레벨별로 상세화된 공간 데이터를 위한 새로운 공간 인덱싱 기법

권준희[†] · 윤용익[††]

요 약

GIS에 있어 효율적인 액세스 기법은 가장 중요한 요구사항 중 하나이다. 레벨별로 상세화된 공간 데이터를 사용하게 되면 한번에 모든 데이터를 검색할 필요가 없어 효율적인 공간 데이터 검색이 가능해진다. 데이터 검색을 위한 기존의 공간 인덱싱 기법은 이러한 레벨별로 상세화된 공간 데이터를 효율적으로 다루지 못한다. 이를 위해 레벨별로 상세화된 공간 데이터를 위한 공간 인덱싱 기법이 알려졌다. 그러나 이 기법들은 단순화와 선택 연산자를 거친 몇 가지 종류의 레벨별로 상세화된 데이터만을 지원한다는 문제점을 가진다. 이러한 문제점들을 해결하고자 본 논문에서는 모든 종류의 레벨별로 상세화된 공간 데이터를 지원하면서 검색이 효율적으로 이루어지는 새로운 공간 인덱싱 기법을 제안한다. 제안된 기법은 레벨별로 각각의 인덱스 구조가 한 개의 인덱스 구조로 통합된다. 실험 결과는 제안된 기법이 데이터 중복이 발생하지 않으면서도 검색 성능이 우수함을 보인다.

A New Spatial Indexing Method for Level-Of-Detailed Data

Joon-Hee Kwon[†] and Yong-Ik Yoon[††]

ABSTRACT

An efficient access technique is one of the most important requirements in GIS. Using level-of-detailed data, we can access spatial data efficiently, because of no access to the fully detailed spatial data. Previous spatial access methods do not access data with level of detail efficiently. To solve it, a few spatial access methods for spatial data with level of detail, are known. However these methods support only a few kinds of data with level of detail, i.e, data through selection and simplification operations. For the effects, we propose a new spatial indexing method supporting fast searching in all kinds of data with level of detail. In the proposed method, the collection of indexes in its own level are integrated into a single index structure. Experimental results show that our method offers both no data redundancy and high search performance.

Key words: Spatial Indexing, GIS, Level Of Detail, Map Generalization, Scale

1. 서 론

지리정보 시스템을 사용하지 않던 과거에는 오랜 기간동안 여러 축척 지도를 통합하는데 있어 어려움을 느껴왔다. 이에 대해 지리정보 시스템은 지도를 다루는 방법을 크게 변화시켰다. 여러 축척의 데이터는 지리정보 시스템의 확대 및 축소 연산에 의해 통합되어 다루어질 수 있다. 즉, 소축척에서 나타나는 지도가 대축척에서 나타나는 지도와 상세화된 내용에 있어 달라져야 하며 이러한 개념을 레벨별 상세화(Level of Detail)라고 부른다[1]. 지리정보 시스템에 있어 이러한 레벨별 상세화 데이터를 이용하면 전통적인 축척별 데이터를 통합할 수 있을 뿐만 아니라, 레벨별로 필요한 데이터만을 검색하게 되어 효율적인 데이터 검색이 가능해진다.

지리정보 시스템의 공간 데이터는 공간 인덱싱 기

[†] 숙명여자대학교 정보학부 컴퓨터 과학과
[††] 숙명여자대학교 정보학부 교수

법을 이용하여 효과적으로 검색될 수 있다. 그러나 레벨별 상세화 데이터를 처리하는 경우 비효율적이라는 문제점을 가진다. 기존의 공간 인덱싱 기법을 사용하여 레벨별 상세화 데이터를 검색하기 위해서는 2가지 방법이 존재한다. 첫째, 레벨별로 각각의 인덱싱 구조로 저장한다. 이러한 방법은 데이터의 중복 문제를 일으킨다. 둘째, 레벨 구분 없이 한 개의 인덱싱 구조에 저장한 후 검색된 모든 레벨의 데이터 중 해당 레벨에 속한 데이터를 추출하여 사용하는 방법이다. 이는 모든 레벨의 데이터를 검색하여야 하므로 검색 성능이 저하된다[1,13].

기존의 인덱싱 구조에 따르는 문제점을 극복하기 위해서는 레벨별 상세화를 지원하는 공간 데이터를 위한 별도의 공간 인덱싱 기법이 요구된다. 그러나, 이러한 기존 연구는 비교적 활발하지 못했으며 몇 가지 연구에 있어서도 레벨별 상세화 데이터의 유형에 제한을 두어 모든 유형의 레벨별 상세화 데이터에 적용이 불가능하다는 문제점을 가진다. 본 논문에서는 이러한 문제점을 극복하고자 레벨별 상세화 데이터의 어떠한 유형에도 적용이 가능한 새로운 공간 인덱싱 기법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구를 살펴보고, 제 3장에서는 레벨별 상세화 데이터를 위한 공간 인덱싱 구조를 기술한다. 제 4장에서는 제안된 인덱싱 구조의 처리 기법을 설명하며, 제 5장에서는 제안된 기법에 대한 실험 및 성능 평가 결과를 고찰한다. 마지막으로 제 6장에서 결론을 맺는다.

2. 관련 연구

2.1 공간 인덱스

공간 데이터를 처리하기 위한 많은 공간 인덱싱 기법이 알려져 있다. 공간 인덱싱 기법은 크게 트리 기반과 해쉬 기반 방법으로 분류될 수 있다[2]. 트리 기반 방법은 계층화된 검색 트리를 기반으로 하며 대표적으로 R트리[3~6]와 Quad트리[7] 등이 있다. 해쉬 기반 방법은 그리드 파일을 기반으로 하며 대표적으로 그리드 파일[8]과 R파일[9] 등이 있다. 이 중 해쉬 기반 방법은 데이터 분포에 의존적이며 오버플로우 발생 및 이에 따라 효율이 저하되는 문제점을 가진다. 따라서 많은 공간 데이터베이스에서는 이러한 해쉬 기반 방법보다 트리 기반을 선호하고 있으며 이 중에서 R트리 방법이 가장 많이 사용되고 있다.

2.2 레벨별 상세화 데이터

레벨별 상세화 데이터에 대한 연구는 지도 제작자들에 의해 오랫동안 연구되어 왔으며 그 목적은 인간의 해석 능력과 분석 능력에 적당하도록 상세화된 정도를 감소시키고 객체의 밀도를 줄이는데 있다. 화면 확대와 축소 연산은 레벨별 상세화 데이터를 디스플레이하는데 있어 중요한 연산 중 하나이다. 이는 지능화된 줌(intelligent zoom)연산으로 알려져 있으며, 화면 확대를 통해 데이터의 크기만을 확대하는 것이 아니고 데이터를 상세화하게 된다[11]. 레벨별 상세화 데이터는 지도 일반화 연산(map generalization)을 통해 이루어지며 그 연산자는 다음과 같다[12].

(a) 선택(selection) : 중요성에 기반해서 레벨별로 필요한 지도 피처를 추출한다.
(b) 단순화(simplification) : 선이나 면을 구성하는 점의 개수를 감소하거나 굴곡이 많던 라인을 보다 매끄럽게 표현한다. 예를 들면, 해안선 표현을 들 수 있다.
(c) 확장(exaggeration) : 지도 피처 중 의미있는 특징을 강조해서 나타낸다.
(d) 분류(classification) : 여러 개의 각 객체들을 공통적인 속성에 따라 그룹화한다.
(e) 심볼화(symbolization) : 면이 선으로, 선이 점으로 변하는 차원의 변화를 의미한다.
(f) 통합(aggregation) : 인접해 있는 여러개의 객체가 하나의 객체로 표현된다.
(g) 대표화(typification) : 유사한 형태를 가진 많은 수의 별개의 객체들을 동일하고 전형적인 형태를 가지는 작은 수의 객체로 나타낸다.
(h) 형태변형(anamorphose) : 인접성 충돌 문제를 해결하기 위해 객체 집합을 변형한다.

2.3 Reactive 트리

Reactive트리[13]는 R트리에 기반한 레벨별 상세화 데이터를 지원하는 공간 인덱스로 R트리와 유사한 속성을 가지고 있으나 다음과 같은 몇가지 차이점이 있다. 첫째, 실객체가 리프 노드 뿐 아니라, 중간 노드에도 나타난다. 둘째, R트리의 노드 형태에 중요도값(importance value)이 추가된다. 세째, 모든 리

프가 같은 레벨에 나타나지 않는다. 즉, 같은 레벨에 있는 모든 노드는 같은 중요도 값을 가진 엔트리를 포함한다. 보다 중요한 엔트리는 상위 레벨에 저장된다. Reactive트리의 중요한 개념은 객체당 중요도를 부여하고, 이렇게 부여된 값에 따라 중요도가 높은 객체가 상위 레벨에 위치하여 중요도가 높은 객체는 보다 빠른 검색이 가능하다는데 있다. 그러나, 선택 연산자만을 제공한다는 문제점을 가진다.

2.4 Priority Rectangle 파일

Priority Rectangle 파일[14]은 그리드 파일 기법의 변형 중 하나인 R파일에 기반한 레벨별 상세화 데이터를 지원하는 공간 인덱싱 기법 중 하나이다. 이 기법은 R파일과 유사하나 우선순위(priority)를 부여했다는 점이 다르다. Priority Rectangle 파일은 디렉토리 구조에 기반하며 이러한 디렉토리가 우선순위가 서로 다른 블록을 레퍼런스하는 방식으로 구성된다. 이 때, 우선순위가 높은 블록은 우선순위가 낮은 블록보다는 항상 일찍 발견된다는 속성을 가진다. 이 방법은 원하는 단계에 따라 선(polyline)으로부터 선 세그먼트의 끝점 중 몇개를 선택하는 단순화 알고리즘을 사용한다. 그러나, 단순화 연산자만을 제공한다는 점과 공간 인덱싱 기법 중 그리드 파일의 문제점을 그대로 가진다는 문제점을 가진다.

2.5 Multi-scale Hilbert R트리

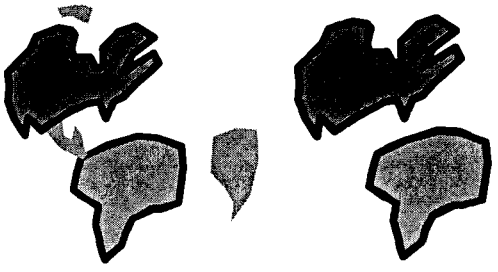
Multi-scale Hilbert R트리[15]는 R트리 중 Hillbert R트리에 기반한 레벨별 상세화 데이터를 지원하는 공간 인덱싱 기법 중 하나이다. Multi-scale Hilbert R트리는 Priority Rectangle 파일과 유사하다. 주요한 차이점으로는 단순화 연산을 수행하는데 따른 검색 속도의 향상을 위해 실제 데이터 파일이 분할되어 저장된다는 것이다. 여기서 사용되는 단순화 알고리즘은 Douglas-Peuker알고리즘을 수정하여 사용한다. 그러나, 단순화와 선택 연산자만을 제공한다는 문제점을 가진다.

3. 레벨별로 상세화된 공간 데이터를 위한 새로운 공간 인덱싱 구조

3.1 개요

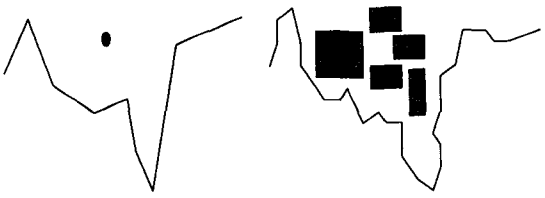
모든 레벨별 상세화 데이터는 일반화 연산을 거쳐 생성된다. 일반화 연산자는 상세화된 데이터로부터 간략화된 데이터를 생성하는 연산자이다. 모든 일반화 연산자에 대해 상세화 레벨에 따른 변화를 살펴보면 다음과 같다. 첫째, 선택 연산자는 상세화된 데이터로부터 일부 데이터 객체를 추출하여 이를 간략화된 데이터로 생성하는 연산자이다. 즉, 선택 연산자를 사용하여 레벨별 데이터가 생성된 경우에는, 상세화된 데이터는 간략화된 데이터와 상세화된 데이터에서 새롭게 추가된 데이터로 이루어진다. 그림 1에서 굵은 선으로 나타난 간략화된 레벨의 데이터가 상세화된 데이터에서 동일하게 나타남을 알 수 있다. 둘째, 선택 연산자 이외의 연산자, 즉, 단순화, 확장, 분류, 심볼화, 통합, 대표화, 형태변형 연산자는 상세화된 데이터로부터 간략하게 수정된 데이터를 생성하는 연산자로 요약될 수 있다. 즉, 선택 연산자 이외의 연산자를 사용하여 단계별 데이터가 생성된 경우에는, 상세화된 데이터는 간략화된 데이터를 공유하지 않고 상세화된 데이터로만 구성된다. 그림 2를 살펴보면 간략화된 데이터와 동일한 데이터가 상세화된 데이터에는 나타나지 않음을 알 수 있다.

레벨별로 상세화된 공간 데이터를 저장하기 위해서는 크게 2가지 방법이 존재한다. 첫 번째 방법은, 레벨별로 상세화된 공간 데이터를 레벨별로 각각 별



(a) 간략화된 데이터 (b) 상세화된 데이터

그림 1. 선택 연산자에 의한 데이터의 예

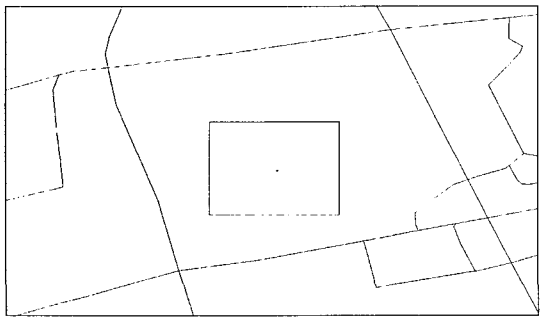


(a) 간략화된 데이터 (b) 상세화된 데이터

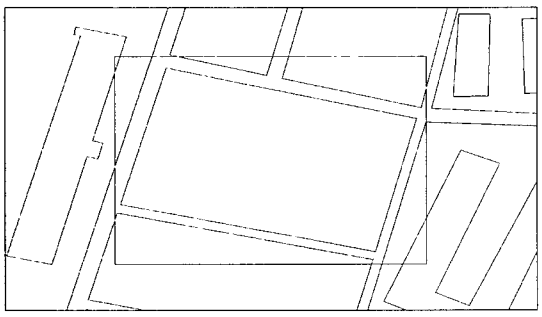
그림 2. 선택 연산자 이외의 연산자에 의한 데이터의 예

개로 저장한다. 두 번째 방법은, 가장 상세화된 공간 데이터만을 저장하고 자동화된 일반화 연산자를 통해 레벨별 데이터를 얻는다. 이 중 두 번째 방법은 첫 번째 방법에 비해 저장 공간 측면에서 효율적이라는 장점을 가지지만 다음과 같은 단점을 가진다. 첫째, 자동화된 일반화 연산자에 대한 연구가 현재까지 완전하지 않아 선택과 단순화 등의 몇 가지 연산자로 연구가 집중된다. 둘째, 일반화 연산자를 거치게 됨으로써 처리 속도가 떨어질 수 있다는 문제점을 가진다. 따라서 본 논문에서는 첫 번째 접근방법을 사용하여 일반화 연산을 거쳐 미리 만들어진 데이터를 그 대상으로 한다.

여기에 비해 그 동안의 레벨별 상세화 데이터를 위한 공간 인덱싱 기법은 두 번째 접근방법을 기반으로 하고 있어 모든 일반화 연산자 중 선택과 단순화 연산자만을 다룬다. 이를 실제 데이터를 사례로 해서 설명하면 다음과 같다. 그림 3은 서울시의 일부 지역을 나타낸 실제 데이터로 1:50,000 축척 데이터의 일부분과 1:5,000 축척 데이터의 일부분을 나타낸다. 여기서 1:50,000 축척 데이터에서 확대를 위한 사각형



(a) 1:50,000 축척 : 레벨 1



(b) 1:5,000 축척 : 레벨 2

그림 3. 레벨별 상세화 데이터의 예

내의 한 개의 점은 1:5,000 축척 데이터에서는 여러 개의 사각형으로 수정되고 있음을 알 수 있다. 이는 지도 일반화 연산자 중 심볼화 연산자의 결과이다. 그러나, 이러한 데이터는 기존의 레벨별 상세화 데이터를 지원하는 공간 인덱스로는 표현할 수 없다. 그러므로 여기서는 레벨별 상세화 데이터 지원을 목적으로 하지 않는 기존의 공간 인덱싱 기법만을 고려한다. 본 논문에서는 기존의 공간 인덱싱 기법 중 R트리를 기반으로 하였다. 이는 R트리가 가장 널리 사용되는 공간 인덱싱 기법이기 때문에 사용되었다.

레벨별 상세화 데이터 지원을 목적으로 하지 않는 기존의 공간 인덱싱 기법을 사용하는 경우는 접근방법에 따라 다음과 같은 문제점을 가진다. 첫 번째 방법은 레벨별로 각각 다른 인덱스에 저장하는 방법에 의해 레벨별 데이터를 지원하는 것이다. 이러한 접근 방법은 각 레벨에서 공유해서 사용하는 데이터가 존재하는 경우 이를 각 레벨별로 중복 저장하는 문제가 발생한다. 예를 들어, 그림 1의 경우 굵은 선으로 나타난 데이터는 간략화된 레벨의 데이터에 대한 인덱스와 상세화된 레벨의 데이터에 대한 인덱스에 중복해서 저장됨을 알 수 있다. 두 번째 방법은 레벨별 데이터를 하나의 인덱스 구조에 저장하는 것이다. 이 방법은 데이터의 중복은 발생하지 않지만 각 레벨에 관계없이 모든 레벨의 데이터를 검색하는 데 따른 검색 속도 저하의 문제가 발생한다.

본 논문에서는 2가지 접근 방법의 장점을 취해 별개로 각각 저장된 여러 개의 인덱스를 한 개의 인덱스 구조로 통합한다. 그림 4는 이러한 통합 인덱스에 대한 개략적인 전체 구조를 보여준다. 제안된 방법은 Reactive트리의 중요도 값에 해당하는 복합 레벨값

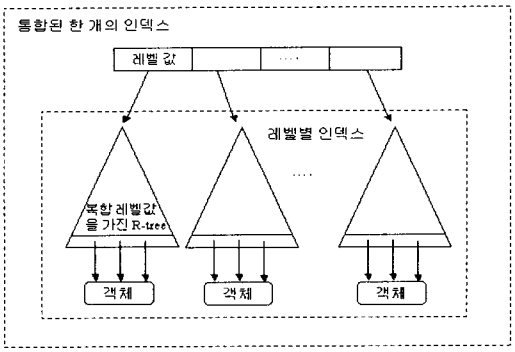


그림 4. 인덱스 전체 구조

(composite level value)을 가진다는 점에서 Reactive 트리와 유사한 속성을 가진다. 그러나, Reactive트리와의 차이점은 Reactive트리는 중요도 값에 의해 선택 연산자만을 지원하는데 비해 제안된 방법은 복합 레벨값에 의해 모든 일반화 연산자를 통한 데이터를 지원한다는 점이다.

3.2 인덱스 구조

그림 5는 제안된 인덱스의 상세 구조를 보여준다. 제안된 인덱스는 레벨 노드와 복합 레벨값을 가진 레벨별 R트리로 구성된다. 본 논문에서는 복합 레벨값을 가진 레벨별 R트리를 LR 트리(Leveled R-tree with composite level value)로 명명한다.

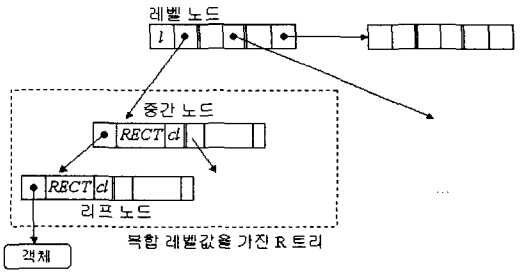


그림 5. 인덱스 상세 구조

레벨 노드는 (*entries*, p_l) 형태로 이루어진다. 여기서 p_l 은 여러 개의 레벨 노드가 존재하는 경우, 다음 레벨 노드를 포인트하는 포인터이며, *entries*는 (l, p_{LR}) 형태를 가진다. 여기서 l 은 레벨값을 의미하며 p_{LR}은 LR 트리의 루트 노드를 포인트하는 포인터이다.

LR 트리는 중간 노드와 리프 노드로 구성된다. LR 트리의 중간 노드는 (*p*, *RECT*, *cl*) 형태를 가진다. 여기서 *p*는 자식 노드를 포인트하는 포인터이고, *RECT*는 하위 노드의 엔트리에 존재하는 모든 사각형을 포함하는 최소 사각형이다. *cl*은 하위 노드의 엔트리에 존재하는 모든 복합 레벨값에 대한 비트 OR (bitwise-OR) 연산자를 수행한 복합 레벨값이다. LR 트리의 리프 노드는 (*id*, *RECT*, *cl*) 형태를 가진다. 여기서 *id*는 데이터 객체를 포인트하는 포인터이고, *RECT*는 *id*에 의해 포인트된 데이터 객체를 포함하는 최소 사각형이다. *cl*은 해당 객체가 나타나는 모든 레벨값에 대해 비트 OR 연산자를 수행한 복합 레벨값이다.

LR트리에 있어 복합 레벨값은 R트리와 차이가 나는 부분으로, 하나의 객체가 나타나는 모든 레벨값을 의미한다. 이 때, 적은 레벨값은 보다 간략화된 데이터를 의미하며 큰 레벨값은 상세화된 데이터를 의미한다. 모든 유형의 일반화 연산자를 통한 레벨별 데이터는 그림 1과 그림 2에서 보는 바와 같이 한 개의 객체가 1개 이상의 레벨에 나타나게 되고, 따라서 한 개의 객체는 1개 이상의 레벨값을 가지게 된다. 본 논문에서는 한 개의 객체가 나타나는 모든 레벨값을 하나의 복합 레벨값으로 나타나는데, 각 레벨별 트리는 복합 레벨값의 사용으로 레벨별 데이터의 중복없이도 데이터간 레벨의 구분이 가능해지는 효과를 가진다. 즉, 레벨별로 중복된 데이터는 가장 간략화된 최상위 레벨에만 오직 한번 저장되고 다른 레벨에서 중복해서 사용할 때는 복합 레벨값을 사용하여 각 레벨을 구분한다.

이 때, 복합 레벨값의 저장 공간과 처리 속도 향상을 위해 본 논문에서는 비트 연산 기법을 사용한다. 이는 1개 이상의 레벨값을 하나의 작은 저장 공간에 저장하면서도, 비트 연산자의 하드웨어 연산에 따른 성능의 향상을 기대할 수 있기 때문이다. 즉, 복합 레벨값을 사용하여 모든 유형의 일반화 연산자를 통한 레벨별 데이터를 작은 저장 공간을 사용하면서도 빠른 검색이 이루어질 수 있도록 한다. 예를 들어, 하나의 객체가 레벨 1, 2, 3에 모두 나타나는 경우 복합 레벨값은 4비트의 비트값으로 '0111'로 표현된다. 즉, 비트값 '0111'은 '0000'으로부터 각 레벨값에 따라 왼쪽으로 레벨값만큼 이동한 결과인 레벨값 1에 대한 비트값 '0001'과 레벨값 2에 대한 비트값 '0010', 그리고 레벨값 3에 대한 비트값 '0100'의 비트 OR 연산에 의해 만들어진다. 각 레벨에 해당하는 데이터를 검색하고자 할 때는 비트 AND(bitwise-AND) 연산자를 사용한다. 즉 레벨값 2에 해당하는 객체를 검색하고자 할 때를 예로 들어보면, 트리 내 모든 노드의 복합 레벨값 *l*에 대해 bitwiseAND(*l*, '0010')을 수행한다. '0010'은 레벨값 2에 대한 비트값에 해당하며 이에 대한 결과가 레벨값 2의 비트값과 동일하면 노드 내 엔트리는 검색된다.

3.3 인덱스 속성

제안된 인덱스 구조는 다음과 같은 속성을 가진

다. 이 때, M_L 은 레벨 노드 내 엔트리의 최대 수라 하고, M_R은 LR 트리 내 엔트리의 최대 수라 가정한다. M_L과 M_R은 한 개의 노드를 읽어들일 때 읽혀질 최대 데이터 수로 읽어들이는 페이지 크기에 따라 결정한다. 이 때 다음은 R트리와 동일한 속성이다.

1. LR 트리의 모든 노드는 LR 트리의 루트 노드가 아니라면 $M_R/2$과 M_R 개 사이의 엔트리를 가진다.
2. LR 트리의 루트 노드는 리프 노드가 아니라면 2개 이상 그리고 M_R 개 이하의 자식 노드를 가진다.
3. LR 트리의 중간 노드에 대한 각 엔트리 (*p, RECT, cl*)에 대해서 *RECT*는 하위 노드의 엔트리에 존재하는 모든 사각형을 포함하는 최소 사각형이다.
4. LR 트리의 리프 노드에 대한 각 엔트리 (*id, RECT, cl*)에 대해서 *RECT*는 *id*에 의해 포인트된 데이터 객체를 포함하는 최소 사각형이다.

다음은 R트리와 다른 속성이다.

1. 레벨 노드 내의 모든 노드는 1과 M_L개 사이의 엔트리를 가진다.
2. 데이터 객체는 레벨 노드 내 각 엔트리 (l, p_{LR})에 대해, 해당 데이터 객체가 나타나는 가장 적은 레벨값 l에 해당하는 엔트리의 p_{LR}에 의해 포인트된 노드에 의해 루트가 된 트리에 나타난다.
3. 레벨 노드 내 각 엔트리 (l, p_{LR})은 적은 레벨값 l에서 큰 레벨값 l 순으로 정렬된다
4. LR 트리 내 각 엔트리(*p, RECT, cl*)에 대해, 복합 레벨값 *cl*은 *p*에 의해 포인트된 모든 엔트리의 *cl* 혹은 레벨값에 대해 비트 OR 연산자를 수행한 결과이다.

4. 레벨별로 상세화된 공간 데이터를 위한 새로운 공간 인덱싱 처리 기법

4.1 검색

본 절에서는 레벨별로 상세화된 공간 데이터를 위한 검색 알고리즘을 기술한다. 검색 알고리즘은 기존의 R트리와 유사하나 레벨 노드와 LR 트리의 엔트리 내 복합 레벨값을 검색한다는 점에서 차이가 존재한다. 검색 방법을 간략히 기술하면 다음과 같이 요약된다. 우선 검색 대상이 되는 레벨값보다 작거나 같은 레벨값을 가지는 LR트리의 루트 노드를 레벨 노드로부터 검색한다. 다음으로, 이렇게 검색된 각 LR트리에서 검색 대상이 되는 레벨값을 포함하는 복합 레벨값을 가지는 노드를 검색하여 최종적으로 해당 데이터 객체가 발견된다. 이를 나타내면 그림 6과 같다.

입력 T : 인덱스 트리, W : 검색 윈도우, L : 검색 레벨 값

출력 W를 오버랩하는 레벨 값 L 내에 있는 모든 객체

단계 1. 검색 레벨 값 L에 해당하는 LR 트리를 검색한다.
1. 인덱스 트리 T 내에 있는 각 엔트리 (l, p_{lr})에 대해서 $l \leq$L인 엔트리를 검색한다.
2. 1로부터 검색된 각 엔트리의 p_{lr}에 의해 포인트된 트리를 LT, l을 L이라 하고, 단계 2를 수행한다.

단계 2. LR 트리 LT에서 레벨값 L을 가진 데이터를 검색한다.
1. 인덱스 트리 LT에 의해 포인트된 노드의 각 엔트리 (*p,RECT,cl*)에 대해서 bitwise AND(*cl*,L)을 수행한다.
2. bitwiseAND 연산의 결과가 레벨값 L과 같으면, *RECT*가 검색 윈도우 W를 오버랩하는지를 체크한다.
3. 2번의 결과로부터 참값을 얻으면 다음을 수행한다. 즉, 해당 노드가 중간 노드면 단계 2를 순환해서 수행하고, 리프 노드면 각 엔트리로부터 데이터 객체를 리턴한다.

그림 6. 레벨별로 상세화된 공간 데이터 검색 알고리즘

4.2 삽입

본 절에서는 레벨별로 상세화된 공간 데이터를 위한 삽입 알고리즘을 기술한다. 삽입 방법을 간략히 기술하면 다음과 같이 2가지 경우로 요약된다. 첫 번째 경우는, 간략화된 데이터로부터 데이터가 수정되는 경우이다. 이는 복합 레벨값을 처리하는 부분을 제외하고는 R트리와 동일하다. 두 번째 경우는, 간략화된 데이터를 공유하여 사용하는 경우이다. 이 때는

해당 데이터를 소유하고 있는 가장 간략화된 데이터의 레벨값에 해당하는 LR트리 내 노드의 복합 레벨값에 해당 레벨의 레벨값을 비트OR 연산자에 의해 추가한다.

입력 T : 인덱스 트리, E(*id,RECT*) : 삽입할 엔트리, L : 삽입될 레벨값
출력 객체 삽입 후의 새로운 인덱스 트리 T

경우 1. 간략화된 레벨의 데이터로부터 수정된 데이터를 삽입한다.
1. 레벨 노드로부터 레벨 L과 동일한 레벨값을 가진 엔트리를 찾아, 이에 해당하는 LR 트리 LRT를 찾는다. 이 때, 레벨 노드에 레벨 L과 동일한 레벨값을 가진 엔트리가 존재하지 않으면 레벨 노드 내에 레벨 값 L을 가진 엔트리를 삽입한다.
2. LR 트리 LRT 내에 레벨값 L을 가진 엔트리 E를 리프 노드에 삽입한다. 이 때의 삽입 과정은 R 트리의 삽입 알고리즘과 동일하다.
3. 삽입된 엔트리를 가진 리프 노드와 이에 따라 새로 분할된 리프 노드가 있는 경우 해당 리프 노드로부터 LR 트리 LRT의 루트 노드까지 상승하면서 하위 노드의 모든 엔트리에 대한 복합 레벨값을 대상으로 bitwiseOR 연산을 수행하여 이 결과를 상위 노드로 전파시킨다.

경우 2. 간략화된 레벨의 데이터를 공유하여 사용하는 데이터를 삽입한다.
1. 엔트리 E를 가지고 있는 리프 노드 LN과 해당 엔트리 LE를 검색한다.
2. 검색된 리프 노드 LN의 엔트리 LN(*id, RECT,cl*)에 대해서 bitwiseOR(*cl*,L)을 수행한다.
3. 리프 노드 LN으로부터 LR 트리의 루트 노드까지 상승하면서 하위 노드의 모든 엔트리에 대한 복합 레벨값을 대상으로 bitwiseOR 연산을 수행하여 이 결과를 상위 노드로 전파시킨다.

그림 7. 레벨별로 상세화된 공간 데이터 삽입 알고리즘

4.3 삭제

본 절에서는 레벨별로 상세화된 공간 데이터를 위한 삭제 알고리즘을 기술한다. 삭제 방법을 간략히 기술하면 삽입 방법과 같이 다음과 같은 2가지 경우로 요약된다. 첫 번째 경우는, 가장 간략화된 데이터로부터 데이터가 삭제되는 경우이다. 이는 데이터가 완전히 삭제되는 것을 의미한다. 이 경우는 복합 레벨값을 처리하는 부분을 제외하고는 R트리의 삭제 방법과 동일하다. 두 번째 경우는, 간략화된 데이터를 공유하여 사용하는 경우이다. 이 때는 해당 데이터를 소유하고 있는 가장 간략화된 데이터의 레벨값

입력 T : 인덱스 트리, E(*id,RECT*) : 삭제할 엔트리, L : 삭제될 레벨값
출력 객체 삭제 후의 새로운 인덱스 트리 T

경우 1. 인덱스 트리로부터 데이터를 완전히 삭제한다.
1. 레벨 노드로부터 레벨 L과 동일한 레벨값을 가진 엔트리를 찾아, 해당하는 LR 트리 LRT를 찾는다.
2. LR 트리 LRT 내의 리프 노드 LE를 삭제한다. 이 때 삭제 과정은 R 트리의 삭제 과정과 동일하다.
3. 삭제된 엔트리의 리프 노드로부터 LR 트리 LRT의 루트 노드까지 상승하면서 하위 노드의 모든 엔트리에 대한 복합 레벨값을 대상으로 bitwiseOR 연산을 수행하여 이 결과를 상위 노드로 전파시킨다.

경우 2. 레벨별로 공유하여 사용되는 데이터에 대한 레벨값을 삭제한다.
1. 엔트리 E를 가지고 있는 리프 노드 LN과 해당 엔트리 LE를 검색한다.
2. 리프 노드 LN의 엔트리 LN(*id,RECT,cl*)에 대해서 bitwiseAND(*cl*, bitwiseNOT(L))을 수행한다.
3. 리프 노드 LN으로부터 LR 트리의 루트 노드까지 상승하면서 하위 노드의 모든 엔트리에 대한 복합 레벨값을 대상으로 bitwiseOR 연산을 수행하여 이 결과를 상위 노드로 전파시킨다.

그림 8. 레벨별로 상세화된 공간 데이터 삭제 알고리즘

에 해당하는 LR트리 내 노드의 복합 레벨값에 해당 단계의 레벨값을 비트 AND 연산자와 비트 NOT 연산자에 의해 삭제한다.

5. 실험 및 성능 평가

5.1 실험 환경

제안된 인덱스와 R트리는 윈도우즈에서 구동되는 Cygwin 기반 하에 GNU C++을 사용하여 구현하였다. 사용된 시스템의 사양은 다음과 같다.

CPU : 펜티엄 III 800EB

RAM : 256MB

운영체제 : Windows 2000

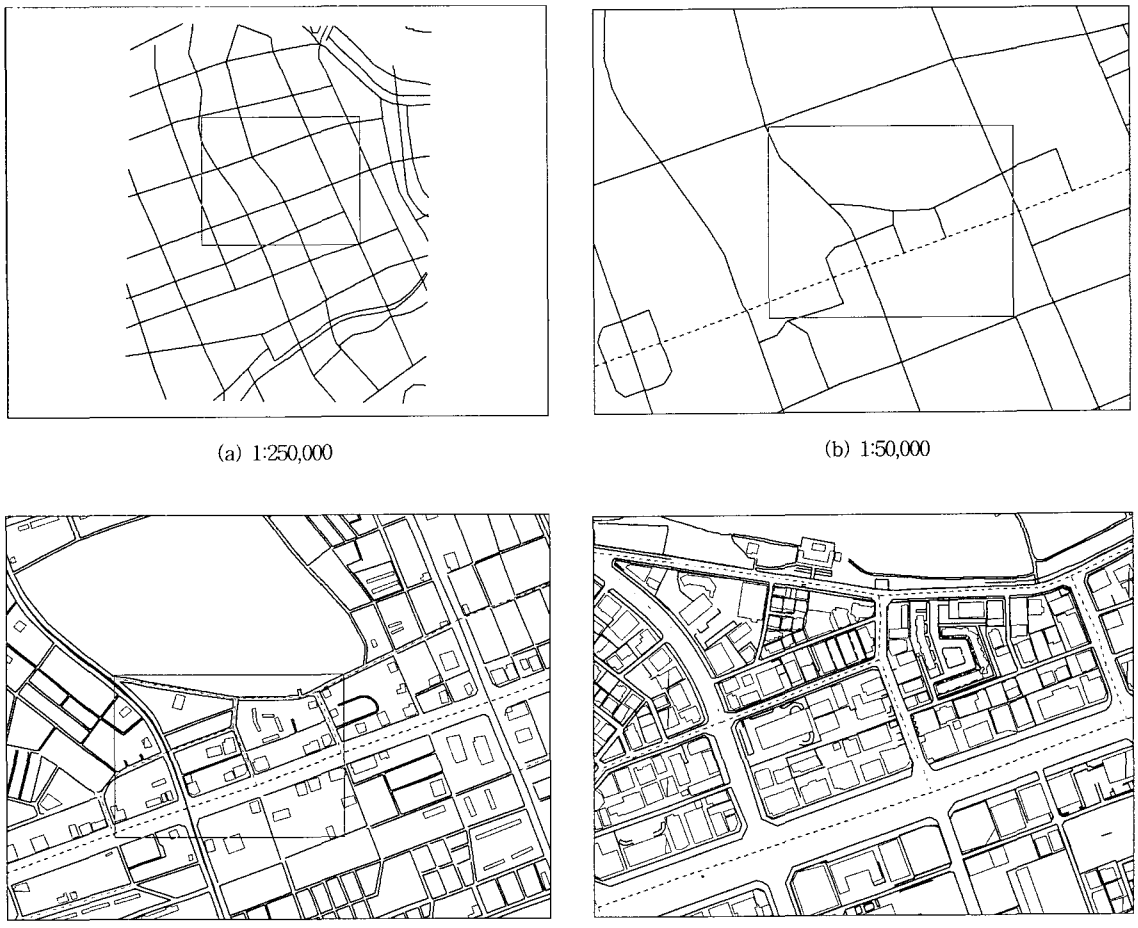
데이터 수의 변화에 따른 성능을 보다 명확히 하기 위해 다음과 같이 가정하였다.

인덱스 트리 내의 최대 엔트리 수 : 5

노드 액세스시 버퍼링 기법 : 적용하지 않음, 즉 필요할 때마다 디스크로부터 읽음

5.2 실험 데이터 및 실험 질의문

실험에 사용된 데이터는 서울시 지역 중 일부에 대한 4개 축척의 같은 영역에 대한 실제 데이터를 대상으로 하였으며 표 1과 같다. 실험 질의문으로는 4개의 다른 레벨별로 10,000개의 질의 윈도우를 랜덤하게 생성하였다. 레벨화된 지리정보 데이터를 검색하는데 있어서, 각 객체는 질의 윈도우가 커지면 적



(a) 1:250,000 (b) 1:50,000

(c) 1:5,000 (d) 1:1000

그림 9. 4개의 다른 레벨에 대한 실험 데이터와 실험 질의 윈도우의 예

표 1. 실험 데이터 요약

레벨값	축척	간략화된 레벨의 객체로부터 수정된 객체 수	간략화된 레벨의 객체와 중복된 객체 수	
			객체의 최소 레벨값	객체수
1	1 : 250,000	37	해당 없음	
2	1 : 50,000	390	1	3
3	1 : 5,000	17228	1	3
			2	34
4	1 : 1,000	74982	1	3
			2	34
			3	5401

은 레벨값에 속한 객체가 나타나게 된다. 그림 9는 실험 데이터에 대한 실험 질의 윈도우를 보인다.

5.3 실험 결과

본 절에서는 5.1과 5.2절에 따라 R트리와 제안된 인덱싱 기법에 대해 검색 성능과 총 메모리 용량을 실험한 결과를 고찰한다. 기존의 R트리에서 레벨별로 상세화된 공간 데이터를 R트리에서 나타내기 위해서는 레벨별로 별도 구축된 R트리의 집합과 한 개의 R트리 내에 레벨별 데이터를 모두 구축한 경우로 나누어서 그 성능을 비교하였다.

검색 성능을 비교하기 위해 본 논문에서는 액세스된 평균 노드의 수와 평균 응답 시간을 측정하였다. 그림 10은 레벨별로 액세스된 노드 수를 보이며 다음과 같은 사항을 알 수 있다. 첫째, 제안된 인덱스 구조는 하나의 R트리로 구축된 경우보다 늘 성능이 우수하다. 둘째, 제안된 인덱스 구조는 레벨별로 별도 구축된 R트리의 집합과 성능면에서 거의 동일하다. 그림 11에서는 레벨별 평균 응답 속도를 보여준다. 제안된

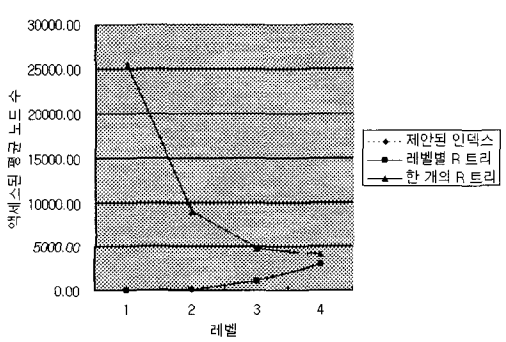


그림 10. 액세스된 평균 노드 수

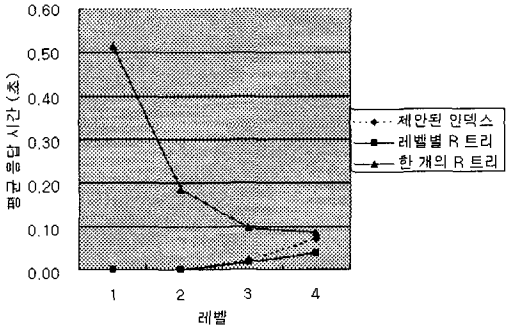


그림 11. 평균 응답 시간

인덱스 구조는 그림 10의 결과와 거의 동일하나 레벨별로 별도 구축된 R트리의 집합에 비해 레벨 4에서 약간의 성능 속도의 차이를 보이고 있다. 이는 복합 레벨값의 처리에 소요되는 시간에 따른 결과이다. 그러나, 그 차이는 0.03초에 불과하므로 레벨별로 별도 구축된 R트리의 집합과 성능면에서 거의 동일하다고 볼 수 있다.

그림 12는 각 인덱스 기법에 의해 구축된 인덱스의 크기를 보여주며 그 결과는 다음과 같다. 제안된 인덱스 구조와 한 개 R 트리의 소요되는 메모리 용량이 적으며, 이는 데이터를 중복 저장하지 않기 때문이다. 이에 비해 검색 속도 측면에서 우수한 성능을 보였던 레벨별 R트리의 집합은 좋지 않은 결과를 얻었다. 이는 레벨별 데이터간 중복되는 데이터를 중복하여 저장하는데 따른 결과이다.

실험을 통해 레벨별 R 트리는 검색 성능에 있어서는 우수하나 데이터 중복성에 의한 부작용과 메모리 용량의 증가라는 단점을 가진다. 또한 한 개의 R트리는 적은 메모리를 사용하나 검색 성능 면에서 좋지

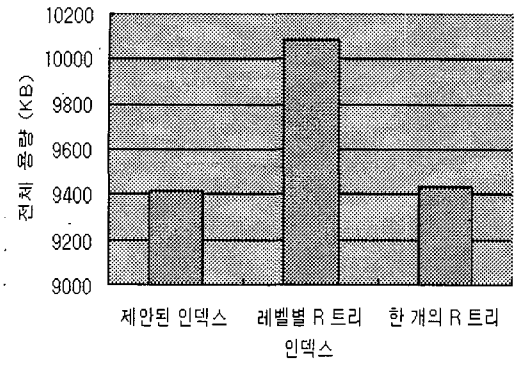


그림 12. 인덱스의 전체 용량

않은 결과를 보였다. 이에 비해, 제안된 인덱싱 기법은 적은 메모리를 사용하면서도 레벨별 R 트리를 사용할 때와 거의 대등한 검색 속도를 보임을 알 수 있다. 따라서, 제안된 기법은 레벨별로 상세화된 공간 데이터 검색에 있어 우수한 인덱스 구조임을 알 수 있다.

6. 결론 및 향후 연구과제

공간 데이터베이스에 있어 가장 큰 요구사항 중 하나는 대용량의 데이터를 효율적으로 처리하는 방법이다. 이를 위해 공간 인덱싱 기법에 대한 많은 연구가 있어 왔다. 그러나 이러한 대용량의 데이터를 보다 효율적으로 처리하기 위해서는 전체 데이터에 대한 공간 인덱싱 기법보다는 레벨별로 상세화 정도가 다른 데이터를 검색할 수 있는 공간 인덱싱 기법이 필요하다.

기존의 레벨별로 상세화된 공간 데이터를 지원하기 위한 공간 인덱싱 기법에 대한 연구는 활발하게 이루어져 오지 않았으며, 기존의 연구도 지도 일반화 연산자 중 2가지만을 지원한다는 문제점을 가지고 있었다. 따라서 모든 지도 일반화 연산자를 반영한 레벨별로 상세화된 데이터를 위해서는 기존의 공간 인덱싱 기법을 사용해야 했다. 그러나, 이런 경우 데이터의 중복성과 데이터 검색의 비효율성이라는 문제점을 가지고 있다.

이를 위해 본 논문에서는 레벨별로 상세화된 공간 데이터를 위한 새로운 공간 인덱싱 기법을 제안하였다. 본 논문에서는 레벨별 데이터간 데이터 중복에 따른 문제점을 없애면서도 데이터 검색이 효율적으로 이루어지는 것을 목표로 하였다. 이를 위해 제안된 기법의 구조와 알고리즘을 소개하였다. 또한, 방대한 양의 실제 데이터를 대상으로 한 실험을 통해 그 효율성을 보였다.

본 논문은 다음과 같은 점에서 이전의 연구와 차별화 된다. 첫째, 레벨별로 상세화된 데이터를 지원하는 기존의 공간 인덱싱 기법과 비교할 때 제안된 기법은 모든 유형의 레벨별로 상세화된 데이터에 적용될 수 있다. 둘째, 레벨별로 상세화된 기존의 공간 데이터 지원을 목적으로 하지 않은 기존의 공간 인덱싱 기법에 비해 제안된 기법은 검색 속도와 메모리 용량 모두에서 효율적이다.

향후 연구과제로는 간략하게 수정된 데이터를 생성하는 일반화 연산자 부분에 대해서 각 유형별 특수성을 세분화하여 보다 효과적인 처리가 가능하도록 하는 연구가 필요하며, 이외에도 레벨간 데이터의 일관성을 고려하지 않은 현재의 기법으로부터 레벨간 데이터의 일관성을 고려하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] C.Davis, A.Laender, "Multiple Representations in GIS: Materialization, Geometric, and Spatial Analysis Operations", Proceedings of the 7th International Symposium on Advances in GIS, pp.60-65, Kansas City, MO, 1999.

[2] V. Gaede, O. Gunther, "Multidimensional Access Methods", ACM Computing Surveys, 30(2), pp.170-231, 1998.

[3] A. Guttman, "R-trees : A Dynamic Index Structure for Spatial Searching", Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.47-54, Boston, MA., 1984.

[4] T.Sellis, N.Roussopoulos, and C.Faloutsos, "The R+-tree : A Dynamic Index for Multidimensional Objects", Proceedings of the 13th International Conference on VLDB, pp.507-518, Brighton, England, 1987.

[5] N.Beckmann, H.P.Kriegel, R.Schneider,R, and B.Seeger, "The R* tree : An Efficient and Robust Access Method for Points and

Rectangles", Proceedings of ACM SIGMOD International Conference on Management of Data, pp.322-331, Atlantic City, NJ, 1990.

[6] I. Kamel, C.Faloutsos., "Hilbert R-Tree : An Improved R-Tree Using Fractals", Proceedings of 20th VLDB, Santiago de Chile, Chile, pp.500-509, 1994.

[7] H.Samet, "The Quadtree and Related Hierarchical Data Structure", ACM Computing Surveys, 16(2), pp.187-260, 1984.

[8] J.Nievergelt, H. Hinterberger, and K.C.Sevcik, "The Grid file : An Adaptable, Symmetric Multikey File Structure", ACM Transactions on Database Systems. 9(1), pp.38-71, 1984.

[9] A.Hutflesz, H-Werner Six, and P.Widmayer, "The R-file : An Efficient Access Structure for Proximity Queries", Proceedings of IEEE 6th International Conference on Data Engineering, pp.372-379, Los Angeles, CA, 1990.

[10] S.Spaccapietra, C.Paren, and C.Vangenot, "GIS Databases :From Multiscale to Multi Representation", Proceedings of the 4th International Symposium, SARA-2000, pp.26-29, Horseshoe Bay, Texas, USA, 2000.

[11] S.Timpf, "Cartographic objects in a multi-scale data structure", Geographic Information Research:Bridging the Atlantic. 1(1), pp.224-234, London, Taylor and Francis, 1997.

[12] A.Ruas, "Multiple Representation and Generalization", Lecture Notes for "sommarkurs : kartografi", 1995.

[13] P.V. Oosterom, "The Reactive-Tree : A Storage Structure for a Seamless, Scaless Geographic Database", Proceedings of Auto-Carto, 10, pp.393-407, 1991.

[14] B.Becker, P.Widmayer, "Spatial Priority Search : An Access Technique for Scaless Maps", Proceedings of ACM SIGMOD, pp.128-137, Denver, Colorado, 1991.

[15] P.F.C Edward, K.W.C Kevin, On Multi-Scale Display of Geometric Objects, International Journal on Data and Knowledge Engineering, 40(1), pp.91-119, 2002.

권 준 희

1992년 숙명여자대학교 전산학과 졸업(학사)
1994년 숙명여자대학교 대학원 전산학과 졸업(석사)
2002년 숙명여자대학교 대학원 컴퓨터과학과 졸업예정(박사)
1994년~현재 쌍용정보통신 GIS기술팀 과장
2000년 전자계산조직응용기술사
관심분야 : 공간 데이터베이스, GIS, 멀티미디어 데이터베이스, 컴포넌트, 객체지향 모델링 및 방법론

윤 용 익

1983년 동국대학교 통계학과 졸업(학사)
1985년 한국과학기술원 전산학과 졸업(석사)
1994년 한국과학기술원 전산학과 졸업(박사)
1985년~1997년 한국전자통신연구원 책임연구원
1997년~현재 숙명여자대학교 정보과학부 교수
관심분야 : 멀티미디어 분산시스템, 멀티미디어 통신, 실시간 처리 시스템, 분산 미들웨어 시스템, 데이터베이스 시스템, 실시간 OS/DBMS