교환기용 관계형 데이타베이스 시스템 상에서의 객체지향 인터페이스 제공 기법

정 희 택[†] · 이 길 행[††] · 조 주 현[†††] · 김 용 민[†] ·
이 도 헌[††††] · 노 봉 남[†††††]

요 약

기존의 교환기 시스템은 주로 화일 시스템이나 관계형 데이타베이스 시스템을 이용하여 교환기 운용에 관련된 데이타를 관리해 오고 있다. 하지만, 통신망이 급속히 발전하면서 교환기에 다양한 기능이 요구됨에 따라 관계형 데이타베이스 시스템으로는 효과적인 데이타 관리가 어렵게 되었다. 본 연구는 교환기 시스템을 위한 관계형 데이타베이스 시스템인 DREAM을 확장하여, 보다 강력한 데이타 모델링 기능을 지원하는 객체지향 인터페이스를 제공하는 방안을 제시한다. 아울러, 제시하는 기법을 DREAM 시스템 상에 적용하는 과정을 예시한다.

Providing an Object-Oriented Interface on a Relational Database System for Switching Systems

Hyithaek Ceong[†] · Gilheang Lee[††] · Juhyun Cho[†††] · Yongmin Kim[†] ·
Doheon Lee[††††] · Bongnam Noh[†††††]

ABSTRACT

Conventional switching systems have been using flat file systems or relational database systems to deal with their operational data. However, newly emerged requirements for advanced switching systems make relational database systems no longer proper solutions. This paper defines object-oriented interfaces that effectively incorporate data characteristics of switching systems. In additon, it exemplifies how the method works on an actual database system for the TDX-10 switching system.

1. 서 론

교환기용 데이타베이스 시스템인 DREAM(Distributed REAl-time database Management System)은 교환기 분산 구조의 지원과 실시간 처리를 위한 관계 데이타 베이스 시스템이다. 그러나 기존의 관계 모델을 기반으로 설계 및 구축된 교환기 데이타베이스 시스템은 다음과 같은 문제점을 내포하고 있다.

첫째, 개체를 표현하는 방법과 관련성을 나타내는 방법에 모두 동일한 메카니즘을 사용한다. 즉, 개체와 관련성을 릴레이션을 통해 표현하므로써 각각을 명료하게 표현하지 못하며 개체간에 관련성을 통한 새로운 릴레이션을 유도하기 위해 많은 조인(join)이 필요하다.

※본 논문은 한국전자통신연구소 '95 위탁과제의 지원을 받은 것임
† 정 회 원:전남대학교 전산통계학과
†† 준 회 원:한국전자통신연구소 책임연구원
††† 정 회 원:한국전자통신연구소 책임연구원
†††† 정 회 원:전남대학교 전산학과
††††† 종신회원:전남대학교 전산학과
논문접수:1996년 3월 15일, 심사완료:1996년 10월 16일

둘째, 관계 모델은 현실 세계의 복잡한 의미를 표현하는데 한계가 있다. 즉, 관계 모델은 실세계 개체들을 원자 값(atomic value)에 의해서만 표현하므로써 다양한 형태의 개체를 표현하지 못한다.

셋째, 관계 모델은 확장이 용이하지 못하다. 기존 릴레이션에 대해 새로운 속성을 추가하거나 삭제하여 새로운 릴레이션을 생성하기 위해 추가적인 작업이 필요하고 저장공간을 효율적으로 관리하지 못하므로써 확장에 용이하지 않다.

본 논문에서는 관계 모델이 지닌 문제들을 해결하고 향후 다양한 교환서비스의 요구사항을 수용할 수 있도록 객체지향 개념 실현 방안을 제시하였다. 제시된 방안은 DREAM 시스템을 기반으로한 객체지향 언어의 제공이며, 기반 시스템과 데이타베이스 언어 차이를 해결하기 위한 변환규칙을 제시하므로써 객체지향 개념을 제공하였다. DREAM 시스템을 기반으로 한 변환규칙을 통해 객체지향 개념에 근거한 접근요구를 제공할 뿐만 아니라 기존에 개발된 많은 접근 요구들을 변환과정 없이 허용한다.

본 논문의 구성은 2장에서 기존에 교환기 데이타베이스 시스템으로서 DREAM에 대한 특성을 간단히 기술하며, 3장에서는 교환기 객체지향 데이타베이스 시스템 개발시 고려사항을 보이며, 4장에서는 교환기용 객체지향 데이타베이스 시스템 개발을 위한 실현 방안에 따른 고려사항 및 변환규칙들을 제시한다. 5장에서는 결론과 향후 연구방향을 기술한다.

2. DREAM

분산화 특성과 실시간 특성을 반영한 관계 데이타베이스 시스템으로서 개발된 DREAM은 관계 데이타베이스를 구축하기 위한 과정과 구축된 데이타베이스를 근간으로 분산 및 실시간 특성을 보장하기 위한 데이타베이스 엔진, 그리고 데이타베이스에 존재하는 데이타에 대한 요구를 기술하는 응용 프로그램 영역 등으로 구현되어 현재 TDX-10에서 운영되고 있는 데이타베이스 시스템이다[4, 6, 7, 8].

각 교환국에서는 관계 데이타베이스를 구축하기 위해 DAS(Data Administration System)와 T10DG (TDX-10 Data Generator)를 이용하여 각 교환국에서 수행될 응용 프로그램에 필요한 데이타를 구성한다. 구성될 데이타들은 실시간 요구를 만족시키기 위해 교환기 운용중에는 데이타베이스 구조, 즉 스키마의 변화가 없으며, 데이타베이스의 구성요소인 많은 개체의 값, 즉 인스턴스들이 각 교환국별 및 설치 용량에 따라 초기에 결정된다. 그러나 데이타베이스 구조의 변경을 위해서는 오프라인에서 수행된다. 데이타베이스에 존재하는 데이타에 대한 요구를 기술하는 응용 프로그램 영역에 있어서, 응용 프로그램 작성자가 자신의 응용 프로그램 내에서 데이타베이스 관리 시스템에게 해당 데이타를 요청하는 방법은 CHILL 삽입형 데이타 조작어인 EDML(Embedded Data Manipulation Language)를 사용한다. EDML은 응용 프로그램과 데이타베이스 관리 시스템의 인터페이스로서 응용 프로그램의 작성자가 CHILL에 삽입하여 원하는 데이타를 검색, 변경, 삽입 및 삭제할 수 있는 영어 구문과 유사한 고급 언어이다[2, 9].

관계 데이타베이스 시스템을 기본으로한 교환기들은 통신망의 발달과 멀티미디어를 비롯한 다양한 데이타의 유지 및 관리를 요구하는 응용들의 변화에 대해 적절히 대응치 못하며 시스템의 확장이 용이하지 못하기 때문에 서비스 요구의 다양화에 따라 폭발적으로 증가하는 데이타를 효율적으로 관리하기에는 적합하지 못한 약점들을 안고 있다. 이러한 문제를 해결하므로써 보다 다양하고 복잡한 데이타를 신속하게 처리 관리할 수 있는 시스템으로 활발히 연구되고 있는 것이 객체지향 데이타베이스 시스템의 개발이다.

3. 교환기 객체지향 DBMS 개발시 고려사항

현재 개발 및 운용되고 있는 대부분의 상용 데이타베이스 시스템은 관계 모델을 기반으로 개발된 시스템이다. 대부분의 교환기용 데이타베이스 시스템 또한 관계 모델 형태로 개발하여 운용되고 있다. 이러한 운용환경에서 객체지향 데이타베이스 시스템을 개발하기 위한 방법은 다음 3가지로 고려될 수 있다.

첫째, 객체지향 핵심 모델에 근거하여 개발되는 순수 객체지향 데이타베이스 시스템이다(ObjectStore, VERSANT 등) [12, 18]. 교환기 데이타베이스 시스템으로 객체지향 데이타베이스 시스템을 구축하는 것은 통신망의 발달과 다양한 데이타의 유지 및 관리,

그리고 풍부한 서비스 제공을 위해 관계 데이타베이스 시스템 보다 많은 장점을 지니고 있다[18]. 그러나, 교환기에서 유지해야 하는 데이타의 객체지향 데이타베이스 시스템으로 개발은 많은 부가적 작업 및 비용을 필요로 한다.

둘째, 객체를 지원하기 위해 관계 데이타베이스 시스템을 확장하는 방법은 객체를 유지 관리할 수 있도록 관계 데이타베이스 시스템을 변경한 것이다(UNISQL)[21]. 관계 데이타베이스 시스템의 확장은 새로운 기능을 제공하기 위해 확장하거나 데이타베이스 시스템 구현 블럭들의 라이브러리들로부터 블럭들을 조합하여 데이타베이스 시스템을 구축하는 것이다. 실세계에 하나의 존재를 표현하는 객체를 지원하기 위해 관계 데이타베이스 시스템에서 제공치 못한 사용자 정의 타입을 허용하거나 복잡한 중첩 데이타 모델링 등을 지원한다. 관계 데이타베이스 시스템의 확장은 사용자 정의 타입이나 중첩 데이타 모델링 등의 개념을 위해 응용 프로그램의 확장과 객체를 관리할 수 있는 데이타베이스 관리 시스템, 그리고 객체를 저장할 수 있는 데이타베이스의 확장을 필요로 한다. 이러한 확장방법은 기존의 시스템을 재구성해야 하는 과중한 부하가 존재하며 확장된 관계 데이타베이스 시스템이 개발된 후 운용 및 관리되고 있는 교환기용 관계 데이타베이스 시스템과의 상호 연동을 통한 서비스 제공을 위해 추가적인 시스템의 구축이 필요하다.

셋째, 객체지향 언어와 함께 기존의 데이타베이스 시스템을 사용하는 방법은 객체지향 언어로서 객체에 대한 질의를 제공하고 객체들의 공유성과 지속성의 보장은 이미 운용되고 있는 관계 데이타베이스 시스템에 의해 제공되도록 하는 것이다(IRIS, O_2, 등)[10, 12, 15, 18]. 관계 데이타베이스 시스템에서 데이타베이스를 구성하는 단위는 릴레이션을 기반으로 행과 열로 구성된 테이블의 특성을 갖기 때문에 객체지향의 핵심 모델을 제공할 수 없다. 그러나, 데이타베이스 시스템 사용자에게 객체지향 언어를 통해 객체지향 핵심 모델의 개념들을 제공하므로써 객체에 대한 생성 및 접근, 갱신을 가능케 하고, 그러한 접근을 관계 데이타 모델에서 릴레이션의 행과 열로 변환하므로써 사용자에게 객체지향 장점을 제공할 수 있다.

교환기용 데이타베이스 시스템을 객체지향 기법을 이용하여 개발하는 방법중 가장 현실성 있고 경제적인 방법은, 앞서 언급한 각 방법의 장단점을 고려할 때, 운용되고 있는 관계 데이타베이스 구조에 변경을 최소화하면서 사용자에게 객체지향 핵심 모델을 제공하고 데이타의 공유성과 지속성을 보장하면서 객체지향 핵심 모델을 제공할 수 있는 방법이 고려되어야 한다. 이것은 객체지향 언어를 사용한 접근을 통하여 DREAM에서 유지되고 있는 릴레이션들을 그대로 유지하면서 사용자에게 객체지향 핵심 모델에 기인한 데이타의 접근 및 갱신이 가능하도록 하는 것으로 DREAM을 객체지향 데이타베이스 시스템화하여 달성될 수 있다.

4. 교환기용 객체지향 데이타베이스 시스템 구현방안

객체지향 언어와 함께 관계 데이타베이스 시스템을 사용하는 방법으로 교환기용 데이타베이스 시스템을 구축하는데 있어, 사용자에게 실제 관계 데이타베이스 시스템임에도 객체지향 데이타베이스 시스템 뷰를 제공하기 위해서 고려되어야 할 문제는 다음과 같다.

4.1 객체지향 데이타베이스 언어 제공

어떻게 데이타베이스 언어로서 객체지향 언어를 DREAM 시스템에 제공해야 할 것인가? 객체지향 언어에서 DDL은 객체지향 개념에 근거한 스키마를 정의할 수 있어야 하며 DML은 해당 클래스, 또는 인스턴스에 적당한 메시지를 보내어 개별 인스턴스의 생성, 수정, 삭제가 가능하도록 지원해야 한다. DREAM에서 객체지향 핵심 개념을 지원하는 DDL과 DML를 제공하는 문제를 해결하기 위해 객체지향 DDL을 지원하는 부분과 객체지향 DML을 지원하는 부분으로 나누어 해결할 수 있다.

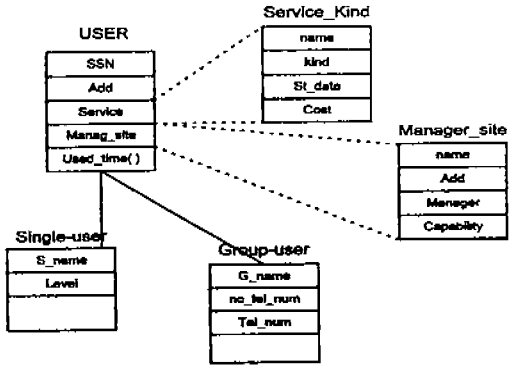
▩교환기용 데이타베이스 시스템에서 객체지향 DDL을 지원하는 방법

- 초기에 교환기용 데이타베이스를 구축할 때 객체지향 DDL을 통해 초기 스키마를 정의할 수 있도록 한다.
- 응용 프로그래머에 의해 프로그램 제작시 발생

하는 데이타를 정의하려 할 때 객체지향 DDL에 의거하여 필요한 클래스 객체를 작성하도록 한다.

데이타 정의어는 객체지향 데이타베이스 스키마를 생성, 삭제, 수정하기 위한 언어이다. 정의어는 개체를 표현하기 위한 클래스의 정의 및 생성과 개체 사이의 관련성을 표현하기 위한 일반화(generalization), 집단화(aggregation) 등을 표현할 수 있어야 한다. 이러한 객체지향 개념에는 클래스를 정의하고 클래스 사이에 존재하는 IS-A 관련성이나 IS-PART-OF 관련성을 표현할 수 있음을 의미한다[5].

다음 그림 1은 데이타 정의어 및 조작어에 대한 간단한 모델로서 전화 사용자 관리를 위한 데이타베이스 스키마를 나타낸 것이며 앞으로 질의어의 정의 및 검색의 대상이 된다.



(그림 1) 전화 사용자 데이타베이스 스키마
(Fig. 1) Telephone user database schema

4.1.1 데이타 정의어

본 연구에서 클래스를 정의 및 구조변경과 삭제하기 위한 데이타 정의어는 다음과 같은 구조를 갖는다.

```
CREATE CLASS class_name
  [AS SUBCLASS OF super_class_name]
  INSTANCE_MAX_NUM instance_max_num,
  PROCESSOR_NAME processor_definition,
  [GLOBAL_PROCESSOR processor_definition,]
  STORAGE_TYPE storage_type_definition,
  LOCATION_TYPE location_type_definition,
  CLASS_TYPE class_type_definition,
  ACCESS_RIGHT
      access_definition [{,access_definition}]
  [(attr_definition [{,att_definition}.)]
  [METHOD
      method_definition [{,method_definition}...] [;]

  ALTER CLASS class_name alter_clause [;]
  DROP CLASS class_spec [{,class_spec}...] [;]
```

각 클래스는 유일한 class_name을 가져야 하며, super_class_name은 일반적으로 사용자가 정의한 클래스를 명시할 수 있다. 만약 클래스 계층 구조상 루트일 경우에는 시스템 정의 'OBJECT'를 명시한다. 그리고, 데이타 정의어 구조 중 DREAM에서 필요한 물리적 정보를 기술하기 위해 INSTANCE_MAX_NUM, PROCESSOR_NAME, GLOBAL_PROCESSOR, STORAGE_TYPE, LOCATION_TYPE, CLASS_TYPE, ACCESS_RIGHT를 정의하였다. METHOD는 해당 클래스의 인스턴스를 생성하거나 삭제, 갱신할 수 있는 프로그램 코드로서 기술되며 C언어에 따른다.

제안된 데이타 정의어는 클래스의 정의 및 클래스 간의 관련성을 표현할 수 있다. 클래스들 간에 존재하는 IS-A 관련성은 super_class_name을 지정하므로써 클래스 사이의 상속계층 관계를 표현한다. 또한 애트리뷰트의 도메인이 다른 클래스가 되도록 허용하여 IS-PART-OF 관계를 표현할 수 있다. 그리고 관계 모델과 달리 애트리뷰트의 도메인이 SET 값을 갖도록 허용한다.

4.1.2 데이타 조작어

제안된 데이타 정의어는 객체지향 데이타베이스 스키마를 생성, 삭제, 수정하기 위해 정의되었으며 정의된 스키마에 대해 해당 클래스의 개별 인스턴스를 생성, 수정, 삭제가 가능하도록 데이타 조작어가 지원되어야 한다.

▩교환기용 데이타베이스 시스템에서 객체지향 DML을 지원하는 방법

- 응용 프로그래머에 의해 응용 프로그램에서 객체지향 DDL에 의해 정의된 객체들에게 메시지

를 보내 인스턴스를 삽입, 삭제, 갱신, 검색하도록 한다.

데이타 조작어는 데이타 정의어에서 정의되는 객체지향 개념을 충분히 반영한 인스턴스의 조작이 가능하여야 한다. 객체지향 데이타를 조작/관리하는 질의어의 표현은 관계 데이타베이스 시스템과 매우 상이하지만 유사한 구문을 사용한다. 임의의 클래스에 대한 인스턴스를 검색·삽입·삭제·갱신하기 위한 구조는 다음과 같다.

```
SELECT attribute_list
FROM class_spec
[WHERE predicates] [;]

INSERT INTO class_name [(attribute_list)]
VALUES (value_list)
[SELECT attribute_list
FROM class_spec
[WHERE predicates]] [;]

DELETE FROM class_spec [correlation]
[WHERE search_condition] [;]

UPDATE class_spec
SET assignment [{, assignment}...]
[WHERE search_condition] [;]
```

하나의 클래스에 대한 질의에서는 attribute_list는 하나 이상의 애트리뷰트 이름을 포함하며 FROM절의 class_name은 데이타베이스에 있는 하나의 클래스 이름을 기술한다. 여기서 attribute_list는 검색의 목적이 되는 것을 기술하는 것이며 class_spec은 특정 클래스와 클래스의 인스턴스를 명세하는 것이고 predicates는 조건절로서 논리연산 조합으로 찾고자 하는 내용에 대한 제한조건을 표시하는 부분이다. 해당 클래스의 인스턴스 객체는 WHERE 절의 조건에 만족하는 애트리뷰트 값에 대한 검색이 이루어진다. 이러한 데이타 조작어는 데이타 정의어에서 제공되는 IS-PART-OF 관련성을 조작하기 위해 질의어에 경로 표현식(path expression)을 표현할 수 있어야 한다. 이러한 경로 표현식은 FROM 절을 제외한 질의 부분에서 표현되어진다. 경로 표현은 "[class_prefix.] attr_name*1* attr_name*2*,..., attr_name*n*"과 같이 점을 이용하여 구분되어진다. 여기서 class_prefix는 데이타베이스에 존재하는 클래스 이름이며 attr_name*i*는 해당 클래스의 애트리뷰트 이름이거나 attr_name*i*에 의해 참조되는 (즉, IS-PART-OF 관련성에 있는) 클래스의 애트리뷰트이다.

4.2 객체지향 모델을 관계 모델로 변환

어떻게 객체지향 핵심 모델에 특성들을 행과 열의 테이블 형태로 구성된 릴레이션으로 변환할 것인가? 변환의 필요성은 객체지향 모델 개념과 관계모델 개념 사이의 존재하는 차이에 기인한다[16, 17]. 변환을 통한 객체지향 개념을 제공하기 위해 객체지향 데이타 정의어와 데이타 조작어를 관계 모델에서 제공하는 데이타베이스 언어로 변환하므로써 달성된다.

4.2.1 객체지향 데이타 정의어를 관계 데이타 정의어로 변환

데이타베이스 스키마 정의 측면에서 볼 때, 관계 모델에서 제공하는 개체에 대한 릴레이션과 개체간의 관련성을 표현하기 위한 기본키 의미를 객체지향 모델에서 제공하는 클래스와 객체 식별자, 복합객체 표현, 그리고 계층 구조로 변환하거나 그 역으로 변환을 통해 사용자에게 객체지향 개념을 제공한다[11, 13, 14, 15, 19]. 이러한 연구들의 대부분은 풍부한 의미적 모델링 개념을 지닌 객체지향 모델을 관계 모델로 변환하는 데 중점을 두었다. 객체지향 모델을 관계 모델로 변환은 클래스를 관계 모델의 릴레이션으로 변환하거나 객체 식별자를 릴레이션의 기본키로 변환하여 생성된 릴레이션이 제 3 정규형을 만족하도록 하였다[14, 19]. 또한 [13]에서는 의미적 정보를 보존하면서 관계 모델에서 객체지향 모델로의 변환 규칙을 제안하였다. 이러한 객체지향 모델에서 관계 모델로 또는 그 역으로의 변환에 관한 연구들은 데이타베이스 스키마 정의 측면에서 일관된 변환 가능성을 제시한다.

객체지향 데이타 정의어에 의해 정의된 스키마를 관계 모델의 스키마로 변환하기 위해서는 관계 모델과 달리 객체지향 개념에 제공되는 개념을 중심으로 관계 모델로 변환하기 위한 과정이 필요하다. 제시되

는 변환 과정은 데이타베이스 스키마 정의를 중심으로 제시하며 DREAM 데이타베이스 스키마 정의에 필요한 물리적 특성 정의 구문은 객체지향 정의어 변환에 중점을 두기 위해 생략하였으며 자세한 변환 결과는 전체적인 변환 예에서 제시한다.

(1)클래스와 객체 식별자

객체지향 모델의 핵심인 클래스는 관계 모델에서 릴레이션으로 변환되며 객체 식별자는 관계 모델에서 기본 키로 변환된다. 또한 해당 클래스의 인스턴스 객체는 릴레이션의 튜플로 변환된다. 즉, 클래스 이름은 릴레이션 이름이 되고 해당 클래스의 인스턴스 객체에 대한 객체 식별자는 릴레이션의 애트리뷰트 OID로서 정의되고 데이타 타입은 char()이나 integer 형이 될 수 있다.

(2)메소드

객체지향 개념과 관계 모델에서 가장 큰 차이를 나타내는 것이 메소드로서, 객체지향 개념에서 메소드는 클래스 정의 내에 포함되어 해당 클래스의 인스턴스 객체를 생성·삭제하는 코드이다. 메소드를 관계 모델로 변환하기 위한 과정에서 메소드는 릴레이션의 애트리뷰트로 변환하지 않고 새로운 메소드 테이블에 해당 정보를 유지하며 메소드 원시 코드는 라이브러리 화일 형태로 재구성한다.

(3)집단화 관련성

클래스의 애트리뷰트 도메인이 클래스인 경우로서, 해당 애트리뷰트는 참조하는 클래스의 단일 인스턴스를 도메인으로 하거나 데이타 정의어 SET OF에 의해 다중 인스턴스를 도메인으로 할 수 있다. 이러한 특성을 반영하여 관계 모델로 변환해야 한다. 그리고, 변환 과정에서 클래스의 집단화 관련성 정보를 스키마 정보 테이블에 유지하므로써 질의 과정에서 사용된다.

①단일 인스턴스를 도메인으로 하는 경우

임의의 클래스의 단일 인스턴스를 도메인으로 하는 애트리뷰트는 클래스들 사이에 일대일의 집단화 관련성을 나타내기 위해 해당 인스턴스의 객체 식별자를 도메인으로 갖는 릴레이션으로 만든다.

②다중 인스턴스를 도메인으로 하는 경우

임의의 클래스의 애트리뷰트가 다중 인스턴스를 도메인으로 하는 경우, 일대다의 관련성을 표현할 수 있는 클래스들 사이에 집단화 관련성을 나타내는 릴레이션을 만든다. 즉, 도메인이 되는 클래스를 릴레이션으로 변환시, 해당 클래스를 도메인으로 갖는 애트리뷰트가 속한 클래스 객체 식별자를 도메인이 되는 클래스의 애트리뷰트로 추가한다.

(4)일반화 관련성

클래스들 사이의 상속 관련성을 나타내는 일반화 관련성은 클래스 계층 구조로 표현되며 관계 모델 릴레이션으로의 변환은 변환 방법에 따라 다음 3가지 방법이 존재한다.

①각 클래스를 하나의 릴레이션으로 변환

클래스 계층 구조에서 일반화 관련성을 나타내는 클래스들 각각을 하나의 릴레이션으로 변환한다. 각 하위 클래스는 상위 클래스의 객체 식별자를 애트리뷰트로 갖는다. 이러한 변환에 의해 얻어진 릴레이션은 제 3 정규형을 만족하나 상속 정보를 나타내지 못하며 임의의 질의에 대한 많은 릴레이션의 접근이 필요하다.

②단말(leaf) 클래스를 하나의 릴레이션으로 변환

각 단말 클래스는 상위 클래스의 애트리뷰트들을 추가하여 하나의 릴레이션으로 변환한다. 이러한 변환은 하위 클래스의 애트리뷰트가 많은 경우 적절하나 상속 구조를 명확히 나타낼 수 없다.

③통합된 하나의 릴레이션으로 변환

클래스 계층 구조상의 상위 및 하위 클래스의 애트리뷰트를 하나의 릴레이션으로 변환한다. 이러한 변환은 하위 클래스의 애트리뷰트가 적은 경우에 적절하나 릴레이션 내에 저장되는 데이타 값들의 많은 부분이 널(null) 값이 된다. 또한 변환된 릴레이션이 제 3 정규형을 만족하지 못한다.

클래스들 사이의 상속 관련성을 나타내는 일반화 관련성을 관계 모델의 릴레이션으로 변환 방법들 중 가장 적절한 변환 방법은 각 클래스를 하나의 릴레이

션으로 변환하는 것이다. 각 클래스를 하나의 릴레이션으로 변환하므로서 관계 모델에서 갖는 이상(anomaly)을 최소화한다. 또한, 질의를 수행하는 과정에서 많은 릴레이션을 검색해야 하는 과부하는 오프라인에서 데이타베이스 스키마를 구성하는 교환기 데이타베이스 시스템 특성에 의해 최소화된다. 변환 과정에서 클래스의 일반화 관련성 정보를 스키마 정보 테이블에 유지하므로써 질의 과정에서 사용된다.

객체지향 모델에서 제공하는 개념들을 관계 모델로 변환하기 위해, 클래스와 객체 식별자, 메소드, 집단화 관련성, 그리고 일반화 관련성을 고려한 변환 규칙은 다음과 같다.

■객체지향 스키마에서 관계 스키마로 변환 규칙

(1)클래스 C의 이름은 릴레이션 R의 이름으로 변환한다.

(2)클래스 C의 상위 클래스가 존재하면 일반화 관련성 변환을 수행한다.

(3)클래스의 인스턴스를 위해 릴레이션 R에서 기본키가 되는 OID 애트리뷰트로 변환한다.

(4)클래스 C의 애트리뷰트 A1,...,An은 릴레이션 R의 애트리뷰트들로 변환된다.

(5)클래스 C의 애트리뷰트 도메인이 클래스 C'인 경우 집단화 관련성 변환을 수행한다.

(6)클래스 C의 메소드는 메소드 변환을 수행한다.

■일반화 관련성 변환

(1)상위 클래스의 OID를 갖는 애트리뷰트를 생성한다.

■집단화 관련성 변환

(1)단일 인스턴스를 도메인으로 하는 경우-예약어 SET OF가 없는 경우-

①클래스 C의 해당 애트리뷰트의 도메인을 클래스 C'에 인스턴스의 OID를 나타내는 integer형으로 변환한다.

(2)다중 인스턴스를 도메인으로 하는 경우-예약어 SET OF가 있는 경우-

①클래스 C의 해당 애트리뷰트의 도메인을 클래스 C'의 OID를 나타내는 integer형으로 변환한다.

②클래스 C'는 클래스 C의 OID를 갖는 애트리뷰트를 추가한다.

■메소드 변환

(1)메소드를 이루는 코드는 라이브러리 형태의 화일로 재구성한다.

(2)메소드 릴레이션에 메소드 이름, 소속 클래스 이름, 매개변수 이름, 반환 값, 메소드 코드 화일 이름을 저장한다.

4.2.2 스키마 정보 테이블

클래스들이 갖는 정의 정보와 계층 정보를 변환 과정을 통해 유실되지 않도록 관리하고 질의 과정에서 효율적인 처리를 위해 다음과 같은 테이블들이 변환 과정에서 생성, 유지된다.

(1)애트리뷰트 테이블

애트리뷰트의 도메인이 클래스인 경우 질의 변환 과정에서 필요한 정보들을 유지한다. 구성하는 필드는 애트리뷰트 이름, 애트리뷰트 타입, SET 유무, 도메인이 되는 클래스 이름, 소속 클래스 이름으로 구성된다.

(2)메소드 테이블

메소드 테이블에는 메소드 이름, 소속 클래스 이름, 매개변수 이름, 반환 값(return value), 메소드 코드 화일 이름 등으로 구성된다.

(3)일반화 관련성 테이블

클래스들 사이에 상속계층 정보를 유지하는 테이블로서 클래스 이름, 클래스 객체 식별자, 상위 클래스 이름으로 구성된다.

4.2.3 객체지향 데이타 조작어를 관계 데이타 조작어로 변환

객체지향 개념을 제공하는 객체지향 DML을 관계 모델에 근거한 DML로 변환하여 사용자에게 객체지향 개념을 제공하는 것이다[1, 5, 20]. 이러한 연구는 응용 프로그래머에 의해 접근되는 객체에 대한 질의어를 변환 과정을 통해 관계 모델의 질의어로 변환함으로 데이타에 접근한다.

객체지향 데이타 조작어에 의해 접근하는 질의어를 관계 모델에서의 데이타 조작어로 변환하기 위해, 클래스, 메소드, 집단화 관련성, 그리고 일반화 관련성 각각에 대한 질의어를 릴레이션에 대한 질의어로 변환한다. 변환은 표준 SQL 문장을 목표로 수행된다.

(1)검색어 변환

제안된 검색어 구조는 다음과 같고, 관계 질의어로 변환하기 위해서 attribute_list 변환, class_name 변환, predicates 변환 각각이 이루어져야 한다.

▩attribute_list 변환

'SELECT' 절에 의해 표현되는 검색 대상의 객체 변수는 'FROM' 절에서 기술된 해당 클래스에 대한 객체 변수로서 변환 과정 없이 릴레이션 변수로 사용된다. 애트리뷰트의 도메인이 단순 자료형인 경우 해당 릴레이션의 애트리뷰트 이름으로 변환 없이 사용할 수 있으나, 클래스인 경우 변환 과정이 필요하다. 애트리뷰트의 도메인이 클래스인 경우, 단일 인스턴스를 도메인으로 갖는 경우나 다중 인스턴스를 도메인으로 갖는 경우에 대해 스키마 정보 테이블 중 애트리뷰트 테이블을 참조하여 도메인이 되는 클래스 이름을 'FROM' 절에 해당 릴레이션 변수와 함께 추가한다. 클래스를 도메인으로 갖는 애트리뷰트는 도메인이 되는 클래스의 애트리뷰트들로 대치된다. 그리고, 변환된 attribute_list에 단순 자료형을 갖지 않는 애트리뷰트에 대해서는 이러한 과정을 반복한다.

▩class_name 변환

'FROM' 절에서 기술된 클래스 이름 및 객체 변수는 추가됨이 없이 클래스 이름과 릴레이션 변수로 사용되며 attribute_list 변환 과정에서 추가되는 클래스 이름이 추가 기술된다. 그러나, 클래스 계층 구조상 임의의 클래스에 대한 하위 클래스들도 검색 대상으로 포함시키는 'ALL' 예약어에 대해 앞서 제시된 일반화 관련성 변환에 따른 class_name변환이 필요하다. 즉, 일반화 관련성 변환에서 각 클래스를 하나의 릴레이션으로 변환하였기 때문에 스키마 정보 테이블 중 일반화 관련성 테이블의 상위 클래스 이름이 class_name에 기술된 것과 일치하는 모든 클래스 이름을 추가해야 한다.

▩predicates 변환

'WHERE' 절에 기술된 조건절에서 단순 자료형을 기술한 애트리뷰트의 조건절은 변환 과정을 거치지 않고 기술된다. 그러나 경로 표현식에 의해 집단화 관련성을 나타내는 애트리뷰트에 대해서는 애트리뷰트 테이블을 참조하여 도메인이 되는 클래스 이름을 'FROM' 절에 해당 릴레이션 변수와 함께 추가한다.

경로 표현식에서 클래스를 도메인으로 갖는 애트리뷰트가 기술된 조건식은 단일 인스턴스를 도메인으로 하는 경우 해당 애트리뷰트와 도메인이 되는 클래스의 객체 식별자에 대한 동등 조건식을 추가하고 'AND'를 추가한 후, 최종 목적 애트리뷰트는 도메인이 되는 클래스 애트리뷰트에 대한 조건식으로 변환한다. 다중 인스턴스를 도메인으로 하는 경우 클래스를 도메인으로 하는 애트리뷰트의 소속 클래스의 객체 식별자와 도메인이 되는 클래스에 존재하여 관련성을 나타내는 객체 식별자와 동등 조건식은 삽입하고, AND를 삽입한 후, 최종 목적 애트리뷰트는 도메인이 되는 클래스 애트리뷰트에 대한 조건식으로 변환한다. 변환 과정을 보이면 다음과 같다.

①단일 인스턴스를 도메인으로 하는 경우 변환 예

```
SELECT   U.name
FROM     USER   U
WHERE        U.SSN = '700208-1559812' AND
U.Manag_site. Manager = '홍길동';
                ⇩
SELECT   U.name
FROM     USER U, Manager_site M
WHERE        U.SSN = '700208-1559812' AND
U.Manag_site = M.OID AND M.Manager = '홍길동';
```

②다중 인스턴스를 도메인으로 하는 경우 변환 예

```
SELECT   U.name
FROM     USER   U
WHERE    U.SSN = '700208-1559812' AND
U.Service.name = 'CTI';
                ⇩
SELECT   U.name
FROM     USER U, Service_Kind S
WHERE    U.SSN = '700208-1559812' AND U.OID =
S.USER_OID AND S.name = 'CTI';
```

(2)조작어 변환

본 연구에서 데이타 조작어란 데이타 검색어를 제외한 데이타 조작을 의미하는 것으로 앞서 정의된 INSERT, DELETE, UPDATE가 있으나 변환 과정이 유사하기 때문에 INSERT에 대한 변환을 보인다.

먼저, 클래스에 해당 인스턴스를 삽입하는 데이타 조작문은 객체 식별자에 대한 값을 삽입 조작문에서는 제공하지 않으나 변환 과정에서 제공한다. 그리고, 인스턴스 삽입을 위한 데이타 조작문을 관계 모델에서의 데이타 조작문으로 변환하기 위해서 집단화 관련성을 표현하는 attribute_list에 대한 변환 과정이 필요하다. attribute_list에서 집단화 관련성을 나타내는 애트리뷰트에 대해서 단일 인스턴스를 도메인으로 하는 경우는 애트리뷰트 테이블을 참조하여 도메인이 되는 클래스의 인스턴스 객체 식별자를 할당한다. 다중 인스턴스를 도메인으로 하는 경우는 애트리뷰트 테이블을 참조하여 도메인이 되는 클래스에서 관련성을 나타내는 애트리뷰트에 해당 객체 식별자를 삽입하므로써 변환된다. 제시된 예에서 각 클래스의 GEN_OID(클래스 이름)은 변환 과정에서 객체 식별자 생성기에 의해 제공된다.

```
INSERT INTO USER (SSN, Add, Service, Manag_site)
VALUES ('720128-1587292', '경기도 안산시 가내동 12',
 INSERT INTO Service_Kind(name, kind, St_date, Cost)
 VALUE ('CTI', '001', '12/25/1995', 7000),
 INSERT INTO Manager_site(name, Add, Manager,
 Capability)
 VALUE('안산교환국', '경기도 안산시 가내동', '홍길동',
 90000))
                ⇩
$tmp1 = GEN_OID(USER)
$tmp2 = GEN_OID(Manager_site)
INSERT INTO Manager_site(OID, name, Add, Manager,
                Capability),
VALUES(tmp2, '안산교환국', '경기도 안산시 가내동',
                '홍길동', 90000);
INSERT INTO USER (OID, SSN, Add, Service,
                Manag_site)
VALUES (tmp1, '720128-1587292', '경기도 안산시 가내
                동 12', Null, tmp2);
INSERT INTO Service_Kind(OID, name, kind, St_date,
                Cost, USER_OID)
VALUE(GEN_OID(Service_Kind), 'CTI', '001', '12/25/
                1995', 7000, tmp1),
```

4.3 DREAM 환경에서 변환 예

제안된 객체지향 데이타베이스 구축 방안에 따라 객체지향 언어를 제공하기 위해 객체지향 데이타 s정의어와 조작어를 제안하였다. 제안된 정의어 및 조작어를 DREAM 환경에서 운용하는 예를 보이기 위해 그림 1에서 제시된 전화 사용자 데이타베이스 스키마 중 USER 클래스의 물리적 특성을 포함한 정의를 보이며, 이를 DREAM에서 대응되는 화일(EDIF로 기술된 "블럭.db")로 변환하는 과정을 기술한다. 다음으로, 객체지향 조작어로 작성된 응용 프로그램을 DREAM에서의 응용 프로그램으로 변환을 기술한다. 먼저, 물리적 특성을 포함한 전화 사용자 데이타베이스 스키마 정의는 제안된 객체지향 데이타 정의어 구조에 따라 다음과 같다.

```
CREATE CLASS Service_Kind as subclass of object
  INSTANCE_MAX_NUM 2000,
  PROCESSOR_NAME OMP,
  STORAGE_TYPE DKB,
  LOCATION_TYPE NOR,
  CLASS_TYPE CDE,
  ACCESS_RIGHT SELECT, UPDATE, INSERT,
                DELETE, LOCK, UNLOCK
  name char(30),
  kind char(3),
  St_date date,
  Cost Integer;

CREATE CLASS Manager_site as subclass of object
  INSTANCE_MAX_NUM 2000,
  PROCESSOR_NAME OMP,
  STORAGE_TYPE DKB,
  LOCATION_TYPE NOR,
  CLASS_TYPE CDE,
  ACCESS_RIGHT SELECT, UPDATE, INSERT,
                DELETE, LOCK, UNLOCK
  name char(30),
```

```
  Add char(40),
  Manager char(20),
  Capability char(10);

CREATE CLASS USER as subclass of object
  INSTANCE_MAX_NUM 5000,
  PROCESSOR_NAME OMP,
  STORAGE_TYPE MEM,
  LOCATION_TYPE NOR,
  CLASS_TYPE CDE,
  ACCESS_RIGHT SELECT, UPDATE, INSERT,
               DELETE, AVG, COUNT, MAXI,
               SUM, LOCK, UNLOCK
               SSN char(14),
  Add char(40),
  Service SET OF Service_kind,
  Manag_site Manager_site,
  METHOD used_time(char(14)) int;

CREATE CLASS Single-user as subclass of USER
  INSTANCE_MAX_NUM 1000,
  PROCESSOR_NAME OMP,
  STORAGE_TYPE MEM,
  LOCATION_TYPE NOR,
  CLASS_TYPE CDE,
  ACCESS_RIGHT SELECT, UPDATE, INSERT,
               DELETE, LOCK, UNLOCK
  S_name char(20)
  Level integer;

CREATE CLASS Group_user as subclass of USER
  INSTANCE_MAX_NUM 1000,
  PROCESSOR_NAME OMP,
  STORAGE_TYPE MEM,
  LOCATION_TYPE NOR,
  CLASS_TYPE CDE,
  ACCESS_RIGHT SELECT, UPDATE, INSERT,
               DELETE, LOCK, UNLOCK
  G_name char(20),
  no_tel_num integer,
  Tel_num set of Tel_num;
```

변환 과정에서 해당 물리적 특성은 데이타 입력화일의 해당 변수로 변환되고 객체지향 개념들의 변환은 앞서 제시된 변환 규칙에 의해 변환된다. 즉, 물리적 특성 변환 및 애트리뷰트 변환은 다음과 같다.

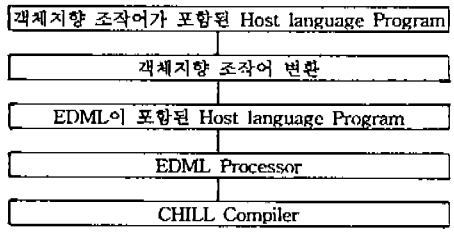
```
INSTANCE_MAX_NUM→.t PROCESSOR_NAME→.l
GLOBAL_PROCESSOR→.g STORAGE_TYPE→.m
LOCATION_TYPE    →.e CLASS_TYPE   →.o
ACCESS_RIGHT     →.a attribute_definition→.d
```

이러한 일대일 변환에 대해 DREAM 시스템에서 사용되는 EDIF로 기술된 데이타 입력 화일로 변환되어 다음과 같은 user.db화일이 된다.

```
#
.r  Service_Kind
.t  2000
.l  OMP
.m  DKB
.e  NOR
.o  CDE
.a  SELECT |UPDATE|INSERT|DELETE
           |LOCK|UNLOCK
.k  OID int
.d  name char(30)
.d  kind char(3)
.d  St_date date
.d  Cost int
.d  USER_OID int
#
.r  Manager_site
.t  2000
.l  OMP
.m  DBK
.e  NOR
.o  CDE
.a  SELECT |UPDATE|INSERT|DELETE
           |LOCK|UNLOCK
.k  OID int
.d  name char(30)
.d  Add char(40)
```

```
d  Manager char(20)
d  Capability char(10)
#
.r  USER
.t  5000
.l  OMP
.m  MEM
.e  NOR
.o  CDE
.a  SELECT |UPDATE|INSERT|DELETE
           |AVG|COUNT|MAXI|SUM
           |LOCK|UNLOCK
.k  OID int
.d  SSN char(14)
.d  Add char(40)
.d  Manag_site int
.d  Service int
#
.r  Single_user
.t  1000
.l  OMP
.m  MEM
.e  NOR
.o  CDE
.a  SELECT |UPDATE|INSERT|DELETE
           |LOCK|UNLOCK
.k  USER_OID int
.d  S_name char(20),
.d  Level int
#
.r  Group_user
.t  1000
.l  OMP
.m  MEM
.e  NOR
.o  CDE
.a  SELECT |UPDATE|INSERT|DELETE
           |LOCK|UNLOCK
.k  USER_OID int
.d  G_name char(20),
.d  no_tel_num int
.d  Tel_num int
```

제안된 객체지향 조작어를 이용한 데이타베이스에 저장된 데이타를 검색하여 처리하는 과정을 보이기 위해 객체지향 조작어로 작성된 응용 프로그램을 DREAM에서의 응용 프로그램으로 변환과정을 기술한다. 변환 과정은 기존에 EDML로 작성된 응용 프로그램 처리 과정에 제안된 객체지향 변환 과정을 추가하여 완성되며 그림 2와 같다.



(그림 2) 응용 프로그램 처리 단계
(Fig. 2) Application program processing steps

앞서 제시된 전화기 사용자 데이타베이스에 대한 객체지향 조작어로 작성된 응용 프로그램의 간단한 예는 다음과 같다. 응용 프로그램상에 기본 구조는 DREAM에서 응용 프로그램 작성구조와 동일하다. 그러나, 'DBMS_METHOD'는 스키마 변환 과정에서 생성된 메소드 코드 화일이다.

```
SPEC MODULE REMOTE DBMS_LIB;
SPEC MODULE REMOTE DBMS_MODE;
SPEC MODULE REMOTE DBMS_METHOD;
$ INCLUDE{USER};
$ SELECT USER WHERE {SSN='700208-1559812',
                     Manager_site.Manager='홍길동'};
if DB_DB_INFO.ACC_STA=DB_SUCESS
then
                        ...
$ EXCLUDE
```

객체지향 개념이 제공되는 이러한 응용 프로그램은 제안된 변환 규칙에 의해 EDML이 포함된 응용 프로그램은 다음과 같다.

```
SPEC MODULE REMOTE DBMS_LIB;
SPEC MODULE REMOTE DBMS_MODE;
SPEC MODULE REMOTE DBMS_METHOD;
$ INCLUDE{ USER, Manager_site};
                    ...
$ SELECT USER U, Manager_site M
  WHERE{U.SSN='700208-1559812',
  U.Manag_site=M.OID, M.Manager='홍길동'};
  if DB_DB_INFO.ACC_STA=DB_SUCESS
  then
                    ...
  $ EXCLUDE
```

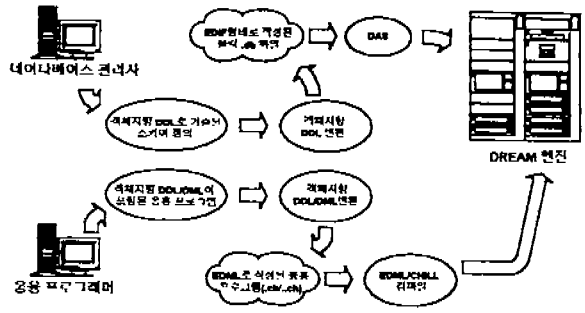
예로 제시된, 응용 프로그램의 객체지향 조작어가 앞서 정의된 객체지향 정의어와 구문상에 차이가 존재하나 응용 프로그램 코딩을 위한 표현상에 차이일 뿐 변환 과정에 있어 영향받지 않는다.

4.4 객체지향 개념을 제공하기 위한 인터페이스 구조

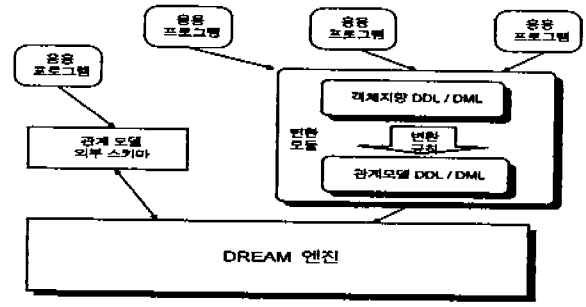
마지막으로 고려해야 할 사항은 관계 데이타베이스 시스템인 DREAM 기반 위에 앞서 제시한 두 가지 문제를 해결하는 교환기용 데이타베이스 시스템을 어떻게 구축할 것인가라는 문제다. 즉, 앞서 제시한 문제들을 해결하는 객체지향 개념을 DREAM의 어떤 요소에서 제공하므로써 사용자에게 객체지향 장점을 제공할 수 있는가 하는 방법론에 대한 제시이다.

데이타의 공용을 전제로 하는 데이타베이스 환경에서 개개 사용자가 접근하는 데이타는 사실상 전체 데이타베이스의 일부가 되고 실제 저장되어 있는 물리적 데이타베이스의 일부분이 되는 것이다, 이러한 특성에 의해 앞서 제시된 규칙들에 의해 데이타베이스 언어로 객체지향 언어를 제공하고 객체지향 개념들을 관계 모델 개념으로의 변환은 외부와 개념 스키마간에 사상이 이루어지는 응용 인터페이스에서 지원하도록 한다. 이는 다음 그림 3과 같은 과정을 통해 DREAM에서 필요한 형태로 변환함을 의미한다.

객체지향 개념과 관계 모델사이의 변환이 응용 인터페이스에서 달성되도록 하기 위해, DREAM에서 응용 프로그램 개발시 객체지향 개념을 지원하도록 응용 프로그램을 제안된 객체지향 데이타베이스 언어로 작성하고 데이타베이스 구축시에 객체지향 데이타 정의어에 의한 데이타베이스를 구축하므로써 달성한다. 이러한 방법은 사용자에게 객체지향 개념에 따른 데이타 접근과 관계 모델에 근거한 데이타 접근을 제공할 수 있다. 객체지향 개념에 따른 데이타 접근과 관계 모델에 근거한 데이타 접근을 제공할 수 있는 인터페이스 모듈은 그림 4과 같다.



(그림 3) 객체지향 개념을 제공하기 위한 변환 구조
(Fig. 3) The transform structure to support object-oriented concepts



(그림 4) 객체지향 개념을 제공하는 DREAM의 인터페이스 구조
(Fig. 4) The interface structure of DREAM to support object-oriented concepts

5. 결론 및 향후 연구 방향

교환기용 데이타베이스 시스템으로서 객체지향 데이타베이스 시스템을 실현하기 위해, 교환기용 데이타베이스 시스템으로서 개발된 DREAM의 구조와 기능을 분석하였다. 또한, 교환기용 데이타베이스 시스템의 구현 방안으로 객체지향 개념을 제공하기 위

해 객체지향 언어를 사용하고 관계 데이타베이스 시스템인 DREAM에 적용하는 방안을 제시하였다.

제시된 방안에 따라 객체지향 언어를 제공하기 위해 객체지향 데이타베이스 언어를 정의하였다. 제안된 객체지향 정의어 및 조작어는 교환기 데이타베이스 시스템의 물리적 특성을 반영한 구조이며, 집단화 관련성을 기술하도록 허용하므로써 개체와 관련성을 명확히 표현할 수 있고 다양한 형태의 개체를 표현할 수 있으며, 그리고 일반화 관련성을 기술할 수 있도록 허용하므로써 데이타베이스의 확장이 용이하다. 기반 시스템과 데이타베이스 언어 차이를 해결하기 위해 객체지향 데이타 정의어와 조작어를 관계 데이타베이스 언어로 변환하는 변환규칙을 제안하였다. 제안된 변환 규칙은 객체지향 정의어에 의해 정의된 객체지향 데이타베이스 스키마를 관계 데이타베이스 스키마로 변환하는 규칙과 객체에 대한 조작어를 관계 데이타 조작어로 변환할 수 있는 변환 규칙으로 구성된다. 또한, 객체지향 모델을 관계 모델로 변환하는 과정에서의 정보의 유실을 막기 위해 스키마 정보 테이블과 객체 식별자 생성기를 간략히 제시하였다. 제안된 변환 규칙에 의해 데이타를 관리하는 시스템이 관계 데이타베이스 시스템임에도 불구하고 객체지향 개념을 제공한다.

제안된 객체지향 데이타베이스 언어로 기술된 정의어와 질의어를 DREAM에서 사용되는 표현 형태(블럭. db)와 EDML이 포함된 응용 프로그램으로 변환을 제시하였으며, 데이타베이스 언어의 변환 규칙을 기반으로 객체지향 개념을 제공하는 시스템을 기존 교환기용 데이타베이스인 DREAM에서 실현하기 위해 객체지향 데이타베이스 언어 변환 인터페이스를 제안하였다.

향후 연구에서, 앞서 정의된 질의어는 객체지향 모델의 기본 개념만을 지원하나 다중상속과 같은 보다 풍부한 기능을 제공하기 위해 정의된 질의어를 개선할 예정이다. 명령어 형태로 정의된 객체지향 데이타베이스 언어를 데이타의 등록 및 삭제 그리고 검색 및 변경을 신속, 정확하고 손쉽게 수행할 수 있는 GUI(graphic user interface)형태의 질의어 처리기 구현에 대한 연구를 수행할 예정이다. 그리고 제안된 객체지향 지원 방안이 분산된 구조에서 고려해야 할 사항들을 분석하고 확장할 것이며 실시간 특성에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] 강은영, 김경창, "관계형 데이타베이스 엔진을 기반으로 한 객체지향 질의어 설계 및 변환", 한국정보과학회 추계학술발표 논문집, Vol. 21, No. 2, pp. 89-92, 1994.

[2] 김응화, 장시웅, 임진희, 박유미, 우왕돈, 조주현, "TDX-10 DBMS에서의 대화형 질의 처리", '93 동계 데이타베이스 학술대회 논문집, Vol. 9, No. 1, pp. 97-102, 1993.

[3] 김원, "객체지향 데이타베이스", 하이테크, 1994.

[4] 박유미, 이길행, 우왕돈, 조주현, "TDX-10 DBMS의 스키마 관리의 설계 및 구현", 한국정보과학회 추계학술발표 논문집, Vol. 20, No. 2, pp. 7-10, 1993.

[5] 신판섭, 안우영, 고병오, 김경창, 임해철, "관계형 데이타베이스 엔진을 기반으로 한 객체지향 데이타베이스 시스템 구현", 한국정보과학회 추계학술발표 논문집, Vol. 20, No. 2, pp. 77-80, 1993.

[6] 이길행, 우왕돈, 장지원, 정석용, "CHILL Embedded DML 처리기의 설계 및 구현", 한국정보과학회 춘계학술발표 논문집, Vol. 16, No. 1, pp. 144-147, 1989.

[7] 이길행, 장지원, 전학성, 김영시, "교환기 데이타베이스 관리 시스템의 설계에서 실시간 처리를 지원하기 의한 고려사항", 한국정보과학회 춘계학술발표 논문집, Vol. 15, No. 1, pp. 157-160, 1988.

[8] 이길행, 전학성, 우왕돈, 김영시, "교환기 데이타 관리 시스템(EDMS) 설계에 관한 연구", 한국정보과학회 춘계학술발표 논문집, Vol. 14, No. 1, pp. 63-65, 1987.

[9] 현온희, 우완돈, 박우구, 홍선미, 최염규, "TDX-10 데이타베이스를 위한 DDL 처리기의 설계 및 구현", 한국정보과학회 춘계학술발표 논문집, Vol. 16, No. 1, pp. 159-162, 1989.

[10] Bharat Bhargava and Shalab Goel, "A Layered Architecture for Supporting Objects in Relational System : A Performance Study", Database & Expert

Systems Applications(DEXA 95), pp. 323-333, 1995.

[11] Christian Fahrner and Gottfried Vossen, "A survey of database design transformations based on the Entity-Relationship model", Data & Knowledge Engineering, Vol. 15, pp. 213-250, 1995.

[12] Elisa Bertino and Lorenzo Martino, "Object-Oriented Database Management Systems: Concepts and Issues", IEEE Computer, Vol. 24, No. 4, pp. 33-47, April 1991.

[13] Ge Yu, Guoren Wang, Huaiyuan Zheng and Akifumi Makinouchi, "Transform More Semantics from Relational Databases into Object-Oriented Database", Proc. of the fourth Intern. Conf, on Database Systems for Advanced Applications, pp. 300-307, April 1995.

[14] James Rumbaugh,Michael Blaha, William Premerian, Frederick Eddy and William Lorensen, "Object-Oriented Modelling and Design", Prentice Hall, pp. 366-396, 1991.

[15] John A. Campbell and V. John Joseph, "The Object-oriented Design and Implementation of a relational database management System", Journal of Object-Oriented Programming, Vol. 8, No. 4, pp. 43-47, August 1995.

[16] Joshua Duhl and Craig Damon, "A Performance Comparision of Object and Relational Databases Using the SUN Benchmark", OOPSLA '88 Proce. pp. 153-163, Sept. 1988.

[17] Karen E. Smith and Stanley B. Zdonik, "Intermedia: A case Study of the Differences Between Relational and Object-Oriented Database Systems", Conf. Proce. OOPSLA '87, pp. 452-465, Oct. 1987.

[18] Mansour Zand, Val Collins and Dale Caviness, "A Survey of Currenct Object-Oriented Database", DATA BASE Advances, Vol. 26, No. 1, pp. 14- 29, 1995.

[19] Michael R. Blaha, William J. Premerlani and James E. Rumbaugh, "Relational Database Design Using An Object-Oriented Methodology", CACM, Vol. 31, No. 4, April, pp. 414-427,1988.

[20] William J. Premerian, Michael R. Blaha, James E. Rumbaugh and Thomas A. Varwig, "An Object-Oriented Relational Database", CACM, Vol. 33, No. 11, pp. 99-109, Nov. 1990.

[21] Kim Won, "UniSQL 소개서", UniSQL, 1995. 4.

정 희 택

1992년 전남대학교 전산통계학과(학사)
1995년 전남대학교 대학원 전산통계학과(석사)
1995년~현재 전남대학교 대학원 전산통계학과 박사과정
관심분야:데이타 웨아하우징, 데이타 마이닝

이 길 행

1984년 전남대학교 계산통계학과(학사)
1986년 한국과학기술원 전산학과(석사)
1996년 한국과학기술원 전산학과(박사)
1988년~1989년 충남대학교 시간강사
1986년~현재 한국전자통신연구소 책임연구원
관심분야:분산처리, 실시간 데이타베이스, 고장허용 구조,객체지향 데이타베이스, 분산 시스템, 음성인식, 통신망 관리

조 주 현

1978년 서울대학교 전자과(학사)
1984년 한국과학기술원 전산학과(석사)
1978년~현재 한국전자통신연구소 책임연구원
관심분야:실시간 시스템, 고장허용구조, 객체지향 프로그래밍

김 용 민

1989년 전남대학교 전산통계학과(학사)
1991년 전남대학교 대학원 전산통계학과(석사)
1996년~현재 전남대학교 대학원 전산통계학과 박사과정
관심분야:통신망 관리 및 보안, 망 서비스 프로토콜

이 도 헌

1990년 한국과학기술원 전산학과(학사)
1992년 한국과학기술원 전산학과(석사)
1995년 한국과학기술원 전산학과(박사)
1996년~현재 전남대학교 전산학과 전임강사
관심분야:데이타 마이닝, 데이타 웨어하우징, 퍼지 데이타베이스

노 봉 남

1978년 전남대학교 수학교육과 졸업(학사)
1982년 한국과학기술원 전산학과(공학석사)
1994년 전북대학교 대학원 전산통계학과(이학박사)
1983년~현재 전남대학교 전산학과 교수
관심분야:객체지향 시스템, 통신망관리, 정보 보안, 컴퓨터와 정보사회 등