論文 96-21-2-15

통신망 관리를 위한 능동 객체 지향 디렉토리 데이타베이스 모델

正會員 李 載 昊*, 林 海 喆**

An Active Object-Oriented Directory Database Model for Management of Telecommunication

Jae-Ho Lee*, Hae-Chull Lim** *Regular Members*

要 約

본 논문에서는 통신망 환경에서 분산 정보 저장소 역할을 수행하는 디렉토리 시스템의 데이타베이스를 모델링 하였으며, 모델링 단계는 다음과 같이 4가지 단계로 나누어 진행하였다.

(1)디렉토리 데이타베이스 저장 정보를 사용자 정보, 운영 정보, 지식 정보, 추가 정보 등으로 분류 하였다. (2) 분류 정보 모델링시에 적용할 몇가지 기준을 개발하였다. (3)모델링 기준을 적용하여 각 분류 정보별로 객체지향 모델링을 하였다. (4)개발 모델에 적용되는 메소드를 분류하고 그 중 일부에 대한 제약조건과 트리거 개념을 적용하여 능동 기반 모델링을 하였다. 이와같이 개발된 제약조건과 트리거 메소드는 각 정보의 속성값과 캡슐화되어 저장되므로써, 능동 객체 지향 디렉토리 데이타베이스 모델을 형성한다.

Abstract

In this paper, we present database model of directory systems that perform a task for distributed information repositories in communication network environments. A new model is developed through four phase:

(1)A directory database information is classified that would be stored in directory database as user, administrative, and supplementary information. (2)The modeling criteria are captured that would be used to model information classified. (3)Object-Oriented concepts are used in modeling classified information according to modeling criteria captured. (4)Methods applied to developed model are grouped, and Active-Based mechanisms such as trigger and

* 한국전자통신연구소
Electronics and Telecommunications Research Institute
** 홍익대학교 컴퓨터공학과
Dept. of Computer Engineering, Honglk Univ.

論文番號:95345-1006
接受日字:1995년 10월 6일

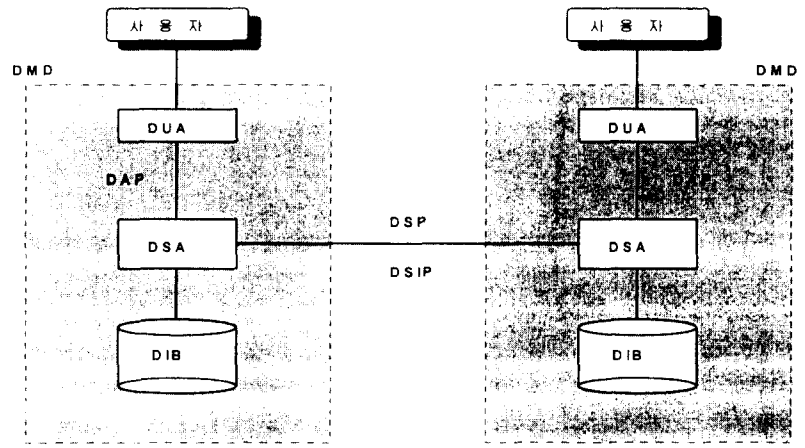
constraints are developed. These selected methods and attributes are encapsulated into objects. Consequently they compose an Active Object-Oriented Directory Database Model.

I. 서 론

현대의 통신망은 이전에 경험하지 못한 빠른 진화를 거듭하고 있다. 또한, 현대 통신망은 엄청난 수의 노드가 전세계적으로 분산되어 있으며, 이러한 노드를 구성하는 구성 요소인 H/W, S/W, 응용 시스템 등은 이질적인 성격을 가지며 복잡하게 구성된다. 반면, 통신망이 복잡해지고 커지면서 관심대상에 대한 정보양은 기하급수적으로 증가하고 있어, 대량의 분산 데이타를 효과적으로 관리할 수 있는 "분산 정보 저장소"의 필요성이 대두되고 있다. 이와같은 요구사항에 부응할 수 있는 후보 기술 중 하나가 ITU의 X.500 디렉토리 시스템[16]이다.

X.500 디렉토리 시스템은 범용의 데이타베이스 시스템과는 몇가지 측면에서 차이점이 있는 특수용도의 데이타베이스 시스템이라 간주할 수 있다. 첫번째 차이점은 디렉토리 데이타베이스내에 저장되는 정보 특성이다. 즉, 저장 정보 내용이 빈번히 변경되는 동적 (dynamic) 데이타 보다는 내용 변경이 빈번하지 않는 정적 (static) 또는 준동적 (semi-dynamic) 데이타를 저장한다. 두번째 차이점은 디렉토리 데이타베이스내의 저장 정보 특성으로 인하여 동작 특성 또한 구분된다. 즉, 일반적인 데이타베이스 시스템의 갱신 (update) 동작보다는 검색 (retrieval) 동작 위주로 운영된다. 마지막 차이점은 분산 데이타들 간의 완벽한 전역 일치 (global commitment)를 요구하지 않으므로, 일시적인 불일치 (transient inconsistency)를 허용하는 시스템이다.

X.500 디렉토리 시스템은 그림 1과 같이 구성된다. 그 중 디렉토리 데이타베이스 부분인 DIB를 제외한 여타 부분은 표준화 작업이 완료된 상태이나, DIB의 실제 구성은 비표준 대상으로 데이타베이스 모델에 대한 언급은 하지않고 있다. 그러므로, 디렉토리 데이타베이스 모델에 대한 연구는 효율적인 통신망 디렉



DMD : Directory Management Domain
DUA : Directory User Agent
DSA : Directory System Agent
DIB : Directory Information Base (비표준화 대상)

DAP : Directory Access Protocol
DSP : Directory System Protocol
DSIP : Directory Shadowing Information Protocol

그림 1. 디렉토리 시스템 구성
Fig. 1. The Configuration of Directory System

토리 운용을 위해서는 필수사항인 것이다. 그러나, 현재까지의 이 분야에 대한 관련 연구는 큰 진척이 없는 상황이다.

지금까지 디렉토리 데이타베이스 관련 연구는 화일 시스템[5][6]과 관계 데이타베이스 시스템 [14]을 이용하여 디렉토리 데이타베이스 부분을 구현한 것과, 관계 모델[13]과 객체 지향 모델 [10]을 적용하여 디렉토리 데이타베이스를 모델링한 것, 분산 디렉토리 환경하에서 디렉토리 스키마 관리[15]를 연구한 것 등이 있었다.

그러나, 현재까지 진행된 연구의 경우 몇가지 문제점을 내포하고 있다. 첫번째의 경우 디렉토리 데이타베이스 모델 선정에 있어서 화일 시스템을 선정하므로써 기존 DBMS의 회복(recovery), 동시성 제어(concurrency control), 보안(security) 기능 등을 이용할 수 없으므로 대량 데이타 처리 시스템으로는 불안정하다는 단점이있고, 관계형 DBMS를 선정한 경우에는 관계형 모델 자체가 갖는 구조적 제약성으로 인하여 계층 구조, 내포 구조, 다중값 등의 디렉토리 정보의 특성을 충분히 반영할 수 없는 단점을 가지고 있다. 두번째의 경우 모델 선정을 객체 지향 모델로 하였지만[10] 모델링 대상 정보를 단순히 사용자 정보로 국한하므로써 디렉토리 관리 정보를 모두 고려하지 않은 단점이 있고, 기존 디렉토리 데이타베이스 저장 정보를 단순히 '이름-주소' 매핑 정보에 국한시키므로써 디렉토리 제공 서비스 수준이 미비하다는 것이다. 이상과 같은 문제점을 해결하기 위해 기존 디렉토리 시스템의 특성을 가장 정확히 반영한 최적의 디렉토리 데이타베이스 모델을 개발하는 것이 본 논문의 목표이다.

이 목표를 달성하기 위하여 본 논문은 다음과 같이 구성하였다. 제2장에서는 디렉토리 데이타베이스 모델링시에 적용할 몇가지 기준을 개발하였으며, 제3장에서는 분류된 정보에 따른 객체지향 모델링을 하였고, 제4장에서는 개발될 모델에 적용할 메소드를 개발하였다.

Ⅱ. 디렉토리 데이타베이스 모델링 기준

디렉토리 데이타베이스 모델링시에 적용되는 공통적인 개념들은 다음과 같은 것이있다.

2.1 디렉토리 저장 정보 분류

디렉토리 데이타베이스를 효과적으로 모델링 하기 위해서는 저장될 수 있는 정보를 분류하여 그 특성에 맞는 기법으로 모델링하는 것이 중요하므로, 본 논문에서는 다음과 같이 4가지로 저장 정보를 분류하였다.

(1)디렉토리 사용자 정보

디렉토리 고유 기능을 위하여 저장하는 정보로서 이름, 주소, 전화번호와 같은 내용이 이에 해당된다.

(2)디렉토리 관리 정보

디렉토리 시스템 관리상에 필요한 정보로서 스키마 관리, 접근 제어, 관리 영역, 유사 관계성 정보 등이 이에 해당한다.

(3)디렉토리 지식 정보

디렉토리 데이타베이스의 분산 (distribution) 및 복제 (replication) 정보를 가리킨다.

(4)부가 정보

기존 디렉토리 시스템의 단순 정보에 네트워크 관리 정보를 추가하므로서 디렉토리 서비스 기능을 향상 시키고자하는 정보이다[3][4][18].

2.2 참조 연관성

디렉토리 논리 구조는 디렉토리 정보 트리 (Directory Information Tree:DIT) 형태로 구성되어 있으며, 이것은 계층 구조를 형성한다. DIT상의 상위 엔트리 (superior entry)와 하위 엔트리 (subordinate entry) 관계는 객체지향 모델의 기존 관계성 (relationship)으로는 표현할 수 없으므로, 새로운 연관성 (association) 정의를 필요로 한다. 이에 DIT 구조상의 상위 클래스에 속한 하위 클래스를 가리키는 값을 참조 연관성 (Referential Association)으로 정의하였다.

2.3 클래스 세분화

본 논문에서 디렉토리 데이타베이스 클래스는 데이타 클래스, 상위 메타 클래스, 사용자 정의 클래스 등으로 세분하여 정의한다. 여기에서, 데이타 클래스 (data class)는 국제 표준에서 정의한 클래스들이 해당하며, 그 예로는 Root, Country, Subschema, Domain 등이 있다. 또한, 사용자 정의 클래스 (user defined class)는 디렉토리 사용자가 필요에 의하여 정의한 클래스로, Crt, Printer, Fax, TaskForceTeam 등이 있을 수 있다. 마지막으로, 상위 메타 클래스는 데이타 클

래스와 사용자 정의 클래스의 상위 클래스 이면서 메타 클래스 성격을 갖는 클래스로서, DIRECTORY, MANAGEMENT, ADMINISTRATION 등이 있다.

2.4 클래스 그룹화

디렉토리 클래스 중 공통적인 특성을 갖는 클래스들을 하나의 상위 메타 클래스의 하위 클래스로 지정하므로써, 디렉토리 클래스 관리의 용이성을 확보하는 것이 클래스 그룹화 (class grouping)의 개념이다. 예를들면, 상위 메타 클래스인 COMPANY에는 Organization과 OrganizationalRole 클래스가 그룹화되고, APPLICATION에는 ApplicationProcess와 ApplicationEntity 클래스가 그룹화된다.

2.5 상위 메타 클래스

상위 메타 클래스 (Super Meta Class)는 메타 정보를 포함하는 메타 클래스와 공통 정보를 포함하는 상위 클래스의 성격을 혼합한 새로운 유형의 클래스이다. 즉, 상위 메타 클래스는 자신의 하위 클래스들을 그룹화하여 공통 정보를 저장하므로써 하위 클래스에 공통 정보 계승 (inheritance) 효과를 발생시키고, 하위 클래스에 대한 메타 정보를 저장하므로써 시스템 운영시 이용할 수 있는 정보를 저장하는 클래스이다.

상위 메타 클래스 예를 표현한 그림 2의 경우, COMPANY는 상위 메타 클래스로서 자신의 하위 클래스들에 대한 메타 정보와 공통 정보를 저장하고 있다. 메타 정보 저장 예를들면, 상위 메타 클래스 COMPANY는 Organization, OrganizationalUnit, OrganizationalRole 클래스들이 꼭 포함해야 할 필수 포함 (MUST Contain) 속성 정보인 organizationName, organizationalUnitName, commonName 등을 저장하고 있으며, 선택 포함 (MAY Contain) 속성 정보는 organizationalAttributeSet 등을 저장하고 있다. 반면 공통 정보의 경우, Organization, OrganizationalUnit, OrganizationalRole 클래스들이 공통적으로 사용 할 수 있는 정보인 president, division, nbr.of staff 등의 속성 값을 각각의 클래스에 저장하는 것이 아니라 상위 메타 클래스 한 곳에만 저장하고 있다. 이 예에서

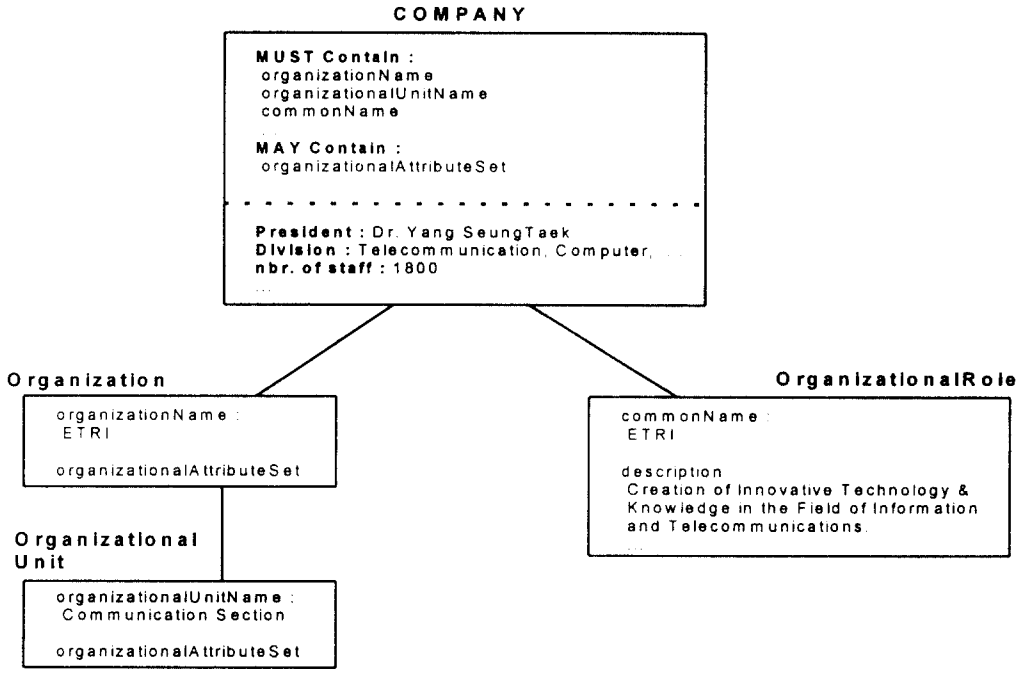


그림 2. 상위 메타 클래스
Fig. 2. Super Meta Class

Organization과 OrganizationalRole 클래스는 객체지향 모델의 IS-A 관계가 아닌 동등 레벨 클래스이지만, 공통의 정보를 상위 메타 클래스에 정의하므로서 계승하여 사용할 수 있는 장점이 발생한다.

Ⅲ. 디렉토리 정보 모델링

디렉토리 데이타베이스 정보 모델링의 개괄적 개념을 클래스 계층 구조로 표현하면, 그림 3과 같다. 여기에서 데이타 클래스인 Root 클래스를 제외한 나머지 클래스들은 상위 메타 클래스이며, Root 클래스의 하위 클래스를 형성한다. DIRECTORY 상위 메타 클래스의 하위 클래스로는 ADMINISTRATION (디렉토리 관리 정보), USER (디렉토리 사용자 정보), KNOWLEDGE (디렉토리 지식 정보) 클래스가 있고, MANAGEMENT (부가 정보:네트워크 관리 정보) 상위 메타 클래스의 하위 클래스로는 STATIC (정적 정보), SEMI-DYNAMIC (준동적 정보) 클래스가 있다. 본 절에서는 디렉토리 저장 정보로 구분한 4가지 정보중 디렉토리 지식 정보를 제외한 3가지 정보에 대한 모델링 작업을 하였다.

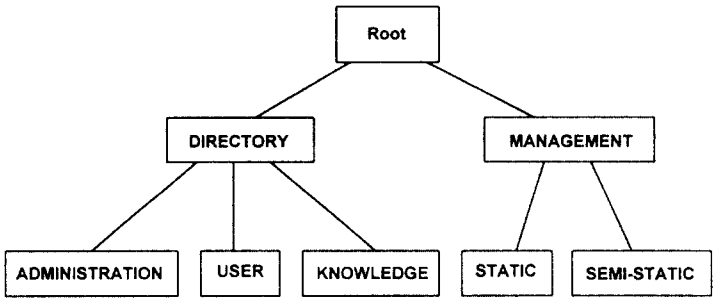


그림 3. 전체 디렉토리 정보 클래스 계층 구조
Fig. 3. The Class Hierarchy of Overall Directory Information

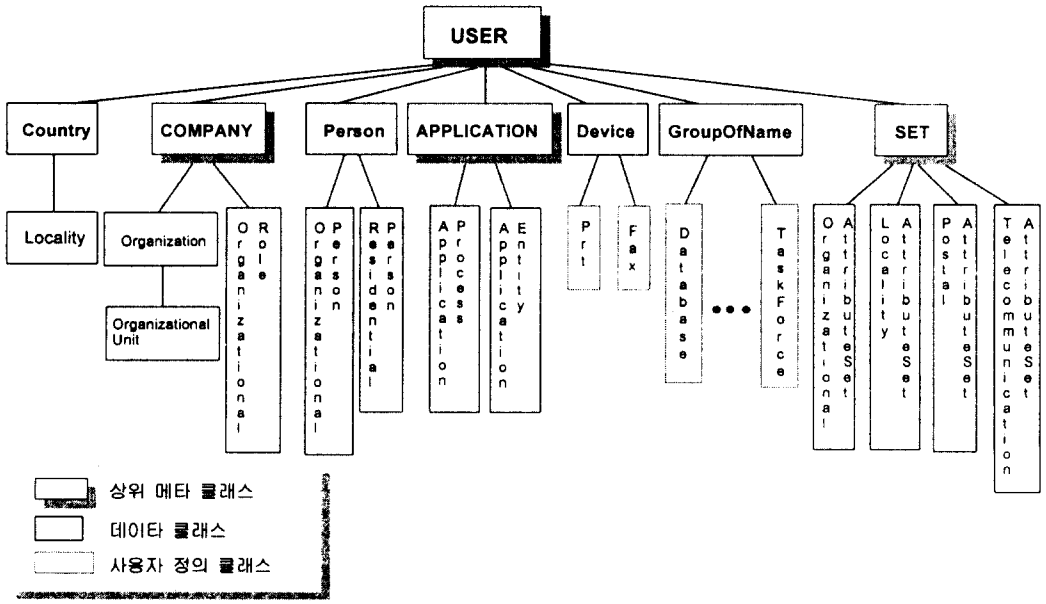


그림 4. 사용자 정보 클래스 계층
Fig. 4. The Class Hierarchy of User Information

3.1 사용자 정보 모델링

디렉토리 사용자 정보 모델은 COMPANY, APPLICATION, SET 등의 상위 메타 클래스와 Country, Person, Device, GroupOfName 등의 데이타 클래스가 USER 상위 메타 클래스의 하위 클래스에 위치하여 그림 4와 같은 클래스 계층 구조를 형성한다. 이러한 클래스 계층 구조는 IS-A 관계로 구성되기 때문에 상위 클래스의 속성 및 메소드는 하위 클래스로 계승 사용이 가능하다. 예를들면, Person 클래스의 속성 및 메소드는 하위 클래스인 OrganizationalPerson 및 ResidentialPerson 클래스로 계승된다.

사용자 정보 클래스 중 일부 객체의 속성 값은 또 다른 클래스를 가리킬 수 있으며, 이것은 IS-PART-OF 관계로 정의할 수 있기 때문에 그림 5와 같은 클래스 복합 계층 구조를 형성한다. 예를들면, Organization 클래스의 OrganizationalAttributeSet 속성은 SET 클래스의 하위 클래스 중 하나인 OrganizationalAttributeSet을 가리킨다.

3.2 관리 정보 모델링

디렉토리 관리 정보 모델은 그림 6과 같이 상위 메타 클래스인 ADMINISTRATION의 하위 클래스로 AdministrativeEntry와 Subentry 클래스가 위치한다. AdministrativeEntry의 참조 연관성은 Subentry를 가리키므로 AdministrativeEntry의 하위에 Subentry가 속함을 알 수 있다. Subentry의 속성 중 AdministrativeRole 값은 또 다른 상위 메타 클래스인 ROLE을 가리키며, ROLE은 Subschema, BasicAccessControl, CollectiveAttributes와 같은 데이타 클래스를 하위 클래스로 갖는다.

3.3 부가 정보 모델링

디렉토리 부가 정보 모델은 그림 7과 같이 네트워크 관리 데이타 중 정적 데이타 클래스를 대표하는 STATIC 상위 메타 클래스와 준정적 데이타 클래스를 대표하는 SEMI-DYNAMIC 상위 메타 클래스로 구성된다. 데이타 클래스 중 Domain, Network, Layer-

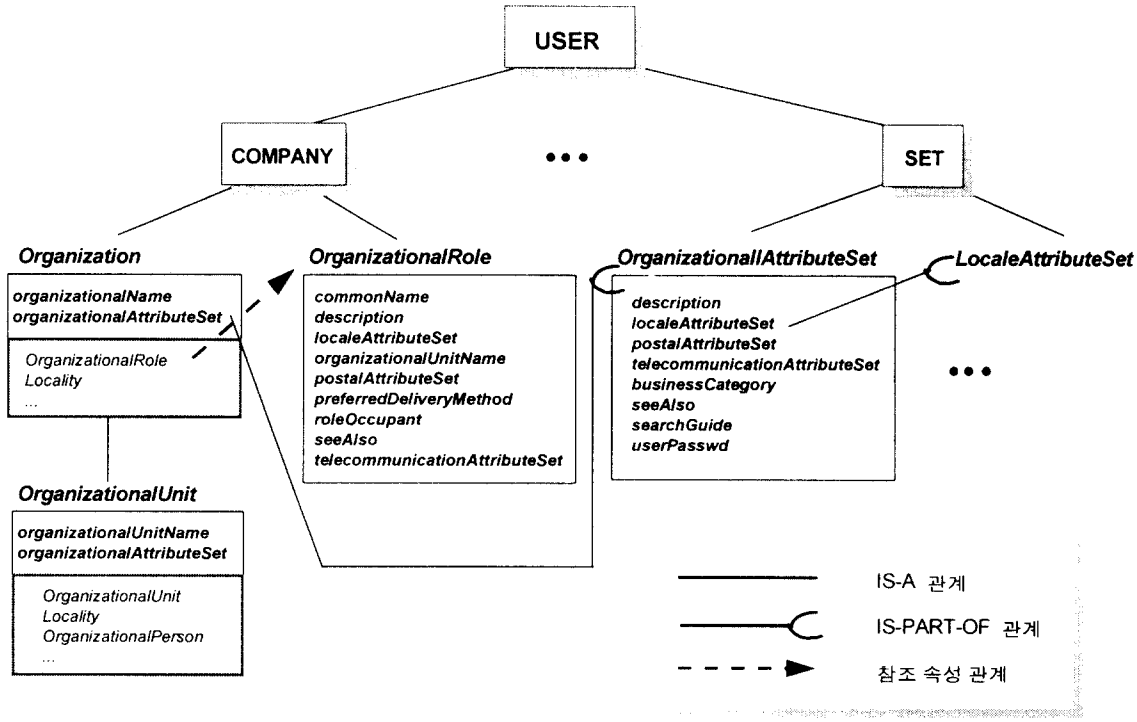


그림 5. 사용자 정보 클래스 복합 계층
Fig. 5. The Class Composition Hierarchy of User Information

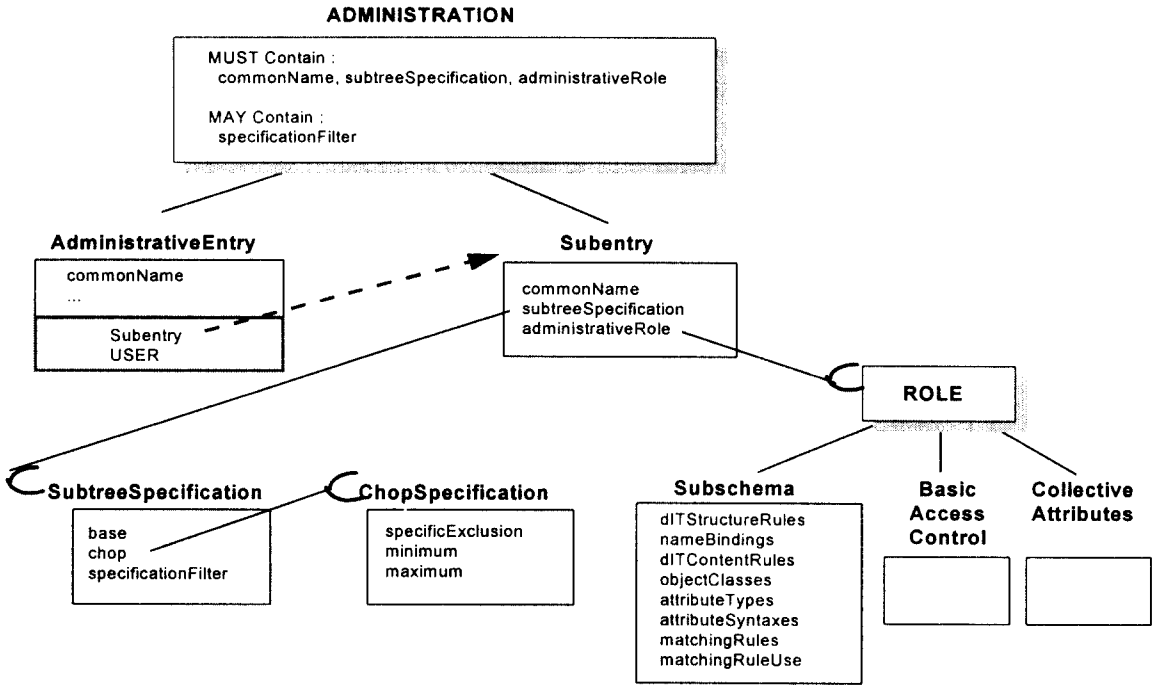


그림 6. 관리 정보 복합 계층
Fig. 6. The Class Composition Hierarchy of Administrative Information

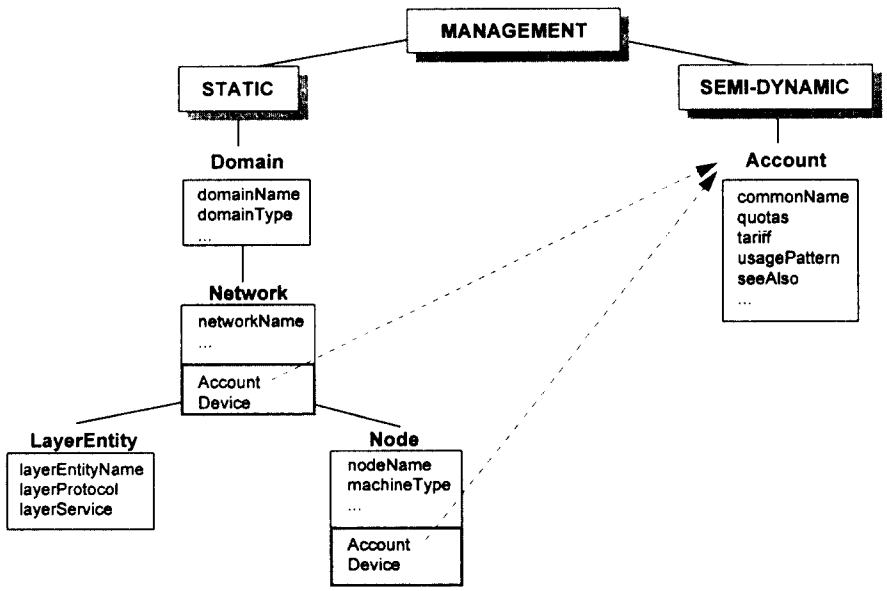


그림 7. 부가 정보 클래스 계층
Fig. 7. The Class Hierarchy of Supplimentary Information

Entity, Node 등은 STATIC 클래스의 하위 클래스에 속하며, Account 클래스는 SEMI-DYNAMIC의 하위 클래스에 속한다. 이때, Account는 Network과 Node 클래스와 참조 연관성이 있다.

Ⅳ. 메소드 정의 및 적용 사례

4.1 메소드 정의

지금까지 정의한 3가지 디렉토리 데이타베이스 정보 모델에 적용할 메소드는 다음과 같이 5가지로 세분화 하였으며, 이 중 제약조건과 트리거는 능동 기반 기법으로 제 3 장에서 정의한 모델의 객체내에 속성값과 함께 캡슐화 (encapsulation)되므로써, 능동 객체 지향 모델을 구성한다.

(1) X.500 정의 추상 서비스

X.500 디렉토리에서 제공하는 17가지 기본 서비스[16]는 디렉토리 데이타베이스 모델 측면에서는 메소드 형태로 정의할 수 있다. 즉, 특정 클래스에 데이타 속성과 메소드가 캡슐화되어 있다가 데이타 속성에 대한 연산 요구가 있을경우, Read 또는 Search와 같은 메소드화된 X.500 추상 서비스 중 해당 메소드가 수행된다.

(2) 매칭 규칙

디렉토리 서비스 수행 중 특성 타입이나 구문 등의 일치성 여부를 검사하는 매칭 규칙 [16]도 메소드로 정의할 수 있다. 즉, 디렉토리 서비스 수행 중 특정 내용의 확인 작업 등을 수행할 때 메소드로 정의된 매칭 규칙이 수행된다.

(3) 추가 메소드

기존 디렉토리 표준에서 정의한 추상 서비스[16]는 디렉토리 데이타베이스 측면에서는 객체 단위의 연산이므로, 속성값 단위로 수행할 수 있는 메소드의 추가 정의가 필요하다.

(4) 제약조건

제약조건 (constraints)이란 데이타베이스의 무결성 (integrity) 제약조건을 감시하는 능동 기반 기법 [2][5] 중 하나이다. 제약조건은 무결성 조건을 위반하는 경우 즉, 조건 부분의 실과값이 거짓일 경우 자동적으로 행동 부분이 실행된다. 디렉토리에서 제약조건으로 정의할 수 있는 메소드는 새로운 속성을 추가하려 할 때 디렉토리 스키마에 정의된 속성 구문 (syntax) 및 크기 (size) 정의와 일치하는 가를 확인하여 일치하지 않을경우 위반 사항에 대한 적절한 조치를 취하는 것이다. 본 논문에서 정의한 두가지 제약조건 메소드는 다음과 같다.

■ 속성 크기 제약조건

IF (size-check (OID, attribute-name)) THEN size-violation (service-stop; return (error-msg));

디렉토리 객체 식별자 (OID)로 지정된 객체내의 속성을 가리키는 attribute-name값의 크기가 디렉토리 스키마에 정의된 것보다 클경우, 트랜잭션을 중단하고 오류 메시지를 출력한다.

■ 속성 구문 제약조건

IF (syntax-check (OID, attribute-name))

THEN syntax-violation (service-stop; return (error-msg));

디렉토리 객체 식별자 (OID)로 지정된 객체내의 속성을 가리키는 attribute-name값의 구문이 디렉토리 스키마에 정의된 것과 다를경우, 트랜잭션을 중단하고 오류 메시지를 출력한다. 이 제약조건 메소드를 사용하면, printable string으로 정의된 속성에 bit string 값을 할당하는 오류를 방지할 수 있다.

이상의 제약조건 메소드는 새로운 속성을 추가하는 메소드인 void add-attribute (OID, attribute-value) 수행 중 무결성 제약조건 검사를 위하여 이용될 수 있다.

(5) 트리거

트리거 (trigger)는 무결성 제약조건과 같이 데이타베이스 조건을 감시하나, 감시 대상 조건이 일관성 위반 (consistency violation) 여부가 아니다[2][12]. 그러므로, 트리거는 제약조건과는 달리 조건의 결과값이 참일 경우 행동 부분을 실행하는 능동 기반 기법이다. 디렉토리에서 트리거로 정의할 수 있는 메소드는 사용자등이 허가되지 않은 연산을 실행하고자 할 경우 조건을 검사하여 자동적으로 대처 방안을 실행시키는 것, 대체 객체 (alias object) 값을 접근할 경우 자동적으로 고유 이름 치환 (dereferencing) 작업을 수행하는 것, 디렉토리 정보 트리 (Directory Information Tree : DIT)상의 비단말 노드(non-leaf node)에 대하여 새로 연산을 방지하는 것 등이 있다. 본 논문에서 정

의한 세가지 트리거 메소드 정의는 다음과 같다.

■허가권이 없는 사용자 접근 방지 트리거

IF (no-permission (UID, OID, attribute-name))

THEN permission-violation (service-stop;return (error-msg));

사용 허가권 (permission)이 없는 사용자 (UID)가 객체를 접근하거나 속성값을 접근할 때, 트랜잭션을 중단하고 오류 메시지를 출력한다.

■대체 객체 고유 이름 치환 트리거

IF (is-alias (OID)) THEN dereferencing ();

디렉토리 객체 식별자 (OID)로 지정된 객체가 대체 객체일 경우, 대체 객체내에 저장된 포인터를 이용하여 실제값으로 변환하는 함수인 dereferencing () 을 호출하여 고유 이름 치환 작업을 수행한다.

■비단말 노드 제거 방지 트리거

IF (is-nonleaf (OID)) THEN remove-violation (service-stop;return (error-msg));

디렉토리 표준에서는 DIT 구조상에서 중간 노드 (객체) 즉, 비단말 노드는 삭제할 수 없는 제한을 두고 있으므로, 디렉토리 객체를 삭제하는 메소드인 void remove-object (OID) 수행 중에는 제거하고자 하는 객체가 비단말 노드인가를 확인하여 결과값이 참인 경우 트랜잭션 수행을 중단하고 오류 메시지를 출력하여 비단말 노드의 제거를 방지한다.

본 절에서 정의한 메소드를 자세히 표현하면 표 1과 같다.

표 1. 메소드 정의

Table 1. The definition of Methods

	메소드 정의		
기존 서비스	Read	string read-object (OID)	OID가 가리키는 객체 판독
	Compare	boolean compare-attribute (OID, attribute-name)	OID내의 속성값 비교
	Abandon	void abandon ()	디렉토리 서비스 요청 포기
	List	string list (OID)	OID가 가리키는 객체 브라우즈
	Search	string search (OID, filter)	OID가 가리키는 객체의 필터 항목을 만족하는 내용 탐색
	AddEntry	void add-object (OID)	OID가 가리키는 객체 추가
	RemoveEntry	void remove-object (OID)	OID가 가리키는 객체 제거
	ModifyEntry	void modify-object (OID, modify-content)	OID가 가리키는 객체에 modify-content 내용 반영
	MidifyDN	void modify-DN (DN1, DN2)	DN1을 DN2로 변경
	Abandoned	string abandoned (error-message)	디렉토리 서비스 포기 결과 출력
	AbandonFailed	string abandon-failed (error-message)	디렉토리 서비스 포기 시도 실패 원인 출력
	AttributeError	string attribute-error (error-message)	속성 오류 발생 원인 출력
	NameError	string name-error (error-message)	이름 사용 오류 발생 원인 출력
	Referral	string referral (access-point)	위임 처리를 위한 타 DSA 접근점 반환
	SecurityError	string security-error (error-message)	보안 오류 발생 원인 출력
	ServiceError	string service-error (error-message)	디렉토리 서비스 오류 발생 원인 출력
	UpdateError	string update-error (error-message)	갱신 오류 발생 원인 출력
매칭 규칙	boolean present-match (OID)		OID가 가리키는 객체 존재 확인
	boolean equality-match (OID1, OID2)		객체간 (OID1, OID2) 일치성 확인
	boolean greaterOrEqual-match (OID1, OID2)		OID1 객체가 OID2 객체 보다 큰가를 확인
	boolean lessOrEqual-match (OID1, OID2)		OID1 객체가 OID2 객체 보다 작은가를 확인
	boolean initial-match (OID1, OID2)		OID1과 OID2 객체의 첫글자가 같은가를 확인
	boolean any-match (OID1, OID2, any)		any가 가리키는 값이 OID1과 OID2 객체에 있는가를 확인
	boolean finial-match (OID1, OID2)		OID1과 OID2 객체의 마지막 글자가 같은가를 확인
	boolean approximate-match (OID1, OID2)		OID1과 OID2 객체가 정확히 일치하지는 않으나 비슷하기는 한가 확인

추가 서비스	void add-attribute (OID, attribute-value) void delete-attribute (OID, attribute-value) void modify-attribute (OID, attribute-value) void add-class () void delete-class () void initialize ()	OID가 가리키는 속성에 attribute-value값 추가 OID가 가리키는 속성에 attribute-value값 제거 OID가 가리키는 속성에 attribute-value값 수정 새로운 클래스 생성 클래스 제거 클래스 초기화
제약조건	boolean size-check (OID, attribute-name) boolean syntax-check (OID, attribute-name)	attribute-name으로 지정한 속성값의 크기가 디렉토리 스키마에 명기된 것보다 클경우 적절한 조치 attribute-name으로 지정한 속성값의 문법이 디렉토리 스키마에 명기된 것과 다를경우경우 적절한 조치
트리거	boolean no-permission (UID, OID, attribute-name) boolean is-alias (OID) boolean is-nonleaf (OID)	사용 허가권이 없는 사용자가 객체나 속성을 접근할 때 적절한 조치 접근 객체가 대체 객체일 경우 고유이름치환 작업 수행 void delete-class() 메소드 수행 요청시 접근 객체가 DIT상의 비단말 노드일 경우 메소드 수행 요청 거부

4.2 적용 사례

본 절에서는 제 3 장에서 개발한 디렉토리 데이타베이스 모델과 4.1절의 메소드를 실세계 통신망 관리 환경에 적용한 사례를 설명한다. 그림 8은 통신망 관리 적용 사례를 나타낸 것이다. 그림 8에서는 공중 디렉토리 서비스 제공자를 kt로 가정하였고, 그 중 사용자 정보는 etri 네트워크에 등록된 사용자를 예로 들었다.

통신망 관리 디렉토리 데이타베이스의 최상위 클래스는 Root이고, 그 하위 클래스로 디렉토리 정보를 포함하는 DIRECTORY 상위 메타 클래스와, 부가 정보 중 네트워크 관리 정보를 포함하는 MANAGEMENT 상위 메타 클래스가 위치한다. 디렉토리 정보 클래스 중 디렉토리 관리 정보는 ADMINISTRATION 상위 메타 클래스와 그 하위 클래스에 저장되고, 디렉토리 사용자 정보는 USER 상위 메타 클래스와 그 하위 클래스에 저장된다. 이와같이 각기 다른 성격의 정보는 참조 연관성을 통하여 상호 관련성을 갖게된다. 즉, ADMINISTRATION 상위 메타 클래스의 하위 클래스인 AdministrativeEntry 클래스의 참조 연관성 중 하나가 USER 상위 메타 클래스이므로 USER 상위 메타 클래스와 그 하위 클래스들은 DIT 상에서 AdministrativeEntry 클래스의 하위에 위치함을 알 수 있고, 이것은 AdministrtiveEntry 클래스 정보 (kt 공중 디렉토리 정보)가 USER 상위 메타 클래스 영역 (etri 네트워크 영역 정보)전체를 관할하는 것임을 알 수 있다. 또한, USER 상위 메타 클래스의 하위 클래스인 Organization 클래스의 참조 연관성 중 하나는 MANAGEMENT 상위 메타 클래스의 하위 클래스인 Network 클래스이므로, Organization 클래스 (조직 정보)에 속하는 Network 클래스 (네트워크 정보)임을 알 수 있다.

대부분의 디렉토리 메소드는 Root 클래스에만 정의되어있고, 필요시 하위 클래스들로 계승된다. 예를 들면, Organization 객체의 내용 검색 서비스 요청이 있을경우 Root 클래스에 저장된 string read-object (OID) 메소드는 Organization 클래스로 계승되어 수행된다. 그러나, 계승이 불가능한 일부 메소드는 특정 클래스에만 정의된다. 예를들어, 디렉토리 표준에서는 DIT 구조상에서 중간 노드 (객체) 삭제시 그 노드의 상하위 노드간 연결이 끊어지는 부작용을 방지하기 위하여 중간 노드의 삭제를 금지하고 있다. 이러한 이유로 인하여 디렉토리 객체를 삭제하는 메소드인 void remove-object(OID)는 organizationalPerson과 같이 DIT상의 단말 클래스에만 정의된다.

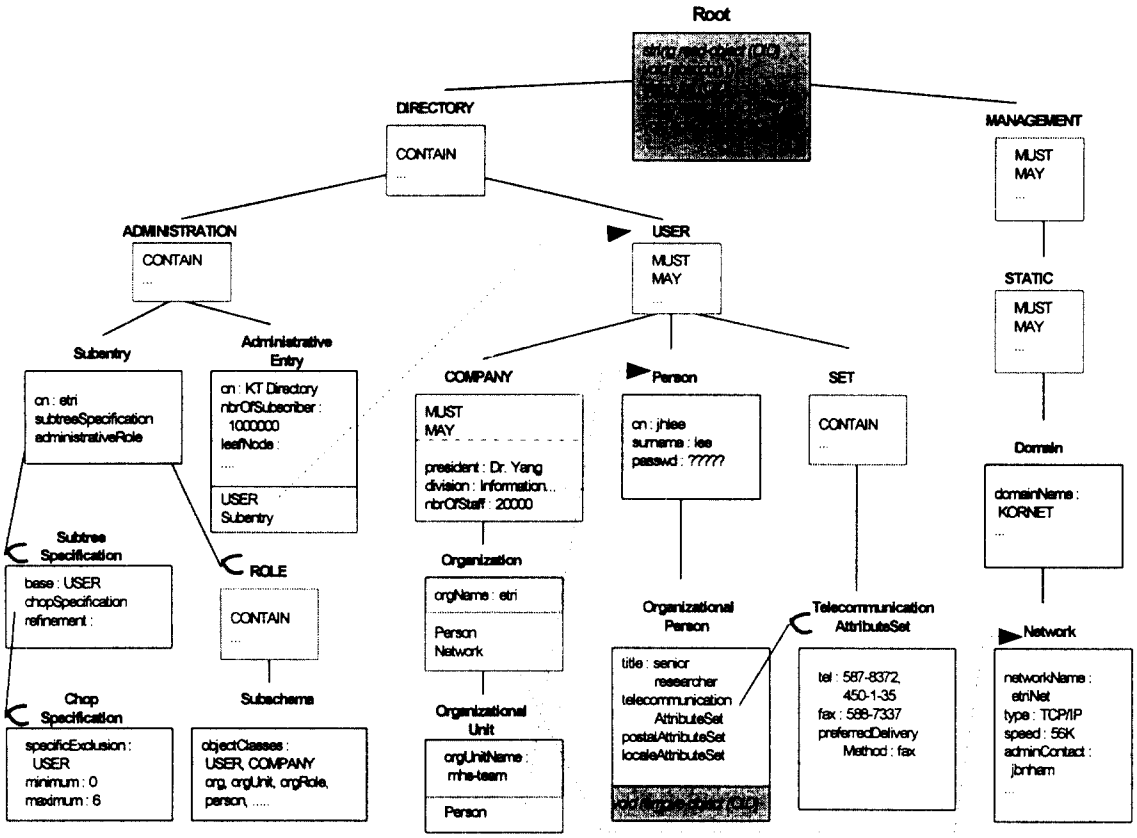


그림 8. 통신망 관리 정보 저장 디렉토리 사례
Fig. 8. Example for Directory Storing Telecommunication Management Information

V. 결 론

본 논문에서는 능동 객체 지향 디렉토리 데이타베이스 모델을 개발하였다. 개발된 디렉토리 데이타베이스 모델은 사용자, 관리, 부가 정보 등으로 구성되며 각각의 정보 모델은 능동 객체 지향 개념을 적용하여 모델링하였다. 디렉토리 데이타베이스 모델을 개발하기 위하여 참조 연관성과 상위 메타 클래스라는 새로운 개념을 제안하였으며, 능동 기반 개념인 트리거와 제약조건 개념을 메소드로 정의하므로써 능동 기반 작동이 가능하도록 설계하였다.

이와같이 개발된 모델은 객체 지향 개념의 속성 및 메소드의 계승 사용이 가능하여 재사용성 (reusability)이 증가하고 복잡한 디렉토리 정보 구조를 자연스럽게 지원할 수 있는 장점이 있으며, 트리거와 제약조건이 메소드 형태로 캡슐화되므로써 예외 처리(exception handling)와 같은 상황에 능동적으로 대처할 수 있는 장점이있다.

앞으로 더 연구가 진행되어야 할 분야는 디렉토리 저장 정보 중 지식 정보에 대한 모델링 작업이 필요하며, 개발 정보 모델간의 연계성 연구 등도 진행되어야 한다. 마지막으로 최종 개발 모델과 기존 모델간의 성능 비교 평가를 통하여 개발 모델의 우수성을 검증할 계획이다.

참 고 문 헌

1. Catriel Beeri, Tova Milo, "*A Model for Active*

Object Oriented Database", the 17th International Conference on VLDB, pp337~349, Sep. 1991.

2. N. H. Gehani, H. V. Jagadish, "*Ode as an Active Database: Constraints and Triggers*", the 17th International Conference on VLDB, pp327~336, Sep. 1991.
3. James W. Hong, et al., "*Integration of the Directory Service in Distributed Systems Management*", 1992 International Conference on Parallel and Distributed Systems, pp142~149, Dec. 1992.
4. James W. Hong, et al., "*Integration of the Directory Service in the Network Management Framework*", the 3rd Int'l Symposium on Integrated Network Management, pp149~160, April 1993.
5. Stephen E. Kille, "*The QUIPU Directory Service*", Message Handling Systems and Distributed Applications, North-Holland, pp173~186, 1989.
6. Stephen E. Kille, "*Implementing X.400 and X.500: The PP and Quipu Systems*", Artech House, Boston MA, 1991.
7. Won Kim, Frederick H. Lochovsky, "*Object-Oriented Concepts, Databases, and Applications*", ACM Press, 1989.
8. Won Kim, "*Introduction to Object-Oriented Databases*", MIT Press, 1990.
9. Won Kim, "*Modern Database Systems-The Object Model, Interoperability, and Beyond*", ACM Press, 1995.
10. J. H. Lee, K. C. Kim, H. C. Lim, "*The Object-Oriented Model of X.500 DIB*", Int'l Conference on Database Applicationas 94, pp80~90, Jan. 1994.
11. Dennis R. McCarthy, Umeshwar Dayal, "*The Architecture of an Active Database Management System*", Proc. ACM-SIGMOD 1989 Int'l Conf. Management of Data, pp215~224, May-June 1989.
12. R. S. Nikhil, "Functional Database, Functional Languages", in Data Types and Persistence, M. P. Atkinson, P. Buneman and R. Morrison(ed.), Springer Verlag, pp51~67, 1988.
13. B. Plattner, et al., "*The ISO/CCITT Directory Service as a Distributed Database: Data Models*", IFIP TC 6/WG 6.5 Working Conference on Message Handling Systems, pp131~140, April 1987.
14. B. Plattner, "*Relational Database Design for an X.500 Directory System Agent*", IFIP COMNET '90, May 1990.
15. Daniel L. Silver, et al., "*X.500 Directory Schema Management*", Data Engineering, pp393~400, Feb. 1994.
16. ITU Recommendation X.500 시리즈, 1992.
17. ITU Recommendation X.700 시리즈, 1990.
18. Xinxin ZHANG, Jean-Luc TESSERON, "*X.500 Directory and OSI Management*", GlobeComm, pp1106~1110, Dec. 1993.

李 載 昊(Jae-Ho Lee) 정회원

1963년 : 12월 7일생
1987년 : 홍익대학교 전자계산학과 학사
1989년 : 홍익대학교 대학원 전자계산학과 석사
1989년~현재 : 한국전자통신연구소 선임연구원

※주관심분야 : 객체지향 DB, GIS, 프로토콜 공학

林 海 喆(Hae-Chull Lim) 정회원

1952년 1월 1일생
1976년 : 서울대학교 계산통계학과 학사
1978년 : 한국과학기술원 전자계산학과 석사
1988년 : 서울대학교 대학원 컴퓨터공학과 박사
1978년~1981년 : 현대엔지니어링
1989년~1990년 : 미국 플로리다대학 방문교수
1981년~현재 : 홍익대학교 컴퓨터공학과 교수

※주관심분야 : 객체지향 DB, 실시간 DB, GIS, 분산 DB