



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

An algorithm for Finding a Relationship Between Entities :

Semi-Automated Schema Integration Approach

Yongchan Kim

College of Business Administration

Seoul National University

ABSTRACT

Database schema integration is a very important issue in information systems. Since schema integration is a time-consuming and labor-intensive task, many studies have attempted to automate this task. In the meantime, the researchers used xml as the source schema and still left much of the work to be done through DBA intervention. For example, 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 names conflicts that occur during schema integration. This algorithm is based on Internet collocation dictionary 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.

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

Student Number: 2015-20590

TABLE OF CONTENTS

1. Introduction.....	5
2. Methodologies for semi-automated schema integration.....	7
3. An algorithm for finding a relationship between entities	18
4. Semi-Automated Schema Integration.....	24
5. Evaluation.....	27
6. Limitations	28
7. Conclusion	29
Reference.....	30

LIST OF TABLES

Table 1. Similarity measure used for each schema element.....	14
Table 2. Formulas for entities and relationships	14
Table 3. Formulars for attributes	14
Table 4. Result of entity similarity comparison.....	15
Table 5. Result of relationship similarity comparison.....	16
Table 6. Results of schema similarity comparison.....	28

LIST OF FIGURES

Figure 1. Mapping Rules for ER to XML	9
Figure 2. "Order Management" Schema 1	10
Figure 3. "Order Management" Schema 2	10
Figure 4. XML document of schema 1	11
Figure 5. XML document of schema 2	12
Figure 6. XML Schema of XML Document.....	13
Figure 7. Transformation of an entity type attribute A into an entity type EA	17
Figure 8. An example of relationship name conflict.....	18
Figure 9. Another example of relationship name conflict	18
Figure 10. Comparison of various dictionaries	19
Figure 11. Algorithm 1	20
Figure 12. Algorithm 2	20
Figure 13. Search results for car in collocation dictionary.....	21
Figure 14. Search results for "person drive car" in Naver dictionary	22
Figure 15. The result of dependency analysis	22
Figure 16. Occurrence of appropriate examples for "Person + collocation + Car".....	23
Figure 17. Result of "Person" and "Car"	24
Figure 18. Result of 19 entity name pairs.....	24
Figure 19. Change the name of elements (Schema 2)	25
Figure 20. Make "Supply_Code" into an entity (Schema 2).....	25
Figure 21. Apply the algorithm to conflicting or newly created relationship name (Schema 2).....	26
Figure 22. Integrated schema – generated by tool (Sitool)	26
Figure 23. Integrated schema – expert 1.....	27
Figure 24. Integrated schema – expert 2.....	28

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

1. Introduction

Conceptual modeling has assumed a relevant role in the development of information systems and of software applications. In fact, conceptual modeling is an essential phase in database design [Castano 1991]. Conceptual modeling of data is a part of most applied system development methods [Jacobson 1992; Yourdon 1989]; and enterprise modeling has emerged as a preliminary design phase in software systems 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 2010]. Moreover, "even a single enterprise may have heterogeneous information bases for reasons of history or departmental autonomy" [Kaul, 1990]. As a result, Interoperability is becoming one of the most critical issues for medium to large size enterprises [Spaccapietra, 1992].

Schema integration is defined as the activity of integrating the schemas of existing or proposed databases into a global, unified schema. [Batini 86] Two types of schema integration are defined: (1) View Integration, which is performed during the database design process, for example at the conceptual design phase, and (2) Database Integration, which produces the global schema of a number of databases [Batini 86]. Schema integration has been a fundamental issue in data sharing among distributed, heterogeneous, and autonomous databases. With the increasing number of databases, integration problem has 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 2010] There has been a large amount of work in the integration area. Batini et al (1986) give a detailed survey of methodologies for view integration and database integration. New contributions often appear in the literature (Motro, 1987; Civelek et al, 1988; Diet and Lochovsky, 1989; Sheth and Gala, 1989; Siegel and Madnick, 1989; Hayne and Ram, 1990; Kaul et al, 1990; Siegel and Madnick, 1991; 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 such as 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 provide are general guidelines and concepts on different steps of the integration process. However, since 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 2010] A number of recent efforts focused on semi-automatic schema integration or merging, including [Chiticariu et al. 2008], [Melnik et al. 2003], [Pottinger and Bernstein 2003, 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 systems 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 describe accurately database applications. Furthermore, the schema integration studies had to deal 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 very 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 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 up, the main contributions of this study can be stated 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, 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 and we conclude in Section 5.

2. Methodologies for semi-automated schema integration

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the 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 E-R model gives graphical and diagrammatical representation of various entities, its attributes and relationships between entities. This in turn helps in the clear understanding of the data structure and in minimizing redundancy and

other problems. Nevertheless, the ER model is easy for humans to handle, but the machine cannot understand and handle it. 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 on the Intranet due to its self-describing data capability and flexibility in organizing data [Abiteboul et al. 2000, Gou, and Chirkova 2007]. The XML tag names are readable and convey the meaning of the data. The information structure is easily discerned by both 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. Since 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[17]'s 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 6 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 sub-element *<attribute>* of the corresponding top-level element in XML. The ER model used in this study contains only entities and binary relations. Entities and binary relations of the ER model, and how the attributes are converted to XML, are as follows.

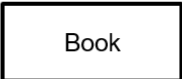
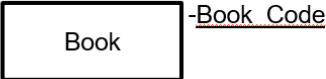

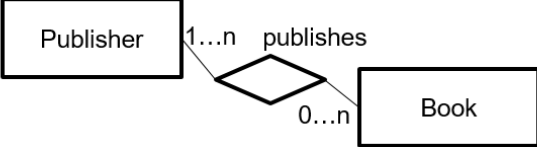
	ER	XML
Entity		<code><entity>Book</entity></code>
Key-Attribute		<code><entity>Book <key-attribute>Book_Code</key-attribute> </entity></code>
Attribute		<code><entity>Book <attribute>Title</attribute> </entity></code>
Binary Relationship		<code><binary-relationship>publishes <entity min-card="1" max-card="unbounded">Publisher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

Figure 1. Mapping Rules for ER to XML

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 sub-element of the top-level element EA 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 sub-element 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 sub-elements. 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).

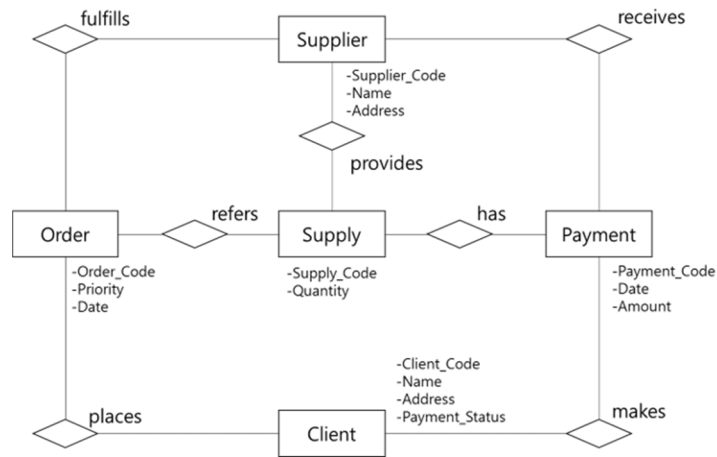


Figure 2. "Order Management" Schema 1

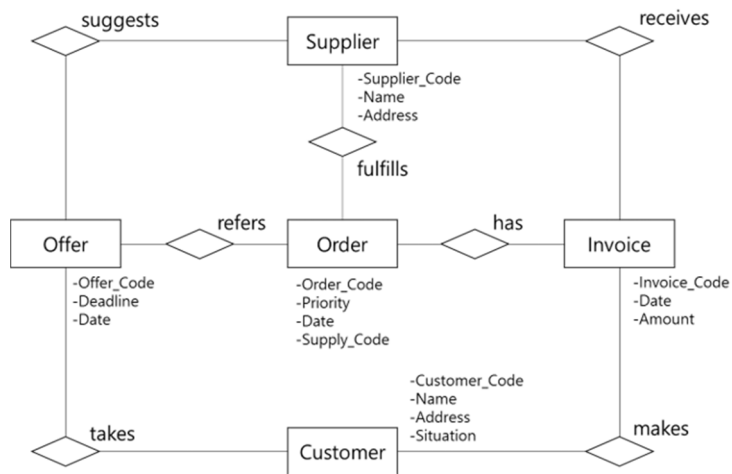


Figure 3. "Order Management" Schema 2

Figure 2 and Figure 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. Figure 4 and 5 are an ER model converted into a XML document, and Figure 6 is a 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>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>

```

```

  <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="unbounded">Supply</entity>
    <entity min-card="1" max-card="1">Order</entity>
  </binary-relationship>
  <binary-relationship>places
    <entity min-card="1" max-card="unbounded">Client</entity>
    <entity min-card="0" max-card="unbounded">Order</entity>
  </binary-relationship>
</erd>

```

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>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>

```

```

  <binary-relationship>suggests
    <entity min-card="1" max-card="unbounded">Supplier</entity>
    <entity min-card="0" max-card="unbounded">Offer</entity>
  </binary-relationship>
  <binary-relationship>takes
    <entity min-card="1" max-card="unbounded">Customer</entity>
    <entity min-card="0" max-card="unbounded">Offer</entity>
  </binary-relationship>

```

```

</erd>

```

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 that finds the corresponding pair for each other with the transformed XML. In this process, we adopted Algergawy et al[1]'s measurement method. They 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 obtained the internal and external element similarity values, 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.

	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 1. Similarity measure used for each schema element

Measuring Similarities between Entities / Relationships	
Internal similarity measures	External similarity measures
Name similarity	Leaf context similarity
$sim_{WuPalmer}(t_1, t_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3}$	$Sim(\mathcal{E}l_1, \mathcal{E}l_2) = \frac{\sum_{i=1}^{i=k} \left[\max_{j=i}^{j=k'} InterSim(\mathcal{E}l_{1i}, \mathcal{E}l_{2j}) \right]}{\max(k , k')}$

Table 2. Formulas for entities and relationships

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

Table 3. Formulars for attributes

Since 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). For attribute internal similarity measure, name, constraint, and data type were measured. As an external similarity measurement method, we use the ancestor context similarity. The table 4 and 5 show the results from the similarity measure.

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 4. Result of entity 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

Table 5. Result of relationship similarity comparison

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

$S1.Supplier \equiv S2.Supplier$

$S1.Payment \cong S2.Invoice$

$S1.Order \cong S2.Order$

$S1.Client \cong S2.Customer$

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.receive \cong S2.receive

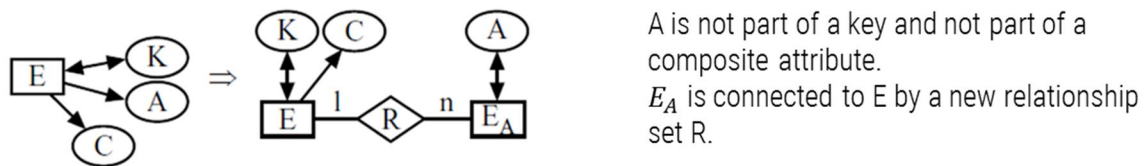
S1.has \cong S2.has

S1.makes \cong S2.makes

S1.fulfills \cong S2.fulfills

S1.refers \cong 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[20]'s study to solve structural conflicts. They 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.



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

To check if there is a structural conflict, we need to make sure 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 attribute of the *S1.Supply* entity in Schema 1, and is the simple attribute of *S2.Order* in the 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 following relationship naming conflict may occur.

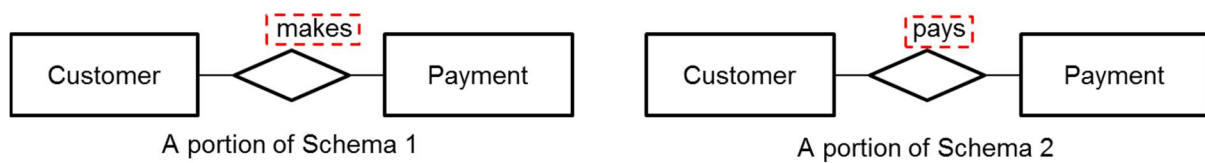


Figure 8. An example of relationship name conflict

This is a very 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.

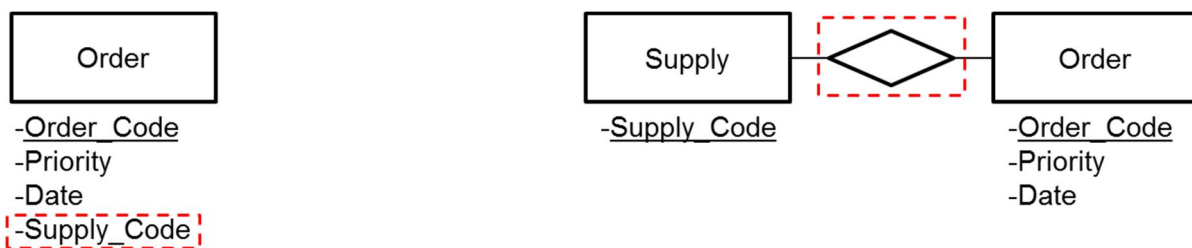


Figure 9. Another example of relationship name conflict

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 very time consuming and labor intensive.

This algorithm automatically generate relationship names in these case. Briefly, our algorithm first searches the internet collocation 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 the most examples are present.

According to Chen[7]'s 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 the more similar sentences including the entities (noun) and the relation (verb) exist, 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 dictionaries from which to extract example sentence set. The comparison criterion was how many examples were searched and whether they supported complex search.








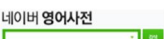
Dictionary	Number of Examples	Complex Search
 Cambridge Dictionary	12	X
 English Oxford Living Dictionaries	42	X
 Merriam-Webster	3	X
 LONGMAN	15	X
 Dictionary.com	10	X
 Collins	19	X
 MACMILLAN DICTIONARY	5	X
 네이버 영어사전	122,809	O (person car : 5,102)

Figure 10. Comparison of various dictionaries

As a result, it was confirmed that Naver English dictionary was overwhelmingly used in the 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

<p>Input : Entity Name1, Entity Name2, Relationship Name1, Relationship Name2</p> <p>Output : Relationship name between Entity1 and Entity2</p>
<p>Step 1. Search “Entity Name1 + Relationship Name1 + Entity Name2” in dictionary and collect the example sentences, repeat for relationship name 2</p> <p>Step 2. Processing Part-of-Speech(POS) and dependency analysis for each sentence</p> <p>Step 3. Counting the occurrence of appropriate examples for each collocation verb</p> <p>Step 4. The most frequently used verb is adopted as the final relationship name.</p>

Figure 11. Algorithm 1

Algorithm 2

<p>Input : Entity Name1, Entity Name2</p> <p>Output : Relationship name between Entity1 and Entity2</p>
<p>Step 1. Search collocations of entity names (Input : Person, Car)</p> <p>Step 2. Search “Entity Name1 + A verb extracted from the collocation set + Entity Name2” in dictionary and collect the example sentences</p> <p>Step 3. Processing Part-of-Speech(POS) and dependency analysis for each sentence</p> <p>Step 4. Counting the occurrence of appropriate examples for each collocation verb</p> <p>Step 5. The most frequently used verb is adopted as the final relationship name.</p>

Figure 12. Algorithm 2

In case of Figure 8, since two relation names have already been given, the step for searching the collocations of entities is skipped. 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, the 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.

Since 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

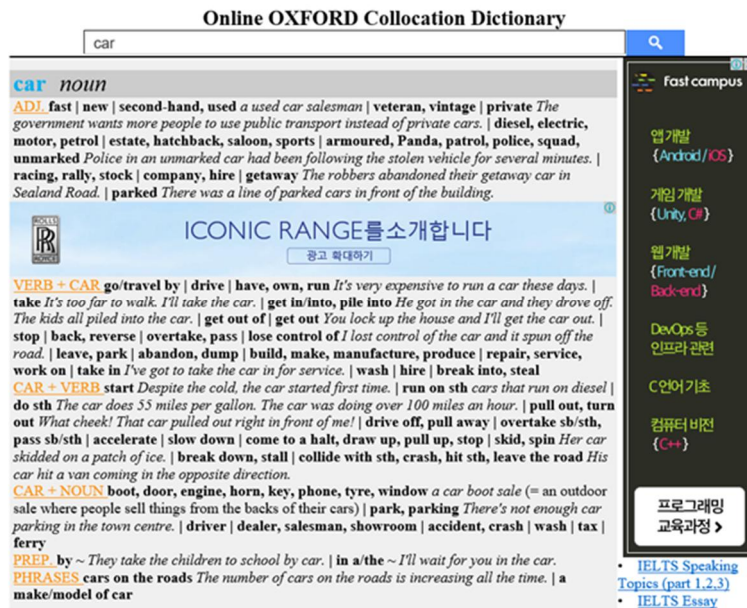


Figure 13. Search results for car in collocation dictionary

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 dictionary and collect the example sentences

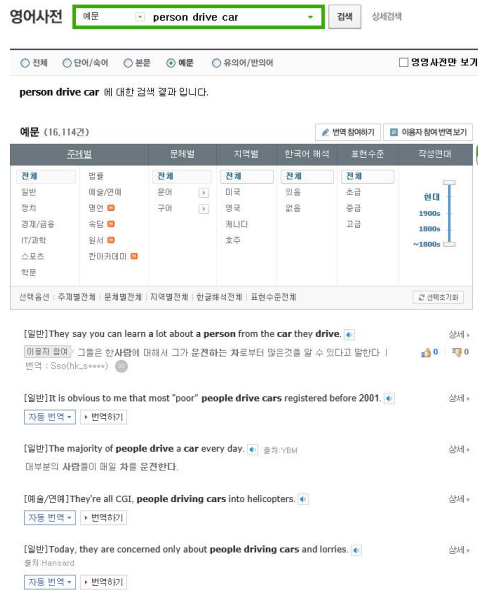


Figure 14. Search results for "person drive car" in Naver dictionary

Extracts the collocation one by one from the collocation set, and searches the combination of the collocation and entity names from the dictionary. The example sentences that are searched 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 analyze for each example sentence.

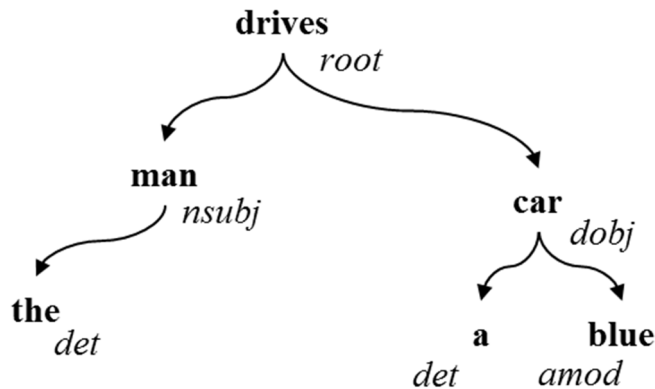


Figure 15. The result of dependency analysis

POS tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both 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 the Figure 15. The dependency "root" is grammatical relation that points 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.

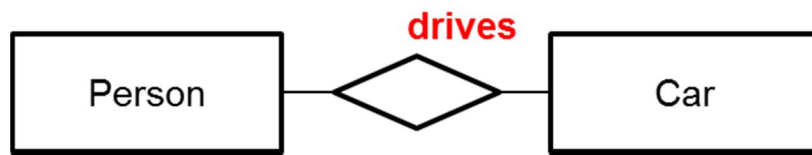


Figure 17. Result of "Person" and "Car"

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

Bank -Branch	Car -Person	Customer -Order	Member -Book	Publisher -Book	Student -Course	Book -Topic	Client -Account	Player -Team	Department -Project
open	drive	make	write	publish	take	cover	open	make	fund
Student -Staff	Nurse -Patient	Doctor -Nurse	Member -Librarian	Employee -Supervisor	Physician -Patient	Order -Payment	Employee -Department	Supplier -Order	
have	see	get	NONE	NONE	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. 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.

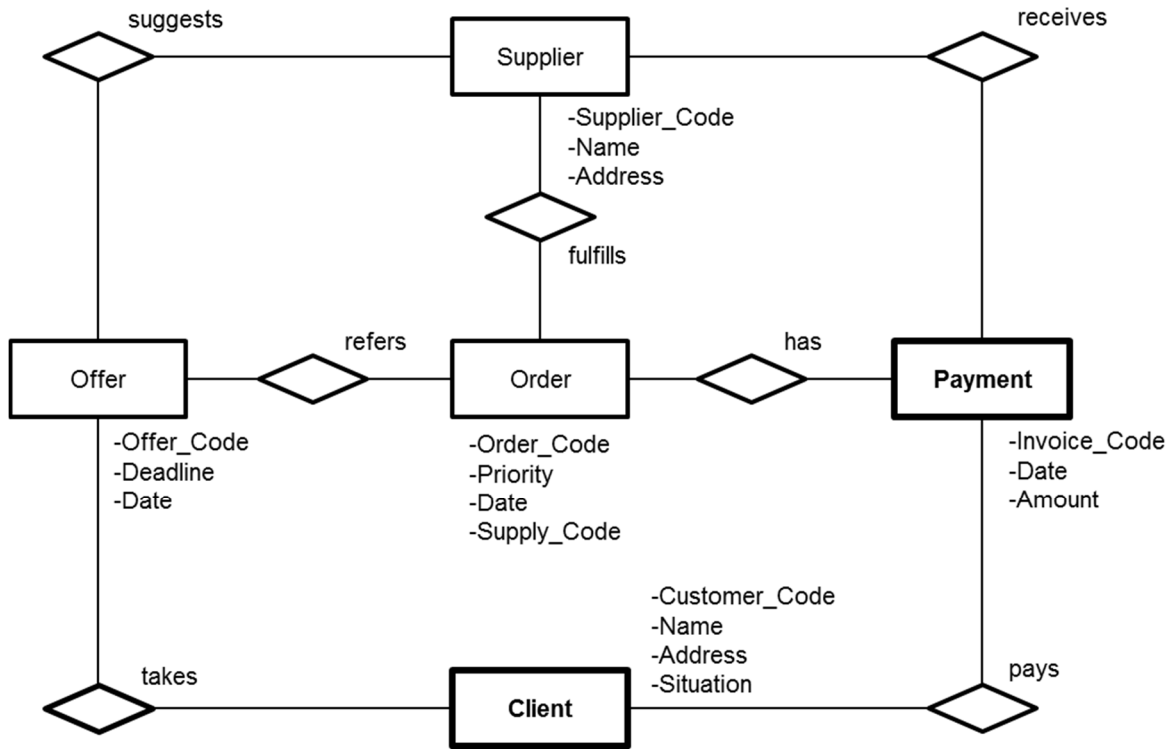


Figure 19. Change the name of elements (Schema 2)

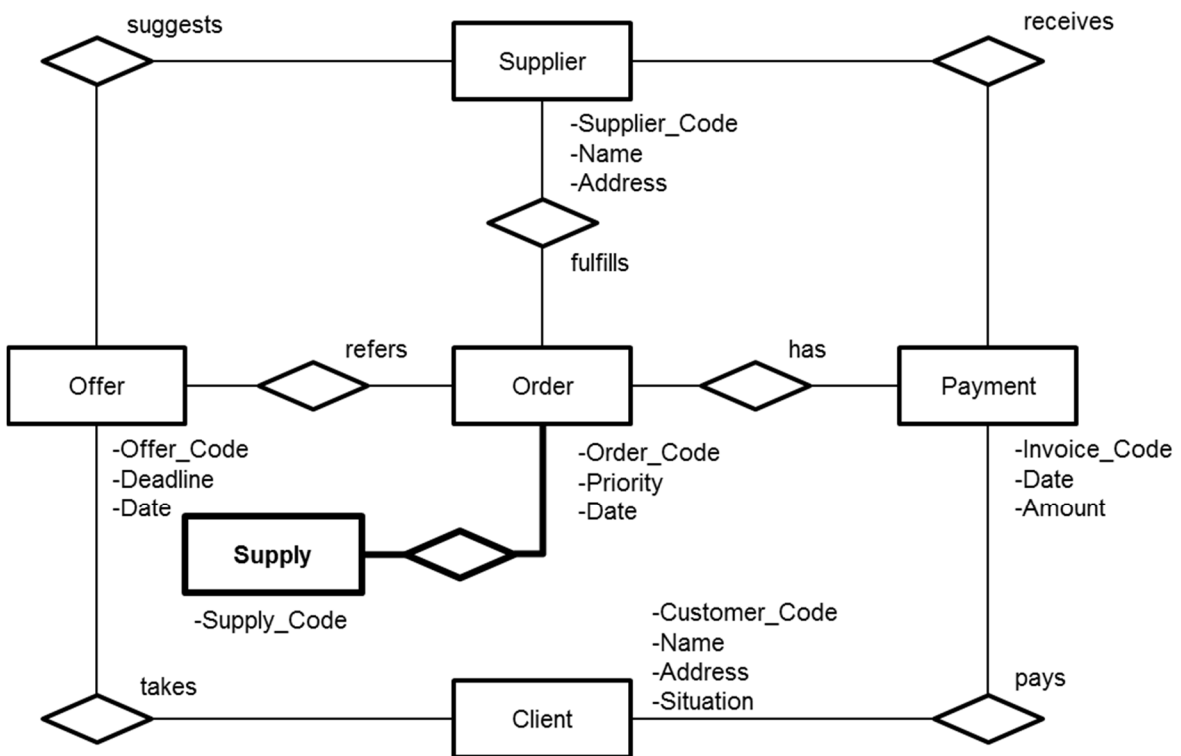


Figure 20. Make "Supply_Code" into an entity (Schema 2)

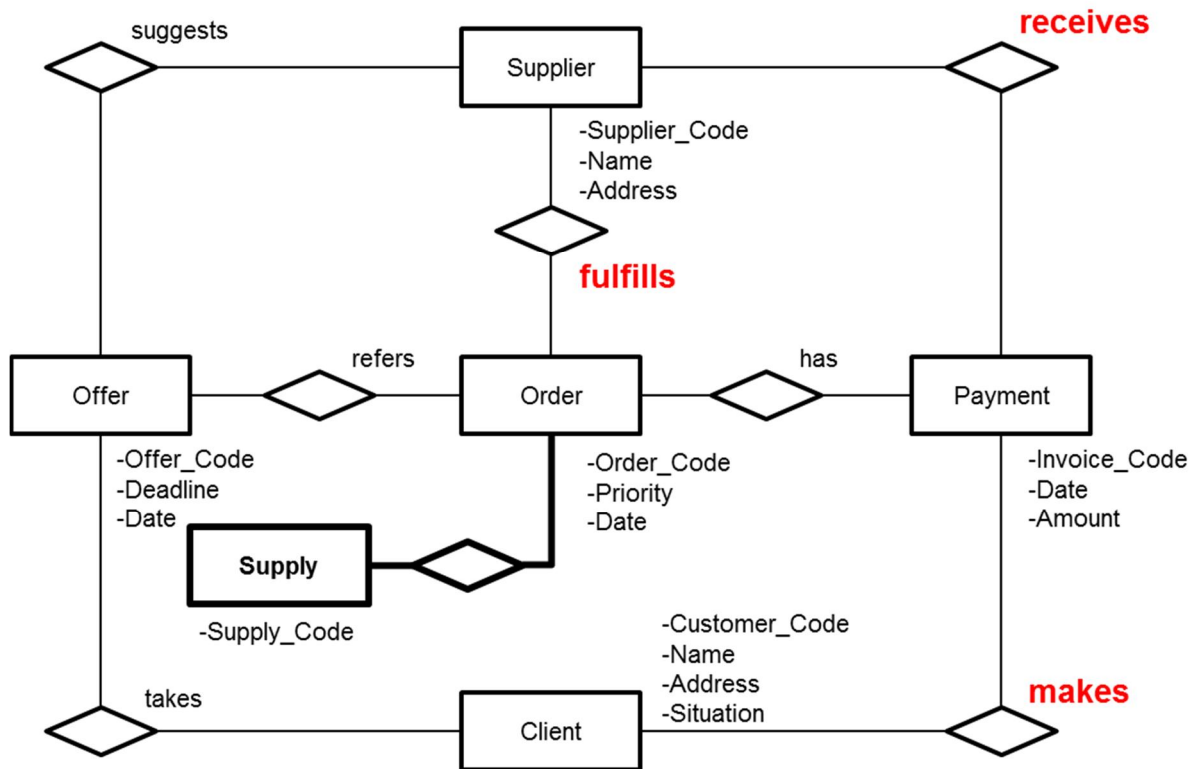


Figure 21. Apply the algorithm to conflicting or newly created relationship name (Schema 2)

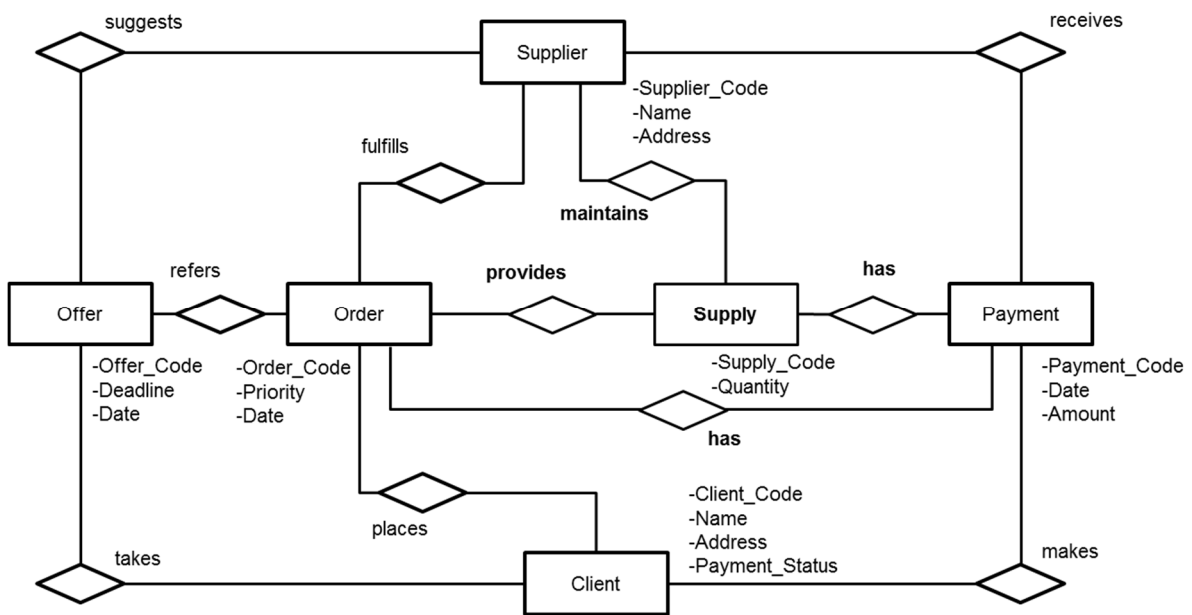


Figure 22. Integrated schema – generated by tool (Si_{tool})

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, 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.

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

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

The schema used for the evaluation was given from two experts.

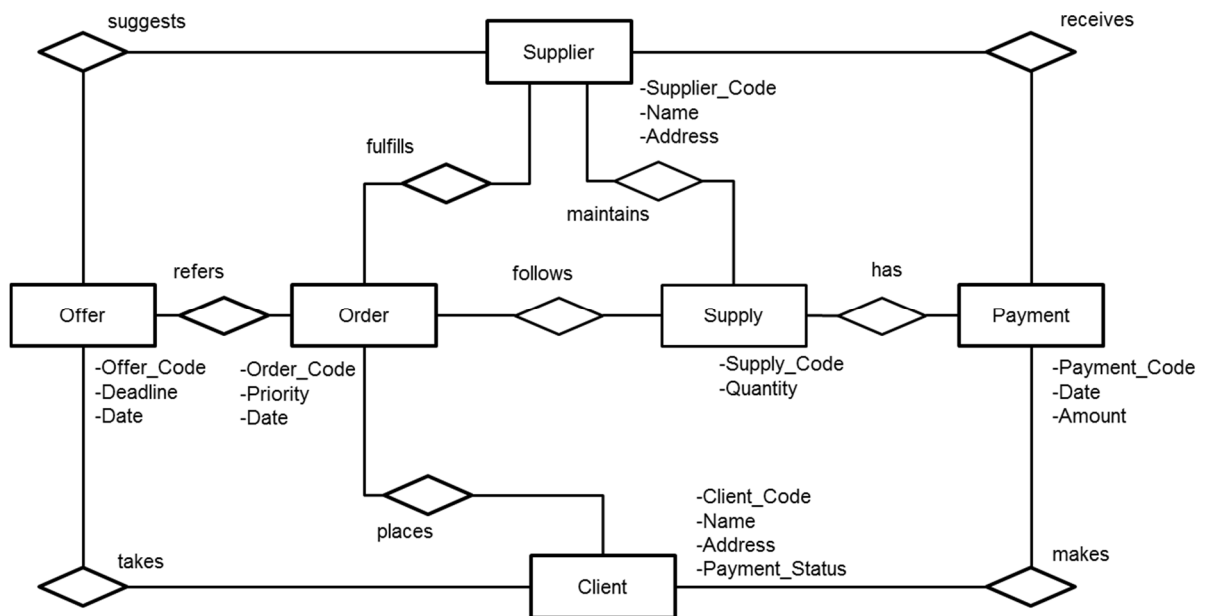


Figure 23. Integrated schema – expert 1

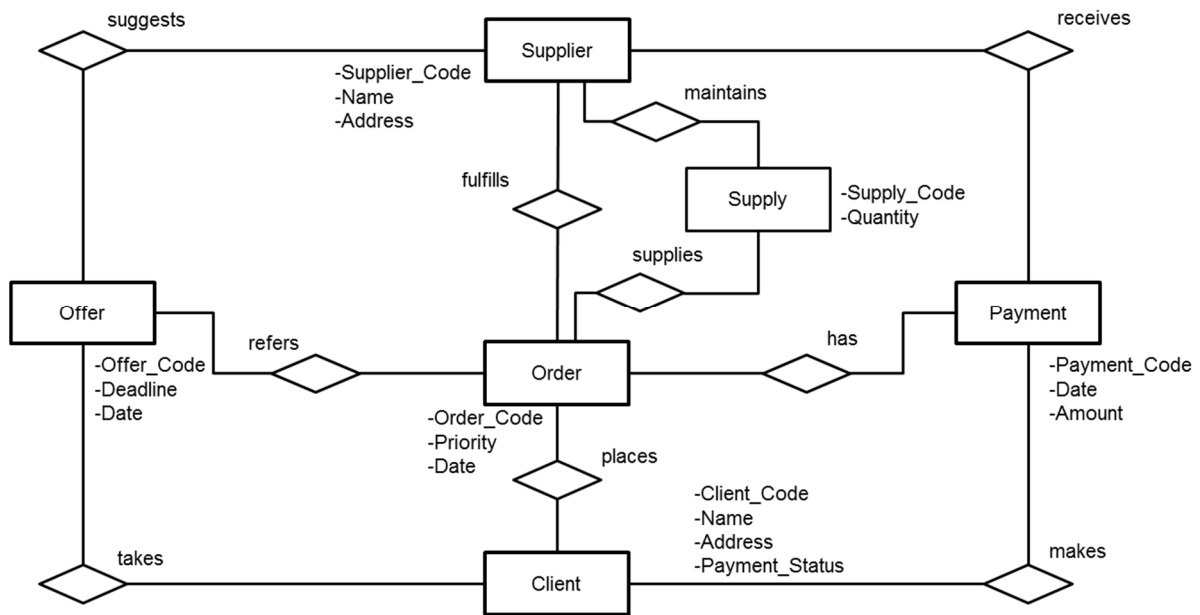


Figure 24. Integrated schema – expert 2

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

Table 6. Results of schema similarity comparison

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 can not be selected because two entity names are not given as input values required. Ternary relations can not be applied to this algorithm because proper example sentences can not be retrieved due to limitations of dictionary search function. 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 domain. 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 can not extract a relationship name. This is because the relationship between people and people can be very diverse. In the case of human nouns, collocation dictionaries often do not search for collocations.

7. Conclusion

In the meantime, 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 can dramatically shorten this process.

Reference

1. Algergawy, Alsayed, Richi Nayak, and Gunter Saake. "Element similarity measures in X ML schema matching." *Information Sciences* 180.24 (2010): 4975-4998.
2. Batini, Carlo, and Maurizio Lenzerini. "A methodology for data schema integration in the entity relationship model." *IEEE Transactions on Software Engineering* 6 (1984): 650-664.
3. Batini, Carlo, Maurizio Lenzerini, and Shamkant B. Navathe. "A comparative analysis of methodologies for database schema integration." *ACM computing surveys (CSUR)* 18.4 (1986): 323-364.
4. Beeri, Catriel, and Tova Milo. "Schemas for integration and translation of structured and semi-structured data." *International conference on database theory*. Springer Berlin Heidelberg, 1999.
5. Castano, Silvana, et al. "Conceptual schema analysis: techniques and applications." *ACM Transactions on Database Systems (TODS)* 23.3 (1998): 286-333.
6. Chaffin, Roger, Douglas J. Herrmann, and Morton Winston. "An empirical taxonomy of part-whole relations: Effects of part-whole relation type on relation identification." *Language and Cognitive processes* 3.1 (1988): 17-48.
7. Chen, Peter Pin-Shan. "English sentence structure and entity-relationship diagrams." *Information Sciences* 29.2 (1983): 127-149.
8. Chen, Peter Pin-Shan. "The entity-relationship model—toward a unified view of data." *ACM Transactions on Database Systems (TODS)* 1.1 (1976): 9-36.
9. Chiticariu, Laura, Phokion G. Kolaitis, and Lucian Popa. "Interactive generation of integrated schemas." *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008.

10. Civelek, Ferda N., Asuman Dogac, and Stefano Spaccapietra. "An expert system approach to view definition and integration." Proceedings of the Seventh International Conference on Entity-Relationship Approach: A Bridge to the User. North-Holland Publishing Co., 1988.
11. Diet, Jürgen, and Frederick H. Lochovsky. "Interactive specification and integration of user views using forms." Proceedings of the Eight International Conference on Entity-Relationship Approach to Database Design and Querying. North-Holland Publishing Co., 1989.
12. Duchateau, Fabien, and Zohra Bellahsene. "Measuring the quality of an integrated schema." International Conference on Conceptual Modeling. Springer Berlin Heidelberg, 2010.
13. Gotthard, Willi, Peter C. Lockemann, and Andrea Neufeld. "System-guided view integration for object-oriented databases." IEEE Transactions on knowledge and Data Engineering 4.1 (1992): 1-22.
14. Hakimpour, Farshad, and Andreas Geppert. "Resolving semantic heterogeneity in schema integration." Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001. ACM, 2001.
15. Halevy, Alon, Anand Rajaraman, and Joann Ordille. "Data integration: the teenage years." Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006.
16. Hayne, Stephen, and Sudha Ram. "Multi-user view integration system (MUVIS): An expert system for view integration." Data Engineering, 1990. Proceedings. Sixth International Conference on. IEEE, 1990.
17. Jin, Sung, and Woohyun Kang. "Mapping Rules for ER to XML Using XML schema." Proc. 10th Southern Association for Information Systems Conference. Jacksonville, Florida, USA. 2007.

18. Kaul, Manfred, Klaus Drost, and Erich J. Neuhold. "Viewsystem: Integrating heterogeneous information bases by object-oriented views." *Data Engineering*, 1990. Proceedings. Sixth International Conference on. IEEE, 1990.
19. Kwan, Irene, and Joseph Fong. "Schema integration methodology and its verification by use of information capacity." *Information Systems* 24.5 (1999): 355-376.
20. Lee, MongLi, and TokWang Ling. "A methodology for structural conflict resolution in the integration of entity-relationship schemas." *Knowledge and Information Systems* 5.2 (2003): 225-247.
21. Madria, Sanjay, Kalpdrum Passi, and Sourav Bhowmick. "An XML Schema integration and query mechanism system." *Data & Knowledge Engineering* 65.2 (2008): 266-303.
22. Melnik, Sergey, Erhard Rahm, and Philip A. Bernstein. "Rondo: A programming platform for generic model management." *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003.
23. Motro, Amihai. "Superviews: Virtual integration of multiple databases." *IEEE Transactions on Software Engineering* 7 (1987): 785-798.
24. Omar, Nazlia, J. R. P. Hanna, and Paul McKeivitt. "Heuristic-based entity-relationship modelling through natural language processing." *Artificial Intelligence and Cognitive Science Conference (AICS)*. Artificial Intelligence Association of Ireland (AIAI), 2004.
25. Passi, Kalpdrum, et al. "A model for XML Schema integration." *International Conference on Electronic Commerce and Web Technologies*. Springer Berlin Heidelberg, 2002.
26. Pottinger, Rachel A., and Philip A. Bernstein. "Merging models based on given correspondences." *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003.
27. Pottinger, Rachel, and Philip A. Bernstein. "Schema merging and mapping creation for r

- relational sources." Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM, 2008.
28. Rahm, Erhard, and Philip A. Bernstein. "A survey of approaches to automatic schema matching." *the VLDB Journal* 10.4 (2001): 334-350.
 29. Sheth, Amit P., and Sunit K. Gala. "Attribute relationships: An impediment in automating schema integration." (1989).
 30. Siegel, Michael, and Stuart E. Madnick. "Context interchange: sharing the meaning of data." *ACM SIGMOD Record* 20.4 (1991): 77-78.
 31. Spaccapietra, Stefano, and Christine Parent. "View integration: A step forward in solving structural conflicts." *IEEE transactions on Knowledge and data Engineering* 6.2 (1994): 258-274.
 32. Spaccapietra, Stefano, Christine Parent, and Yann Dupont. "Model independent assertions for integration of heterogeneous schemas." *The VLDB Journal—The International Journal on Very Large Data Bases* 1.1 (1992): 81-126.
 33. Storey, Veda C. "Understanding semantic relationships." *The VLDB Journal—The International Journal on Very Large Data Bases* 2.4 (1993): 455-488.
 34. Unal, Ozgul, and Hamideh Afsarmanesh. "Semi-automated schema integration with SAS MINT." *Knowledge and information systems* 23.1 (2010): 99-128.
 35. Wang, Stuart E. Madnick Y. Richard, et al. "CISL: composing answers from disparate information systems." (1989).
 36. Wermter, Joachim, and Udo Hahn. "You can't beat frequency (unless you use linguistic knowledge): a qualitative evaluation of association measures for collocation and term extraction." Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 2006.

김 용 찬
서울대학교 경영대학

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 XML 을 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.

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

학번: 2015-20590