Transformation of a Relational Database to RDF/RDFS with ER\textsuperscript{2}iDM

Mi-Young CHOI\textsuperscript{1}, Nonmember, Chang-Joo MOON\textsuperscript{1(\dagger)}, Member, and Doo-Kwon BAIK\textsuperscript{2(\dagger)}, Nonmember

SUMMARY The Semantic Web uses RDF/RDFS, which can enable a machine to understand web data without human interference. But most web data is not available in RDF/RDFS documents because most web data is still stored in databases. It is much more favorable to use stored data in a database to build the Semantic Web. This paper proposes an enhanced relational RDF/RDFS interoperable data model (ER\textsuperscript{2}iDM) and a transformation procedure from relational data model (RDM) to RDF/RDFS based on ER\textsuperscript{2}iDM. The ER\textsuperscript{2}iDM is a data model that plays the role of an inter-mediator between RDM and RDF/RDFS during a transformation procedure. The data and schema information in the database are migrated to the ER\textsuperscript{2}iDM according to the proposed translation procedures without incurring loss of meaning of the entities, relationships, and data. The RDF/RDFS generation tool makes a RDF/RDFS XML document automatically from the ER\textsuperscript{2}iDM. The proposed ER\textsuperscript{2}iDM and transformation procedure provides detailed guidelines for transformation from RDM to RDF/RDFS unlike existing studies; therefore, we can more efficiently build up the Semantic Web using database stored data.

key words: Semantic Web, relational data model, relational database, RDF/RDFS

1. Introduction

The Semantic Web describes an intelligent web that can understand the content of the Web on behalf of humans and where computers can communicate with each other [1]. The Semantic Web uses resource description framework (RDF) and RDF Vocabulary Description Language (RDFS) [2], [3], which includes the data of Semantic Web and can enable a machine to understand web data without human interference, and thus correct RDF/RDFS generation is important part of Semantic Web [4]. But most web data is not available in RDF/RDFS documents because most web data is still stored in databases [5], [6]. It is much more favorable to use stored data in a database to build the Semantic Web; therefore, the RDF/RDFS document generation from the data of a relational database is important research topic regarding the Semantic Web.

This paper proposes an enhanced relational RDF/RDFS interoperable data model (ER\textsuperscript{2}iDM) and a transformation procedure from RDM to RDF/RDFS based on ER\textsuperscript{2}iDM. The ER\textsuperscript{2}iDM is a data model that plays the role of an inter-mediator between RDM and RDF/RDFS during a transformation procedure. The ER\textsuperscript{2}iDM is designed to effectively store the RDF data of databases based on Lee’s approach [5] which is accepted starting point for transformation from RDB to RDF/RDFS and EAV data model [6] which enable only not null values of the RDB to be transformed to RDF.

The ER\textsuperscript{2}iDM can store the database schema information including the hierarchical structure of entity and attribute. The database schema information is used to generate RDFS which supplies understanding about RDF data [4]. The data and schema information in the database are migrated to the ER\textsuperscript{2}iDM according to the proposed translation procedures without incurring loss of meaning of the entities, relationships, and data. The proposed transformation procedure is divided into schema information migration and data migration to provide detailed transformation guidelines. The RDF/RDFS generation tool automatically makes RDF/RDFS XML document using the migrated data of ER\textsuperscript{2}iDM. The proposed ER\textsuperscript{2}iDM and transformation procedure provide detailed guidelines for transformation from RDM to RDF/RDFS, and thus these are more reliable than existing researches. The proposed ER\textsuperscript{2}iDM, transformation procedure and RDF/RDFS generation tool enable us to build up the Semantic Web efficiently using database stored data.

The remainder of the paper is organized as follows: Sect. 2 summarize related research about RDB2RDF. Section 3 present the ER\textsuperscript{2}iDM and explain each of its components. Section 4 presents the migration steps from RDM to the ER\textsuperscript{2}iDM while Sect. 5 present RDF/RDFS XML document generation using SQL and a tool from ER\textsuperscript{2}iDM. Section 6 present the evaluation and verification of ER\textsuperscript{2}iDM and the transformation procedure. Finally, in Sect. 7, we present our conclusions.

2. Related Research

Lee’s research [7] is to map the primary key (PK) to the subject, mapping column to the predicate and mapping the value of the column to the object. The existing researches [5], [8]–[16] which were studied based on Lee’s research [7] only consider simple mapping between the relational database and RDF. This approach generates RDF statements from single table data and does not consider the hierarchical structure of RDM. Shufeng [5] and D2R [8], [15], [16] add the class concept, which is that the table of the
The relational database (RDB) and RDF/RDFS document. The data of RDB is migrated in ER\textsuperscript{2i}DM and RDF/RDFS is automatically generated from ER\textsuperscript{2i}DM by the tool. The components of ER\textsuperscript{2i}DM are designed for efficient RDF/RDFS generation using the RDM. Figure 1 shows the ER\textsuperscript{2i}DM as an entity relationship diagram (ERD). The notations of the ER\textsuperscript{2i}DM follow Barker’s method [22]. A round cornered box is an entity, and an entity in an entity is a subtype entity, and subtype entities are grouped within a rectangular box. The notation # refers to the identifier (the PK in the table); the notation * indicates the attributes are not null; and the notation o means the attributes are nullable. To represent the relationships between entities, the crow’s foot notation [23] is used.

3. Components of the ER\textsuperscript{2i}DM

According to the basic concept of the RDF, the ER\textsuperscript{2i}DM defines resources and properties independently and properties are separated into characteristics of resource and relationships between resources. Independent entities of the ER\textsuperscript{2i}DM, Resource, is mapped to resources of RDF. Property and Predicate, are mapped to resources, properties, and predicates of RDF. Also, associative entities of the ER\textsuperscript{2i}DM, ResourceProperty and ResourcePredicate, represent the concepts of the RDF where resources have properties having values and relationships between resources are predicated.

The Resource manages all entity names, hierarchical structure information, and the values of the identifiers of each entity. The names of entities are migrated to ‘Class’ subtype entities and the values of the identifiers of entities are migrated to ‘Instance’ subtype entities. Resource of RDF can be grouped in a hierarchical structure; a recursive relationship named higherClassID of the Resource is used to group hierarchical structured entities in ER\textsuperscript{2i}DM.

![Diagram of ER\textsuperscript{2i}DM](image-url)
Namely, if an entity whose ResourceType is ‘Class’ exists on hierarchical structure entity, then the higher entity’s name is stored at higherClassID of lower entity. If ResourceType is ‘Instance’, the entity’s name which the identifier (ResourceName) value belongs to is stored at higherClassID.

The Property manages all names of attributes of all entities, which are stored at Resource. Generally, attributes are located inside of entities, but in the ER²iDM all names of attributes of entities are saved to Property except for identifiers and identifier name, which are saved at the ResourceID and the ResourceName of the Resource. For example, customer numbers attribute and customer names attribute are saved at the ResourceID and the ResourceName of the Resource, while the values and names of customer address attribute and customer telephone number attribute are saved to the Property.

The Predicate manages relationship names of the RDM. Relationships always exist between entities having names [24]. Relationships names can exist as more than one type. For example, an ‘order’ as a noun is the relationship name between a customer and product. But an ‘orders’ as active voice or ‘be ordered’ as passive voice also can be the relationship name at the position of customer or at the position of product. The Predicate has a subtype classifier named PredicateType whose value is ‘PredicateProperty‘ or ‘Single’. If a relationship names is noun or active voice, the value of PredicateType become ‘PredicateProperty‘. If a relationship name is passive voice, the value of PredicateType become ‘Single‘.

If relationship name is not given, the names of associative entities, dependent entities, and foreign keys can be relationship. For example, an ‘order’ can be a name of an associative entity that resolves the many-to-many relationship between a customer entity and a product entity. Therefore if the relationship names are not given in the RDM, the name of associative entities or dependent entities become a relationship names and the value of PredicateType become ‘PredicateProperty‘. Dependent relationships can exist between relationships of RDM. For example, a ‘delivery‘ occurs after an ‘order’, and therefore a ‘delivery‘ has a dependency with an ‘order‘. A dependent relationship is specified by the higherPropertyID which represent the hierarchical structure between relationships, and the precedence relationship name is stored at higherPropertyID.

The ResourceProperty is composed of three attributes, same as a RDF triple. The ResourceID, PropertyID, and the PropertyValue of the ResourceProperty are mapped to the subject, property, and property value of the RDF. Property only manages names of attributes. This approach can prevent unnecessary RDF creation because PropertyID coding properties is saved separated table so null property value is not inputted in ResourceProperty and this approach can be applied all database table so provide extendability and generality. The ResourcePredicate is composed of three attributes, which all are foreign keys from the Resource and Predicate. The ResourceID, the PredicateID, and the OtherResourceID of the ResourcePredicate are mapped to subject, predicate, and object of the RDF.

Likewise, the entities’ information including the hierarchical structure and identifier’s value of entity are migrated to the Resource. The relationships information including the hierarchical structure is migrated to the Predicate. All information of attributes is migrated to the Property. The separated information of entities and relationships are connected at the ResourcePredicate. The separated information of entities and properties are connected at the ResourceProperty and also values of properties are migrated to the ResourceProperty. The schema information related to relationship cardinality is not migrated to the Predicate. Many-to-many relationships should not appear in the RDM, which is implemented to the real world RDB, and 1-to-1 relationship is a subset of 1-to-many relationships. Therefore, most relationship cardinalities can be integrated into 1-to-many cardinality. Consequently, almost all schema information of the RDM and data of the RDB are migrated to the five components of the ER²iDM.

3.2 RDFS Description Using ER²iDM

The ER²iDM also supports RDFS transformation from RDM. RDFS helps interpretation of RDF data and supplies good understanding about RDF data, so RDFS expands the inference mechanism which makes a computer system more autonomous and smarter. Inference supplies information about data groups instead of independent data. RDFS can be generated from the Resource and the Predicate, which have information of the entity group and the relationship group in the ER²iDM.

According to RDFS description, the groups of resources should be represented as classes and should be defined separately from attributes [25]. The names of entities are stored at the Resource without having attributes, and the rdfs:Class can be defined with these entities’ names. The higherClassID in the entities hierarchical structure is used to define rdfs:subClassOf. The hierarchical structure between predicates are defined using rdfs:subPropertyOf. The names of higher relationships are stored in the higherPropertyID and are used to define rdfs:subPropertyOf.

4. The Transformation from RDM to RDF/RDFS

To build a more correct meaning of a Semantic Web using RDB data, the logical meaning of RDM has to be transformed to RDF/RDFS [8]. Figure 2 illustrates the transformation from RDM to RDF/RDFS graphically.

T₀, which transforms RDM to RDF/RDFS, is the final target of this paper. But it is difficult to directly transfer from RDM to RDF/RDFS because the structures of RDM and RDF/RDFS are very different. The ER²iDM is placed between RDM and RDF/RDFS to handle the differences.

Figure 3 shows the transformation for T₁, which is composed of two phases. At phase 1, whole schema information including entities, attributes, and relationships of
RDM is migrated to five components of the ER²iDM. The schema information is reconstructed in the ER²iDM according to the detail procedure such as PR, R1, R2, PD1, PD2 and RPD in Fig. 3. At phase 2, RDB data are migrated to the ER²iDM according to the detail procedure such as R3 and RPR in Fig. 3.

Figure 4 shows the RDM of shopping mall business to explain the T₁ transformation. Figure 5 shows sample data of RDM in Fig. 4.

The Customer manages customers’ data, and customers are divided into ‘VIP’, ‘Black List’, and ‘Normal’ subtypes. Figure 5 (a) shows sample data of Customer. The RelatedWorker manages the workers’ data and the WorkerType subtype classifies workers into Receptionist, Deliverer, and others. Figure 5 (b) shows the sample data of RelatedWorker. The Product manages products of a shopping mall, and products are divided into DVD and Book, and books are divided into paper, audio, and e-book. Figure 5 (c) shows sample data of the Product. The Customer, the RelatedWoker, and the Product are hierarchical entities having subtypes. The Order, an associative entity, manages the customers’ order information. Figure 5 (d) shows sample data of the Order. The Delivery has a dependent relationship on the Order, because a delivery occurs after an order has occurred. Figure 5 (e) shows sample data of the Delivery. The Return and the Refund manage the information related to return and refund, and the Return and the Refund have dependent relationships on the Delivery and the Return, respectively. Figure 5 (f) and Fig. 5 (g) present sample data of the Return and the Refund.

### 4.1 The Transformation Procedures from RDM to the ER²iDM

This section shows the T₁ transformation procedures from RDM to the ER²iDM according to Fig. 3’s two phases using Fig. 4’s RDM and Fig. 5’s sample data.

**Phase 1: Migration of RDM schema information**

**R1. Migrate the entities’ names to the ResourceName of the Resource**

Migrate all entities’ names including subtype entities to the ResourceName of the Resource, giving sequence numbers to the ResourceID. At this time, ‘Class’ is stored to ResourceType and will be used to assert rdfs:Class.

**R2. Store the ResourceIDs of higher entities in higherClassID**

The entities having the hierarchical structure are naturally migrated the Resource and the ResourceIDs of higher entities are stored in the higherClassIDs. Figure 6 presents the
result of R1 and R2 step. The ResourceIDs of Customer, RelatedWorker, and Product which are has the hierarchical structure are assigned to higherClassID of subtype entities.

PD 1. Migrate the relationships’ names to the PredicateName of the Predicate

All relationship names, associative and dependent entity names, and foreign key names, are migrated to the PredicateName of the Predicate giving sequence numbers to the PredicateID. At this time, ‘PredicateProperty’ value is assigned in the PredicateType.

PD 2. Store the PredicateID of higher relationships in the higherPropertyID

Dependent relationships are integrated into a recursive relationship in the RDM. That is, PredicateID of higher relationships is stored on the higherPropertyID in the Predicate. If passive voices are added to Predicate as the relationship, the value of the PredicateType is ‘Single’ and the PredicateID of infinitive relationship of added relationship are stored to higherPropertyID.

Figure 7 shows the results of the PD1 and PD2. The relationship dependencies among Order, the Delivery, the Return, and the Refund are specified by recursive relationship and the PropertyIDs of predicate property type relationship is assigned to higherPropertyIDs of all passive relationship.

RPD. Link entities’ names of the Resource and relationships’ names of the Predicate

The subject and object of the RDF which correspond to entities in the RDM are migrated to the Resource without separation. Therefore, subject and object of Resource are linked by relationship of Predicate in the ResourcePredicate. As a result, the components of the ResourcePredicate are mapped to RDF triple components.
In Fig. 8, if the values are 20, 8, and 28 for the ResourceID, the PredicateID, and the OtherResourceId, then they denote “Karl ordered Harry Potter 1 (e-book)”.

**PR. Migrate the attributes’ names to the PropertyName of the Property**

Migrate all attributes’ names of entities to the PropertyName of the Property giving sequence numbers to the PropertyID. At this time, identifiers and the identifiers’ names of independent entities and foreign keys of associative entities and dependent entities are excluded. That is, the CustomerID and the CustomerName of the Customer are excluded, and the CustomerID, the ProductID, and the ReceptionistID of the Order also are not migrated to the Property. Figure 9 shows the results of the sample data of PR.

In PR and PD1 steps, the attribute-value pair of two properties is saved in Property and Predicate as one row. Then NULL values do not exist in two tables and some attributes and relationships do not need to transform to the RDF/RDFS.

**4.2 Phase 2: Migration of RDB Data**

**R3. Migrate the values of identifiers and identifiers’ names to the ResourceID and the ResourceName of the Resource**

The values of identifiers of RDB tables are changed to the ResourceID of the Resource according to specific rules such as sequence numbers. The identifier names of independent entities are migrated to the ResourceName. But most associative and dependent entities do not have identifier names; therefore, the concatenations of entity’s name and identifier’s value are stored in the ResourceName. ResourceType is an ‘Instance’ and the higherClassID means the entity’s reference, which includes the value of ResourceName. Figure 10 presents the results of the sample data of R3.

**RPR. Link attributes’ names and values of the Property according to the values of the ResourceID**

The property name and value are linked according to
the ResourceID which has ‘Instance’ type value in the ResourceProperty. The property is the attribute of ResourceID referencing table and has not null value. In Fig. 11, if the values are 25, 4, and $25 for the ResourceID, the PropertyID, and the PropertyValue, then they indicate that “Harry Potter 1 (DVD)’s price is $25”.

5. RDF/RDFS XML Document Generation

This section shows T2 transformation for the RDF/RDFS document generation from the data of the ER2iDM tables.

5.1 RDF Statement Generation

All tables of the ER2iDM are implemented on the RDB, and then RDF statements can be generated with SQL. Figure 12(a) and Fig.12(a) are the SQL and the RDF statements as results, which mean <subject> <property> <property’s value> [25] using data of the ResourceProperty. Figure 12(c) and Fig. 12(d) are the SQL and the RDF statements as results, which mean <subject> <predicate> <object> [2] using data of the ResourcePredicate. During this procedure, the values of the ResourceID, the PropertyID, the PredicateID, and the OtherResourceID are numbers such as 1, 2, 3, …. Therefore, these values need to be labeled with names. With Join SQL, the values of the identifiers can be labeled with names instead of rdfs:label RDFS vocabulary, since these names have been stored at the ResourceName, the PropertyName, and the PredicateName.

5.2 RDFS Statement Generation

The separated hierarchical structure information of entities and relationships is migrated to the Resource and the Predicate using the recursive relationship of the ER2iDM. Migrated set information of entities and relationships is mapped to set information of classes and predicates. This section describes the assertion method for rdfs:Class, rdfs:subClassOf, and the rdfs:subProperty using the set information of the Resource and the Predicate.

1) Assertion of classes

All entities of the RDM comprise an independent data set, while the basic component for a set is a rdfs:Class in RDFS. Therefore, all entities of RDM are mapped to classes of RDFs. In R1 of Phase 1, ‘Class’ is given to the ResourceType of all migrated entities including subtypes.
Therefore, rdfs:Class can be asserted by connection of character strings such as ‘rdf:type’ and ‘rdfs:Class,’ to the values of the ResourceName where ResourceType is ‘Class’, as presented in Fig. 13.

2) Assertion of sub classes
The subtype entities are asserted to subclasses in the same manner as classes. The entities on the hierarchical structure are specified with recursive relationships in the ER2iDM. With a recursive query, the data having a hierarchical

```sql
select '':|||ResourceName subject, 'rdfs:subClassOf' predicate, '':|| prior ResourceName object
from ( select * from Resource
where ResourceType = 'Class')
connect by prior ResourceID = higherClassID
start with higherClassID = 2
```

Fig. 14 rdfs:subClassOf generation.

```sql
select '':|||PredicateName subject, 'rdfs:subPropertyOf' predicate, '':|| prior PredicateName object
from ( select * from Predicate
where PredicateType = 'PredProp')
connect by prior PredicateID = higherPredicateID
start with higherPredicateID = 2
```

Fig. 15 rdfs:subPropertyOf generation.

(a) RDF/RDFS generation windows

(b) RDF triple window

(c) ER2iDM access setting

---

Fig. 16 RDF/RDFS generation tool.
structure can be represented as a tree structure [26]. The hierarchical structures of entities of Fig. 4 are transformed to rdfs:subClassOf with a recursive query in Fig. 14.

3) Assertion of sub properties
In PD1 and PD2 of Phase 2, the names of associative entities, dependent entities, and foreign keys are migrated to the Predicate instead of the names of relationships. During migration, the business dependencies of the RDM are specified with recursive relationships. The RDFS statement for subproperties can be generated with a recursive query SQL in the same manner as sub classes. The dependent relationships of Fig. 4 are transformed to rdfs:subPropertyOf in Fig. 15.

5.3 RDF/RDFS Generation Tool
Figure 16 (a) shows the RDF/RDFS generation tool. The tool receives the SQL of Sects. 5.1 and 5.2 as input and query data of ER 2iDM in order to make RDF/RDFS XML document. Actually XML form document is needed to provide Semantic Web service. The left window of the tool is SQL input window. The user can input SQLs in left windows considering need data order for generating RDF/RDFS XML document. The RDF/RDFS XML document is displayed in right window of the tool when user push “Generate” button after finished SQLs input. The user can review content of RDF/RDFS XML document in right window and user also can check RDF triple using “View Triples” button which is located on right top of tool. Figure 16 (b) show the RDF triple after pushing “View Triples” button. If the created RDF/RDFS XML document is correct, user can save displayed RDF/RDFS XML document as text file form using “Save” button. The “Setting” button is used in order to configure information of ER 2iDM data access like as Fig. 16 (c). The T2 transformation can be achieved full automatically when we use proposed RDF/RDFS generation tool.

6. Evaluation and Verification
6.1 Qualitative Evaluation
Table 1 shows qualitative evaluation of ER 2iDM and existing studies that are mentioned in Sect. 2. Almost existing studies mentioned the RDF transformation but the RDFS transformation is restrictively addressed about Class and Property or not addressed. Especially existing studies do not mention about more important evaluation items which are hierarchical structure such as subClass and subProperty, detail transformation procedure and auto transformation. But this paper approach need to more database knowledge than existing studies.

6.2 Comparison with D2RQ
D2RQ [16], [28] offers RDF-based access to content of relational databases using SPARQL or Jena API without having to replicate it into an RDF store. Also D2RQ can create custom dumps of the database in RDF formats for loading into an RDF store. D2RQ research is progressed Open Source software unlike other studies and provides several tools such as D2R Server, generate-mapping, d2r-query and dump-rdf. But D2RQ only contemplates simple-mapping which is addressed in Sect. 2, and can do a simple query for database using SPARQL. Also D2RQ is needed for database expert intervention like ER 2iDM, in order to access meaningful RDF information at a specific domain database, because the important semantic information such as the hierarchical structure of RDB should be an applied RDF application.

ER 2iDM maintains RDF/RDFS data which is extracted from the database and is migrated in an extra database table such as ResourceProperty, ResourcePredicate. Therefore ER 2iDM has the burden of managing copied data in the database. But ER 2iDM can more effectively manage RDF/RDFS information only using SQL which is a more power tool than SPARQL and Jena API in database. Also ER 2iDM can more effectively query using RDFS information, unlike the D2RQ. A RDF/RDFS XML document is generated by the RDF/RDFS generation tool which is similar to the dump-rdf tool of D2RQ, and thus the semantic web application can do an inference function, such as the Sect. 6.3 verification and inference on the ER 2iDM. D2RQ is suitable for frequently changed database content because of don’t maintain copied data whereas ER 2iDM is suitable for infrequently changed database content.

6.3 Verification and Inference on the ER 2iDM
This section verifies the ER 2iDM and performs a simple inference with the resulting RDF statements, which do not exist in the RDB data and shows that the generated RDFS statements keep the W3C’s regulations related the meaning of the RDFS vocabulary.

The RDF Graph of Fig. 17 was implemented with transformed RDF statements. “Karl ordered product by OrderID 1”, and after “Karl returned the order by ReturnID 1” in the graph like as sample data of Fig. 4. In the other sample, the customer as an orderer can be traced from the ‘ReturnID 1’. Therefore, semantic loss does not occur...
During the transformation procedures.

For an inference, if ‘Complain’ property, “Return rdfs:subPropertyOf Complain”, “Refund rdfs:subPropertyOf Complain”, and “Complain rdfs:domain BlackList” are added to the RDF graph of Fig. 17’s, then “Complain rdfs:domain Customer” can be inferred. Also, with these sample data “Karl return OrderID1” and “Karl refund OrderID1”, then “Karl complain OrderID1” can be inferred.

A class is a type or a category for the grouping of diverse resources, and a subclass has an is-a relationship with a higher class, and a subclass inherits characteristics of a higher class and uses the inherited characteristics to represent a specific concept.

If the C1 class is a subclass of the C2 class and the C3 class is a subclass of the C1 class, then the C1 class is also a subclass of the C3 class. Furthermore, when S1, S2, and S3 are instances of each class, then S1 ⊆ S2 and S1 ⊆ S3. In the sample of Sect. 4, e-book ⊆ Book and Book ⊆ Product; then e-book ⊆ Product. In addition, e-book can have all the properties of Book. Also, a subclass has an is-a relationship to a higher class, and then VIP is-a customer and Black List is-a customer also.

The transitivity between properties can also be verified. If the relationships P1 rdfs:subPropertyOf P2 and P2 rdfs:subPropertyOf P3 exist, then the relationship P1 rdfs:subPropertyOf P3 is logically satisfied. Thus, the resources S and O satisfy the triples “S P1 O “,” “S P2 O “,” and “S P3 O “.”[24]. In the sample of Sect. 4, if :Return rdfs:subPropertyOf :Delivery, and :Delivery rdfs:subPropertyOf :Order exists, then :Return rdfs:subPropertyOf :Order is satisfied, which indicates that “Karl returned Harry Porter 1” satisfies “Karl ordered Harry Porter 1”.

7. Conclusion

This paper transforms entities, relationships, and data to RDF/RDFS without semantic loss with an interoperable data model named the ER$^2$iDM. The basic concepts of RDF/RDFS are applied to the RDM, and then implemented as the ER$^2$iDM. The trials to generate RDF statement with null valued data and whole data grouping did not occur at the ER$^2$iDM. Moreover, a RDF/RDFS statement can be generated from the ER$^2$iDM by SQL automatically. In particular, the concepts of classes and properties of RDFS are fully mapped to the hierarchical structures of entities and relationships of the RDM.

This paper is the first trial to fully map all schema information of a RDM to the basic concept of a RDF. Therefore, this study provides much more developed and robust mapping to RDF/RDFS compared with existing researches. The sample case of this paper, a shopping mall business, is a universal business case in the real world, and the shopping mall business can be adopted in other business areas such as commercial contracts, diagnosis in hospitals, and others.

References

2010.


Mi-Young Choi received a M.S. degree in Computer Science from POSTECH, Pohang, Republic of Korea in 1992. She is candidates of Ph.D. degree in the Software System Lab. from Korea University. Her research areas include data architecture, data modeling, commercial DBMS performance tuning, information strategy planning, and information security.

Chang-Joo Moon received a Ph.D. degree in the Department of Computer Science and Engineering from Korea University, Seoul, Republic of Korea in 2004. He is an associate professor in the Department of Aerospace Information Engineering, Konkuk University, Seoul, Republic of Korea. His research areas include real-time embedded systems, computer security, and system integration.

Doo-Kwon Baik received his M.S. and Ph.D. degrees in Computer Science from Wayne State University in the U.S. in 1983 and 1986, respectively. He is a full professor in the Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea. He is interested in semantic web, ontology, data engineering, modeling & simulation, and metadata registry.