인터넷 지리정보시스템에서 단계화된 지리정보의 효율적인 데이터 검색을 위한 공간 인덱싱 기법

Spatial Indexing Method for Efficient Retrieval of Levelized Geometric Data in Internet-GIS

권 준 희* 윤 용 익**
Joon-Hee Kwon Yong-Ik Yoon

요 약

최근 인터넷 지리정보시스템에 대한 요구가 증가하면서 효율적인 공간 데이터 검색에 대한 필요성이 커지고 있다. 효율적인 공간 데이터 검색을 위해서는 단계별로 상세화된 데이터를 검색하는 기법이 요구되며, 이러한 데이터를 효율적으로 처리하는 공간 인덱싱 기법이 필요하다. 본 논문에서는 효율적인 공간 데이터 검색을 위한 단계화된 지리정보 데이터 검색을 지원하는 공간 인덱싱 기법을 제안한다. 기존의 공간 인덱싱 기법은 단계별 데이터를 검색하는데 비효율적이며, 단계별 데이터를 지원하는 몇 가지 공간 인덱싱 기법도 모든 종류의 단계별 데이터를 지원할 수 없는 문제점을 가진다. 제안된 구조는 모든 종류의 단계별 데이터를 지원하며, 메모리 용량과 검색 시간 모두에서 이전의 공간 인덱싱 기법보다 우수하다.

Abstract

Recently, Internet GIS(Geographic Information Systems) is increasing. From the results, more efficient spatial data retrieval is needed. For more efficient retrieval, a spatial indexing method is needed. This paper proposes an efficient spatial indexing method for levelized geometric data retrieval. Previous indexing methods are not adequate to retrieve levelized geometric data. For the effects, a few indexing methods for levelized geometric data, are known. But these methods support only a few kinds of levelized geometric data. The proposed method supports all kind of levelized geometric data and outperforms to the previous method both in retrieval time and memory capacity.

1. 서 론

최근 인터넷 기반의 지리정보시스템에 대한 요구가 점차 증가하면서 대용량 데이터에 대한 처리가 가장 큰 문제로 대두하고 있다. 인터넷 기반의 시스템에 있어 특징적인 것은 원격의 데이터를 가지고 와야 한다는 것이다. 이러한 과정은 특히 데이터가 큰 경우 매우 심각한 결과를 초래한다. 이러한 문제는 지리정보시스템과 같은 대용량 데이터를 가지고 있는 경우 더욱 큰 문제가 된다. 이를 위해, 전통적으로 존재해 왔던 지도의 축척별 상세화 개념을 사용함으로써 보다 효율적인 인터넷 지리정보시스템 검색을 기대할 수 있다. 전통적으로 지도는 동일한 정보라 하더라도 대상 영역에 따라 상세화 정도가 다른 축척에서 다르게 그려져 왔다. 즉, 소축척에서 나타나는 지도가 대축척에서 나타나는 지도와 상세화된 내용에 있어 달라져야 한다. 이러한 단계별로 상세화된 지리정보 데이터를 사용하게 되면 효율적인 검색 이외에도 단계별로 필요한 데이터만을 검색하게 되므로 검색 결과의 품질도 높아진다. 지도 제작에 있어서 이러한 문제는 오랫동안 논의되어 왔으며 지도 일반화(map generalization)문제로 알려져 왔다[1]. 즉, 지도 일반화를 통해 단계별로 사용자가 인식할 수 있는 수준의 데이터만을 보임으로써 효율적인 검색과 비쥬얼라이징의 향상을 기대할 수 있게 된다.

* 숙명여자대학교 컴퓨터과학과 박사과정
kwonjh24@hotmail.com
** 숙명여자대학교 정보과학부 교수
yiyoon@sookmyung.ac.kr

지리정보 데이터를 보다 효과적으로 처리하기 위한 연구 중 하나로는 공간 인덱싱 기법에 대한 연구가 있다. 대용량 지리정보 데이터 검색은 일반적으로 공간 인덱싱 기법을 이용해 이루어지게 되는데 알려져 있는 다양한 공간 인덱싱 기법이 존재한다. 그러나, 상세화 정도에 따른 지리정보 데이터를 검색하고자 할 때 기존의 공간 인덱싱 기법을 사용하는 경우 문제가 발생하게 된다. 기존의 공간 인덱싱 기법을 사용하여 단계별 데이터를 검색하기 위해서는 2가지 방법이 존재한다. 첫째, 상세화 단계별로 각각의 인덱싱 구조로 저장한다. 이러한 방법은 데이터의 중복 문제를 일으킨다[2]. 둘째, 한 개의 인덱싱 구조에 저장한 후 뷰 관점에서 다르게 가져와 사용하는 방법이다. 이는 각 단계별 데이터 검색에 따른 비효율성 문제가 발생한다.

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

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구를 살펴보고, 제 3장에서는 단계화된 지리정보 데이터 검색을 위한 공간 인덱싱 구조와 알고리즘을 기술한다. 제 4장에서는 제안된 기법에 대한 구현과 이에 대한 성능 평가 결과를 설명하고, 마지막으로 제 5장에서 결론을 맺는다.

2. 관련 연구

2.1 단계화된 지리정보 데이터

지리정보시스템은 지도를 기반으로 하는 정보시스템이므로 지도의 축척을 지원해야 한다. 지도는 주어진 축척에 따라 그려진다. 동일한 정보라 하더라도 다른 축척에 따라 다르게 그려져야 한다. 이를 위해 소축척에서 나타나는 지도가 대축척에서 나타나는 지도와 상세화 된 내용에 있어 달라져야 하며 이는 단계별 상세화(Level Of Detail)로 알려진다. 단계화된 지리정보 데이터에 대한 연구는 지도 제작자들에 의해 오랫동안 연구되어 왔으며 이는 지도 일반화로 알려진다. 또한 이러한 개념을 지원하기 위해서는 다중 표현(multiple representaion)이 요구된다[2,3,4]. 이러한 단계화된 지리정보 데이터의 목적은 인간의 해석 능력과 분석 능력에 적당하도록 상세화된 정도를 감소하고 객체의 밀도를 줄이는데 있다. 화면 확대와 축소 연산은 단계화된 지리정보 데이터를 디스플레이하는데 있어 중요한 연산 중 하나이다. 이는 지능화된 줌(intelligent zoom)연산으로 알려져 있으며, 화면 확대를 통해 데이터의 크기만을 확대하는 것이 아니고 데이터를 상세화한다[5].

단계화된 지리정보 데이터는 지도 일반화 연산을 통해 이루어지며 그 연산자는 다음과 같다[6].

(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 의미 있는 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 여러 개의 객체가 하나의 객체로 표현된다.
(g) 대표화(typification) : 유사한 형태를 가진 많은 수의 별개의 객체들을 동일하고 전형적인 형

태를 가지는 작은 수의 객체로 나타낸다.

(h) 형태변형(anamorphose) : 인접성 충돌 문제를 해결하기 위해 객체 집합을 변형하여 나타낸다.

2.2 공간 인덱스

공간 데이터베이스는 1차원 데이터 뿐 아니라 2차원 이상의 다차원 데이터베이스이다. 따라서 기존의 데이터 처리 방법은 적합하지 않으며 이를 위한 별도의 처리 방법이 있어야 한다. 이러한 연구 분야 중 하나가 공간 인덱싱이다.

공간 인덱싱 기법은 크게 트리 기반과 해쉬 기반 방법으로 분류될 수 있다[7]. 트리 기반 방법은 계층화된 검색 트리를 기반으로 하며 대표적으로 R트리[8,9,10]와 Quad트리[11,12] 등이 있다. 해쉬 기반 방법은 그리드 파일을 기반으로 하며 대표적으로 그리드 파일[13], R파일[14] 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.

R트리는 트리 기반 방법 중 최소 사각형(MBR : Minimum Bounding Rectangle)에 기반한 방법으로 현재 가장 널리 쓰이고 있는 인덱싱 기법이다. 데이터 구조는 중간 노드(non leaf node)와 리프 노드(leaf node)로 구성된 높이 균형 트리이다. 데이터 객체는 리프 노드에 저장되고 중간 노드는 하위 레벨에 해당하는 모든 사각형을 완전히 둘러싸는 사각형이다. M이 각 노드의 최대 엔트리 수라 할 때 R트리는 다음과 같은 속성을 가진다.

(a) 루트 노드는 리프 노드가 아니라면 적어도 2개의 자식을 가진다.

(b) 모든 중간 노드는 루트가 아니라면 M/2에서 M개 사이의 자식을 가진다.

(c) 모든 리프 노드는 루트가 아니라면 M/2에서 M개 사이의 자식을 가진다.

(d) 모든 리프는 같은 레벨에 나타난다.

2.3 단계화된 지리정보 데이터를 제한적으로 지원하는 공간 인덱스 : Reactive 트리

단계별 데이터를 지원하는 공간 인덱스로 알려진 기법으로는 Reactive트리와 PR파일이 있다. 이 중, Reactive트리는 R트리에 기반하며 PR파일은 그리드 파일에 기반한다. 그러나, 이러한 공간 인덱싱은 지도 일반화 연산자 중에서 선택 연산자 혹은 단순화 연산자라는 한가지 일반화 연산자만을 제공한다. 따라서 두가지 이상의 일반화 연산자를 사용하고자 하는 경우에는 사용할 수 없다는 문제점을 가진다.

Reactive트리[15]는 R트리에 기반한 단계별 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 중요도 값(importance value)이 추가된다. 셋째, 모든 리프가 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 점과 서로 다른 레벨의 객체들간 겹침이 없어야 한다는 문제점을 가진다.

2.4 단계화 된 지리정보 데이터를 제한적으로 지원하는 공간 인덱스 : PR 파일

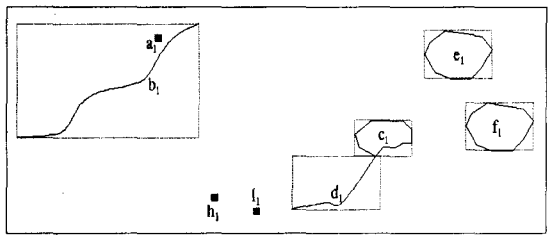
PR(Priority Rectangle) 파일[16]은 그리드 파일 기법의 변형 중 하나인 R파일에 기반한 단계별 데이터를 지원하는 공간 인덱싱 기법 중 하나이다. 이

기법은 R파일과 유사하나 우선순위(priority)를 부여했다는 점이 다르다. PR파일은 디렉토리 구조에 기반하며 이러한 디렉토리가 우선순위가 서로 다른 블록을 레퍼런스 하는 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 몇 개를 선택하는 단순화 알고리즘을 사용한다. 그러나, 단순화 연산자만을 제공한다는 점과 공간 인덱스 중 그리드 파일의 문제점을 그대로 가진다는 문제점을 가진다.

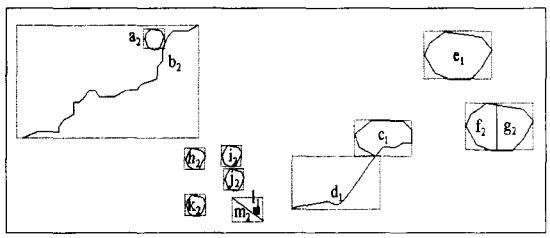
3. 단계화 된 지리정보 데이터 검색을 위한 공간 인덱싱 기법

하나의 지도를 서로 다른 단계의 지도로 저장하는데는 2가지 접근방법이 있다. 첫째, 단계화된 각각의 지리정보 데이터를 별개로 저장한다. 둘째, 가장 상세화 된 지리정보 데이터만을 저장하고 자동화된 일반화 연산자를 통해 단계화 된 지리정보 데이터를 얻는다. 이 중 두 번째 방법은 첫 번째 방법보다 효율적이라는 장점을 가진다. 그러나, 자동화된 일반화 연산자에 대한 연구가 완전하지 않으며 일반화 연산자를 거치게 됨으로써 검색 속도가 떨어질 수 있다는 문제점을 가진다. 따라서 본 논문에서는 첫 번째 접근방법을 사용하여 일반화 연산을 거쳐 미리 만들어진 데이터를 그 대상으로 한다. 2장에서 언급한 바와 같이, 단계화 된 지리정보 데이터 검색을 위한 기존의 공간 인덱싱 기법은 모든 일반화 연산자 중 선택과 단순화 연산자만을 다룬다. 이는 자동화된 일반화 연산자가 상대적으로 단순한 선택과 단순화 등의 몇 가지 연산자를 중점적으로 연구해 왔으며 기존의 공간 인덱싱 기법도 이를 기반으로 하기 때문이다. 예를 들어, [17]에서 제시한 예제를 수정한 그림 1을 살펴보면 단계 1에 존재하는 데이터 a1은 단계 2에서는 데이터 a2로 수정된다. 이는 지도 일반화 연산자 중 심볼화 연산자의 결과이다. 그러나, 이러한 데이터는 기존의 단계화된 지리정보 데이터를 지원하는 공간 인덱스로는 표현이 불가능하다. 그러므로 여기서는 단계화 된 지리정보 데이터 지원을 목적으로 하지 않은 기존의 공간 인덱스만을 고려한다. 본 논문에서는 기존의 공간 인덱싱 기법 중 R트리를 기반으로 하였다. 이는 R트리가 가장 널리 사용되는 공간 인덱스이기 때문에 사용된 것으로, 본 논문의 기본 아이디어는 어떠한 기존 공간 인덱스 구조에도 응용 가능하다.

일반화 연산자는 상세화 된 데이터로부터 간략화 된 데이터를 생성하는 연산자이다. 이러한 일반화 연산자를 통해 생성된 단계별 데이터의 관계를 살펴보는 일은 중요하다. 모든 일반화 연산자에 대해 이를 살펴보면 다음과 같다. 첫째, 선택 연산자는 상세화 된 데이터로부터 일부 데이터 객체를 추출하여 이를 간략화 된 데이터로 생성하는 연산자이다. 즉, 선택 연산자를 사용하여 단계별 데이터가 생성된 경우에는, 상세화 된 데이터는 간략화 된 데이터와 상세화 된 데이터에서 새롭게 추가된 데이터로 이루어진다. 둘째, 선택 연산자 이외의 연산자, 즉, 단순화, 확장, 분류,



(a) 단계 1



(a) 단계 2

(그림 1) 단계화된 지리 정보 데이터의 예

심볼화, 통합, 대표화 연산자는 상세화 된 데이터로부터 간략하게 수정된 데이터를 생성하는 연산자로 요약될 수 있다. 즉, 선택 연산자 이외의 연산자를 사용하여 단계별 데이터가 생성된 경우에는, 상세화 된 데이터는 간략화 된 데이터를 공유하지 않고 상세화 된 데이터로만 구성된다.

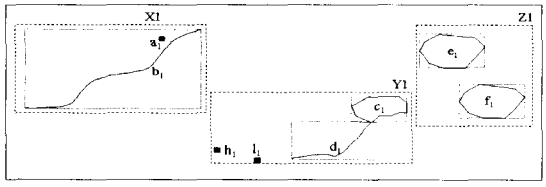
단계화 된 데이터를 지원하는데 있어 단계화 된 지리정보 데이터 지원을 목적으로 하지 않는 기존의 공간 인덱싱 기법을 사용하는 경우는 접근 방법에 따라 다음과 같은 문제점을 가진다. 첫 번째 방법은 단계별로 각각 다른 인덱스에 저장하는 방법에 의해 단계화 된 데이터를 지원하는 것이다. 이러한 접근 방법은 동일한 데이터가 존재하는 경우 이를 각 단계별로 중복 저장하는 문제가 발생한다. 그림 1을 예로 들면, 단계 1과 단계 2에 모두 존재하는 데이터 c1,d1,e1,l1 이 각각 별개의 인덱스 구조에 별도로 중복 저장된다. 두 번째 방법은 단계별 데이터를 하나의 인덱스 구조에 저장하는 것이다. 이 방법은 데이터의 중복은 발생하지 않지만 각 단계에 관계없이 모든 단계의 데이터를 검색하는데 따른 검색 속도 저하의 문제가 발생한다. 그림 1을 예로 들면, 단계 1의 데이터만을 검색하고자 할 때에도 단계 1과 단계 2의 모든 데이터를 검색해야 한다.

본 논문에서는 2가지 접근 방법의 장점을 취해 다음과 같은 방법을 제안한다. 즉, 별개로 각각 저장된 여러 개의 인덱스를 한 개의 인덱스 구조로 통합한다. 그림 2는 이러한 통합된 구조를 보여준다. 그림 2와 3을 통해 단계 1과 단계 2에 존재하는 데이터 c1,d1,e1,l1 이 단계 1에만 저장됨을 알 수 있다. 이를 위해 단계 1의 노드 N2, N5, N6에 단계값 '11'이 기존의 R트리와 달리 추가되었다. 단계값 '11'은 단계 1의 단계값 '01'과 단계 2의 단계값 '10'의 비트OR(bitwise-OR) 연산의 결과로 단계값 1과 단계값 2에서 모두 사용 가능한 노드임을 나타낸다. 따라서, 단계 2에 해당하는 데이터를 검색하고자 할 때에는 단계 1의 노드 중 단계 2의 단계값 '11'을 가지고 있는 노드만을 방문함으로써 데이터의 중복 저장 없이도 검색 속도의 향상을 기대할 수 있게 된다.

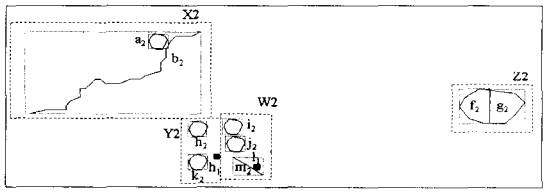
제안된 인덱스 구조는 선택 연산자에 있어 단계값을 가진다는 점에서 기존의 Reactive트리가 가지고 있던 중요도 값과 유사한 속성을 가진다. 그러나, 기존의 Reactive트리는 선택 연산자만을 지원한다는 점과, 각 단계별 객체간 겹침이 발생하면 안된다는 제한점을 가진다는 점에서 제안된 구조는 보다 우수함을 알 수 있다. 다음의 3.1, 3.2, 3.3, 3.4 절에서는 이러한 제안된 새로운 인덱싱 기법에 대한 특징과 알고리즘을 기술한다.

3.1 단계화된 지리정보 데이터 검색을 위한 인덱스의 특징

본 절에서는 제안된 인덱스의 구조와 속성을 기술하도록 한다. 그림 4는 제안된 인덱스의 전체

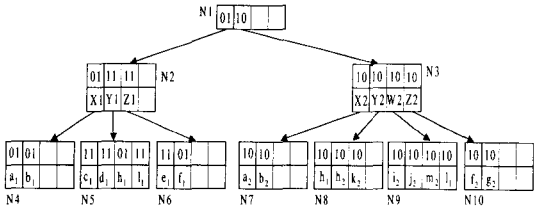


(a) 단계 1

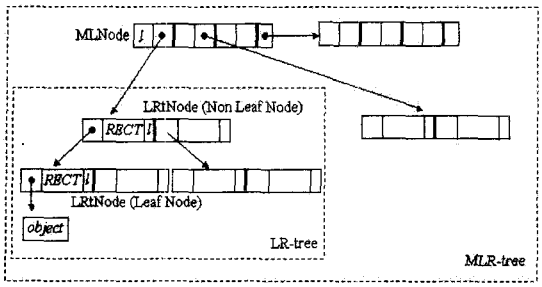


(a) 단계 2

(그림 2) 그림 1에 대한 통합된 인덱스로 만들어진 사각형



(그림 3) 그림 1에 대한 통합된 인덱스로 구성된 인덱스 트리



(그림 4) 인덱스의 전체 구조

구조를 보여준다. 구조는 크게 단계별 R트리를 통합하는 단계값 정보를 가지고 있는 MLNode(Multi Leveled Node)와 각 단계별 데이터를 가진 인덱스 트리에 해당하는 LRtNode(Leveled R-tree Node)로 이루어진다. 이 중, MLNode를 루트 노드로 한 전체 인덱스 트리는 MLR트리(Multi Leveled R-tree)로, MLNode에 의해 포인트된 LRtNode를 루트 노드로 한 트리는 LR트리(Leveled R-tree)로 명명한다.

MLNode는 다음과 같은 형태로 이루어진다.

(*entries, pml*)

여기서 *pml* 은 여러 개의 MLNode가 존재하는 경우, 다음 MLNode를 포인트하는 포인터이며, *entries*는 다음과 같은 형태를 가진다.

(*l, plr*)

여기서 *l* 은 단계값(level value)을 의미하며 *plr* 은 LRtNode의 루트 노드를 포인트하는 포인터이다. LRtNode는 중간 노드와 리프 노드로 구성된다. LRtNode의 중간 노드는 다음과 같은 형태를 가진다.

(*p, RECT, l*)

여기서 *p* 는 LRtNode의 자식 노드를 포인트하는 포인터이고, *Rect*는 하위 노드의 엔트리에 존재하는 모든 사각형을 포함하는 최소 사각형이다. *l* 은 하위 노드의 엔트리에 존재하는 모든 단계값에 대한 비트 OR (bitwise-OR) 연산자를 수행한 결과이다. LRtNode의 리프 노드는 다음과 같은 형태를 가진다.

(*id, RECT, l*)

여기서 *id* 는 데이터 객체를 포인트 하는 포인터이고, *Rect* 는 *id* 에 의해 포인트 된 데이터 객체를 포함하는 최소 사각형이다. *l* 은 해당 객체가 나타나는 모든 단계값에 대해 비트 OR 연산자를 수행한 결과이다.

단계값은 기존의 R트리와 차이가 나는 부분으로, 상세화 된 정도값을 의미한다. 적은 단계값은 보다 간략화 된 데이터를 의미하며 큰 단계값은 상세화 된 데이터를 의미한다. 한 개의 객체가 나타나는 모든 단계값은 하나의 단계값으로 나타나는데 이를 위해 본 논문에서는 비트 연산 기법을 사용한다. 예를 들어, 하나의 객체가 단계 1과 2에 동일하게 나타나는 경우 단계값은 8비트의 비트값으로 표현할 때 '00000011'로 표현된다. 즉, 비트값 '00000011'은 '00000000'로부터 각 단계값에 따라 왼쪽으로 단계값만큼 이동한 결과인 단계값 1에 대한 비트값 00000001과 단계값 2에 대한 비트값 00000010의 비트 OR 연산에 의해 만들어진다. 각 단계에 해당하는 데이터를 검색하고자 할 때는 비트 AND(bitwise-AND) 연산자를 사용한다. 즉 단계값 2에 해당하는 객체를 검색하고자 할 때를 예로 들어보면, 트리 내 모든 노드의 단계 1에 대해 bitwiseAND(1, 00000010)을 수행한다. '00000010'은 단계값 2에 대한 비트값에 해당하며 이에 대한 결과가 단계값 2의 비트값과 동일하면 노드 내 엔트리는 검색된다.

제안된 인덱스 구조인 MLR트리는 다음과 같은 속성을 가진다. 이 때, M은 LRtNode내 엔트리의 최대 수라 가정한다. 여기서 속성 1부터 속성 5까지는 기존의 R트리와 동일한 속성이며, 6부터 9까지의 속성은 기존의 R트리와는 다른 속성이다.

속성 1) LRtNode 내 모든 노드는 MLNode내의 plr에 의해 직접 포인트된 노드가 아니라면 M/2과 M개 사이의 엔트리를 가진다.

속성 2) LRtNode의 중간 노드에 대한 각 엔트리 (*p, RECT, l*)에 대해서 RECT는 하위 노드의 엔트리에 존재하는 모든 사각형을 포함하는 최소 사각형이다.

속성 3) LRtNode의 리프 노드에 대한 각 엔트리 (*id, RECT, l*)에 대해서 *RECT*는 *id* 에 의해 포인트된 데이터 객체를 포함하는 최소 사각형이다.

속성 4) MLNode내의 *plr* 에 의해 직접 포인트된 노드가 리프 노드가 아니라면 2개 이상의 자식 노드를 가진다.

속성 5) MLNode내의 *plr*에 의해 포인트된 노드에 의해 루트가 된 트리는 높이 균형 트리이다.

속성 6) 전체 트리는 높이 균형 트리가 아니다.

속성 7) 데이터 객체는 MLNode 내 각 엔트리 (*l, plr*)에 대해, 해당 데이터 객체가 나타나는 가장 작은 단계값 *l* 에 해당하는 엔트리의 *plr* 에 의해 포인트된 노드에 의해 루트가 된 트리에 나타난다.

속성 8) MLNode 내 각 엔트리 (*l, plr*)은 적은 단계값 *l* 에서 큰 단계값 *l* 순으로 정렬된다

속성 9) LRtNode의 각 엔트리(*p, RECT, l*)에 대해, 단계값 *l* 은 *p* 에 의해 포인트된 모든 엔트리의 *l* 에 대해 비트 OR 연산자를 수행한 결과이다.

3.2 단계화된 지리정보 데이터 검색 방법

본 절에서는 단계화 된 지리정보 데이터를 위한 검색 알고리즘을 기술한다. 검색 알고리즘은 기존의 R트리와 유사하나, MLNode와 LRtNode의 엔트리 내 단계값을 검색한다는 점에서 차이가 존재한다. 검색 방법을 간략히 기술하면 다음과 같이 요약된다. 우선 검색 대상이 되는 단계값과 동일한 단계값을 가지는 LRtNode의 루트 노드를

```
Algorithm Search (MLR, W, L)
Input
  MLR : An tree rooted at node MLR
  W : search window W
  L : search level L
Output
   All data objects in level L overlapping W
Method
1.  result = {}
2.  for (each MLNode of  MLR)
3.     for (each entry(l, p_lr) of MLNode)
4.        if (l <= L)
5.           result = result + SearchLRtree(LR, W, L),
                    where LR is the node pointed by p_lr
6.        else  goto line 7
7.  return result

Algorithm SearchLRtree (LR, W, L)
Input
  LR : Leveled R-tree rooted at node LR
  W : search window
  L :  search level value L
Output
  All data objects in level L overlapping W
Method
1.  if (LR is not a leaf node)
2.     for (each entry(p, RECT, l) of LR)
3.        if ( bitwiseAND(l, L) = L)
4.           if (RECT overlaps W)
5.              SearchLRTree(CHILD, W  Rect, L),
                    where CHILD is the node pointed by p
6.  else
7.     for (each entry(p,RECT,l) of LR)
8.        if ( bitwiseAND(l, L) = L)   return all objects overlapping W
```

(그림 5) 단계화된 지리정보 데이터 검색 알고리즘

MLNode로부터 검색한다. 다음으로, 이렇게 검색된 각 LR트리에서 검색 대상이 되는 단계값과 동일한 단계값을 가지는 모든 노드를 검색하여 최종적으로 해당 데이터 객체가 발견된다. 이를 나타내면 그림 5와 같다.

3.3 단계화된 지리정보 데이터 삽입 방법

본 절에서는 단계화된 지리정보 데이터를 위한 삽입 알고리즘을 기술한다. 삽입 방법을 간략히 기술하면 다음과 같이 2가지 경우로 요약된다. 첫 번째 경우는, 간략화된 데이터로부터 데이터가 수정되는 경우이다. 이는 단계값을 처리하는 부분을 제외하고는 R트리와 동일하다. 두 번째 경우는, 간략화된 데이터를 공유하여 사용하는 경우이다. 이 때는 해당 데이터를 소유하고 있는 가장 간략화된 데이터의 단계값에 해당하는 LR트리 내 노드의 단계값에 해당 단계의 단계값을 비트

OR 연산자에 의해 추가한다.

3.4 단계화된 지리정보 데이터 삭제 방법

본 절에서는 단계화된 지리정보 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 단계값을 처리하는 부분을 제외하고는 R트리의 삭제 방법과 동일하다. 두 번째 경우는, 간

Algorithm Insert (MLR, E, L, CL)
Input
MLR : tree rooted at node MLR
E : index entry E (*id*, *RECT*)
L : level value
CL : coarser level-value, where CL is UNDEFINED if a coarser level does not exist
Output
The new MLR-tree that results after the insertion of E
Method
1. if (CL is UNDEFINED)
2. if (an entry(l, p_{lr}) does not exist in entry of MLNodes of MLR, where l = L)
3. create a new Leveled R-tree node, NewLR
4. insert an entry(L, p_{lr}) in MLNodes, where plr is pointing the NewLR and each entry is sorted from coarse level to detailed level
5. find an entry(l, p_{lr}) in MLNodes of MLR, where l = L
6. InsertLRtree(LR, E, L), where LR is the node pointed by p_{lr}
7. else
8. if (an entry(l, p_{lr}) does not exist in entry of MLNodes of MLR, where l = CL) return failure
9. find an entry(l, p_{lr}) in MLNodes of MLR, where l = CL
10. InsertLowLevelLRtree(LR, E, CL), where LR is the node pointed by p_{lr}

Algorithm InsertLRtree (LR, E, L)
Input
LR : Leveled R-tree rooted at node LR
E : index entry (*id*, *RECT*)
L : level value
Output
The new Leveled R-tree that results after the insertion of E in level L
Method
1. find a leaf node LN in which to place a new index entry E
2. if (LN has room for another entry) insert (E, L) in a LN
3. else SplitNode(LN, E, L)
4. AdjustLRTree(LN, LR), where the method is similar to algorithm 'AdjustTree' of R-tree. The difference with AdjustTree algorithm is propagating changes by bitwiseOR(l,L) calculation ascending from leaf nodes to LR, where l is an element of entries(p,*RECT*,l) of nodes of the Leveled R-Tree.
5. if (root is splited)
create a new root whose children are the two resulting nodes, propagating changes by bitwiseOR calculation for l of entries(p,*RECT*,l) of two resulting nodes.

Algorithm InsertLowLevelLRtree (LR, E, CL),
Input
LR : Leveled R-tree rooted at node LR
E : index entry(*id*, *RECT*)
CL : level value
Output
The new Leveled R-tree that results after the insertion of E in level CL
Method
1. search a leaf node LN in which to place a new index entry E
2. perform bitwiseOR(l, CL) calcuation for l of entries(p,*RECT*,l) of LN
3. propagate changes by bitwiseOR calculation, ascending from a leaf node LN to the node LR

(그림 6) 단계화된 지리정보 데이터 삽입 알고리즘

략화된 데이터를 공유하여 사용하는 경우이다. 이 때는 해당 데이터를 소유하고 있는 가장 간략화된 데이터의 단계값에 해당하는 LR트리 내 노드의 단계값에 해당 단계의 단계값을 AND비트 연산자에 의해 삭제한다.

4. 구현 및 실험

이 장에서는 본 논문에서 제시하는 단계화된 지리정보 데이터 검색을 위한 인덱스에 대한 구현 및 실험을 기술한다. 이를 위해 제안된 인덱스 구조인 MLR트리를 R트리를 사용한 방법과 비교한다.

4.1 구현 환경

본 논문에서 제시한 단계화된 지리정보 데이터 검색을 위한 인덱스의 구현 환경은 다음과 같다. 구현 언어는 윈도우즈에서 구동되는 Cygwin 기반하에 GNU C++을 사용했다. Cygwin은 래드 햇(Red Hat)에 의해 개발된 Windows 운영체제에서의 Unix 환경이다[18]. Cygwin은 표준 Unix와 Linux 환경을 그대로 따르므로 Unix와 Linux환경으로의 이식이 원활히 이루어져 운영체제에 관계없이 동일한 개발이 가능하다는 장점을 가진다. 실험을 위해 본 논문에서는 R트리와 제안된 인덱스 구조 모두를

```
Algorithm Delete (MLR, E, L)
Input
  MLR : MLR-tree rooted at node MLR
  E : index entry E(id, RECT)
  L : level value
Output
  The new MLR-tree that results after the deletion of E in level L
Method:
1.   for (each MLNode of  MLR)
3.       for (each entry(l, p_l) of MLNode)
4.           if (l <= L)
5.  .          find a leaf node LN in which to place an entry(id, RECT, l), where e=E
6.               if (l = L)
7.                   DeleteEntryLRtree(LR, E, LN, L),where LR is the node pointed by p_l
8.               else
9.                   DeleteLevelLRtree(LR, E, LN, L), where LR is the node pointed by p_l
10.              return the new MLR-tree that results after the deletion of E


Algorithm DeleteEntryLRtree (LR, E, LN, L)
Input
  LR : Leveled R-tree rooted at node LR
  E : index entry E(id, RECT)
  LN : leaf node containing E
  L : level value
Output
  The new Leveled R-tree that results after the deletion of E in level L
Method :
1. remove E from LN
2. CondenseLRTree(LN, LR), where the method is similar to algorithm 'CondenseTree' of R-tree.
   The difference with CondenseTree algorithm is propagating changes by bitwiseOR(l,L) calculation
   ascending from a leaf node LN to the node LR, where l is an element of entries(p,RECT,l) of nodes
   of the Leveled R-Tree.
3.  if (root node of  an Leveled R-tree has only one child)
4.      make the child the new root node of an Leveled R-tree


Algorithm DeleteLevelLRtree (LR, E, LN, L)
Input
  LR : Leveled R-tree rooted at node
  E : index entry(id, RECT)
  LN : leaf node containing E
  L : level value
Output
  The new Leveled R-tree that results after the deletion of E in L
Method
1. perfom bitwiseAND(l, bitwiseNOT(L)) calculation for entry (id, RECT, l) of LN, where e=E
2.  propagate changes of level value ascending from LN to LR
```

(그림 7) 단계화된 지리정보 데이터 삭제 알고리즘

구현하였으며, 구현된 프로그램은 Windows 2000 운영체제에서 펜티엄III 800EB와 256MB 메인 메모리 환경 하에서 실험하였다.

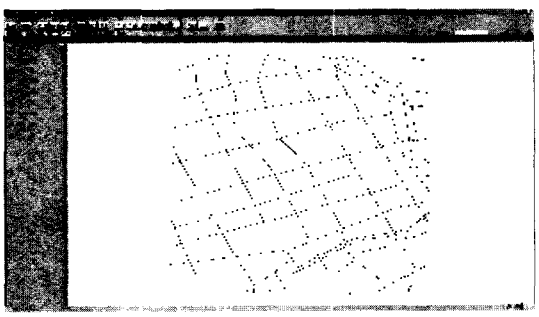
4.2 실험 모델

본 논문에서 사용된 데이터는 서울시 강남구 지역을 대상으로 4개의 축척에 대한 실제 데이터를 대상으로 하였다. 각 단계별 데이터는 표 1로 요약하여 설명하였으며, 이를 ESRI의 ArcView 프로그램을 이용하여 디스플레이한 결과는 그림 8과 같다.

제안된 기법의 성능을 보다 객관적으로 실험하기 위해 다양한 크기와 많은 수의 질의 윈도우를 그 대상으로 하였다. 단계화된 지리정보 데이터를 검색하는데 있어서, 각 객체는 일정 윈도우 크기 범위에서만 해당 단계의 객체가 나타난다. 이 때 윈도우 크기와 단계값의 관계를 살펴보면, 윈도우 크기가 커지면 단계값은 적어지게 된다. 이를 위해 단계별 데이터의 윈도우 범위를 10개씩 랜덤하게 생성하였다. 이렇게 생성된 각 윈도우 크기의 범위 내에서 각각 1000개의 질의를 랜덤하게 생성하고 이렇게 생성된 질의를 대상으로 실험하였다.

(표 1) 실험 데이터 요약

단계값	축척	전체 객체수	간략화된 단계의 객체로부터 수정된 객체수	간략화된 단계의 객체와 중복된 객체수	
				객체의 최소 단계값	객체수
1	1:250000	37	37	0	
2	1:50000	393	390	1	3
3	1:5000	17265	17228	1	3
				2	34
4	1:1000	80420	74962	1	3
				2	34
				3	5401



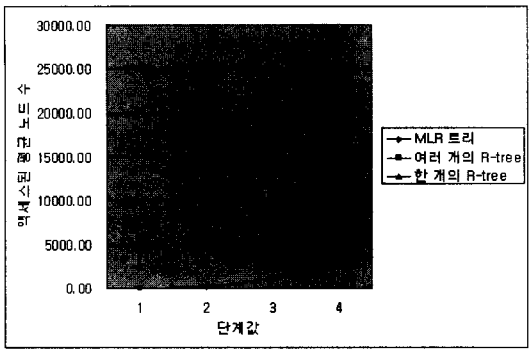
(a) 1:250000

(b) 1:50000

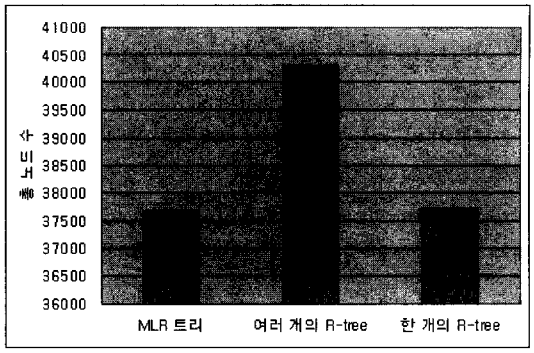
(c) 1:5000

(d) 1:1000

(그림 8) 실험 데이터



(그림 9) 액세스된 평균 노드 수 비교



(그림 10) 소요된 전체 노드 수 비교

4.3 실험 결과

4.2절의 실험 모델에 따라 각 R트리와 제안된 인덱스 기법에 대해 검색 속도의 평균값과 총 메모리 용량을 계산하였다. 단계화된 지리정보 데이터를 R트리에서 나타내기 위해서 단계별로 별도 구축된 R트리의 집합과 한 개의 R트리 내에 단계별 지리정보 데이터를 모두 구축한 경우로 나누어서 그 성능을 비교하였다. 이에 대한 결과는 그림 9와 그림 10과 같다.

검색 속도를 비교하기 위해 본 논문에서는 액세스된 노드의 수를 측정하였다. 이는 데이터 검색에 소요되는 응답 시간이 외부 요인에 따라 변화가 생길 수 있는 데 비해 액세스된 인덱스 노드의 수를 측정하는 것이 보다 객관적이기 때문이다. 그림 9에서는 단계별로 액세스된 노드 수의 변화를 꺾은선 그래프로 각 인덱스 구조에 따라 비교하여 나타내었다. 이를 통해 다음과 같은 결과를 얻는다. 첫째, 제안된 인덱스 구조는 하나의 R트리로 구축된 경우보다 늘 성능이 우수하다. 둘째, 제안된 인덱스 구조는 단계별로 별도 구축된 R트리의 집합과 성능면에서 거의 동일하다.

그림 10은 메모리 용량에 대한 성능 평가 결과를 보여준다. 메모리 용량을 비교하기 위해 본 논문에서는 각 인덱스에서 소요된 전체 노드 수를 측정하였다. 이를 위해, 각 인덱스에 데이터를 삽입할 때마다 액세스된 노드 수를 계산하였다. 이렇게 계산된 결과는 각 인덱스 구조별로 소요된 전체 노드 수를 막대 그래프로 표시하였다. 이를 통해 다음과 같은 결과를 얻는다. 첫째, 제안된 인덱스 구조는 다른 인덱스 구조 중 소요되는 메모리 용량이 가장 작다. 둘째, 검색 속도 측면에서 우수한 성능을 보였던 단계별 R트리의 집합은 최악의 결과를 얻었다. 이는 단계별 데이터간 중복되는 데이터를 중복하여 저장하는 데 따른 결과이다. 이에 비해 제안된 기법은 중복되는 데이터를 통합관리하여 이러한 문제를 해결하였다.

그림 9와 그림 10을 통해 제안된 인덱스 기법은 알려진 기존의 공간 인덱싱 기법에 비해 검색 속도와 메모리 용량 모두에서 우월함을 알 수 있다. 따라서, 단계별 지리정보 데이터 검색에 있어 매우 효율적인 인덱스 구조임을 알 수 있다.

5. 결 론

지리정보시스템의 대중화와 이에 따른 인터넷 지리정보시스템의 급속한 요구에 따라 효율적인 지리정보 데이터 검색에 대한 필요성이 매우 높아지고 있다. 이를 위해 본 논문에서는 단계화된 지리정보 데이터 검색의 필요성과 이에 필요한 공간 인덱싱 기법을 제안하였다. 기존의 공간 인덱싱 기법은 단계화된 지리정보 데이터 검색을 목적으로 하지 않아 비효율적이며, 이를 지원하는

몇가지 공간 인덱싱 기법도 모든 경우의 단계화된 지리정보 데이터를 지원할 수 없다는 문제점을 가진다.

이에 비해, 본 논문에서 제안된 인덱싱 기법은 다음과 같은 장점을 가진다. 첫째, 기존의 단계화된 지리정보 데이터 검색을 목적으로 한 공간 인덱싱 기법과는 달리 모든 경우의 단계화된 지리정보 데이터를 지원한다. 둘째, 단계화된 지리정보 데이터 검색을 목적으로 하지 않은 기존의 공간 인덱싱 기법에 비해 검색 속도와 메모리 용량 모두에서 효율적이다. 셋째, 알고리즘이 단순하여 어떠한 공간 인덱스에도 쉽게 적용이 가능하다. 향후 연구과제로는 단계별 데이터간 일관성을 고려하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] "Automatic generalization on geographic data", project report, VBB Viak, 1997.

[2] Clodoveu A.Davis Jr., Alberto H. F. Laender, "Multiple Representations in GIS : Materialization, Geometric, and Spatial Analysis Operations", Proceedings of the 7th international symposium on advances in GIS, pp. 60~65, 1999.

[3] Stefano Spaccapietra, Christine Paren, Christelle Vangenot, "GIS Databases : From Multiscale to MultiRepresentation", Symposium on Abstraction, Reformulation and Approximation, pp. 57~70, 2000.

[4] Michael Garland, "Multiresolution Modeling : Survey&Future Opportunities", Eurogra-phics, State of the Art Report, 1999.

[5] Sabine Timpf, "Cartographic objects in a multiscale data structure", in geographic information research:Bridging the Atlantic. Craglia,M., & Couclelis, H., eds., 1 vols., Vol.1, London, Taylor&Francis, pp. 224~234, 1997.

[6] Ruas, A. "Multiple representation and generalization", Lecture Notes for "sommarkurs : kartografi", 1995.

[7] Volker Gaede, Oliver Gunther, "Multi-dimensional access methods", ACM Computing Surveys, Vol.30, No.2, June 1998.

[8] Guttman, A. "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.

[9] Sellis,T., Roussopoulos, N., Faloutsos,C. "The R+-tree : A dynamic index for multi-dimensional objects", Proceedings of the 13th International conference on Very Large DataBases, brighton, England, pp. 507~518, 1987.

[10] Beckmann, N., Kriegel, H., Schneider,R, Seeger, B. "The R* tree : An efficient and robust access method for points and rectangles", In Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 322~331, Atlantic City, NJ, 1990.

[11] Hanan Samet, "The quadtree and related hierarchical data structure", ACM Computing Surveys, 16(2), pp. 187~260, 1984.

[12] Hanan Samet, Webber, R.E, "Storing a collection of polygons using quadtrees", ACM Transactions. Graph, 4(3), pp. 182~222, 1985.

[13] Nievergelt, J., Hinterberger, H., and Sevcik, K.C. "The grid file : An adaptable, symmetric multikey file structure", ACM Transactions on Database Systems. 9(1), pp. 38~71, 1984.

[14] Hutflesz, A., Six, H.-W., Widmayer, P, "The R-file : An efficient access structure for proximity queries", Proceedings of IEEE 6th International Conference on Data Engineering, pp. 372~379, 1990.

[15] Peter van Oosterom, "The Reactive-Tree : A yt Geographic database", in Auto Carto, 10, pp.

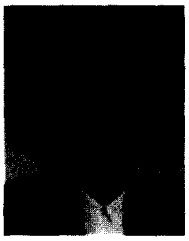
393~407, 1991.

[16] Bruno becker, Peter Widmayer, "Spatial priority search : An Access Technique for Scaless Maps", Proceedings of ACM SIGMOD, Denver, Colorado, pp. 128~137, 1991.

[17] Michela Bertolotto, Max J.Egenhofer, "Progressive Vector Transmission", 7th ACM Symposium on Advances in Geographic Information Systems, pp. 152~157, Nov. 1999.

[18] http://www.cygwin.com.

◐ 저 자 소 개 ◑



권 준 희

1992년 숙명여자대학교 전산학과 졸업(학사)
1994년 숙명여자대학교 대학원 전산학과 졸업(석사)
1999년 숙명여자대학교 대학원 컴퓨터과학과(박사)
1994년~현재 : 쌍용정보통신 GIS기술팀 과장
2000년 전자계산조직응용기술사
관심분야 : 공간데이터베이스, GIS, 멀티미디어 데이터베이스, 컴포넌트, 객체지향 모델링 및 방법론
E-mail : kwonjh24@hotmail.com

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
E-mail : yiyoon@sookmyung.ac.kr