Uplifting Relational Data to Knowledge Graphs using RDF*

Ju-Ri Kim[1]
College of Liberal Arts, WonKwang University, Iksan, JeonBuk, 54538, Korea

Article Info
Page Number: 611 – 623
Publication Issue:
Vol. 71 No. 3 (2022)

Abstract. The mapping of relational data into Knowledge Graphs (KGs) is a prospective method to create large-scale KGs. Due to the mapping methods' effectiveness and practicality to uplift relational data into KGs, various mapping approaches such as R2RML have been proposed. The conventional mapping methods are based on the RDF data model. However, RDF reveals some drawbacks in representing complex knowledge structures. A simple syntactic extension of RDF called RDF* has been proposed to resolve difficult problems in RDF. This paper presents a novel mapping method to transform relational data into RDF* constructs. The proposed method focuses on the decomposition of relational data into RDF* constructs instead of the conventional mapping approaches to configure RDF structures according to the dependency relations of relational data. This paper analyzes the functional properties of RDF* constructs and demonstrates the decomposition of relational data into RDF* constructs with typical examples. In addition, a mapping schema diagram (MSD) and a mapping description language are described in this study. This paper proposes the construction of KGs based on RDF* and creates a new research area concerning mapping relational data into RDF*.

Keywords: knowledge graph, RDF*, RDF, direct mapping, RDB-to-RDF, functional dependency, decomposition, mapping description

Article History
Article Received: *12 January 2022*
Revised: *25 February 2022*
Accepted: *20 April 2022*
Publication: *09 June 2022*

1. Introduction

Since Google has launched Knowledge Graph (KG) to refine search engine products, G has been widely disseminated to all scientific and industrial fields to realize intelligent knowledge services(Hogan, 2021; Ji, 2022; Noy, 2019). Though Google initiated the KG revolution, the Semantic Web, which began in the late 1990s with the notion of a Web of data, attempted to make global-scale KGs more substantial(Gandon, 2018). W3C has developed and proposed several standards to support the creation of KGs and Linked Open Data (LOD), such as Resource Description Framework (RDF) as a standard data model for knowledge sharing and SPARQL as a query language for RDF(Hayes, 2014; Klyne, 2004; Garlick, 2013).

[1]+ Corresponding author. Tel.: +82-010-5029-3115.
E-mail address: cyanic@wku.ac.kr.

To develop specific or common KGs, ontology development methods, semantic annotation utilizing domain ontologies, knowledge extraction based on natural language processing (NLP), and knowledge learning through deep learning (DL) have been used(Lehmann, 2015; Michael, 2017; Wang, 2020; Al-Moslmi, 2020). LPGs based on key-value pairs, on the other hand, have garnered considerable attention for enterprise knowledge graphs (EKGs) as they provide a more compact, expressive representation for graph data modeling than RDF, as well as good performance, making knowledge representation and processing more flexible and easy. However, KGs based on LPG models show a lacks of syntactic/semantic interoperability since various vendors use their proprietary version of LPG (Angles, 2020). To settle this complicated circumstance, RDF*, a simple syntactic extension of RDF, has been proposed (Hartig, 2017; Hartig, 2020). RDF* not only embraces RDF's benefits and resolves the difficult problems but also provides enhanced expressiveness comparable to LPG models. RDF* also develops a query language SPARQL* by the extension of SPARQL (Hartig, 2021). Thus, RDF* regards as a promising foundation for KG creation and applications.

This paper addresses KG creation based on RDF* by mapping relational data into RDF* constructs. Since RDF* is more a general model than RDF, the mapping method from relational data to RDF* constructs requires a more distinct method than a direct mapping of RDF. Most of the research on RDF* focuses on developing a theoretical foundation and applying RDF* in the real domain. Only a few studies discuss the mapping of relational data into the RDF* concept. This paper presents the decomposition and the transformation of relational data into RDF* construct.

2. Related Work

Since the vast amount of valuable data is stored in RDB, the mapping of relational data into RDF has attracted significant attention for the development of KGs effectively and practically. Various mapping approaches from different perspectives have been proposed over the last decade (Michel, 2014; Satya, 2009; Sequeda, 2012; Kim, 2020). Those proposed methods have different perspectives on mapping description, mapping implementation, and query processing (Matthias, 2011).

Direct mapping is a representative method recommended by the W3C to support the automatic mapping of relational data into RDF data sets (Satya, 2009; Kim, 2020; Terrazas, 2012). Direct mapping automatically transforms relational data, including relational schema, into RDF data sets. For the transformation, the direct mapping defines explicit mapping rules for transforming both relational schema and instance data into RDF data. The Direct Mapping method typically applies when no ontology suitably describes the domain of the relational database or the rapid publication of RDF data sets with little concern for semantic interoperability is required. Several tools such as BD2OWL and RDBToOnto have been developed to support direct mapping (Michel, 2014; Satya, 2009).

In the direct mapping approach, semantics preservation is critical to ensure that the mapping method generates RDF data without information loss or incorrect semantic data generation. Several modified direct mapping has been proposed to perform a semantics-preserving transformation and optimization of relational databases into RDF data sets without

semantic information loss (Chebotko, 2009; Hee-Gook, 2020). However, several issues such as integrity constraints and incorrect semantic data generation have remained.

A method called Augmented Direct Mapping has been proposed to increase the quality of direct mapping.(Michel, 2014; Sequeda, 2012; Kim, 2020). Augmented Direct Mapping, which aims to preserve semantics, detects common integrity constraints that can convey domain semantics automatically. Since integrity constraints define the semantics of the RDBs, the quality of augmented direct mapping depends on the transformation of the integrity constraints of the RDBs. However, augmented direct mapping is restricted to supporting only referential integrity constraints.

The direct mapping approach is insufficient in real-world applications that have domain-specific knowledge. Domain semantics-driven mapping is a manual mapping method usually applied when the relational database must be translated using classes and properties of existing domain ontologies (Michel, 2014; Sequeda, 2012).

Mapping description languages are used in domain semantics-driven systems to describe expressive mappings and bridge the conceptual gap between RDB and RDF. The W3C RDB2RDF Working Group has recommended the R2RML mapping language for customizing mappings. Several tools such as D2RQ and Ultrawrap have been provided to support manual mapping (Michel, 2014; Sequeda, 2012).

At the moment, it is hard to find references about mapping relational data to RDF* model. Since RDF* is more a general model than RDF, the conventional mapping methods for RDF should be revised to embrace the expressive power of RDF*.

3. Structural Analysis of RDF*

RDF* is a syntactic extension of RDF's conceptual data model and concrete syntaxes that treat an RDF triple as a single resource. In this section, we analyze the structural characteristics of RDF* to explore the valuable features representing relational data.

3.1. Resource Description in RDF*

RDF* as the simple extension of RDF resolves some limitations and enhances the descriptive power of RDF. The emerging proposal RDF* extends RDF's benefits to describe the complex semantic relationships. RDF* merely emphasizes a syntactic extension of RDF that makes it possible to create the compatible RDF assertion triples as a resource. The base structure of RDF* is defined as follows:

[Definition: RDF*] An RDF* triple is a 3-tuple that is defined recursively as follows:

- any RDF triple t of <s, p, o> is an RDF*,
- where s is the subject, p the predicate, and o the object, respectively.
- if t and t' are RDF* triples, the tuples <t, p, o>, <s, p, t>, and <t, p, t' > are RDF* triples.

The resource with IRI or literal can be a subject or object as in RDF. Note that, by definition, an RDF* triple cannot contain itself and cannot be nested infinitely. In RDF*, any triple representing metadata about another triple can be directly embedded in another triple as its subject or object. Hence, there are two types of resources in RDF*: the conventional RDF resource and the triple structure. This paper defines them as a single resource and triple

resource, respectively. While a single resource is an ordinary resource, a triple resource is defined as follows:

[Definition: Triple resource] A triple resource is the conceptual resource of a certain domain that has a single resource structure <s, p, o> internally and has a relationship with the single or triple resource externally.

By definition, RDF* and RDF represent a binary relationship between resources regardless of whether they are a single or triple resource. This binary relationship of RDF* is a crucial feature compared to the relational database tables that usually represent N-ary relationships. To emphasize the binary relationship of RDF*, this paper represents a triple with the form <head resource, relation, tail resource> (abbreviated as <h, r, t>). This convention is also commonly used in KGs rather than <s, p, o> to specify the relation without a biased view.

3.2. Categorization of RDF* Constructions

We categorized RDF* constructs to analyze the detailed relational properties between resources. RDF* construct can be categorized by the head and tail structures since the relation plays the role of the semantic connection between head and tail. By the definition of RDF*, there exist four types of constructs, as shown in Fig 1.

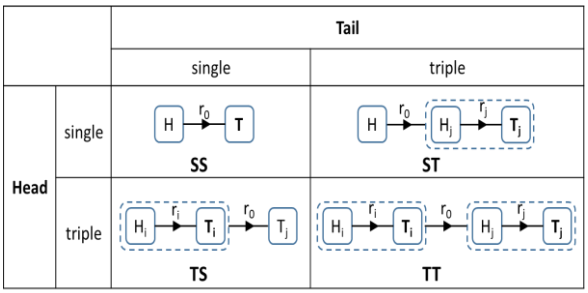


Fig. 1: Types of RDF* Constructs.

- **SS Type**

This type is the typical RDF construct to represent the factual relationships of resources. Although SS type is a simple binary relation, it can represent 1:1 or 1:N relationships, for example, <ex:John, ex:works_with, ex:David> and <ex:John, ex:works_with, ex:Anna>.

- **ST Type**

This construct represents the general structure of the reification defined by the RDF standard (Orlandi, 2021). Although RDF reification has recently been removed from RDF Recommendation's normative sections, it is valuable to describe assertions such as beliefs, assumptions, reliability, and assurance (Hernández, 2021). Many theoretical arguments about the description of the reification have been published (Orlandi, 2021). However, this paper focuses on the relationships among the resources.

To be an ST type construct, the tail can be an arbitrary, meaningful triple structure containing the head and tail. The head of ST constructs functionally dominates the triple

resource. The relationships between head and tripe resource r_0 are independent of the relation r_j in the triple resource.

- **TS Type**

The TS type represents adjunct relationships. This type is suitable for the representation of N-ary relationships. The triple resource as the head plays the role of the composite key to address the tail.

- **TT Type**

The TT type represents relationships between triple resources. The relationships r_0, r_i, and r_j are independent of each other. The head triple resource like TS type plays the role of the composite key.

3.3. Relational Analysis of RDF* Constructs

The relation between head and tail in RDF* constructs show a binary association similar to relational database functional dependency (FD). When we deal with triple resources the same as single resources, the relations in RDF* can hold FD. This paper defines functional dependency shown between head and tail as follows:

[Definition: Functional Dependency of RDF* (FDR)]

A relation r of RDF* satisfies functional dependency FDR: H →T whenever for each resource instance $\alpha\ \alpha$ and $\beta\ \beta$ such that $H[\alpha] = H[\beta]\ H[\alpha] = H[\beta]$, then it must be that $T[\alpha] = T[\beta]\,T[\alpha] = T[\beta]$, where $H[x]H[x]$] and $T[x]T[x]$ are the head and tail of resource instances x, respectively.

The FDR is an RDF* version of FD of the relational database. Note that FDR only holds in RDF* constructs such as ST, TS, and TT that have triple resource regarded as a single resource. In the case of the SS type construct, although it represents a simple binary relation between head and tail, the relation can be 1:1 or 1:N. Fortunately, the 1:1 relation of SS type can apply FDR. For the 1:N relation of SS type, we can recognize that each triple represented in 1:N relation expresses the factual data similar to the triple resource. In other words, the triple in 1:N relation behaves like the triple resource, not a single resource. So, if we handle the 1:N relation of SS type as an independent triple resource, we can unify the relationships of the RDF* construct within FDR. We discuss the application of the triple resource of 1:N relation in Section 4.

From the relational database perspective, the head of FDR plays the role of the primary key to access the tail. In TS and TT type, the primary key is a composite key containing the head and tail of the triple resource.

4. Decomposition of Relational Data into RDF* Constructs

The traditional RDB-to-RDF mapping approaches focus on the dependency relationships among the columns in relational tables. After then, the mapping methods try to discover the appropriate structures suitable for RDF (Satya, 2009). Since those mapping methods depend

on the organization of relational tables, they are complicated to realize the expressive power of RDF. Our approach first extracts RDF* constructs from a relational table's organization, then uses RDF* constructs to represent the decomposed structures.

4.1. Considerations for Decomposition Mapping

The RDF* constructs are based on the binary relationship between the head and the tail. On the other hand, relational tables, in general, represent the complex relationships in one table. It is common that the functional relations of the entities are separated or distributed into several tables. These invisible relationships are usually connected by the association of the primary key and the foreign key. When we consider the motivations and objectives of mapping relational data to the RDF* model, a mapping method for two heterogeneous structures is required to comply with some principles to generate the qualitative RDF* data sets. Although some researches including the W3C proposal, have analyzed formal requirements for flawless mapping, the followings are the core principles that a mapping method should materialize (Terrazas, 2012; Chebotko, 2009).

- **Semantic preservation:** No matter how the structure varies significantly between two heterogeneous models, the relationships and knowledge inherent in relational tables should be preserved in RDF* data sets in itself. In other words, the mapping method accompanying the decomposition and restructure should be a lossless transformation.
- **Completeness:** The mapping method should cover all types of relations in relational data such as 1:1, 1:N, M:N, and recursive relationships completely and precisely in a consistent manner. In addition, it should also handle the invisible relationships by the association of the primary and foreign keys.
- **Intelligibility:** The mapping description should be simple to define complex relationships and comprehensive to understand and implement. The complexity of the mapping description shown in several mapping approaches hinders disseminating the effective way to create RDF* data sets (Chebotko(2019)).
- **Quality of the generated RDF* data sets:** The resulting data sets should be usable in actual applications, as the mapping's primary purpose is to build high-quality RDF* data sets. Furthermore, the mapping method should facilitate the use of related ontologies to improve the semantic expressiveness of the RDF* data set.

This paper adopts a mapping schema diagram (MSD) as shown below to realize the above considerations. The MSD plays the role of the mediator or facilitator to mapping description. Since MSD is a visual description, MSD is easy to develop and transform into a mapping description.

4.2. Decomposition of Relational Structures

This section describes the typical decomposition of the principal relational structures into RDF* constructs with examples. It also demonstrates the binary decomposition of relational tables with MSD based on RDF* constructs and FDR categorization.

- **Primary decomposition**

Two tables, EMP and DEPT of Fig. 1, are representative examples of data organization. Although this example does not contain RDF* triple constructs, we analyze relational structure to explain MSD.

We can easily find out two FDs: FD:ENO → {ENAME, JODB, DEPT} in the table EMP and FD: DNO → {DNAME, LOC}. We can also figure out that table EMP joins DEPT together using EMP.DEPT and DEPT.DNO. The MSD of Fig. 2(b) is an RDF* data model representing the tables' relationships in Fig. 2(a). MSD uses a notation T.C to identify column C of table T. This paper defines a linking resource to represent the join relation of two tables. A linking resource has two T.C consisting of correspondence of the primary key and the foreign tables.

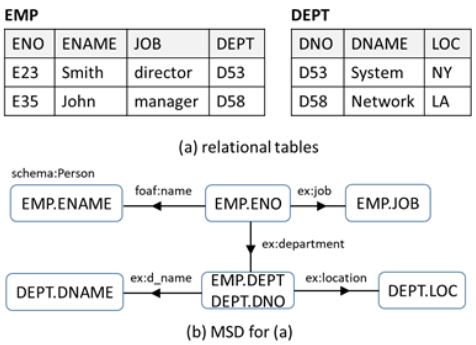


Fig. 2: Decomposition and MSD representation of two relational tables.

[Definition: Linking resource] A linking resource consisting of Ti.Ki and Tj.Kj represents the status that two tables Ti and Tj are joined by the key equivalence Ti.Ki = Tj.Kj .

This paper utilizes a declarative linking resource rather than a join operation to connect two tables.

In Fig 2(b), {EMP.DEPT, DEPT.DNO} is a linking resource representing that they are a connector to join table EMP and DEPT. In MSD, we can specify additional information such as resource type on head and tail. We can use an appropriate relation name with the well-known namespace. Note that the head and the tail should be in the same table since the RDF* construct is based on a binary relationship.

From MSD of Fig 2(b), we can generate RDF* data set as follows, to name a few:

By foaf:name relation,

ex:E23 rdf:type schema;Person ;
foaf:name “Smith” .
ex:E35 rdf:type schema;Person ;
foaf:name “John” .

By ex:department relation,

ex:E23 ex:department ex:D53 .
ex:E35 ex:department ex:D58 .

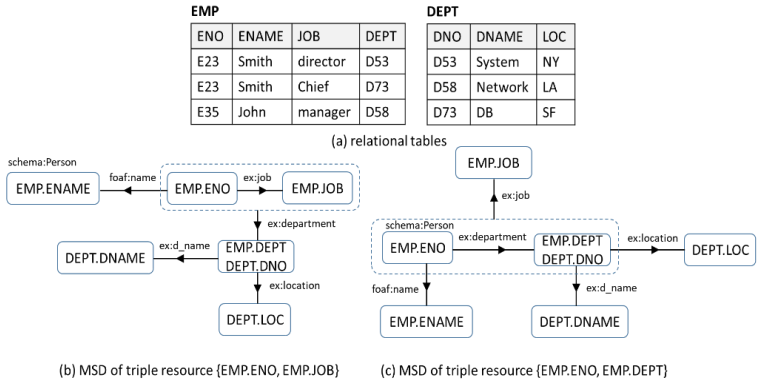
By ex:location,

ex:D53 ex:location “NY” .
ex:D58 exlocation “LA” .

■ **Decomposition of 1:N relations**

This paper reveals that the 1:N relationship of SS type behaves as a triple resource, and this kind of triple resource can unify the application of FDR. Fig 3, which is a modified version to compare FDR, comprises 1:N relation in the table. We can recognize the 1:N relation between ENO and {JOB, DEPT}. Thus, two triple resources are possible in this case. We depict two MSDs in Fig 3(b, c) to compare their context.

Fig. 3: Decomposition and MSD representation of 1:N relational tables.



Although the two MSDs in Fig 3 represent the same information, their context is different. The MSD in Fig 3(b) focuses on the department when the employee has the job, while MSD in Fig 3(c) stresses the employee's job when the employee works in the department. We can choose one of the MSDs according to the context of the domain.

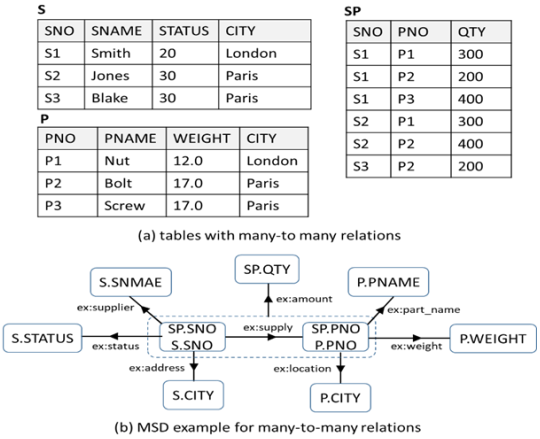
The MSD in Fig 3(b) generates TS type of RDF* data sets, for example, as follows:
By ex:job and ex:department relations,
<ex:E23 ex:job "director"> ex:department ex:D53 .
<ex:E23 ex:job "chief"> ex:department ex:D73 .
By ex:location relations,
ex:D552 ex:location "NY" .
ex:D58 ex:location "LA" .
ex:D79 ex:location "SF" .

The mapping method of this paper can capture all aspects of relational data and is flexible in choosing the appropriate context according to the view of the application domain.

■ **Decomposition of M:N Tables**

The MSD of this paper can describe the complex relations even though they do not relate with the RDF* construct. As we illustrate above, MSD smoothly represents the structure of SS types with the linking resources. The following example in Fig 4 referred from contains many-to-many relations.

Fig. 4: Representation of many-to-many relations.



S

SNO	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	30	Paris
S3	Blake	30	Paris

SP

SNO	PNO	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

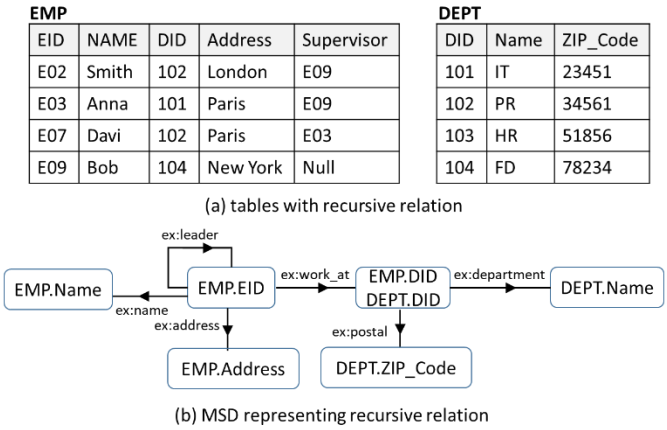
P

PNO	PNAME	WEIGHT	CITY
P1	Nut	12.0	London
P2	Bolt	17.0	Paris
P3	Screw	17.0	Paris

The M:N relation shown in the SP table of Fig 4(a) is represented by the TS type of RDF* construct. The composition of MSD and the generation of the RDF* data set are trivial.

■ Decomposition of recursive relations

The recursive relation is one of the cumbersome problems in relational databases since it requests self-join. In general, the self-join is time-consuming unless using some special techniques such as indexing and the addition of extra key fields. However, in RDF* model based on binary relation, the recursive relation is a kind of FDR.



EMP

EID	NAME	DID	Address	Supervisor
E02	Smith	102	London	E09
E03	Anna	101	Paris	E09
E07	Davi	102	Paris	E03
E09	Bob	104	New York	Null

DEPT

DID	Name	ZIP_Code
101	IT	23451
102	PR	34561
103	HR	51856
104	FD	78234

Fig. 5: Recursive relation and MSD

The EMP.Supervisor shows the recursive relationship with EMP.EID. The generation of the RDF* data set is simply accomplished by the decomposition of two columns as follows:
By ex:leader relations,
ex:E02 ex:leader ex:E09 .
ex:E03 ex:leader ex:E09 .
ex:E07 ex:leader ex:E03 .

In case that the head or the tail is NULL, RDF* data set are not created because they cannot be a triple.

5. Mapping Description for MDS

We adopt MSD to decompose relational data into RDF* constructs. The decomposed model represented by MSD should be expressed by mapping language to generate RDF* constructs. RDF uses W3C-recommended R2RML as a mapping description language (Terrazas, 2012). The structure of an R2RML document consists of one or more triples maps, which contain a logical table, a subject map, and a number of predicate-object maps to generate RDF triples. So, the mapping description described in R2RML is complicated and hard to understand. This mapping description is represented as an RDF graph using the R2RML vocabulary and serialized in the Turtle syntax, the recommended syntax for writing R2RML mapping documents.

This paper uses a more compact description based on feature structure or attribute-value matrix well-known in NLP and NoSQL databases (Besta(2019)). The feature structure consisting of key-value pairs provides a flexible description for the hierarchical properties of entity and relation. So we can organize the related properties systematically and expand the functionalities incrementally.

Fig 6 shows the part of the mapping description of Fig 3(b). The feature structure provides a convenient way to describe and append related information to improve the generated RDF* constructs' quality smoothly. For entity description, the additional information not shown in relational data such as ontology class, language, and comments are inserted with key-value pairs. The EMP-DEPT shows the description of a linking resource.

The relation description consists of two core features, head and tail. Similar to entity description, the relevant information for the relation is appended with ease. The head of the relation ex:department shows that its head is a triple resource.

Fig. 6: Mapping description of Fig 3(b)

```
// prefix
// entity description

EMP.ENAME
[type      = entity
 class     = schema:Person
 datatype  = literal]

EMP.ENO
[type      = entity]

EMP.JOB
[type      = entity
 datatype  = literal
 language  = 'en']

EMP-DEPT
[type      = linking_resource
 resource  = EMP.DEPT
 sameAs    = DEPT.DNO]

// relation description

foaf:name
[type = relation
 head = [resource = EMP.ENO]
 tail = [resource = EMP.ENAME]]

ex:job
[type = relation
 sameAs = schema.job
 head = [resource = EMP.ENO]
 tail = [resource = EMP.JOB]]

ex:department
[type = relation
 description = 'working department'
 head = [resource = ex:job
         type= relation]
 tail = [resource = EMP-DEPT
         type= linking_resource]]
```

In general, since the mapping description is a kind of specialized knowledge, the feature structure is suitable for representing diverse knowledge necessary for mapping relational data and generating RDF* constructs. The mapping description using feature structure maintains consistency in the representation of entity and relation.

6. Conclusions

The efficient, practical construction of large-scale KGs is the groundwork for the dissemination of intelligent knowledge services. Mapping relational data into KGs is an effective and practical way to create large-scale KGs seamlessly. Many mapping approaches for RDF-to-RDF have been proposed to realize the Web of Data and KGs. However, the practical mapping approach for RDF is still an open question. Moreover, few studies mention mapping relational data into the RDF* construct.

Since RDF* is a simple extension of RDF but has potent features to represent more complex knowledge structures, a noble mapping method embracing RDF* capabilities play a significant role in KG creation. This paper describes a practical construction of KGs by mapping of relational data to RDF* constructs. While most of the RDB-to-RDF mapping methods concentrate on dependency relationships among relational data, our approach focuses on the extraction of RDF* constructs innate in relation data. This paper analyzes the functional property of RDF* constructs and shows how the complex relationships in relational data can be decomposed into RDF* constructs with typical examples. The mapping relation is visualized in this paper using a mapping schema diagram (MSD). This also presents a simple mapping description based on MSD.

The mapping method proposed in this paper can be applied to whether relational tables are normalized or not. It also can process CSV (comma-separated values) files commonly encountered in spreadsheets and databases.

There have been many mapping methods proposed for RDF, but few for RDF*. This paper addresses a new mapping method that can realize the functional properties of RDF*. More research works for the implementation of the mapping method and validation with the real data are necessary.

7. Acknowledgements

This paper was supported by WonKwang University in 2022.

8. References

1. Aidan Hogan, Eva Blomqvist, etc. (2021). Knowledge Graphs. *ACM Comput. Surv*, 54, 4, 71:1-71:37. DOI: 10.1145/3447772
2. Al-Moslmi, Tareq, et al. (2020). Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8, 32862 – 32881. DOI: 10.1109/ACCESS.2020.2973928.
3. Besta, Maciej, et al. (2019). Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *Dblp computer science bibliography*, 1-41. DOI: arxiv-1910.09017
4. Boris Villazón-Terrazas, Michael Hausenblas. (2012). R2RML and Direct Mapping Test Cases. *W3C Editor's Draft 24 July 2012*, http://www.w3.org/2001/sw/rdb2rdf/test-cases.
5. Chebotko, A., et al.(2009). Semantics Preserving SPARQL-to-SQL Translation. *Data & Knowledge Engineering*, 68, 10, 973-1000. DOI:10.1016/j.datak.2009.04.001
6. Chenguang Wang, Xiao Liu, Dawn Song. (2020). Language Models are Open Knowledge Graphs. *We gratefully acknowledge support from the Simons Foundation and member institutions*. DOI: arXiv:2010.11967.

7. Fabrizio Orlandi, Damien Graux, Declan O'Sullivan. (2021). Benchmarking RDF Metadata Representations: Reification, Singleton Property and RDF*. *IEEE*, 15, 233-240. DOI: 10.1109/ICSC50631.2021.00049
8. Gandon F. (2018). A survey of the first 20 years of research on semantic web and linked data. *Ingénierie des Systèmes d'Information*, 23, 3-4, 11-56. DOI : 10.3166/ISI.23.3-4.11-56
9. Garlick, S. and Seaborne, A. (2013). SPARQL 1.1 Query Language. *W3C Recommendation*. http://www.w3.org/ TR/sparql11-query/
10. Hee-Gook Jun, Dong-Hyuk Im. (2020). Semantics-Preserving RDB2RDF Data Transformation Using Hierarchical Direct Mapping. *Appl. Sci*. 2020, 10, 20, 7070. DOI : 10.3390/app10207070.
11. Hernández, Daniel, Aidan Hogan, and Markus Krötzsch. (2015). Reifying RDF: What works well with wikidata?. *SSWS@ ISWC*, 1457, 32-47. DOI:10.1.1.711.6594
12. Hert, Matthias, Gerald Reif, and Harald C. Gall. (2011). A comparison of RDB-to-RDF mapping languages. *Proceedings of the 7th international conference on semantic systems*, 25-32. DOI: 10.1145/2063518.2063522.
13. Ji S, Pan S, Cambria E, Marttinen P, Yu PS. (2022.) A survey on knowledge graphs: Representation, acquisition and applications. *IEEE Transactions on Neural Networks and Learning*, 33, 2, 494-514. DOI: 10.1109/TNNLS.2021.3070843
14. Ju-Ri Kim. (2020). Implementation of RDB(Relational DataBases) Tables Using RDF(Resource Description Framework) Schema Mapping Modeling. *International Next-generation Convergence technology Association*, 4, 4, 381-387.
15. Klyne G, Carroll JJ, McBride B (2004) Resource description framework (RDF): concepts and abstract syntax. *W3C Recommendation*. https://www.w3.org/TR/rdf-concepts/
16. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Van Kleef P, Auer S, Bizer C. (2015). DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6, 2, 167–195. DOI:10.3233/SW-140134.
17. Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, Max Welling. (2017).,Modeling Relational Data with Graph Convolutional Networks. *We gratefully acknowledge support from the Simons Foundation and member institutions*. DOI:arXiv:1703.06103
18. Michel, F., Montagnat, J., Faron-Zucker, C. (2014). A Survey of RDB to RDF Translation Approaches and Tools. *Rapport de Recherche*, ISRN I3S/RR 2013-04-FR. https://hal.archives-ouvertes.fr/hal-00903568v2.
19. O. Hartig. (2017). Foundations of RDF* and SPARQL*: An Alternative Approach to Statement-Level Metadata in RDF, *SEMWEB 2017*.
20. Olaf Hartig, Pierre-Antoie Champin, Gregg Kellogg, Andy Seaborne. (2021). RDF-star and SPARQL-star. Draft Community Group Report 01 July 2021. https://w3c.github.io/rdf-star/cg-spec/2021-07-01.html
21. P. Hayes and P. Patel-Schneider. (2014). RDF 1.1 semantics. W3C recommendation. https://www.w3.org/TR/rdf11-mt/
22. Rebzo Angles, Harsh Thakkar, Dominik Tomaszuk. (2020). Mapping RDF Databases to Property Graph Databases, IEEE Access, 8, 86091 – 86110, DOI: 10.1109/ACCESS.2020.2993117

23. Satya S. Sahoo, Halb W., Hellmann S., Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, Ahmed Ezzat. (2009). A Survey of Current Approaches for Mapping of Relational Databases to RDF. W3C RDB2RDF Incubator Group January 08 2009. http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport_01082009.pdf.
24. Sequeda, J., et al. (2012). Relational Database to RDF Mapping Patterns. Proceedings of the 3rd International Conference on Ontology Patterns (WOP'12), 929, 97-108. DOI:10.5555/2887724.2887733.
25. Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, Jamie Taylor. (2019). Industry-Scale Knowledge Graphs: Lessons and Challenges. Communications of the ACM, 62, 8, 36-43. DOI : 10.1145/3331166