ISSN 2288-4866 (Print)
ISSN 2288-4882 (Online)
http://www.jiisonline.org

http://dx.doi.org/10.13088/jiis.2018.24.3.243

An Algorithm for Finding a Relationship Between Entities: Semi-Automated Schema Integration Approach

Yongchan Kim
College of Business Administration,
Seoul National University
(*nowmyt@gmail.com*)

Jinsoo Park
College of Business Administration,
Seoul National University
(*jinsoo@snu.ac.kr*)

Jihae Suh
Big Data Institute,
Seoul National University
(*jihaesuh77@snu.ac.kr*)

Database schema integration is a significant issue in information systems. Because schema integration is a time-consuming and labor-intensive task, many studies have attempted to automate it. Researchers typically use XML as the source schema and leave much of the work to be done through DBA intervention, e.g., there are various naming conflicts related to relationship names in schema integration. In the past, the DBA had to intervene to resolve the naming-conflict name. In this paper, we introduce an algorithm that automatically generates relationship names to resolve relationship name conflicts that occur during schema integration. This algorithm is based on an Internet collocation and English sentence example dictionary. The relationship between the two entities is generated by analyzing examples extracted based on dictionary data through natural language processing. By building a semi-automated schema integration system and testing this algorithm, we found that it showed about 90% accuracy. Using this algorithm, we can resolve the problems related to naming conflicts that occur at schema integration automatically without DBA intervention.

Key Words : Schema Integration, Naming Conflicts, Natural Language Processing, XML, Entity Relationship Diagram (ERD)

Received : June 3, 2018 Revised : September 17, 2018 Accepted : September 28, 2018
Publication Type : Regular Paper(Fast-track) Corresponding Author : Jihae Suh

1. Introduction

Conceptual modeling has assumed a relevant role in the development of information systems and software applications because it is an essential phase in database design (Castano et al., 1998). Conceptual modeling of data is part of most applied system development methods further, enterprise modeling has emerged as a preliminary design phase in software system development to capture the most important aspects in an organization. The increase in the number of databases has entailed the management of related data in different formats across these databases. In order for organizations to use other organizations' data for better decision-making and success, they need to understand the semantics and retrieve from these other distributed and heterogeneous data sources (Unal and Afsarmanesh, 2010). Moreover, "even a single enterprise may have heterogeneous information bases for reasons of history or departmental autonomy" (Kaul et al., 1990). As a result, interoperability is becoming one of the most critical issues for medium- to large-size enterprises (Spaccapietra et al., 1992).

Schema integration is defined as the activity of integrating the schemas of existing or proposed databases into a global, unified schema (Batini et al., 1986). Two types of schema integration are defined: (1) view integration, which is performed during the database design process, e.g., at the conceptual design phase, and (2) database integration, which produces the global schema of a number of databases (Batini et al., 1986). Schema integration has been a fundamental issue in data sharing among distributed, heterogeneous, and autonomous databases. With the increasing number of databases, integration problems have become more apparent. Schema integration aims at finding a unified representation of schemas by merging them. In order to integrate schemas, syntactic, semantic, and structural relationships among elements of these schemas need to be identified (Unal et al., 2010). A large amount of work has been done in the integration area. Batini et al. (1986) offer a detailed survey of methodology for view and database integration. New contributions often appear in the literature (Motro, 1987; Hayne and Ram, 1990; Kaul et al, 1990; Gotthard et al, 1992; Spaccapietra et al., 1992; Spaccapietra and Parent, 1994; Beeri and Milo, 1999; Kwan and Fong, 1999). Most of the work has been performed in the context of the relational model, the functional model (Motro, 1987), and semantic data models, e.g., the object-oriented model, and the ER model (Spaccapietra and Parent, 1994). The majority of these approaches do not aim at developing semi-automated systems. What they do provide are general guidelines and concepts on different steps of the integration process. However, because schema integration is a difficult and complex task, there is a need to help users with this complicated task by providing some semi-automatic mechanisms (Unal and Afsarmanesh, 2010). A number of recent efforts focused on semi-automatic schema integration or merging, including Melnik et al. (2003), and Pottinger and Bernstein (2008). However, most of these studies used XML schemas as source schemas and do not use ER models as source schemas. The ER model (Chen, 1976) has attracted considerable attention in system modeling and database design (Lee and ling, 2003). The ER concepts (entities and relationships) correspond to structures naturally occurring in information systems. This enhances the ability of designers to accurately describe database applications. Furthermore, the schema integration studies dealt with the DBA's involvement in the new relationship names that occurred during the process of resolving structural conflicts. Choosing one between two relationship names in a synonym relationship or naming a newly created relationship is a cumbersome task for the DBA. Thus, automating the relationship name issues that occur during the schema integration process will improve the efficiency of the overall schema integration process.

In this respect, this study focuses on the two problems found in previous studies. The first is to build a semi-automated schema integrity system using the ER model as the source schema. Second, this study suggests an algorithm that can automatically solve problems related to relationship names in the schema integration process. We first describe the process of transforming the ER model into machine-understandable XML in order to build a semi-automated schema integration system using the ER model as the source schema. This process takes place during the pre-integration process and must be done manually by the DBA. The next step is to find the identical elements among the schemas through schema matching. Here we use Stanford core NLP to measure the similarity between each element name. After that, we resolve the structural differences between the two schemas through algorithms that resolve structural conflicts. In the process of resolving a structural conflict, if there is a relationship with a newly created entity, we apply the algorithm we have developed to deal with this problem automatically. The intermediate schema generated through this process is integrated to finally generate the integrated schema. We also measured the quality of the final integrated schema by measuring completeness and minimality by comparing the integration schema generated by the system with the integration schema received from the experts.

To sum, the main contributions of this study are as follows:

- The ER model is used as the source schema to construct a semi-automated schema integration system;
- We automatically solve the problems related to relationship names in the schema integration process through our algorithm.

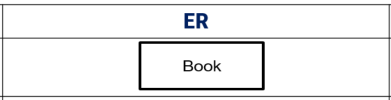
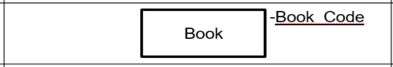
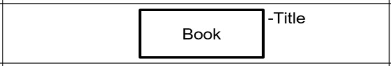
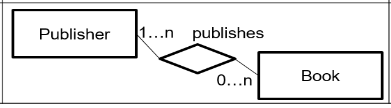
The rest of the paper is organized as follows. Section 2 describes the methodologies used in the schema integration system, such as conversion of ER to XML, schema matching, and structural conflict resolution. Section 3 briefly describes the algorithm for finding a relationship between entities we have developed. In Section 4, we apply the methodologies described in Section 2 and our algorithm described in Section 3 to implement semi-automated schema integration. We conclude in Section 5.

2. Methodologies for semi-automated schema integration

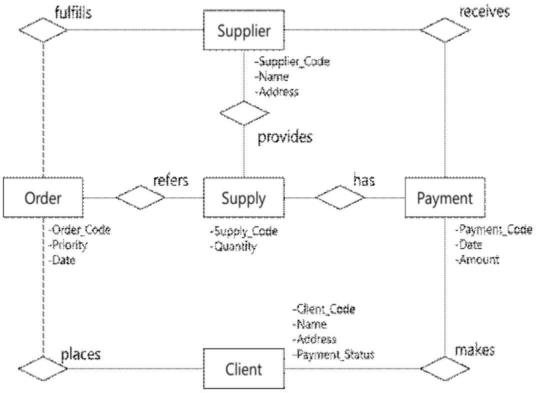
Peter originally proposed the entity-relationship (ER) model in 1976 as a way to unify network and relational database views. Simply stated, the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the entity-relationship diagram, which is used to visually represent data objects. An ER model gives a graphical and diagrammatical representation of various entities, i.e., its attributes and relationships between entities. This is turn helps to clarify understanding the data structure and in minimizing redundancy and other problems. Nevertheless, the ER model is easy for humans to handle but not so much for the machine. Therefore, in order to carry out automated schema integration on a machine, it is necessary to process the ER model into another form.

The eXtensible Markup Language (XML) has emerged as a standard for information representation and exchange on the Web as well as the Intranet due to its self-describing data capability and flexibility in organizing data (Gou and Chirkova, 2007). The XML tag names are readable and convey the meaning of the data. The information structure is easily discerned by humans and computers, as each XML tag immediately precedes the associated data. The data structure follows a noticeable and useful pattern, making it easy to manipulate and exchange the data (Algergawy et al., 2010). Thus, we convert the ER model to XML so that the machine can understand it. Because the majority of data in the world is stored in databases, the conversion of such data into XML documents is indispensable for real world usage. In this conversion, rules and algorithms for preserving the information of the database schema and generating XML documents based on such information are necessary.

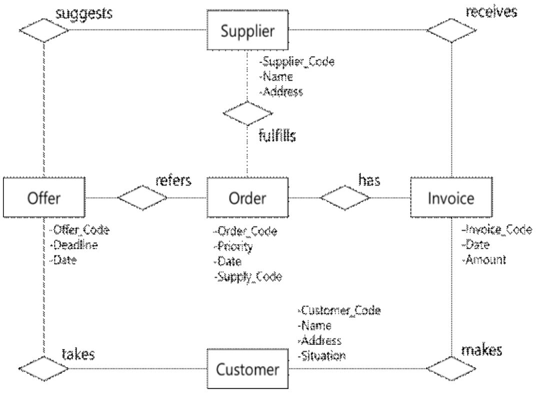
We adopted Jin and Kang's (2007) rules to convert the ER model to XML. They describe ER-to-XML mapping rules at the schema level. Each entity type and relationship type in the ER diagram is mapped into the top-level element in the XML document. There are six top-level XML elements that represent different entity types and relationship cardinalities: *<entity>, <weak entity>, <unary-relationship>, <binary-relationship>, <ternary-relationship>, and <n-ary relationship>.* The content (i.e., data value) of a top-level element is the same as the corresponding name of an entity type or a relationship type. For example, an entity type STUDENT is represented in XML as *<entity>*STUDENT*</entity>*. The attributes of an entity type in the ER diagram are mapped into the subelement *<attribute>* of the corresponding top-level element in XML. The ER model used in this study contains only entities and binary relations. Figure 1 presents entities and binary relations of the ER model and how the attributes are converted to XML.

	ER	XML
Entity		<entity>Book</entity>
Key-Attribute		<entity>Book <key-attribute>Book_Code</key-attribute> </entity>
Attribute		<entity>Book <attribute>Title</attribute> </entity>
Binary Relationship		<binary-relationship>publishes <entity *min-card*="1" *max-card*="unbounded">Publisher</entity> <entity *min-card*="0" *max-card*="unbounded">Book</entity> </binary-relationship>

〈Figure 1〉 Mapping Rules for ER to XML



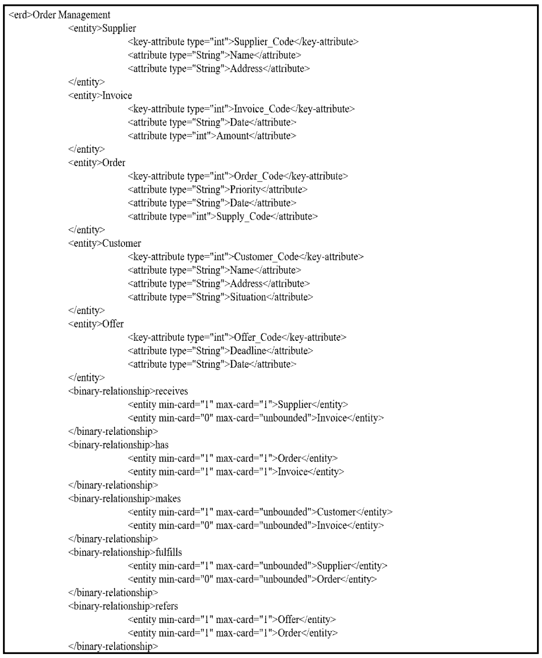
〈Figure 2〉 Order Management Schema 1



〈Figure 3〉 Order Management Schema 2

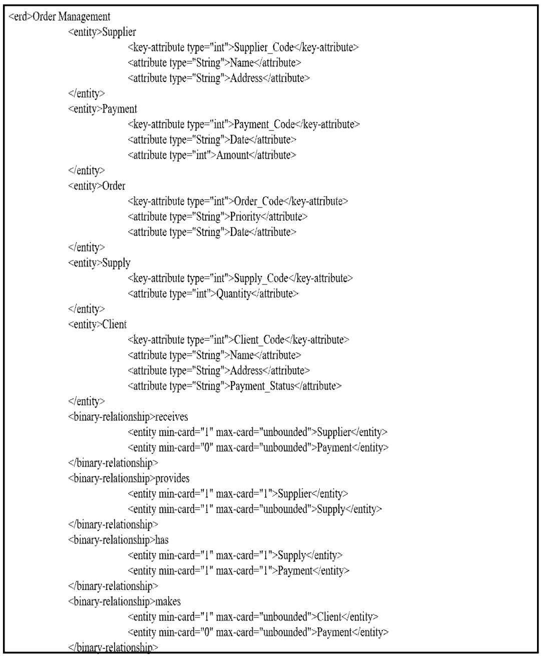
The strong entity type S in the ER diagram is mapped to the <*entity*> element in the XML document. The key attribute A of the entity E is mapped in a similar way to the simple attribute. In this case, the <*key-attribute*> element is added as a subelement of the top-level element E. A simple attribute A of the entity E in the ER diagram is represented in XML using the <*attribute*> element. The <*attribute*> element is placed as a subelement of the belonging top-level XML element. The binary relationship R between two entity types S and T is mapped to the top-level element <*binary-relationship*>. In addition, the two participating <*entity*> elements are also placed as subelements. In this case, for the associated <*entity*> element, there are two required XML attributes to express the minimum and maximum cardinality constraints (i.e., min-card and max-card, respectively).

Figures 2 and 3 are the ER models we used for schema integration. The results of converting these ER models into XML according to the conversion rules described above are as follows: Figures 4 and 5 are an ER model converted into an XML document, and Figure 6 shows an XML schema of the corresponding XML document.



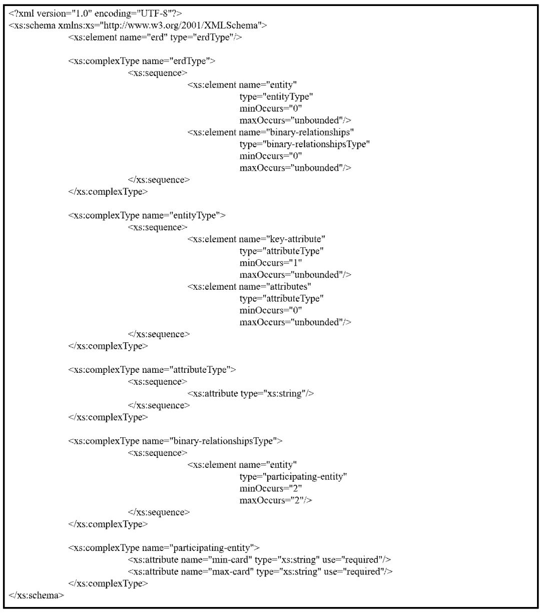
```
<erd>Order Management
        <entity>Supplier
                <key-attribute type="int">Supplier_Code</key-attribute>
                <attribute type="String">Name</attribute>
                <attribute type="String">Address</attribute>
        </entity>
        <entity>Invoice
                <key-attribute type="int">Invoice_Code</key-attribute>
                <attribute type="String">Date</attribute>
                <attribute type="int">Amount</attribute>
        </entity>
        <entity>Order
                <key-attribute type="int">Order_Code</key-attribute>
                <attribute type="String">Priority</attribute>
                <attribute type="String">Date</attribute>
                <attribute type="int">Supply_Code</attribute>
        </entity>
        <entity>Customer
                <key-attribute type="int">Customer_Code</key-attribute>
                <attribute type="String">Name</attribute>
                <attribute type="String">Address</attribute>
                <attribute type="String">Situation</attribute>
        </entity>
        <entity>Offer
                <key-attribute type="int">Offer_Code</key-attribute>
                <attribute type="String">Deadline</attribute>
                <attribute type="String">Date</attribute>
        </entity>
        <binary-relationship>receives
                <entity min-card="1" max-card="1">Supplier</entity>
                <entity min-card="0" max-card="unbounded">Invoice</entity>
        </binary-relationship>
        <binary-relationship>has
                <entity min-card="1" max-card="1">Order</entity>
                <entity min-card="1" max-card="1">Invoice</entity>
        </binary-relationship>
        <binary-relationship>makes
                <entity min-card="1" max-card="unbounded">Customer</entity>
                <entity min-card="0" max-card="unbounded">Invoice</entity>
        </binary-relationship>
        <binary-relationship>fulfills
                <entity min-card="1" max-card="unbounded">Supplier</entity>
                <entity min-card="0" max-card="unbounded">Order</entity>
        </binary-relationship>
        <binary-relationship>refers
                <entity min-card="1" max-card="1">Offer</entity>
                <entity min-card="1" max-card="1">Order</entity>
        </binary-relationship>
```

〈Figure 4〉 XML document of schema 1



```
<erd>Order Management
        <entity>Supplier
                <key-attribute type="int">Supplier_Code</key-attribute>
                <attribute type="String">Name</attribute>
                <attribute type="String">Address</attribute>
        </entity>
        <entity>Payment
                <key-attribute type="int">Payment_Code</key-attribute>
                <attribute type="String">Date</attribute>
                <attribute type="int">Amount</attribute>
        </entity>
        <entity>Order
                <key-attribute type="int">Order_Code</key-attribute>
                <attribute type="String">Priority</attribute>
                <attribute type="String">Date</attribute>
        </entity>
        <entity>Supply
                <key-attribute type="int">Supply_Code</key-attribute>
                <attribute type="int">Quantity</attribute>
        </entity>
        <entity>Client
                <key-attribute type="int">Client_Code</key-attribute>
                <attribute type="String">Name</attribute>
                <attribute type="String">Address</attribute>
                <attribute type="String">Payment_Status</attribute>
        </entity>
        <binary-relationship>receives
                <entity min-card="1" max-card="unbounded">Supplier</entity>
                <entity min-card="0" max-card="unbounded">Payment</entity>
        </binary-relationship>
        <binary-relationship>provides
                <entity min-card="1" max-card="1">Supplier</entity>
                <entity min-card="1" max-card="unbounded">Supply</entity>
        </binary-relationship>
        <binary-relationship>has
                <entity min-card="1" max-card="1">Supply</entity>
                <entity min-card="1" max-card="1">Payment</entity>
        </binary-relationship>
        <binary-relationship>makes
                <entity min-card="1" max-card="unbounded">Client</entity>
                <entity min-card="0" max-card="unbounded">Payment</entity>
        </binary-relationship>
```

〈Figure 5〉 XML document of Schema 2

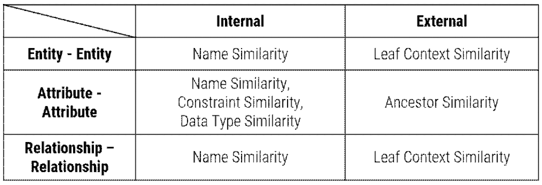


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="erd" type="erdType"/>

        <xs:complexType name="erdType">
                <xs:sequence>
                        <xs:element name="entity"
                                type="entityType"
                                minOccurs="0"
                                maxOccurs="unbounded"/>
                        <xs:element name="binary-relationships"
                                type="binary-relationshipsType"
                                minOccurs="0"
                                maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="entityType">
                <xs:sequence>
                        <xs:element name="key-attribute"
                                type="attributeType"
                                minOccurs="1"
                                maxOccurs="unbounded"/>
                        <xs:element name="attributes"
                                type="attributeType"
                                minOccurs="0"
                                maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="attributeType">
                <xs:sequence>
                        <xs:attribute type="xs:string"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="binary-relationshipsType">
                <xs:sequence>
                        <xs:element name="entity"
                                type="participating-entity"
                                minOccurs="2"
                                maxOccurs="2"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="participating-entity">
                <xs:attribute name="min-card" type="xs:string" use="required"/>
                <xs:attribute name="max-card" type="xs:string" use="required"/>
        </xs:complexType>
</xs:schema>
```

〈Figure 6〉 XML Schema of XML Document

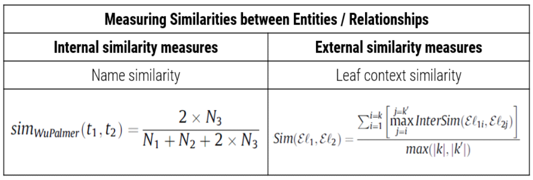
The next step is a schema matching process, which finds the corresponding pair with the transformed XML. In this process, we adopted Algergawy et al. (2010)'s measurement method. The authors categorize element similarity measures guided by the following observation: a number of similarity measures make use of element internal features without considering its surrounds. On the other hand, several element similarity measures exploit element relationships making use of element surrounds. The former is called *internal element similarity*, and the latter is called *external element similarity*. Once the internal and external element similarity values are obtained, a total similarity value between a pair of elements can be determined. Table 1 shows the measurement methods applied to each schema element in this study. Table 2 and 3 are formulas for each measurement method.

〈Table 1〉 Similarity Measure Used for Each Schema Element



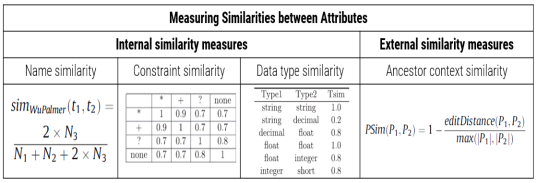
	Internal	**External**
Entity - Entity	Name Similarity	Leaf Context Similarity
Attribute - Attribute	Name Similarity, Constraint Similarity, Data Type Similarity	Ancestor Similarity
Relationship – Relationship	Name Similarity	Leaf Context Similarity

〈Table 2〉 Formulas for Entities and Relationships



Measuring Similarities between Entities / Relationships	
Internal similarity measures	**External similarity measures**
Name similarity	Leaf context similarity
$sim_{WuPalmer}(t_1,t_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$	$Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = \frac{\sum_{i=1}^{i=k}\left[\max_{j=i}^{j=k'} InterSim(\mathcal{E}\ell_{1i}, \mathcal{E}\ell_{2j})\right]}{max(\lvert k\rvert, \lvert k'\rvert)}$

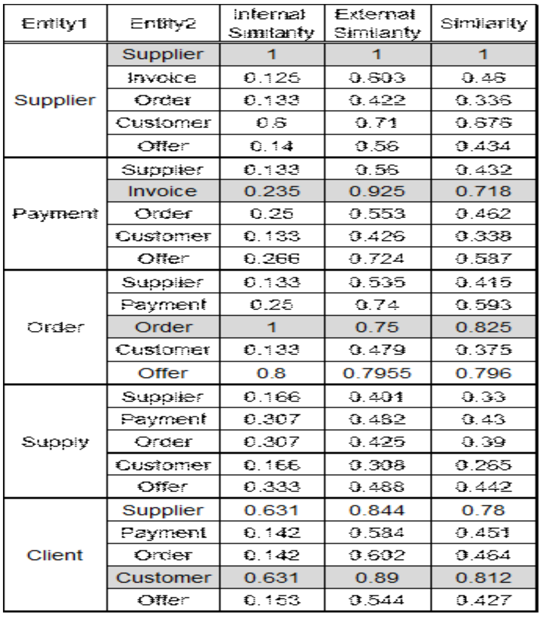
〈Table 3〉 Formulas for Attributes



<table>
<tr><th colspan="4">Measuring Similarities between Attributes</th></tr>
<tr><th colspan="3">Internal similarity measures</th><th>External similarity measures</th></tr>
<tr><td>Name similarity</td><td>Constraint similarity</td><td>Data type similarity</td><td>Ancestor context similarity</td></tr>
<tr>
<td>$sim_{WuPalmer}(t_1,t_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$</td>
<td><table>
<tr><td></td><td>*</td><td>+</td><td>?</td><td>none</td></tr>
<tr><td>*</td><td>1</td><td>0.9</td><td>0.7</td><td>0.7</td></tr>
<tr><td>+</td><td>0.9</td><td>1</td><td>0.7</td><td>0.7</td></tr>
<tr><td>?</td><td>0.7</td><td>0.7</td><td>1</td><td>0.8</td></tr>
<tr><td>none</td><td>0.7</td><td>0.7</td><td>0.8</td><td>1</td></tr>
</table></td>
<td><table>
<tr><th>Type1</th><th>Type2</th><th>Tsim</th></tr>
<tr><td>string</td><td>string</td><td>1.0</td></tr>
<tr><td>string</td><td>decimal</td><td>0.2</td></tr>
<tr><td>decimal</td><td>float</td><td>0.8</td></tr>
<tr><td>float</td><td>float</td><td>1.0</td></tr>
<tr><td>float</td><td>integer</td><td>0.8</td></tr>
<tr><td>integer</td><td>short</td><td>0.8</td></tr>
</table></td>
<td>$PSim(P_1,P_2) = 1 - \frac{editDistance(P_1,P_2)}{max(\lvert P_1 \rvert, \lvert P_2 \rvert)}$</td>
</tr>
</table>

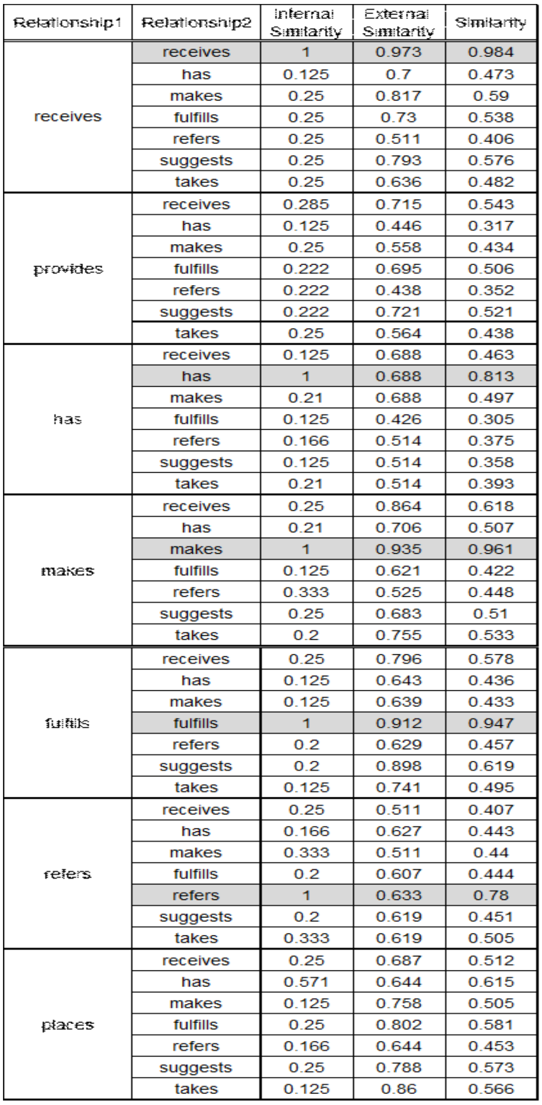
Because, in the XML document that transformed the ER model, the internal information about the entity and relationship is only the name, we used the name similarity only as a measure of the internal similarity of the entities and relationships. For measuring external similarity between entities (relationships), we used leaf context similarity to measure the similarity of the attributes of each entity (relationship). Since the effective content of a node is often captured by the leaf nodes of the subtree rooted at that node, we compute leaf context similarity (Zerdazi and Myriam, 2007). For attributing the internal similarity measure, name, constraint, and data type were measured. As an external similarity measurement method, we use the ancestor context similarity. Tables 4 and 5 show the results from the similarity measure.

〈Table 4〉 Result of Entity Similarity Comparison



Entity1	Entity2	Internal Similarity	External Similarity	Similarity
Supplier	Supplier	1	1	1
	Invoice	0.125	0.603	0.46
	Order	0.133	0.422	0.336
	Customer	0.6	0.71	0.676
	Offer	0.14	0.56	0.434
Payment	Supplier	0.133	0.56	0.432
	Invoice	0.235	0.925	0.718
	Order	0.25	0.553	0.462
	Customer	0.133	0.426	0.338
	Offer	0.266	0.724	0.587
Order	Supplier	0.133	0.535	0.415
	Payment	0.25	0.74	0.593
	Order	1	0.75	0.825
	Customer	0.133	0.479	0.375
	Offer	0.8	0.7955	0.796
Supply	Supplier	0.166	0.401	0.33
	Payment	0.307	0.482	0.43
	Order	0.307	0.425	0.39
	Customer	0.166	0.308	0.265
	Offer	0.333	0.488	0.442
Client	Supplier	0.631	0.844	0.78
	Payment	0.142	0.584	0.451
	Order	0.142	0.602	0.464
	Customer	0.631	0.89	0.812
	Offer	0.153	0.544	0.427

〈Table 5〉 Result of Relationship Similarity Comparison



Relationship1	Relationship2	Internal Similarity	External Similarity	Similarity
receives	receives	1	0.973	0.984
	has	0.125	0.7	0.473
	makes	0.25	0.817	0.59
	fulfills	0.25	0.73	0.538
	refers	0.25	0.511	0.406
	suggests	0.25	0.793	0.576
	takes	0.25	0.636	0.482
provides	receives	0.285	0.715	0.543
	has	0.125	0.446	0.317
	makes	0.25	0.558	0.434
	fulfills	0.222	0.695	0.506
	refers	0.222	0.438	0.352
	suggests	0.222	0.721	0.521
	takes	0.25	0.564	0.438
has	receives	0.125	0.688	0.463
	has	1	0.688	0.813
	makes	0.21	0.688	0.497
	fulfills	0.125	0.426	0.305
	refers	0.166	0.514	0.375
	suggests	0.125	0.514	0.358
	takes	0.21	0.514	0.393
makes	receives	0.25	0.864	0.618
	has	0.21	0.706	0.507
	makes	1	0.935	0.961
	fulfills	0.125	0.621	0.422
	refers	0.333	0.525	0.448
	suggests	0.25	0.683	0.51
	takes	0.2	0.755	0.533
fulfills	receives	0.25	0.796	0.578
	has	0.125	0.643	0.436
	makes	0.125	0.639	0.433
	fulfills	1	0.912	0.947
	refers	0.2	0.629	0.457
	suggests	0.2	0.898	0.619
	takes	0.125	0.741	0.495
refers	receives	0.25	0.511	0.407
	has	0.166	0.627	0.443
	makes	0.333	0.511	0.44
	fulfills	0.2	0.607	0.444
	refers	1	0.633	0.78
	suggests	0.2	0.619	0.451
	takes	0.333	0.619	0.505
places	receives	0.25	0.687	0.512
	has	0.571	0.644	0.615
	makes	0.125	0.758	0.505
	fulfills	0.25	0.802	0.581
	refers	0.166	0.644	0.453
	suggests	0.25	0.788	0.573
	takes	0.125	0.86	0.566

As a result of measuring the similarity between the entities of Schemas 1 and 2, the following results were obtained:

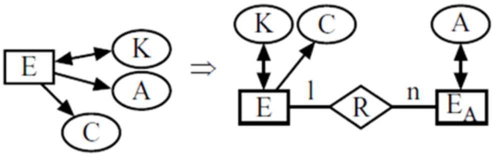
S1.Supplier ≡ S2.Supplier
S1.Payment ≅ S2.Invoice
S1.Order ≅ S2.Order
S1.Client ≅ S2.Customer

The threshold is 0.7, and if two or more entities are above the threshold, the entity having the highest value is adopted. *S1.Supplier* and *S2.Supplier* were found to be completely identical, and *S1.Payment* and *S2.Invoice* were found to be quite similar. Entities similar to *S1.Order* have *S2.Order* and *S2.Offer*, but the highest value of *S2.Order* is most similar to *S1.Order*. As a result of the similarity measurement of relationships, the following results were obtained:

S1.receives ≅ S2.receives
S1.has ≅ S2.has
S1.makes ≅ S2.makes
S1.fulfills ≅ S2.fulfills
S1.refers ≅ S2.refers

For schema integration, it is necessary to find corresponding pairs between schemas through schema matching and to resolve naming conflicts or structural conflicts among the corresponding elements. In order to resolve the naming conflict, the name of the entity in Schema 1 was adopted. We adopted Lee and Ling (2003)'s study to solve structural conflicts. The authors present a schema integration methodology with particular focus on the resolution of structural conflicts. They find that, if the individual schemas have been designed properly and the semantic equivalences among the schemas identified correctly, then the key structural conflict is that between an entity type and an attribute. In their work, they insist that resolving all structural conflicts between entities and attributes will solve all sorts of structural conflicts. Structural conflicts between entities and attributes occur when an object exists as an entity in one schema and an attribute exists in the other.

To check if there is a structural conflict, we need to ensure that the entity that exists as an entity in one schema exists as an attribute in the other. One way to confirm this is that, if the key attribute of one schema entity exists as a simple attribute of the entity in the other schema, then the simple attribute is an entity type. Another case is that an entity name in one schema is included in an attribute in the other schema. In our example schema, we can see that *Supply_Code* is the key



A is not part of a key and not part of a composite attribute.
E_A is connected to E by a new relationship set R.

〈Figure 7〉 Transformation of an entity type attribute A into an entity type E_A.

attribute of the *S1.Supply* entity in Schema 1 and is the simple attribute of *S2.Order* in Schema 2. In this case, because of the structural conflict, we transformed the *S2.Order.Supply_Code* attribute into an entity by applying the above transformation.

3. An algorithm for finding a relationship between entities

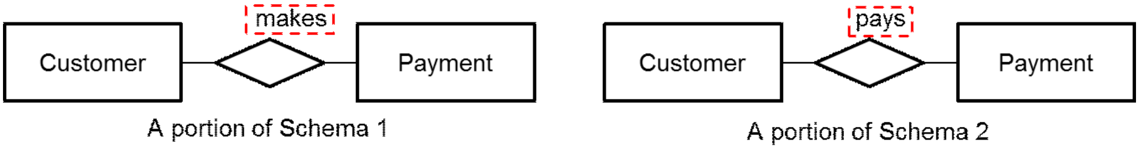
In this paper, conflicts deal only with naming conflicts and structural conflicts. The process of conflict resolution in schema integration is divided into naming conflict and structural conflict. During the resolving naming conflicts, the relationship naming conflict, as shown in Figure 8, may occur.

This is a common case where different relationship names are assigned to identical entities. In this case, DBA must select manually one of the two relationship names. Another case is when a new entity is created in the process of resolving a structural conflict. The relationship between the newly created entity and the existing entity has not yet been given a name. In the existing research, it was necessary to manually specify the relationship name through the intervention of the DBA.

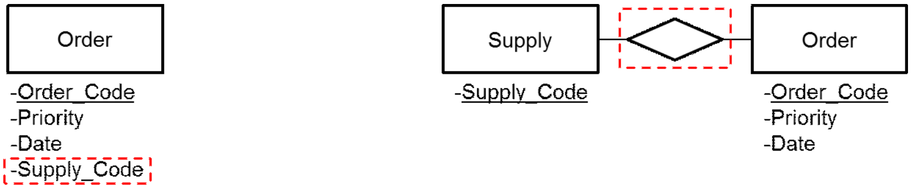
During the resolving structural conflicts process, the following may occur:

Figure 9 shows a case where a space occurs in the relationship name when an attribute *Supply_Code* is transformed into an entity. In this case, the DBA must also choose one of the two relationship names. DBA intervention in this schema integration process makes it difficult to automate schema integration and is time-consuming and labor intensive.

This algorithm automatically generates relationship names in these cases. Briefly, our algorithm first searches the Internet collocation



〈Figure 8〉 Example of relationship name conflict (1).



〈Figure 9〉 Example of Relationship Name Conflict (2).

dictionary for a specific entity name's collocation. In the generated collocation set, a combination of each element and entity name is searched in the dictionary to find the collocation where most examples are present.

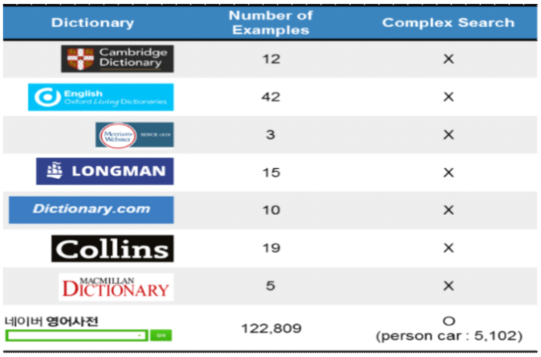
According to Chen's (1983) research, nouns in English sentences appear as entities in the ER model, and verbs appear in the form of relationships. Thus, we have found that more similar sentences, including the entities (noun) and the relation (verb), can infer the relationship between entities.

In computational linguistics, a wide variety of lexical association measures have been employed for the task of (semi-)automatic collocation identification and extraction:

- frequency-based measures (e.g., based on absolute and relative co-occurrence frequencies);
- information-theoretic measures (e.g., mutual information, entropy);
- statistical measures (e.g., chi-square, t-test, log-likelihood, Dice's coefficient).

We adopt the frequency-based measurement method and adopt the most frequent verb as the relationship between two entities.

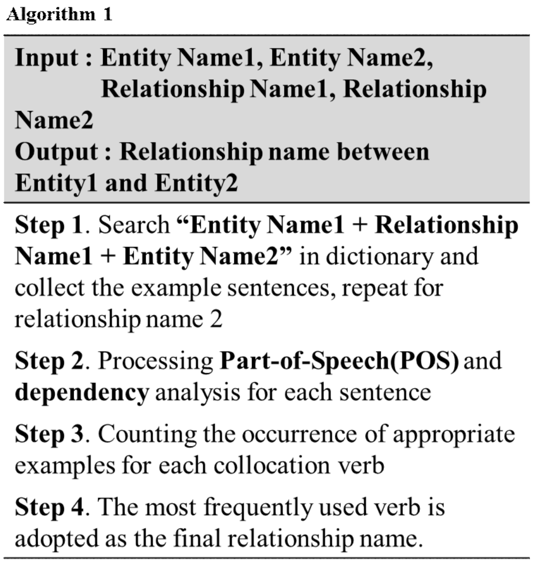
We compared various dictionaries to select those from which to extract an example sentence set. The comparison criterion was how many examples were searched and whether they supported a complex search.



Dictionary	Number of Examples	Complex Search
Cambridge Dictionary	12	×
English Oxford *Living* Dictionaries	42	×
Merriam-Webster SINCE 1828	3	×
LONGMAN	15	×
Dictionary.com	10	×
Collins	19	×
MACMILLAN DICTIONARY	5	×
네이버 영어사전	122,809	O (person car : 5,102)

〈Figure 10〉 Comparison of Various Dictionaries

As a result, it was confirmed that the *Naver English Dictionary* was overwhelmingly used in a number of example sentences and also supports the complex search function. Therefore, we selected the *Naver English Dictionary* as an example extract dictionary.



Algorithm 1

Input : Entity Name1, Entity Name2, Relationship Name1, Relationship Name2
Output : Relationship name between Entity1 and Entity2

Step 1. Search **"Entity Name1 + Relationship Name1 + Entity Name2"** in dictionary and collect the example sentences, repeat for relationship name 2

Step 2. Processing **Part-of-Speech(POS)** and **dependency** analysis for each sentence

Step 3. Counting the occurrence of appropriate examples for each collocation verb

Step 4. The most frequently used verb is adopted as the final relationship name.

〈Figure 11〉 Algorithm 1

Algorithm 2

Input : Entity Name1, Entity Name2
Output : Relationship name between Entity1 and Entity2

Step 1. Search collocations of entity names (Input : Person, Car)

Step 2. Search **"Entity Name1 + A verb extracted from the collocation set + Entity Name2"** in dictionary and collect the example sentences

Step 3. Processing **Part-of-Speech(POS)** and **dependency** analysis for each sentence

Step 4. Counting the occurrence of appropriate examples for each collocation verb

Step 5. The most frequently used verb is adopted as the final relationship name.

〈Figure 12〉 Algorithm 2

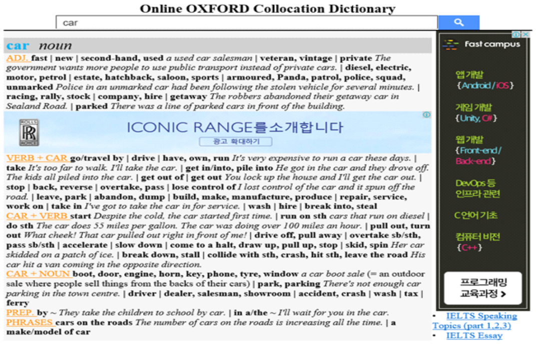
In the case of Figure 8, because two relation names have already been given, the step for searching the collocations of entities is skipped. In Figure 9, the relationship name does not exist at all and thus includes a process of searching for collocations of entities. Thus, the rules for applying our algorithm are as follows:

Rule1. If the name of 1 and the name of 2 conflict with each other, Algorithm 1 selects one of them.

Rule2. If a new relationship is created in the process of resolving the structural conflict, the relationship name is created through Algorithm 2.

Because Algorithm 1 only omits the step of searching for a collocation in Algorithm 2, the description is based on Algorithm 2 here.

Step 1. Search collocations of entity names


Online OXFORD Collocation Dictionary

car

car *noun*

ADJ. **fast | new | second-hand, used** *a used car salesman* | **veteran, vintage | private** *The government wants more people to use public transport instead of private cars.* | **diesel, electric, motor, petrol | estate, hatchback, saloon, sports | armoured, Panda, patrol, police, squad, unmarked** *Police in an unmarked car had been following the stolen vehicle for several minutes.* | **racing, rally, stock | company, hire | getaway** *The robbers abandoned their getaway car in Sealand Road.* | **parked** *There was a line of parked cars in front of the building.*

VERB + CAR **go/travel by | drive | have, own, run** *It's very expensive to run a car these days.* | **take** *It's too far to walk. I'll take the car.* | **get in/into, pile into** *He got in the car and they drove off. The kids all piled into the car.* | **get out of | get out** *You lock up the house and I'll get the car out.* | **stop | back, reverse | overtake, pass | lose control of** *I lost control of the car and it spun off the road.* | **leave, park | abandon, dump | build, make, manufacture, produce | repair, service, work on | take in** *I've got to take the car in for service.* | **wash | hire | break into, steal**

CAR + VERB **start** *Despite the cold, the car started first time.* | **run on sth** *cars that run on diesel* | **do sth** *The car does 55 miles per gallon. The car was doing over 100 miles an hour.* | **pull out, turn out** *What cheek! That car pulled out right in front of me!* | **drive off, pull away | overtake sb/sth, pass sb/sth | accelerate | slow down | come to a halt, draw up, pull up, stop | skid, spin** *Her car skidded on a patch of ice.* | **break down, stall | collide with sth, crash, hit sth, leave the road** *His car hit a van coming in the opposite direction.*

CAR + NOUN **boot, door, engine, horn, key, phone, tyre, window** *a car boot sale* (= an outdoor sale where people sell things from the backs of their cars) | **park, parking** *There's not enough car parking in the town centre.* | **driver | dealer, salesman, showroom | accident, crash | wash | tax | ferry**

PREP. **by ~** *They take the children to school by car.* | **in a/the ~** *I'll wait for you in the car.*

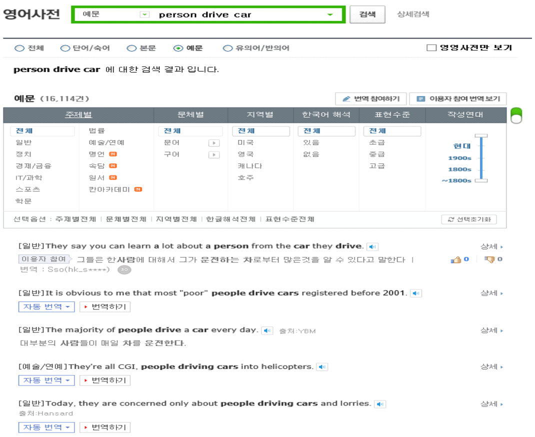
PHRASES **cars on the roads** *The number of cars on the roads is increasing all the time.* | **a make/model of car**

〈Figure 13〉 Search results for "car" in the *Collocation Dictionary*

When a specific word is searched in the *Collocation Dictionary*, a list of verbs used with the word appears. We collect these and store them in the collocation set. This process is performed twice for the first entity name and the second entity name. The list of collocations generated from the entity name "Person" and "Car" is as follows.

go by, travel by, drive, have, own, run, get in, get into, pile into, get out of, get out, stop, back, reverse, overtake, pass, lose control of, leave, park, abandon, dump, build, make, manufacture, produce, repair, service, work on, take in, wash, hire, break into, steal, start, run on, do, pull out, turn out, drive off, pull away, accelerate, slow down⋯

Step 2. Search **"Entity Name1 + A verb extracted from the collocation set + Entity Name2"** in the dictionary and collect the example sentences:



person drive car

person drive car 에 대한 검색 결과 입니다.

예문 (16,114건)

선택옵션 : 주제별전체 | 문체별전체 | 지역별전체 | 한글해석전체 | 표현수준전체

[일반]They say you can learn a lot about a **person** from the **car** they **drive**.
이용자 참여 그들은 한**사람**에 대해서 그가 **운전하는 차**로부터 많은것을 알 수 있다고 말한다 | 번역 : Sso(hk_s****) 30 0 0

[일반]It is obvious to me that most "poor" **people drive cars** registered before 2001.

[일반]The majority of **people drive** a **car** every day. 출처:YBM
대부분의 **사람**들이 매일 **차**를 **운전한다**.

[예술/연예]They're all CGI, **people driving cars** into helicopters.

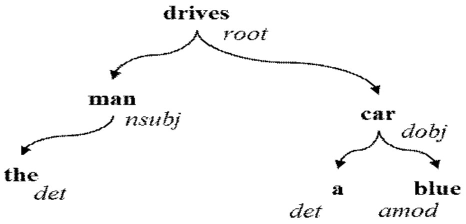
[일반]Today, they are concerned only about **people driving cars** and lorries.
출처:Hansard

〈Figure 14〉 Search results for “person drive car” in *Naver Dictionary*

Extract the collocation one by one from the collocation set and search the combination of the collocation and entity names from the dictionary. The searched example sentences are stored as a list of example sentences of the corresponding collocations.

Step 3. Processing **Part-of-Speech (POS)** and **Dependency Analysis** for each example sentence

Take an example sentence from the example sentence list and process **Part-of-Speech** and **Dependency Analysis** for each example sentence.



〈Figure 15〉 Result of dependency analysis

POS tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on its definition and its context. For example, POS tagging analysis for a sample sentence “The man drives a blue car” is as follows.

“The man drives a blue car”
-> drives-VBZ (root)
  -> man-NN (nsubj)
  -> the-DT (det)
-> car-NN (dobj)
  -> a-DT (det)
  -> blue-JJ (amod)

Next to the POS tag is a dependency for each term, which has the structure as shown in Figure 15. The dependency “root” is a grammatical relation that points to the root of the sentence. We proceeded with the above analysis for each example, and when the relationship name is located in the root part and both entity names exist in the example sentence, the example sentence is meaningful.

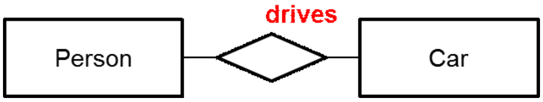
Step 4. Counting the occurrence of appropriate examples for each collocation verb.

drive is : 150	**build is : 20**	**meet is : 7**
have is : 103	**start is : 18**	**repair is : 5**
make is : 70	**produce is : 17**	**dump is : 5**
leave is : 65	**abandon is : 16**	**service is : 3**
take is : 62	**pass is : 13**	**reverse is : 2**
stop is : 47	**wash is : 10**	**accelerate is : 1**
own is : 45	**represent is : 10**	**stall is : 1**
park is : 37	**hire is : 9**	**manufacture is : 1**
run is : 30	**attract is : 9**	**skid is : 0**
steal is : 26	**crash is : 7**	**back is : 0**

〈Figure 16〉 Occurrence of appropriate examples for “Person + Collocation + Car”

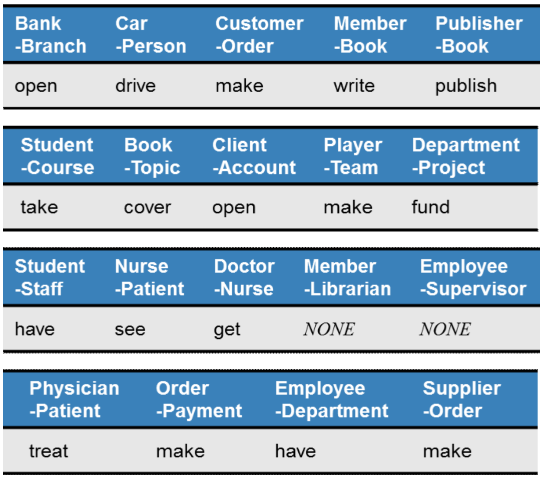
Figure 16 shows how many appropriate examples exist for each word based on the collation list of entity names “Person” and “Car.” According to the above results, the most appropriate relationship between “Person” and “Car” is “drive.”

Step 5. The most frequently used verb is adopted as the final relationship name.



〈Figure 17〉 Result of “Person” and “Car”

We tested the algorithm with 19 entity name pairs, and the result is shown in Figure 18.



Bank -Branch	Car -Person	Customer -Order	Member -Book	Publisher -Book
open	drive	make	write	publish

Student -Course	Book -Topic	Client -Account	Player -Team	Department -Project
take	cover	open	make	fund

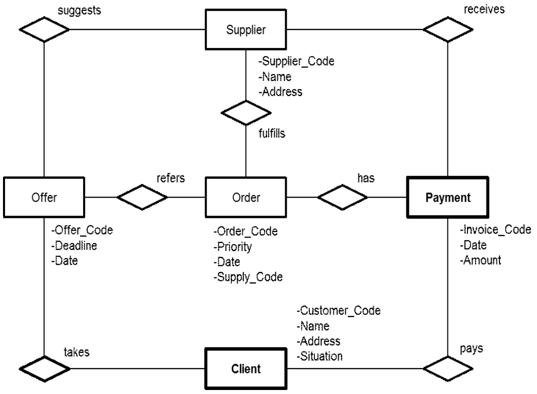
Student -Staff	Nurse -Patient	Doctor -Nurse	Member -Librarian	Employee -Supervisor
have	see	get	*NONE*	*NONE*

Physician -Patient	Order -Payment	Employee -Department	Supplier -Order
treat	make	have	make

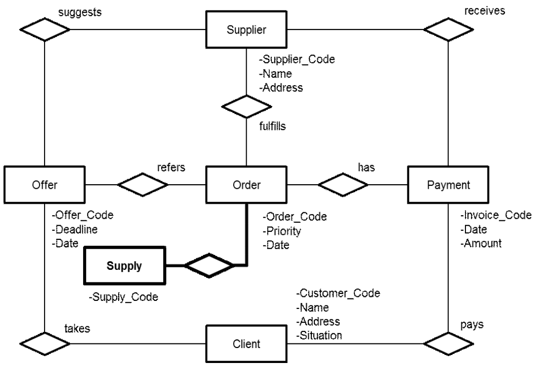
〈Figure 18〉 Result of 19 Entity Name Pairs

4. Semi-Automated Schema Integration

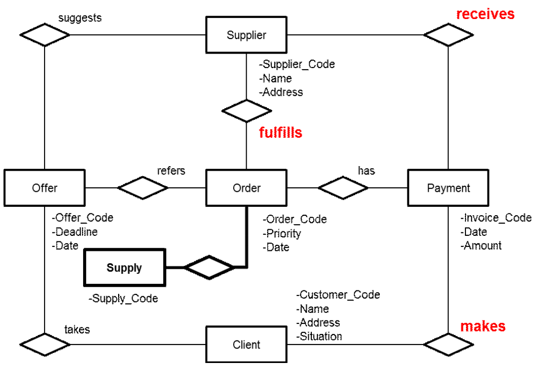
In this section, we will proceed with the actual semi-automated schema integration by applying the methodologies outlined above and the algorithms we have developed. We used the library project domain for the experiment. The reason for using the library project domain is that the domain is very familiar to the expert (Suh and Park, 2017) therefore, it is useful to evaluate the our schema integration methodology with expert integration. The schemas used are shown in Figures 2 and 3, and the schemas were preprocessed to convert them into XML document format. First, in Figure 19, name conflicts are resolved. In this paper, only the relationship names are considered. Therefore, the entity name is assumed to follow Schema 1.



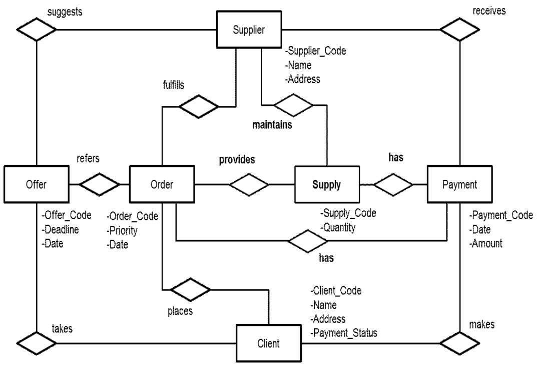
〈Figure 19〉 Change the name of elements (Schema 2)



〈Figure 20〉 Make “Supply_Code” into an entity (Schema 2)



〈Figure 21〉 Apply the algorithm to conflicting or newly created relationship name (Schema 2)



〈Figure 22〉 Integrated schema: Generated by tool (Si_{tool})

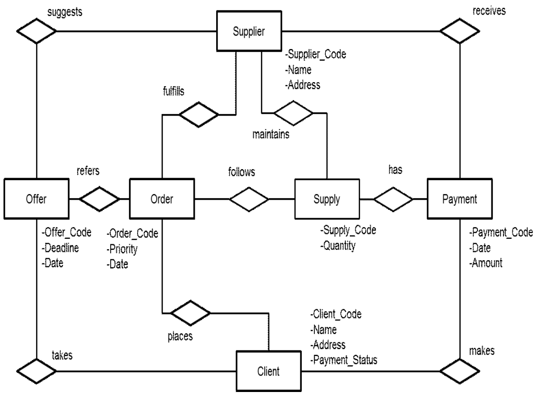
5. Evaluation

Now we have an integrated schema produced by a matching tool, named Si_{tool}, and an expert integrated schema Si_{exp}. Recall that this expert integrated schema is ideal. $|Si_{exp}|$ stands for the number of elements in schema Si_{exp}. Thus, completeness, as given by Formula (1), represents the proportion of elements in the tool integrated schema, which are common with the expert integrated schema. Minimality is computed thanks to Formula (2), and it is the percentage of extra elements in the tool integrated schema w.r.t. expert integrated schema. Both metrics are in the range (0; 1), with a 1 value, meaning that the tool integrated schema is totally complete (respectively, minimal) related to expert integrated schema.

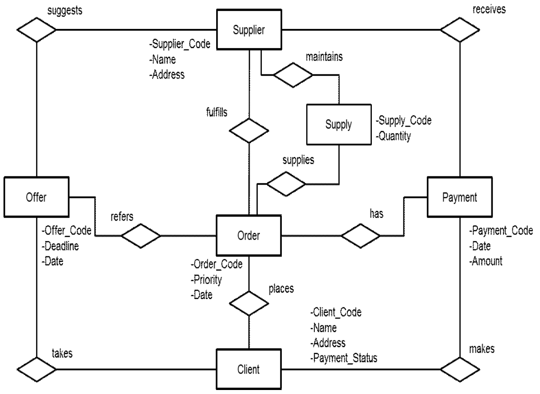
$$comp(Si_{tool}, Si_{exp}) = \frac{|Si_{tool} \cap Si_{exp}|}{|Si_{exp}|} \qquad (1)$$

$$min(Si_{tool}, Si_{exp}) = 1 - \frac{|Si_{tool}| - |Si_{tool} \cap Si_{exp}|}{|Si_{exp}|} \qquad (2)$$

The schema used for the evaluation was given from two experts. Davies et al. (2006) presented the top six most commonly used modeling techniques stratified according to the years of modeling experience of the practitioners and the results presented that a significant increase in usage from the 0-3 years level to the 4-10 years level of experience. Accordingly, we selected two experts who fit the above category. One has10 year database and 7 year modeling field experience and the other has 5 year experience in modeling field. In addition, Model comparison is well used evaluation method in conceptual modeling research (Chau and Hu, 2001). In detail, it is comparison of schema developed by expert and system. The reason for using model comparison is that conceptual model related to the representation of the entire information content of the database being designed in somewhat abstract terms relative to the way the data is physically stored (Date, 1990).



〈Figure 23〉 Integrated schema: Expert 1



〈Figure 24〉 Integrated schema: Expert 2

〈Table 6〉 Results of Schema Similarity Comparison

	$comp(Si_{tool,} Si_{exp})$	$\min(Si_{tool,} Si_{exp})$	$prox(Si_{tool,} Si_{exp})$
Si_{exp1}	93.75%	87.5%	90.63%
Si_{exp2}	87.5%	81.25%	84.38%

6. Limitations

This algorithm selects and creates relationship names but has the following limitations. First, this algorithm deals only with binary relationships. No unary relationships or ternary relationships were considered. Binary relationships cannot be selected because two entity names are not given as input values required. Ternary relations cannot be applied to this algorithm because proper example sentences cannot be retrieved due to limitations of dictionary search functions. Second, the algorithm generates a relationship name based on a simple frequency only. For example, in the case of a relationship between a person and a car, a relationship name "driving" through this algorithm is generated, but this may not be appropriate for some domains. In the case of an insurance company database, the relationship name between people and cars will be "insure." On the other hand, in the case of a car sales company database, the relationship between people and cars would be "purchase" or "buy." To solve this problem, you first need to know the domain of the target schema. However, even if you know the domain of the target schema, it is not easy to know which verb is appropriate for that domain. The entity name, that is, the noun, some method, such as

TF-IDF, can be used to extract nouns that are often used in a particular domain, but in verbs there is a variety of meanings in one verb. It is not easy to specify the domain of the verb. Finally, if both entities are human, this algorithm cannot extract a relationship name. This is because the relationship between people and people can be diverse. In the case of human nouns, collocation dictionaries often do not search for collocations.

7. Conclusion

Schemas integration as a systematic procedure of recognizing the similarities and reconciling the differences in the data descriptions of databases could verify useful in supporting the initial phases of developing a database. Therefore, much research has been done on schema integration, and, in recent decades, efforts have been made to build an automated schema integration system. However, most of the automated schema integration studies in the past have used XML as the source schema and still require some intervention by the DBA. The ER schema integration, which is difficult to be automatically performed, can be automatically done by converting the ER model to XML on the system. Problems related to relationship names that occur during the schema integration process entail more work than necessary for the DBA. Using the relationship name generation algorithm, proposed by our research, can dramatically shorten this process and time. If DBA apply our methodology to schema integration, DBA can save his or her time and power to the analysis and modelling scripts and rules. Furthermore, the database integration of even complex information systems could be an easier task if it was based on our approach.

Reference

Algergawy, A., Richi, N., and Gunter S, "Element similarity measures in XML schema matching." *Information Sciences,* Vol. 180, No. 24 (2010), 4975-4998.

Batini, C., and Lenzerini, M, "A methodology for data schema integration in the entity relationship model," *IEEE Transactions on Software Engineering*, Vol.10, No.6 (1984), 650-664.

Batini, C., Lenzerini, M., and Navathe, S. B, "A comparative analysis of methodologies for database schema integration," *ACM computing surveys,* Vol.18, No.4 (1986), 323-364.

Beeri, C., and Milo, T, "Schemas for integration and translation of structured and semi-structured data," *International conference on database theory*, Springer Berlin Heidelberg, 1999.

Castano, S., De Antonellis, V., Fugini, M. G., and Pernici, B, "Conceptual schema analysis: techniques and applications," *ACM Transactions on Database Systems*, Vol. 23, No.3 (1998), 286-333.

Chau, P. Y., and Hu, P. J. H., “Information technology acceptance by individual professionals: A model comparison approach," *Decision sciences*, Vol. 32, No. 4 (2001), 699-719.

Chen, P. P. S, "English sentence structure and entity-relationship diagrams," *Information Sciences,* Vol.29, No.2 (1983), 127-149.

Chen, P. P. S, "The entity-relationship model—toward a unified view of data." *ACM Transactions on Database Systems*, Vo.1, No.1 (1976), 9-36.

Date, C. J. (1990). An Introduction to Database Systems, Vol. 1, Fifth Edn, Reading: Addison-Wesley.

Davies, I., Green, P., Rosemann, M., Indulska, M., and Gallo, S. " How do practitioners use conceptual modeling in practice?" *Data & Knowledge Engineering*, Vol. 58, No. 3 (2006), 358-380.

Gotthard, W., Lockemann, P. C., and Neufeld, A, "System-guided view integration for object-oriented databases," *IEEE Transactions on knowledge and Data Engineering,* Vol.4, No.1 (1992), 1-22.

Gou, G., and Rada C, "Efficiently querying large XML data repositories: A survey." *IEEE Transactions on Knowledge and Data Engineering*, Vol.19, No. 10 (2007), 1381-1430

Hayne, S., and Ram, S, "Multi-user view integration system (MUVIS): An expert system for view integration," *Data Engineering*, 1990.

Jin, S., and Kang, W, "Mapping Rules for ER to XML Using XML schema," *Proceedings 10th Southern Association for Information Systems Conference*. Jacksonville, Florida, USA. 2007.

Kaul, M., Drosten, K., and Neuhold, E. J, "Viewsystem: Integrating heterogeneous information bases by object-oriented views," *Data Engineering*, 1990.

Kwan, I., and Fong, J, "Schema integration methodology and its verification by use of information capacity," *Information Systems*, Vol. 24, No.5 (1999), 355-376.

Lee, M. L., and Ling, T. W, "A methodology for structural conflict resolution in the integration of entity-relationship schemas," *Knowledge and Information Systems*, Vol.5, No.2 (2003), 225-247.

Melnik, S., Rahm, E., and Bernstein, P. A, "Rondo: A programming platform for generic model management," Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003.

Motro, A, "Superviews: Virtual integration of multiple databases," *IEEE Transactions on Software Engineering*, Vol.7 (1987), 785-798.

Pottinger, R., and Bernstein, P. A, "Schema merging and mapping creation for relational sources," Proceedings of the *11th international conference on extending database technology: Advances in database technology.* ACM, 2008.

Spaccapietra, S., and Parent, C, "View integration: A step forward in solving structural conflicts," *IEEE transactions on Knowledge and data Engineering,* Vol. 6, No.2 (1994), 258-274.

Spaccapietra, S., Parent, C., and Dupont, Y, "Model independent assertions for integration of heterogeneous schemas," *The International Journal on Very Large Data Bases,* Vol.1, No.1 (1992), 81-126.

Storey, V. C, "Understanding semantic relationships," *The International Journal on Very Large Data Bases* Vol.2, No.4 (1993), 455-488.

Suh, J., and Jinsoo P, "Effects of Domain Familiarity on Conceptual Modeling Performance." *Journal of Database Management*, Vol 28, No. 2 (2017), 27-55.

Unal, O., and Afsarmanesh, H, "Semi-automated schema integration with SASMINT," *Knowledge and information systems,* Vol.23, No.1 (2010), 99-128.

Zerdazi, A., and Myriam L, "Matching of Enhanced XML Schemas with a Measure of Structural-context Similarity." *WEBIST* (2007)

국문요약

엔티티 간의 관계명을 생성하는 알고리즘: 반자동화된 스키마 통합

김용찬* · 박진수* · 서지혜**

데이터 베이스 스키마 통합은 정보 시스템에서 매우 중요한 이슈이다. 스키마 통합은 시간과 노력이 상당히 많이 필요하기 때문에 그동안 많은 연구들은 자동화된 스키마 통합 시스템을 구축하기 위해 노력했다. 하지만 지금까지의 연구에서는 XML을 소스 스키마로 사용하고 여전히 많은 부분을 데이터 베이스 관리자의 개입이 필요하도록 남겨두었다. 예를 들면, 스키마 통합 시 발생하는 관계명 명칭 충돌과 같은 문제는 데이터 베이스 관리자가 직접 개입하여야 해결할 수 있었다. 이 논문에서는 스키마 통합 시 발생하는 관계명 명칭 충돌을 해결하기 위해 관계명을 자동으로 생성해주는 알고리즘을 소개한다. 이 알고리즘은 인터넷 연어(Collocation) 사전과 영어 예문을 기반으로 한다. 사전 데이터를 기반으로 하여 추출한 예문들을 자연어처리 과정을 통해 분석한 후 두 엔티티 사이의 관계명을 생성한다. 반자동화된 스키마 통합 시스템을 구축하여 이 알고리즘을 테스트해보았으며 그 결과 약 90%의 정확도를 나타냈다. 이 알고리즘을 적용하면 스키마 통합 시에 데이터 베이스 관리자의 개입을 최소화할 수 있으며 이는 자동화된 스키마 통합 시스템을 구축하는 데에 큰 도움이 될 것이다.

주제어 : 스키마 통합, 자연어 처리, 명칭 충돌, 개체관계모델, XML

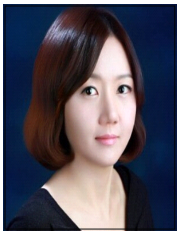
논문접수일 : 2018년 6월 3일　논문수정일 : 2018년 9월 17일　게재확정일 : 2018년 9월 28일
원고유형 : 일반논문(급행)　교신저자 : 서지혜

* 서울대학교 경영대학 경영정보시스템
** 교신저자: 서지혜
서울대학교 빅데이터 연구원
416 Gaepo-ro, Gangnam-gu, Seoul, 06324, Korea
Tel: +82-10-8770-3593, Fax: +82-2-573-0985, E-mail: jihaesuh77@snu.ac.kr

저자소개

김 용 찬
서울대학교에서 석사학위를 취득하였고 고용노동부가 지원하는 서울대학교 4차산업 혁명 아카데미, 서울시가 지원하는 서울대학교 빅데이터 아카데미에 연구원으로 참여하였다. 관심분야는 데이터 마이팅, 데이터 통합 등이다.

박 진 수
현재 서울대학교 경영대학 교수로 재직 중이다. 미국 미네소타대학 조교수, 고려대학교 조교수를 역임하였다. 관심분야는 온톨로지, 빅데이터 분석, 데이터 통합 등이다. MIS Quarterly, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Information Systems, IEEE Computer, Journal of Database Management, Data & Knowledge Engineering, The Data Base for Advances in Information Systems 등 국내외 저널에 다수의 논문을 게재하였다.



서 지 혜
현재 서울대학교 빅데이터연구원 연구교수로 재직 중이다. 서울대학교에서 경영정보시스템 전공으로 박사학위를 취득하였다고 관심분야는 빅데이터 분석, 데이터 마이닝, 데이터베이스 등이다.