위치 기반 질의를 지원하기 위한 셀 레벨링 공간 색인 기법

정연욱° 김유성
인하대학교 정보통신공학과
c2021027°@inhavision.inha.ac.kr, yskim@inha.ac.kr

(Cell Leveling Spatial Indexing Technique to Support Location Based Query)

Yun-Wook Jung° Yoo-Sung Kim
Dept. of Information Technology & Telecommunication, Inha University

요 약

최근 GPS기능을 탑재한 휴대폰·PDA등의 모바일 장치를 사용하여 위치 기반 서비스 (LBS : Location Based Service)를 이용하는 사용자가 급증하고 있다. 이에 대용량의 공간 데이터베이스에 대해 효율적 검색을 가능하게 하기 위한 색인이 필요하다. 공간 데이터베이스를 위한 다차원 공간 색인 기법으로는 R-Tree가 널리 사용되고 있다. 기존의 R-Tree를 이용한 검색은 질의 영역과 관계없는 공간 데이터까지 검색하는 고비용의 연산이 요구되며, 사용자의 질의 위치 단위(Granularity)를 고려하지 않아 사용자의 빠른 검색 응답시간 및 질의 영역에 대한 정확한 공간 객체 검색에 대해 충족하지 못한다. 이에 본 논문에서는 임의의 셀 안에 존재하는 공간 데이터가 자신이 속한 노드의 전체 MBR(Union MBR)영역과 셀 영역에 따라 셀 레벨 값을 구성하는 CLR-Tree(Cell Leveling R-Tree)를 제안한다. CLR-Tree를 사용할 경우 사용자의 질의 영역 셀 레벨 값과 데이터베이스에 저장된 공간 데이터의 셀 레벨 값을 비교한 뒤 겹침 연산 대상이 되는 공간 객체 수를 줄임으로써 검색 시간을 향상 시킬 수 있다.

1. 서 론

무선 통신 시스템의 발전으로 많은 사람들은 이동하는 동안 '언제(Anytime)', '어디서나(Anywhere)' 무선 통신을 통하여 서로 통신할 뿐 아니라 자신이 원하는 정보를 얻기를 원한다. 여기서 위치(Location)라는 정보는 위치 기반 서비스(LBS) 즉, 사용자의 위치를 기반으로 GPS기능을 탑재한 휴대폰·PDA등의 모바일 장치를 사용하여 사람이나 사물의 위치를 정확하게 파악하고 이를 활용하는 응용 시스템 및 서비스에 중요한 요소로 작용한다. 이러한 위치 정보를 바탕으로 구성된 지리적 공간 데이터는 대용량의 공간 데이터베이스 안에 저장되며 위치기반 질의 시 다차원 공간 색인 기법을 사용하여 검색하게 된다.[1, 2]

다차원 공간 색인 중에서 지리적 특성을 반영한 공간 데이터베이스의 대표적인 색인 방법으로 R-Tree[3]가 많이 사용되고 있다. R-Tree에 저장되는 공간 데이터는 지리적 위치를 기반으로 최소 경계 사각형(minimum bounding rectangle : MBR) 안에 단말 노드의 객체로 표현된다. 사용자가 '가장 가까운 호텔을 찾아라'라는 위치 기반 질의(LDQ : Location Based Query)에 대한 검색을 하고자 할 경우 지리적 위치를 기준으로 사용자의 질의 영역과 겹쳐지는 일정 거리 안에 모든 MBR을 검색 대상으로 한다. 이와 같은 검색 방법은 데이터베이스에 저장된 데이터 위치 단위(Granularity)와 사용자의 위치 단위가 서로 다를 경우 정확한 질의 검색을 못하게 되며 MBR내에 질의 영역과는 관계없는 공간 데이터까지 검색 후보 객체로 선택하여 겹침(Overlap) 및 거리(Distance) 연산을 수행하게 된다.

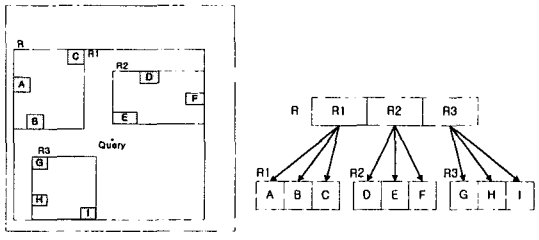
본 논문에서 제안하는 CLR-Tree는 R-Tree의 변형으로 위치 기반 질의 시 지리적인 위치 단위 정보를 포함한 공간 데이터의 검색을 위해 임의의 셀 안에 존재하는 공간 객체가 자신이 속한 전체 MBR(Union MBR)영역과 셀 영역에 따라 셀 레벨 값을 계산하고 노드에 저장한 뒤, 사용자 질의영역의 셀 레벨 값과 동일하지 않은 공간 객체는 질의 처리 시 제외하는 색인 기법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구와 문제점에 대해 살펴보고, 제 3장에서는 제안하고 있는 CLR-Tree 자료 구조를 소개한다. 제 4장에서는 제안된 CLR-Tree의 알고리즘을 소개하고 제 5장에서는 실험을 통한 성능을 평가하며 마지막으로 제 6장에서 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

공간 데이터는 공간 색인을 사용하여 검색하게 되는데 지금까지 연구되어온 공간 색인 기법은 크게 두 가지로 분류할 수 있다. 첫 번째는 최소 경계 사각형(MBR)을 기반으로 한 색인 구조들로서 질의에 만족하지 않는 공간 객체들을 신속하게 제거할 수는 있으나, 단순한 사각형으로 된 MBR로는 공간 객체의 속성 정보 및 위치단위를 정확하게 표현할 수 없으므로 질의 조건을 만족시키지 못하는 많은 불필요한 후보 객체들을 포함한다. 두 번째는 영역 분할(region decomposition)을 기반으로 한 색인 구조들로서 질의에 대한 검색 대상이 되는 공간 객체에 대한 정확도는 향상시킬 수 있으나, 수 많은 구성요소들로 인한 높은 질의 처리비용이 부담이 된다.

이러한 공간 데이터의 공간 질의 처리 시에는 비공간 질의에 비해 데이터의 복잡성과 방대함으로 인해 그 처리비용이 매우 비싼 편이다. 기존 연구[4, 5]에서는 공통적으로 공간 질의 처리 시 효과적인 처리를 위해 근사 기하 알고리즘(approximatio-n geometry algorithm)을 이용한 여과 단계(Filter Step)와 여과 단계 후에 선택되어진 MBR과 질의 영역과의 정확한 겹침 연산을 통해 공간객체를 검색하는 정제 단계(Refine Step)로 나누어 처리하는 방법을 제안하였다.



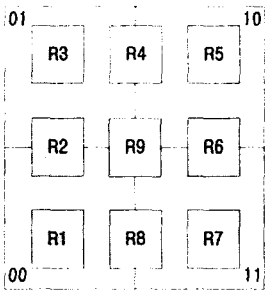
[그림 1] R-Tree 색인 구조로 구성

[그림 1]과 같이 R-Tree로 구성된 공간 색인 구조에서 루트 R아래에 서브 트리(Sub-Tree) R1, R2, R3가 존재하며, 이때 사용자는 자신의 위치 정보를 기반으로 영역질의를 통해 공간 객체를 검색하고자 한다고 가정하자. 질의에 나타난 위치정보는 위/경도, Cell ID, Zone 등으로 분류 될 수 있다. [그림 1]과 같은 자료구조에서는 위/경도 정보만으로 검색하고자 했을 경우 여과 연산에 의해 선택되어지는 MBR은 R1, R2, R3가 해당

되므로 공간 객체 9개를 대상으로 검색해야 되며, 이에 따라 질의 결과는 실제 질의 영역과 관계없는 서브 트리 R1, R3에 대한 불필요한 출력 연산이 발생된다.

3. CLR-Tree의 자료 구조

위치 기반 질의에서는 사용자의 위치 단위 레벨에 따라 질의 영역 데이터의 위치 바인딩이 틀려지며 질의 영역과 결과 등도 틀려진다.[5] CLR-Tree 공간 색인에서는 사용자의 질의 영역과 사용자의 위치 단위가 각각의 셀 레벨에 바인딩되며, 사용자의 위치 정보를 나타내는 셀 단위를 질의 처리 조건의 하나로 사용할 경우 각 영역을 하나의 셀 레벨 영역이라는 공통된 영역으로 간주하여, 이를 공간 질의 시 질의 검색 조건의 하나로 사용하며 질의 영역과 동일한 셀 레벨 값을 가진 노드만을 출력 하게된다.



	셀 레벨 테이블		
	셀 레벨 ID	이진수	십진수
R1	00	00000001	1
R2	00,01	00000010	2
R3	01	00000100	4
R4	01,10	00001000	8
R5	10	00010000	16
R6	10,11	00100000	32
R7	11	01000000	64
R8	11,00	10000000	128
R9	00,01,10,11	11111111	255

[그림 2] 입력 데이터의 Cell Level Value 경우의 수

[그림 2]는 공간 객체 자신이 포함된 전체(Union) MBR 영역 안에서 나타낼 수 있는 모든 셀 레벨 값으로 전체 영역을 사분할 한 뒤에 공간 객체는 R1에서 R9까지 시계 방향으로 모두 9개의 형태를 가질 수 있게 된다. 각각의 형태는 다시 이진 값으로 구성하여 8비트 범위(0~255) 안에서 모두 표현 가능하게 된다. 8비트를 이용하여 공간객체의 셀 레벨 값을 나타낸 이유는 비단말 노드 구조에서 자식 노드가 포함하고 있는 공간 객체를 독립된 형태로 구별하여 각각의 셀 레벨에 대한 합한 값을 저장하기 위한 것이다. 만약, 8비트 이하로 각 셀 레벨을 표현하게 될 경우 각각의 셀 레벨 값을 합하였을 때 개별적인 형태로 맵핑(Mapping)할 수 없게 된다. 이러한 셀 레벨 값을 저장할 수 있는 단말 노드의 엔트리(Entry) 형태는 아래와 같다.

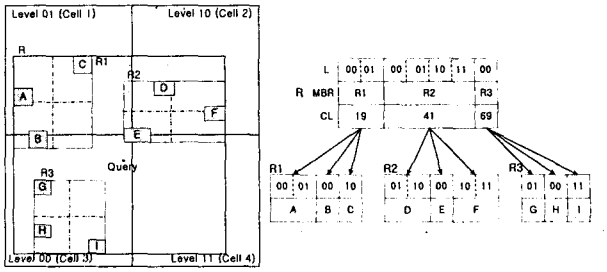
< LevelID, O_ID[], MBR[] >

LevelID는 단말 노드의 셀 레벨 값으로 임의의 셀 안에 존재하는 단말 노드 전체(Union) MBR 영역 안에서의 공간객체에 대한 위치 값을 나타내며 객체 식별자(O_ID)와 공간 객체의 MBR 정보가 동적 배열 구조로 되어 있는 것은 입력된 공간 객체가 저장되어질 노드안에 동일한 레벨 값을 지닌 엔트리가 있을 경우 엔트리의 MBR[] 배열 값을 확장시켜 저장하게 된다. 단말 노드가 아닌 비단말 노드의 엔트리 형태는 아래와 같다.

< CLevel_ID, LevelID, MBR, Child_Pointer[] >

CLevel_ID는 자식노드의 엔트리에 존재하는 셀 레벨(LevelID)값의 합을 나타내며, 비단말 노드의 셀 레벨 값인 LevelID는 전체 셀 영역 안에서 엔트리에 저장된 MBR에 대한 위치 값을 나타낸다. MBR은 자식노드의 엔트리에 존재하는 모든 MBR을 포함하는 사각형을 나타내며 Child_Pointer[]는 자식 노드의 주소를 가리키는 포인터를 의미한다.

위의 셀 레벨을 이용하여 [그림 1]의 자료구조를 CLR-Tree로 재구성하면 [그림 3]처럼 표현된다. 우선 루트 R 안에 포함된 서브 트리 R1, R2, R3 영역은 셀 영역안에서 셀 레벨 테이블을 참조하여 계산한 뒤 셀 레벨 값을 저장하며, 각 셀 안에 포함되어있는 MBR 영역 또한 사분할 처리한 뒤에 셀 레벨 테이블을 참조하여 셀 레벨 값 정보를 저장하게 된다.



[그림 3] CLR-Tree 색인 구조로 구성

CLR-Tree에서의 여과연산은 겹침 연산 후에 노드에 존재하는 셀 레벨 값과 AND(Bit) 연산을 실행하게 된다. [그림 3]에서 여과 연산시 루트 R과 질의 영역의 겹침 연산 후 선택된 후보 공간 객체는 R1, R2, R3에 해당되고 셀 레벨 값이 동일한 공간 객체를 선택하는 연산에서도 질의 영역의 셀 레벨 값 00, 01, 10, 11이 루트 R에 겹쳐진다. 하지만, 검색 영역이 되는 노드에 CLevel_ID값까지 비교하는 AND 연산을 하게 되면, 질의 영역의 셀 레벨 값은 R1에서 11, R2에서 00, R3에서 10의 셀 레벨 값을 가지게 되므로 AND 연산 시 True가 되는 공간 객체는 R2만 해당되며 R2 대해서만 출력연산이 일어나게 된다.

4. CLR-Tree 알고리즘

이번 장에서는 CLR-Tree의 삽입, 분할, 검색, 삭제 알고리즘을 제안한다. 제안한 색인 알고리즘은 아래와 같은 특성을 나타낸다.

- 위치 기반 질의 시 사용자의 지리적 위치뿐만 아니라 셀 위치 정보까지 질의 요건에 포함시킨 위치 단위 바인딩에 따른 검색이 가능하다.
- 질의 영역과 겹쳐지는 MBR의 공간 객체에 대한 셀 레벨 AND 연산으로 인해 검색 성능이 향상된다.

4.1 삽입 알고리즘

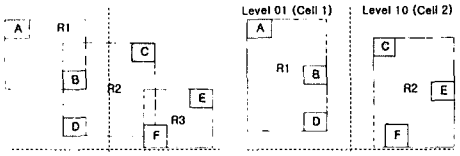
CLR-Tree에 새로운 공간 데이터를 삽입 하고자 할 경우 입력 데이터는 자신이 속한 전체(Union) MBR에 대응하여 셀 레벨 테이블을 참조 한 뒤 적절한 셀 레벨 값을 가지고 삽입되어야 하며, 노드 안에 최대 엔트리 개수 보다 작거나 같고 동적 배열로 구성된다.

```
Algorithm Insert(SpatialObject SOb)
//Insert Spatial Object cell level value lv_n (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. 삽입될 SOb의 셀 레벨 값 체크 한 후 삽입
SOb.lv_n = Cell_Level_Check(SOb, node);
while ( !node.last_entry )
  if ( node.level AND SOb.lv_n)
    해당되는 엔트리의 MBR[] 배열구조를 확장시켜 저장
  else
    빈 엔트리에 셀 레벨 값과 공간 데이터 저장
3. if ( 노드의 전체 영역 크기나 셀 레벨 값이 변경 )
     while ( !node.last_entry )
       노드 자신의 셀 레벨 값과 상위 노드의 CLevel_ID 값
       을 계산하여 갱신
```

4.2 분할 알고리즘

CLR-Tree에서는 노드의 분할이 일어나는 경우는 두 가지 형태로 구분되어진다. 첫 번째는 기존 R-Tree와 동일하게 공간 객체와의 영역 계산을 통해 빈 공간(Dead space)영역과 최소 겹침(Overlap)영역을 비교하여 MBR의 최소 영역을 기준으로 분할하게 되는 경우이다. 두 번째는 공간 질의 영역과 노드간의 셀 레벨 AND 연산을 위해 동일한 레벨 값을 가진 공간 객체를 중심으로 레벨 값에 따라 객체를 패킹(Packing)[6]하는

개념으로 먼저 동일한 레벨 값을 가진 공간 객체를 기준으로 두 그룹으로 묶은 뒤 이 두 그룹을 기준으로 나머지 공간 객체와의 빈 공간 영역과 최소 겹침 영역을 비교하여 분할하는 것이다.



[그림 4] a)R-Tree의 분할 방식 b)제안된 셀 레벨 분할 방식

```
Algorithm Split (Node node)
1. 분할된 노드를 저장할 새로운 노드 A, B 생성
2. 각 노드에 엔트리를 저장하기 전에 공간 데이터의 셀 레벨
   을 체크하여 동일한 레벨을 가진 두 그룹으로 구성
3. 새로운 노드 A, B에 두 그룹을 각각 저장시킨 후
   if ( 다른 셀 레벨 값을 가진 나머지 공간데이터가 존재 )
    while ( 남은 공간 데이터 )
      두 그룹과의 빈공간 영역과 최소겹침영역 비교한 뒤 저장
4. 분할된 노드가 비단말 노드일 경우 셀 영역에 대한 셀 레
   벨 값 계산한 뒤 저장
5. 분할된 노드가 단말 노드일 경우 MBR 전체 영역에 대한
   셀 레벨 값 계산한 뒤 저장
6. 분할된 노드의 상위노드가 존재할 경우 CLevel_ID 값 갱신
7. 분할이 완료된 뒤, 분할 전 노드 삭제
```

4.3 검색 알고리즘

CLR-Tree에서의 검색은 먼저 여과 단계에서 공간 객체에 대한 질의 영역의 겹침 연산과 셀 레벨 값간의 AND 연산을 모두 만족하는 후보 공간 객체를 선택한다. 그 후, 선택된 후보 공간 객체의 CLevel_ID에 대해 AND 연산을 실행하여 동일한 셀 레벨 값을 포함한 경우 후보 공간 객체로 선택한다. 정제 단계에서는 후보 공간 객체의 지리적 위치와 사용자의 지리적 위치 정보를 계산하여 정확한 겹침 연산에 따른 결과를 산출하게 된다.

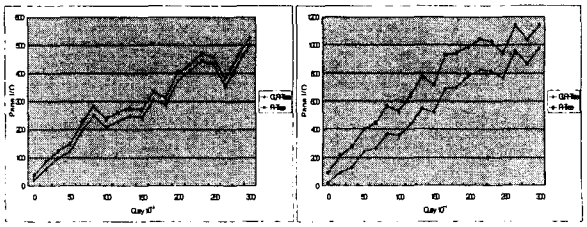
```
Algorithm Search (Node node, Query q)
//Query region cell level value lvn (n = 0,1,2…8)
1. if ( Overlap(q, node.Union_mbr) )
     if( node == Leaf_Node )
       질의영역과 공간 객체와의 Overlap(q, node.entry_mbr)
       연산 후 TRUE이면 return node.O_ID;
     else
       q.lvn = Cell_Level_Check(q, cell.Union_size);
       if ( q.lvn AND node.LevelID )
         후보 공간 객체에 대한 질의영역 셀 레벨 값
         q.lvn = Cell_Level_Check(q, node.entry_mbr);
         if ( q.lvn AND node.CLevel_ID )
           Search(node.childnode, q);
```

4.4 삭제 알고리즘

CLR-Tree에서의 삭제 방법은 먼저 삭제하고자 하는 공간 객체가 존재하는 노드를 검색한 뒤 공간 객체와 셀 레벨 값을 삭제시키면 된다. 하지만 이 때, 동일한 셀 레벨 값을 가진 공간 객체가 존재한다면 삭제되는 공간 객체의 MBR 정보만을 삭제시켜야 한다. 또한, 삭제 시 전체 MBR 크기가 변하게 된다면 노드에 존재하는 공간 객체의 셀 레벨 값과 자신의 상위 노드에서 사용되고 있는 CLevel_ID 값도 재 계산 해야한다.

5. 실험 및 성능 평가

본 장에서는 제안된 CLR-Tree을 구현하여 실험한다. 시스템 환경은 2.0Ghz CPU, 1G의 메모리, 언어는 Java(v1.4.1), OS는 Windows 2000으로 구현된 시스템에서 차원은 2차원으로 기존 R-Tree와의 검색 시 입출력(Page I/O)수를 기준으로 성능을 평가하였다. 성능 평가에 사용될 데이터의 집합은 실제 지리데이터 Tiger System(U.S. dataset)에 사용되는 Missouri주에 St.Louis Road 51311건의 라인(line) 데이터를 대상으로 테스트하였다. 실험 평가 시 구성되는 색인의 페이지 크기는 512Byte, 1Kbyte로 각각 설정하였으며 실행되는 질의영역은 전체영역의 0~30% 사이로 점차 증가하며 랜덤(random)하게 200번씩 질의 수행하였고 이에 따른 입출력 수를 비교 분석하였다.



[그림 5] Page Size 1Kbyte [그림 6] Page Size 512byte

[그림 5]에서는 페이지 크기를 1Kbyte로 설정하여 실험한 결과를 나타낸 것으로 이 경우에는 하나의 노드에 들어가는 공간 객체의 수가 많아지며 질의 영역과 동일한 셀 레벨을 가진 노드의 수가 증가하기 때문에 R-Tree와 거의 같은 수의 입출력이 일어났다. [그림 6]에서는 페이지 크기를 512byte로 설정한 뒤 동일한 데이터에 대해 테스트 한 결과, 노드 안에서 질의 영역과 동일한 셀 레벨 값을 가진 공간 객체의 수가 감소하게 되므로 기존 R-Tree에서의 입출력 수보다 현저히 줄어든 입출력 수를 결과 그래프를 통해 확인할 수 있다.

현재 무선 통신 요금이 패킷 요금제로 변화하고 있다는 점을 감안할 때 패킷 하나의 크기는 약 512byte이다. [그림 6]처럼 CLR-Tree로 공간 색인이 구성되었다면 실제 사용자의 모바일 장치에서 위치 기반 서비스를 받고자 할 경우 동일한 결과에 대해 낮은 비용으로 사용이 가능해지며 공간 데이터베이스에 저장되어 있는 공간 데이터 단위를 고려한 검색이 가능해진다.

6. 결론 및 향후 연구

본 논문에서는 임의의 셀 안에 존재하는 공간 데이터가 자신이 속한 노드의 전체 MBR영역과 셀 영역에 따라 셀 레벨 값을 구성하는 CLR-Tree를 제안하였다. 공간 데이터베이스에 저장된 공간 객체의 위치 단위와 사용자의 질의 위치 단위를 서로 반영한 CLR-Tree는 기존 공간 색인보다 효율적인 결과를 실험을 통해 증명하였다. 사용자의 질의 영역은 셀 레벨 테이블을 참조하여 질의 처리 시 공간 데이터베이스 안에 저장된 공간 객체의 위치 단위와 사용자의 위치 단위까지 고려하며 질의영역 안에 포함되지 않은 MBR의 질의 대상 객체 수를 줄여 디스크 출력 및 검색 시간을 향상시킬 수 있었다. 향후 연구로는 사용자의 이동성을 고려한 실시간 질의 검색을 지원하는 인덱스 구조로 확장하는 연구가 필요하다.

7. 참고문헌

[1] Zhang, J., Zhu, M., Papadias, D., Tao, Y., Lee, D. "Location-Based Spatial Queries. To appear in Proceedings of ACM Conference on Management of Data" SIGMOD, pp. 467-478, June 9-12, 2003

[2] Ayse Y. Seydim, Margaret H. Dunham, Vijay Kumar "An Architecture for Location Dependent Query Processing" MDDS 01, DEXA Workshop, 2001

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

[4] J. A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System.", ACM SIGMOD, 1986

[5] T.Brinkhoff, H. -P. Kriegel, R. Schneider, B. Seeger, "Multi-step Processing of Spatial Joins." ACM SIGMOD, 1994.

[6] I.kamel, C. Faloutsos, "On Packing R-tree" CIKM, 1993.