R-tree에서 위치 기반 질의를 지원하기 위한 동적 셀 레벨링†

Dynamic Cell Leveling to Support Location Based Queries in R-trees

정연욱*, 구경이**, 김유성***

Yun-Wook Jung, Kyong-I Ku, Yoo-Sung Kim

요 약 최근 GPS기능을 탑재한 휴대폰·PDA등의 모바일 장치를 사용하여 위치 기반 서비스를 이용하는 사용자가 급증하고 있다. 이에 위치 정보를 지닌 공간 데이타를 저장하고 관리하는 대용량의 공간 데이타베이스는 사용자의 다양한 질의 조건과 빠른 검색을 가능하게 하기 위한 색인이 필요하며 대표적인 색인 기법으로는 R-tree가 널리 사용되고 있다. 기존의 R-tree를 이용한 검색은 질의 영역과 관계없는 공간 데이타까지 검색하여 불필요한 입출력을 유발하기 때문에 응답시간이 늦어진다. 본 논문에서는 공간 데이타가 위치 기반 질의를 지원하기 위해 자신이 속한 노드의 전체(Union) MBR 영역에 따라 동적으로 셀 레벨 값을 구성하는 CLR-트리(Cell Leveling R-tree)를 제안한다. 또한, CLR-트리에서의 새로운 공간 데이타의 삽입, 분할, 삭제, 검색 알고리즘을 제안한다. 제안된 CLR-트리에서는 위치 기반 질의 시 사용자 질의 영역의 셀 레벨 값과 공간 데이타의 셀 레벨 값을 비교하여, 겹치지 않는 셀에 대해서는 검색 대상으로부터 제거하고 연관된 셀만을 검색하기 때문에 빠른 응답시간을 제공한다. 디스크 입출력 실험에서 CLR-트리가 기존 R-tree보다 디스크 접근 수를 5~20% 감소시켜 사용자의 위치 기반 질의에 대해 빠르게 처리함을 알 수 있었다.

Abstract Location Based Services(LBSs) in mobile environments become very popular recently. For efficient LBSs, spatial database management systems must need a spatial indexing scheme such as R-trees in order to manage the huge spatial database. However, it may need unnecessary disk accesses since it needs to access objects which are not actually concerned to user's location-based queries. In this paper, to support the location-based queries efficiently, we propose a CLR-tree(Cell Leveling R-tree) in which a dynamic cell is built up within the minimum bounding rectangle of R-trees' node. The cell level of nodes is compared with the query's cell level in location-based query processing and determines the minimum search space. Also, we propose the insertion, split, deletion, and search algorithms for CRL-trees. From the experimental results, we see that a CLR-tree is able to decrease 5~20% of disk accesses from those of R-trees. So, a CLR-tree can be used for fast accessing spatial objects to user's location-based queries in LBSs.

주요어 : 위치 기반 서비스, 공간 색인, R-tree, 셀 분할.

KeyWords : Location based service, spatial index, R-tree, cell clipping.

† **본 연구는 한국과학재단 지정 인천대학교 동북아전자물류연구센터의 지원에 의한 것임**

* 팅크웨어 연구원 yunoogi@hotmail.com

** 인하대학교 정보통신공학부 mabtovle@hanmail.net

*** 인하대학교 정보통신공학부 교수 yskim@mha.ac.kr

1. 서 론

휴대용 컴퓨터 장치 및 무선 통신 시스템의 발전으로 많은 사람들은 이동하는 동안 '언제'(Anytime), '어디서나'(Anywhere)나 무선 통신을 이용하여 서로 통신할 뿐만 아니라 자신의 위치와 관련된 정보를 얻을 수 있다. 여기서 위치(Location)라는 정보는 무선 통신을 사용하는 어플리케이션에서 중요한 요소로 작용하며, 사용자의 지리적 위치와 밀접하게 관련되어 있다. 최근에는 GPS를 탑재한 휴대폰·PDA등의 모바일 장치를 사용하여 위치 기반 서비스(Location Based Service : LBS) 즉, 사용자의 위치를 기반으로 사람이나 사물의 위치를 정확하게 파악하고 이를 활용하는 응용 시스템 및 서비스를 이용하고자 하는 요구가 급증하고 있다. 이러한 위치 기반 서비스를 이용할 경우 사용자는 자신의 위치 정보를 기반으로 위치 기반 질의(Location Dependent Query : LDQ)를 하게 되며, 공간 데이타베이스에 구성된 다차원 공간 색인 기법을 이용하여 사용자의 질의영역과 연관된 공간 데이타를 검색하게 된다[1][2][3].

다차원 색인 중에서 지리적 특성을 반영한 공간 데이타베이스의 색인 방법으로는 R-tree[4], R*-tree[5], Quad-tree[6] 등 다양한 공간 색인 방법이 사용되고 있다. 이 중 R-tree는 데이타의 공간적 위치에 따라 데이타 항목을 검색하며 널리 사용되고 있다. R-tree에 저장되는 공간 데이타는 최소 경계 사각형(Minimum Bounding Rectangle : MBR)안에 단말 노드의 데이타로 표현되며 지리적 위치를 기반으로 나타난다[7]. 사용자가 '반경 5Km 안에 위치한 호텔을 찾아라'라는 위치 기반 질의에 대해, 사용자의 지리적 위치를 기준으로 일정 거리 안에 있는 모든 공간 데이타를 대상으로 검색한다. 그러나 위치 기반 질의를 처리하기 위해 R-tree는 질의 영역과 관계없는 공간 데이타까지 검색하기 때문에 불필요한 디스크 접근으로 빠른 검색 응답 시간을 제공하지 못한다.

본 논문에서는 위치 기반 질의를 효율적으로 처리하기 위해 R-tree의 변형으로서 노드의 전체(Union) MBR 영역을 4개의 셀로 분할하고 사용자 질의 영역과 연관된 셀만을 선택하여 검색하도록 하는 색인 기법인 CLR-트리를 제안한다. CLR-트리에서 입력되어지는 공간 데이타는 저장되어질 노드의 전체(Union) MBR 영역에 따라 셀 레벨 테이블을 참조하여 셀 레벨 값을 생성하고 공간 데이타와 함께 노드에 저장된다. 위치 기반 질의 시에는 셀 레벨 알고리즘을 이용하여 사용자의 질의 영역에 대해 셀 레벨 값을 구성한 뒤 데이타베이스 안에 저장된 공간 데이타의 셀 레벨 값과 비교하며 질의를 수행하게 된다. 즉, 사용자의 질의 영역과 공간 데이타간의 셀 레벨 정보도 질의 처리 조건의 하나로 사용하여 질의 영역과 동일한 셀 레벨 값을 갖는 공간 데이타만을 검색하게 된다. 또한, CLR-트리에서의 새로운 공간 데이타의 삽입, 분할, 삭제, 검색 알고리즘을 제안하였다. 실험 결과, 사용자의 질의 영역과 겹치는 영역만을 검색의 대상으로 삼아 검색 단계에서 필요로 하는 디스크 입출력의 횟수를 줄이기 때문에 증가되는 사용자의 위치 기반 질의를 효율적으로 처리할 수 있다. 제안된 CLR-트리는 R-tree뿐만 아니라, 공간 색인에서 MBR을 사용하는 다른 R-tree 계열의 색인 기법(R*-tree 또는 R+-tree 등)에 대해서도 적용 가능하다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 공간 데이타베이스에서의 색인 및 질의 처리를 소개하며 기존의 대표적인 공간 색인 트리인 R-tree의 문제점에 대해 살펴본다. 제 3장에서는 제안하고 있는 CLR-트리의 색인 구조를 설명하며, 제 4장에서는 CLR-트리에서 노드의 삽입, 분할, 삭제 및 검색 알고리즘에 대해 소개한다. 제 5장에서는 제안된 알고리즘의 구현 및 실험을 통한 성능을 평가한다. 마지막으로 제 6장에서 결론 및 향후 연구 과제로 본 논문을 맺는다.

2. 관련 연구

2.1 공간 데이타베이스 색인 및 질의 처리

공간 데이타베이스는 일반적인 숫자, 문자뿐만 아니라 2차원 이상에 나타낼 수 있는 공간 데이타인 점(Point), 선(Line), 면(Polygon)을 표현, 저장하며 복잡한 구조와 대용량 데이타라는 특성을 갖는다. 이러한

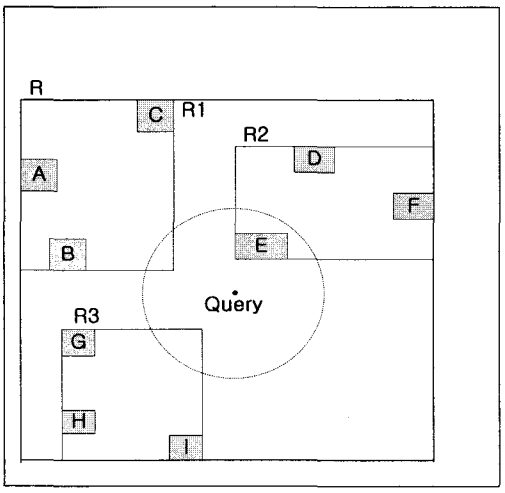
공간 데이타를 빠르고 효율적으로 검색하기 위한 공간 색인 기법은 크게 두 가지로 분류할 수 있다.

첫 번째는 MBR을 기반으로 한 색인 기법으로서 R-tree, R+-tree[8], R*-tree, LSD-tree[9], X-tree[10] 등이 있으며, 두 번째는 영역 분할(region decomposition)을 기반으로 한 색인 기법으로서 KD-tree, 사분할 트리, Z-ordering, TR*-tree[11] 등이 있다. MBR을 기반으로 한 색인 기법들은 위치 기반 질의를 만족하지 않는 공간 데이타들을 신속하게 제거할 수는 있으나, 단순한 사각형으로 된 MBR로는 공간 데이타의 위치 단위를 정확하게 표현할 수 없기 때문에 질의와 연관되지 않은 많은 후보 객체들을 포함하게 된다. 반면에 영역 분할을 기반으로 한 색인 구조들은 질의에 대한 검색 대상이 되는 공간 데이타에 대한 정확도는 향상시킬 수 있으나, 색인 구조의 복잡성이나 색인을 구성하는 많은 구성요소들로 인해 높은 질의 처리비용을 갖는다[12].

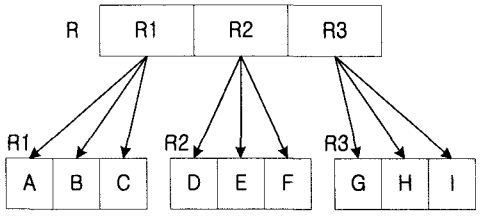
공간 질의는 비공간 질의에 비해 데이타의 복잡성과 방대함으로 인해 그 처리비용이 매우 비싼 편이다. 그러므로 여러 연구에서 공통적으로 공간 질의 처리 시 효과적인 처리를 위해 여과 단계(Filter Step)와 정제 단계(Refine Step)로 나누어 처리하는 방법을 제안하였다[13]. 첫 번째 단계인 여과 단계는 질의 영역과 겹쳐지는 MBR을 검색하는 것으로 복잡한 형태의 공간 데이타를 단순한 형태로 근사(approximation) 시킨 후 이 근사치에 대해 질의를 처리한다. 두 번째 단계인 정제 단계는 질의 영역과 겹쳐지는 MBR 안에 공간데이타를 검색하는 것으로 여과 단계를 통과한 공간 데이타 중에 정확히 질의 영역과 겹쳐지는 객체들만 선택하여 검색 결과로 제공한다. 대부분의 공간 색인들은 공간 데이타에 대한 근사 기하 알고리즘(approximation geometry algorithm)을 이용해 여과 단계 시 질의 처리를 신속히 처리하도록 한다.

2.2 기존 공간 색인 기법의 제약사항

R-tree에 저장되는 공간 데이타는 최소 경계 사각형(MBR)안에 단말 노드의 객체로 표현되며 지리적 위치를 기반으로 나타난다. <그림 1>과 같이 R-tree에서 루트 R 아래에 서브 트리(subtree) R1, R2, R3이 존재하며, 이 때 사용자는 모바일 장치를 사용하여 자신의 위/경도 위치 정보를 기반으로 반경 몇 Km내 공간 데이타를 검색한다고 가정하자.



(a) 사용자의 위치 기반 질의



(b) R-tree 색인 구조

<그림 1> R-tree 색인 구조로 구성

<그림 1>과 같은 색인구조에서 여과 단계에 의해 선택되어지는 MBR은 R1, R2, R3에 해당되므로 <표 1>에 나타난 공간 데이타 9개를 대상으로 검색하게 된다. 따라서 실제 질의 영역과 연관되지 않은 서브 트리 R1, R3를 처리하기 위해 불필요한 디스크 접근이 발생된다. 이로 인해 시스템에 오버헤드는 커지게 되어 질의 처리 시간이 늦어지게 된다.

<표 1> R-tree에서 사용자의 위치 기반 질의를 위한 검색 영역

영역	검색 후보 공간 데이타
R1	A, B, C
R2	D, E, F
R3	G, H, I

즉, 기존 R-tree에서 위치 기반 질의를 처리 시, 연관되지 않은 영역을 검색하는 오버헤드가 발생한다. 따라서 위치 기반 질의 시 사용자의 질의 영역과 연관된 공간 데이타만을 검색하여, 불필요한 오버헤드를 줄이는 색인 기법이 필요하다.

3. CLR-트리의 자료구조

3. 1 CLR-트리 색인의 필요성

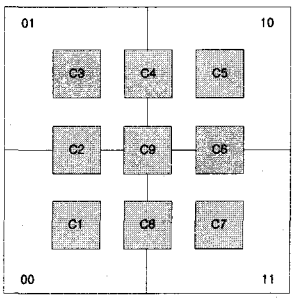
본 논문에서는 공간 색인 기법으로서 공간 데이타가 자신이 속한 노드의 전체(Union) MBR 영역에 따라 동적으로 셀 레벨 값을 구성한 뒤 이를 이용하여 위치 기반 질의 시 사용자의 질의 영역과 연관된 영역 내의 공간 객체들만을 검색하도록 하는 CLR-트리를 제안한다.

위치 기반 질의에서는 질의 영역의 위치 정보에 따라 위치 바인딩이 틀려지며 질의 영역과 결과 등도 달라진다[2, 7]. GPS를 지닌 핸드폰이나 PDA를 가진 사용자의 질의는 위/경도 위치 정보를 이용하여 위치 기반 질의를 처리하며, 예전의 모바일 장치를 가진 사용자의 질의는 Cell ID 정보를 이용하여 질의를 처리한다. 공간 데이타베이스 안에 저장된 공간 데이타의 위치 정보 또한, 위/경도, ZipCode, Cell ID, Zone등 다양하게 구성되어 질 수 있으므로 서로 다른 위치 단위 정보를 기반으로 공간 데이타를 검색 할 경우 질의 영역에 대한 결과도 달라진다.

이와 달리 CLR-트리 공간 색인을 이용할 경우 질의 영역과 데이타베이스 안에 저장된 공간 데이타는 셀 레벨 테이블에 나타난 각각의 셀 레벨에 바인딩된다. 위치 기반 질의 시 질의 영역의 셀 레벨 값과 공간 데이타의 셀 레벨 값이 서로 연관되지 않은 경우 불필요한 공간 데이타는 질의 검색 대상에서 제외하게 된다. 또한, CLR-트리는 R-tree의 변형으로 불균형한 데이타에 대해서도 높이 균형 트리를 유지한다.

3.2 CLR-트리 구성

CLR-트리에서는 <그림 2>처럼 공간 위치 정보를 셀 레벨 값(Cell Level Value)으로 표현한다. 셀 레벨 값이란 여과 단계시 공간 데이타를 빠르게 검색하기 위한 MBR의 공간 위치 값으로 각 노드의 전체(Union) MBR 영역에 대응되어 동적으로 만들어지게 된다. 단말 노드에서는 공간 데이타가 포함된 노드의 전체(Union) MBR 영역에 대해 CLevel로 공간 데이타 자신의 셀 레벨 값이 표현되며, 비단말 노드에서는 CLevel 뿐만 아니라 자신의 자식노드가 포함하고 있는 공간 데이타에 대한 셀 레벨 값의 합인 CNSLevel(Child Node Sum Level)로 셀 레벨 값을 표현한다.



(a) 공간 데이타의 위치 정보

	셀 ID	이진수	십진수
C1	00	00000001	1
C2	00, 01	00000010	2
C3	01	00000100	4
C4	01, 10	00001000	8
C5	10	00010000	16
C6	10, 11	00100000	32
C7	11	01000000	64
C8	11, 00	10000000	128
C9	00, 01, 10, 11	11111111	255

(b) 셀 레벨 테이블

<그림 2> 공간 데이타의 위치와 셀 레벨 테이블

<그림 2>(a)처럼 공간 데이타는 자신이 속한 노드의 전체(Union) MBR 영역에서 위치 정보 값, 즉 셀 레벨 값으로 C1에서 C9까지 시계 방향으로 모두 9개의 위치 정보 값 중 하나를 갖게 된다. 각각의 셀 레벨 값은 <그림 2>(b)와 같다. 8비트를 이용하여 공간 데이타의 셀 레벨 값을 표현할 경우 비단말 노드는 자식 노드가 포함하고 있는 공간 데이타를 독립된 형태로 구별하면서 각각의 셀 레벨에 대해 합한 값을 저장할 수 있다. 만약, 8비트 미만으로 각 셀 레벨을 표현 할 경우 셀 레벨 값의 합으로부터 자식 노드의 공간 데이타를 개별적인 형태로 맵핑(Mapping)할 수 없다. 입력된 공간 데이타의 셀 레벨 값은 노드 안에 입력된 공간 데이타와 함께 저장하게 된다.

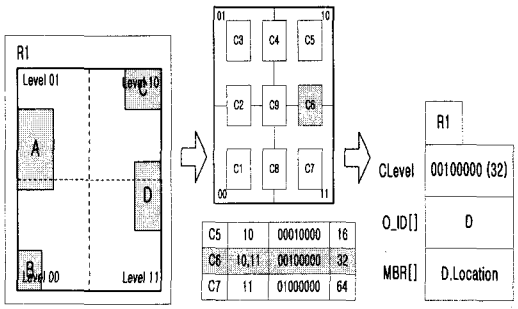
3.2.1 CLR-트리의 단말 노드

단말 노드는 실제 공간 데이타를 저장하는 노드로서 공간 데이타 입력시 적절한 셀 레벨 값을 함께 저장할 수 있다. 단말 노드의 엔트리(Entry)는 <CLevel, O_ID[], MBR[]> 이며, 각 요소는 <표 2>와 같다.

<표 2> 단말 노드 엔트리 형태의 요소

	설 명
CLevel	단말 노드의 셀 레벨 값으로 전체(Union) MBR 영역 안에서 엔트리 자신이 저장한 공간 데이타의 위치 정보 값
O_ID[]	객체 식별자
MBR[]	공간 데이타의 Location (Left_Bottom(x1, y1), Right_Top(x2, y2))

동적 배열 구조로 구성된 객체 식별자와 공간 데이타의 MBR은 입력된 공간 데이타가 저장될 노드안에 동일한 셀 레벨 값을 지닌 엔트리가 존재할 경우, 새로운 엔트리를 생성하지 않고 동일한 셀 레벨 값을 갖는 엔트리의 객체식별자와 MBR 배열 값을 확장시켜 저장한다. 따라서 위치 기반 질의 시 검색 대상이 되는 노드의 전체 엔트리를 검색하는 대신 동일한 셀 레벨 값을 가진 엔트리에 대해서만 검색하면 되므로 겹침 연산 횟수를 줄일 수 있다.



(a) 공간 데이타의 위치 (b) 셀 레벨 맵핑 (c) 단말노드 R1

<그림 3> 단말 노드안에서의 셀 레벨

<그림 3>(a)은 공간 데이타 A, B, C, D를 저장하고 있는 단말 노드 R1을 나타낸 것이다. <그림 3>(b)처럼 공간 데이타 D는 단말 노드 R1의 전체(Union) MBR 영역에 대해 레벨 10, 11 위치에 해당되며, 셀 레벨 값 맵핑을 통해 <그림 3>(c)처럼 실제 노드에 저장되는 셀 레벨 값은 32가 된다.

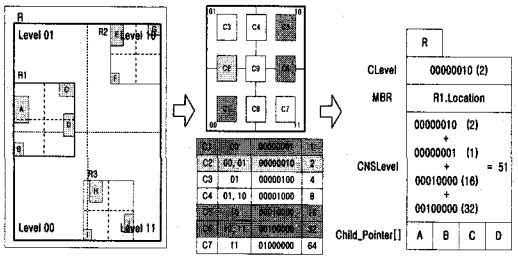
3.2.2 CLR-트리의 비단말 노드

단말 노드의 상위 노드인 비단말 노드는 단말 노드와 달리 CLevel, CNSLevel 즉, 두 개의 셀 레벨 값을 갖는다. 셀 레벨 값 CLevel은 단말 노드처럼 비단말 노드의 엔트리 안에 저장된 각각의 MBR이 비단말 노드의 전체(Union) MBR 영역에 대응하여 맵핑된 셀 레벨 값이다. 셀 레벨 값 CNSLevel은 비단말 노드의 자식 노드 각각의 엔트리에 저장되어 있는 셀 레벨 값들을 합한 값이다. CNSLevel에서는 자식 노드의 엔트리에 존재하는 각각의 셀 레벨 값 중 동일한 셀 레벨 값이 존재시 오직 하나의 셀 레벨 값만을 합하여 값을 구성한다. 즉, 동일한 셀 레벨 값을 가진 엔트리가 둘 이상일 경우는 하나의 셀 레벨 값만으로 합을 구한다. 비단말 노드의 엔트리(Entry)는 <CNSLevel, CLevel, MBR, Child_Pointer[]> 이며, 각 요소는 <표 3>과 같다.

<표 3> 비단말 노드 엔트리의 요소

	설 명
CNSLevel	자식 노드의 엔트리에 존재하는 셀 레벨 값(CLevel)의 합
CLevel	비단말 노드의 전체(Union) MBR 영역 안에서의 엔트리가 포함한 MBR의 위치 값
MBR	자식 노드의 엔트리에 존재하는 모든 MBR을 포함하는 사각형 Location (Left_Bottom(x1, y1), Right_Top(x2, y2))
Child_Pointer[]	자식 노드의 주소를 가리키는 포인터

<그림 4>는 단말 노드 R1, R2, R3를 저장하고 있는 비단말 노드 R을 나타낸 것이다. 저장된 단말 노드 R1은 비단말 노드 R의 전체(Union) MBR 영역에 대해 레벨 00, 01에 해당하므로 맵핑된 셀 레벨 값(CLevel)은 2의 값을 갖게 되며, 자식 노드에 대한 셀 레벨을 합한 값(CNSLevel)은 단말노드 R1에 존재하는 공간 데이타들의 셀 레벨 값을 모두 합한 값으로 51을 갖게 된다.

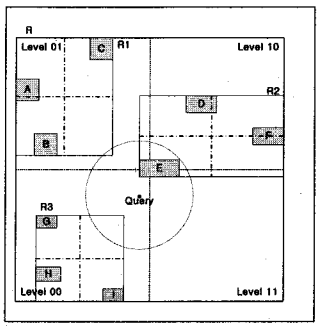


(a) 공간 데이타의 위치 (c) 비단말 노드 R

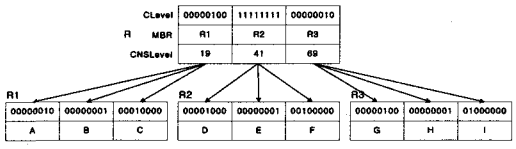
(b) 공간 데이타 R1의 셀 레벨 맵핑

<그림 4> 비단말 노드안에서의 셀 레벨

CLR-트리 공간 색인 구조에선 셀 레벨 값을 이용하여 <그림 1>의 색인 구조를 다시 재구성하면 <그림 5>와 같다. 공간 제약으로 비단말 노드와 단말노드의 MBR은 표시하지 않았다.



(a) 재구성된 공간 데이타의 위치 정보



(b) CLR-트리

<그림 5> CLR-트리 공간 색인 구조

우선 루트 R 안에 포함된 서브 트리 R1, R2, R3 영역은 <그림 5>(a)처럼 루트 R의 전체(Union) MBR 영역에 대해 셀 레벨 테이블을 참조하여 셀 레벨 값을 계산한 뒤 저장한다. 각 셀 안에 포함되어 있는 R1, R2, R3의 MBR 또한 사분할 처리후 셀 레벨 테이블을 참조하여 각 MBR에 존재하는 공간 데이타의 셀 레벨 값을 계산한 뒤 저장한다. 또한, <그림 5>(b)에서 비단말 노드인 R은 자식 노드인 R1, R2, R3에 포함된 공간 데이타 각각의 셀 레벨 값에 대해 합친 정보를 지니고 있으므로, 공간 질의 검색 시 하위 노드의 공간 데이타까지 검색이 가능하다.

<그림 5>(a)의 위치 기반 질의를 처리하기 위해 CLR-트리에서 여과 연산은 겹침 연산 후에 노드에 존재하는 셀 레벨 값과 AND(Bit) 연산을 실행하게 된다. <그림 5>(a)에서 질의 영역이 레벨 00, 01, 10, 11에 존재하므로 <그림 2>(b)에서 루트 R의 CLevel 값과 AND 연산시 R1, R2, R3이 검색 대상에 포함된다.

하지만, CNSLevel 값을 이용한 AND 연산에서는 True가 되는 MBR은 R2이다. 즉, R2만이 출력연산이 일어나게 되며 R2 안에 공간 데이타가 후보 셋으로

지정되어 겹침 및 거리 연산을 실행하게 된다. R-tree에서는 위치 기반 질의 처리 시 질의 영역에 연관되지 않은 영역까지 검색해야 했지만, CLR-트리에서는 셀 레벨 값을 이용하여 질의 영역에 연관된 영역만을 검색함으로써 디스크 접근 비용을 낮추었다. 따라서, 위치 기반 질의에 대해 검색 효율을 높일 수 있다.

4. CLR-트리의 알고리즘

이번 장에서는 CLR-트리의 삽입, 분할, 검색, 삭제 알고리즘을 제안한다. 제안한 색인 알고리즘의 특성은 아래와 같은 특성을 나타낸다.

- 위치 기반 질의 시 사용자의 질의 영역을 셀 레벨 값으로 맵핑한 뒤 질의 요건에 포함시켜 셀 레벨에 따른 검색이 가능하다.
- 질의 영역과 겹쳐지는 공간 데이타의 셀 레벨 AND 연산으로 질의와 관련 있는 영역만을 검색한다.

4.1 CLR-트리에서 삽입 알고리즘

CLR-트리에 새로운 공간 데이타를 삽입하고자 할 경우 다음을 고려한다.

첫째, 삽입하고자 하는 새로운 데이타는 자신이 저장될 노드의 전체(Union) MBR 영역에 대해 셀 레벨 알고리즘을 사용하여 적절한 셀 레벨 값을 가지고 삽입되어야 한다.

둘째, 노드의 용적률이 높아져 입력 데이타가 증가한다고 해도 비단말 노드에 저장되는 셀 레벨 값 CNSLevel은 최대 크기 2Byte를 넘지 않는다. 공간 데이타의 위치 단위를 셀 레벨 테이블을 참조하여 셀 레벨 값으로 표현했을 때 나올 수 있는 셀 레벨 값은 <그림 2>에서 총 9가지로 표현되며 9개의 셀 레벨 값을 전부 합하더라도 2Byte 이하이기 때문이다.

셋째, 입력 데이타는 노드 안에 최대 엔트리 개수 M에 따라 동적 배열로 구성된다. 동적 배열 크기는 M 보다 작거나 같아야 하며 노드에 존재하는 공간 데이타의 셀 레벨 값에 따라 데이타 배열은 동적으로 변한다.

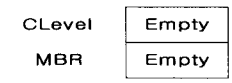
CLR-트리에 새로운 공간 데이타를 삽입하는 Insert 알고리즘은 다음과 같다.

Algorithm Insert(Spatial_Object SOb)

```
//Insert Spatial Object cell level value lvn
(n = 0, 1, 2 ··· 8)
Node node, Spatial_Object SOb;
1. SOb가 삽입되기 위한 단말 노드의 선택
node = Choose_LeafNode(Node root, Spatial_Object
       SOb);
2. 노드의 전체(Union) MBR 영역에 대해 삽입될 SOb의
셀 레벨 값 체크 한 후 삽입
SOb.lvn = Cell_Level_Check(SOb, node. Union_mbr);
while ( node의 마지막 엔트리까지 검색 ) {
  if ( node.level AND SOb.lvn)
    해당되는 엔트리의 MBR[] 배열구조를 확장시켜
    저장
  else
    빈 엔트리에 셀 레벨 값과 공간 데이타 저장
}
3. if ( 노드의 전체(Union) MBR 영역 크기나 셀 레벨
   값이 변경 )
    while ( node의 마지막 엔트리까지 검색 )
      노드 자신의 셀 레벨 값 CLevel과 상위 노드의
      CNSLevel 값을 계산하여 갱신
```

아래는 삽입 알고리즘을 단계별로 표현한 것이다.

<그림 6> CLR-트리 단말노드

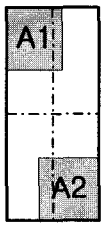


<그림 6>에서 처음에 CLR-트리의 노드는 데이타가 하나도 없는 상태이므로 단말 노드의 엔트리는 비어있는 상태로 있게 된다.

(a) 공간데이타 (b) CLR-트리

<그림 7> 공간 데이타 A1의 삽입

<그림 7>은 비어있는 노드에 공간 데이타 A1이 입력되었을 때의 그림이다. 공간 데이타 A1의 MBR 크기는 단말노드의 전체(Union) MBR 크기이기도 하므로 공간 데이타 A1이 입력된 단말노드의 엔트리 셀 레벨은 00, 01, 10, 11에 해당된다.

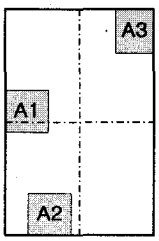


CLevel	0000010	10000000
MBR	A1	A2

(a) 공간데이타 (b) CLR-트리

<그림 8> 공간 데이타 A2의 삽입

공간 데이타 A1이 입력된 상태에서 공간 데이타 A2가 입력되었을 경우 단말노드의 전체(Union) MBR 영역은 확장되므로 공간 데이타에 대한 셀 레벨 값을 다시 재 계산해 주어야 한다. <그림 8>에서 공간 데이타 A1은 레벨 01, 10에 해당되며, A2는 레벨 00, 11에 해당된다.

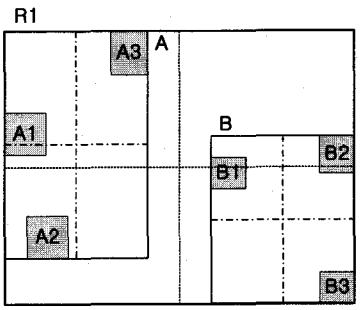


CLevel	00000010	00000001	00010000
MBR	A1	A2	A3

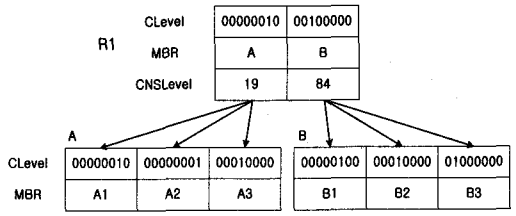
(a) 공간데이타 (b) CLR-트리

<그림 9> 공간 데이타 A3의 삽입

<그림 9>는 공간 데이타 A3이 들어온 그림이다. 여기서 또다시 단말 노드의 전체(Union) MBR 영역이 확장되므로 각 엔트리에 저장되어 있는 공간 데이타는 전체(Union) MBR 영역에 대해 셀 레벨 값을 재 계산해 주어야 한다.



(a) 공간데이타



(b) CLR-트리

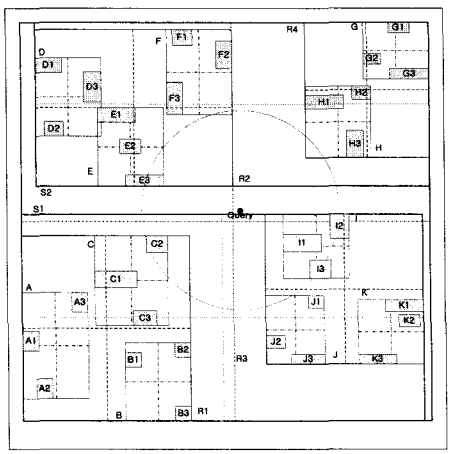
<그림 10> B1, B2, B3이 삽입

<그림 10>은 공간 데이타 B1, B2, B3가 새롭게 입력되어 트리 높이 2의 색인 구조를 나타낸 것이다. 단말 노드인 A와 B는 비단말 노드인 R1의 전체(Union) MBR 영역에 대해 셀 레벨 알고리즘을 적용한다. 즉, A의 MBR이 가지게 되는 셀 레벨 값은 R1의 전체(Union) MBR 영역에 대해 00, 01이 되고 B의 MBR은 10, 11의 셀 레벨을 갖게 된다. 비단말 노드 R1은 자신의 자식노드인 A, B에 저장되어 있는 공간 데이타 각각의 셀 레벨 값 합을 저장하고 있다. A안에 저장된 공간 데이타 A1, A2, A3의 셀 레벨 값은 셀 레벨 알고리즘에 의해 A1은 2, A2는 1, A3는 16의 값을 갖게 된다. 이 때, A와 B 노드 안에 저장된 공간 데이타 셀 레벨 값은 비단말 노드 R1의 전체(Union) MBR 영역에 영향을 받지 않는다.

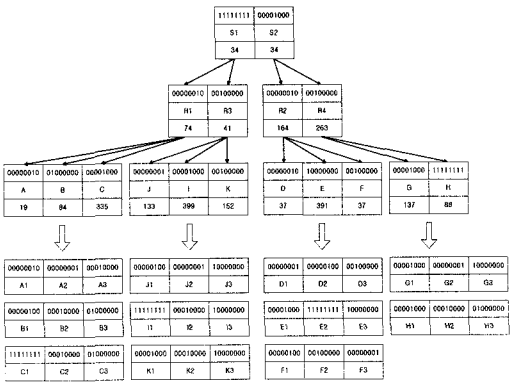
위와 같이 새로운 공간 데이타가 입력되어 질 때 공간 데이타는 저장될 노드의 전체(Union) MBR 영역에 따라 셀 레벨 값 CLevel을 계산하는 Cell_Level_Check 알고리즘은 아래와 같다.

Algorithm Cell_Level_Check(Object InputData, MBR Union_MBR)

1. Spatial_Object SOb = InputData;
2. switch (SOb or Query_area) {
   //연산에 해당되는 MBR 중심점에 대한 영역 위치 계산
   case Union_MBR의 왼쪽 아래에 위치 : SOb.CLevel = 1; break; //00
   case Union_MBR의 왼쪽에 위치 : SOb.CLevel = 2; break; //00, 01
   case Union_MBR의 왼쪽 위에 위치 : SOb.CLevel = 4; break;//01
   case Union_MBR의 위에 위치 : SOb.CLevel = 8; break;//01, 10
   case Union_MBR의 오른쪽 위에 위치 : SOb.CLevel = 16; break;//10
   case Union_MBR의 오른쪽에 위치 : SOb.CLevel = 32; break;//10, 11
   case Union_MBR의 오른쪽 아래에 위치 : SOb.CLevel = 64; break;//11
   case Union_MBR의 아래에 위치 : SOb.CLevel = 128; break;//11, 00
   case Union_MBR의 중심에 위치 : 255; break; //00, 01, 10, 11
   }
3. return SOb.CLevel



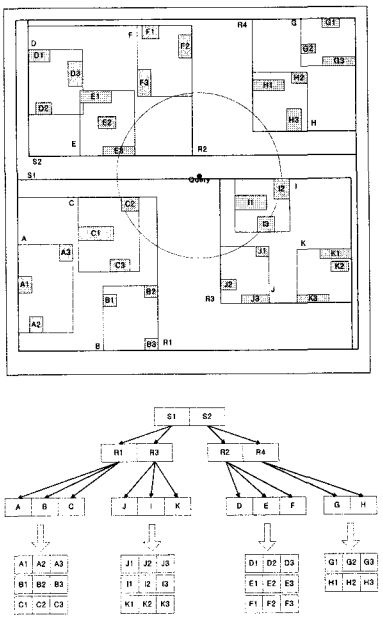
(a) 공간 데이타



(b) CLR-트리

<그림 11> 다수의 공간 객체가 들어간 CLR-트리 색인 구조

<그림 11>은 다수의 공간 객체가 들어갔을 경우 CLR-트리로 구성하여 나타낸 것이다. 비단말 노드와 단말 노드 각각은 자신이 포함한 각각의 공간 데이타에 대한 셀 레벨 값을 가지고 있다.



<그림 12> <그림 11>의 색인 구조를 R-tree로 표현

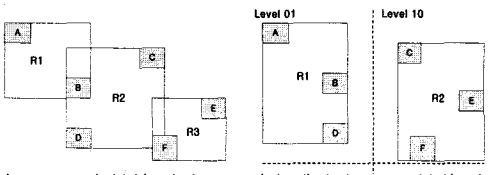
<그림 12>는 <그림 11>의 색인 구조를 기존 R-tree로 재구성한 그림으로 CLR-트리보다 좀 더 간결한 색인 구조를 가지고 있다.

4.2 CLR-트리에서 분할 알고리즘

CLR-트리에서는 오버플로우(overflow)가 발생되어 분할이 일어나는 경우 두 가지 중 하나를 이용하여 분할한다.

첫 번째는 기존 R-tree와 동일하게 MBR의 최소 영역을 기준으로 분할하게 되는 경우이다. R-tree의 분할 알고리즘을 이용할 경우 먼저 두 개의 공간 데이타를 기준으로 정한 뒤 기준이 된 공간 데이타와 나머지 공간 데이타간에 각각의 빈 공간(Dead space)영역과 최소 겹침(Overlap)영역을 비교하여 가장 효과적인 영역을 대상으로 분할하게 되며, 각각의 공간 데이타가 노드에 삽입되어지는 순간 해당 노드의 전체(Union) MBR 영역에 따라 셀 레벨 값을 계산하여 저장하게 된다.

두 번째는 <그림 13>에 나타난 것처럼 공간 질의 영역과 노드간의 셀 레벨 AND 연산을 위해 동일한 셀 레벨 값을 가진 공간 데이타를 중심으로 분할하는 것이다. 먼저 동일한 레벨 값을 가진 공간 데이타를 기준으로 두 그룹으로 묶은 뒤 이 두 그룹을 기준으로 나머지 공간 데이타와의 빈 공간 영역과 최소 겹침 영역을 비교하여 분할하게 되는 것으로 첫 번째 방법과 마찬가지로 각각의 노드에 삽입되는 순간에 셀 레벨 값을 계산하여 저장하게 된다. 동일한 셀 레벨 값을 기준으로 분할하는 것은 레벨 값에 따라 공간 데이타를 패킹(Packing)[14]하는 개념으로 공간 질의 검색 시 셀 레벨 값 비교에 의해 제거되는 대상을 좀 더 효율적으로 구분한다. 따라서, CLR-트리에서는 두 번째 분할 방법을 사용한다. 오버플로우가 발생된 노드를 분할하는 Split 알고리즘은 아래와 같다.



(a) R-tree의 분할 방식 (b) 제안된 Level 분할 방식

<그림 13> 오버플로우가 발생된 트리의 노드 분할

Algorithm Split (Node node)

1. 분할된 노드를 저장할 새로운 노드 A, B 생성
2. 각 노드에 엔트리를 저장하기 전에 공간 데이타의 셀 레벨을 체크하여 동일한 레벨을 가진 두 그룹으로 구성
3. 새로운 노드 A, B에 두 그룹을 각각 저장시킨 후
   if (다른 셀 레벨 값을 가진 나머지 공간데이타가 존재)
   while (남은 공간 데이타) {
   두 그룹과의 빈공간 영역과 최소 겹침 영역 비교한 뒤 저장
   }
4. 분할된 노드 안에 각각의 엔트리에 저장된 MBR은 자신이 속해있는 노드 전체(Union) MBR 영역에 대해 CLevel 계산한 뒤 저장
5. 분할된 노드의 상위노드가 존재할 경우 CNSLevel 값 갱신
6. 분할이 완료된 뒤, 분할 전 노드 삭제

4.3 CLR-트리에서 검색 알고리즘

CLR-트리에서의 검색 방법은 기존 연구에서와 마찬가지로 다단계 단계로 이루어진다. 여과 단계의 첫 번째 단계로 먼저 질의 영역이 공간 데이타와의 겹침 연산을 실행한다. 두 번째 단계에서는 질의 영역과 선택된 후보 공간 데이타와의 겹침 영역에 대한 셀 레벨 값을 구한 뒤 후보 공간 데이타의 셀 레벨 값과 AND 연산을 하여 동일한 셀 레벨 값을 가진 경우 후보 객체로 선택한다. 정제단계에서는 후보 객체의 지리적 위치와 사용자의 지리적 위치 정보를 계산하여 정확한 겹침 연산에 따른 결과를 제공하게 된다. CLR-트리에서 위치 기반 질의를 지원하기 위한 Search 알고리즘은 아래와 같다.

Algorithm Search (Node node, Query q)

```
//Query region cell level value lvn (n = 0, 1, 2 ··· 8)
1. if ( Overlap(q, node.Union_mbr) )
     if( node == Leaf_Node ) {
       질의 영역과 공간 데이타와의 Overlap(q,
       node.entry_mbr) 연산 후 TRUE이면 return
       node.O_ID;        }
     else {
          while ( node의 마지막 엔트리까지 검색 ) {
```

```
            //후보 공간 데이타에 대한 질의영역 셀
            레벨 값 비교
        q.lvn = Cell_Level_Check(q, node.entry_mbr);
            if ( q.lvn AND node.CNSLevel )
              Search(node.childnode, q);      }
    }
```

4.4 CLR-트리에서 삭제 알고리즘

CLR-트리에서의 삭제 방법은 기존 R-tree와 비슷하지만 삭제된 노드의 셀 레벨 값을 포함하고 있는 상위 노드의 CNSLevel 값을 변경시켜주어야 한다. 이는 루트 노드까지 계속 처리될 수 있다. 먼저 삭제하고자 하는 공간 데이타가 존재하는 단말 노드를 검색한 뒤 노드 안의 엔트리에서 공간 데이타와 셀 레벨 값을 삭제한다. 하지만, 이 경우에 동일한 셀 레벨 값을 가진 공간 데이타가 있다면 삭제되는 공간 데이타의 MBR 정보만을 삭제시켜야 한다. 또한, 삭제 시 전체(Union) MBR 영역 크기가 변했을 경우 노드에 존재하는 공간 데이타의 셀 레벨 값이 변경되므로 검색된 노드의 셀 레벨 값을 다시 재 계산하며 자신의 상위노드에서 사용되고 있는 CNSLevel 값도 갱신 시켜 주어야 한다. 삭제 연산이 실행 된 후 루트 노드가 하나의 자식 노드만 가지고 있다면 이 자식노드는 루트 노드가 된다. CLR-트리에서 하나의 노드를 삭제하는 Delete 알고리즘은 아래와 같다.

Algorithm Delete (Node node)

```
1. while ( 단말 노드까지 삭제될 공간 데이타 검색 ) {
     if ( 삭제될 공간 데이타를 발견 )
       if ( 엔트리안에 동일한 셀 레벨 값을 가진 MBR
            존재 )
         delete node.MBR[], node.O_ID[];
       else
         delete node.MBR, node.O_ID, node.CLevel;
   }
2. if ( 노드의 전체(Union) MBR 영역 크기나 셀 레벨 값
     이 변경 )
     노드의 셀 레벨 값과 부모 노드의 CNSLevel 값을
     셀 레벨 테이블을 참조하여 재계산한 뒤 갱신
```

5. 실험 및 성능 평가

5.1 환경 설정

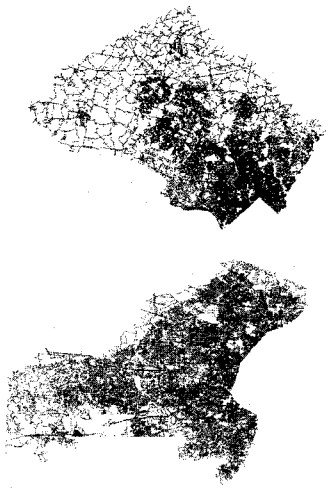
본 장에서는 위치 기반 질의시 공간 데이타에 셀 레벨 값을 반영하여 질의 영역과 연관되지 않은 셀 레벨 값을 갖는 공간 데이타는 검색에서 제외하는 CLR-트리를 구현하여 실험하였다. 대용량의 공간 정보를 관리하는 공간 데이타베이스에서 사용자의 질의 시간은 주기억 장치 또는 CPU의 단위 사용시간보다는 상대적으로 크게 디스크 입출력에 좌우된다. 따라서 본 논문에서는 차원은 2차원으로 기존 R-tree와의 삽입, 검색 시 입출력(Page I/O)수를 기준으로 성능을 평가하였다. 시스템 환경은 <표4>와 같다.

<표 4> 시스템 환경

중앙 처리 장치	Pentinum 2.0Ghz
메인 메모리	1024MB
디스크 크기	40GB
운영 체제	Windows 2000 Professional
프로그램 언어	Java (v.1.4.1)

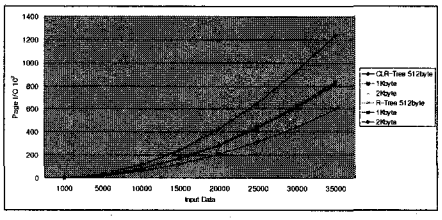
성능 평가에 사용될 데이타의 집합은 <그림 14>와 같이 실제 지리데이타 Tiger System(U.S. dataset)에 사용되는 Maryland주의 Montgomery Road 39570건, Missouri주의 St. Louis Road 51311건의 라인(line) 데이타를 대상으로 실험하였다. 실험 평가 시 사용되는 질의 영역은 전체 영역의 0~30% 사이로 점차 증가하며, 무작위(random)로 10,000번씩 질의 수행하였고, 이에 따른 입출력(Page I/O)수를 비교 분석하였다. 페이지 크기는 각각 512Byte, 1KByte, 2KByte로 설정하여 실험하였다.

5.2 성능 비교

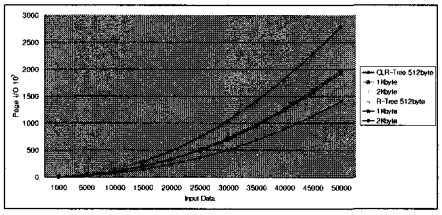


<그림 14> Montgomery county in Maryland와 St. Louis county in Missouri

<그림 15>는 페이지 크기 512Byte에서 2KByte까지 CLR-트리에 입력되는 데이타의 양을 증가시키며 디스크 접근 횟수를 비교하였다. CLR-트리는 셀 레벨 값의 저장과 재 계산을 통해 디스크 접근 수가 증가하므로 R-tree보다 약 5% 정도 낮은 성능을 보인다.

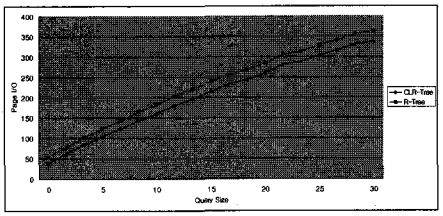


(a) Montgomery County

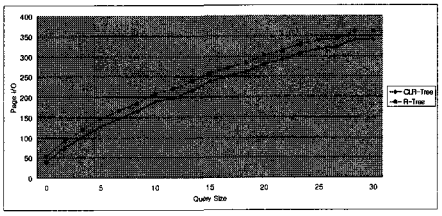


(b) St. Louis County

<그림 15> 페이지 크기에 따른 공간 객체 삽입

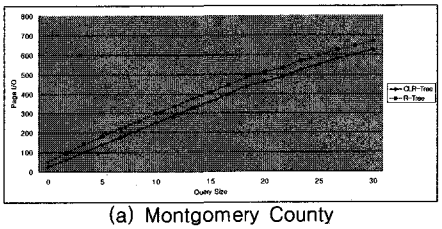


(a) Montgomery County

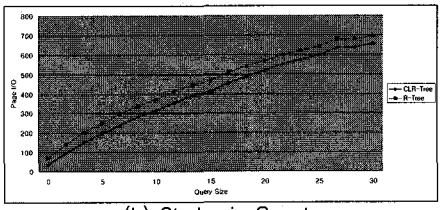


(b) St. Louis County

<그림 16> 페이지 크기 2KByte로 구성된 CLR-트리의 검색 결과



(a) Montgomery County

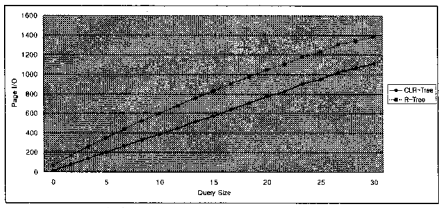


(b) St. Louis County

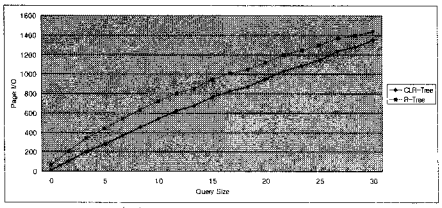
<그림 17> 페이지 크기 1KByte로 구성된 CLR-트리의 검색 결과

<그림 16>, <그림 17>에서는 페이지 크기를 2KByte, 1KByte로 각각 설정하여 CLR-트리에 색인을 구성한 후에 위치 기반 질의 검색에 따른 디스크 접근 수를 나타낸 것이다. 페이지 크기가 2KByte나 1KByte일 경우에는 하나의 노드에 들어가는 공간 데이타의 수가 많아지며 질의 영역과 동일한 레벨을 가

진 노드의 수가 증가함에도 불구하고 기존 R-tree보다 디스크 접근 수에서 약 10%의 향상을 보여주고 있다.



(a) Montgomery County



(b) St. Louis County

<그림 18> 페이지 크기 512Byte로 구성된 CLR-트리의 검색 결과

<그림 18>에서는 페이지 크기를 512Byte로 설정하여 동일한 데이타에 대해 테스트 한 결과, 질의 영역과 동일한 셀 레벨 값을 지닌 공간 데이타가 노드 안에 존재하는 수가 적어지므로 기존 R-tree에서의 출력 수보다 현저히 줄어든 입출력 수를 결과 그래프를 통해 확인할 수 있다.

공간 색인 트리에서 삭제와 갱신 연산보다 삽입 연산과 검색 연산이 중요하게 차지한다. 동적 셀 레벨링을 도입함에 따라 삽입연산에 대해서는 약 5%의 추가 비용이 소요되며 검색 시에는 약 10%의 비용 절감(디스크 접근 횟수 기준)을 가져오는 것으로 실험을 통해서 알 수 있었다. 그러나 일반적으로 삽입 연산의 횟수보다 검색 연산의 횟수가 상대적으로 많기 때문에 검색 연산을 많이 실행하면 할수록 삽입연산의 비용의 상쇄 효과를 볼 수 있을 것이라고 판단된다. 또한, 공간 데이타의 양이 많아지고, 사용자의 위치 기반 질의가 증가되어도, CLR-트리를 사용하는 경우 CLR-트리를 도입하지 않은 경우보다 빠른 사용자 응답시간을 제공할 수 있다.

6. 결론 및 향후 연구

무선 통신의 발달로 사용자의 위치를 기반으로 사람이나 사물의 위치를 정확하게 파악하며 이를 활용하는 응용 시스템 및 서비스 요구가 현재 급증하고 있다. 사용자가 요구하는 데이타는 위치라는 정보를 지닌 지리적 공간 데이타로 표현 가능하며 이 데이타를 기반으로 공간 데이타베이스를 구축하고 서비스에 활용하게 된다. 공간 데이타베이스를 통해 위치 기반 서비스를 이용하기 위해서는 다차원 공간 색인 기법을 사용하여 공간 질의를 처리하게 되며 대표적인 공간 색인으로는 R-tree가 널리 사용되고 있다.

하지만, R-tree를 이용한 검색으로는 질의 영역과 관계없는 공간 데이타까지 검색하여 불필요한 입출력을 유발하며, 사용자가 증가되고 공간 데이타베이스가 대용량화 될수록 사용자가 요구하는 시간 안에 공간 데이타를 빠르고 효율적으로 검색하지 못하게 된다.

본 논문에서는 위치 기반 질의 시 질의와 연관되지 않은 영역을 제거하여 불필요한 디스크 입출력을 감소시키기 위해, 공간 데이타가 저장된 노드의 전체(Union) MBR 영역을 4개의 셀로 분할하고, 사용자의 질의 영역과 공간 데이타간에 연관된 셀만을 선택하여 검색하도록 하는 색인 기법인 CLR-트리를 제안하였다. 사용자의 질의 영역은 셀 레벨 테이블을 참조하여 질의 처리 시 공간 데이타의 셀 레벨 값과 AND 연산을 통해 질의 영역에 포함되지 않은 질의 대상 객체 수를 줄였으며, 실험을 통해 제안된 CLR-트리의 효율성을 입증하였다. 제안된 CLR-트리의 동적 셀 레벨링 기법과 제안된 삽입, 분할, 삭제 및 검색 알고리즘은 R-tree뿐만 아니라, 공간 색인에서 MBR을 사용하는 다른 R-tree 계열의 색인 기법(R*-tree 또는 R+-tree 등)에 대해서도 적용 가능하다.

향후 연구로는 사용자의 이동성을 고려한 실시간 질의 검색을 지원할 수 있도록 인덱스 구조의 확장[15]과 질의시 MBR 영역 결정 방법을 달리하여 필터링 효율을 높일 수 있는 연구가 필요하다.

참고문헌

[1] Zhang, J "Location-Based Spatial Queries," Proceedings of ACM Conference on Management of Data (SIGMOD), pp.467-478, San Diego, CA, 2003.

[2] Ayse Y. Seydim, "An Architecture for Location Dependent Query Processing," Proceedings of the International Workshop on Mobility in Databases and Distributed Systems (MDDS' 01), 2001.

[3] Guting, R.H, "An Introduction to Spatial Database System," VLDB Journal, Vol 3, No. 4, pp.357-399, 1994.

[4] Guttman, A, "R-Tree: An Dynamic Index Structure for Spatial Searching," Proceedings of the ACM SIGMOD, pp.47-57, 1984.

[5] N. Beckmann, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proceedings of the ACM SIGMOD, pp.322-331, 1990.

[6] I. Gargantini, "An effective way to represent quadtrees," Communications of ACM(CACM), Vol 25, No 12, pp.905-910, 1982.

[7] M. H. Dunham, V. Kumar, "Location dependent data and its management in mobile databases," Proceedings of DEXA Workshop, pp. 414-419, Vienna, Austria, 1998.

[8] Timos Sellis, "The R+-Tree : A Dynamic index for multi-dimensional objects," Proceedings of the 13th VLDB Conference, Brighton, 1987.

[9] Henrich, A, "The LSD tree: Spatial access to multidimensional point and non-point objects," Proceedings of the 15th International Conference on VLDB, pp.45-53, 1989.

[10] S. Berchtold, "The X-Tree: an index structure for high dimensional data," Proceedings of the 22nd International Conference on VLDB, 1996.

[11] R. Schneider, H. P. Kriegel, "The TR*-tree: A new representation of polygonal objects supporting spatial queries and operations," Proceedings of the 7th Workshop on computational Geometry, Lecture Notes in Computer Science 553, Springer-Verlag, pp.249-264, 1991.

[12] Jack A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System," SIGMOD Conference, pp.326-336, 1986.

[13] T.Brinkhoff, "Multi-step Processing of Spatial Joins," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.197-208, 1994.

[14] I.kamel, C. Faloutsos (1994), "On Packing R-tree," Proceedings of the Information and Knowledge Management (CIKM), pp.490-499, 1994.

[15] 전봉기, 홍봉희, "이동체의 색인을 위한 시간 기반 R-트리의 설계 및 구현", 한국정보과학회, Vol. 30, No. 3, pp.320-335, 2004.

정연욱

2002년 한서대학교 컴퓨터정보학과 (공학사)

2004년 인하대학교 정보통신대학원 (공학석사)

2004년 ~ 현재 팅크웨어 연구원

관심분야 : 이동 데이타베이스, 지리 정보 시스템

구경이

1997년 인하대학교 전자계산공학과 (공학사)

1999년 인하대학교 전자계산공학과 (공학석사)

2004년 인하대학교 컴퓨터 · 정보공학과(공학박사)

2004년 ~ 현재 인하대학교 박사후과정

관심분야 : 이동 데이터베이스, 멀티미디어 데이타베이스

김유성

1986년 인하대학교 전자계산학과 (이학사)

1988년 한국과학기술원 전산학과 (공학석사)

1992년 한국과학기술원 전산학과(공학박사)

1996년 ~ 1997년 미국, 퍼듀대학교 전산학과 방문연구원

1992년 ~ 현재 인하대학교 정보통신공학과 교수

관심분야 : 이동 데이타베이스, 멀티미디어 정보검색, 트랜잭션 관리