한국항해항만학회 제29권 제2호 추계학술대회논문집 pp. 353~360, 2005. 10

XML을 이용한 데이터베이스 통합방안에 관한 연구

오세웅* · 이홍걸** · 이철영*** · 박종민**** · 서상현****

*한국해양대학교 대학원, **경남발전연구원, ***한국해양대학교 물류시스템공학과 교수, ****한국해양연구원

A Study on the Database Integration Methodology using XML

Se-Woong OH · Hong-Girl Lee** · Chul-Young Lee*** · Jong-Min Park**** · Sang-Hyung Suh****

Graduate school of Korea Maritime University, Busan 606-791, Korea

Gyeongnam Development Institute, ChangWon 641-060, Korea

Department of Logistics Engineering, Korea Maritime University, Busan 606-791, Korea

Korea Ocean Research & Development Institute, Daejeon 305-343, Korea

요 약 : 물류환경에 있어서 데이터베이스 통합의 문제는 중요한 과제로 인식되어 왔으나, 기존 연구들은 스키마 통합 시 발생하는 스키마 충돌을 이론적인 측면에서의 해결 방법만 제시하고 실제 시스템 구현 측면에서의 연구는 부족한 실정이다. 본 연구는 실제적인 DB 통합법과 관련하여 XML 기술을 이용한 통합법을 제시하고 통합 알고리즘으로 개체 및 속성간의 유사도 측정에 기반을 둔 계량화된 충돌 식별법을 사용한다. 구체적으로 DB 스키마를 XML 스키마로 변환시켜 개체명과 속성명을 추출한 다음, DB 통합 시 빈번히 발생하는 의미적 충돌(Semantic Conflict) 현상인 이른바 "Name Conflict"의 식별을 위한 하나의 해결법으로서 시소러스(Thesaurus) 사전과 형태소 분석을 통해, 개체 및 속성 간 종합적인 유사도 측정하는 계량화된 식별방안을 사용하였다.

핵심용어 : 물류정보시스템, 데이터베이스 통합, 스키마 통합, 의미적 충돌, XML Schema, XSLT

ABSTRACT : *Database Integration problems has been recognized as a critical issue for effective logistics service in logistics environment. However, researches related to effective methodology for this have been studied theoretically in the DB schema integration, are insufficient in the side of the system realization. The aim of this paper is to present a schema integration technique to integrate DB using XML(eXtensible Markup Language) in the part of practical DB integration, a quantitative methodology for the identification of conflict that is a representative problem on database integration. To achieve this aim, we extracted the entity name and attribute name from DB schema and suggested a quantitative methodology to easily fine name conflict that frequently give raise to a trouble when schema integration, based on the level of semantic similarity between attributes and entities.*

KEY WORDS : *Logistics information system, Database integration, Schema integration, Semantic conflict, XML Schema, XSLT*

1. 서 론

물류정보시스템은 분산된 환경에서 서로 다른 데이터베이스를 사용하여 구축되고 있다. 이러한 데이터베이스의 이질성은 데이터모델의 이질성, 데이터베이스 스키마의 이질성 등 다양한 형태를 가지고 있으며, 이러한 데이터베이스들의 상호 작동은 지역 데이터베이스의 자치성 유지 여부, 데이터베이스 이질성 여부, 데이터의 분산성 여부 등에 따라 통합 방법이 다양하게 결정된다. 데이터베이스의 상호 작동은 분산 데이터베이스, 연합 데이터베이스, 멀티 데이터베이스 등 여러 가지 개념으로 연구되어 왔고 이러한 연구의 공통 해결 요구사항은 이형, 이질의 데이터베이스 스키마 통합이다.

현재 물류정보시스템은 분산된 환경에서 지역 데이터베이스의 자치성은 유지하면서 데이터베이스들을 상호 공유하는 데이터 웨어하우징, 데이터마이닝 등의 응용이 많이 늘고 있다. 이러한 응용의 첫 번째 요구사항은 이질의 데이터베이스 통합이나 기존의 데이터베이스 통합은 통합을 위한 스키마 공통 모델의 부재, 스키마 통합 이론의 복잡성, 스키마 통합을 위한 실제적인 방법이 부재하여 어려움을 겪어 왔다.

본 연구에서는 현재까지 연구에 있어서 문제점인 표준 데이터

*대표저자 : 오세웅(학생회원), osw@moeri.re.kr, 016)418-5125

모델의 부재와 스키마 충돌 해결 도구의 부족을 해결하고자 웹 표준양식으로 각광을 받고 있는 XML Schema를 이용하여 스키마 통합을 수행하며 스키마 충돌개체 식별을 위해 유사도 측정에 기반을 둔 계량화된 충돌식별법을 사용하였다. 이 방법은 DB통합 시 빈번히 발생하는 의미적 충돌(Semantic Conflict) 현상인 이른바 "Name Conflict"의 식별을 위한 하나의 해결법으로서 시소러스(Thesaurus) 사전과 형태소 분석을 통해, 개체 및 속성간 종합적인 유사도 측정을 주 내용으로 한다.

2. 데이터베이스 통합 문제점과 기존연구

2.1 데이터베이스 통합 시 충돌의 분류

데이터베이스를 통합하는데 있어서 다음과 같이 문제점들이 발생된다.

1) 이름 충돌

이름 충돌(Naming Conflict)은 엔터티나 어트리뷰트 간의 이름의 상이함으로 발생하는 것으로 동일한 어트리뷰트가 각 정보 자원에서 다른 이름을 사용하는 경우(synomym)나 서로 다른 엔터티나 어트리뷰트가 각 정보 자원에서 동일한 이름을 사용하는 경우(homonym)에 발생한다. 이러한 경우에는 이름의 대응관계를 전역 스키마의 목록에 기록함으로써 해결하며, 전역 질의는 목록을 보고 적합한 지역 질의로 바꾸게 된다.

2) 구조 충돌

구조 충돌(Structural Conflict)은 각 정보 자원들의 구조가 다른 경우, 즉 정보자원 1에서는 엔터티인 데이터가 정보자원 2에서는 엔터티의 어트리뷰트로 사용되는 경우나 엔터티의 어트리뷰트 구성이 서로 다른 경우에 발생한다. 엔터티의 구성이 다른 경우, 즉 정보자원 1에서는 엔터티인 데이터가 정보자원 2에서는 엔터티의 어트리뷰트로 표현될 때 정보자원 2의 엔터티의 어트리뷰트를 다른 엔터티로 변환하거나 정보 자원 1의 엔터티를 정보자원 2처럼 엔터티의 어트리뷰트로 변환하여야 한다.

3) 데이터타입 충돌

데이터 타입 충돌(Type Conflict)은 동일한 어트리뷰트의 데이터 타입이 각 정보 자원마다 다른 경우 발생한다. 데이터 타입 충돌은 데이터 타입 변환 규칙이 있어야 한다. 변환 규칙은 시스템 설계자가 정보 손실을 최소로 하는 방법을 선택하여 결정하게 된다.

4) 단위 충돌

단위충돌(Measurement Conflict)은 동일한 어트리뷰트의 데이터 단위가 각 정보 자원마다 다른 경우, 즉 cm와 m, kg와 g, "monthly pay"와 "yearly pay"의 관계처럼 데이터 단위를 다르게 사용하는 경우에 발생한다. 이처럼 단위가 다른 경우에는 충돌 발생 시 변환 공식을 이용하여 해결해야 한다. 질의를 할 때나 질의 결과를 가져올 때도 항상 수식을 이용하여 값을 변환해야 한다.

5) 표현상의 충돌

표현상의 충돌(Representation Conflict)은 동일한 의미의 데이터를 각각의 정보 자원에서 다르게 표현하는 경우, 즉 학생의 성적을 "A, B, C"와 "Excellent, Very Good, Good" 등으로 다르게 표현하는 경우에 발생한다. 이러한 충돌은 변환규칙을 이용하여 일대일 대응시켜 통합한다.

2.2 기존 연구 검토

데이터베이스 스키마 통합을 위한 연구들은 주로 스키마 통합 규칙을 찾는 것과 스키마 통합을 위한 공통 모델의 개발에 치중해 왔다. 스키마의 통합에 있어서 통합 규칙은 종류가 다양할 뿐만 아니라, 규칙을 포현하는 표현형식이 대부분 수학적인 공식으로 이루어져 있어 통합하는 방식에 대한 이해가 어렵고, 복잡한 규칙을 기반으로 하기 때문에 통합을 위한 프로그램을 개발하기 어렵다. 또 통합 모델은 초기에는 개발자들이 임의로 정의하여 사용하였고, 93년에 만들어진 ODMG의 객체 지향 데이터베이스 모델의 표준화 후에는 ODMG 모델을 사용하는 연구가 많이 이루어졌다. ODMG 모델을 사용하는 방법은 스키마를 객체지향 데이터 모델로 변환하고 OQL을 사용하여 데이터베이스를 필터링하며 포맷팅한다.

Soon M. Chung은 이질의 멀티 데이터베이스 시스템에서 하나의 인터페이스로서 다양한 데이터베이스를 접근하는데 있어서 스키마 충돌, 다른 데이터모델의 이질성과 데이터 불일치 등과 같은 문제가 발생하는 것을 해결하기 위해 Unified Model을 제안하였다. 이 모델은 스키마 통합에 적절한 객체모델과 질의 변화에 용이한 관계 모델의 각각에 대한 장점들의 조합이다.

Lawrence는 관계데이터베이스 스키마의 통합과 이에 따른 데이터 매핑 문제를 XML 기술을 이용하여 시도하였다. 관계 데이터베이스 스키마를 XML Schema 문법과 유사한 X-Spec을 사용하여 표현하며 표준화된 스키마 표현을 시도하였다. 스키마간의 충돌문제는 스키마를 구성하는 원소들 간에 사전(Dictionary)을 이용한 매핑으로 통합된 스키마와 소스 데이터 간의 대응관계를 해결하였다.

3. XML Schema 표현 및 개체충돌 식별방법

3.1 데이터베이스 스키마와 XML Schema

데이터베이스 스키마 통합을 용이하게 하기 위한 기존의 연구

들이 많이 있다. 기존의 연구들에서 스키마 충돌 해결을 쉽게 하기 위하여 중간 모델로 GDM(Generic Data Model), Object-Oriented 개념의 기반으로 한 Unified Model 등을 사용하였다. 본 연구에서는 XML 표준에 의한 XML Schema를 데이터베이스 스키마를 표현하는 공통된 모델로 사용한다. 공통된 데이터 모델로 XML Schema를 사용하면 여러 가지 장점을 얻을 수 있다. 표준화된 모델을 사용함으로서 데이터베이스 간의 통합 및 스키마간의 변환을 용이하게 하고, 변환 프로그램을 재활용하기 쉽다.

중간모델로 XML Schema를 사용하기 위해서는 기존의 데이터베이스를 XML Schema 로 변환하는 방법이 필요하다. 스키마는 데이터베이스에 정의된 데이터 타입을 XML Schema의 타입으로 변환하며 이 과정은 해당 데이터베이스 시스템에 따라 변환기를 만들어 자동화 할 수 있다. 변환 알고리즘은 다음과 같으며 Duckett에 자세히 기술되어 있다. 개략적인 변환 알고리즘은 다음과 같다.

(1) 스키마의 이름을 루트 엘리먼트(root element)로 정의한다.
(2) 데이터베이스 테이블 속성들의 데이터 타입은 simpleType 엘리먼트로 정의한다.
(3) 테이블 구조는 complexType 엘리먼트와 attribute 엘리먼트로 정의한다.
(4) 테이블간 참조는 KEY/KEYREF 엘리먼트 및 부모-자식 관계의 엘리먼트를 이용하여 정의한다.

```
CREATE TABLE Expensive_car (
        modelname           varchar(50) PRIMARY KEY,
        manufacturervarchar(50),
        maximumspeed        int,
        price               int
)
```

Fig. 1 Database Schema(SQL)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
  <xs:element name="Expensive_car" type="cartype">
  <xs:complexType name="cartype">
    <xs:attribute name="modelname" type="nametype"/>
    <xs:attribute name="manufacturer" type="nametype"/>
    <xs:attribute name="maximumspeed" type="valuetype"/>
    <xs:attribute name="price" type="valuetype"/>
  </xs:complexType>
  <xs:simpleType name="nametype">
    <xs:restriction base="xs:string">
      <xs:length value="50"/>
    </xs:restriction>
  <xs:simpleType name="valuetype">
    <xs:restriction base="xs:integer"/>
  </xs:simpleType>
</xs:schema>
```

Fig. 2 Database Schema Presentation using XML Schema

3.2 스키마간 대응규칙

문법적인 구조를 갖는 문서를 다른 문서로 변환하는 방법은 SDTS, AG, TT-grammer 등 여러 가지 기법이 있는데 이 중 TT-grammer 같은 트리 구조를 갖는 문서를 변환하는 방법을 본 연구의 XML Schema 파일을 변환하는데 적용할 수 있다. XML 스키마 문서를 트리 구조로 볼 때 한 개의 트리를 다른 트리로 변환하는 작업은 몇 가지 기본적인 작업들이 합쳐져서 구성된다. 관계 데이터베이스 스키마를 XML 스키마로 표현하면 루트 노드는 데이터베이스 이름이 되고 자식 노드는 테이블이며 손자 노드는 속성의 이름인 높이가 3인 트리가 된다. 스키마의 통합은 2개의 트리를 1개의 트리로 만드는 과정이다. 스키마 통합연산은 다음과 같다.

(정의) 기호 T : 데이터베이스 스키마에 대한 테이블 이름을 나타낸다. <schema>는 스키마명, <tablename>은 테이블 이름을 나타낸다.
문법) T : <schema>.<tablename>

(정의) rename 연산, R : 테이블의 이름과 테이블에 속한 속성의 이름을 한꺼번에 바꾸는 연산이다. 명시되지 않은 이름들은 변하지 않는다. T1, T2는 테이블명, <attributename>은 속성명, <domainname>은 도메인 이름을 나타낸다고 하자
문법) rename(T1→T2, {<attributename>→<attributename>}*, {<domainname>→<domainname>}*)
예) rename(S1.person→S.peopel,age→myage,name→surname)

(정의) 기호 TR : 데이터베이스 스키마에 대한 테이블 이름 혹은 rename( )을 적용한 테이블의 이름을 가리키는 기호
문법) TR : <schema>.<tablename> 혹은 <schema>.rename()
예)S1.person : 스키마 S1에서 person 테이블
S1.rename(S1.person→S.peopel, age→myage,name→surname)

(정의)copy 연산 : 1개 이상의 인자를 가지며 원본 테이블을 통합 결과 테이블에 복사한다.
문법) copy(TR→T, <-{attributename}>*)
예) copy(S1.book → S.books, -ISBN)

(정의) concatenate 연산 : 1개 이상의 인자를 가지고 두 개의 테이블을 합치는 연산
문법) concatenate(TR1, TR2, S, <-{attributename}>*)
예) concatenate(S1.book+S2.rename(S2.books→S.book,
auther→writer, bookname→title)→S.allbooks)

(정의) change 연산 : 통합된 테이블에 속성의 기본키 및 외래키에 대한 속성을 지정한다. PK 인자는 기본키를 변경하며,

FK 인자는 외래키를 변경한다.
문법) change(FK or PK, T1.attributename, T2.attributename)
예) change(FK, S.Car.ownerid, S.Person.pin)

Table 1 Schema Integration Operation

사용용도	함수명
이름 변경	rename(T1→T2, {<attributename> → <attributename>}*, {<domainname> → <domainname>}*)
테이블 복사	copy(TR → T, <-{attributename}>*)
테이블 합치기	concatenate(TR1, TR2, S, <-{attributename}>*)
키 속성 변경	change(FK or PK, T1.attributename, T2.attributename)

3.3 개체충돌 식별방법

1) 형태소 분석과 시소러스 사전의 구축

DB 내에 데이터 항목명칭의 용어는 단일어인 경우와 복합어인 경우로 구분할 수 있다. 복합어는 "수식어+주요어+구분어"의 순서로 용어가 배열될 때 데이터 항목을 명확히 정의할 수 있는 표준적인 어순이 되며, 용어들은 이러한 형식으로 구분될 수 있다. 본 연구에서는 유사성 식별시 효율을 도모하기 위한 방안으로 형태소 분석에 의거한 시소러스 구축과 유사도 측정 방안을 활용한다.

2) 속성(Attribute) 간의 유사도 정의

시소러스 사전에 등록된 유사성 정보를 토대로 속성간의 의미관계를 다음과 같이 정량적으로 표현할 수 있다. 여기서 dic을 속성 간 의미관계를 나타내는 변수로 정의한다.

dic = 1 : 속성명을 구성하는 구분어와 주요어가 각각동일 혹은 동의어 관계일 경우

dic = 0.75 : 속성명을 구성하는 주요어가 유사어이며, 그리고 구분어가 동일 혹은 동의어 관계일 경우

dic = 0.25 : 속성명을 구성하는 구분어가 동일 혹은 유사어이며, 그리고 그 외의 용어는 동일하거나, 동의어/유사어의 관계가 아닐 경우

dic = 0 : 상기 이외의 경우

속성간 의미관계 변수 dic가 정의된 후, 속성간 유사도 변수 Sim_{atr}은 다음과 같이 정의된다.

$$Sim_{atr} = \begin{cases} 1 & (dic = 1) \\ dic \times w_d + type \times w_t + length \times w_l & (0 \le dic < 1) \end{cases} \quad (1)$$

단, $type$: 각 속성에 해당되는 데이터형의 일치정도
($type$ = 0 or 0.5 or 1)
$length$: 각 속성에 정의된 데이터 길이의 일치 정도
($length$ - 0 or 1)
w_d, w_t, w_l : 각각 dic, $type$, $length$의 가중치
$w_d + w_t + w_l = 1$

여기서, $type$은 각 속성에서 정의된 데이터 형의 일치정도를 나타내는 변수이다. 본 연구에서는 데이터형의 완전히 일치하는 경우는 1로 두고, 대분류 카테고리가 같은 경우 0.5, 대분류 카테고리조차 일치하지 않는 경우 0으로 일치 정도를 구분하였다. $length$의 경우도 일치하는 경우와 일치하지 않는 경우로 구분하여 각각 1, 0의 값을 가지게 하였다.

3) 개체(Entity)간 유사도 정의

각 개체명의 일치성을 나타내는 변수 $name$은 다음과 같이 정의된다.

$name$ = 1 : 개체명이 일치하거나 동의어인 경우
$name$ = 0 : 그 외의 경우

개체명의 경우도 속성명의 의미관계를 나타내는 dic과 같이, [0, 1]사이의 다양한 값으로 나타낼 수 있다. 각 개체와 속하는 속성들이 지닌 유사도와의 관련성이 중요하며, 이러한 관련성은 최종적으로 Name Conflict를 판정하는 수단이 된다.

본 연구에서는 각 개체의 공통속성 비율을 판정하기 위한 변수로 식 (1)을 활용한다. 즉, 본 연구에서의 공통속성은 각 개체에 속하는 속성들 사이에서 동일하거나 유사한 속성집합을 의미하며, 그 값은 Sim_{atr}을 통해 구해진다. 한편, Sim_{atr}은 유사한 정도에 따라 [0,1]의 값으로 표현되므로, 이 중에서 유사하다고 판단되는 속성들을 추출하여 공통속성집합을 구성하기 위해서는 어떤 기준이 필요하다. 본 연구에서는 편의상 유사성 임계값을 0.5로 두며, 이는 DB 통합 시 상황에 따라 DB 설계자가 임의로 조절할 수 있다. 다만, 이러한 임계값이 낮아질수록 공통속성의 비율이 그 만큼 많아지게 되므로 유의할 필요가 있다.

공통속성 비율 com_{atr}은 다음과 같이 정의된다.

$$com_{atr} = \frac{2\sum_{(x,y)} Sim_{atr}(x,y)}{n(A)+n(B)} \quad (\text{if } Sim_{atr}(x,y) \ge 0.5) \quad (2)$$

단, $n(A)$: A라는 개체의 속성 수
$n(B)$: B라는 개체의 속성 수
x : 개체 A의 속성
y : 개체 B의 속성
$Sim_{atr}(x,y)$: 속성 x와 y의 유사도

최종적인 개체간 유사도 Sim_{ent}는 다음과 같이 정의된다.

$$Sim_{ent} = \frac{name \times w_n + Com_{atr} \times w_c}{w_n + w_c} \quad (3)$$

단, w_n, w_c : 각각 $name$, com_{atr}의 가중치

$w_n + w_c = 1$

개체간 유사도 Sim_{ent}는 식(1)과 유사한 형태를 보이고 있으며, 가중치의 존재 이유와 부여방법 역시 앞서 설명한 것과 동일하다. 결과적으로 최종적인 Name Conflict의 식별은 식(3)을 통해서 판별되나, 식(3)을 구하기 위해서는 앞서 언급된 모든 계산과정을 거쳐야만 한다.

Name Conflict 식별을 위한 전체적인 산출과정을 정리하면 다음 그림과 같다.

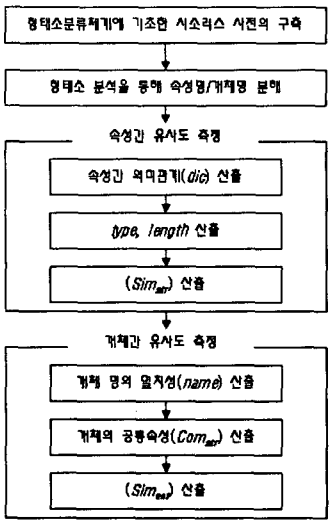


Fig. 3 Flowchart of Quantitative Methodology

4. 통합 사례

4.1 예제 데이터베이스 스키마

본 연구에서는 두 개의 스키마 예제를 통해 충돌개체식별 알고리즘을 이용하여 개체 간 유사도를 측정한 후 XML Schema를 이용하여 통합하고자 한다. 단, 본 예제에 해당하는 시소러스 사전은 다음과 같이 사전에 마련되어 있는 것으로 가정한다.

Table 2 Thesaurus for Examples

구분	동의어	유사어	용어종별
부문	부서,부,계열	과,계,지역	수주
코드	번호,기호,넘버	암호,국번지역	수주구
명	명칭,이름,성명	별명,약칭,상호	수주구
주소	번지,현주소,근무처	소재지,장소	수주구
전화	TEL,폰,다이얼	내선,휴대폰	수주
종업원	사원,직원,피고용인	사무원,점원	수주
FAX	팩시밀리		수주
우편	편지,메일	소포,택배	수주
번호	코드,넘버,#,No	국번	수주구
거래처	고객	업체,협력업체	수주
제품	상품		수주

다음은 통합을 목적으로 하는 두 스키마이다.

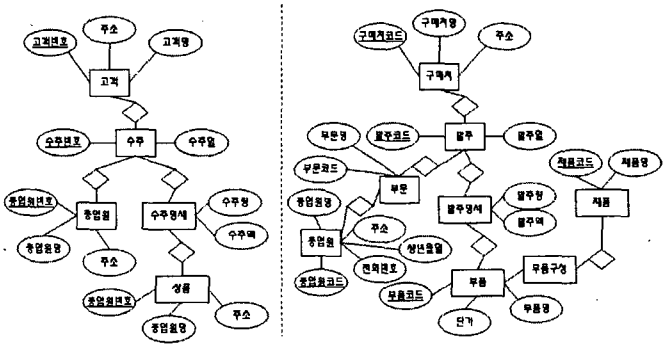


Fig. 4 Schema 1 and Schema 2

4.2 충돌개체 식별

유사도 측정과정을 토대로 최종적인 Sim_{ent}가 구해지면, 이러한 개체간 유사도를 토대로 군집화한다. Sim_{ent}은 유사성이 높을수록 1에 가까운 값을 가지게 된다. 본 연구에서는 아래 그림과 같이 전체 개체 수에 대한 Sim_{ent} 값에 따라 군집분석을 행하여 각 스키마 사이에 충돌우려 개체를 파악한다.

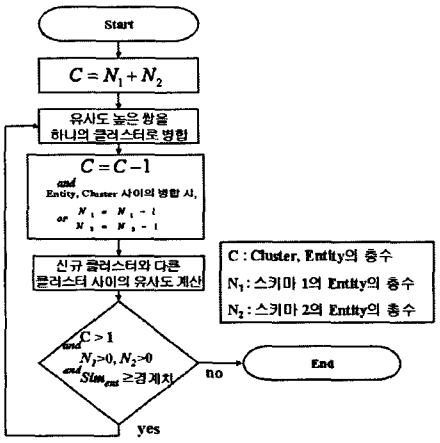
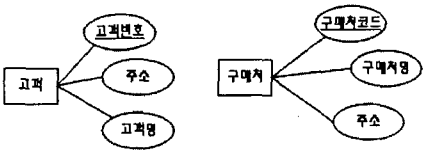


Fig. 5 Flowchart of Clustering

먼저, 두 예제 스키마의 고객 개체와 구매처 개체의 통합을 살펴보면 다음과 같다.



스키마	개체	속성명	데이터형	데이터길이
1	고객	고객번호	수치형	10
		고객명	문자형	40
		주소	문자형	40
2	구매처	구매처코드	문자형	5
		구매처명	문자형	10
		주소	문자형	30

Fig. 6 Conceptual Schema 1 and 2

속성간의 의미관계인 dic 값을 산출하기 위해서는 우선 형태소 분석을 통해 각 속성을 분해해야 한다. 그 결과는 다음표와 같다.

Table 3 Decomposition of Attributes

속성명		수식어	주요어	구분어
고객	고객번호	-	고객	번호
	고객명	-	고객	명
	주소	-	(고객)	주소
구매처	구매처코드	-	구매처	코드
	구매처명	-	구매처	명
	주소	-	(구매처)	주소

Table 4 Calculation of dic, type, and length between attributes

스키마1 / 스키마2	구매처코드			구매처명			주소		
고객번호	d	t	l	d	t	l	d	t	l
	0.75	0	0	0.25	1	0	0	1	0
고객명	d	t	l	d	t	l	d	t	l
	0.25	1	0	0.75	1	1	0	1	0
주소	d	t	l	d	t	l	d	t	l
	0	1	0	0	1	0	1	1	0

(단, d : dic, t : $type$, l : $length$)

Sim_{atr}을 산출하기 위해, 각 변수들을 계산한 결과는 Table 4와 같으며, 그 결과를 토대로 식(1)에 의거 각 스키마의 Sim_{atr}(가중치: w_d=0.7, w_t=0.2, w_l=0.1)을 산출한 결과는 Table 5와 같다.

Table 5 Sim_{atr} between attributes in entities

스키마1 / 스키마2	구매처코드	구매처명	주소
고객번호	0.525	0.235	0.2
고객명	0.375	0.825	0.2
주소	0.2	0.2	0.9

식(2)에 따라 com_{atr}을 구하면 0.75이며 $name$ 값은 0이므로 가중치 w_n=0.2, w_c=0.8으로 할 때 최종적인 유사도를 산출하면 Sim_{ent}=0.6이 된다.

이러한 방식으로 Fig. 5의 충돌 개체 식별과정에 따라 스키마 1의 모든 개체에 대한 스키마 2의 모든 충돌 개체를 측정한 결과는 Table 6과 같다.

Table 6 Calculation of Sim_{atr}

	구매처	발주	직원	부문	발주명세	부품	제품
고객	0.6	0	0.18	0	0	0	0
수주	0	0.72	0	0	0	0	0
종업원	0	0	0.52	0	0	0	0
수주명세	0	0	0	0	0.8	0	0
상품	0	0	0	0	0	0.22	0.576

스키마 1과 스키마 2의 충돌개체 측정을 통해 고객-구매처, 수주-발주, 종업원-직원, 수주명세-발주명세, 상품-제품 간에 충돌이 발생함으로 알 수 있었다.

4.3 스키마 대응 연산

충돌이 발생하는 개체는 다음과 같이 스키마 대응 연산을 통해 XSLT 프로그램으로 변환된다. 스키마 통합 단위 연산은 XSLT 템플릿으로 변환되어 있기 때문에 이 템플릿을 이용하여 XSLT 프로그램을 생성한다.

```
/* 스키마 대응 규칙(Correspondence Assertion) */

T1 = rename(S1.고객 → S.구매처, 고객번호 → 구매처코
     드, 고객명 → 구매처명)
T2 = rename(S2.구매처 → S.구매처)
concatenate(T1, T2, S.구매처)

T1 = rename(S1.수주 → S.발주, 수주번호 → 발주코드,
수주일→발주일)
T2 = rename(S2.발주 → S.발주)
concatenate(T1, T2, S.발주)

T1 = rename(S1.종업원 → S.직원, 종업원번호 → 직원코
드, 종업원명 → 직원명)
T2 = rename(S2.직원 → S.직원)
concatenate(T1, T2, S.직원)

T1 = rename(S1.상품 → S.제품, 상품코드 → 제품코드,
상품명 → 제품명)
T2 = rename(S2.제품 → S.제품)
concatenate(T1, T2, S.제품)
```

4.4 스키마 통합

아래는 스키마 S1과 S2의 통합을 위해 ER 다이어그램을 XML Schema로 변환한 문서이다. 위 스키마 대응 규칙에 따라 XSLT가 생성되는데, XSLT 생성 알고리즘은 다음과 같다.
① DBA가 정의한 스키마간 대응 규칙과 매칭되는 속성들을 변환한다. 첫 번째 스키마에 대하여 테이블의 이름을 변환하는 XSLT 코드, 테이블에 포함된 속성 이름을 변환하는

XSLT 코드, 테이블에 포함된 속성을 결과 스키마에 출력하는 XSLT 코드가 생성된다.

② 입력된 스키마에서 스키마간 대응 규칙에 정의되어 있지 않은 속성들을 검사하여 추가시킨다. 두 번째 스키마에만 포함된 속성을 결과 스키마에 출력하는 XSLT 코드가 생성된다.

```
<?xml version"1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="스키마1" type="스키마1루트타입"/>
  <xs:complexType name="스키마1루트타입">
    <xs:sequence>
      <xs:element name="고객" type="고객타입"
        minOccurs=0" maxOccurs="unbounded">
    <xs:sequence>
  </xs:complexType>
<xs:complexType name="고객타입">
  <xs:sequence>
    <xs:element name="고객번호" type="수치형타입">
    <xs:element name="고객명" type="문자형타입">
    <xs:element name="주소" type="문자형타입">
    <xs:element name="수주개체" type="수주개체타입"
minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="수주개체타입">
  <xs:sequence>
    <xs:element name="수주번호" type="수치타입">
    <xs:element name="수주일" type="날자타입">
    <xs:element name="종업원개체" type="종업원개체타입"
minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="수주명세개체" type="수주명세개체타입"
minOccurs="1" maxOccurs="unbounded"/>
  </xssequence>
</xscomplexType>
.....(생략)
  <xs:element name="스키마1" type="스키마2루트타입"/>
  <xs:complexType name="스키마2루트타입">
    <xs:sequence>
      <xs:element name="구매처" type="구매처타입"
        minOccurs=0" maxOccurs="unbounded">
    <xs:sequence>
  </xs:complexType>
<xs:complexType name="구매처타입">
  <xs:sequence>
    <xs:element name="구매처코드" type="수치형타입">
    <xs:element name="구매처명" type="문자형타입">
    <xs:element name="주소" type="문자형타입">
    <xs:element name="발주개체" type="발주개체타입"
minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="발주개체타입">
  <xs:sequence>
    <xs:element name="발주코드" type="문자형타입">
    <xs:element name="발주일" type="날자타입">
    <xs:element name="부문개체" type="부문개체타입"
minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="발주명세개체" type="발주명세개체타입"
minOccurs="1" maxOccurs="unbounded"/>
  </xssequence>
</xscomplexType>
.....(생략)
```

다음은 스키마 통합 연산을 변환하여 생성된 XSLT 코드이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xs="http://www.w3.org/1999/XSL/Format">
<xsl:output indent="yes"/><xsl:strip-space eleemnts="*"/>
<xsl:template match="/">
  <xs:schema>
    <xs:element name="IS">
      <xs:complexType>
       <xs:sequence>
        <xsl:apply-templates select="/xs:schema/
xs:element/xs:complexType/xs:sequence/xs:element"/>
       </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xsl:apply-templates select="/xs:schema/
xs:simpleType"/>
    </xs:schema>
</xsl:template>
<!-- 테이블 명에 대한 rename 처리 -->
<!-- 1. rename(S1.고객 → S.구매처, ...) -->
<!-- 2. rename(S2.구매처 → S.구매처, ...) -->
<xsl:template match="/xs:schema/xs:element/
xs:complexType/xs:sequence/xs:element">
<xsl:if test="@name='고객' or @name='구매처'">
  <xsl:if test="ancestor::xs:elelment[@name='S1']">
    <xsl:element name="xs:element">
      <xsl:attribute name="xs:element">
      <xsl:attribute name="name">구매처</xsl:attribute>
      <xsl:attriute name="maxOccurs">unbounded
</xsl:attribute>
      </xsl:apply-templates/>
      </xsl:element>
    </xsl:if>
  </xsl:template>
  .....(생략)
```

위 XSLT 코드를 이용하여 이질의 예제 데이터베이스를 하나의 데이터베이스로 통합할 수 있었다. 전체적인 ER 개념도는 아래 그림과 같다.

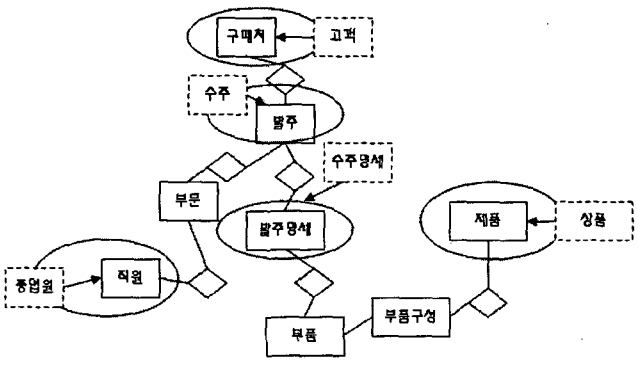


Fig. 7 The Integration Result

5. 결 론

본 연구는 현재 물류분야에 있어, 물류정보시스템의 통합화의 중요한 과제로서 빈번히 지적되고 있는 DB 통합의 문제를 연구의 대상으로 하였다. XML의 표준 중 데이터베이스 스키마 표현에 적합한 XML Schema를 이용하여 이질의 데이터베이스 스키마를 통합하는 방법을 제시하였고, DB 통합과 관련하여 합리적이고 효율적인 통합을 저해하는 대표적인 요인으로 알려져 있는 Name Conflict 문제에 주목하여 그것을 효율적으로 식별할 수 있는 계량적인 방안을 사용하였다.

표준으로 인정된 XML을 공통 데이터 모델로 사용함으로써 스키마 통합을 표준화 할 수 있고, 스키마의 통합뿐 아니라 데이터의 통합에도 같은 모델을 사용하여 데이터의 변환의 표준화와 확장성을 얻을 수 있으며, 또 일관된 데이터베이스 스키마를 표현함으로서 서로 다른 데이터모델의 스키마 통합 뿐 아니라 XML 스키마로 기술된 XML Schema 파일의 통합에도 유용하게 사용될 수 있다.

충돌개체 검사 시 사용한 유사도 측정 방법은 DB통합의 여러 가지 과제들 중 한부분에 해당하는 것으로써, 실제 DB통합문제를 실무적인 차원에서 효율적으로 처리하기 위해서는 대규모 시소러스 사전의 구축, 구조적인 측면의 충돌문제(Structural Conflict), 실질적인 문제에 대한 스키마통합 등 향후 과제로 남아 있다. 적용한 방법론에 있어 임계값 및 유사도의 높고 낮음의 문제는 다소 통합설계자의 주관적 판단과 식견을 필요로 하므로 DB 통합의 편의성과 효율성을 더욱 높이기 위해서는 이러한 부분까지 기계적으로 해석할 수 있는 계량적 방안의 수립이 요구된다. 따라서, 상기의 여러 가지 문제들이 향후 추진되어야 과제로 남아있다.

참 고 문 헌

[1] Litwin W., Mark L., Roussopoulos N., "Interoperability of Multiple Autonomous Databases," ACM Computing Servey 22, 3, Sept. 1990, pp267-193.

[2] Soon M. Chung and Pyeong S. Mah, "Schema Integration for Multidatabases Using the Unified Relational and Object-Oriented Model", ACM, 1995, pp208-25.

[3] J.A. Larson, S.B. Navathe, and R. Elmasri. "A theory of attribute equivalence in database with application to schema integration". IEEE Transactions on Software Engineering, pages 449-463, 1989.

[4] Spaccapietra S., Parent C., Dupont Y., "Model-Independent Assertions for Integration of Heterogeneous Schemas", Very Large Data Bases Journal, Vol.1, No.1, July 1992.

[5] Batini C., Ceri S., and Navathe S.B.(1997), "Conceptual Database Design: An Entity-Relationship Approach", Benjamin/Cummings Publishing Company Inc.

[6] Batini, C., Lenzerini, M., Navathe, S.B.(1986), "A Comparative Analysis of Methodologies for Database S초듬 Integration", ACM Computing Surveys, Vol. 18, No. 4, pp.650-663.

[7] Fong J., Karlapalem K., Li A., and Kwan I.(1999), "Methodology of Schema Integration for New Database Application: A Practitioner's Approach", Journal of Database Management, Vol. 10. No. 1, pp3-18.

[8] Larson, J.A., Navathe S.B., and Elmasri R.(1989), A Theory of Attribute Equivalence in Databases with Application to Schema Integration", IEEE transaction on Software Engineering, Vol. 15, No.4, pp449-463

[9] Song W.W., Johannesson P., and Buenko J.A.(1996), "Semantic Similarity Relations and Computation in Schema Integration", Data & Knowledge Engineering, 19, pp65-97

[10] Tseng, F.S.C, Chiang J.J., Yang W.P.(1998), "Integration of Relations with Conflicting Schema Structures in Heterogeneous Database Systems", Data & Knowledge Engineering, 27, pp231-248