Article

Journal of Information Science
1–16
© The Author(s) 2016
Reprints and permissions:
sagepub.co.uk/journalsPermissions
.nav
DOI: 10.1177/0165551510000000
jis.sagepub.com

Semantics-preserving optimization of mapping multi-column key constraints for RDB to RDF transformation

Hee-Gook Jun
School of Computer Science and Engineering, Seoul National University

Dong-Hyuk Im
Department of Computer Engineering, Hoseo University

Hyoung-Joo Kim
School of Computer Science and Engineering, Seoul National University

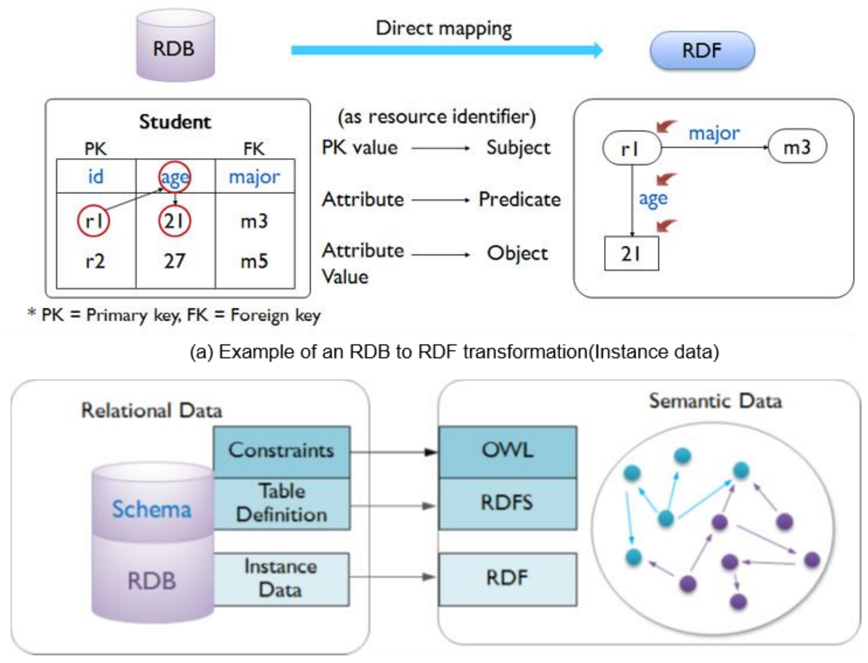
Abstract
The relational database (RDB) to Resource Description Framework (RDF) transformation is a major semantic information extraction method because most web data are managed by RDBs. Existing automatic RDB to RDF transformation methods generate RDF data without losing the semantics of original relational data. However, two major problems have been observed during the mapping of multi-column key constraints: repetitive data generation and semantic information loss. In this paper, we propose an improved RDB to RDF transformation method that ensures mapping without the aforementioned problems. Optimized rules are defined to generate an accurate semantic data structure for a multi-column key constraint and to reduce repetitive constraint data. Experimental results show that the proposed method achieves better accuracy in transforming multi-column key constraints and generates compact semantic results without repetitive data.

Keywords
Semantic web, Relational database (RDB), Resource Description Framework (RDF), RDF schema (RDFS), RDB to RDF transformation (RDB2RDF), Web Ontology Language (OWL)

1. Introduction

Semantic web data publication methods based on relational data are widely studied because more than 70% of web documents are backed up by relational databases (RDBs) [1]. The RDB to Resource Description Framework (RDF) (RDB2RDF) transformation is a major semantic data publication method that extracts semantic data from relational data [2], [3]. In particular, direct mapping [4] is a representative automatic RDB2RDF mapping method recommended by the World Wide Web Consortium (W3C). The rules of direct mapping are defined to transform relational instance data, including attributes and attribute values, into semantic RDF graph data. Figure 1. illustrates an example of the RDB2RDF transformation. In the transformation, relational data are mapped to RDF triples. Suppose “Student” is a relational table, “age” is an attribute in Student, “r1” is a primary key of an instance record of Student, and “21” is an instance value of age in *r*1. Then, the relational data are transformed into a triple (*r*1, *age*, 21).

Corresponding author:
Dong-Hyuk Im, Department of Computer Engineering, Hoseo University
Email: dhim@hoseo.edu

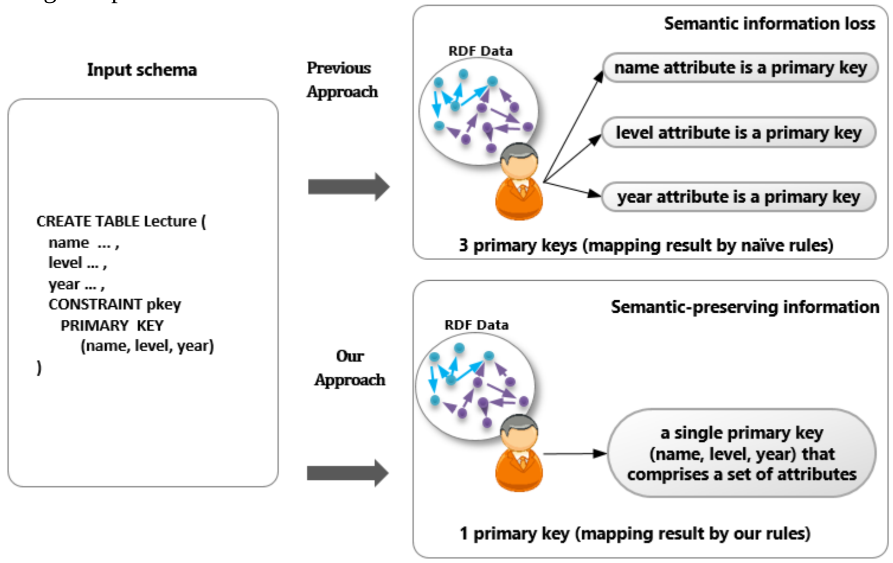


(a) Example of an RDB to RDF transformation(Instance data)

(b) RDBtoRDF transformation with RDFS and OWL(Semantics-preserving)

Figure 1. Example of direct mapping of a single-column primary key

However, existing works have experienced problems in the transformation of multi-column key constraints into semantic data. Multi-column key constraints are primary, foreign, or unique key constraints that are composed of two or more attributes. First, it has been observed that information loss occurs if multi-column key constraints are included in relational input data because most existing methods define key constraint mapping rules based on single-column key constraints [5], [6], [13], [14]. They may generate incorrect semantic data, and transformed data may not be equivalent to original input data. A few of these methods consider scenarios in which key constraints can be composed of one or more attributes. Nevertheless, these methods do not implement rules and algorithms to resolve multi-column key constraint mapping issues. Second, a repetitive constraint data generation problem is inevitable during the mapping of multi-column key constraints based on previous rules. The main reason for this problem is that each column data item for a multi-column key constraint shares identical key constraint semantics. Repetitive constraint data generation may cause a storage overlay with additional computation costs, reduced readability, and fewer intuitive semantics. Therefore, in this paper, we propose an optimized method to improve the direct mapping in the RDB2RDF transformation for the specific case of multi-column key constraints existing in input data.


Input schema
Previous Approach
Our Approach
Semantic information loss
RDF Data
name attribute is a primary key
level attribute is a primary key
year attribute is a primary key
3 primary keys (mapping result by naïve rules)
Semantic-preserving information
RDF Data
a single primary key (name, level, year) that comprises a set of attributes
1 primary key (mapping result by our rules)

```
CREATE TABLE Lecture (
  name  ... ,
  level ... ,
  year ... ,
  CONSTRAINT pkey
    PRIMARY  KEY
       (name, level, year)
)
```

Figure 2. Semantics-preserving RDB to RDF transformation

Figure 2. shows the generic comparisons of our scheme with existing methods. To the best of our knowledge, the implementation of multi-column key constraint mapping rules has not yet been addressed. The contributions of this work are as follows:

- Presentation of semantics-preserving multi-column key constraint transformation: We consider the prevention of two major problems, i.e., semantic information loss and repetitive data generation, and define semantics-preserving mapping rules. These problems occur in the mapping of multi-column key constraints because mapping at the schema level has not yet been appropriately implemented. However, a direct mapping method recommended by W3C involves multi-column key mapping at the instance data level. Thus, we aim to resolve the above problems for building a semantics-preserving direct mapping method.
- Accurate mapping of multi-column key constraints: We define mapping rules that compensate for the structural difference between relational data and semantic data to transform multi-column key constraints without semantic information loss. As the defined rules cover key constraints and multi-columned hierarchical structures, we can prevent the generation of incorrect semantic data, which are different from original relational data.
- Pruning of repetitive output data: We optimize mapping rules to eliminate repetitive data, which are generated during multi-column key constraint mapping. Unlike in existing methods, optimized rules generate compact constraint semantic data. Moreover, we use an experimental approach and show that the output data transformed by the provided mapping rules still preserve the semantics of input relational data.

The remainder of this paper is structured as follows: In Section 2, we introduce an overview of RDB2RDF transformation methods. Next, we provide the preliminaries of direct mapping and describe the problems that occur during the mapping of multi-column key constraints. In Section 4, we present our mapping method in detail. Section 5 reports experimental results. Finally, we present the conclusions and the perspectives for future work.

2. Related Work

The RDB2RDF transformation is a mapping method for generating semantic data from relational data. The format of semantic output data is defined by RDF. RDF is a modelling language for describing semantic resources on the semantic web [14]. The structure of RDF data is graph data, in which nodes are semantic resources and edges are the relationships among resources [15]. RDF graph data are comprised of RDF triples, which are the sets of semantic resources formed by (subject, property, and object). A subject is a semantic resource that contains a uniform resource identifier (URI) [16] to be uniquely identified. An object can be a semantic resource that contains a URI, a literal value, or a blank node [17] containing no URI or literal value information. A property (also referred to as a predicate) is a semantic resource that expresses a relationship between a subject and an object.

Two types of approaches exist for mapping generation: domain semantics-driven mapping and direct mapping [18], [19]. The first approach manually transforms relational data into RDF data [20], [21]. The RDB2RDF mapping language [22], [23] is a representative mapping language for manual mapping. It was developed in 2010 and recommended in 2012 by W3C. Manual mapping applications are provided for users to manage mapping processes, such as D2RQ [24], Virtuoso [25], and Ultrawrap [26]. Direct mapping was developed in 2010 as an automatic mapping approach, and it was recommended in 2012 by W3C [27]. Direct mapping uses relational instance data and schema data as input data, and it outputs RDF semantic data.

Methods have been developed for improving the performance of direct mapping [7-14]. RDFS [28] is an extension of the RDF vocabulary for describing classes, properties, and utility properties. OWL [29] is a description-logic-based language that extends RDF and RDFS. Therefore, the improved approaches adopt RDFS and OWL to generate semantic data without information loss. The entire integrity constraint information in schema data cannot be transformed into semantic data. Thus, the author of [13] proposed an improved approach for generating semantics-preserving output data. However, the method still lacks support for the transformation of all integrity constraints. In particular, a multi-column primary key or a foreign key cannot be transformed by the method. Thus, we focus on solving the transformation of multi-column key constraints and reducing the output data size in mapping multi-column key constraints. Other recent approaches on mapping into RDF data transform various types of data (heterogeneous data [30], object-oriented database [31], and UML [32]) to RDF data. However, we mainly focus on direct mapping to manage large-scale data on the Web.

3. Preliminaries and problem description

In this section, we explain existing direct mapping rules[*] with examples. Then, we describe two problems of mapping multi-column key constraints: semantic information loss and the generation of repetitive constraint data.

3.1. Direct mapping rules

Table 1. General rules for mapping relational data.

Rule	General Rules	Description of Rules (RDB to RDF)
1	Class(*r*) ← Rel(*r*) ∧ ¬ BinRel(*r* , ,)	Relational table OWL class
2	DatatypeP(*x*, *r*, *type(x)*) ← NonFKeyAttr(*x*, *r*)	Attribute → OWL data-type property
3	ObjP(*r*, *s*, *t*) ← BinRel(*r*, *s*, *t*) ∧ ¬ BinRel(*s*, ,) ∧ ¬ (*t*, ,)	Binary relation → OWL object property
4	ObjP(*x*, *r*, *s*) ← FKeyAttr(*x* ,*r*, *s*)	Foreign key attribute → OWL object property
5	SubClassOf(*r*, *s*) ← Rel(*r*) ∧ Rel(*s*) ∧ PKey(*x*, *r*) ∧ FKey(*x*, *r*, *s*)	Parent table and child table → OWL subclass

Table 2. Examples of RDB2RDF transformation using general rules.

Rule	Output Triple Data	Description
1	(Person, rdf:type, owl:Class)	Person is a relational table
2	(age, rdf:type, owl:DatatypeProperty) (Person1, age, “23”)	Age is an attribute. “23” is a value of age for Person1
3	(Order, rdf:type, owl:ObjectProperty) (Customer, Order, Product)	Order is a binary relation with domain Customer and range Product
4	(major, rdf:type, owl:ObjectProperty) (Student1, major, m002) (m002, rdf:type, Major)(m002, name, “Math”)	Major is a foreign key referencing table Major. Math is a major of Student1
5	(Cat, owl:subClassof, Mammal)	Cat is a Mammal

Table 3. Constraint rules for mapping relational data.

Rule	Constraint Rules	Simplified Description (RDB to RDF)
6	Class(*r*) ← Rel(*r*) ∧ ¬ BinRel(*r* , ,)	Relational table OWL class
7	Card(*x*,*r*,1) ← NotNull(*x*,*r*)	Rule for mapping a not-null constraint
8	InverseFunctionalIP(*x*) ← Unique(*x*,*r*)	Rule for mapping a unique constraint
9	InverseFunctionalIP(*x*), Card(*x*,*r*,1) ← PKey(*x*,*r*)	Rule for mapping a primary key constraint
10	ObjP(*x*,*r*,*s*) ← FKey(*x*,*r*,*s*)	Rule for mapping a foreign key constraint
11	defValue(*x*,*r*,*v*) ← Default(*x*,*r*,*v*)	Rule for mapping a default constraint

Table 1 shows the general rules for mapping relational data. A relational table is transformed into an OWL class by Rule 1. Rule 2 transforms an attribute of the table into an OWL data-type property. It contains a relational instance defined by a URI as a subject and a literal value defined as an object in RDF triple data, such as (instance of a table, attribute, literal value). A binary relation is transformed by Rule 3. A foreign key attribute is transformed by Rule 4. The relationship between a parent table and a child table is transformed by Rule 5. The simplified examples of the general rules are outlined in Table 2 for better understanding. Constraint rules are defined to transform the integrity constraints of relational schema

[*] The rules are fundamentally based on the method reported in [13], which demonstrated the improvement of semantics-preserving mapping over previous methods.

data (Table 3)[+]. Suppose an attribute, x, contains a primary key constraint. Then, the rules for mapping constraints are used to transform attribute x. First, attribute x is transformed by Rule 2 because it is a nonforeign key attribute. Second, attribute x is transformed by Rule 6 to assign an atomic value constraint of relational data. Finally, attribute x is transformed by Rule 9 to generate primary key constraint semantic data.

3.2. Problem description

In this section, we define two challenging problems that occur during the mapping of multi-column key constraints. The first is the generation of repetitive constraint data; the second is semantic information loss. Problem 1 illustrates the generation of repetitive constraint data during the mapping of multi-column key constraints, as follows:

Problem 1: Suppose r is a relational table, A is a set of attributes used to define a primary key (or a foreign key) of table r, and n is the number of attributes in A. To transform the semantics of multi-column key constraints, mapping by Rule 2 is performed n times, mapping by Rule 6 is performed n times, and Rule 9 for the primary key constraint (Rule 10 for the foreign key) is performed n times. Thus, n^3 mapping processes are required to transform a single multi-column primary constraint.

Problem 1 does not violate the semantics preservation of direct mapping. Rather, it outputs a repetitive and complex semantic data structure for a single multi-column key constraint. Thus, we define advanced rules to resolve Problem 1. Problem 2 illustrates the semantic information loss of mapping multi-column primary key constraints (comments on the foreign key constraints are omitted because the same description can be applied to the foreign key constraints), as follows:

Problem 2: Suppose r is a relational table, A is a set of attributes used to define a primary key of table r, and n is the number of attributes in A. To transform the semantics of multi-column key constraints, PKey(a1, r), PKey(a2, r),…, PKey(an, r) are performed by Rule 9. However, on a machine level, the results are understood as there being n primary keys in table r and not a single primary key that comprises a set of attributes.

Figure 3. illustrates the examples of Problem 2. If a single-column key is transformed by Rule 9, then output data are generated without semantic loss (Figure 3(a).). On the contrary, if a multi-column key is transformed by Rule 9, then constraint data are processed identically as a single-column key constraint. The results contain no explicit metadata such that it is a multi-column primary key (Figure 3(b). and (c).). As a result, transformed data (more than one primary key in a table) are not equivalent to original input data (a single primary key of a table). Thus, we define advanced rules to resolve Problem 2 in next section.

4. Improved approach

In this section, In this section, a set of rules are provided for mapping a multi-column primary key and a multi-column foreign key to solve the problems described in the previous section. Predicate logic is used to define rules, and graphical examples are provided for improved comprehension.

4.1. Base rules for key constraint mapping

The following rules are used for mapping primary key and foreign key constraints:

- Base primary key rule: NonFKeyAttr(*a*,*r*) ∧ FunctionalP(*a*) ∧ InverseFunctionalP(*a*) ∧ SubClassOf(*r*,_*b*) ∧ Card(*a*,_*b*,1) → PKey(*a*, *r*);
- Base foreign key rule: FKeyAttr(*a*,*r*,*s*) ∧ FunctionalP(*a*) ∧ SubClassOf(*r*,_*b*) ∧ MinCard(*a*,_*b*,1) → FKey(*a*,*r*,*s*),

[+] The predicates used in the rules are explained in the appendices.

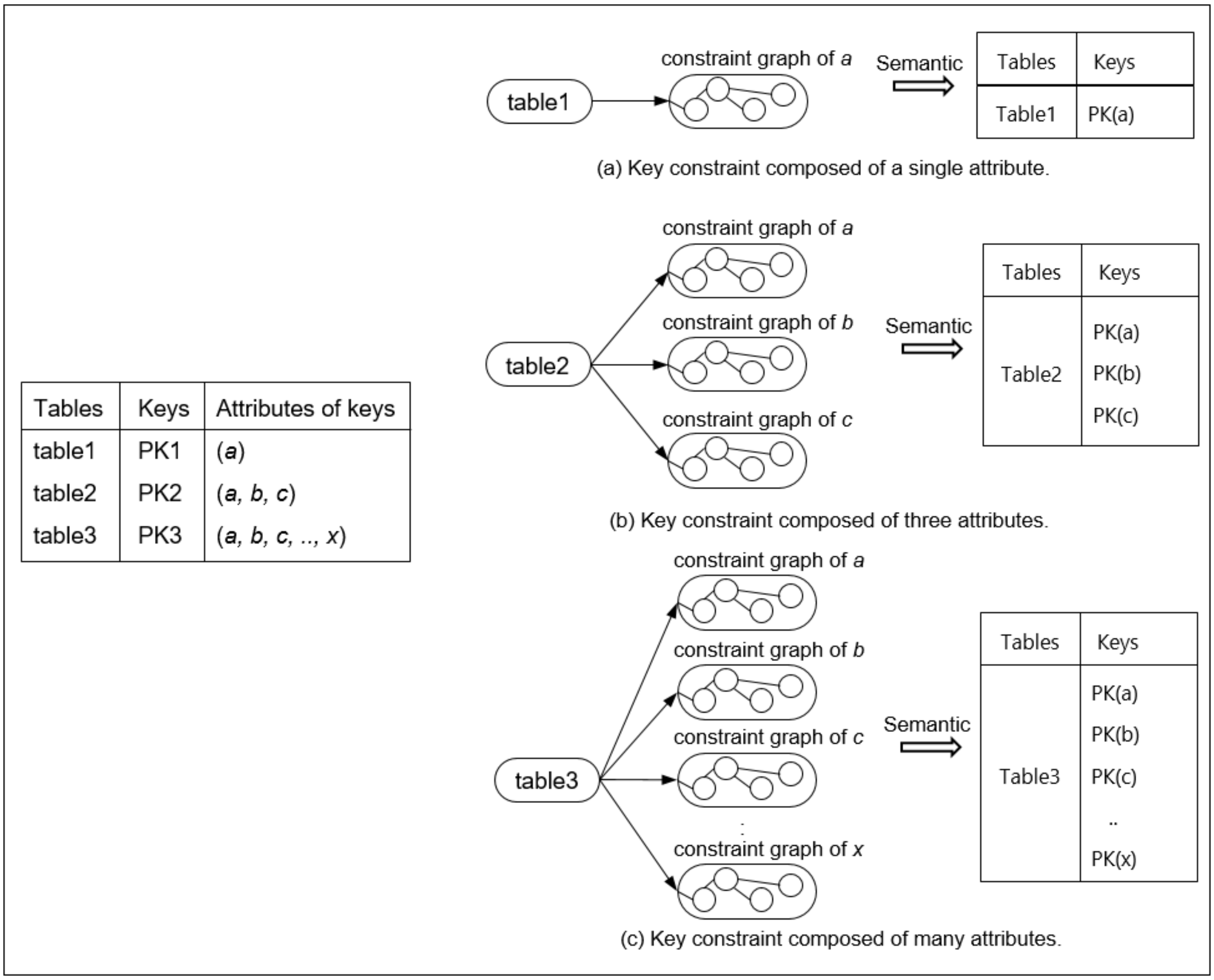


Figure 3. Examples of mapping primary keys

where *a* is an attribute of relational table *r*, *_b* is a blank node, *v* is an attribute value, and *s* is a table referenced by table *r*. A detailed explanation of the predicates on the left side is provided in the Appendix. The base primary key rule specifies the primary key by Card(*a*,*_b*,1) to assign attribute *a* with a primary key (Figure 4(a).). The base foreign key rule uses MinCard(*a*,*_b*,1) to specify a lower bound of the cardinality because relational tables can reference more than one other table. In addition, the base foreign key rule uses FKeyAttr(*a*,*r*,*s*) to describe the semantics of the type of attribute *a* being an OWL object property with domain r and range *s* (Figure 4(b).). Note that the two base key rules use FunctionalP(a), which is the OWL functional property, to notify an N:1 relationship to assign the constraint that attribute *a* must contain at most one attribute value. Therefore, the mapping by the base key rules can circumvent the processes of Rule 6 and thereby reduce the output size compared to the previous rules.

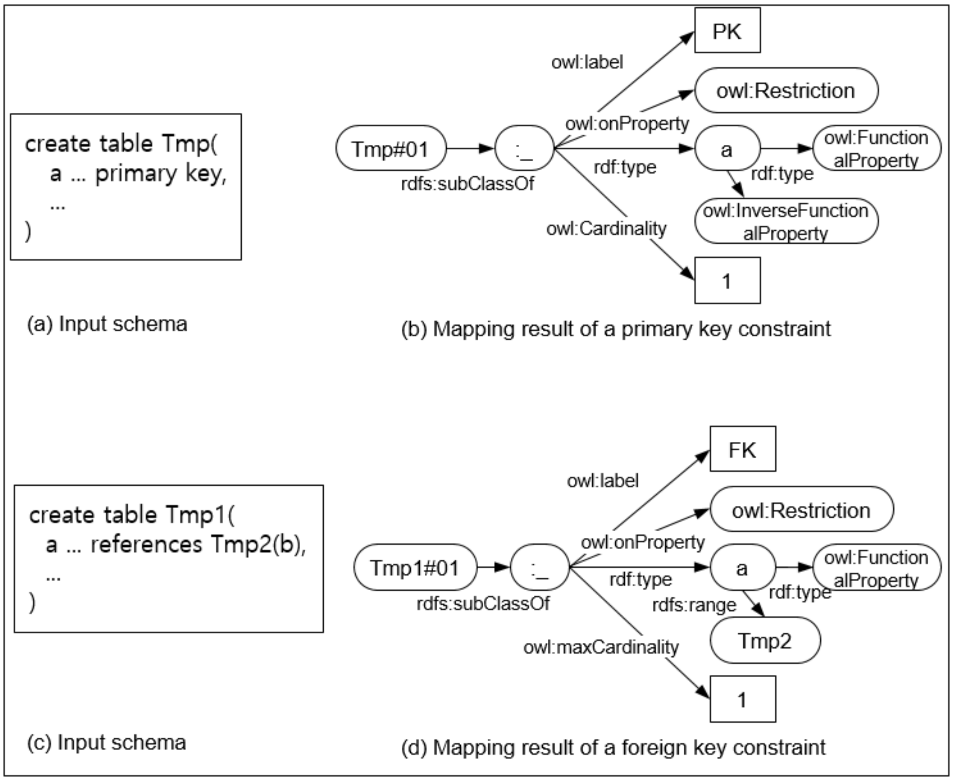


Figure 4. Example of mapping a single-column primary key and a foreign key

Problem 1 can be solved using the base key rules. However, Problem 2 still occurs during mapping with the base key rules. For example, if the base key rules are used to transform a table with a primary key or a foreign key comprised of multiple attributes, then multiple subgraphs that express key constraints are generated by the mapping rule (Figure 5.). The output result contains the repetitive subgraphs of constraint data and can be misinterpreted as a table having more than one key. To solve Problem 2, the base rules are modified in the next section using additional features.

4.2. Rules for relational integrity constraints

The following rules are employed for mapping the primary key and foreign key constraints without information loss:

- Primary key grouping rule: NonFKeyAttr(*A*,*r*) FunctionalP(*a*) ∧ InverseFunctionalP(*a*) ∧SubClassOf(*r*,*_g*) ∧ SubClassOf(*_g*,*_B*) ∧ Card(*A*,*_B*,1) → PKey(*A*,*r*);
- Foreign key grouping rule: FKeyAttr(*A*,*r*,*s*) ∧ FunctionalP(*a*) ∧ SubClassOf(*r*,*_g*) ∧ SubClassOf(*_g*,*_B*) ∧ MinCard(*A*,*_B*,1) → FKey(*A*,*r*,*s*),

where *A* is a set of attributes of which a multi-column key is composed, *r* is a relational table that contains *A*, *_B* is a blank node set that represents the key constraints of *A*, *_g* is a blank node for grouping constraints, and *V* is a value set of *A*. the details of the predicates on the left side are provided in the Appendix. The primary key grouping rule specifies a primary key using Card(*A*,*_B*,1) to assign attribute set *A* with a multi-column primary key.

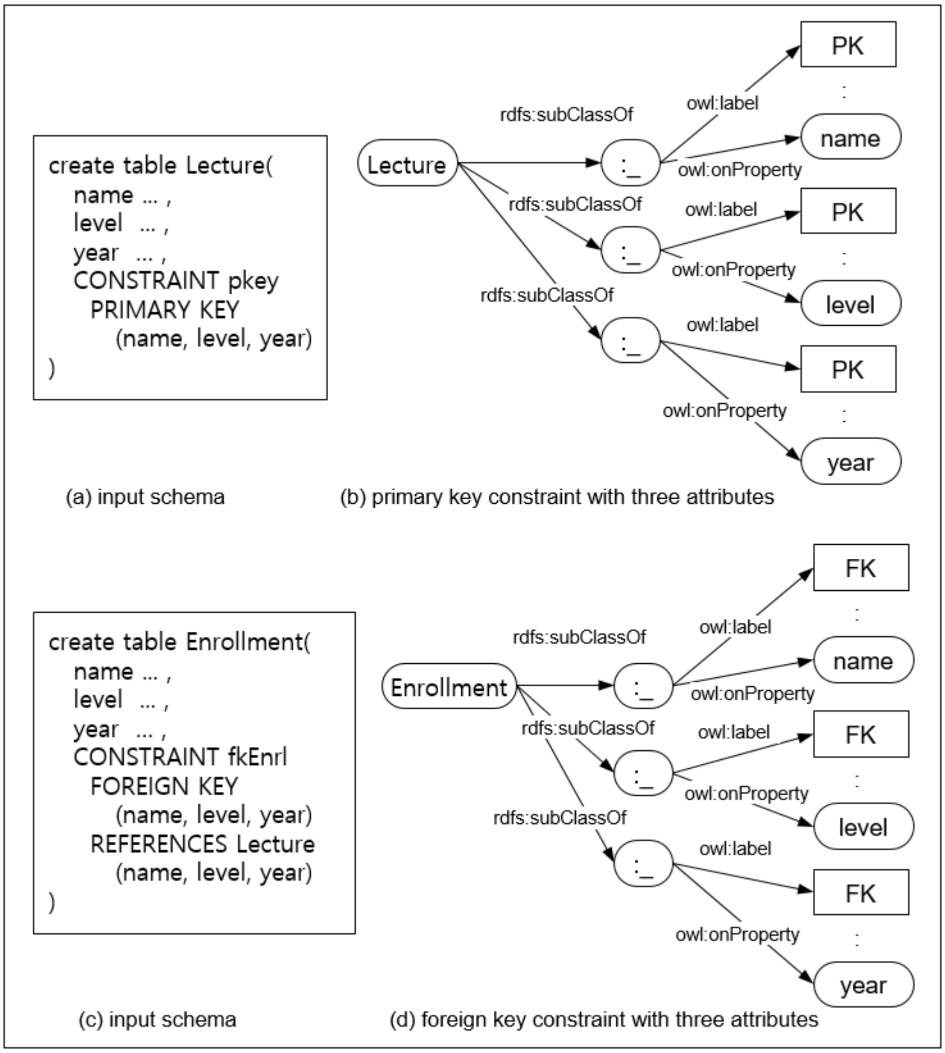


Figure 5. Transformation of a multi-column primary key and a foreign key using the base key rules

The foreign key grouping rule specifies a foreign key using MinCard(*A*,_*B*,1) to assign attribute set A with a multi-column foreign key. The output results contain merged key subgraphs to resolve Problem 2, i.e., the misinterpretation problem.

The repetitive data generated by Rule 6 can be reduced using the base key rules, and semantic information loss can be prevented by utilizing the grouping rules. However, repetitive constraint data generation still exists because of the semantic data structure of the multi-column key constraints that comprise two or more constraint subgraphs (Figure 6.). To reduce repetitive constraint data, the grouping rules are modified in the next section by employing separating mapping processes.

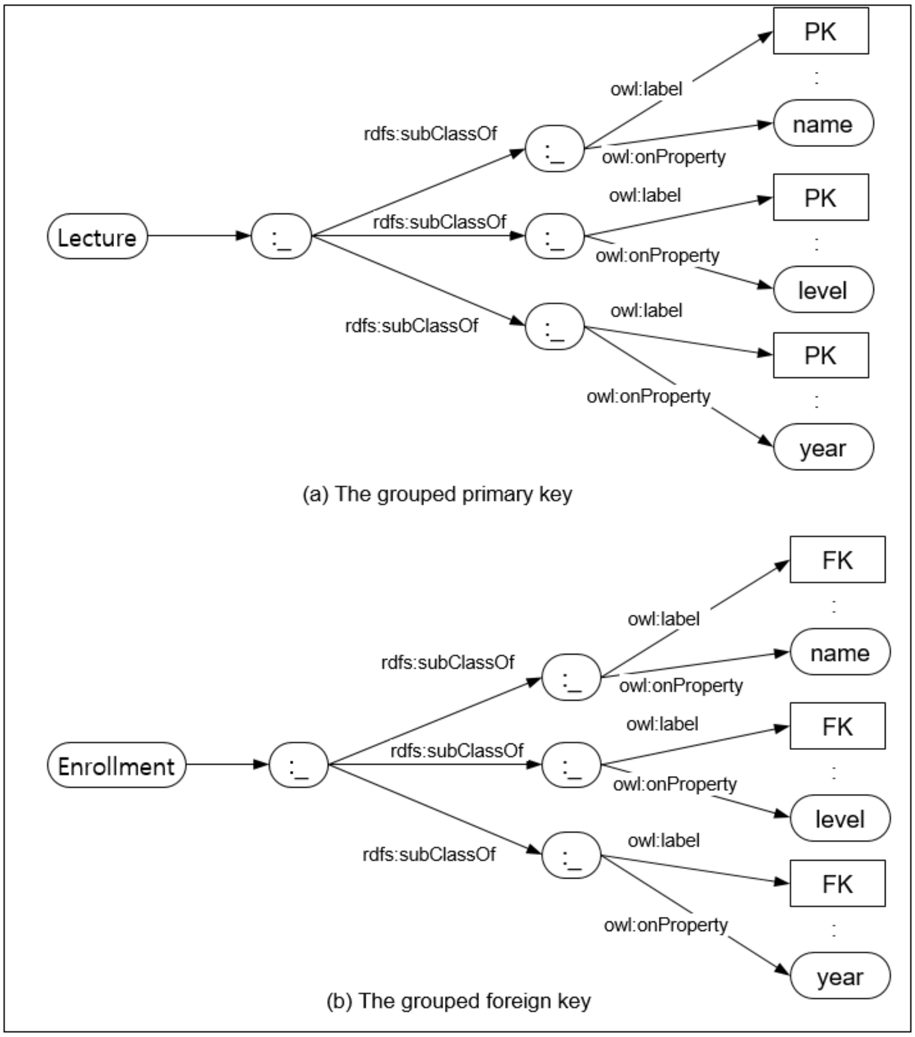


Figure 6. Example of transformation using the grouped primary key and foreign key rules

4.3. Optimized rules for multi-column key mapping

The following rules are employed for mapping primary key and foreign key constraints without information loss:

- Optimized primary key rule: NonFKeyAttr(*pkey*,*r*) ∧ FunctionalP(*a*) ∧ InverseFunctionalP(*a*) ∧ SubClassOf(*r*,*_b*) ∧ Card(*pkey*,*_b*,1) ∧ type(*A*,*pkey*) → PKey(*A*,*r*);
- Optimized foreign key rule: FKeyAttr(*fkey*,*r*,*s*) ∧ FunctionalP(*a*) ∧ SubClassOf(*r*,*_b*) ∧ MinCard(*fkey*,*_b*,1) ∧ type(*A*,*fkey*) → FKey(*A*,*r*,*s*),

where *A* is a set of the attributes of which a multi-column key is composed, *r* is a relational table that contains *A*, *pkey* is an OWL property used to represent a primary key constraint of *r*, *fkey* is an OWL property to represent a foreign key constraint of *r*, *_b* is a blank node, and *V* is a value set of *A*. The details of the predicates on the left side are presented in the Appendix. The optimized primary key rule specifies a primary key using Card(*pkey*,*_b*,1) and type(*A*,*pkey*) to assign attribute set *A* with the primary key.

The optimized foreign key rule specifies a foreign key using MinCard(*fkey*,*_b*,1) and type(*A*,*fkey*) to assign attribute set A with the foreign key. As shown in Figure 7., the optimized key rules output less constraint data. Instead of producing every key constraint subgraph (Figure 7(a). and 7(c).), each optimized key rule only generates a single key constraint, which is linked by key column attributes (Figure 7(b). and 7(d).). The output result contains a merged key subgraph to resolve Problem 2. Moreover, the rule generates compact output data with reduced repetitive constraint data; this solves Problem 1. Figure 8. describes an example of how the optimized rules are applied to actual tables.

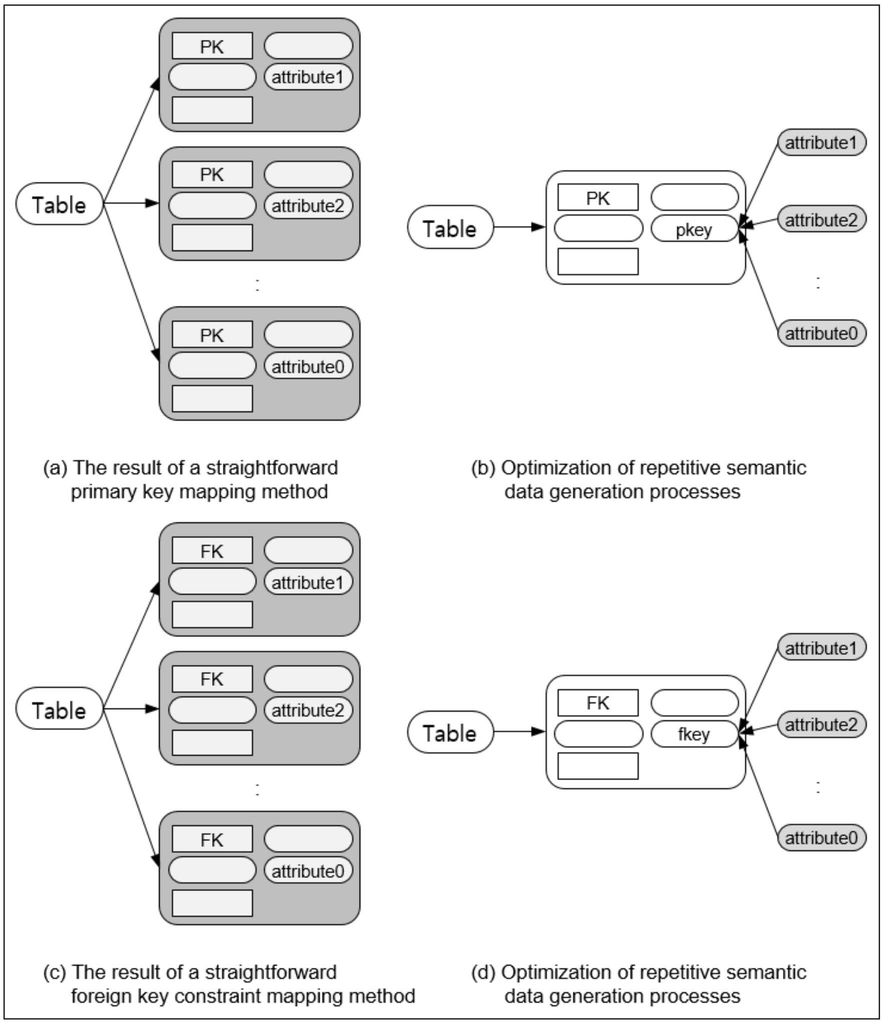


Table
PK
attribute1
attribute2
attribute0
pkey
(a) The result of a straightforward primary key mapping method
(b) Optimization of repetitive semantic data generation processes
FK
fkey
(c) The result of a straightforward foreign key constraint mapping method
(d) Optimization of repetitive semantic data generation processes

Figure 7. Transformation of a multi-column primary key and a foreign key using the optimized key rules

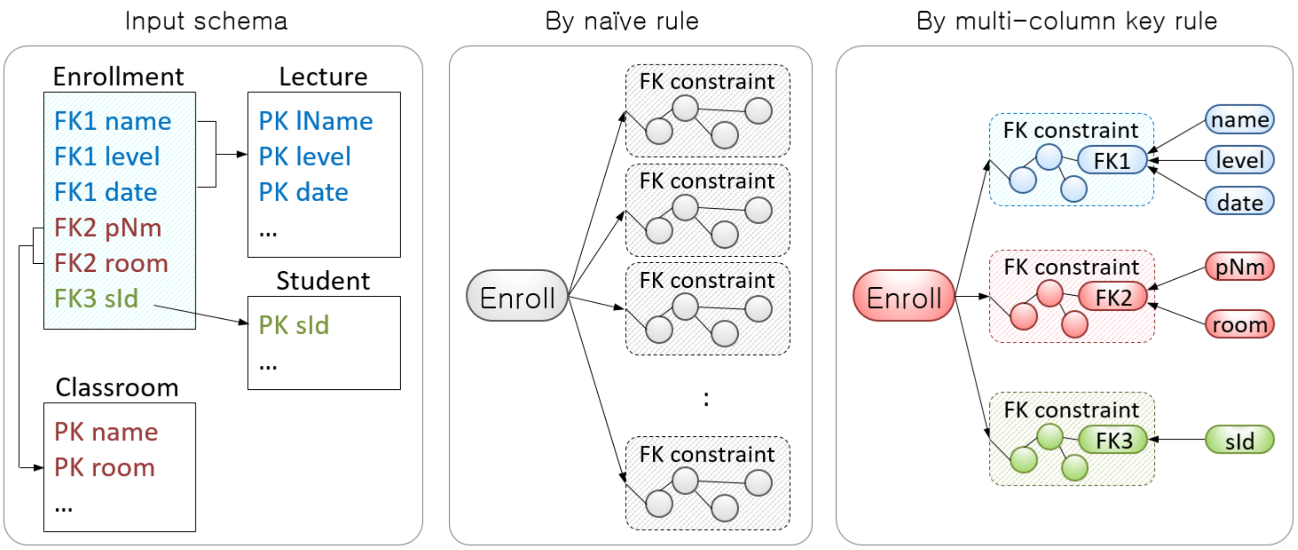


Figure 8. Example of transformation using the optimized key rules

5. Experimental evaluation

5.1. System setup

Evaluations were conducted using a single node of a 3.1-GHz quad-core CPU, 4 GB of memory, and a 2 TB hard disk. The experiments were conducted on real data and synthetic data. Each real data item was defined by relational schema, including integrity constraints: Ensembl-compara (DB1)[γ], Ensembl (DB2)[η], PHPmyadmin (DB3)[ι], and MusicBrainz (DB4)[φ]. The DBT2 benchmark was used as synthetic data[κ]. A synthetic dataset was generated by DBT2, and schema data were restructured by adding integrity constraints to evaluate the performance of mapping multi-column key constraints. The previous approach (OWL-ontology-based augmented direct mapping) was employed [13] to analyze the direct mapping methods.

5.2. Results

Figure 9. and 10. depict the performances of mapping multi-column primary key constraints and multi-column foreign key constraints, respectively. Single-column primary keys, multi-column primary keys, single-column foreign keys, and multi-column foreign keys are used as input data. The previous approach, base key rules, key grouping rules, and optimized key rules are compared.

In Figure 9(a)., the horizontal axis represents the size of the synthetic primary key constraint data (synthetic foreign key constraint data in Figure 10(a).). The vertical axis represents the number of semantic triples as output data. According to Figure 9(a). and 10(a)., our approach generates fewer triples compared to the previous method, and the optimized key rule generates the smallest output data. Assuming that two output results are identical in terms of semantics, the method that generates a result with a smaller size result is better with regard to space and computation.

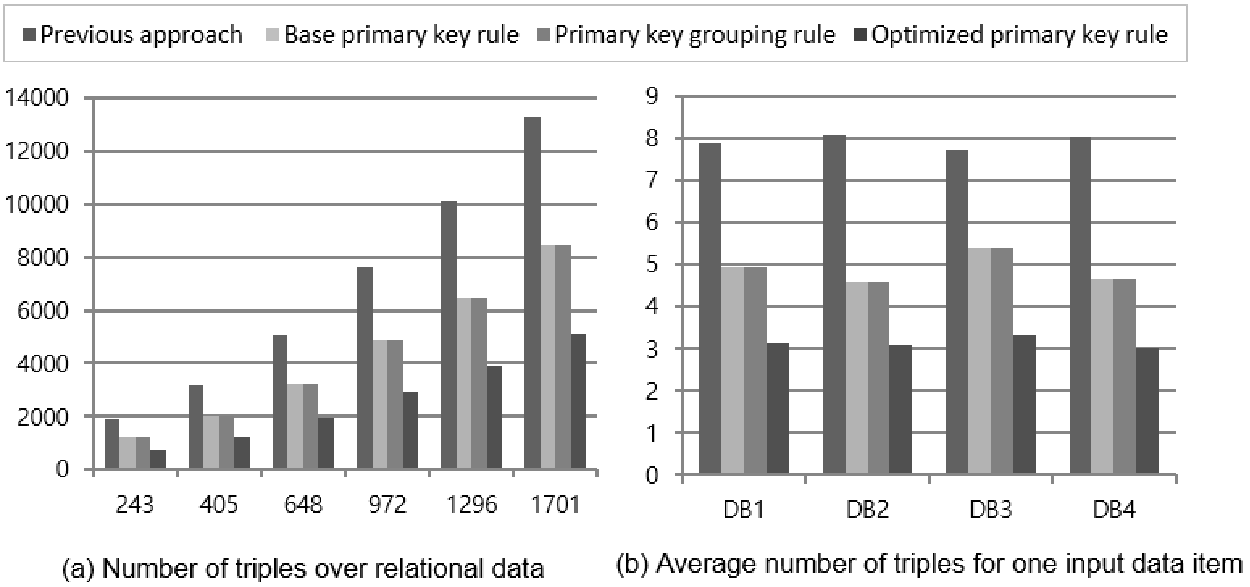


Figure 9. Comparative results of mapping multi-column primary key constraints

Figure 9(b). and 10(b). show the average number of triples obtained as a result of each transformation method. The horizontal axis represents the key constraint data (primary key constraints in Figure 9(b). and foreign key constraints in Figure 10(b).) from each real dataset. The vertical axis represents the average number of triples generated from

[γ] Ensembl Compara, www.ensembl.org/info/docs/api/compara

[η] Ensembl, www.ensembl.org

[ι] phpMyadmin, https://www.phpmyadmin.net

[φ] MusicBrainz, https://musicbrainz.org

[κ] DBT2, http://osdldbt.sourceforge.net transforming a key constraint. The results show that the optimized key rule, which reduces repetitive multi-column key constraint data, generates compact output data with fewer resources.

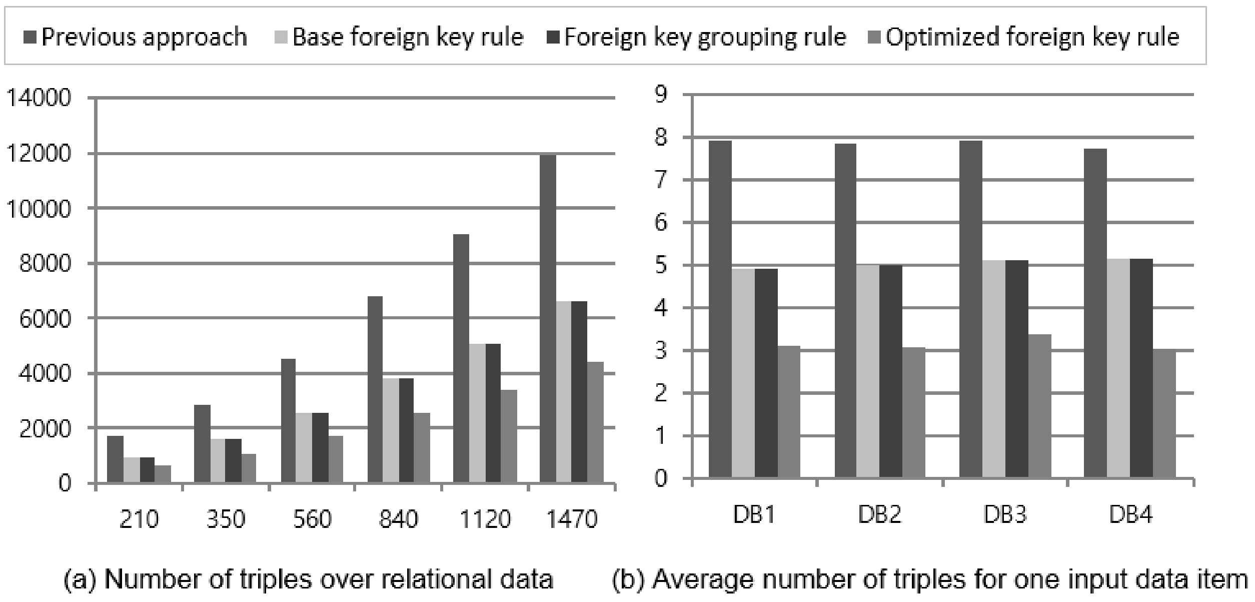


Figure 10. Comparative results of mapping multi-column foreign key constraints

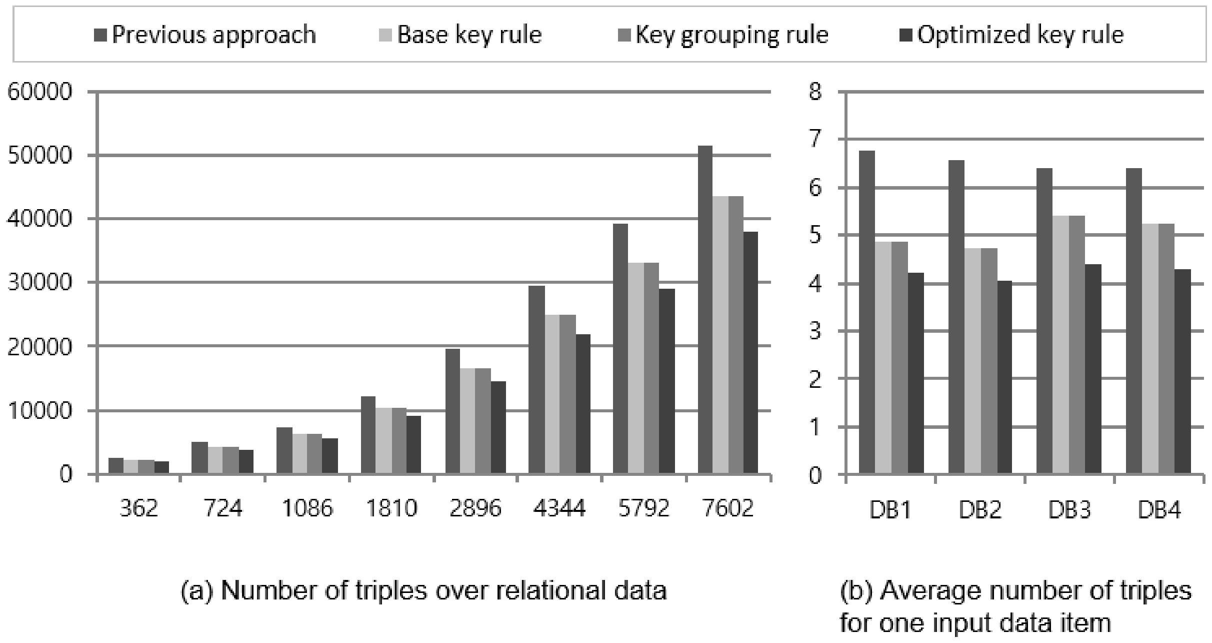


Figure 11. Comparative results of mapping rules

Figure 11. shows the performance of all mapping rules. In Figure 11(a)., the horizontal axis represents the size of the synthetic relational dataset as input data. The vertical axis is the number of semantic triples as output data. According to Figure 11(a)., the proposed approach generates fewer triples compared to the previous method and the optimized key rule generates the smallest output data.

Figure 11(b). shows the average number of triples obtained as a result of each mapping rule. The horizontal axis represents each real dataset; the vertical axis represents the average number of triples generated from transforming a single relational element. The results show that the optimized key rule generates the fewest output data among the mapping methods. In Figure 12., the failure rates of the mapping methods in each database are presented. The failure rates are calculated by dividing the total number of output data by the number of failed transformed output data with semantic information loss.

The horizontal axis represents each relational dataset; the vertical axis is the failure rate of the transformation of relational data into semantic data. The previous approach produces the most failed outputs because it lacks support for multi-column key constraints. In contrast, the proposed approach improves the mapping rules and generates fewer false mapping results. False mapping results could be generated by our method when input data are defined based on the practical SQL statements that are not included in the SQL standard. Figure 13. shows the evaluation of the performance of solving Problem 2. The key constraint mapping failure rates are calculated by dividing the total number of output key constraint data by the number of failed transformed key constraint output data with semantic information loss. The horizontal axis represents the total size of the synthetic key constraint data (single-column primary/foreign key constraint data and multi-column primary/foreign key constraint data). The vertical axis is the failure rate of the transformation of the key constraint data. The method that uses the base key rules is more semantic preserving compared to the previous approach, except for solving Problem 2. Thus, the previous approach and the method that uses the base key rules produce similar failed outputs in view of solving Problem 2. The method that utilizes the key grouping rules and optimized key rules is defined to transform multi-column key constraint data. The only difference between the key grouping rules and optimized key rules is the performance of reducing redundant output data. Hence, the two methods show better performance of solving Problem 2 compared to the previous approach. Consequently, we have observed that our method outputs compact semantic data and generates multi-column key constraints without information loss.

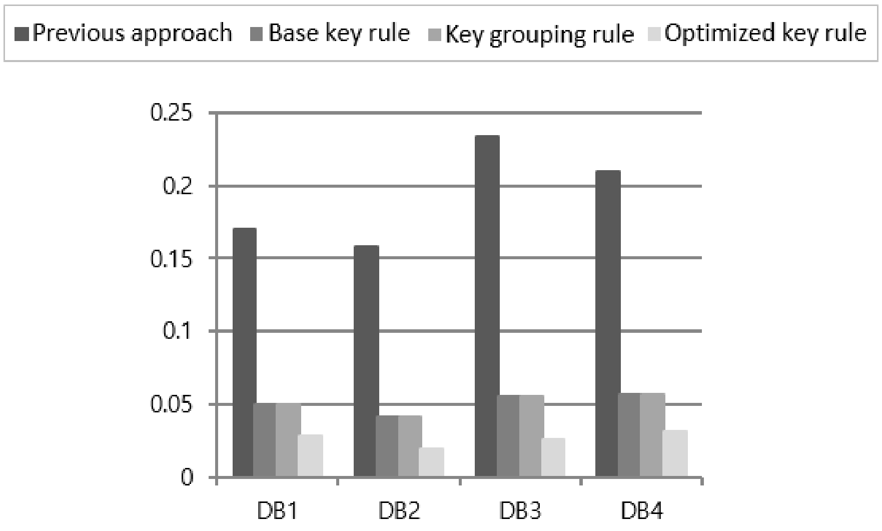


Figure 12. Results of failure rates of mapping

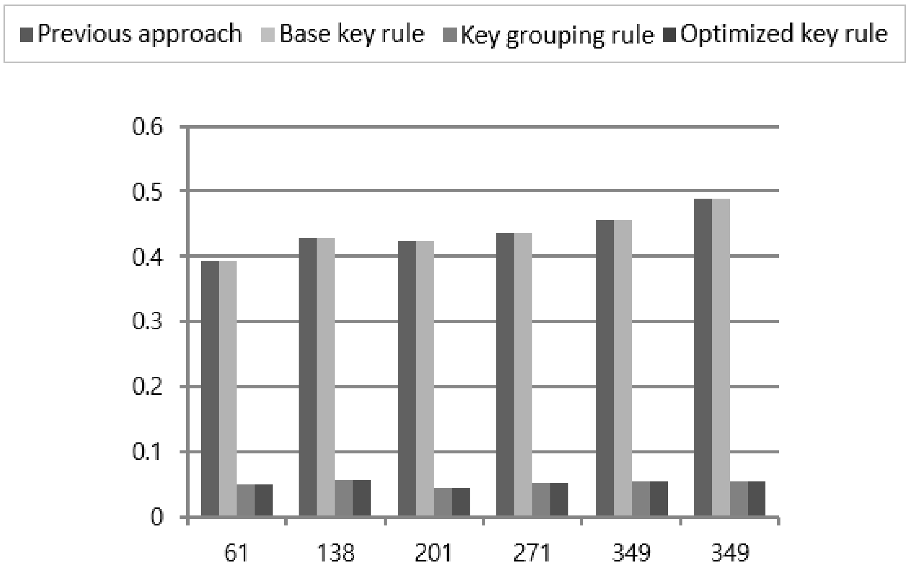


Figure 13. Results of failure rates of key constraint mapping

6. Conclusion

In this paper, we proposed an optimized direct mapping method to transform multi-column key constraints. A key constraint may comprise one or more attributes. Thus, repetitive data generation and semantic information loss can occur during mapping processes. Therefore, we therefore proposed a set of rules to regulate the mapping of multi-column key constraints. First, base key rules were defined for mapping the multi-column key constraints to be transformed with a smaller output data size. The rules covered primary keys and foreign keys when the constraints comprised two or more attributes. Second, key grouping rules were defined to prevent semantic information loss during the mapping of multi-column key constraints. Finally, we defined optimized key rules, which ensured that the mapping results contained only one constraint definition for one multi-column key constraint without repetitive data and semantic information loss. The results of an evaluation showed that the proposed approach generated significantly less output data while retaining original input data semantics without losing semantics information. The management of data consistency between updated relational data and semantic data remains a challenging issue for future research [33].

Acknowledgements

This work was supported by a grant from the National Research Foundation of Korea (NRF) funded by the Korean government (No. NRF-2017R1C1B1003600), and supported by the Ministry of Science and ICT (MSIT), Korea, with a program at the Information Technology Research Center (IITP-2020-2018-0-01417) supervised by the Institute for Information & Communications Technology Promotion (IITP).

References

[1] He B, Patel M, Zhang Z et al. Accessing the deep web. *Commun ACM* 2007;50(5):94-101.

[2] De Laborda CP and Conrad S. Database to semantic web mapping using RDF query languages. In: *Proceedings of International Conference on Conceptual Modeling*, Springer, 2006, pp. 241-254.

[3] Spanos DE, Stavrou P and Mitrou N. Bringing relational databases into the semantic web: A survey. *Semantic Web 2012;* 3(2): 169-209.

[4] Berners-Lee T. Relational Databases on the Semantic Web. *http://www.w3c.org/DesignIssues/RDB-RDF.html*, 1998;

[5] Sequeda JF, Arenas M and Miranker DP. On directly mapping relational databases to RDF and OWL. In: Proceedings of *International conference on World Wide Web. ACM, 2012*, pp. 649-658.

[6] Sequeda JF, Arenas M and Miranker DP. A completely automatic direct mapping of relational databases to RDF and OWL. In: Proceedings of ISWC*. 2011*.

[7] Li, M and Du XY and Wang S. Learning ontology from relational database. In: Proceedings of *International Conference on Machine Learning and Cybernetics*. *IEEE, 2005,* pp. 3410-3415.

[8] Shen G, Huang Z, Zhu, X et al. Research on the Rules of Mapping from Relational Model to OWL. In: Proceedings of *OWLED.* 2006.

[9] Astrova I, Korda N and Kalja A. Rule-Based Transformation of SQL Relational Databases to OWL Ontologies. In: Proceedings of *International Conference on Metadata & Semantics Research.* 2007.

[10] Cullot N, Ghawi R and Yetongnon K. DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In: Proceedings of *SEBD.2007*, pp. 491-494.

[11] Cerbah F. Learning highly structured semantic repositories from relational databases. In: Proceedings of *European Semantic Web Conference.2008,* pp. 777-781.

[12] Tirmizi SH, Sequeda J and Miranker D. Translating sql applications to the semantic web. In: Proceedings of *International Conference on Database and Expert Systems Applications*. Springer, 2008, pp. 450-464.

[13] Lim KB, Jun HG and Kim HJ. Semantics preserving MapReduce process for RDB to RDF transformation. *Int J Metadata Semant Ontol 2015; 10(4):*229-239.

[14] Manola F, Miller E, McBride B. RDF 1.1 Primer. *W3C Working Group Note, February 2014;*25.

[15] Hayes J. A graph model for RDF. *Darmstadt University of Technology/University of Chile,* 2004;.

[16] Coates T, Connolly D, Dack D et al. Uris, urls, and urns: Clarifications and recommendations 1.0. *World Wide Web Consortium, Note NOTE-uri-clarification-20010921* 2001;.

[17] Mallea A, Arenas M, Hogan A et al. On blank nodes. In: Proceedings of *International Semantic Web Conference,* Springer, 2011, pp. 421-437.

[18] Sahoo SS, Halb W, Hellmann S et al. A survey of current approaches for mapping of relational databases to RDF. *W3C RDB2RDF Incubator Group Report 2009;* 1: pp. 113-130.

[19] Michel F, Montagnat J and Faron-Zucker C. A survey of RDB to RDF translation approaches and tools. PhD Thesis, *I3S,* 2014.

[20] Byrne K. Having triplets-holding cultural data as rdf. In: Proceedings of *ECDL 2008 Workshop on Information Access to Cultural Heritage*.

[21] Green J, Dolbear C, Hart G et al. Creating a semantic integration system using spatial data. In: Proceedings of *International Conference on Posters and Demonstrations-Volume 401. CEUR-WS.org.* 2008, pp. 70-71.

[22] Das S, Sundara S, Cyganiak R R2RML: RDB to RDF mapping language. *http://www. w3. org/TR/r2rml/* W3C Recommendation 2012;.

[23] Hert M, Reif G and Gall HC. A comparison of RDB-to-RDF mapping languages. In: Proceedings of *7th International Conference on Semantic Systems. 2011,* pp. 25-32.

[24] Bizer C and Seaborne A. D2RQ-treating non-RDF databases as virtual RDF graphs. In: Proceedings of *International semantic web conference (ISWC2004),* 2004.

[25] Erling O and Mikhailov I. RDF Support in the Virtuoso DBMS. In: Proceedings of *Networked Knowledge-Networked Medi.*Springer, 2009, pp. 7-24.

[26] Sequeda JF, Depena R and Miranker DP. Ultrawrap: Using sql views for rdb2rdf. In: Proceedings of *International Semantic Web Conference 2009,* 2009.

[27] Arenas M, Bertails A, Prud E et al. A direct mapping of relational data to RDF. W3C recommendation 2012;.

[28] Brickley D, Guha RV and McBride B. RDF Schema 1.1. *W3C recommendation* 2014; 25:2004-2014.

[29] McGuinness DL and Van Harmelen F. OWL Web Ontology Language Overview. W3C Recommendation 2004. *Latest version is available at http://www. w3c. org/TR/owl-features.*

[30] Malik, KR, Ahmad T, Farhan M et al. Big-data: transformation from heterogeneous data to semantically-enriched simplified data. *Multimed Tools Appl 2016; 75(20):*12727-12747.

[31] Tong Q. Mapping Object-Oriented Database Models Into RDF(S). *IEEE Access 2018;* 6:47125-47130.

[32] Tong Q, Zhang F and Cheng J. Construction of RDF (S) from UML class diagrams. *J Comput Inf Technol 2014;22;*pp. 237-250.

[33] Im DH, Lee SW and Kim HJ. A version management framework for RDF triple stores. IJSEKE 2012; 22(1): 85-106.

Appendices

A. Predicates of OWL ontology

Predicates	Descriptions
Class(*r*)	*r* is an OWL class.
ObjP(*p, d, r*)	*p* is an OWL object property with domain *d* and range *r*.
DatatypeP(*p, d, t*)	*p* is an OWL data-type property with domain *d* and data-type *t*.
FunctionalP(*p*)	*p* is an OWL functional property.
InverseFunctionalP(*p*)	*p* is an OWL inverse functional property.
Card(*p*, *v*)	Cardinality of property *p* is *v*.
MinCard(*p*, *v*)	Minimum cardinality of property *p* is *v*.
MaxCard(*p*, *v*)	Maximum cardinality of property *p* is *v*.
Type(*x*, *t*)	Data type of *x* is *t*.

SubClassOf(*x*, *y*)	*x* is a subclass of *y*.

B. Predicates of relational data

Predicates	Descriptions
Rel(*r*)	*r* is a relation.
BinRel(*r*, *s*, *t*)	*r* is a binary relation between relation *s* and *t*.
FKeyAttr(*a*, *r*)	*a* is a foreign key attribute.
NonFKeyAttr(*a*, *r*)	*a* is not a foreign key attribute.
AllColumns(*r*)	Returns all columns of relation *r*.

C. Predicates of relational integrity constraints

Predicates	Descriptions
NotNull(*x*, *r*)	*x* is an attribute in relation *r* with not-null constraint.
Unique(*x*, *r*)	*x* is an attribute in relation *r* with unique constraint.
PKey(*x*, *r*)	*x* is a primary key of relation *r*.
FKey(*x*, *r*, *s*)	*x* is a foreign key of relation *r*, referencing relation *s*.
Default(*x*, *r*, *v*)	*x* is an attribute in relation *r*.
defValue(*x*, *r*, *v*)	*v* is a value of attribute *x* in relation *r* and is used in Default(*x*, *r*, *v*).