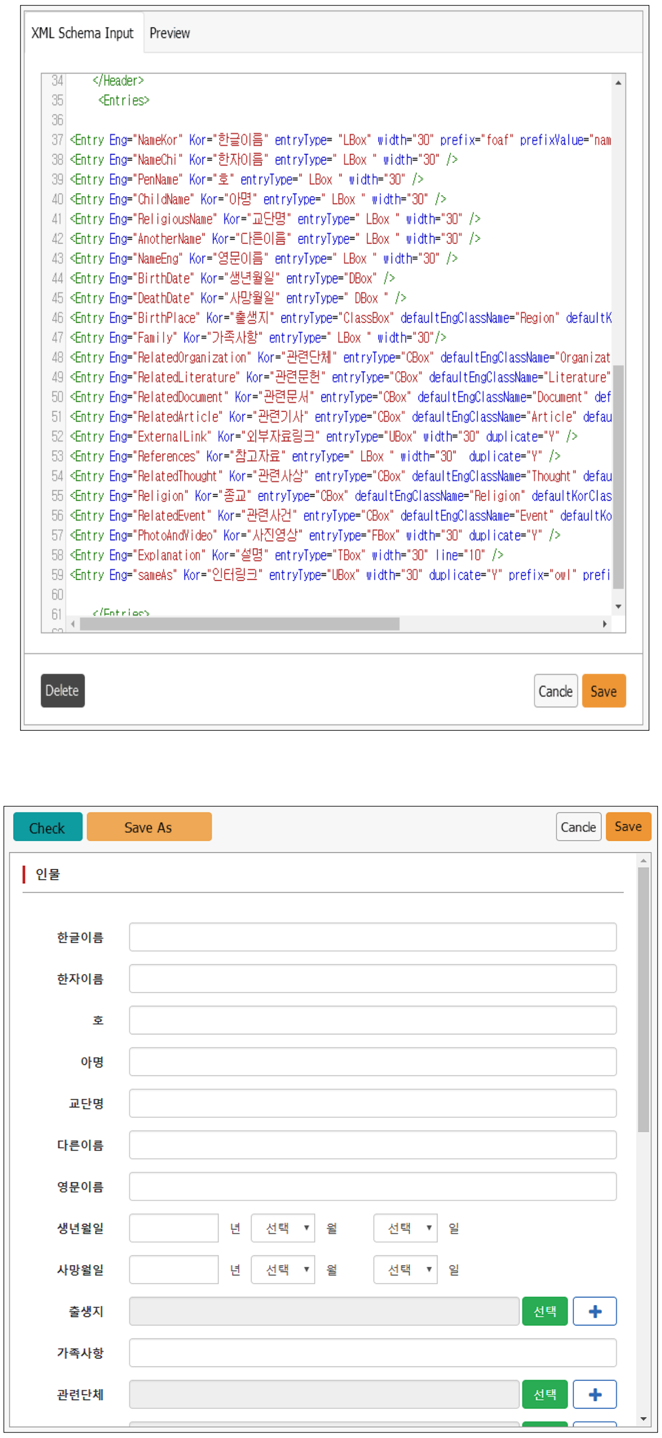
```
[Algorithm: Knowledge Graph (KG)]
input: a specified subject node, sNode
output: knowledge graph, kgraph
data: LOD Store

input(sNode);                        // initial subject node
repeat
        while(snode==Null) do no-op; // wait until sNode
        kgraph(sNode);               // recursive node expansion
until false;

// recursive triple expansion of node
function kgraph(node)
        while (node != null) do
        {
                triples = getTriples(node) from SPARQL Engine;
                for all pred_obj pairs in triples
                        connect(node, pred_obj);   // connect p-o pair
                for all onode in triples
                        kgraph(oNode);             // expand all oNode
        }
return (kgraph);
```

Figure 4: Algorithm for Knowledge Graph Generation.

Algorithms for Instance Management

i-Manager using schema-instance separation approach is applying to publish the instances of scholarly knowledge about religion. The religion as the kind of cultural heritage has the vast amount of heterogeneous resources. The instance developers usually do not have enough knowledge about ontology and LOD. So i-Manager can provide appropriate development environment.

The screen layout for the instance entry is automatically created from SII document. Since the SII document is generated from the domain ontology, the domain-specified instance development system is obtained immediately from ontology specification. Fig. 5 shows SII document of religion ontology and its corresponding screen layout for the instance development.



```
    </Header>
     <Entries>

<Entry Eng="NameKor" Kor="한글이름" entryType= "LBox" width="30" prefix="foaf" prefixValue="nam
<Entry Eng="NameChi" Kor="한자이름" entryType=" LBox " width="30" />
<Entry Eng="PenName" Kor="호" entryType=" LBox " width="30" />
<Entry Eng="ChildName" Kor="아명" entryType=" LBox " width="30" />
<Entry Eng="ReligiousName" Kor="교단명" entryType=" LBox " width="30" />
<Entry Eng="AnotherName" Kor="다른이름" entryType=" LBox " width="30" />
<Entry Eng="NameEng" Kor="영문이름" entryType=" LBox " width="30" />
<Entry Eng="BirthDate" Kor="생년월일" entryType="DBox" />
<Entry Eng="DeathDate" Kor="사망월일" entryType=" DBox " />
<Entry Eng="BirthPlace" Kor="출생지" entryType="ClassBox" defaultEngClassName="Region" defaultK
<Entry Eng="Family" Kor="가족사항" entryType=" LBox " width="30"/>
<Entry Eng="RelatedOrganization" Kor="관련단체" entryType="CBox" defaultEngClassName="Organizat
<Entry Eng="RelatedLiterature" Kor="관련문헌" entryType="CBox" defaultEngClassName="Literature"
<Entry Eng="RelatedDocument" Kor="관련문서" entryType="CBox" defaultEngClassName="Document" def
<Entry Eng="RelatedArticle" Kor="관련기사" entryType="CBox" defaultEngClassName="Article" defau
<Entry Eng="ExternalLink" Kor="외부자료링크" entryType="UBox" width="30" duplicate="Y" />
<Entry Eng="References" Kor="참고자료" entryType=" LBox " width="30"  duplicate="Y" />
<Entry Eng="RelatedThought" Kor="관련사상" entryType="CBox" defaultEngClassName="Thought" defau
<Entry Eng="Religion" Kor="종교" entryType="CBox" defaultEngClassName="Religion" defaultKorClas
<Entry Eng="RelatedEvent" Kor="관련사건" entryType="CBox" defaultEngClassName="Event" defaultKo
<Entry Eng="PhotoAndVideo" Kor="사진영상" entryType="FBox" width="30" duplicate="Y" />
<Entry Eng="Explanation" Kor="설명" entryType="TBox" width="30" line="10" />
<Entry Eng="sameAs" Kor="인터링크" entryType="UBox" width="30" duplicate="Y" prefix="owl" prefi


    </Entries>
```

인물

한글이름	
한자이름	
호	
아명	
교단명	
다른이름	
영문이름	
생년월일	년 선택 월 선택 일
사망월일	년 선택 월 선택 일
출생지	선택 +
가족사항	
관련단체	선택 +

Figure 5: SII document and its screen layout.

Since the screen generated from SII document is similar to the traditional data entry screen, the instance developers can easily edit and store LOD datasets. In i-Manager, the various useful functions for LOD datasets management can be easily implemented. Fig. 6 shows the example of KG using the algorithm in Fig. 4.

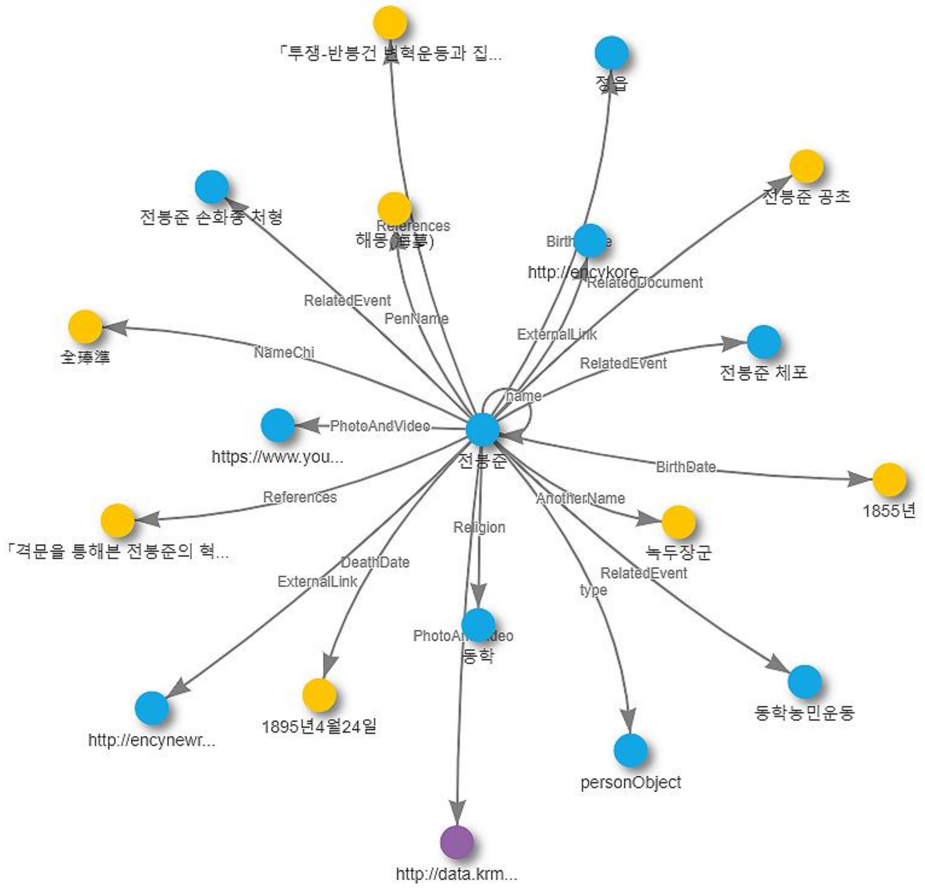


Figure 6: SII Example of Knowledge Graph.

CONCLUSIONS

The data is the key player to realize intelligent information services. Especially, the sharing data on the Web is crucial to develop new information technologies. LOD is the standard data model to share the domain knowledge on the Web. However, the development of LOD datasets have been relied on the development of ontology. This makes it difficult and unreasonable to develop LOD datasets effectively.

This paper presents a noble and practical method for the effective development of LOD datasets. This paper shows how to separate the instance layer from ontology layer. A Schema-Instance Interface (SII) between ontology layer and instance layer is devised to separate and interconnect layers. The SII document can be generated automatically or semi-automatically from the domain ontology specification. Since SII can be customizable, the application-specific instance development environment can be easily configured to realize more flexible user-friendly development environment. Many supporting functions such as knowledge graph, navigation and SPARQL query can be easily integrated in this system.

The proposed system can be applied for the practical LOD datasets development in the diverse application areas since even novices without knowledge about domain ontology can develop LOD datasets effectively.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning(NRF-2017R1A2B4005185).

REFERENCES

[1] Bizer, C., The emerging web of linked data, *IEEE Intelligent Systems*, 24 (2009), 87-92.

[2] Bizer, C., Heath, T., and Lee, T. B., Linked Data -The Story So Far, *Semantic services, interoperability and web applications: emerging concepts*, 5 (2009), 205-227.

[3] Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J. and Auer, S., Quality Assessment for Linked Data: A Survey, *Semantic Web*, 7 (2016), 63-93.

[4] Carrara, W., Chan, W. S., Fischer, S. and Steenbergen, E., Creating Value through Open Data: European Data Portal, Digital Agenda for Europe, pp. 24-58, 2015.

[5] S. Auer, V. Bryl, and S. Tramp, Linked Open Data - Creating Knowledge Out of Interlinked Data: Results of the LOD2 Project, Springer, pp. 1-44, 2014.

[6] D-Lib Magazine, Analysis of International Linked Data Survey for Implementers, *http://www.dlib.org/dlib/july16/smith-yoshimura/07smith-yoshimura.html*.

[7] Michel, F., Montagnat, J. and Faron-Zucker, C., A Survey of RDB to RDF Translation Approaches and Tools, Ph.D, Thisis, I3S, France, 2014.

[8] Jackson, D. and Kang, E., Separation of concerns for dependable software design, *Proc. 10th FSE/SDP workshop on Future of software engineering research*. Association for Computing Machinery, Santa Fe, NM, USA, 2010, pp. 173-176.

[9] Bizer, C., Dong, L., Ilyas, I. and Vidal, M., Special issue on web data quality, *Journal of data and information quality((JDIQ)*, 8 (2016), 1-3.

[10] Zaveri, A., Maurino, A. and Equille, L., Web data quality: Current state and new challenges, *International Journal on Semantic Web and Information Systems (IJSWIS),* 10 (2014), 1-6.

[11] Petrou, I., Meimaris, M. and Papastefanatos, G., Towards a methodology for publishing Linked Open Statistical Data, *eJournal of eDemocracy and Open Government(JeDEM)*, 6 (2014), 97-105.

[12] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, 2011.

[13] Klein, E., Gschwend, A. and Neuroni, A. C., Towards a Linked Data Publishing Methodology, *International Conference for E-Democracy and Open Government (CeDEM)*, IEEE, Krems, Austria, 2016, pp. 188-196.

[14] W3C, A Direct Mapping of Relational Data to RDF. W3C Recommendation 27 September 2012, *https://www.w3.org/TR/rdb-direct-mapping/*.

[15] AI3, The Fundamental Importance of Keeping an ABox and TBox Split, *http://www.mkbergman.com/489/ontology-best-practices-for-data-driven-applications-part-2/.*