MLR 트리 : 다중 레벨 지리정보 데이타의 윈도우 질의를 위한 공간 인덱싱 기법

(MLR-tree : Spatial Indexing Method for Window Query of Multi-Level Geographic Data)

권 준 희 † 윤 용 익 ††
(Joon-hee Kwon) (Yong-ik Yoon)

요 약 다중 레벨 지리정보 데이타는 화면 확대와 축소와 같은 윈도우 질의를 통해 다루어질 수 있다. 다중 레벨 지리정보 데이타를 효율적으로 다루기 위해서는 이러한 윈도우 질의를 지원하는 공간 인덱싱 기법이 필요하다. 그러나, 기존의 전통적인 공간 인덱싱 기법은 다중 레벨 지리정보 데이타를 액세스하는데 비효율적이다. 이를 위해 다중 레벨 지리정보 데이타를 위한 몇 가지 공간 인덱싱 기법이 알려진다. 그러나, 이 공간 인덱싱 기법은 모든 유형의 다중 레벨 지리정보 데이타를 지원하지 못한다는 문제점을 가진다. 본 논문에서는 다중 레벨 지리정보 데이타의 윈도우 질의를 위한 공간 인덱싱 기법, MLR 트리를 제안한다. MLR 트리는 우수한 검색 성능을 보이면서도 데이타 중복성이 발생하지 않으며, 이를 실험을 통해 보인다. 이 외에도 MLR 트리는 모든 유형의 다중 레벨 지리정보 데이타를 지원한다.

키워드 : 다중 레벨, 공간 인덱싱 기법 GIS, R 트리

Abstract Multi-level geographic data can be manipulated by a window query such as a zoom operation. In order to handle multi-level geographic data efficiently, a spatial indexing method supporting a window query is needed. However, the conventional spatial indexing methods are not efficient to access multi-level geographic data quickly. To solve it, other a few spatial indexing methods for multi-level geographic data are known. However these methods do not support all types of multi-level geographic data. This paper presents a new efficient spatial indexing method, the MLR-tree for window query of multi-level geographic data. The MLR-tree offers both high search performance and no data redundancy. Experiments show them. Moreover, the MLR-tree supports all types of multi-level geographic data.

Key words : Multi-Level, Spatial Indexing Method, GIS, R-tree

1. 서 론

정보화 기술의 비약적인 발전과 더불어 GIS (Geographic Information Systems)의 대중화에 따라, 지도와 같은 지리정보 데이타의 사용이 보다 다양화되고 광범위해지고 있다. 이에 따라 과거의 단순한 지리정보 데이타의 사용을 넘어 보다 높은 품질의 지리정보 데이타의 사용과 대용량 지리정보 데이타에 대한 효율적인 처리가 큰 문제로 대두되고 있다.

지리정보 데이타는 전통적인 종이 지도를 디지털 지도화한 것이다. 전통적인 종이 지도에는 축척(scale) 개념이 존재하는데, 이를 통해 대축척의 데이타는 소축척의 데이타와 비교할 때 지도에서 더 좁은 지역을 자세하게 나타낸다[1]. 이러한 축척별 데이타는 인간의 해석 능력과 분석 능력에 적합하도록 데이타를 간략화하는 것이다.

이는 디지털 지도로 변환된 후에도 마찬가지로 생각해볼 수 있다. 따라서, 넓은 영역을 보는 경우는 단순화되고, 좁은 영역을 보는 경우는 보다 상세하게 나타나는 다중 레벨 데이타가 필요하다. 이는 GIS에 있어 많이 사용되고 있는 확대, 축소 연산에 있어서도 객체를 확대할 때 객체를 단순히 크게 보여주는 것만으로는 높은 품질의 지도를 기대할 수 없다는 것을 의미한다. 즉, 각 객체가 확대될 때 각 객체는 보다 상세하게 보여져야

† 종신회원 : 경기대학교 정보과학부 교수
kwonjh@kyonggi.ac.kr
†† 종신회원 : 숙명여자대학교 정보과학부 교수
yiyoon@sookmyung.ac.kr
논문접수 : 2002년 5월 14일
심사완료 : 2003년 5월 21일

하고, 중요도가 떨어지는 객체가 보여져야만 한다[2].

그러나, 이러한 다중 레벨 지리정보 데이타를 검색하고자 할 때 기존의 공간 인덱스를 사용하는 경우 문제가 발생하게 된다. 기존의 공간 인덱스를 사용할 경우의 접근 방법은 레벨별로 각각의 인덱스 구조로 저장하거나, 혹은 한 개의 인덱스 구조에 저장하는 방법이 있다. 그러나 이러한 방법은 각각 데이타 중복과 검색 저하라는 문제가 발생한다.

기존의 인덱스 구조에 따르는 문제점을 극복하기 위해서는 다중 레벨 지리정보 데이타를 위한 별도의 공간 인덱싱 기법이 요구된다. 그러나, 이러한 기존 연구는 비교적 활발하지 못했으며 몇 가지 연구에 있어서도 다중 레벨 데이타의 유형에 제한을 두었다. 즉, 기존의 몇 가지 연구는 모든 지도 일반화 연산자 중 선택 연산자와 단순화 연산자만을 지원함으로써, 모든 유형의 다중 레벨 지리정보 데이타에는 적용이 불가능하다는 문제점을 가진다.

본 논문에서는 이러한 문제점을 극복하고자 다중 레벨 지리정보 데이타의 모든 유형에 적용할 수 있는 윈도우 질의를 위한 효율적인 공간 인덱스 기법인 MLR 트리를 제안하였다. 제안된 방법에서는 레벨별 다중 인덱스가 한 개의 인덱스로 통합된다. 이를 통해 다중 레벨 지리정보 데이타를 지원하는데 있어 발생하는 데이타 중복 문제와, 검색 저하라는 문제점 모두를 해결하였다. 또한, 기존 몇 가지 다중 레벨 지리정보 데이타를 지원하는 공간 인덱싱 기법의 문제점을 극복하여, 모든 유형의 다중 레벨 지리정보 데이타를 지원할 수 있다는 장점도 가진다. 또한 가상 데이타와 실제 데이타에 대한 광범위한 실험을 통해 이를 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구와 이에 대한 문제점을 살펴본다. 3장에서는 다중 레벨 지리정보 데이타를 지원하는 새로운 공간 인덱스인 MLR 트리를 제안하고 그 구조와 처리방법을 설명한다. 4장에서는 본 논문에서 제안한 MLR 트리의 성능 평가를 위하여 실험을 수행하고 그 결과를 기술한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 전통적인 공간 인덱싱 기법

지리정보 데이타를 처리하기 위한 공간 인덱싱 기법은 크게 트리 기반과 해쉬 기반 방법으로 분류될 수 있다[3]. 트리 기반 방법은 계층화된 검색 트리를 기반으로 하며 대표적으로 R트리[4-7]와 Quad트리[8,9] 등이 있다. 해쉬 기반 방법은 그리드 파일을 기반으로 하며 대표적으로 그리드 파일[10]과 R파일[11] 등이 있다. 이 중 해쉬 기반 방법은 데이타 분포에 의존적이며 오버플로우 발생 및 이에 따라 효율이 저하되는 문제점을 가진다. 따라서 많은 공간 데이타베이스에서는 이러한 해쉬 기반 방법보다 트리 기반 방법을 선호하고 있으며 이 중에서 R트리 방법이 가장 많이 사용되고 있다. 본 논문에서는 이러한 공간 인덱싱 기법을 2.2절의 공간 인덱싱 기법과 구분하여 전통적인 공간 인덱싱 기법으로 부른다.

2.2 다중 레벨 데이타를 위한 공간 인덱싱 기법

전통적인 공간 인덱싱 기법은 다중 레벨 지리정보 데이타를 다루는데 있어 문제점을 보인다. 이를 위해 다중 레벨 지리정보 데이타를 지원하는 공간 인덱스가 몇 가지 알려져 있다. 다중 레벨 데이타를 지원하는 공간 인덱싱 기법으로는 Reactive트리[12], PR 파일[13], 그리고 Multi-scale Hilbert R트리[1]가 있다.

Reactive트리는 R트리에 기반한 구조로 R트리와 유사한 속성을 가지고 있으나 다음과 같은 몇 가지 차이점이 있다. 첫째, 실객체가 리프 노드(leaf node) 뿐 아니라, 중간 노드(non-leaf node)에도 나타난다. 둘째, R트리의 노드 형태에 중요도 값(importance value)이 추가된다. 세째, 모든 리프가 같은 레벨에 나타나지 않는다. 즉, 같은 레벨에 있는 모든 노드는 같은 중요도 값을 가진 엔트리를 포함한다. 이를 통해, Reactive트리는 중요도가 높은 객체가 상위 레벨에 위치하여 중요도가 높은 객체는 보다 빠른 검색이 가능하다. 그러나, 지도 일반화 연산자 중 선택 연산자만을 제공한다는 문제점을 가진다.

PR(Priority Rectangle) 파일은 그리드 파일 기법의 변형 중 하나인 R파일에 기반한 다중 레벨 지리정보 데이타를 지원하는 공간 인덱싱 기법 중 하나이다. PR 파일은 Reactive 트리와는 달리 객체 단위로 저장되지 않는다. PR 파일은 원하는 레벨에 따라 폴리라인(poly-line)으로부터 폴리라인 세그먼트의 끝점 중 몇 개를 선택하는 라인 단순화 알고리즘을 사용한다. PR 파일에서는 객체를 버텍스(vertex)로 분리하고 각 버텍스마다 객체에서의 위치를 나타내는 일련 번호를 부여한다. 부여된 일련 번호와 버텍스는 우선 순위별로 나누어져 그룹화 된다. 그러나, 지도 일반화 연산자 중 단순화 연산자만을 제공한다는 점과 공간 인덱싱 기법 중 그리드 파일의 단점을 그대로 가진다는 문제점을 가진다.

Multi-scale Hilbert R트리는 Hilbert R트리에 기반한 다중 축척 데이타를 지원하는 공간 인덱싱 기법 중 하나이다. Multi-scale Hilbert R트리는 PR 파일과 유사하다. 주요한 차이점으로는 단순화 연산을 수행하는데 따른 검색 속도의 향상을 위해 실제 데이타 파일이 분할되어 저장된다는 것이다. 단순화 연산자를 지원하기 위해 사용되는 단순화 알고리즘은 Douglas-Peuker알고

리즘[14]을 수정하여 사용한다. 선택 연산자를 지원하기 위해서는 데이타 크기에 기반하여 객체를 선택한다. 그러나, Multi-scale Hilbert R트리는 지도 일반화 연산자 중 단순화와 선택 연산자만을 제공한다는 문제점을 가진다.

3. MLR 트리

3.1 기존 연구의 문제점

다중 레벨 지리정보 데이타를 저장하기 위해서는 2가지 방법이 존재한다. 첫 번째 방법은, 레벨별로 상세화된 지리정보 데이타를 레벨별로 각각 별개의 지리정보 데이타로 저장한다. 두 번째 방법은, 가장 상세화된 지리정보 데이타만을 저장하고 자동화된 일반화 연산자를 통해 레벨별 데이타를 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[2]. 둘째, 일반화 연산자를 거치게 됨으로써 검색 속도가 저하된다는 문제점을 가진다. 따라서 본 논문에서는 첫 번째 접근방법을 사용하여 일반화 연산을 거쳐 미리 만들어진 데이타를 그 대상으로 한다.

여기에 비해 그 동안의 다중 레벨 지리정보 데이타를 위한 공간 인덱스는 두 번째 접근 방법을 기반으로 하고 있어 모든 일반화 연산자 중 선택과 단순화 연산자만을 다룬다는 문제점을 가진다.

따라서, 모든 유형의 다중 레벨 데이타를 지원하기 위해서는 다중 레벨 지리정보 데이타 지원을 목적으로 하지 않는 전통적인 공간 인덱싱 기법만을 사용할 수 있다. 이를 위해서는 다음과 같은 2가지 접근 방법이 존재한다. 첫째, 다중 레벨 지리정보 데이타가 별개의 레벨별 인덱스 구조에 저장되는 방법이다. 이러한 방법은 레벨별 데이타 검색시 빠르다는 장점을 가지나, 레벨별로 동일한 데이타가 존재하는 경우 이를 중복 저장하는 문제점을 가진다. 둘째, 다중 레벨 데이타를 레벨 구분 없이 하나의 인덱스 구조에 저장하는 방법이다. 이러한 방법은 데이타의 중복은 발생하지 않지만, 특정 레벨에 해당하는 데이타를 검색하고자 할 때도 모든 레벨의 데이타를 검색함으로써 검색 속도 저하의 문제가 발생한다.

본 논문에서는 위에서 언급한 기존 연구의 문제점을 극복하기 위해, 모든 유형의 다중 레벨 지리정보 데이타의 윈도우 질의를 지원하는 효율적인 공간 인덱스, MLR 트리를 제안한다. MLR 트리는 다중 레벨 지리정보 데이타를 부분적으로 지원하던 기존의 공간 인덱싱 기법과는 달리 다중 레벨 지리정보 데이타의 유형에 제한을 두지 않는다. 또한, MLR 트리는 다중 레벨 지리정보 데이타 지원을 목적으로 하지 않는 전통적인 공간 인덱싱 기법의 2가지 접근 방법이 가진 장점을 통합한 새로운 인덱스 구조이다.

3.2 MLR 트리의 구조

MLR 트리는 다중 레벨 노드(multi-level node)와 복합 레벨값을 가진 R 트리인 LR 트리(Leveled R-tree)로 구성된다. 레벨별로 독립적으로 만들어진 LR 트리는 다중 레벨 노드에 의해 레벨별로 중복된 데이타 없이 하나의 인덱스로 통합되어 모든 레벨의 객체를 하나의 인덱스로 관리할 수 있게 하며, 그 구조는 다음과 같다.

(a) 루트 노드 : (*entries*, p_l)

루트 노드는 다중의 LR 트리를 포인트하는 다중 레벨 노드로 p_l은 여러 개의 다중 레벨 노드가 존재하는 경우, 다음 다중 레벨 노드를 포인트하는 포인터이며, *entries*는 (l, p_{LR}) 형태를 가진다. 여기서 l은 각 레벨값에 해당하는 비트화된 레벨값을 의미하며 p_{LR}은 복합 레벨값을 가진 R 트리의 루트 노드를 포인트하는 포인터이다.

(b) LR 트리 중간 노드 : (*p*, *RECT*, *cl*)

*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.

(c) LR 트리 리프 노드 : (*id*, *RECT*, *cl*)

*id*는 데이타 객체를 포인트하는 포인터이고, *RECT*는 *id*에 의해 포인트된 데이타 객체를 포함하는 최소 사각형이다. *cl*은 해당 객체가 나타나는 모든 비트화된 레벨값에 대해 비트 OR 연산자를 수행한 복합 레벨값이다.

MLR 트리는 별개로 저장된 레벨별 다중 인덱스를 한 개의 인덱스로 통합하기 위해 복합 레벨값(composite level value)을 제안한다. 제안된 복합 레벨값은 기존의 R 트리와 차이가 나는 부분으로, 하나의 객체가 나타나는 모든 레벨값을 의미한다. 적은 레벨값은 보다 간략화된 데이타를 의미하며 큰 레벨값은 상세화된 데이타를 의미한다. MLR 트리의 복합 레벨값을 위해 본 논문에서는 비트 연산자를 사용한다. 이는 1개 이상의 레벨값을 하나의 작은 저장 공간에 저장하면서도, 비트 연산자의 하드웨어 연산에 따른 연산 성능의 낭비를 줄일 수 있기 때문이다.

한 개의 객체가 나타나는 모든 레벨값은 하나의 복합 레벨값으로 나타나는데 이를 위해 본 논문에서는 비트 연산 기법을 사용한다. 예를 들어, 하나의 객체가 레벨 1, 2에 모두 나타나는 경우 복합 레벨값은 2비트의 비트값으로 '11'로 표현된다. 즉, 비트값 '11'은 '01'로부터 각 레벨값에 따라 왼쪽으로 레벨값-1 만큼 이동한 결과 즉,

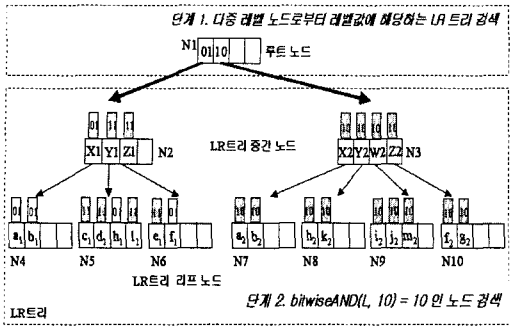


그림 1 MLR 트리의 구조와 검색 질의 처리 과정 예

복합 레벨값 '11'은 레벨값 1에 대한 비트값 '01'과 레벨값 2에 대한 비트값 '10'의 비트 OR 연산에 의해 만들어진다. 이렇게 구성된 복합 레벨값으로부터 특정 레벨에 해당하는 데이타를 검색하고자 할 때는 비트 AND (bitwise-AND) 연산자를 사용한다.

그림 1은 제안된 인덱스, MLR 트리의 구조와 검색 질의 처리 과정을 보여준다. 그림 1의 LR 트리에 나타난 알파벳 문자들은 *RECT*를 표현한 것이며, 각 노드별 상단부에 위치한 작은 사각형 내의 '01','10','11'은 *cl*을 표현하고 있다.

즉, "레벨 2에 속한 모든 객체를 찾아라"라는 검색 질의어를 예로 들어보면, 이를 위해 우선 루트 노드를 검색하여 해당되는 LR 트리를 검색하고, 다음으로 bitwiseAND(*l*,'10')='10'인 노드를 찾는다. 이에 따라 검색될 노드는 '11'과 '10'의 복합 레벨값을 가진 LR 트리 내 노드로 국한되며 이를 그림 1에 음영으로 색칠하여 표시하였다.

3.3 MLR 트리의 속성

MLR 트리는 다음과 같은 속성을 가진다. 이 때, M_L은 다중 레벨 노드 내 엔트리의 최대 개수라 하고, M_R은 복합 레벨값을 가진 R 트리 내 엔트리의 최대 개수라 가정한다. MLR 트리는 R 트리를 확장한 구조이므로, R 트리와 동일한 속성과 동일하지 않은 속성을 동시에 가진다. 즉, MLR 트리 중 LR 트리에 대해서는 복합 레벨값을 제외하고는 R 트리와 기본적으로 속성이 동일하다고 볼 수 있다. 이에 비해, 다중 레벨 노드 및 LR 트리 내의 복합 레벨값에 대한 속성은 MLR 트리만의 고유한 속성이다. 이 중 R 트리와 다른 속성은 다음과 같다.

(a) 전체 트리는 높이 균형 트리가 아니다.

(b) 다중 레벨 노드 내의 모든 노드는 1과 M_L 개 사이의 엔트리를 가진다.

(c) 데이타 객체는 다중 레벨 노드 내 각 엔트리(l, p_{LR})에 대해, 해당 데이타 객체가 나타나는 비트화된 레벨값 l 중 최소값에 해당하는 엔트리의 p_{LR}에 의해 포인트된 노드에 의해 루트가 된 LR 트리에 나타난다.

(d) 다중 레벨 노드 내 각 엔트리(l, p_{LR})은 최소값 l로부터 최대값 l순으로 정렬된다.

MLR 트리는 다중 레벨 노드를 통해 레벨별 데이타를 하나의 노드로 통합 관리할 수 있게 한다. 또한 각 레벨별 트리는 복합 레벨값의 사용으로 레벨별 데이타의 중복 없이도 데이타간 레벨의 구분이 가능해지는 효과를 가진다. 즉, 레벨별로 중복된 데이타는 가장 간략화된 최상위 레벨에만 오직 한번 저장되고 다른 레벨에서 중복해서 사용할 때는 복합 레벨값을 사용하여 각 레벨을 구분한다. 또한, 복합 레벨값에 대한 비트 연산자의 사용으로 적은 저장 공간을 차지하면서도 비트 연산의 빠른 처리 속도에 따라 연산에 따른 처리 속도의 저하 문제를 최소화하고 있다.

3.4 MLR 트리의 처리 방법 : 검색

검색 방법은 R 트리와 유사하지만 다중 레벨 노드와 LR 트리의 복합 레벨값을 검색한다는 점에서 큰 차이점을 보인다. 그 과정은 다음과 같으며, 그림 1에서 그 예를 보여주고 있다.

(a) 단계 1 : 검색하고자 하는 레벨값을 가지는 LR 트리 검색

1. 레벨값을 비트화된 레벨값으로 계산
2. 다중 레벨 노드를 검색하여 계산된 비트화된 레벨값보다 적거나 같은 값을 가지는 엔트리를 찾아 해당되는 LR 트리 추출

(b) 단계 2 : 검색된 각 LR 트리에서 해당 레벨의 검색 영역에 속하는 데이타 검색

1. 단계 1을 통해 검색된 각 LR 트리에서 검색 대상이 되는 레벨값을 포함하는 복합 레벨값을 가지는 노드인지 검색
2. (b) 1을 통해 검색된 노드를 비트 AND 연산자를 사용하여 검색 영역에 속하는지 검색
3. (b) 1과 (b) 2를 각 LR 트리 루트 노드로부터 리프 노드까지 순환하여 방문

검색 과정에 있어 특징적인 부분 중 하나는 다중 레벨의 데이타를 통합된 하나의 데이타로 간주하여 사용자들의 전처리 과정이 필요하지 않다는 것이다. 즉, 레벨별 R 트리 집합의 경우에서는 각 레벨의 데이타를 검색하고자 할 때 해당 레벨의 인덱스를 각 레벨마다 알려주어야 하며, 한 개의 R 트리에서는 각 레벨에 해당하는 데이타를 검색하기 위한 별도의 질의문 혹은 뷰를 구성해야 한다. MLR 트리는 이러한 기존의 방법과는 달리 각 레벨의 데이타를 하나의 인덱스로 통합하여 이러한 과정 없이 다중 레벨 데이타를 액세스할 수 있게 한다.

3.5 MLR 트리의 처리 방법 : 갱신

갱신 방법은 삽입과 삭제라는 2가지 방법으로 구성된다. 이 중 삽입 방법을 우선 간략히 기술하면 다음과 같이 2가지 경우로 요약된다.

(a) 간략화된 상위 레벨의 데이타로부터 데이타가 수정되는 경우, 즉, 선택 연산자 이외의 연산자에 의한 결과로 만들어진 다중 레벨 데이타를 삽입하는 경우

1. R 트리의 삽입 처리 과정 수행
2. 복합 레벨값을 리프 노드에 저장하고 이를 상위 노드로 전파

(b) 간략화된 상위 레벨의 데이타를 공유하여 사용하는 경우, 즉, 선택 연산자에 의한 결과로 만들어진 다중 레벨 데이타를 삽입하는 경우

1. 해당 데이타를 소유하고 있는 가장 간략화된 레벨에 해당하는 LR 트리 검색
2. 검색된 LR 트리 내 노드의 복합 레벨값에 해당 레벨의 레벨값을 비트OR 연산자에 의해 추가

다음으로, 삭제 방법을 간략히 기술하면 다음과 같이 2가지 경우로 요약된다.

(a) 데이타가 모든 레벨에서 완전히 삭제

1. R 트리의 삭제 처리 과정 수행
2. 삭제된 노드와 엔트리에 따른 복합 레벨값을 상위 노드로 전파

(b) 특정 레벨로부터 데이타 삭제, 즉, 데이타는 그대로 존재하면서 데이타가 속한 레벨값만을 수정

1. 해당 데이타를 소유하고 있는 가장 간략화된 레벨에 해당하는 LR 트리 검색
2. 검색된 LR 트리 내 노드의 복합 레벨값에 해당 레벨의 레벨값을 비트NOT 연산자와 비트AND 연산자에 의해 삭제

이러한 갱신 방법 중 간략화된 상위 레벨의 데이타를 공유하여 사용하는 경우, 즉, 선택 연산자에 의한 결과로 만들어진 다중 레벨 데이타를 삽입하는 경우를 그림 1로 예를 들어보면 다음과 같다. 즉, 노드 N4의 객체 b_1이 레벨 2에서 데이타가 삽입된다고 가정하면, b_1의 복합 레벨값 '01'은 '11'로 수정되고 이는 상위 노드 N2의 엔트리인 X1의 복합 레벨값로 전파되어 엔트리 X1의 복합 레벨값은 '11'로 갱신된다. 이를 통해, 엔트리 X1의 자식 노드에는 레벨 1과 레벨 2에 모두 속한 객체, 즉 선택 연산자에 의한 결과로 만들어진 다중 레벨 데이타가 존재함을 알 수 있게 된다.

3.6 높이 균형을 위한 수정된 MLR 트리 : MMLR 트리

MLR 트리는 레벨별 데이타 개수가 같은 높이를 가지는 분포로 되어 있는 경우 효과적이다. 그러나, 레벨별 데이타 개수가 같은 높이를 가지는 분포가 아닌 경우 MLR 트리에는 높이 불균형 문제가 발생하고, 이에 따라 트리의 높이가 최소가 되지 않음으로써 최악의 검색 시간 측면에서 문제점을 가질 수 있다. 따라서, 본 절에서는 이러한 문제점을 해결하기 위해서 높이 균형을 위한 수정된 MLR 트리, MMLR 트리(Modified MLR-tree)를 추가 제안한다.

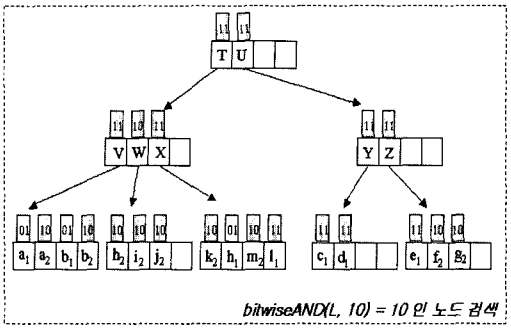


그림 2 MMLR 트리의 구조와 검색 질의 처리 과정 예

MMLR 트리는 MLR 트리로부터 다중 레벨 노드를 삭제하고, 복합 레벨값에 의해서만 각 데이타의 레벨을 처리하는 구조이다. 그림 2는 추가적으로 제안된 인덱스 구조, MMLR 트리의 상세 구조와 이를 통한 검색 방법의 예를 보인다.

즉, "레벨 2에 속한 모든 객체를 찾아라"라는 검색 질의어를 예로 들어보면, MLR 트리의 경우는 우선 루트 노드를 검색하여 해당되는 LR 트리를 검색한다. 그러나, MMLR 트리는 루트 노드로부터 리프 노드까지 bitwiseAND(l,'10')='10'인 노드를 찾아, '11'과 '10'의 복합 레벨값을 가진 트리 내 노드를 검색하게 된다.

MMLR 트리의 구조가 MLR 트리와 다른 점은 레벨별로 인덱스 트리를 구분하지 않아 레벨별로 공유되는 데이타를 최상위 레벨에 저장하지 않는다는 것이다. 그러므로, MMLR 트리는 복합 레벨값이 각 노드의 엔트리에 추가된 것을 제외하고는 R 트리와 동일한 특성을 가지게 된다. 즉, 데이타의 저장 방법에 있어 복합 레벨값의 추가를 제외하고는 R 트리와 동일한 저장 구조를 취하게 된다. 따라서, R 트리가 가지고 있는 높이 균형 특성을 그대로 유지할 수 있다는 장점을 가진다.

MLR 트리와 MMLR 트리의 검색 처리 방법의 차이점은 다음과 같다. 첫째, 다중 레벨 데이타의 상위 레벨 데이타일수록 MLR 트리의 검색 성능이 좋을 것임을 예상할 수 있다. 이는 상위 레벨 데이타에서 MLR 트리를 사용하는 경우 하위 레벨의 LR 트리는 전혀 검색을 하지 않아도 되는데 비해, MMLR 트리는 레벨별로 트리가 구분되어 있지 않으므로, 트리의 모든 노드를 대상으로 엔트리 내 복합 레벨값을 검색해야 한다. 둘째, 레

벨별 데이타 개수가 같은 높이를 가지는 분포가 아닌 경우 MMLR 트리가 MLR 트리에 비해 우수한 성능을 보일 것임을 예상할 수 있다. 즉, MMLR 트리는 높이 균형 특성을 가지게 되므로, 레벨별 데이타 개수와 무관하게 비슷한 성능을 보이게 되는데 비해, MLR 트리는 특정 레벨의 LR 트리가 한쪽으로 치우친(skewed) 트리 모양을 가지게 되기 때문이다. 이를 통해, 레벨별 데이타 개수의 차이가 큰 경우는 MMLR 트리를 사용하고, 레벨별 데이타 개수의 차이가 많지 않은 경우는 MLR 트리를 사용하게 된다면 좋은 효과를 얻을 수 있을 것임을 알 수 있다.

MMLR 트리를 R 트리의 성능 분석[4]에 비추어 검색 성능 측면에서 간략히 분석해 보면 다음과 같다. 각 노드당 최소 엔트리 수를 m이라 하고, 레벨별로 중복된 데이타를 한 개의 데이타로 고려하여 모든 레벨의 총 데이타 수를 N개라 할 때 MMLR 트리의 높이는 $\lceil \log_m N-1 \rceil$ 이다. 이는 MMLR 트리의 검색 시간이 최악의 경우 시간 복잡도가 O(log N)이 됨을 의미한다. 이로부터 MLR 트리와는 달리 특정 레벨의 LR 트리 크기와 관계없이 높이 균형 트리의 검색 속도를 유지함을 알 수 있다. 이는 MMLR 트리가 복합 레벨값이 각 노드의 엔트리에 추가된 것을 제외하고는 R 트리와 동일한 특성을 가지게 되므로 높이 균형트리임을 보장받기 때문이다.

4. 성능 평가

본 장에서는 실험적 방법을 사용하여 MLR 트리의 성능 평가 내용을 다룬다. 본 실험에서는 다중 레벨 지리정보 데이타에 대해 레벨별 R 트리 집합, 한 개의 R 트리, 그리고 MLR 트리의 성능을 비교 평가한다. 그러나, 다중 레벨 지리정보 데이타를 지원하는 공간 인덱스인 Reactive 트리, PR 파일, Multi-scale Hilbert R 트리에서는 모든 유형의 다중 레벨 지리정보 데이타가 표현이 되지 않는다는 문제점으로 인해 비교 대상에서는 제외하기로 한다.

실험은 가상 데이타와 실제 데이타의 2가지 종류의 데이타로 나누어 수행하였다. 가상 데이타는 데이타의 특성에 따른 성능을 살펴 볼 목적으로, 각 데이타의 특성에 따라 임의 생성된 데이타를 그 대상으로 하였다. 실제 데이타는 실제 환경에서의 적용 가능성을 보일 목적으로, 실제 사용되는 데이타를 대상으로 하였다.

4.1 실험 환경

본 논문에서는 실험을 위해 R 트리와 MLR 트리 모두를 구현하였다. 구현 프로그램은 윈도우즈에서 구동되는 Cygwin[15] 기반 하에 GNU C++을 사용하여, 펜티엄 III 800EB와 256MB 메인 메모리의 개인용 컴퓨터에서 수행되었다. 또한, 데이타 수의 변화에 따른 성능을 보다 명확히 하기 위해 인덱스 트리 내 최대 엔트리 수는 매우 적은 수의 엔트리 수인 5를 선택하였고, 노드 액세스시 어떠한 버퍼링 기법도 사용하지 않았다.

4.2 실험 데이타 및 실험 질의문 : 가상 데이타

실험에 사용된 데이타는 데이타 특성에 따른 성능을 평가하기 위한 목적으로 유형화 하였다. 이를 위해 레벨별 데이타간 특성에 따른 성능을 확인하기 위한 유형과, 데이타 개수와 레벨 수의 변동에 따른 성능을 확인하기 위한 유형의 데이타로 크게 2가지 범주로 분류하였다.

우선 레벨별 데이타간 특성에 따른 성능을 확인하기 위한 유형의 데이타는 다음과 같이 2가지 유형으로 분류하였다. 첫째, 상위 레벨의 간략화 된 모든 데이타를 중복해서 사용하는 유형 즉, 선택 연산자에 의한 다중 레벨 데이타 유형으로, 이를 본 논문에서는 '*중복 유형*'으로 부른다. 둘째, 상위 레벨의 간략화 된 데이타와 중복되는 데이타 없이 모든 데이타가 수정되는 유형, 즉 선택 연산자 이외의 연산자에 의한 다중 레벨 데이타 유형으로, 이를 본 논문에서는 '*수정 유형*'으로 부른다. 이 때 *중복 유형*과 *수정 유형*의 각 레벨별 데이타의 개수는 중복된 데이타를 포함하여 동일하도록 하였다.

다음으로, 데이타 개수의 변동과 레벨 수의 변동에 따른 성능을 확인하기 위한 유형의 데이타는 다음과 같이 2가지 유형으로 분류하였다. 첫째, 데이타 개수의 변동에 따른 데이타는 전체 데이타 개수가 각각 60,000개, 120,000개, 180,000개, 240,000개, 300,000개가 되도록 분류하고 레벨 수는 5로 고정하였다. 이를 본 논문에서는 '*데이타 개수 변동 유형*'으로 부른다. 둘째, 레벨 수의 변동에 따른 데이타는 2, 3, 4, 5, 6으로 레벨 수를 분류하고, 중복 유형의 데이타인 경우 중복된 데이타를 제외한 데이타의 개수가 50,000 개가 되도록 고정하였다. 즉, 중복 유형의 경우 중복된 데이타를 제외한 데이타 개수의 변화가 발생하지 않고 수정 유형에서도 데이타의 변화가 크지 않도록 최하위 레벨의 데이타 개수를 50,000개로 고정하였다. 본 논문에서는 이를 '*레벨 수 변동 유형*'으로 부르도록 한다.

이러한 유형별 분류에 따라 각 데이타는 0부터 10,000 사이의 사각형 좌표로 임의로 분포되도록 난수 생성기에 의해 생성되었다. 레벨별 데이타의 개수를 살펴보면, 각 레벨별 데이타의 개수는 간략화된 상위 레벨의 데이타 개수에 비해 구체화된 하위 레벨의 데이타 개수가 많도록 하였다. 이 때, 상위 레벨로부터 하위 레벨간 데이타 개수는 동일한 개수만큼 증가하도록 하였다.

유형별 데이타의 특징을 간략하게 요약하면 다음과 같다. 첫째, 전체 데이타 개수의 변화를 살펴보면, *데이타 개수 변동 유형*에서는 유형 값의 증가에 따라 전체

표 1 가상 데이타의 실험 질의문 집합

실험 질의문 집합	질의문 개수	레벨별 윈도우 질의 영역
Q1	레벨 수 × 1,000	윈도우 최대크기 / 레벨값
Q2	레벨 수 × 1,000	윈도우 최대크기/ 레벨값^2
Q3	레벨 수 × 1,000	윈도우 최대크기 / 레벨값^4

데이타 개수가 모두 증가하는데 비해 *레벨 수 변동 유형*에서는 중복 유형에서의 현재 레벨에서 추가되는 전체 데이타 개수가 일정하다. 둘째, 레벨별 데이타 개수의 변화를 살펴보면, *데이타 개수 변동 유형*에서는 유형 값의 증가에 따라 레벨별 데이타 개수가 증가하는데 비해 *레벨 수 변동 유형*에서는 유형 값의 증가에 따라 레벨별 데이타 개수가 감소한다.

가상 데이타의 검색 성능을 평가하기 위한 실험 질의문은 3가지 집합으로 그룹화 된다. 실험 질의문은 Q1이 가장 넓은 영역이며, Q3로 갈수록 윈도우 질의 영역이 좁아진다. 각 실험 질의문 집합은 레벨별로 각각 1,000개씩 랜덤한 위치로 생성하였다. 이러한 실험 질의문 집합은 표 1로 요약된다.

4.3 실험 데이타 및 실험 질의문 : 실제 데이타

실험에 사용된 데이타는 서울시를 대상으로 4개 레벨의 같은 영역에 대한 데이타를 대상으로 하였다. 사용된 데이타는 약 10만 개의 Shape 형태의 데이타를 사용하였다. 간략 레벨로부터 상세 레벨까지의 데이타의 총개수는 37개, 393개, 17,265개, 80,420개이며, 간략화된 레벨의 객체와 중복된 객체 수는 레벨별로 3개, 37개, 5438개이다. 또한, 그림 1은 이러한 실험 데이타의 일부분을 보여준다.

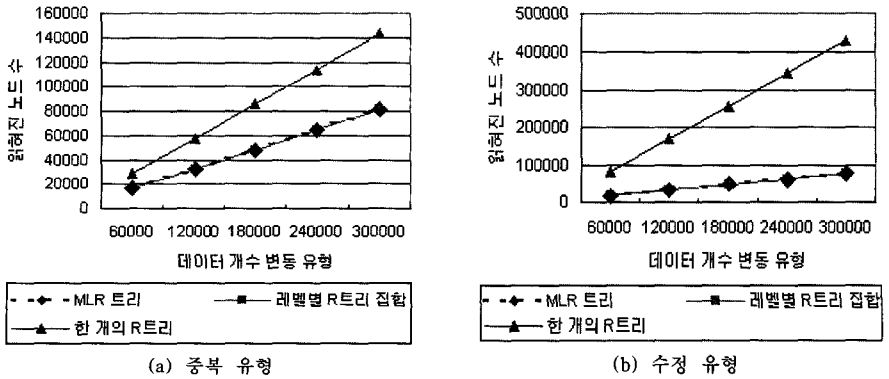
실제 데이타의 검색 성능을 평가하기 위한 실험 질의문은 4개의 레벨당 10,000개의 질의 윈도우를 임의로 생성하였다. 이 때, 생성된 각 질의 윈도우는 윈도우 크기가 커지면 레벨 값이 적어진다.

4.4 실험 결과 : 가상 데이타

검색 성능을 측정하기 위해 본 논문에서는 검색시 읽혀진 평균 노드 수를 측정하였다. 그림 3은 *데이타 개수 변동 유형*에서의 검색시 읽혀진 평균 노드 수를 보인다. 그림 4와 그림 5는 *레벨 수 변동 유형*에서의 검색시 읽혀진 평균 노드의 수를 보인다. 검색 성능을 평가하기 위해 본 논문에서는 3가지 유형의 실험 질의문에 대한 평균 노드의 수에 대한 평균값을 구했다.

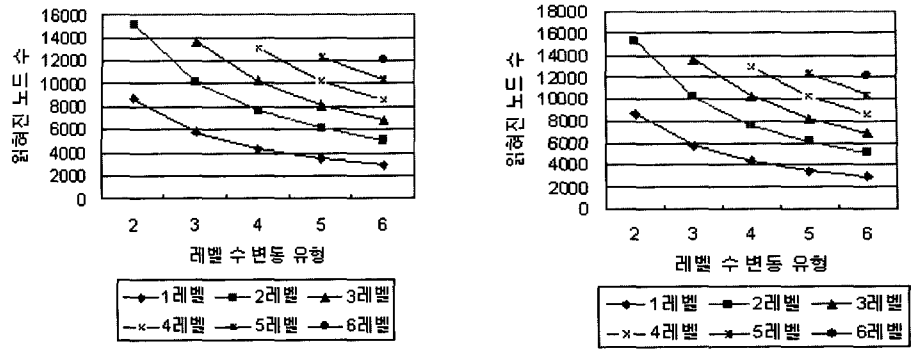
그림 3으로부터 MLR 트리와 레벨별 R트리가 데이타 개수와는 상관없이 늘 한 개의 R 트리에 비해 읽혀진 노드의 수가 적음을 알 수 있다. 또한 이러한 차이는 데이타 개수가 증가함에 따라 더 커지며 *중복 유형*보다 *수정 유형*에서 더 분명히 나타나고 있다. 이는 다중 레벨 데이타를 검색할 때, 한 개의 R 트리는 레벨별 데이타가 아닌 전체 데이타를 대상으로 검색을 수행하며, 데이타를 중복 저장하지 않기 때문에 중복되지 않은 데이타의 수에 따라 전체 데이타의 수가 결정되기 때문이다. 이에 비해 MLR 트리와 레벨별 R트리 집합의 실험 결과는 거의 동일하여 마치 하나의 그래프인 것처럼 보임을 알 수 있다.

그림 4로부터 MLR 트리는 레벨의 증가에 따라 검색시 읽혀진 노드의 수가 감소한다. 또한 레벨별 R 트리의 실험 결과도 그림 4와 거의 동일하며, 본 논문에서는 생략하였다. 이는 *레벨 수 변동 유형*에서는 유형 값의 증가에 따라 레벨별 데이타 개수가 감소하는 데이타의 특성과 동일한 결과이며 이로부터 MLR 트리와 레벨별



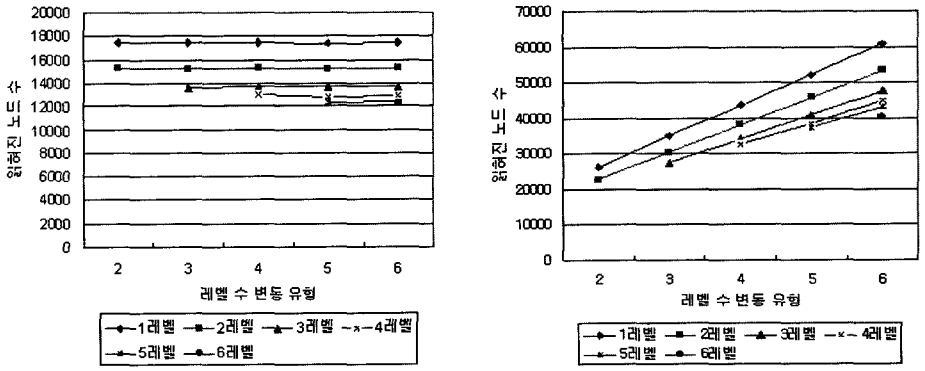
(a) 중복 유형 (b) 수정 유형

그림 3 검색시 읽혀진 노드 수 (데이타 개수 변동 유형)



(a) 중복 유형 (b) 수정 유형

그림 4 MLR 트리 검색시 읽혀진 노드 수 (레벨 수 변동 유형)



(a) 중복 유형 (b) 수정 유형

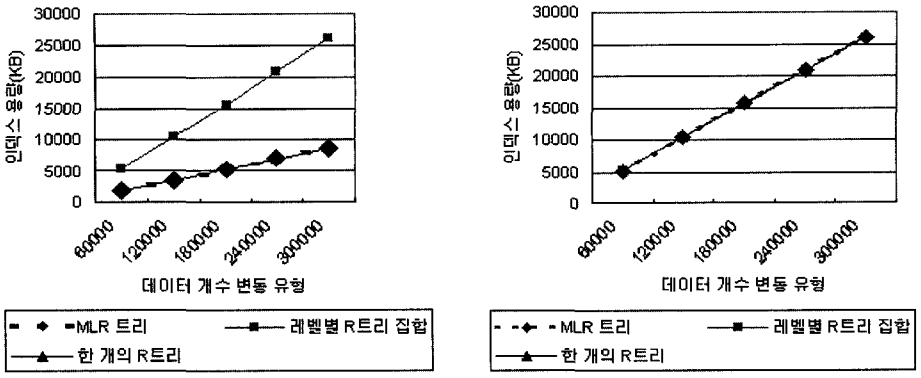
그림 5 한 개의 R 트리 검색시 읽혀진 노드 수 (레벨 수 변동 유형)

R 트리는 레벨별 데이타 개수에 따라 검색시 읽혀진 노드의 수가 결정됨을 알 수 있다. 즉, 같은 수의 데이타에 대해 레벨별 데이타 개수는 레벨 수가 증가할수록 감소하게 된다.

이에 비해 한 개의 R 트리는 그림 5에서 보는 바와 같이 레벨의 증가에 따라 읽혀진 노드의 수가 늘 일정하거나 증가하게 된다. 이는 중복된 데이타를 제외한 데이타 개수의 변화와 같은 유형이다. 이로부터 한 개의 R 트리는 레벨의 변화와 관계없이 전체 데이타의 개수에 비례하여 검색 성능이 결정됨을 알 수 있다.

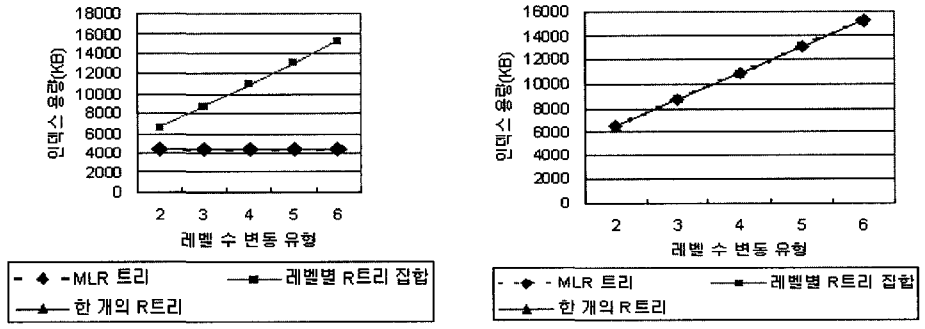
저장 공간을 평가하기 위해서 본 논문에서는 전체 데이타를 삽입한 후 디스크에 저장된 인덱스의 용량을 측정하였다. 그림 6은 *데이타 개수 변동 유형*의 저장된 인덱스의 용량을 보인다. 그림 7은 *레벨 수 변동 유형*의 저장된 인덱스의 용량을 보인다. 인덱스의 용량은 유형별 데이타의 중복된 데이타를 제외한 데이타 개수와 동일한 유형을 보인다. 모든 유형에서 공통적인 결과는 MLR 트리와 한 개의 R 트리가 거의 비슷한 결과를 보이며, 레벨별 R 트리의 경우 인덱스의 용량이 가장 크다는 것이다.

수정 유형의 경우에 대한 인덱스 용량은 그림 6(b)와 그림 7(b)에서 보이는 것처럼 모든 인덱스가 거의 동일한 결과를 보인다. 그러나, *중복 유형*의 경우는 레벨별 R 트리가 인덱스 용량이 가장 크며, *수정 유형*과 용량이 거의 동일함을 알 수 있다. 이는 *중복 유형*과 *수정 유형*의 경우 모두 각 레벨별 데이타의 개수는 중복된 데이타를 포함하여 동일하기 때문인데, 이를 통해 레벨별 R 트리는 레벨별로 중복되는 데이타를 중복 저장한다는 사실을 알 수 있다. 이에 비해, 그림 7(a)에서 보이는 바와 같이 중복된 데이타를 제외한 데이타 개수가 늘 일정한 *레벨 수 변동 유형*이고 *중복 유형*의 데이타인 경우에는 MLR 트리와 한 개의 R 트리는 인덱스의 용량에 거의 변화가 없음을 알 수 있다. 이는 각 데이타의 특성 중 중복된 데이타를 제외한 데이타 개수와 동일한 특성을 보이는 것으로, MLR 트리와 한 개의 R 트리는 레벨별로 중복된 데이타를 중복 저장하지 않는



(a) 중복 유형 (b) 수정 유형

그림 6 인덱스 용량 (데이타 개수 변동 유형)



(a) 중복 유형 (b) 수정 유형

그림 7 인덱스 용량 (레벨 수 변동 유형)

다는 것을 알 수 있다.

4.5 실험 결과 : 실제 데이타

가상 데이타와 마찬가지로, 본 논문에서는 검색 성능을 측정하기 위해 검색시 읽혀진 평균 노드 수를 측정하고, 저장 공간을 측정하기 위해 인덱스의 용량을 측정하였다.

그림 8은 레벨별 데이타 검색시 읽혀진 평균 노드 수를 보인다. 이를 통해 다음과 같은 결과를 관찰할 수 있다. 첫째, MLR 트리는 레벨별 R 트리 집합과 읽혀지는 평균 노드 수가 거의 동일하다. 둘째, 레벨값이 적을수록 한 개 R 트리와 성능의 차이가 증가한다. 이는 한 개 R 트리는 전체 데이타를 대상으로 검색이 이루어지는데 비해 MLR 트리와 레벨별 R 트리 집합은 레벨별 데이타를 대상으로 검색이 이루어지기 때문이다.

그림 9는 각 인덱스 기법에 의해 저장된 인덱스 용량을 보여준다. 이를 통해 다음과 같은 결과를 얻는다. 첫째, MLR 트리와 한 개의 R 트리에 대한 저장 공간 용량이 가장 적다. 둘째, 검색 성능 측면에서 우수한 성능

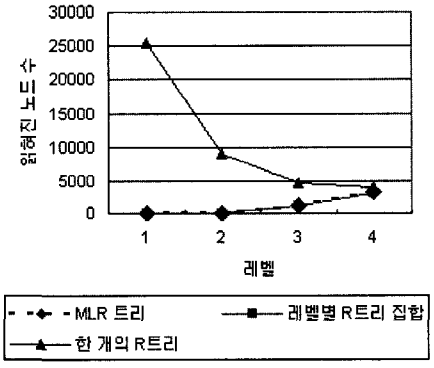


그림 8 검색시 읽혀진 노드 수 (실제 데이타)

을 보였던 레벨별 R 트리 집합은 최악의 결과를 얻었다. 이는 레벨별 데이타 중복성에 따른 결과이다. 이 때, 각 인덱스 용량의 차이가 매우 크지 않은 것은 실제 데이타의 경우 레벨간 중복되는 데이타가 상대적으로 크지 않기 때문이다.

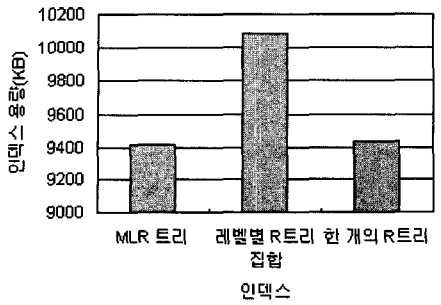


그림 9 인덱스 용량 (실제 데이타)

5. 결 론

본 논문에서는 다중 레벨 지리정보 데이타의 윈도우 질의를 지원하는 공간 인덱싱 기법인 MLR 트리를 제안하였다. 본 논문에서는 기존의 공간 인덱스 중 R 트리를 기반으로 하였다. 이는 R 트리가 가장 널리 사용되는 공간 인덱스이기 때문에 사용된 것으로, 본 논문의 아이디어는 어떠한 기존 공간 인덱스 구조에도 응용 가능하다. MLR 트리는 다음과 같은 특징을 가진다. 첫째, 모든 유형의 다중 레벨 데이타를 지원한다. 둘째, 전통적인 공간 인덱싱 기법에서 발생하는 검색 속도 저하와 데이타 중복 문제가 발생하지 않는다. 셋째, 다중 레벨 지리정보 데이타의 윈도우 질의를 수행하기에 편리하도록 하나의 통합된 구조로 이루어진다.

또한 이를 가상 데이타를 통해 데이타 유형별로 MLR 트리의 성능을 관찰하였으며, 실제 데이타를 통해 실제 환경에서의 적용 가능성을 입증하였다. 이를 통해 MLR 트리는 다음과 같은 우수한 성능을 보였다. 첫째, 검색 성능 측면에서 레벨별 R 트리 집합과 동일하거나 거의 대등한 성능을 보였다. 둘째, 저장 공간 측면에서 한 개의 R 트리와 거의 동일한 성능을 보였다. 또한 이러한 원인이 레벨별 R 트리에서 발생하는 데이타 중복성 문제가 발생하지 않기 때문임을 보였다. 셋째, 데이타의 유형별 분류를 통해 모든 유형의 다중 레벨 지리정보 데이타를 지원함을 보였다. 넷째, MLR 트리는 레벨 수가 증가할수록, 데이타 수가 증가할수록, 그리고 레벨별로 중복된 데이타 수가 증가할수록 검색 성능과 저장 공간 측면에서 R 트리를 사용한 방법에 비해 그 우월성이 보다 더 증가함을 알 수 있었다.

향후 연구 과제로는 첫째, 하나의 객체에 대한 레벨별로 다중 표현된 데이타를 하나의 객체로 인식하도록 함으로써 중복 갱신 문제를 해결할 수 있도록 하는 연구가 필요하다. 둘째, 성능 평가에 있어 현재의 실험 데이타 외에도 일반적으로 많이 사용되고 있는 몽고메리카운티 데이타와 롱비치카운티 데이타를 사용하여 실험결과의 객관성을 보다 높이는 것이 필요하다. 셋째, 선택과 단순화 연산자로 제한하여 기존의 Reactive 트리, PR 파일, Multi-scale Hilbert R 트리와의 성능 비교가 필요하다. 넷째, 높이 균형 문제를 해결하기 위한 MMLR 트리에 대한 성능 평가가 필요하다. 이를 위해 데이타의 유형을 구분하여 MLR 트리와 MMLR 트리의 성능을 분석하고 이를 실험하는 것이 추가적으로 요구된다. 또한 이러한 비교 분석을 통해 각 데이타 특성에 적합한 하이브리드(hybrid) 방식의 새로운 인덱스 구조도 고려해 볼만하다.

참 고 문 헌

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] N. Beckmann, H.P. Kriegel, R. Schneider, 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.

[7] 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.

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

[9] H.Samet, R.E.Webber, "Storing a Collection of Polygons using Quadtrees," ACM Transactions. Graph, 4(3), pp.182-222, 1985.

[10] 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.

[11] A. Hutflesz, H-W. 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.
[12] P.V. Oosterom, "The Reactive-tree : A Storage Structure for a Seamless, Scaless Geographic Database," Proceedings of Auto-Carto, 10, pp.393-407, 1991.
[13] B. Becker, H-W. Six, and P. Widmayer, "Spatial Priority Search : An Access Technique for Scaless Maps," *Proceedings of ACM SIGMOD*, Denver, Colorado, pp.128-137, 1991.
[14] D.H. Douglas, T.K.Peucker, Algorithms for the Reduction of Points Required to Represent a Digitized Line or its Caricature, Canadian Cartography, 10, pp.112-122, 1973.
[15] http://www.cygwin.com.

권 준 희

1992년 숙명여자대학교 전산학과 졸업(학사). 1994년 숙명여자대학교 대학원 전산학과 졸업(석사). 2002년 숙명여자대학교 대학원 컴퓨터과학과 졸업(박사). 1994년~2003년 2월 쌍용정보통신 Mobile/GIS팀 과장. 2003년 3월~현재 경기대학교 정보과학부 교수. 2000년 전자계산조직응용기술사. 관심분야는 공간 데이타베이스, GIS, LBS, 멀티미디어 데이타베이스, 모바일 데이타베이스, 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. 관심분야는 멀티미디어, GIS, 미들웨어, 실시간 데이타베이스, 실시간 운영체제, 병렬 처리, 분산시스템, 텔레커뮤니케이션 시스템