객체지향 데이터베이스에서 다계층 데이터베이스 설계 및 유지

김 남 진[†] · 신 동 천[††]

요 약

오늘날 같이 대량의 데이터를 갖는 데이터베이스에서 원하는 정보를 찾는 작업은 많은 비용과 시간을 요구하게 된다. 따라서 대용량의 데이터를 수용하고 있는 데이터베이스에서 원하는 정보를 효과적으로 찾기 위한 기술이 필요하다. 지식 추출 도구 중에서 AOG(attribute-oriented generalization) 기법을 기반으로 하는 다계층 데이터베이스(multiple layered database)는 다양한 상황에서 효과적으로 지식을 추출할 수 있는 매우 유용한 방법이다. 본 논문에서는 AOG 기법을 이용하여 객체지향 데이터 모델에서 다계층 데이터베이스를 설계하는 방법론을 제안한다. 또한 구축된 다계층 데이터베이스에서 효과적인 정보 제공의 지속성을 유지하기 위한 방법으로 동적 스키마 변화 모델과 구현 전략을 제시한다.

A Multiple Layered Database Design and Maintenance in Object-Oriented Databases

Nam-Jin Kim[†] · Dong-Cheon Shin[††]

ABSTRACT

In very large databases, the problem of searching for interesting information effectively is very important in terms of efficiency and flexibility. A multiple layered database approach based on AOG(attribute-oriented generalization) method is one of the useful approaches for knowledge discovery under various situations. In this paper, we propose a multiple layered database design methodology based on AOG method in object-oriented databases. In addition, we propose a dynamic schema evolution model and implementation strategy in order to continue providing information effectively in multiple layered databases.

1. 서 론

최근 데이터베이스의 활용분야가 다양해짐에 따라 대량의 데이터를 포함하는 데이터베이스에서 원하는 정보를 찾는 작업은 용이한 일이 아니다. 따라서 원하는 정보를 효과적으로 추출하는 기술 즉 "데이터베이스에서의 지식 추출(knowledge discovery in database or data mining)" 기술이 필요하게 되었고 다양한 기법들이 이미 개발되었다[3].

데이터베이스에서의 지식 추출 기법들은 "일반화(generalization)"를 기반으로 지식을 추출하는 부류와 일반화 과정 없이 지식을 추출하는 부류로 나눌 수 있다[10]. 일반화를 기반으로 하는 부류는 다시 AOI (attribute-oriented induction : 이하 AOI) 기법에 기반하는 부류와 AOG(attribute-oriented generalization : 이하 AOG) 기법에 기반 하는 다계층 데이터베이스(multiple layered database : 이하 MLDB)의 두 부류로

† 준 회 원 : 중앙대학교 산업정보학과
†† 정 회 원 : 중앙대학교 산업정보학과
논문접수 : 1997년 7월 30일, 심사완료 : 1997년 11월 10일

나눌 수 있다.

AOI 기법은 관계형 데이터베이스에서 지식을 추출하기 위해서 개발되었다. AOI 기법은 미리 정의된 개념 계층에 의존해서 상위의 일반화된 개념으로부터 요약된 릴레이션을 얻은 후 질의에 나온 일반화 규칙을 적용하여 질의에 대한 답변을 유도해 낸다[9]. 이 기법은 궁극적으로 정적인 환경에서의 정보 추출에 기반을 두고 있다. 따라서 동적인 환경 즉, 연속적으로 주어지는 질의들 사이에 내용상 상관 관계가 없으며 질의의 내용이 수시로 변하는 상황에서 즉각적으로 정보를 추출하기에는 비용이나 기술의 복잡성 측면에서 합리적이지 못하다.

한편 AOG기반의 MLDB는 AOI 기법의 위와 같은 단점을 해결하는 지식 추출을 위한 매우 유용한 방법이다[8, 10, 11, 19]. AOG 기법은 AOI 기법과 비슷하게 미리 정의된 개념 계층[12]에 의존하지만 AOI 기법만큼 의존도가 크지 않고 동적인 상황에 즉각적으로 대처할 수 있도록 미리 예측된 상황에 맞게 끔 일정 수준까지 필요한 속성이나 릴레이션을 일반화해서 요약된 릴레이션까지만을 유도해낸다. 관계형 데이터 모델을 기반으로 한 AOG 기법은 참조 빈도 수가 낮은 속성을 제거함으로서 크기가 작고 요약된 행을 얻기 위한 "데이터 일반화"와 동일한 행을 합쳐서 크기가 작고 요약된 릴레이션을 얻기 위한 "릴레이션 일반화"로 나눌 수 있으며 릴레이션 일반화는 다시 일반화 과정에서 릴레이션의 기본키 또는 외래키가 변하는 "키 변경 일반화"와 기본키 또는 외래키가 변하지 않는 "키 보존 일반화"로 나누어진다. 키 보존 일반화는 상위 계층의 일반화된 릴레이션간에 필요에 의한 조인을 할 수가 있지만 키 변경 일반화는 상위 계층간에 필요에 의한 조인을 할 경우에 기본키 또는 외래키의 삭제로 인해서 불일치로 인한 이상 현상이 발생할 수도 있으므로 필요에 의한 조인을 할 수가 없다[10, 19].

MLDB는 사전에 빈번한 질의 기록(query history)의 통계치나 사용자 또는 전문가에 의해서 주어지는 질의를 분석해서 빈번히 참조되는 속성(attribute)이나 질의 패턴을 기초로 요약된 여러 개의 정보의 계층(layer)들로 구성된다. MLDB에서 계층(layer)은 요약된 릴레이션을 의미하며 전통적인 데이터베이스에 저장되던 데이터들이 MLDB의 최하위의 계층(layer-0, primitive layer)을 구성한다. MLDB는 최하위의 계층을 기반으로 전문가나 설계자에 의해서 설계된 일정 수준의 계층까지 AOG 기법을 적용하여 하위 계층에 저장된 데이터로부터 일반화된 정보를 상위 계층에 저장한다[8, 10, 19].

지금까지 데이터베이스에서의 지식 추출 분야에 대한 연구, 특히 MLDB에 관련한 연구는 주로 관계형 데이터베이스를 위한 지식 추출에 집중되어 있었다[3, 10, 11]. 관계형 데이터베이스에서 MLDB를 설계하기 위해서는 우선 빈번한 질의 기록의 통계치나 사용자 또는 전문가에 의해서 주어지는 질의를 분석해서 빈번히 참조되는 속성이나 질의 패턴을 기초로 일정 수준의 계층을 결정한다. AOG 기법에 기반해서 참조 빈도 수가 낮은 속성을 제거함으로서 크기가 작고 요약된 행을 얻기 위한 데이터 일반화와 동일한 행을 합쳐서 크기가 작고 요약된 릴레이션을 얻기 위한 릴레이션 일반화를 수행하여 결정된 수준까지 요약된 계층을 쌓는다. 구축된 MLDB의 계층간의 관계를 보여주는 계층표(route map)와 하위 계층에서 상위 계층으로 일반화된 경로(generalization path)를 보여주는 일반화 경로를 작성한다[10, 11].

그러나 계층적으로 요약된 정보를 표현하는 MLDB의 특징 때문에 관계형 데이터 모델보다는 객체지향 데이터 모델에서의 MLDB 설계가 여러 가지 잇점을 가질 수 있다. 객체의 계층 구조를 지원하고 풍부한 자료형을 제공하는 객체지향 데이터 모델은 자연스러운 계층의 표현 기능과 풍부한 자료형을 제공한다. 그러므로 관계형 데이터 모델에서 요구되던 계층 사이의 연결을 위한 추가적인 기법과 이에 따르는 비용을 감소시키고 제한적인 자료형으로 인한 정보 제공의 획일성을 감소시켜 지식 추출의 효과성을 높일 수 있다.

한편 MLDB는 빈번한 질의 기록의 통계치나 주요한 질의 패턴에 기초하고 있기 때문에 효과적인 정보 제공의 지속성을 유지하기 위해서는 질의 기록의 통계치나 질의 패턴의 변화에 대한 대처 기능을 제공해야 한다. 이러한 변화는 정보의 양이 조금씩 증가하고 데이터의 형태가 단순하던 시대에는 변화의 폭에 어느 정도 제한적인 측면을 갖고 있었다. 그러나 정보의 양이 대량으로 증가하고 데이터의 형태가 다양한 오늘날에는 사용자의 정보 요구 형태가 큰 변화의

폭을 갖게 되었을 뿐만 아니라 변화의 속도도 빨라지게 되었다. 그러므로 MLDB 설계 단계에서 변화에 대한 충분한 반영이 있어도 MLDB에서 효과적인 정보 제공의 지속성을 유지하기 위해서는 스키마 변화 기능의 필요성이 요구되며 나아가 MLDB와 같이 다양한 질의에 유연하게 대처해야하는 데이터베이스 시스템에서는 시스템을 정지시키지 않고도 데이터베이스의 전체적인 일관성을 유지하고 시스템의 성능을 저하시키지 않는 동적 스키마 변화 기능을 제공해야 한다.

본 논문에서는 관계형 데이터 모델에서 MLDB 설계를 위해 제안된 AOG 기법을 객체지향 데이터 모델에 적합하도록 적용하여 객체지향 데이터 모델에서 지식 추출을 위한 MLDB 설계 방법론을 제안한다. 또한 구축된 MLDB에서 효과적인 정보 제공의 지속성을 유지하기 위한 방법으로 동적 스키마 변화 모델을 제안하고 제안된 동적 스키마 변화 모델의 구현 전략을 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 객체지향 데이터 모델에 적합하도록 적용한 AOG 기법을 기반으로 하는 MLDB 설계 방법론을 제안한다. 3절에서는 MLDB의 동적 스키마 변화를 위한 접근 방법을 설명하고 MLDB의 동적 스키마 변화 모델을 제안한다. 4절에서는 3절에서 제안된 동적 스키마 변화 모델의 구현 전략을 제시한다. 끝으로 5절에서는 결론과 추후 연구 방향을 언급한다.

2. MLDB 설계 방법론

이 절에서는 관계형 데이터 모델에서 MLDB 설계를 위해 제안된 AOG 기법을 객체지향 데이터 모델에 적합하도록 적용하여 객체지향 데이터 모델에서 지식 추출을 위한 MLDB 설계 방법론을 제안한다.

2.1 객체지향 데이터 모델에서의 AOG 기법

객체지향 데이터 모델은 클래스/서브클래스(class/subclass) 계층에 속한 클래스들 안에 포함된 여러 개의 객체(object)들로 구성되며 객체, method, 계승(inheritance) 개념이 그 근간을 이루고 있다[1, 14, 15]. 실세계의 모든 개체(entity)들은 시스템 내에서 고유한 식별자인 객체 식별자(object identifier: 이하 OID)를 부여받아 객체로 모델링된다. 모델링 된 객체는 인스턴스(instance)로 표현되며 공통된 특성을 가진 인스턴스들이 모여서 하나의 클래스를 구성한다. 각 클래스 안의 객체는 속성으로서 자신의 데이터를 저장하고 계산 루틴이나 클래스에 관련된 규칙을 명시한 method를 정의한다. 클래스 계층 구조에 속한 객체는 계승 개념에 따라서 하위 객체가 상위 객체의 특성을 모두 계승받을 수 있다[4, 13, 16].

객체지향 데이터 모델에서 AOG 기법은 MLDB 설계시 하위 계층의 클래스에서 상위 계층의 클래스로 일반화된 정보를 추출하기 위해서 사용한다. 클래스는 공통된 특성을 가진 객체들로 이루어져 있다. 그러므로 하위 계층의 클래스를 일반화시키기 위해서는 클래스와 클래스에 정의된 method를 일반화시키는 클래스 일반화와 더불어 클래스에 속한 객체들을 일반화 시켜야 한다. 객체들은 시스템에 의해서 부여받는 고유한 식별자를 가지고 있고 속성으로서 자신의 데이터를 저장하고 있기 때문에 객체 일반화는 객체의 속성과 객체의 식별자인 OID 일반화로 구분된다. 그러므로 객체지향 데이터 모델에서의 AOG 기법은 객체의 속성과 OID를 일반화시키는 객체 일반화와 클래스와 클래스에 정의된 method를 일반화시키는 클래스 일반화로 나누어진다.

(1) 객체 일반화

객체 일반화는 객체의 속성을 일반화시키고 MLDB의 계층간 자연스러운 연결을 위해서 OID를 일반화시킴으로서 크기가 작고 요약된 객체를 얻는 방법이다.

①속성 일반화

속성 일반화는 다시 명시적으로 객체가 속성을 소유하고 있는 경우와 객체가 암시적으로 속성을 소유하고 있는 경우로 나뉜다. 명시적으로 객체가 속성을 소유하고 있는 경우는 AOI 기법의 데이터 일반화와 유사하다. 다만 객체지향 데이터 모델은 다양한 자료형의 표현이 가능하기 때문에 각각의 자료형에 적합한 일반화를 적용해야 한다. 본 논문에서는 문자와 숫자와 같은 구조적인 데이터와 긴 텍스트, 이미지, 음성, 지도, 그림, 영상과 같은 비구조적인 데이터로 구분하여 일반화시킨다. 구조적인 데이터는 미리 정의된 개념 계층에 의존해서 일반화시키고 비구조적

인 데이터는 크기, 색깔, 모양, 반복되는 멜로디, 음량, 음높이 같은 데이터의 특징이나 질의 패턴을 인식해서 일반화시킨다.

암시적으로 객체가 속성을 소유하고 있는 경우에는 다른 클래스에 속하는 객체의 OID를 저장해서 다른 클래스의 객체를 자신의 속성으로 소유하는 이종 객체들의 모임인 중첩 객체(nested object)의 경우에 해당한다. 중첩 객체는 실제로 다른 클래스에 속한 객체들을 조직화함으로서 만들어진 객체이기 때문에 속성이 속한 객체를 일반화해서 일반화된 객체의 OID를 해당 객체에 저장하는 방법을 사용한다.

② 객체 식별자 일반화

OID 일반화는 객체지향 데이터 모델에서 객체가 시스템에 의해서 유일하게 부여받는 값인 OID의 일반화에 해당한다. 하위 계층이 일반화 될 때 상위 계층의 객체가 일반화되기 전 하위 계층 객체의 OID를 특수한 목적의 속성인 previous_OID 속성에 보관함으로서 기존 객체의 특성을 보존할 수가 있다. 이것은 사용자가 상위 계층에서 세부적이고 구체적인 정보를 요구하는 경우에 관계형 데이터 모델에서의 MLDB처럼 별도의 하위 계층과의 추가적 연결 기법 없이 자연스러운 상하 계층간 연결을 가능하게 해준다.

(2) 클래스 일반화

클래스 일반화는 클래스에서 동일한 값을 갖는 객체를 합치고 동일한 객체의 수를 기록하기 위한 특수한 목적의 속성인 count 속성의 값을 갱신함으로서 크기가 작고 요약된 클래스를 얻는 방법이다. 클래스에서 정의된 method는 클래스에 속한 객체들의 상태를 변화시키는 객체의 고유한 행동을 나타낸다. 그러므로 클래스를 일반화시킨 후 일반화된 클래스에 속한 객체들의 속성을 고려하여 저장된 데이터의 연산을 처리할 수 있는 method를 정의해 주는 방법을 사용한다.

위에서 제시한 AOG 기법을 적용하여 객체지향 데이터 모델에서 MLDB를 구축하는 방법은 다음과 같이 정리된다.

- 입력:

객체지향 데이터베이스 스키마, 개념 계층, 빈번하게 참조되는 속성과 빈번하게 사용되는 질의 패턴

- 출력: MLDB 스키마
- 방법론:

〈단계 1〉 계층을 결정한다.

단계 1.1: 질의 기록의 통계치나 사용자 또는 전문가에 의해서 주어지는 질의를 분석해서 입력 값으로 주어진 참조 빈도수가 높은 속성이나 질의 패턴을 중심으로 발생할 수 있는 질의의 형태를 예측한다.

단계 1.2: 예측한 질의의 형태에 포함된 속성 값이나 질의 패턴에 기초해서 해당 속성을 어느 단계까지 일반화시킬 것인가를 결정한다.

〈단계 2〉 위의 〈단계1〉에서 결정된 일반화 계층에 따라서 상위의 계층을 쌓는다.

일반화 될 속성을 객체가 명시적으로 소유하고 있는지 암시적으로 소유하고 있는지를 구분하여 일반화 될 속성을 객체가 명시적으로 소유하고 있는 경우에는 개념 계층에 의존하거나 특성이나 질의 패턴을 인지하여 일반화시킨다. 일반화 될 속성을 객체가 암시적으로 소유한 경우에는 실제로 속성이 속한 객체를 일반화하여 일반화된 객체의 OID를 해당 객체에 저장한다.

〈단계 3〉 각 일반화된 클래스에서 동일한 객체를 합쳐서 count 속성에 동일한 객체의 수를 기록하고 이전 객체의 OID를 previous_OID 속성에 기록한다. 이와 함께 생성된 일반화 계층에 속한 객체들의 속성을 고려하여 저장된 데이터의 연산을 처리할 수 있는 method를 정의한다.

〈단계 2〉와 〈단계 3〉의 과정을 〈단계 1〉에서 결정된 계층까지 반복한다.

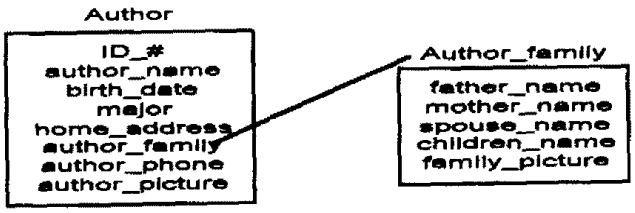
〈단계 4〉 계층간의 관계를 보여주는 계층표와 하위 계층에서 상위 계층으로 일반화 된 경로를 보여주는 일반화 경로를 포함한 새로운 스키마를 작성한다. 일반화 경로는 하위 계층이 상위 계층의 속성으로 어떻게 일반화되었는가를 알려주는 역할을 하며 〈일반화되기전 속성, 일반화된 속성, 일반화 방법〉의 형태로 작성된다.

2.2 MLDB 설계 예

이 절에서는 위에서 제안한 MLDB 설계 방법론을 적용한 설계 예를 제시한다.

저자에 대한 정보를 저장하고 있는 저자 데이터베이스를 기반으로 MLDB를 설계한다고 가정하자. MLDB 설계 방법론의 입력 값으로는 저자 데이터베이스의 스키마와 저자 데이터베이스에 저장된 데이터들의 배경지식인 개념 계층 그리고 빈번하게 참조되는 속성과 빈번하게 사용되는 질의 패턴이 있다. 저자 데이터베이스는 저자에 대한 정보를 저장하고 있는 Author 클래스와 저자의 가족에 대한 정보를 저장하고 있는 Author_family 클래스로 구성되어 있다. Author 클래스의 author_family 속성은 정의역으로 Author_family 클래스를 갖는 중첩 객체이다.

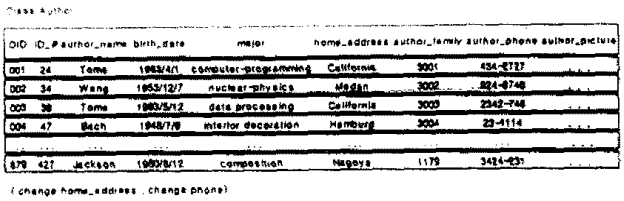
클래스 계층은 다음과 같다.



책의 저자에 대한 정보를 담고 있는 Author 클래스의 개념 계층의 내용은 다음과 같다.

{computer programming, data processing,
data system repair, ... , microcomputer software}
⊂computer science
{analytical chemistry, astronomy, earth science,
nuclear physics, ... , geology}
⊂ physical science
{composition, music theory, instrumental performance,
. . . . . , music history} ⊂ music
{textile design, interior decoration, fashion design,
. . . . . , commercial art} ⊂ design
{computer science, physical science} ⊂ science
{music, design} ⊂ art
{science, art } ⊂ ANY(major)

Author 클래스의 구체적인 인스턴스와 method의 내용은 다음과 같다.



OID	ID_#	author_name	birth_date	major	home_address	author_family	author_phone	author_picture
001	24	Tome	1963/4/1	computer programming	California	3001	434-2727	
002	34	Wang	1953/12/7	nuclear physics	Medan	3002	824-9748	
003	38	Tome	1960/5/12	data processing	California	3003	2342-748	
004	47	Bach	1948/7/8	interior decoration	Hamburg	3004	23-4114	
878	427	Jackson	1960/8/12	composition	Nagoya	1178	3424-631	

(change home_address , change phone)

Author 클래스는 원시적인 형태의 데이터만을 포함하고 있는 것으로 MLDB의 계층-0에 해당한다. 계층-0에서는 포괄적이며 개략적인 정보도 얻을 수 없고 지적인 질의 답변도 기대할 수 없다. 따라서 계층-0을 기반으로 AOG 기법과 미리 정의된 개념 계층을 사용해서 계층-1을 유도해 낸다.

〈단계 1〉 계층을 결정한다.

단계 1.1: 질의 기록의 통계치나 사용자 또는 전문가에 의해서 주어지는 질의를 분석해서 input 값으로 주어진 참조 빈도수가 높은 속성을 birth_date, major, home_address, author_family, author_picture에 대한 정보라고 가정하고 발생 가능한 질의를 예측한다.

단계 1.2: 빈번하게 참조되는 속성과 질의 패턴은 주로 저자의 나이 대별로 넓은 의미의 전공 명칭을 통한 검색이나 또는 저자의 거주 지역을 통한 검색이라고 가정해서 관련 속성들을 2계층까지 일반화시키기로 결정했다고 가정한다.

〈단계 2〉 위의 〈단계 1〉에서 결정된 일반화 계층에 따라서 상위의 계층을 쌓는다.

author_family 속성만 객체가 암시적으로 소유하고 있고 나머지 속성은 모두 명시적으로 소유하고 있다.

기본키 ID_#와 author_name 속성은 그대로 두고 author_phone 속성과 같이 일반화의 필요성이 없는 정보는 제거한다. 구조적인 데이터를 가진 속성인 birth_date는 years_old로 home_address는 home_area로 각각 일반화되고 major 속성의 경우에는 미리 정의된 개념 계층을 기반으로 상위 단계의 개념으로 일반화된다. author_picture 속성과 같이 비구조적인 데이터는 이미지 데이터의 크기를 이용해서 author_picture_size 속성으로 일반화 시켰다.

암시적으로 객체가 속성을 소유한 author_family 속성의 일반화는 Author_family 클래스를 위와 비슷한 방법으로 일반화 시켜서 객체의 모든 속성을 #_of_family로 일반화 시켰다고 가정하고 일반화된 객체의 OID를 Author 클래스의 #_of_family 속성에

기록한다.

〈단계 3〉 일반화된 객체 중에서 동일한 객체를 합쳐서 count 속성에 개수를 기록하고 일반화되기 이전 객체의 OID를 previous_OID 속성에 기록한다. 또한 나이와 주소의 변경을 처리할 수 있는 change years_old( )와 change home_area( ) method를 부여함으로서 Author' 클래스를 생성한다.

Class Author'

OID	ID_#	author_name	years_old	major	home_area	#_of_family	author_picture_size	previous_OIDCOUNT
1111	24	Tom	36	computing science	USA	5111	54k	001	1
1112	34	Wang	45	physics science	Indonesia	5112	121k	002	1
1113	38	Tom	36	computing science	USA	5113	54k	003	1
1114	47	Bach	42	design	Germany	5114	35k	004	1
1987	427	Jackson	38	music	Japan	5987	77k	879	1

(change years_old , change home_area)

계층-1인 Author' 클래스는 다시 〈단계 2〉와 〈단계 3〉의 과정을 반복해서 일반화된다. 기본키 ID_#와 author_name, #_of_family, author_picture_size 속성들은 제거된다. years_old는 years_old_range로 일반화되고 major 속성은 개념 계층의 보다 상위의 개념으로 일반화된다. 그리고 동일한 객체가 합쳐져서 count 속성에 기록되고 일반화되기 이전 객체의 OID를 previous_OID 속성에 기록한다. 아울러 일반화된 Author 클래스에서 객체의 특성을 고려하여 일어날 수 있는 연산을 처리하기 위한 change years_old_range( )와 change home_area( ) method를 부여함으로서 계층-2인 Author 클래스를 생성한다.

Class Author

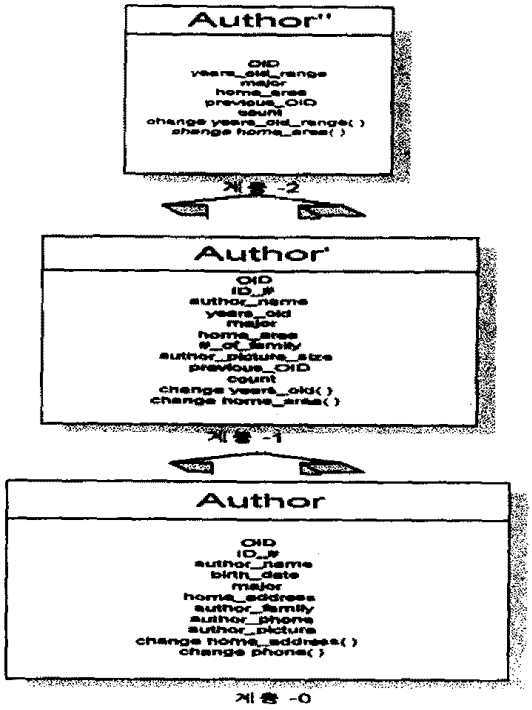
OID	years_old_range	major	home_area	previous_OID	COUNT
2111	31-40	science	USA	1111,1113,...	42
2112	41-50	science	Indonesia	1112,...	10
2113	41-50	art	Germany	1114,...	27
...	...	...	...	...	...
2542	31-40	art	Japan	1987,...	...

(change years_old_range , change home_area)

〈단계 4〉 계층간의 관계를 보여주는 계층표는 다음과 같다.((그림 1) 참조)

하위 계층에서 상위 계층으로 일반화 된 경로를 보여주는 일반화 경로는 다음과 같다.((그림 2) 참조)

최종적으로 구축된 MLDB에는 포괄적이며 요약적인 질의가 가능하다. 예를 들어 "나이가 30-40대이며 U.S.A에 거주하는 저자는 전체의 얼마나 되는가?"라는 질의는 계층-2의 Author 클래스를 검색하여 전체 저자 중에서 42명이 조건에 부합됨을 알 수 있고 더 세부적인 정보는 하위 계층을 검색하여 얻을 수 있다.



(그림 1) MLDB의 계층표
(Fig. 1) Route map of MLDB

<계층 1>
<birth_date, years_old, 생년월일을 나이로 계산>
<major, major, 개념 계층에 의존>
<home_address, home_area, 지역범위 확장>
<author_family, #_of_family, 가족인원의 수로 일반화>
<author_picture, author_picture_size, 이미지 데이터의 크기로 일반화>

<계층 2>
<years_old, years_old_range, 나이대별로 일반화>
<major, major, 개념 계층에 의존>

(그림 2) 일반화 경로
(Fig. 2) Generalization path

3. MLDB의 동적 스키마 변화

이 절에서는 MLDB의 동적 스키마 변화를 위한 접근 방법을 설명하고 동적 스키마 변화 시에 기반 데이터베이스와 MLDB의 일관성 유지를 위한 MLDB의 동적 스키마 변화 모델을 제안한다.

3.1 접근 방법

MLDB는 하위 계층을 기반으로 상위 계층으로 일

반화되기 때문에 하위 계층의 바탕이 되는 기반 데이터베이스의 스키마 변화는 상위의 MLDB에 직접적인 영향을 미치게 된다. 그러므로 MLDB의 동적 스키마 변화는 MLDB의 스키마 변화뿐만 아니라 기반 데이터베이스의 스키마 변화도 고려해야 한다.

3.1.1 기반 데이터베이스의 스키마 변화

기반 데이터베이스의 스키마 변화는 기반 데이터베이스를 기반으로 구축된 MLDB에 영향을 미치게 된다. 그러나 기반 데이터베이스의 스키마 변화가 항상 MLDB에 영향을 미치는 것은 아니다. 기반 데이터베이스의 스키마 변화가 MLDB에 영향을 미치느냐 미치지 않느냐의 판단은 스키마 변화에 해당하는 기반 데이터베이스에 포함된 객체의 일반화 여부를 기준으로 결정된다. 예를 들어 기반 데이터베이스에 새로운 속성이 첨가되었다고 가정하자. 새로운 속성은 설계자나 전문가에 의해서 일반화 가치 여부를 판단 받게 되고 이때 만약 새로운 속성이 일반화 가치가 있다고 판단되면 MLDB의 스키마에 영향을 미치게 되고 그렇지 않다면 MLDB의 스키마에 영향을 미치지 않는다.

기반 데이터베이스인 객체지향 데이터베이스의 동적 스키마 변화를 위한 접근 방법은 논리적 스키마에 대한 직접적인 수정인 적응식 접근(adaptational approach)과 스키마에 대한 버전(version)을 생성하는 스키마 버저닝 접근(schema versioning approach)으로 나누어진다[7]. 기반 데이터베이스의 동적 스키마 변화의 2가지 접근 방법 중 스키마 버저닝 접근은 객체지향 데이터베이스에 저장된 객체들이 스키마의 변화에 대응하여 실제로 변경되지 않고 변화된 스키마의 정의에 맞는 새로운 객체를 만들어서 뷰를 통하여 접근하도록 하는 방법이다[17, 18]. 이 방법은 버전들을 저장하기 위한 별도의 추가적 저장 공간이 필요하고 시스템에게 버전 관리에 관계된 많은 오버헤드를 주어 시스템의 성능을 저하시킨다. 그러므로 구축된 MLDB에 영향을 미치는 기반 데이터베이스의 동적 스키마 변화는 논리적 스키마에 대한 직접적인 수정인 적응식 접근 방법으로 제한한다.

3.1.2 MLDB의 스키마 변화

MLDB의 스키마 변화는 빈번히 참조되는 속성의 변경이나 주요한 질의 패턴의 변화에 기인한 스키마 변화와 데이터베이스에 저장된 데이터들의 배경 지식인 개념 계층의 변경에서 기인한 스키마 변화로 나누어진다.

전자의 경우에는 일반화된 속성을 삭제하거나 일반화되지 않은 속성을 새롭게 일반화시키거나 새로운 일반화 과정을 통해서 계층을 형성하는 경우에 해당한다. 후자의 경우에는 개념 계층에 의존해서 일반화된 속성을 새로운 일반화 과정을 통해서 재구축 하는 경우에 해당한다.

전자의 경우뿐만 아니라 후자의 경우도 궁극적으로는 필요에 의해서 기반 데이터베이스로부터 추출된 요약된 정보를 삭제하거나 일반화시키는 작업이다. MLDB 내에서의 일반화된 속성의 삭제나 새로운 속성의 일반화는 MLDB 설계 단계에서와 같은 속성의 일반화 가치 판단의 작업이기 때문에 MLDB의 정보 제공원인 기반 데이터베이스의 변화를 초래하지는 않는다. 그러므로 MLDB의 스키마 변화는 기반 데이터베이스에 영향을 미치지 않는다.

3.2 MLDB의 동적 스키마 변화 모델

데이터베이스의 동적 스키마 변화는 특정한 데이터 모델에 의존하여 설계된 동적 스키마 변화 모델에 기초해서 수행된다[2]. 동적 스키마 변화 모델이란 스키마 변경 시에도 데이터베이스의 일관성 유지를 위해 준수되어야 하는 성질 즉 "스키마 불변성"을 정의하고 이들 불변성들을 유지하기 위한 규칙들로 이루어진다. 데이터베이스의 동적 스키마 변화는 데이터베이스내의 모든 객체들에게 파급적으로 영향을 미친다.

MLDB는 여러 개의 정보의 계층으로 구성되며 하위 계층은 상위 계층에 연결되어 세부적인 정보를 제공한다. 그러므로 MLDB는 동적 스키마 변화가 발생한 후에도 각 계층이 완전히 연결되어 격리된 노드가 없어야 하며 하위 각 계층은 상위 계층으로부터 접근 가능하여야 한다. 아울러 하위 계층은 상위 계층에 지속적인 정보의 제공을 보장해야 한다. MLDB의 동적 스키마 변화 모델은 각 계층의 완전한 연결을 위하여 "계층 불변성"을 정의하고 하위 계층이 상위 계층에 지속적인 정보 제공의 보장을 유지하기 위한 "일반화 속성 불변성"과 "계층간 일반화 객체 불변

성"을 정의한다.

각 불변성의 정의는 다음과 같다.

- 계층 불변성

  MLDB의 각계층은 완전히 연결되어 격리된 노드가 없어야 하며 하위 각 계층은 상위 계층으로부터 접근 가능하여야 한다.
- 일반화 속성 유지 불변성

  상위 계층으로 일반화된 속성은 하위 계층에서 일반화 근원 속성의 삭제 없이는 상위 계층에서 삭제 될 수 없어야 한다. 이는 상위 계층에서 하위 계층에 보다 구체적인 정보를 요구하는 경우에 하위 계층은 정보의 제공을 보장해야 한다는 점에 기인한다.
- 계층간 일반화 객체 불변성

  상위 계층으로 일반화된 객체는 하위 계층의 일반화 근원 객체의 OID를 속성으로 가지고 있다. 계층 불변성과 일반화 속성 유지 불변성을 만족하는 한 하위 계층 객체의 OID를 저장하고 있는 속성은 변경될 수 없어야 한다.

또한 MLDB의 동적 스키마 변화 모델은 위의 3가지 불변성들을 준수하기 위한 6가지 규칙들로 이루어진다. 각 규칙들은 O2 객체지향 데이터베이스의 스키마 변화 모델의 분류[7]에 기초해서 기반 데이터베이스의 스키마 변화와 MLDB의 스키마 변화의 경우로 분리하여 적용하였다.

- Rule 1. 속성의 첨가
  - R1.1

    기반 데이터베이스에 첨가된 새로운 속성이 일반화시킬 가치가 있는 속성이라면 MLDB 설계 방법론의 〈단계 2〉와 〈단계 3〉 과정을 통하여 상위 계층으로 점진적으로 일반화된다. 이때 동일한 객체의 수를 기록하는 속성인 count 속성의 값은 변경된다.
  - R1.2

    기반 데이터베이스에 첨가된 새로운 속성이 일반화시킬 가치가 없는 속성이라면 MLDB에 영향을 미치지 않는다.
  - R1.3

    MLDB에서 새롭게 일반화의 대상이 된 속성은 MLDB 설계 방법론의 〈단계 2〉와 〈단계 3〉 과정을 통하여 상위 계층으로 점진적으로 일반화된다. 이때 동일한 객체의 수를 기록하는 속성인 count 속성의 값은 변경된다.

속성의 첨가는 기반 데이터베이스에 새로운 속성이 첨가되는 경우와 구축된 MLDB에 이전에도 존재하고 있었지만 일반화의 대상에서 제외되었다가 주요한 질의 패턴의 변경으로 인해서 참조 빈도수가 증가되어 일반화의 필요성이 증대된 속성이 상위 계층으로 일반화되는 경우로 나눌 수 있다. 전자의 경우에는 새로운 속성의 일반화 가치를 판단하여 일반화의 가치가 없다고 판단되면 R1.2에 따라서 구축된 MLDB에 영향을 미치지 않는다. 그러나 새로운 속성이 일반화 가치가 있다고 판단되면 계층 불변성을 준수하기 위해서 MLDB의 최하위 계층에 삽입된 후 R1.1에 따라서 상위 계층으로 점진적인 일반화 여부를 결정하게 된다. 후자의 경우에는 계층 불변성을 준수하기 위해서 최하위 계층에서부터 R1.3에 따라서 상위 계층으로 점진적인 일반화 여부를 결정하게 된다. 이때 기반 데이터베이스에는 영향을 미치지 않는다.

- Rule 2. 속성의 삭제
  - R2.1

    기반 데이터베이스에서 삭제된 속성이 이미 일반화된 속성이라면 속성이 삭제 되기전 MLDB의 일반화된 최상위의 계층에서부터 속성이 삭제된 계층까지 모두 제거된다.
  - R2.2

    기반 데이터베이스에서 삭제된 속성이 일반화되지 않은 속성이라면 MLDB에는 영향을 미치지 않는다.
  - R2.3

    MLDB에서 속성의 삭제는 속성이 삭제 되기전 일반화된 최상위의 계층에서부터 속성이 삭제된 계층까지 모두 제거된다. 그러나 속성이 삭제된 계층의 하위 계층으로는 영향을 미치지 않는다.

속성의 삭제는 기반 데이터베이스에서 속성이 삭제되는 경우와 구축된 MLDB에서 일반화된 속성이

주요한 질의 패턴의 변경으로 참조 빈도수가 감소되어 삭제되는 경우로 나눌 수 있다. 전자의 경우에는 일반화 여부를 기준으로 일반화되어 있지 않은 속성이라면 R2.2에 따라서 구축된 MLDB에 영향을 미치지 않는다. 그러나 속성이 상위 계층으로 일반화되었다면 일반화 속성 유지 불변성을 준수하기 위해서 R2.1에 따라서 속성이 삭제 되기전 최상위의 계층에서부터 속성이 삭제된 계층까지 모두 제거된다. 후자의 경우에는 일반화 속성 유지 불변성을 준수하기 위해서 R2.3에 따라서 속성이 삭제 되기전 상위 계층에서의 변화만 가져온다. 이때 MLDB의 하위 계층과 기반 데이터베이스에는 영향을 미치지 않는다.

- Rule 3. 일반화된 속성의 도메인 변경
  일반화된 속성의 도메인 변경은 일반화 단계의 재구축을 요구 할 수도 있다. 하위 계층 속성의 도메인에 의존해서 일반화를 수행하는 MDLB에서 하위 계층의 도메인 변경은 새로운 일반화 과정을 통한 상위 계층의 재구축을 요구한다.

MLDB 계층-0의 birth-date 속성의 도메인을 {1900.1.1-1950.1.1}이라고 가정한다면 저자의 나이는 {100-40} 사이의 값을 가지게 된다. 저자의 나이에 기초해서 크게 세 부분으로 범주를 지으면 [{100-90}, {89-70}, {69-40}]이고 저자의 나이는 이 세 범주의 하나에 속하게 된다. 이제 계층-0의 birth-date 속성의 도메인이 {1910.1.1-1970.1.1}로 변경되었다고 가정하자. 저자의 나이는 {90-20}사이의 값을 가지게 되고 저자의 나이에 기초해서 크게 세 부분으로 범주를 지으면 [{90-70}, {69-40}, {39-20}]이고 저자의 나이는 이 세 범주의 하나에 속하게 되며 도메인 변경전과는 전혀 다른 범주로 일반화되었다. Rule3에 따라서 일반화되어진 속성의 도메인 변경은 새로운 일반화 과정의 재구축을 요구하기 때문에 도메인 변경시 많은 주의를 필요로 한다.

- Rule 4. 한 클래스가 일반화된 클래스의 상위 클래스로 첨가
  - R4.1
    계승받은 새로운 속성이 일반화시킬 가치가 있는 속성이라면 R1.1을 따른다.
  - R4.2
    계승받은 새로운 속성이 일반화시킬 가치가 없는 속성이라면 R1.2를 따른다.

한 클래스가 일반화된 클래스의 상위 클래스로 첨가되는 경우는 기반 데이터베이스에 새로운 클래스가 일반화된 클래스의 상위 클래스로 첨가되는 경우에 해당한다. 새로운 클래스는 하위 클래스들에게 자신의 속성과 메소드를 계승한다. 그러므로 설계자는 계승받은 속성의 일반화 가치를 판단하여 일반화시킬 가치가 있다면 R1.1을 따르고 구축된 MLDB에 영향을 미치지만 그렇지 않다면 R1.2를 따르고 구축된 MLDB에 영향을 미치지 않는다.

- Rule 5. 한 클래스가 일반화된 클래스의 상위 클래스에서 삭제
  - R5.1
    삭제된 상위 클래스에서 계승받은 속성이 이미 일반화된 속성이라면 R2.1을 따른다.
  - R5.2
    삭제된 상위 클래스에서 계승받은 속성이 아미 일반화되지 않은 속성이라면 R2.2를 따른다.

한 클래스가 일반화된 클래스의 상위 클래스에서 삭제되는 경우는 기반 데이터베이스에 존재하는 클래스가 일반화된 클래스의 상위 클래스에서 삭제되는 경우에 해당한다. 삭제된 클래스는 하위 클래스들에게 자신의 속성과 메소드를 계승했다. 그러므로 설계자는 계승받은 속성이 일반화되었는가를 판단하여 일반화되었다면 R2.1을 따르고 구축된 MLDB에 영향을 미치지만 그렇지 않다면 R2.2를 따르고 구축된 MLDB에 영향을 미치지 않는다.

- Rule 6. 일반화된 클래스의 삭제
  일반화된 클래스를 삭제하는 것은 해당 클래스를 기반으로 구축된 MLDB의 삭제를 의미한다.

위의 6가지 규칙들을 MLDB의 동적 스키마 변화를 위한 접근 방법의 측면에서 정리하면 기반 데이터베이스의 스키마 수정이 MLDB에 영향을 미치는 경우는 R1.1, R2.1, R3, R4.1, R5.1, R6에 해당하고 기반 데이터베이스의 스키마 수정이 MLDB에 영향을

미치지 않는 경우는 R1.2, R2.2, R4.2, R5.2에 해당한다. 그리고 MDLB의 스키마 수정은 R1.3, R2.3에 해당한다.

4. MLDB에서 동적 스키마 변화의 구현 전략

이 절에서는 객체지향 데이터베이스에서 동적 스키마 변화를 구현하는 전략들을 살펴보고 MLDB에서 동적 스키마 변화를 구현하는 전략이 갖추어야할 조건들을 제시한다. 제시된 조건들을 기초로 MLDB에서 동적 스키마 변화를 구현하는 전략을 제시한다.

동적 스키마 변화가 발생한 후부터 데이터베이스 내의 데이터들이 새로운 스키마의 정의에 맞게 갱신되는 과정을 "데이터베이스 변형"이라고 한다[5]. 객체지향 데이터베이스에서 동적 스키마 변화를 구현하는 방법은 즉각갱신(immediate update) 방법과 지연갱신(deferred or lazy update) 방법 그리고 혼합갱신 방법으로 분류할 수 있다[5, 6]. 세 가지 방법들은 모두 변형 함수(conversion function)를 이용하는 방법이다. 변형 함수란 현재의 스키마 클래스의 정의가 새로운 스키마 클래스의 정의에 맞도록 어떻게 변형되어야 하는 가를 시스템에게 알려주는 역할을 한다[5, 6]. 변형 함수는 변형시 참조하는 객체의 유·무에 따라서 단순 변형 함수(simple conversion function)와 복합 변형 함수(complex conversion function)로 나눌 수 있다. 단순 변형 함수는 어떤 한 클래스가 변경시 다른 클래스를 참조하지 않고 자기 클래스 객체의 값만으로 변경되는 함수이다. 반면에 복합 변형 함수는 어떤 한 클래스가 변경시 다른 클래스 객체의 값을 참조해서 변경되는 함수이다.

객체지향 데이터베이스에서 동적 스키마 변화의 구현 전략들은 MLDB의 각 계층이 객체지향 데이터베이스를 기반으로 일반화되었기 때문에 MLDB의 각 계층에는 적용 가능하다. 그러나 단일 계층에서의 스키마 변화만을 고려하였기 때문에 MLDB의 모든 계층에 일괄적으로 적용하기에는 부적당하다.

MLDB는 대용량의 데이터베이스를 기반으로 구축되기 때문에 한 번의 스키마 변화가 장시간에 걸쳐서 연속적으로 계층화된 대용량의 데이터의 변화를 초래할 수도 있다. 그러므로 MLDB에서 동적 스키마 변화의 구현 전략은 비용의 낭비를 최소화하며 응용 프로그램의 접근 제한이 없어야 한다. 또한 일반화 과정을 통하여 계층화된 여러 객체에서 데이터베이스의 일관성을 유지하는 적절한 변형 전략을 선택하는 것은 용이한 일이 아니다. 그러므로 MLDB의 동적 스키마 변화의 구현 전략은 시스템이나 설계자에게 부가되는 추가적 작업 없이 비교적 자동화된 변형 절차를 거쳐야 한다.

MLDB에서 동적 스키마 변화의 구현 전략이 갖추어야 할 조건을 정리하면 다음과 같다.

(조건 1) 기반 데이터베이스와 MLDB간의 일관성이 보장되어야 한다.
(조건 2) 비용의 낭비를 최소화해야 한다.
(조건 3) 시스템이나 설계자에게 부가되는 추가적 작업 없이 비교적 자동화된 변형 절차를 거쳐야 한다.

즉각갱신 방법은 스키마 변화가 발생하자마자 곧바로 변화된 클래스의 모든 객체를 응용 프로그램이 데이터베이스에 접근하기 전에 변화된 스키마의 정의에 맞게 변경시키는 방법이다. 그러나 MLDB는 대용량의 데이터베이스를 기반으로 구축되기 때문에 한 번의 스키마 변화가 긴 시간에 걸쳐서 연속적으로 계층화된 대용량의 데이터의 변화를 초래할 수도 있다. 그러므로 즉각갱신 방법은 MLDB의 동적 스키마 변화 시에 비용의 낭비를 가져올 뿐만 아니라 긴 시간에 걸친 응용 프로그램의 접근 제한을 가져 올 수도 있다.

지연갱신 방법은 스키마 변화가 발생한 후에 변화된 클래스의 객체를 응용 프로그램이 접근할 때까지 물리적으로는 변경을 하지 않고 논리적으로만 변경시키는 방법이다. 그러나 MLDB의 각 계층은 하위 계층으로부터 상위 계층으로 요약된 정보를 추출하여 구축되기 때문에 각 계층이 밀접하게 연관되어 있다. 그러므로 어떤 경우에는 스키마 변화가 발생한 후에 즉각갱신이 이루어지지 않으면 계층간 불일치를 가져 올 수도 있다.

혼합갱신 방법은 즉각갱신 방법과 지연갱신 방법을 상황에 따라 바꿔가며 사용하는 방법으로 비관적인 혼합 데이터베이스 변형 방법과 낙관적인 혼합 데이터베이스 변형 방법으로 나누어진다. 비관적인 혼

합 데이터베이스 변형 방법은 데이터베이스의 불일치를 초래 할 수도 있는 복합 변형 함수일 때는 무조건 즉각갱신 방법을 사용하고 단순 변형 함수일 때는 무조건 지연갱신 방법을 사용하는 방법이다. 반면에 낙관적인 혼합 데이터베이스 변형 방법은 가능한 비용이 많이 들고 전체 데이터베이스에 대한 접근이 제한되는 즉각갱신 방법의 사용을 피하고 지연갱신 방법을 적용하면 데이터베이스의 불일치를 초래하는 경우에만 즉각갱신 방법을 사용하는 방법이다. 그러나 MLDB의 각 계층은 대부분의 경우에 상위의 계층 또는 하위의 계층과 연관을 갖고 있기 때문에 대부분 복합 변형 함수이다. 그러므로 비관적인 혼합 데이터베이스 변형 방법은 MLDB의 동적 스키마 변화시에 대부분 즉각갱신 방법을 적용하여 비용의 낭비를 가져온다. 반면에 낙관적인 혼합 데이터베이스 변형 방법은 시스템이나 설계자에게 적절한 변형 전략을 선택하기 위한 추가적 부담을 준다.

따라서 새로운 MLDB의 동적 스키마 변화의 구현 전략은 동적 스키마 변화 모델과 데이터베이스 변형 사이에 보다 유기적인 결합을 통하여 데이터베이스 변형과 동시에 데이터베이스의 불일치 문제점들을 해결하는 것이다. MLDB에서 동적 스키마 모델의 구현 전략은 MLDB의 동적 스키마 변화 모델에 기반해서 기반 데이터베이스와 MLDB의 불일치를 초래하는 경우인 기반 데이터베이스의 스키마 변경이 MLDB에 영향을 미치는 경우에는 계층간의 일관성 유지를 위하여 즉각갱신 방법을 적용한다. 그러나 기반 데이터베이스와 MLDB의 불일치를 초래하지 않는 경우인 기반 데이터베이스의 스키마 변경이 MLDB에 영향을 미치지 않는 경우와 MLDB의 스키마 변화의 경우에는 비용의 낭비를 최소화하며 응용 프로그램의 접근 제한이 없는 지연갱신 방법을 적용한다.

MLDB의 동적 스키마 변화의 구현 전략을 동적 스키마 변화 모델의 규칙에 적용하면 다음과 같다.((그림 3) 참조)

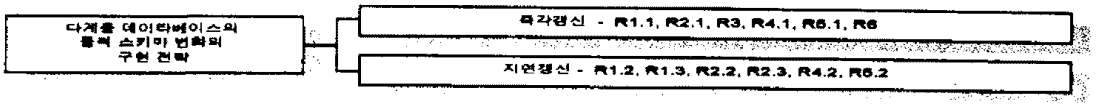
5. 결 론

본 논문에서는 관계형 데이터 모델에서 MLDB 설계를 위해 제안된 AOG 기법을 객체지향 데이터 모델에 적합하도록 적용한 객체지향 데이터베이스 환경에서 지식 추출을 위한 MLDB 설계 방법을 제안하였다. 또한 구축된 MLDB에서 정보 제공의 지속성을 유지하기 위한 방법으로 동적 스키마 변화 모델과 구현 전략을 제시하였다.

복합 중첩 자료(composite nested data)나 계층형 자료(hierarchical data)와 같은 풍부한 자료형과 데이터 내용과 관계의 의미를 폭넓게 표현할 수 있는 객체지향 데이터 모델을 사용함으로서 객체지향 데이터베이스에서 AOG 기법은 MLDB에 유용성과 더 많은 표현력을 제공해 준다. 뿐만 아니라 사용자가 상위 계층에서 세부적인 정보를 요구할 경우 기존의 관계형 데이터 모델에서의 MLDB처럼 별도의 하위 계층과의 연결 기법이 필요 없고 객체지향 데이터 모델의 요소중 하나인 OID를 이용해서 자연스럽게 상하 계층간의 연결이 가능하다.

MLDB의 동적 스키마 변화 모델은 동적 스키마 변화 모델과 구현 전략 사이에 보다 유기적인 결합을 통하여 데이터베이스 변형과 동시에 데이터베이스의 불일치 문제점들을 해결한다. 제시한 MLDB의 동적 스키마 변화의 구현 전략은 기반 데이터베이스와 MLDB의 일관성을 보장하며 비용의 낭비를 최소화한다. 또한 데이터베이스 변형 과정 동안 응용 프로그램의 접근 제한이 없으며 비교적 자동화된 변형 절차를 거친다.

본 논문에서는 MLDB의 일반화된 상위 계층간의 결합은 정보의 요약 정도를 높일 수 있는 방법이지만 고려하지 않았다. 또한 개념 계층의 자동화된 변형은



(그림 3) MLDB의 동적 스키마 변화의 구현전략
(Fig. 3) Implementation strategy of dynamic scheme evolution for MLDB

MLDB의 유연성을 향상시키기 위한 방법이지만 MLDB의 동적 스키마 변화의 구현 전략에 개념 계층의 자동화된 변경은 고려하지 않았다. 데이터베이스에서의 지식 추출 기법들은 일반화를 기반으로 하는 부류와 일반화를 기반으로 하지 않는 부류로 나누어지며 현재 각 부류에 속하는 다양한 기법들이 개발되어 있다. 향후 데이터베이스에서의 지식 추출 차원에서 일반화를 기반으로 하는 기법과 일반화를 기반으로 하지 않는 기법들간의 비교뿐만 아니라 MLDB를 비롯한 데이터베이스에서의 지식 추출을 위한 여러 기법들간의 비교 분석 연구가 기대된다.

참 고 문 헌

[1] M. Atkinson, F. Bancilhon, D. Dewitt, K. Dittich, D. Maier, and S. Zdonik, "The Object-Oriented Database System Manifesto", Proc. of 1st Int'l Conf. on Deductive and Object-Oriented Databases(DOOD'89), Japan, 1989.

[2] P. Breche and M. Worner, "Schema Updates Primitives for ODB Design", Technical Report, 7/15, DBIS, University of Frankfurt, May, 17, 1995.

[3] M. S. Chen, J. Han, and P. S. Yu, "Data Mining : An Overview from Database Perspective" IEEE Transaction on Knowledge and Data Engineering, 8(6), 1996, pp. 866-883.

[4] Joseph Fong, "Mapping Extended Entity Relationship Model to Object Modeling Technique", SIGMOD Record, 24(3), Sept. 1995, pp. 18-22.

[5] F. Ferrandina, T. Meyer, and R. Zicari, "Implementing Lazy Database Updates for an Object Database System", Proc. of 20th Int'l Conf. on Very Large Databases, Santiago, Chile, Sept. 1994, pp. 261-272.

[6] F. Ferrandina, T. Meyer, and R. Zicari, "Correctness of Lazy Database Updates for an Object Database System", Proc. of 6th Int'l Workshop on Persistent Object Systems, Tarascon, France, Sept. 1994, pp. 284-301.

[7] F. Ferrandina and S. -E. Lautemann, "An Integrated Approach to Schema Evolution for Object Databases", Proc. of 3rd Int'l Conf. on Object-Oriented Information Systems(OOIS), London, UK, Dec. 1996, pp. 280-294.

[8] J. Han, O. R. Zaiane, and Y. Fu, "Resource and Knowledge Discovery in Global Information Systems : A Multiple Layered Database Approach, Technical Report CMPT94-10, Nov. 19 94.

[9] J. Han, Y. Cai and N. Cercone, "Knowledge Discovery in Databases : An Attribute-Oriented Approach", Proc. of 1992 Int'l Conf. on Very Large DataBases(VLDB'92), Vancouver, Canada, Aug. 1992, pp. 547-559.

[10] J. Han, Y. Huang, N. Cercone, and Y. Fu, "Intelligent Query Answering by Knowledge Discovery Techniques", IEEE Transactions on Knowledge and Data Engineering, 8(3), 1996, pp. 373-390.

[11] J. Han, Y. Fu, and R. Ng, "Cooperative Query Answering Using Multiple Layered Databases", Proc. of 2nd Int'l Conf. on Cooperative Information Systems(CoopIS'94), Toronto, Canada, May, 1994, pp. 47-58.

[12] J. Han and Y. Fu, "Dynamic Generation and Refinement of Concept Hierarchy for Knowledge Discovery in Databases", AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94), Seattle, WA, July, 1994, pp. 157-168.

[13] H. Ishikawa, Y. Yamane, Y. Izumida, and N. Kawato, "An Object-Oriented Database System Jasmine : Implementing, Application, and Extension", IEEE Transaction on Knowledge and Data Enegineering, Vol. 8, No. 2, April, 1996, pp. 285-304.

[14] W. Kim, "Object-Oriented Databases : Definition and Research Directions", IEEE Transaction on Knowledge and Data Engineering, 2(3), Sept. 1990, pp 327-341.

[15] W. Kim저, 이 윤준 외 3인 역, "Introduction to Object-Oriented Databases", 하이테크정보, 1994.

[16] ByungS. Lee, "OODB Design with EER", Jour-

nal of Object-Oriented Programming, 9(1), 1996, pp. 61-64.

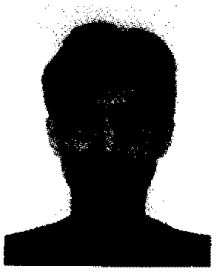
[17] E. A. Rundensteiner, "Object-Oriented View Technology: Challenges and Promises", International Symposium on Cooperative Database System for Advanced Applications, Heian, Schrine, Kyoto, Japan, Dec. 5-7, 1996, (invited paper).

[18] E. A. Rundensteiner, "Tools for View Generation in Object-Oriented Databases", ACM 2nd Int'l Conf. on Information and Knowledge Management(CIKM'93), Nov. 1993, pp. 635-644.

[19] O. R. Zaiane and J. Han, "Resource and Knowledge Discovery in Global Information Systems: A Preliminary Design and Experiment", Proc. 1st Int'l Conf. on Knowledge Discovery and Data Mining (KDD'95), Montreal, Canada, Aug. 1995, pp. 331-336.

신 동 천

1985년 2월 서울대학교 컴퓨터공학과 졸업(학사)
1987년 2월 한국과학기술원 전산학과 졸업(공학석사)
1991년 2월 한국과학기술원 전산학과 졸업(공학박사)
1991년 1월~1993년 2월 한국전산원 선임연구원
1993년 3월~현재 중앙대학교 산업정보학과 조교수, 부교수
관심분야: 다중데이타베이스, 데이터웨어하우스, 전자상거래



김 남 진

1996년 중앙대학교 산업정보학과 졸업(학사)
1996년~현재 중앙대학교 대학원 산업정보학과 석사과정
관심분야: 객체지향 데이터베이스, 데이터베이스 설계, 데이터베이스에서의 지식 추출