□ 기술해설 □

GIS를 위한 공간 색인 및 공간 질의 처리 기법

한국과학기술원 김덕환 · 박호현 · 정진완*

목 차

1. 서 론
2. 공간 색인 기법
2.1 점 접근 방법
2.2 공간 접근 방법의 분류
2.3 R 트리
2.4 R^+ 트리
2.5 R* 트리
3. 공간 질의 처리 기법
3.1 공간 색인을 이용한 질의 처리
3.2 근사 기법
3.3 질의 최적화
4. 연구 동향
5. 결 론

1. 서 론

지리 정보 시스템(Geographic Information System)은 지리적인 데이타를 효과적으로 저장, 갱신, 조정 및 분석하며 지도의 형태로 데이타를 표시할 수 있는 컴퓨터 소프트웨어 시스템이다. GIS에서 지리적인 데이타를 저장 및 검색하기 위하여 특성에 맞는 데이타베이스 관리 시스템이 필요하다.

기존의 데이타베이스 시스템은 문자, 숫자 등의 정형 데이타는 효율적으로 검색할 수 있으나 위치 정보를 표현하는 점, 선분, 다각형 등의 벡타 데이타와 화상(image)과 같은 대용량의 래스터 데이타를 취급하기는 어렵다.

GIS에서 데이타베이스는 2차원 지도상에 존재하는 공간 객체들을 저장한다. 각 공간 객체는 도시, 도로, 호수 등과 같은 특정한 엔티티 클래스에 속하는 것으로 분류될 수 있다. 객체들은 비공간 속성(예 : 인구, 이름, 용도)과 공간 속성(예 : 위치)에 의해 표현된다. 이와 같은 속성을 갖는 데이타베이스를 공간 데이타베이스(spatial database)라고 한다. 공간 데이타베이스에서는 효율적으로 공간 객체들을 검색하기 위해 교차(intersect), 포함(contains), 근접(proximity) 질의를 위한 공간 연산자들이 추가된다. 예를 들어 "지도상의 어느 한 지점(예 : 서울)에서 5 Km 이내에 있는 도시를 찾아라"와 같은 영역 질의(range query)의 경우 주어진 질의 영역과 겹치는 모든 객체들을 검색하게 된다. 영역 질의의 특별한 경우가 질의 영역이 점으로 줄어드는 완전 일치 질의(exact match query) 또는 점질의(point query)이다. 이와 같이 공간 관계를 다루는 질의를 효율적으로 처리하기 위해 별도의 색인 구조에 의존해야 한다.

본 글에서는 공간 데이타베이스를 관리하기 위한 공간 질의 처리와 공간 색인 기법에 대해 소개하므로써 공간 데이타베이스 기술에 대한 이해를 돕고자 한다. 2장과 3장에서는 지금까지 연구된 공간 색인 및 공간 질의 처리기법에 대하여 기술하고 4장에서는 이들에 대한 최근의 연구 동향을 살펴보며 마지막으로 5장에서 결론을 맺는다.

*정회원

2. 공간 색인 기법

GIS에서 데이타베이스는 다차원 공간상에 있는 데이타 객체들을 다룬다. 효율적인 질의 처리를 위해 객체들에 대한 공간 색인을 지원하는 별도의 자료 구조가 필요하다. 주어진 질의 영역내에 있는 객체들을 찾거나 객체에 근접해 있는 다른 객체를 검색하는 공간 연산들은 공간 색인을 이용하여 효율적으로 지원될 수 있다.

기존의 데이타베이스 시스템은 문자, 숫자 등의 정형 데이타는 효율적으로 검색할 수 있으나 공간 관계에 기반을 두고 데이타를 접근하기는 어렵다. 여기서 공간 관계란 2차원 공간에서 2 영역 사이의 overlap, disjoint, meet, equal, contains, inside, cover, covered-by 등의 위상 관계를 의미한다. 또한 데이타 객체들의 방대한 크기로 인해 모든 데이타 객체들 사이의 공간 관계를 미리 계산하여 저장하는 것은 비효율적이다. 대신에 질의처리시 필요할 때마다 공간 관계를 동적으로 생성할 수 있어야 하며 특히 인접한 공간 객체를 찾기 위해 공간적 위치를 기반으로 하는 색인 기법이 필요하다.

GIS에서 도로, 호수, 행정 구역과 같은 객체들은 불규칙한 형태를 갖고 있어 실제 객체의 정확한 크기와 위치를 사용한 공간 탐색(예 : 교차 질의, 포함 질의)을 실행하는 비용이 비싸진다. 따라서 초기에 근사(approximation)나 여과(filtering)를 하여 탐색되는 객체 수를 줄여야 한다. 영역 분할(region decomposition)기법과 최소 경계 사각형(Minimum Bounding Rectangle)기법이 불규칙한 형태의 공간 객체를 근사하기 위해 사용된다.

영역 분할은 quad 트리라는 자료구조를 이용하여 공간 객체를 적절한 해상도를 가진 겹치지 않은 래스터 사각형으로 모자이크하는 방법이며 이미지 처리 응용에 적합하다. MBR은 객체를 둘러싼 최소 경계 사각형이며 근접(proximity) 질의 처리에 효율적이다. 두 공간 객체의 MBR들이 겹치지 않으면 두 객체들도 겹치지 않는 성질을 이용하여 두 다각형에 대한 교차 질의시 빠른 속도로 질의를 처리할 수 있게 해준다.

2.1절에서는 점 접근 방법(Point Access Method)을 살펴보고 2.2절에서는 지금까지 제안된 공간 색인 기법들을 3가지 형태의 공간 접근 방법(Spatial Access Method)으로 분류하고 2.3절, 2.4절, 2.5절에서는 특히 최소 경계 사각형(MBR)을 이용한 색인 기법중 많이 쓰이는 R 트리, R^+ 트리, R* 트리의 구조를 살펴보고 이들의 장단점을 기술하고자 한다.

2.1 점 접근 방법(Point Access Method)

B 트리, ISAM 색인, 해쉬, 이진 트리 등의 다양한 자료 구조가 데이타의 삽입,삭제 및 효율적인 검색을 위해 사용되었다. 그러나 이 기법들은 다차원 공간에서의 영역 질의(range query)에는 적합하지 않다. 다차원 자료 구조로서 grid-file, 다차원 B 트리, kd 트리와 quad 트리 등이 제안되었으며 이 기법들은 다차원 공간에서 점으로 표현되는 데이타 객체들을 색인하기 위해 설계되었으므로 점 접근 방법(Point Access Method: PAM)이라고 한다. 그러나 PAM은 자동 지도 제작이나 이미지 처리에 적합하지 않다. 특히 사각형, 다각형, 원과 같은 다차원 공간 객체들을 구성하기 위해 새로운 색인 기법들이 제안되었다.

2.2 공간 접근 방법 (Spatial Access Method)

지금까지 다양한 공간 색인 기법들이 제안되었으며 이들을 3가지 형태의 공간 접근 방법(Spatial Access Method: SAM)으로 분류할 수 있다. 또한 PAM을 다차원 공간의 영역에 대한 색인으로 확장하는 기법들도 SAM에 의해 분류할 수 있다.

2.2.1 객체 사상(Object Mapping)

k차원의 공간 객체를 2k 차원의 공간에 있는 점으로 변환하며 Grid-file, BANG file, K-D-B 트리[1], hB 트리, LSD 트리 등이 이에 해당한다. 객체 사상의 다른 방법으로 공간 객체를 더 낮은 차원 공간에 있는 점으로 사상할 수 있으며 Peano 곡선이나 Hilbert 곡선 등의 공간 순서화

기법(space-filling curve technique)을 이용한 색인 기법들(예 : Z-순서 색인 기법[2])이 제안되었다.

2.2.2 객체 중복(Object Duplication) 또는 객체 분할(Object Clipping)

k차원의 데이타 공간을 겹치지 않는 영역들로 분할하는 bucketing 기법으로 공간 객체가 둘 이상의 영역과 겹치는 경우 공간 객체를 잘라서 겹치지 않는 영역에 나누어 저장한다. grid file, EXCELL, mkd 트리, R^+ 트리[3], cell 트리, BANG file 등이 이에 해당한다.

2.2.3 겹치는 객체 영역 (Object Bounding)

k차원의 데이타 공간을 겹치는 영역들로 분할하는 bucketing 기법으로 공간 객체를 영역안에 완전히 포함되도록 한다. R 트리[4], R* 트리[5], Buddy 트리, skd 트리, R files, PLOP-Hashing, Packed R 트리 등이 이에 해당한다.

2.3 R 트리

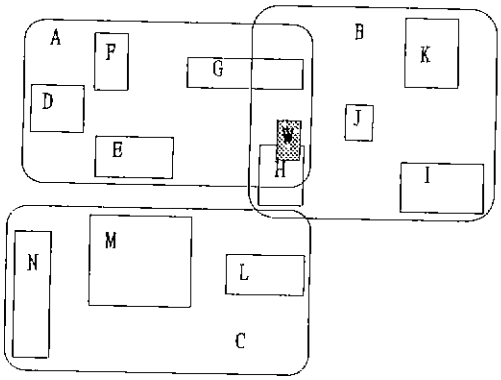
R 트리[4]는 k차원의 공간 객체를 자르거나 변환하지 않고 겹치는 k차원의 사각형 영역안에 객체가 완전히 포함되도록 하는 B 트리와 비슷한 구조의 높이 균형 트리(height balanced tree)이다.

리프 노드는 〈rectangle, object-identifier〉 형태의 엔트리를 가지고 있으며 object-identifier는 데이타베이스에 저장된 공간 객체 식별자이고 rectangle은 그 공간 객체를 둘러싸는 k차원의 사각형 영역이다.

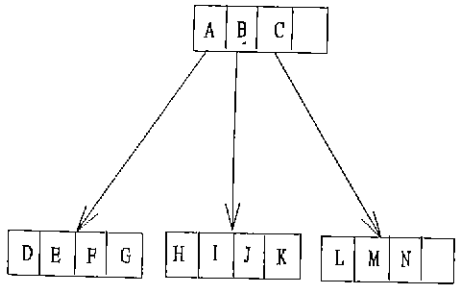
중간 노드나 루트 노드는 〈rectangle, child-pointer〉 형태의 엔트리를 가지고 있다. 여기서 child-pointer는 자식 노드의 주소 지시자이며 rectangle은 자식 노드의 모든 사각형 영역을 포함하는 MBR 영역이다.

그림 1은 R 트리와 저장된 공간 객체들을 나타내고 있다.

R 트리의 검색 알고리즘은 루트부터 시작해서 트리의 아래 방향으로 검색하며 질의 영역과 겹치는 중간 노드의 사각형들에 대해 대응하는 자식 노드들을 루트로 하여 재귀적으로 검색을 한다. R 트리에서는 많은 영역들이 서로 겹쳐질 수 있기 때문에 트리의 검색 경로가 많아지게 되어 검색할 때 좋지 않은 성능을 나타낼 수도 있다.



(a) R 트리를 형성하기 위해 그룹지어진 사각형들과 질의 윈도우의 예



(b) 위의 사각형을 위한 R 트리

그림 1

R 트리의 삽입 알고리즘은 중간 노드의 사각형 영역이 최소로 증가하도록 하여 간접적으로 겹치는 영역을 줄인다. 그리고 삽입할 리프 노드에 범람(overflow)이 발생했을 때 리프 노드에서 분할(split)이 일어나고 계속해서 그 상위 노드에도 분할이 일어날 수 있다. 새로운 객체를 리프 노드에 삽입한 후 삽입된 객체를 포함하는 사각형 영역을 가진 상위 노드의 사각형 영역의 크기가 변경되어야 한다. 즉 새로운 객체를 포함하도록 MBR 영역의 크기가 조정되어야 한다.

R 트리의 삭제 알고리즘은 객체를 삭제한 후 그 노드의 엔트리수가 최소 엔트리수에 미달하면

그 노드를 삭제하고 그 노드의 엔트리들을 재삽입(Re-insertion)하는 전략을 쓴다. 또한 리프 노드에서 객체를 하나 삭제하면 삭제된 객체를 포함하는 사각형 영역을 가진 상위 노드의 MBR 영역의 크기는 나머지 객체들의 사각형 영역으로 조정해야 한다.

R 트리의 분할 알고리즘은 분할 후 두노드들의 MBR 영역들의 면적이 최소가 되도록 하며 가능하면 분할 결과로 나온 새로운 영역이 다른 영역들과 겹치는 영역의 크기가 작을수록 좋다.

R 트리의 분할 알고리즘은 크게 3가지로 나누어 볼 수 있다.

첫째는 지수 함수 비용 알고리즘으로서 범람된 노드에 있는 모든 객체 쌍을 비교하여 영역의 겹침이 가장 작은 2개의 그룹으로 분할한다. 노드의 최대 엔트리수를 M이라고 할 때 알고리즘의 비용은 $O(2^{M-1})$이며 너무 느리다.

둘째는 2차 함수 비용 알고리즘으로서 범람된 노드에 존재하는 엔트리들 중에 가장 멀리 떨어진 2 사각형 영역을 구해 각기 다른 2개의 그룹으로 분할해 간다. k 차원 객체들에 대한 분할 비용은 $O(M^2k)$로 비용이 적게 드나 지수 함수 비용 알고리즘에서 얻을 수 있는 분할의 최적해를 이 알고리즘에서는 구하지 못한다.

세째는 선형 비용 알고리즘으로 2차 함수 비용 알고리즘보다 좋은 분할이 못되지만 가장 적은 분할 비용이 든다.

2.4 R^+ 트리

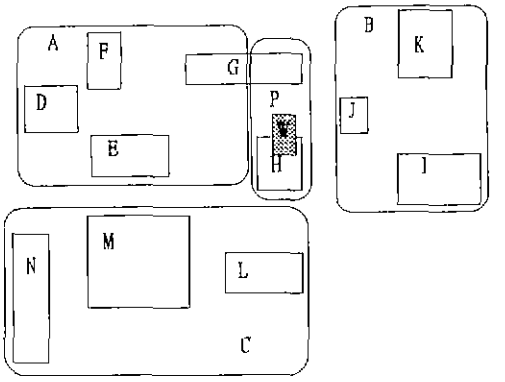
R^+ 트리[3]는 R 트리에 K-D-B 트리[1]의 장점을 추가하여 영역 사각형의 겹침을 제거함으로써 R 트리보다 검색 효율을 높였다.

R^+ 트리가 R 트리와 다른 점을 살펴보면 다음과 같다.

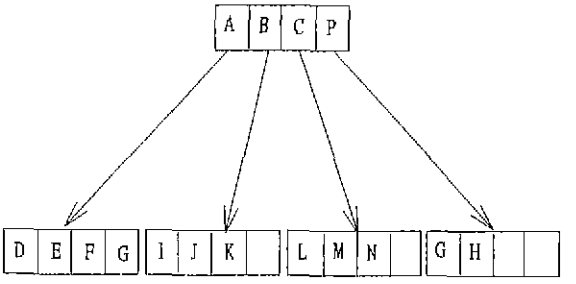
첫째, R^+ 트리의 노드들은 절반(M/2) 이하의 엔트리들로 채워질 수도 있다.

둘째, 임의의 중간 노드의 엔트리들은 겹치지 않는다.

세째, 공간 객체가 하나 이상의 리프 노드에 나누어 저장될 수 있다.



(a) R^+ 트리를 형성하기 위해 그룹지어진 사각형들과 질의 윈도우의 예



(b) 위의 사각형을 위한 R^+ 트리

그림 2

그림 2는 그림 1에 대한 R^+ 트리의 색인 구조를 보여주고 있다.

R 트리에서는 완전 일치 질의에 대해 검색 경로가 여러 개인데 비해 R^+ 트리에서는 영역이 겹쳐지는 것을 제거하였기 때문에 검색 경로가 유일하게 결정된다.

R^+ 트리의 삽입 알고리즘은 R 트리의 삽입 알고리즘과 비슷하지만 삽입되는 객체의 MBR이 여러개의 영역 사각형과 겹치는 경우 겹치는 영역의 아래 방향에 있는 리프 노드들에 각각 같은 객체를 저장하게 된다. R^+ 트리의 삽입 알고리즘은 완전하지 않다[6]. 때때로 엔트리들의 영역 사각형들이 새로운 객체를 포함하도록 확장되는 것을 서로 방해하기 때문이다. 이를 해결하기 위해 하나 이상의 영역 사각형이 분할되며 분할이 일어날 때 트리의 아래 방향으로도 일어날 수 있다. 즉 범람이 발생하지 않은 노드까지도 분할이 일어날 수 있으며 이로 인해 기억 장소의 효율성이 낮아질 수 있다.

R^+ 트리의 삭제 알고리즘도 R 트리의 삭제 알고리즘과 비슷하나 여러 개의 리프 노드들에 객체가 중복되어 저장된 경우 그 객체를 가리키는 모든 식별자를 삭제할 필요가 있다.

2.5 R* 트리

R* 트리[5]는 R 트리의 색인 구조와 같으며 중간 노드의 영역 사각형들이 겹치는 것을 허용한다. R 트리가 색인 구조를 최적화하기 위해 1가지 기준을 적용하여 영역 사각형의 면적을 최소화하는 것에 비해 R* 트리는 4가지의 최적화 기준을 적용한다.

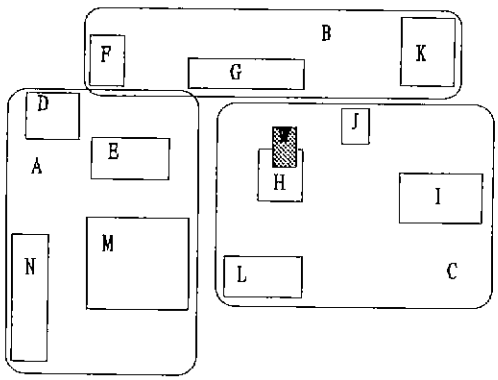
1) 영역 사각형들간에 겹치는 영역(overlap)의 크기를 최소화 한다.
2) 영역 사각형의 면적(coverage)을 최소화 한다.
3) 영역 사각형의 둘레(margin)의 합을 최소화 한다.
4) 기억 장소 이용률을 최적화 한다.

그림 3은 그림 1에 대한 R* 트리의 색인 구조를 보여주고 있다.

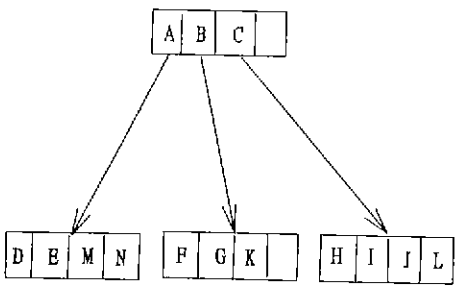
R* 트리의 삽입 알고리즘은 R 트리의 삽입 알고리즘이 영역의 면적만을 고려하는 데 비해 영역의 면적, 둘레, 겹치는 영역의 크기를 모두 고려하여 공간 객체를 삽입할 최적의 리프 노드를 선택한다. R* 트리는 영역 사각형들간에 겹치는 영역의 크기를 최소화하기 때문에 완전 일치 질의와 작은 영역 질의에 대해 R 트리보다 검색 성능이 좋다.

새로운 공간 객체를 삽입할 때 임의의 레벨의 중간 노드나 리프 노드에서 최초로 범람이 발생하는 경우 강제적으로 그 노드에 있는 p개의 엔트리들을 재삽입한다. 여기서 p는 재삽입될 엔트리의 갯수로서 최대 엔트리수 M의 30%일 경우 가장 좋은 성능을 보여준다[5]. 범람이 발생하는 최초의 경우가 아니면 R 트리의 경우처럼 리프 노드에서 분할이 일어나고 계속해서 그 상위 노드에서도 분할이 일어날 수 있다. 노드의 범람시 강제적인 재삽입(forced reinsert) 전략은 노드 영역의 중심에서 먼거리에 있는 엔트리들을 재삽입하여 영역 사각형의 형태를 정사각형으로 만들고 노드간에 겹치는 영역을 줄이며 기억 장소 효율을 개선하고 분할이 적게 일어나도록 트리를 전체적으로 재구성한다.

R* 트리의 분할 알고리즘은 R 트리의 분할 알고리즘이 분할된 두 노드의 영역의 합이 최소가 되도록 하며 1가지 기준을 적용하는 데 비해 분할된 2 노드의 둘레의 합이 최소가 되도록 하며 그 다음으로 겹치는 영역의 크기가 최소가 되도록 하고 그 다음에는 영역의 합이 최소가 되도록 노드를 분할한다.



(a) R^- 트리를 형성하기 위해 그룹지어진 사각형들과 질의 윈도우의 예



(b) 위의 사각형을 위한 R^+ 트리

그림 3

3. 공간 질의 처리

공간 데이타베이스에서의 질의는 공간 연산자와 비공간 연산자가 함께 섞여서 나타난다. 공간 질의로는 점 질의(point query), 영역 질의(range query), 근접 질의(proximity query), 포함 질의

(containment query), 교차 질의(intersection query), 공간 죠인(spatial join) 등이 있다.

공간 질의는 참여하는 피연산자(operand)가 공간 변수(spatial variable)인지 공간 상수(spatial constant)인지에 따라 공간 선택(spatial selection) 또는 공간 죠인으로 다시 분류할 수 있다. 여기서 공간 변수라 함은 공간 데이타베이스에서 점, 선분, 다각형, 영역 등을 그 도메인(domain)으로 하는 임의의 공간 속성을 의미하며 공간 상수는 공간 변수와 같은 도메인에 속한 특정 위치를 말한다. 공간 선택은 점 질의 또는 영역 질의에서 처럼 피연산자중 하나는 공간 변수이고 다른 하나는 점, 창과 같은 공간 상수이다. 공간 선택에 대한 예가 아래에 나와 있다.

```
select all
  from roads
  where in_window(road_coords, w) and
    road_type=freeway;
```

위의 질의에서 공간연산자는 in_window (road_coords, w)이며 비공간 연산자는 road_type=freeway이다.

공간 죠인은 두 피연산자가 모두 공간 변수인 것으로 교차 질의, 포함 질의, 근접 질의 등이 있다. 아래의 예는 공간 죠인의 예를 보여주고 있다.

```
select road_name
  from roads regions
  where region_name=College Park
    and intersect(region_location, road_
coords);
```

3.1절, 3.2절에서는 공간 질의 처리에서 공간 색인 및 근사기법의 역할에 대하여 설명하며 3.3절에서는 공간 질의 처리 전략 및 최적화에 대해서 기술한다.

3.1 공간 색인을 이용한 질의 처리

관계형 데이타베이스에서 질의 처리 시간을 단축시키기 위하여 B-트리와 같은 색인 구조를 사용하듯이 공간 데이타베이스에서도 공간 질의 처리를 위하여 2장에서 언급된 형태의 공간 색인을 사용한다. 공간 색인 기법은 공간 선택과 공간 죠인에 모두 효율적으로 사용될 수 있다.

3.1.1 공간 선택

앞에서 공간 선택에는 점 질의, 영역 질의 등이 있다고 언급했다. 공간 색인을 이용한 점 질의 처리는 주어진 위치를 키로하여 그 위치를 포함하는 공간 객체를 공간 색인을 이용해 찾아가는 것이다. 점 질의에 대한 R-트리에서의 검색 과정이 2장에서 이미 언급되었다.

영역 질의 처리는 질의 영역과 겹치는 MBR을 공간 색인의 루트 노드로부터 시작해 찾는다. 공간 색인의 중간 노드에 대한 MBR이 질의 영역과 겹치면 그 노드의 자식 노드들을 검색한다. 결국 리프노드에서 질의 영 역과 겹치는 MBR을 가진 객체들이 영역 질의의 후보 집합이 된다. 이들 후보 집합들은 3.2절에서 언급할 다음 단계인 정제(refinement) 단계에서 기하 알고리즘을 써서 계속 처리된다.

3.1.2 공간 죠인

2장에서 언급된 많은 공간 색인 기법들에 대해 점 질의, 영역 질의 등의 공간 선택을 효율적으로 처리하기 위한 연구가 활발히 진행된 반면에 공간 데이타베이스에서 가장 비용이 많이 들고 중요한 연산인 공간 죠인에 대한 연구는 별로 많지 않았다. 최근에 들어와서는 기존의 공간 색인을 이용한 공간 죠인 방법[7] 및 공간 죠인에 적합한 새로운 공간 색인 기법[8]이 제안되었다.

공간 색인을 이용한 공간 죠인은 죠인을 할 두 공간 객체의 색인을 루트 노드로부터 차례대로 비교한다. 죠인을 하는 두 노드들에 속한 엔트리의 MBR에 대하여 공간 연산을 적용시켜 서로 겹치는 MBR의 쌍에 대해서는 그 자식 노드들에 대해 위의 방법을 계속 적용시키고 공간 연산을 만족하지 않는 MBR 쌍은 앞으로의 공간 연산에서 제거된다. 이렇게 하여 얻어진 리프 노드들의 MBR 쌍에 포함되는 객체들이 공간 질의에 대한 후보 집합이 된다. 이들 후보 집합들은 3.2절에서 언급할 다음 단계인 정제 단계에서 기하 알고리즘을 써서 계속 처리된다. 이와

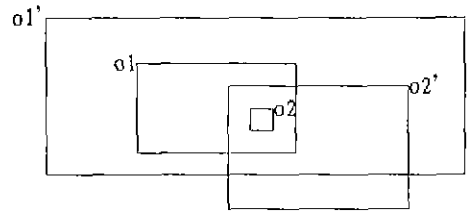


그림 4 o1'와 o2'는 교차하고 o1은 o2를 포함한다.

같이 공간 색인의 MBR을 이용한 조인을 MBR-공간조인 또는 간단히 줄여서 MBR-조인이라 한다([7], [9]).

MBR-조인에서 주의할 점은 원래의 공간 연산과 MBR-join에서의 공간 연산이 다를 수 있다는 점이다. 예를 들어 공간 연산이 포함 질의이면 중간 노드들의 MBR-join에서는 교차 질의가 적용되어야 한다[10]. 그림 4가 이를 잘 설명하고 있다. o2는 o1에 포함되지만 공간 색인에서는 o2에 대한 MBR o2'가 o1에 대한 MBR o1'에 포함되지 않고 교차된다.

3.2 근사 기법

공간 질의 처리시 각각의 공간 객체에 대해 공간 연산을 직접 적용하는 것은 상당히 많은 비용을 필요로 한다. 따라서 공간 질의 처리 비용을 줄이기 위해 공간 객체에 대해 직접 공간 연산을 적용하기 전에 각 공간 객체의 근사치를 구한 다음 그 근사치에 대해 공간 연산을 적용한다. 근사치에 대한 공간연산을 만족하는 객체에 대해서는 정확한 기하 알고리즘을 써서 공간 연산을 다시 실시한다. 이와 같이 공간 질의는 흔히 2단계를 거쳐 처리되는데 전단계를 여과(filtering) 단계라 하고 후단계를 정제(refinement) 단계라 한다.

다음 두 절에서는 근사기법에 기반을 둔 몇 가지 질의 처리 기법중 대표적인 2 가지 방법을 소개한다.

3.2.1 z-값에 의한 근사기법

이 방법은 PROBE[11]에서 사용하는 방법으로 전체 공간을 요소(elememt)라는 단위로 나누어 각 요소들의 z-값을 가지고 공간 객체를 근사시킨다. 요소들은 공간 객체의 그리드(grid) 표현 방법에 그 기본을 두고 있다.

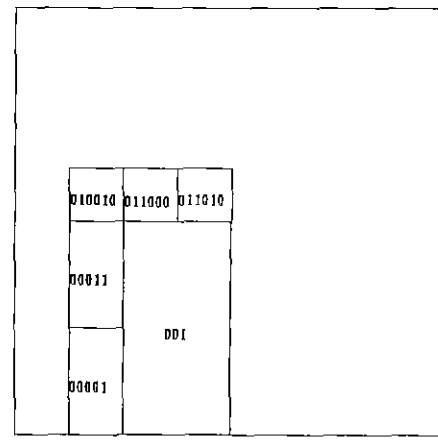


그림 5 영역 분할로 얻어진 요소들의 z-값들

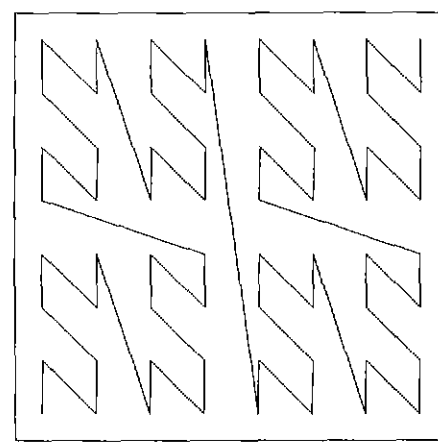


그림 6 영역 분할로 얻어진 요소들의 z-순서

전체 공간은 먼저 수직 분할되어 왼쪽 영역은 0(x0=0)의 값을 취하고 오른쪽 영역은 1(x0=1)의 값을 취한다. 다음 분할은 수평 분할이다. 앞의 수직 분할에서 얻어진 영역은 다시 수평 분할되어 아래쪽은 0(y0=0)의 값을 할당받고 윗쪽은 1(y0=1)의 값을 할당 받는다. 이런식으로 수직 분할과 수평분할을 계속 반복해서 얻어진 영역 r은 x0y0x1y1x2y2…와 같은 비트열(bit string)을 가지게 되는데 이 비트열이 영역 r에 대한 z-값이다. 분할 과정은 분할로 인해서 얻어진 영역이 공간 객체에 완전히 포함되거나 분할된 영역의 크기가 임계치(threshold)에 도달할 때까지 계속

된다. 그림 5는 영역 분할로 인해 얻어진 요소들의 예를 보여준다.

위에서 언급한 분할 방법에 의해 얻어진 영역이 요소이다. 한 공간 객체에 대해 얻어진 요소들이 그 객체에 대한 근사치를 형성하고 공간 질의 처리시 여과 단계에서 처리된다.

각 요소들의 z-값들은 그림 6에서 보는 것처럼 전체 공간에서 일정한 순서를 가지는데 이 순서를 z-순서라 한다. z-순서는 k-차원 데이타를 1-차원으로 변환한 것으로 볼 수 있다. z-값은 1-차원 값이므로 R-트리와 같은 공간 색인을 사용할 필요없이 기존의 데이타베이스가 가지고 있는 B-트리와 같은 색인 기법을 그대로 사용할 수 있다.

z-값에 의한 근사 기법은 요소에 대한 임계치를 너무 작게 잡으면 공간 객체에 대한 요소가 너무 많이 생겨 기억장소 부담이 많아질 뿐만 아니라 여과 단계에서의 질의 처리 시간이 길어진다. 또한 임계치를 너무 크게 잡으면 근사할 때 false hit이 많이 생겨 근사의 정확도가 저하된다[2].

3.2.2 볼록 다각형에 의한 근사 기법

MBR에 의한 근사는 false hit이 많이 생겨 근사의 질이 좋지 않다. 따라서 보다 높은 질을 가진 근사기법이 필요한데 이것이 Brinkhoff 등이 제안한 볼록 다각형(convex polygon)에 의한 근사 기법이다[12].

Brinkhoff 등은 3단계 공간 죠인 방법을 제시하였다[9]. 첫 단계는 3.1.2절에서 언급한 바 있는 공간 색인을 이용한 MBR-죠인이고 둘째 단계는 첫 단계의 MBR-죠인에서 나온 후보 MBR 쌍들의 집합에 대해 볼록 다각형을 가지고 한번 더 근사를 시킨다. 마지막 단계에서는 둘째 단계에서 나온 후보 집합에 대해 정확한 기하 알고리즘을 적용하여 최종 결과를 얻는다. 위의 과정이 그림 7에 잘 표현되어 있다.

볼록 다각형에 의한 근사 기법으로는 Rotated minimum bounding rectangle(RMBR), Convex hull(CH), Minimum bounding m-corner(m-C), Minimum bounding circle(MBC), Minimum bounding ellipse(MBE) 등이 있다(그림 8). Convex hull은 근사의 정확도가 가장 좋지만 근사에

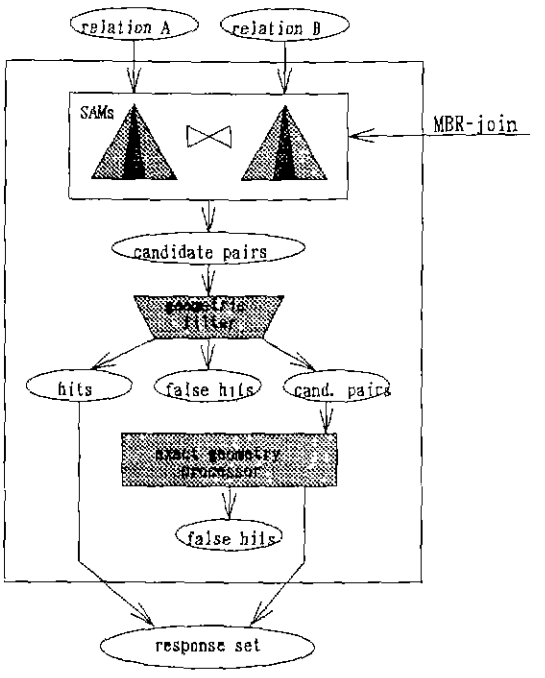


그림 7 공간 죠인 처리기

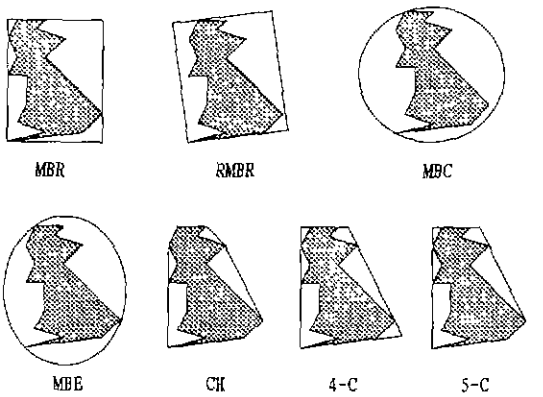


그림 8 볼록 다각형에 의한 근사 형태

필요한 파라미터의 갯수가 너무 많아 기억장소를 많이 필요로 하고 Minimum bounding circle은 기억장소 요구량은 적지만 근사의 정확도가 낮다. [12]에서는 근사의 정확도와 기억장소 부담에 대한 좋은 타협으로 5-corner에 의한 근사기법을 추천하고 있다.

3.3 질의 최적화

공간 질의 처리시 공간 연산을 먼저 처리할지 비공간 연산을 먼저 처리할지 결정을 해야 한다. 일반적으로 공간 데이타는 비공간 데이타보다 양이 방대하고 복잡하여 질의 처리 시간도 많이 소요된다. 그래서 지금까지 언급한 공간 색인 기법과 근사 기법을 이용하여 공간 질의를 처리한다.

비공간 연산을 먼저 처리하면 공간 연산에서 처리될 공간 데이타의 양을 미리 줄여 놓을 수 있어서 공간 연산에 소요될 시간을 단축시킬 수 있다. 그러나 비공간 연산을 적용한 후 생긴 중간 결과에 대해서는 색인 구조가 없기 때문에 공간 색인을 이용한 질의 처리의 잇점을 살릴 수 없다. 공간 연산을 먼저 처리하면 공간 색인을 효율적으로 활용할 수는 있지만 방대한 공간 데이타를 모두 질의 처리 대상으로 하는 것은 적은 부담이 아니다. 공간 연산과 비공간 연산을 함께 고려한 가격 모델을 설정하여 질의 최적화를 해야한다.

일반적인 질의 처리 전략 중 하나는 조인 연산을 하기전에 선택 연산을 먼저 처리하는 것이다. 비공간 선택 연산을 먼저 하고 공간 조인을 할 경우 위에서 언급한 것처럼 중간 결과에 대해서는 공간 색인을 활용할 수 없다. 그러나 공간 조인의 경우 색인이 없으면 질의 처리 비용이 너무 비싸다. 따라서 조인을 하기전에 중간 결과에 대해 공간 색인을 구성한 후 조인을 하면 효율적이다. 또한 공간 색인 구성 시간도 빨라야 한다. [8]에서는 중간 결과에 대해 seeded tree라는 공간 색인을 만들어 조인을 한다.

4. 연구 동향

지금까지 R 트리 계열의 색인 기법을 제안한 논문들은 동시성 제어와 회복 기법을 언급하지 않았다. 특히 색인 노드들의 분할(split)과 합병(merge)시 동시성을 극대화하고 회복 기능을 보장하면서도 회복으로 인한 부담을 최소화하도록 하는 전략을 추가해야 할 것이다.

GIS에서는 여러개의 주제도(thematic map)들이 존재하며 주제도의 레이어(layor)를 위한 색인들이 각각 존재할 수 있다. overlay질의는 2개 이상의 주제도 레이어를 필요로 하며 2 색인들을 쉽게 결합할 수 있어야 한다. 공간 순서화 기법을 이용한 구조는 두 색인을 쉽게 결합할 수 있도록 한다고 알려져 있다[13]. Hilbert R 트리, 다중 레이어 grid file, Peano 또는 Hilbert 키들을 이용한 quad 트리 등이 제안되었으며 앞으로도 다중 레이어를 위한 색인 기법에 대한 많은 연구가 필요하다.

질의 최적화 기법에 있어서 GEOQL[6], SAND 등에서 몇 가지 질의 최적화 전략을 세워놓고 있으나 아직 공간 질의 처리를 가격 모델은 별로 제시되지 않고 있다. 가격 모델에 기반을 둔 질의 처리 기법에 대한 연구가 필요하다.

마지막으로 GIS 응용들은 I/O bound이며 한 디스크에 들어갈 수 없을 정도의 대량의 데이타를 다루어야 한다. 다중처리기(multi-processor)를 이용한 parallel PMR-quad 트리 색인 기법[14]과 디스크 배열을 이용하여 병렬적인 I/O를 하는 parallel R 트리 기법[15] 등이 제안되었다. 본 연구실에서도 이 분야에 대한 연구를 진행하고 있으며 컴퓨터 아키텍처의 급속한 발전과 더불어 활발한 연구가 필요하다고 본다.

5. 결 론

지금까지 지리 정보 시스템을 위한 공간 색인 기법과 공간 질의 처리 기법에 대해 소개하고 그 장단점을 살펴보았다.

공간 색인 기법들은 SAM에 의해 분류해 보았으며 MBR을 이용한 색인 기법중 많이 쓰이는 R 트리, R^+ 트리, R^* 트리의 구조를 소개하고 이들의 장단점을 살펴보았다. 또한 공간 색인 기법에 대한 최근의 연구 동향을 간략하게 기술하였다.

공간 질의 처리 기법으로는 SAM을 이용한 근사 기법, z-값을 이용한 근사 기법과 볼록 다각형에 의한 근사 기법 등을 소개하였으며 공간 연산과 비공간 연산이 혼합된 공간 질의에서의 질의 최적화 방안에 대해서도 언급하였다.

GIS를 위한 공간 데이타베이스 시스템으로 관계형 DBMS가 많이 쓰이고 있으나 앞에서 살펴본 바와 같이 공간 데이타를 효율적으로 취급하기 위해서는 객체 지향 데이타 모델을 적용할

필요가 있다. 그리고 이에 대한 연구가 활발히 진행되고 있으나 아직 뚜렷한 결과가 없다. 따라서 앞으로 더욱 많은 연구가 필요한 분야라고 생각된다.

참고문헌

[1] Robinson. J.T., "The K-D-B Tree: A Search Structure for Large Multidmensional Dynamic Indexes", Proc. ACM SIGMOD Conf., 1981.

[2] J. A. Orenstein, "Redundancy in spatial databases", Proc. ACM SIGMOD Conf., 1989.

[3] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R^+-tree: A dynamic index for multi-dimensional objects", Proc. 13th VLDB Conf., Sep., 1987.

[4] A. Guttman, "R-trees: A dynamic index structure for spatial searching", Proc. ACM SIGMOD Conf., June, 1984.

[5] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles", Proc. ACM SIGMOD Conf., May, 1990.

[6] Beng Chin Ooi, "Efficient query processing in a geographic information system", Lecture Notes in Computer Science 471, Springer-Verlag, 1990.

[7] T. Brinkhoff, H. P. Kriegel and B. Seeger, "Efficient processing of spatial joins using R-tree", Proc. ACM SIGMOD Conf., 1993.

[8] M.L. Lo and C.V. Ravishankar, "Spatial join using seeded trees", Proc. ACM SIGMOD Conf., 1994.

[9] T. Brinkhoff, H.P. Kriegel, R. Schneider and B. Seeger, "Multi-step processing of spatial joins", Proc. ACM SIGMOD Conf., 1994.

[10] O. Gunther, "Efficient computation of spatial joins", Proc. 19th IEEE Data Engineering Conf., 1993.

[11] J.A. Orenstein, F. Manola, "PROBE spatial data modeling and query processing in an image database application", IEEE Transactions on Software Engineering, Vol. 14, 1988.

[12] T. Brinkhoff, H.P. Kriegel and R. Schneider, "Comparison of Approximations of Complex Objects used for Approximation-based Query Processing in Spatial Database Systems", Proc. ACM SIGMOD Conf., 1993.

[13] Hanan Samet, "The Design and Analysis of Spatial Data Structures", Addison Wesley, 1990.

[14] E.G. Hoel, Hanan Samet,"Performance of Data-Parallel Spatial Operations", Proc. 20th VLDB Conf., 1994.

[15] Ibrahim Kamel and C. Faloutsos, "Parallel R-trees", Proc. ACM SIGMOD Conf., June, 1992.

김 덕 환

1987 서울대학교 계산통계학과(이학사)
1987 ~현재 LG전자(주) 통신기기 연구소 선임 연구원
1993 ~현재 한국과학기술원 정보 및 통신공학과 석사 과정
관심 분야 : GIS, 공간 데이타베이스의 질의 처리 및 색인 기법, 객체 지향 시스템, 분산 시스템

소속 : SIGDB 회원

박 호 현

1987 서울대학교 계산통계학과(이학사)
1987 ~현재 삼성전자(주) 통신 개발실 주임 연구원
1993 ~현재 현재 한국과학기술원 정보 및 통신공학과 석사 과정
관심 분야 : GIS, 공간 데이타베이스, 객체 지향 데이타베이스, 분산 데이타베이스

정 진 완

1973 서울대학교 공과대학 전기공학과(학사)
1983 University of Michigan 컴퓨터 공학과(박사)
1983 ~1987 미국 GM 연구소 선임 연구원
1987 ~1993 미국 GM 연구소 책임 연구원
1993 ~현재 한국과학기술원 정보 및 통신공학과 부교수
관심 분야 : GIS, 객체지향 데이타베이스, 멀티미디어 데이타베이스, 분산 데이타베이스, CIM