ODMG2.0을 토대로 한 다중데이터베이스 스키마 통합에 관한 연구

김경수·주경수
순천향대학교 공과대학 전산학과

A Study on Schema Integration for Multidatabase based on ODMG2.0

Kyung soo Kim, Gyung soo Joo
Dept. of Computer Science, College of Engineering
SoonChunHyung Univ.

요 약

멀티데이터베이스는 기존에 존재하는 데이터베이스와 파일 시스템들의 최상위에 존재하는 데이터베이스 시스템으로서 사용자에게는 단일 데이터베이스의 모습으로 투명하게 보여주고자 하며, 반면에 사용자 데이터는 로컬 데이터베이스가 관리한다. 통합 스키마는 각각의 로컬 데이터베이스의 스키마 합병으로 구성되며 합병은 각각의 로컬 데이터베이스의 로컬 스키마 충돌을 중재하며 이루어진다. 한편 멀티데이터베이스 구조에서 각각의 로컬 데이터베이스의 데이터를 통합할 때에 각각의 스키마 정보에 대한 유사성이나 이질성 때문에 스키마 충돌이 발생한다. 본 연구에서는 OMG에서 제시한 ODBMS의 표준인 ODMG를 기반으로 멀티데이터베이스 시스템의 스키마 통합에 대한 방안을 제시했다.

1. 서 론

개인용 컴퓨터 및 네트워크 기술의 급속한 향상, 시스템의 개방화 추세는 기업의 컴퓨팅 환경을 분산 처리 환경으로 변화시키고 있다. 이러한 환경에서는 데이터베이스의 분산, 어플리케이션의 분산 등을 통합하고 서로 다른 이기종 하드웨어 및 네트워크 프로토콜 사이의 인터페이스 등이 기업의 정보전략 시스템 구축에 가장 큰 문제이며, 극복해야 할 과제이다. 아울러 이미 존재하는 자치적인 데이터베이스 통합에 대한 필요성이 오늘날 모든 큰 기업에서 절실히 느껴지고 있다.

하지만 이질적인 데이터베이스 시스템을 통합하고자 몇 차례 시도에서 얻은 결과는 스키마 통합 과정이 전적으로 분산되어 있고, 이질적인 시스템들의 실현에 가장 어려운 점으로 되어있다.

스키마 속성이 수백 또는 수천이 되는 것이 일반적이다. 그러한 많은 데이터베이스들을 통합하는 것은 힘든 작업이지만 매우 필요한 일이다. 따라서 스키마 통합을 위해 데이터베이스 관리자에게 도움을 주기 위한 기술과 툴들은 분산 데이터베이스 구현에 의해서 활발히 연구되고 있다. 하지만 지금까지 스키마 통합부분의 연구는 아직 방법론이나 툴에서 확실치 않다.

본 연구를 위해 먼저 ODMG2.0의 관련 기술을 알아본다. 제 3장에서는 멀티데이터베이스에 대해 고찰해 보았으며, 4장에서는 스키마 통합에 대한 기술을 기술했으며, 마지막으로 결론을 맺는다.

2. ODMG 2.0

ODMG[1]에서 추구하는 표준은 디자인 이식성을 강조하고 있다. 디자인 이식성은 객체를 기본으로 삼는다. 실세계를 모델링하는 데에 있어서 어떤 대상을 객체로 삼고서, 그 객체의 행동과 상태를 함께 정의함으로써 객체에 대한 정의가 이루어진다. 객체의 행동적 특성을 나타내는 부분들은 조작(operations)들로 표현이 된다. 또한 객체의 대표적인 상태 정보들은 속성(property)로써 표현을 하고 있다. 이런 식으로 조작과 속성을 분리함으로써, 조작을 통하여서만 외부와 교신이 가능하고, 객체 내부의 상태가 변경할 수 있게 되어 자료추상화와 정보은닉이 실현된다.

이렇게 정의된 객체들은 다시 형(type)으로 묶어질 수 있고, 형은 하나의 인터페이스와 하나이상의 구현(implementation)을 가질 수 있다. 이러한 객체들의 추상적인 묶음인 타입도 그 자체로서 객체가 된다.

```
interface Section
 {
  type properties:
   supertype: Atomic Object
   key: (course, section_number)
  instance properties:
   section_number: String
   day_offered: set<weekday>
   time_offered: Struct<from:Time,
                        to Time>
  instance operations:
       cancel()
        reschdule(to: Time)
 }
```

위 프로그램을 보면 Section이라는 형이 정의되어 있다. 형 정의를 위해서 ODMG 에서는 interface라는 keyword로서 타입 선언을 시작하고 있다. 이것은 아까 설명했던 타입의 인터페이스를 좀더 넓게 속성과 조작을 다 포함한 용어로 사용하고 있으니, 구별이 필요하다. 기본적으로 ODMG에서 정의하는 타입은 interface라는 keyword로 시작하면서, 그 주요 내용은 슈퍼타입과 인덱스처럼 쓰이는 Key, 그리고 속성과 조작을 포함한다. 여기서 슈퍼타입은 Atomic-Object이며, time-offered 등 세 개를 정의하고, 조작은 cancel(), reschedule()를 정의하고 있다. 기본적으로 ODMG에서 정의하는 기본타입은 외연을 표시할 수 있는 객체(denotable objects)와 특징요소(characteristics)로 나누어질 수 있고, 특징요소들은 속성과 조작들로 나뉘어진다. 타입 중에서 추상 타입(abstract type)이 있는데, 이것은 순수하게 다른 서브타입들의 계승을 위해 존재하는 타입이며, 자신의 인스턴스를 가질 수 없는 타입을 말한다. 그리고 응용 프로그램을 위한 타입들을 정의할 때에는 반드시 Object 타입의 하위타입으로 정의를 해주어야 한다. 다중상속시 발생하는 중복이름 문제는 문제를 일으킬 만한 이름을 정확하게 상속클래스를 지정하든가, 아니면 이름을 바꾸는 방법으로 해결할 수 있다. 어떤 타입의 extent는 그 타입의 인스턴스들의 집합을 말한다. extent를 선언함으로써 ODMG에게 자동으로 그 모든 인스턴스들의 집합을 관리하도록 한다

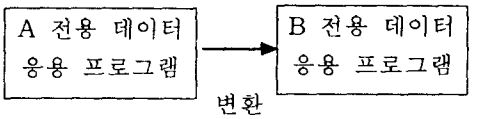
는 것을 의미한다. 타입은 내포적인 의미가 있어서, 만약 A 타입이 B 타입의 서브 타입이라면, A의 extents는 B의 extents의 부분집합이 된다. 하나의 타입은 하나 이상의 구현을 가질 수 있다. 구현은 표현과 방법들의 집합으로 구성된다. 표현은 그 타입의 자료구조에 해당하며, 방법들은 함수들의 몸체들을 말한다. 왜 이렇게 구현을 여러 가지로 하는 것을 허용하는가? 나름대로 장점이 있어서인데, 먼저 네트워크 상에서 상이한 기계들간의 데이터베이스 접근을 지원할 수 있으며, 둘째, 여러 개의 언어나 컴파일러가 혼재하고 있는 환경을 지원할 수 있고, 셋째, 사용자가 여러 개의 선택사항들 중에 장단점을 따져서 서로 다른 구현을 가능하게 하고, 그 중에서 택일 할 수 있도록 한다. 특별히 ODMG에서 클래스라 하면, 타입의 인터페이스 정의 부분과 여러 개의 구현들 중에서 하나를 묶어서 클래스라고 부른다. Denotable Object는 크게 객체와 리터럴(literal)로 나눌 수 있다. 우선, Denotable Object의 특징은 자신을 구별할 수 있는 identity가 있는 것이다. 이 identity는 리터럴일 경우에는 자신의 값이 identity가 되고, object일 때에는 object identifier가 별도로 있다.

3. 멀티데이터베이스

일반적인 업무에서의 데이터베이스 관련 응용 프로그램은 파일 시스템을 사용하는 방법과 관계형 데이터베이스 및 객체지향 데이터베이스 등의 데이터베이스 프로그램을 이용하여 설계, 관리하는 방법이 있다. 근래에 들어서는 데이터베이스 응용 범위가 넓어지면서 이질 데이터 소스를 접근하는 요구가 발생하게 되는데 이러한 방법은 데이터 변환을 이용하여 사용하기도 하며 게이트웨이를 이용하여 사용하기도 한다.

3.1 데이터 변환 방법

데이터 변환을 이용하여 이질 소스를 접근하는 방법을 제공하는 [그림1]와 같이 이전에 사용하던 데이터를 새로운 환경에서 사용할 수 있도록 변환하는 방법이다. 변환의 대상은 데이터 뿐 만 아니라 응용 프로그램까지 모두 변환하기도 하며 데이터 변환 후에는 기존의 응용 프로그램과 같이 작동 할 수 있는 에뮬레이터를 작성하여야 한다.

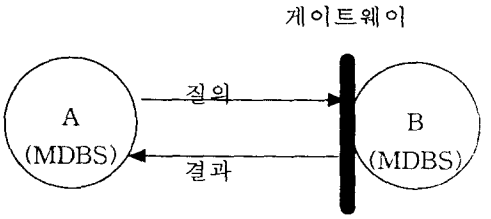


[그림1] 데이터 변환 방법

그러나 데이터 변환 방법은 데이터의 변환 비용이 오버헤드가 될 뿐만 아니라 응용 프로그램까지 바꿔주어야 하므로 이질 데이터베이스에서 사용하는 데에는 많은 문제점을 안고 있다.

3.2 게이트웨이 방법

게이트웨이를 사용하는 방법은 질의 변환 방법으로 단순히 접근하고자 하는 데이터베이스의 언어로 원래의 질의를 변환하여 수행하는 방법이다. [그림2]의 모습과 같이 멀티데이터베이스인 A에서 로컬 데이터베이스인 B로 접근하고자 할 때 게이트웨이를 거치게 되는데 게이트웨이에서 하는 역할은 A의 질의를 B의 질의로 변환하는 것이다. 게이트웨이를 통하여 B의 질의로 변환된 후에는 B의 로컬 데이터베이스에서 수행을 하게 되고 그 결과는 A에게 전달되게 된다.



[그림2] 게이트웨이 구성 방식

게이트웨이를 사용할 때 발생할 수 있는 문제점은 다음과 같다.

첫째, 트랜잭션 관리 기능이 없다. 따라서

동시 수행 제어나 회복 기능을 자유로이 사용할 수 없다.

둘째, 단순히 질의 변환만 하기 때문에 두 데이터베이스가 스키마의 구조적인 면에서의 유사성이나 스키마간의 차이를 표현하는 방법을 제공하지 못한다는 것이다. 즉 두 데이터베이스의 스키마에는 이름은 다르지만 의미적으로는 같은 데이터가 있을 수 있으며 그 반대의 경우도 발생할 수 있으나 단순한 질의 변환을 이용한 접근 방법에서는 그와 같은 유사성 또는 이질성을 판단할 수 없게 된다.

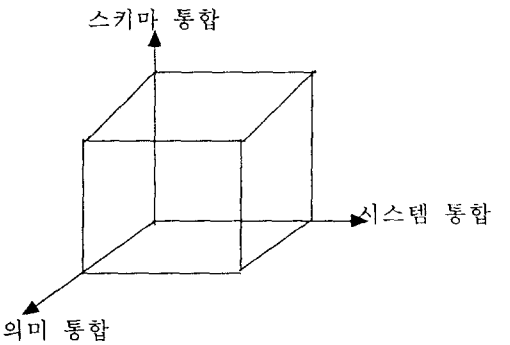
3.3 멀티데이터베이스

데이터 변환이나 게이트웨이의 문제점을 해결하고 이질 데이터베이스에 대한 접근 방법을 제공하는 것이 멀티데이터베이스[2]이다. 멀티데이터베이스는 기존에 존재하는 데이터베이스와 파일 시스템들의 최상위에 존재하는 데이터베이스 시스템으로서 사용자에게는 단일 데이터베이스의 모습으로 보여주는 완전한 데이터베이스의 기능을 갖춘 DBMS이다.

멀티데이터베이스는 통합 하고자 하는 데이터베이스의 통합된 총체적인 스키마만을 관리하며, 모든 사용자 데이터는 로컬 데이터베이스가 관리한다. 통합 스키마는 각각의 로컬 데이터베이스의 스키마의 합병으로 구성되며 합병은 각각의 로컬 데이터베이스의 로컬 스키마 충돌을 중재하며 이루어진다. 로컬 데이터베이스의 트랜잭션은 실제로 데이터를 관리하고 있는 로컬 데이터베이스에 의하여 조정되며, 멀티데이터베이스 시스템의 구성은 그 하부구조로 통합 가능한 여러 개의 로컬 데이터베이스를 가지고 있으며 멀티데이터베이스와 로컬 데이터베이스는 각각의 게이트웨이를 통하여 데이터를 전달한다.

4. 스키마 통합

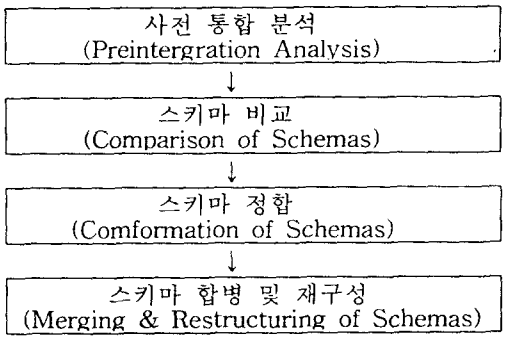
멀티데이터베이스 접근에서 데이터베이스 통합은 시스템 통합, 스키마 통합, 시멘틱 통합과 같이 3차원으로 볼 수 있다[그림3]. 시스템 통합은 사용자 투명성과 시스템 투명성 모두를 제공한다. 데이터베이스들이 분산되어 있는 것과 같이, 멀티데이터베이스 시스템은 사용자가 모든 원격 데이터베이스에 접근하기 위한 통신과 허가를 쉽게 해 주어야 한다. 스키마 통합은 멀티데이터베이스의 정형화된 전체 개념을 제공하며, 시멘틱 통합은 콤포넌트 데이터베이스들에 존재할지 모르는 데이터 충돌을 해결한다.



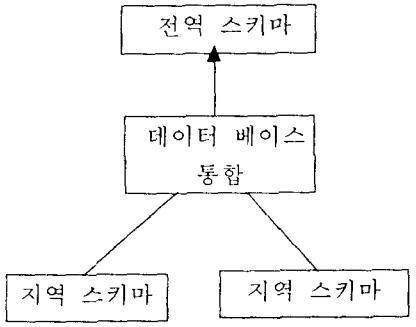
[그림3] 데이터베이스 통합의 3차원

데이터베이스 스키마 통합은 유사한 응용문제에 대하여 서로 다른 사용자가 생성한 스키마이거나, 새로 생성한 스키마들을 전역적이고 통합된 스키마로 통합하는 행위이다[3]. 스키마 통합은 일반적으로 [그림4]와 같은 Batini의 4단계 통합과정을 기본으로 하고 있으며, 데이터베이스 설계 단계에서 서로 다른 사용자에 의해 제안된 사용자 뷰를 전역 개념 스키마로 통합하는 뷰 통합과 분산 데이터베이스 환경에서 각 지역 데이터베이스 스키마를 전역 가상 스키마로 통합하는 데이터베이스 통합[그림5]의 두 분야로 연구가 진행되고 있다[3,4,5,6,7,8].

멀티데이터베이스 구조에서 각각의 로컬 데이터베이스의 데이터를 통합할 때에 각각의 스키마 정보에 대한 유사성이나 이질성 때문에 스키마 충돌이 발생한다. [표1]은 관계형 데이터 모델을 지원하는 데이터베이스



[그림4] 스키마 통합 단계



[그림5] 데이터베이스 통합

시스템과 객체지향 데이터 모델을 지원하는 객체지향 데이터베이스 시스템간의 통합에 대한 분류[9]에 객체 지향 개념인 일반화, 집단화, 메소드가 추가되었을 경우에 발생할 수 있는 스키마 충돌을 추가한 것이다[10].

멀티데이터베이스의 충돌은 개체 충돌, 스키마 충돌, 의미 충돌, 데이터 충돌이 있다. 멀티데이터베이스에서는 위와 같은 스키마 충돌을 데이터베이스 언어로서 해결한다. 즉 멀티데이터베이스용 데이터 언어로서 각각의 로컬 데이터베이스에 저장되어 있는 로컬 스키마에 대한 통합을 할 수 있도록 하며, 통합된 새로운 스키마인 글로벌 스키마를 정의할 수 있다. UniSQL/M[11]은 대표적인 멀티데이터베이스로서 [표2]와 같은 형식의 통합을 위한 데이터베이스 언어를 제공하고 있다.

1. Entity-vs-Entity
a. One-to-One Entity
i. Entity Name
· different names for equivalent entity
· same name for different entities
ii. Entity Structure
· missing attributes
· missing but implicit attributes
iii. Entity Constraints
iv. Entity Inclusion
b. Many-to Many Entities
2. Attribute Name
a. One-to-One Attribute
i. Attribute Name
· different names for equivalent attribute
· same name for different attributes
ii. Attribute Constraints
· integrity constraints
· data type
· composition
iii. Default Values
iv. Attribute inclusion
v. Methods
b. Many-vs-Many Attributes
3. Entity-vs-Attributes
4. Different Representation for Equivalent Data
a. Different Expression denoting same information
b. Different Units
c. Different Levels of Precision

[표1] 멀티데이터베이스에서의 스키마 충돌

UniSQL/M의 언어는 글로벌 스키마의 가상 클래스와 로컬 스키마의 명세로 구분되며 가상 클래스의 시그니쳐에는 글로벌 스키마의 속성을 명세하며, 로컬 스키마 명세에는 통합하고자 하는 각각의 로컬 데이터베이스

```
CREATE VCLASS virtual_class name
SIGNATURE attr_def_list
AS SELECT selection_list
FROM entity_spec_list
WHERE search_conditions,
..

..
SELECT selection_list
FROM entity_spec_list
WHERE search_conditions

   entity_spec_list ::= cdb_entity_name
   [variable]
            {,cdb_enrity_name[variable]}
cdb_entity_name ::= [cdb_name]
                              entity_name
```

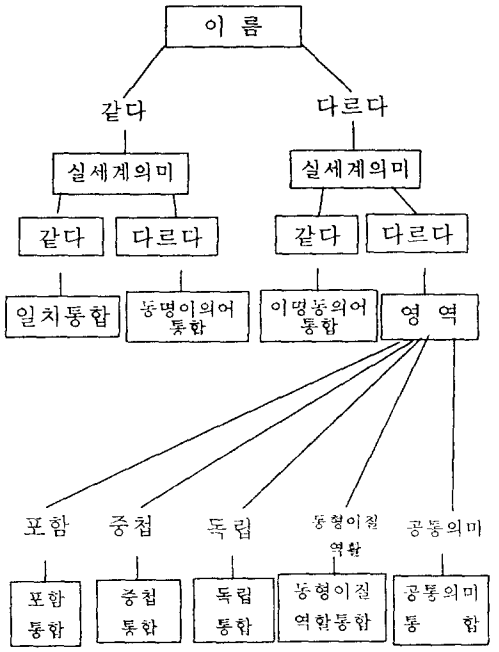
[표2] 데이터베이스 언어

의 내용을 기록한다.

멀티데이터베이스 언어는 사용자로 하여금 스키마 충돌이 발생하였을 경우 사용자가 언어로 충돌을 해결할 수 있는 메카니즘을 제공하며, 통합 시 발생하는 충돌의 해결은 사용자가 언어로서 충돌을 피할 수 있도록 표현함으로써 해결된다. 따라서 멀티데이터베이스 언어를 사용하여 사용자는 로컬 데이터베이스의 내용 중에서 원하는 자료를 추출하여 글로벌 스키마로 통합한다.

객체 지향 모델은 실세계 객체들이 갖는 공통적인 특성을 정의한 클래스를 기본으로 한 데이터 모델링 기법이다. 클래스 구조는 공통적인 특성을 갖는 유사 객체의 속성과 행위를 캡슐화한 추상화 단위이다. 객체는 객체 클래스의 인스턴스로서 객체의 상태는 속성으로 표현되며, 속성은 원시 데이터형이거나 또 다른 객체일 수 있다. 객체의 행위는 메소드라고 하며 속성을 조작하는 루틴들이다. 객체지향 모델은 상위 클래스와 하위 클래스로 구성되는 클래스 계층 구조를 갖게 되며, 이 구조에서 하위 클래스는 상위 클래스의 속성과 행위를 상속받거나 자신의 고유한 속성과 행위를 가질 수 있다[12].

객체 지향 모델에서 가장 기본적인 추상화 단계는 객체 클래스이다. 클래스는 실세계의



[그림6] 객체 클래스의 통합 분류

모델과 데이터베이스 설계자의 논리적 인식을 가장 자연스럽게 반영한다. 객체 클래스의 통합은 객체의 이름, 실세계 의미 및 객체 인스턴스의 영역을 비교함으로서 수행하며, 통합이 가능한 경우의 정의는 [그림6]과 같다[13].

5. 결 론

90년대에 들어와서부터 분산 DBMS는 네트워크의 각 사이트에서 서로 다른 사용자에 의해 독립적으로 설계, 관리, 유지보수 되고 있는 지역 스키마들을 통합하여 전역 스키마를 제공하며, 특정 사이트의 사용자가 다른 사이트의 지역 데이터베이스를 투명하게 이용할 수 있는 환경을 지원하고, 각 지역 스키마에 부여된 스키마 구성 개체들이 통합된 전역 스키마에서도 유지되도록 해야한다. 최근까지 다양한 형태의 분산 DBMS에 대한

스키마 통합의 연구가 활발히 진행되어 왔다.

멀티데이터베이스 스키마 모델로서 EER 및 객체지향 모델 또는 객체지향 모델을 확장하여 사용하였으며, 이를 이용한 통합 연산자를 정의한다. 통합 방법은 데이터베이스 스키마의 기본 구성요소이자 실세계 응용영역의 기본적인 추상화 단위인 객체 클래스와 이들 사이의 관계에 대한 통합을 중심으로 연구되어졌다.

본 연구에서는 멀티데이터베이스 스키마 모델로서 ODMG2.0을 이용하여 객체 클래스 통합과 관계 통합에 대한 방안을 제시한다.

참 고 문 헌

(1) R.G.G Cattell, Douglas K. Batty "The Object Database Standard: ODMG2.0" Morgan Kaufmann Publishers, Inc.

(2) S. Navathe, R. Elmasri and J, Larson, "Integrating User Views in Database Design", IEEE COMPUTER Magazine, pp.50-62, 1986.

(3) C.Batini, M. Lenzerini and S.B.Navathe, "A Comparative Analysis of Methodolgies for Database Schema Integration", ACM Computing Surveys, Vol.18, No.4, pp.323-364, Dec.1986

(4) C. Batini, S.Ceri,S,B. Navathe, Conceptual Database Design, The Benjamin/Cummings Publishing Company, pp.119-137, 1992.

(5) W. Kim, Modern Database Systems,Addison-Wesley Publishing Company, pp.-521-550, 1995.

(6) J. Larson, S. Navathe and R. Elmasri, "A Theory of Attribute Equivalence in Databases with Application to Schema Integration", IEEE Trans. on Software Engineering, Vol.15, No.4, pp.449-463, Apr.1989.

(7) Amihai Motro, "Superviews : Virtual-Integration of Multiple Database", IEEE Trans on Software Engineering, Vol.13, No.7, pp.785-798, Jul.1987.

(8) Won Kim, Introduction to Part 2: Technology for Interoperating Legacy Databases, Modern Database Systems, 1995, 515-520

(9) Kim, W., and Seo I. 1991. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. IEEE Computer, December.

(10) Won Kim, Injun Choi, Sunit K. Gala, Mark Scheevel, On Resolving Schematic Heterogeneity in Multidatabase Systems. Modern Database Systems, 1995, 521-550

(11) UniSQL/M Manual, UniSQL Inc., 1994

(12) T.J. Teorey, Database Modeling and Design, Morgan Kaufmann Publishers, Inc., pp.46-60, 1990.

(13) 박우근, 노봉남 "보안 분산 객체지향 데이터베이스 스키마의 통합", 한국 정보처리 학회 논문지, 제2권 제5호, pp.645-654