PML-tree : 대용량 공간데이터베이스를 위한 병렬처리 공간색인구조

방 갑 산†

요 약

본 논문에서는 PML-트리라는 공간색인구조를 제안한다. PML-트리는 object distribution heuristics를 사용하여 공간 데이터 객체를 여러 개의 데이터 공간에 균일하게 배치함으로써 질의처리 속도를 향상시킨다. 두 가지의 object distribution heuristics(absolute crowd index와 relative crowd index)가 제안이 된다. PML-트리는 공간 객체를 분배함으로써 R^+-트리의 말단 노드 내에 존재하는 데이터의 중복을 제거하면서, R-트리의 단점인 색인 사각형들 사이에 중첩을 허용치 않는다. PML-트리의 성능은 여러 타입의 테스트 데이터를 사용하여 MXR-트리와 비교된다. PML-트리는 MXR-트리에 비해 높은 공간활용도와 빠른 질의 반응시간을 보임으로써 공간 데이터베이스를 위한 효율적인 색인구조로 사용이 될 것으로 기대된다.

PML-tree : Parallel Spatial Index Structure for Large Spatial Databases

Kap-San Bang†

ABSTRACT

In this paper, a new dynamic parallel index structure called a parallel multiple-layer(PML) tree is proposed. The PML-tree increases speed of query processing by distributing data objects evenly among the multiple data spaces using object distribution heuristics. The author proposes and implements two heuristic methods, absolute crowd index and relative crowd index for an even distribution of objects over the multiple disks on which the PML-tree resides. The PML-tree does not require extra search paths as in R-tree, and does not contain any duplicated entries in leaf node as in R^+-tree. The performance the PML-tree and the MXR-tree are compared and analyzed using test data. Compared with the MXR-tree, the PML-tree increases space utilization and improves query performances on a system with multiple disks and expected as an efficient index structure for spatial databases.

1. 서 론

현대의 데이터베이스 시스템에 있어 공간 데이터(예 : 점, 선, 다각형, 곡선, … 등)의 역할은 대단히 중요한 자리를 차지하고 있으며, 그 응용분야를 점차로 늘려가고 있다. 공간 데이터는 많은 응용분야(예 : GIS [13], CAD, VLSI layout[1], 컴퓨터 비젼, 로보틱스, 이미지 데이터베이스, … 등[12, 16])에서 사용되고 있다. 또한 공간색인구조는 일반적인 1차원 데이터베이스에서 여러 개의 속성에 대한 다차원 질의를 할 때 k개의 속성을 k 차원 상의 한 점으로 표현함으로서 빠른 접근을 제공하기 위해 사용 가능하다. 방대한 양의 공간 데이터를 처리하기 위한 효율적인 공간색인구조는

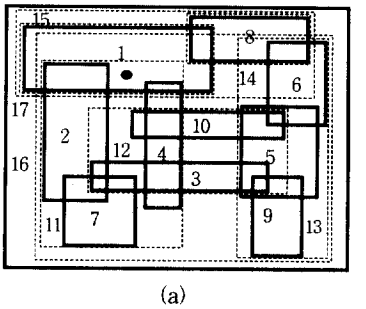
※ 본 논문은 '99 한성대학교 교내연구지원에 의한 것임.
† 정 회 원 : 한성대학교 정보시스템공학과 교수
논문접수 : 2000년 4월 4일, 심사완료 : 2000년 10월 18일

DBMS의 중요한 요소 가운데 하나이다.

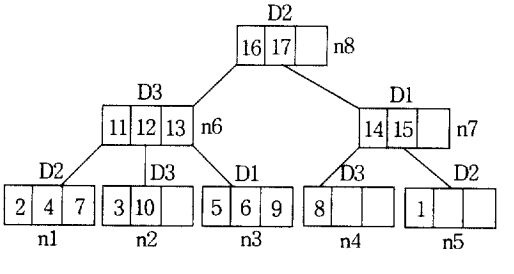
1차원 색인구조분야와는 달리 공간 색인구조분야에서는 어느 구조도 모든 응용분야에서 안정된 성능을 갖고있지 못하다. 그 이유는 응용분야에 따라 데이터의 분포와 질의의 형태가 다르며 공간색인구조에서 공간분할에 고려되어야 할 요소가 많기 때문이다. 공간색인구조에 있어서는 그 성능에 영향을 주는 인자가 많이 있으며 또한 그들이 상호간에 복잡하게 작용을 한다. 공간 데이터를 활용하는 시스템의 구축에 있어서 데이터의 성격과 데이터에 대한 질의의 형태가 특정한 어느 색인구조를 더 선호하도록 할 수 있기 때문에 공간색인구조의 선택은 응용분야에 따라 달라져야 한다. 예를 들어 GIS 와 VLSI에서의 데이터의 분포나 질의의 특성의 차는 너무나도 다르다. GIS 데이터의 처리에 있어 커다란 복잡한 객체는 작은 객체들로 분해가 되고 각 객체들은 다른 객체들에 대한 참조 정보를 가지고 있어 이것으로 공간정보를 형성한다. 그러나 VLSI 데이터는 경계를 공유하는 데이터의 패턴이 없기 때문에 이런 참조정보가 없다. 대부분의 공간색인구조는 질의 연산을 순차적인 방식으로 처리한다. 이러한 방식의 공간색인구조는 참고문헌 [2-5, 8, 9, 11, 14]에 기술되어있다. 병렬처리 공간색인구조의 연구는 비교적 최근에 시작된 분야라 하겠다.

MXR-트리는 최초의 병렬처리 공간색인구조이다[10]. MXR-트리는 R-트리계열의 구조로써 공간 객체가 그려진 본래의 공간을 공간 분할하는 native space indexing방식을 사용한다. MXR-트리는 R-트리의 노드를 여러 개의 디스크에 배치한다. 이를 위해 MXR-트리는 4가지의 heuristics(round robin, minimum area, minimum intersection, proximity index)를 제시하였다. 이들 가운데서 proximity index(PI)방식이 가장 좋은 성능을 가지고 있다. Proximity index의 목적은 새로이 만들어진 노드가 기존의 노드들과 함께 검색될 확률이 가장 적은 디스크에 새 노드를 배치하는 것이다. 이상적인 노드의 분배를 위해서는 같은 레벨에 있는 모든 형제 노드들을 proximity index에서 고려해야하지만 이를 위해서는 너무 많은 디스크 엑세스가 필요하므로 같은 부모 노드를 가진 형제 노드들만을 proximity index계산에서 고려한다. 따라서 proximity index heuristic은 경우에 따라서 노드 분배에 불균형을 가질 수 있다.

MXR-트리는 R-트리의 구조적인 특성을 그대로 갖고있기 때문에 R-트리의 불필요한 검색경로와 같은 문제를 갖고 있다. 병렬처리 공간색인구조의 질의 연산에서 D(노드배치에 사용된 디스크의 수)개의 프로세스가 만들어지며 각 프로세스는 동시에 할당된 디스크를 검색한다. MXR-트리의 구조는 하나의 트리를 가지며 모든 노드들은 D개의 디스크에 분배된다. 따라서 MXR-트리는 질의 연산동안에 D개의 프로세스 각각이 검색 또는 삭제 영역과 중첩하는 색인 엔트리를 찾을 때마다 프로세스간에 정보교환이 필요하다. 따라서 프로세스간에 정보교환에 필요한 시간은 사용되는 디스크의 숫자에 비례하여 증가한다. (그림 1a)는 주어진 10개의 공간 객체를 MXR-트리가 공간 분할하는 것을 보여준다. (그림 1b)는 (그림 1a)에 대한 MXR-트리의 구조를 보여준다(D_i는 해당 노드가 저장되는 디스크 번호를 나타내고, n_i는 노드에 대한 ID이다). (그림 1b)에서 보는바와 같이 노드 n1과 n5는 동일한 디스크 D2에 저장되었다. 그러나 이들 두 노드에 있는 객체들은 공간상에서 상당히 근접해있고, 따라서 동시에 질의 처리동안 접근될 확률이 높아 다른 디스크에 분리 저장되어야 하지만 MXR-트리의 구조는 부모 노드가 서로 다른 경우에 대한 근사계수(proximity index)는 고려되지 않는다.



(a)



(b)

(그림 1) (a)점선은 색인 사각형, 실선은 객체 사각형을 각각 나타낸다 ;
(b)는 (a)에 대한 MXR-트리 구조

PML-트리의 자세한 구조가 2장에서 소개가 되고, 3장에서는 PML-트리의 성능이 MXR-트리와 비교 분석이 된다. 4장에서 결론이 주어진다.

2. PML-트리

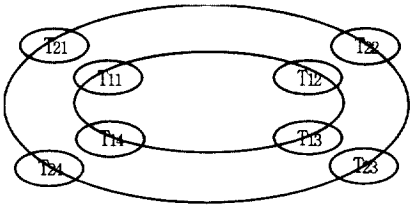
이 장에서는 병렬처리 공간 색인구조인 PML(Parallel Multi Layer)-트리의 설계가 소개된다. 병렬처리 공간색인구조에서 주어진 검색 영역과 중첩하는 엔트리가 서로 다른 디스크에 있는 경우 질의 처리동안에 동시에 엑세스된다. 효율적인 병렬처리 공간색인구조는 질의처리에 있어 전체 디스크 엑세스의 수가 적어야함은 물론 엑세스되는 노드가 디스크 상에 균일하게 분포되어 있어야 한다.

질의처리동안에 전체 디스크 엑세스의 수를 줄이기 위해서 R-트리[9], R^+-트리[15], R*-트리[6]의 단점을 향상시켜야한다. PML-트리는 공간 객체를 여러 개의 데이터 공간에 분배함으로써 색인 사각형들 사이의 중첩을 완전히 제거하고 말단 노드에 존재하는 중복된 엔트리들을 제거한다. 이를 위해, PML-트리는 native space indexing과 disjoint space decomposition방식을 사용한다. Disjoint space decomposition방식은 색인 사각형사이의 중첩을 R^+-트리와 같은 방식의 공간분할로 제거하여 불필요한 검색경로를 제거한다. 그러나 R^+-트리는 색인 사각형의 중첩을 제거하기 위해 색인 사각형에 완전히 포함되지 않는 공간 객체는 그 객체와 중첩하는 여러 말단 노드에 중복하여 저장하는 방식을 택한다. 이러한 중복된 객체 엔트리는 결국 노드의 수를 증가시켜 검색영역이 커지는 경우 성능에 치명적인 영향을 미치게 된다. 검색 영역이 아주 작은 경우 예를 들어 point 검색의 경우에 어떠한 공간 색인구조도 R^+-트리 보다 우수한 성능을 보일 수 없는 이유가 바로 여기에 있는 것이다. 또한 R^+-트리는 삭제 연산 시에 중복된 모든 객체 엔트리를 삭제하기 위해 다중의 삭제 과정을 거쳐야 한다. PML-트리의 상위 레벨의 색인 사각형은 모든 하위 레벨 사각형들을 완전히 포함한다. 이러한 특성을 유지하기 위해 PML-트리는 여러 개의 데이터 공간을 사용한다. 새로운 공간객체를 삽입하기 경우 존재하는 데이터 공간 중 새로운 객체를 완전히 포함할 수 있는 공간이 있다면 그 공간에 삽입이 되고, 만일 어떠한 데이터 공간의 색인 사각형도 객체를 완전히 포함할 수 없는 경우 새로운 데이터 공간이 생성되고 그 곳에 삽입이 된다. 객체의 삽입과정에서 노드의 분할(node split)이 발생하는 경우 해당 공간에 저장될 수 없는 객체는 그 공간으로부터 제거되어 재 삽입된다. PML-트리의 노드 분할에서 이러한 과정이 설명된다.

병렬처리 공간색인구조의 성능을 향상시키기 위해서 공간 객체를 분배하는 heuristic이 필요하다. 병렬처리 공간색인구조의 질의처리성능은 여러 개의 디스크를 동시에 엑세스할 때 가장 많은 디스크 엑세스를 갖는 디스크에 의해 결정된다. 효율적인 object distribution heuristic은 데이터 구조의 특성을 유지하면서 공간 객체를 여러 개의 데이터 공간에 균일하게 분배함으로써 질의 반응시간을 줄여야한다. PML-트리는 두 가지의 object distribution heuristic을 사용을 한다. PML-트리의 노드 분리 알고리즘(split algorithm)과 object distribution heuristic은 또한 높은 공간 활용도를 유지한다. 본 논문에서 사용되는 병렬처리 공간색인구조들은 여러 개의 프로세서와 여러 개의 디스크가 사용되는 하드웨어 구조를 가지고 있다.

2.1 PML-트리의 구조

PML-트리는 multiple-layer(계층)와 multiple-tree(트리)의 구조를 가지고 있다. 각 계층은 여러 개의 트리들로 구성되어 있고, 트리의 개수는 사용되는 디스크의 수와 같다. 각 트리는 height-balanced 트리이며 같은 계층에 있는 트리들은 같은 height를 갖는다. 각 데이터 공간은 하나의 디스크 상에 위치하는 독립된 하나의 트리와 연관되어있다. 하나의 디스크는 하나이상의 트리를 가질 수 있다. PML-트리에서는 각 트리들이 독립적인 구조를 가지고 있기 때문에 질의 연산동안에 프로세스간의 정보교환이 불필요하다. 따라서 모든 디스크의 완전한 병렬처리가 가능하다. 그림2는 2개의 계층과 4개의 트리를 가진 PML-트리의 layout을 보여준다.

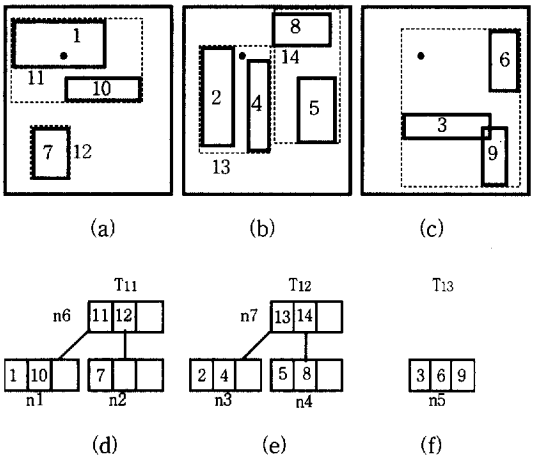


(그림 2) 2-계층 4-트리로 구성된 PML-트리의 구조

각 트리들은 트리 ID인 T_{ij}(i는 계층의 번호이고 j는 트리의 번호를 나타낸다. 트리번호는 디스크의 번호이기도 하다)에 의해 식별될 수 있다. 디스크 j는 트리 ID T_{ij} (i = 1, 2, 3, ···, L, L은 계층의 수)를 가진 트리들을 포함한다. 예를 들어, (그림 2)에서, 디스크 1은 트리 T_{11}과 T_{21}을 포함한다. 따라서 PML-트리의 질의 반응시간은 디스크 엑세스의 최대 수에 비례한다. 만일 P_{ij}가 트리 T_{ij}안에 있는 노드들을 처리하는데 걸리는 시간이라면 PML-트리의 질의 반응시간(R)은 :

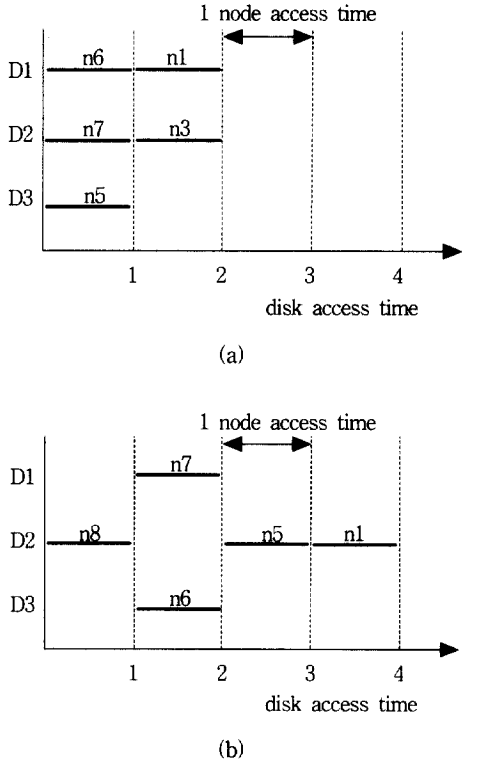
$$R = \max_{j=1\cdots D}\left(\sum_{i=1}^{L} P_{ij}\right)$$

D는 디스크의 수를, L은 계층의 수를 나타낸다.



(그림 3) 점선은 색인 사각형, 실선은 객체 사각형을 각각 나타낸다 : (a)첫 번째 데이터공간 (b) 두 번째 데이터 공간 (c)세 번째 데이터 공간 ; (d),(e)(f)는 (a),(b),(c)에 대한 PML-트리 구조

(그림 3)는 2차원 공간 객체들에 대한 PML-트리의 객체 분배의 예를 보여준다. PML-트리는 R-트리계열의 노드구조를 가진다. 즉 모든 객체는 말단 노드에 저장이 되고, 색인 노드는 색인 사각형들을 나타내는 엔트리들로 구성이 된다. 그림3에서 PML-트리는 객체 사각형 1에서 10까지를 3개의 데이터 공간에 간단한 heuristic인 minimum entry number(엔트리의 수가 최소인 트리의 선택)를 사용하여 분배한 것이다. (그림 3a), (그림 3b), (그림 3c)에서 모든 데이터 공간은 중첩되지 않은 색인 사각형들을 가지고있고 모든 사각형들은 상위 레벨 사각형들에 의해 완전히 포함된다. (그림 3d), (그림 3e), (그림 3f)는 3개의 데이터 공간에 상응하는 트리를 보여준다. (그림 3d), (그림 3e), (그림 3f)에서 트리 T_{ij}에 있는 모든 노드들은 디스크 j에 저장이 된다. PML-트리의 말단 레벨에는 중복된 엔트리가 존재하지 않는다. (그림 3)에서 PML-트리는 1-계층 3-트리의 구조를 갖는다.



(그림 4) MXR-트리와 PML-트리의 병렬처리 디스크 엑세스 시간

(그림 4a)와 (그림 4b)는 (그림 1)과 (그림 3)의 예에서 주어진 point(점) 질의를 처리하기 위한 MXR-트리와 PML-트리의 디스크 엑세스 시간을 각각 보여준다. 하나의 노드를 엑세스하는데 걸리는 디스크 엑세스 시간이 상수이고 한 노드의 크기가 1 페이지라 가정한다. 그림3에서 오직 공간 객체 1만이 주어진 검색점과 중첩한다. (그림 4b)에서 주어진 점 질의를 처리하기 위한 PML-트리의 디스크 엑세스의 수는 5이고, 이 5개의 노드를 병렬로 엑세스하는데는 단위시간 2가 걸린다. PML-트리에서 각 프로세스는 하나의 디스크

와 연계되어, 오직 주어진 검색 영역과 중첩하는 노드들만 엑세스하고 프로세스간의 정보교환이 없다. 그러나 MXR-트리는 주어진 point 질의처리에 단위시간 4가 걸리는 것을 볼 수 있다. PML-트리의 알고리즘들은 다음 장에서 주어진다.

2.2 PML-트리의 알고리즘

이장에서 주어지는 알고리즘들은 2차원 공간 객체에 기준을 둔 것이나, 2차원 이상의 공간 객체 처리를 위해 쉽게 확장될 수 있다.

2.2.1 PML-트리의 삽입

새로운 엔트리(NEW)를 삽입하기 위해 PML-트리의 삽입 알고리즘은 object distribution heuristic을 사용하여 하나의 트리를 선택을 한다. Object distribution heuristic은 새로운 엔트리를 받아들일 수 있으면서 PML-트리의 특성을 만족시키는 하나의 트리를 선택한다.

- Absolute Crowd Index

공간 객체는 임의의 크기와 모양을 갖는다. 또한 이러한 객체들은 서로 중첩될 수 있고 공간상에 임의의 위치에 분포한다. 이는 대부분의 응용분야에서 데이터 공간의 어느 한 부분이 다른 부분에 비해 공간 객체의 밀도가 아주 높거나 낮을 수 있음을 의미한다. 질의처리에서는 D개의 프로세스가 만들어져 각 프로세스는 할당된 디스크 안에 저장된 트리들을 주어진 검색 영역에 대해 검색한다. 따라서 좋은 성능을 갖기 위해서는 PML-트리의 같은 계층에 있는 트리들은 거의 엇비슷한 수의 노드를 각각의 검색경로에 가져야한다. (그림 3)의 예에서 사용했던 minimum entry number에 의한 공간 객체의 분배는 객체의 공간상의 위치를 고려한 것이 아니기 때문에 각 트리의 검색 영역 안의 노드들의 숫자가 트리들 사이에 불균형을 이룰 수 있다. 삽입 경로를 따라 노드의 분리 가능성이 큰 트리는 보다 많은 노드(색인 노드와 말단 노드)를 만들려는 경향을 가지고 있다. 결과적으로, 높은 노드 분리의 가능성을 가졌던 삽입 경로에 해당하는 영역에 대한 검색 연산에서는 보다 많은 노드가 엑세스되어야한다. Absolute crowd index(AC)에 의한 객체 분배는 새로운 엔트리인 "NEW"를 삽입하기 위한 삽입경로를 따라서 최소한의 노드 분리 가능성을 가진 하나의 트리를 찾는 것이다. AC distribution heuristic은 노드 분리의 가능성을 결정하기 위해 삽입경로를 따라서 노드의 crowdness(복잡도)를 측정을 한다.

$$\sum_{j=1}^{h}(W \times j \times N_j \rightarrow EntryNum)$$

h는 트리의 오더, W는 가중치 상수,
N_j는 삽입경로의 오더가 j인 노드,
EntryNum은 노드 N_j안의 엔트리의 수이다.

가중치 상수인 W(>1)는 높은 오더의 노드에 보다 많은 가중치를 주기 위해서 사용이 된다. 오더가 높은 노드는 엔트리 NEW의 삽입에 의해 보다 쉽게 분리하려는 경향이 있다. 왜냐하면, 새로이 삽입되는 엔트리 NEW는 말단 노드에 삽입되고 말단 노드는 가장 높은 오더를 가지고 있기 때문이다. 가장 낮은 AC index값을 가진 트리가 엔트리 NEW의 삽입을 위해서 선택된다. 만일 말단 노드가 NEW를 받아들일 수 있다면 트리의 상태는 말단 노드내의 엔트리의 수에 의해 A(accept)또는 S(accept and split)로 설정된다. 상태가 A 또는 S인 트리들에 대해서 알고리즘은 4가지의 범주(트리 오더, index 값, 트리 상태, 트리에 속한 엔트리의 수)를 사용을 해서 하나의 트리를 선택을 한다. 이들 4개의 criteria의 적용 우선 순위는 :

1. 트리 오더- 최소 오더를 가진 트리를 선택
2. index 값- 최소 AC index값을 가진 트리를 선택
3. 트리 상태-상태 A를 가진 트리를 상태 S를 가진 트리에 우선해서 선택
4. 트리에 속한 엔트리의 수-최소 숫자의 엔트리를 가진 트리를 선택

만일 현재의 계층에 어떤 트리도 새로운 엔트리인 NEW를 받아들일 수 없다면, 현재의 계층을 하위의 계층로 설정을 하고 선택 프로세스를 반복한다. 만일 하위의 계층이 존재하지 않는다면 새로운 계층이 만들어진다.

- Relative Crowd Index

Relative crowd(RC) index는 새로운 엔트리를 삽입하기 위한 트리의 삽입경로에 있는 노드들의 relative crowdness를 사용한다. RC index distribution heuristic은 삽입경로 상에 있는 색인 사각형들의 크기 대한 엔트리의 상대밀도가 낮은 트리를 선택한다. 질의 연산에서 좋은 성능을 갖기 위해서는 PML-트리는 같은 계층에 있는 트리의 삽입경로에 있는 노드들의 숫자가

거의 유사해야만 한다. 즉, 각 데이터 공간에서 주어진 검색 영역과 중첩하는 색인 사각형의 숫자가 거의 유사해야한다. 왜냐하면, PML-트리에서는 중첩되지 않은 공간 분할방식이 사용이 되고 하나의 노드는 데이터 공간에 있는 색인 사각형에 의해 표현되기 때문이다. 따라서, 각 데이터 공간의 유사한 영역에 있는 색인 사각형의 숫자가 거의 유사해야한다. 이러한 조건을 만족시키기 위해서 RC index distribution heuristic은 삽입경로를 따라서 엑세스된 노드(단 root 노드는 제외한다)의 엔트리당 평균 영역크기를 계산하고 그 값들을 합한다. 값이 크면 클수록 그 영역의 엔트리 밀도는 낮은 것이다. RC index distribution heuristic은 가장 낮은 엔트리 밀도를 가진 트리를 선택한다. RC index 값은 다음과 같이 계산될 수 있다 :

$$\sum_{j=2}^{h}\left(\frac{M_j \times W^{(j-2)}}{N_j \rightarrow EntryNum}\right)$$

h는 트리의 오더, W는 가중치 상수, N_j는 삽입경로의 오더가 j인 노드, M_j는 N_j에 대한 minimum bounding 사각형의 크기를 나타낸다.

높은 오더를 가진 노드를 나타내는 사각형들은 낮은 오더를 가진 노드를 나타내는 사각형에 의해 완전히 포함이 되므로, 가중치 상수인 W(>1)는 삽입경로에 있는 노드를 나타내는 색인 사각형들의 size를 normalize하기 위해 사용이 된다. 만일 말단 노드가 NEW를 받아들일 수 있는 상태라면 트리의 상태는 말단 노드내의 엔트리의 수에 의해 A(accept)또는 S(accept and split)로 설정된다. 상태가 A 또는 S인 트리들에 대해서 알고리즘은 4가지의 criteria(트리 오더, 트리 상태, index 값, 트리에 속한 엔트리의 수)를 사용을 해서 하나의 트리를 선택을 한다. 이들 4개의 criteria의 적용 우선 순위는 :

1. 트리 오더 - 최소 오더를 가진 트리를 선택
2. 트리 상태 - 상태 A를 가진 트리를 상태 S를 가진 트리에 우선해서 선택
3. index 값 - 최대 RC index값을 가진 트리를 선택
4. 트리에 속한 엔트리의 수 - 최소 숫자의 엔트리를 가진 트리를 선택

RC index distribution heuristic에서 트리 오더가 index값의 적용보다 우선한다. Index값이 트리 오더보다 우선 적용이 되었을 때, PML-트리의 질의 성능은 향상이 되나 공간활용도가 낮아지는 단점이 생긴다. 따라서 PML-트리의 RC index distribution heuristic은 공간활용도를 높이기 위해 다소간의 질의 성능의 감소가 있어도 트리 오더를 index값 보다 우선해서 적용한다. 만일 현재의 계층에 어떤 트리도 새로운 엔트리인 NEW를 받아들일 수 없다면, 현재의 계층을 하위의 계층로 설정을 하고 선택 프로세스를 반복한다. 만일 하위의 계층이 존재하지 않는다면 새로운 계층이 만들어진다.

Object distribution heuristic중의 하나를 사용해서 새로운 엔트리를 삽입할 하나의 트리를 선택한 후에 PML-트리의 삽입 알고리즘은 선택된 트리를 root 노드에서 말단 노드까지 삽입경로를 따라 엑세스한다. 새로운 엔트리를 말단 노드에 삽입한 뒤에 알고리즘은 삽입경로에 있는 노드들을 나타내는 사각형들의 minimum bounding 사각형들을 계산하여 update한다. 만일 삽입경로에 있는 노드가 넘친다면(노드의 용량 이상의 엔트리가 삽입될 때) 노드 분리 알고리즘이 (용량+1) 개의 엔트리들을 2개의 노드(본래의 노드 N과 새로이 만들어진 분리 노드 SP)로 재분배한다. 만일 공간 객체(말단 레벨 엔트리)가 삽입 과정에서 노드 분리에 의해 제거가 된다면 이들 엔트리는 재 삽입된다.

2.2.2 PML-트리의 노드 분리

만일 삽입 연산동안에 노드가 넘치는 경우 노드의 분리가 필요하다. PML-트리의 노드 분리에서 고려하는 parameter와 그들의 적용 우선 순위는 다음과 같다 :

(1) 노드사이의 엔트리개수의 균형,
(2) 분리 선 위에 걸치는 사각형수의 최소화,
(3) 분리 이후 두 색인 사각형들의 면적합의 최소화.

각 차원에 대해, 용량초과 노드내의 엔트리들을 나타내는 사각형들의 시작과 끝 좌표 값을 하나의 배열 Split_Pos[]에 저장을 하고 정렬한다. 분리 알고리즘은 노드내의 엔트리의 개수가 capacity(CAP)를 초과할 때 불려지므로 노드 N내의 사각형들의 시작과 끝 좌표 값의 개수는 총 2(CAP+1)이다. Split_Pos[j], j = 1, 2, 3, …, 2(CAP + 1), 안에 있는 각 좌표에 대해, 분리 알고리즘은 용량초과 노드내의 엔트리를 나타내는 사각형

의 주어진 차원에 대한 시작과 끝의 두 좌표 값이 모두 Split_Pos[j]보다 작거나 같은 사각형의 숫자를 계산해서 배열 LO_count[j]에 저장한다. 또한 Split_Pos[j], j = 1, 2, 3, ⋯, 2(CAP+1), 안에 있는 각 좌표에 대해, 용량초과 노드내의 엔트리를 나타내는 사각형의 주어진 차원에 대한 시작과 끝의 두 좌표 값이 모두 Split_Pos[j]보다 크거나 같은 사각형의 숫자를 계산해서 배열 UP_count[j]에 저장한다. Split_Pos[j], j = 1, 2, 3, ⋯, 2(CAP+1), 안에 있는 각 좌표에 대해 노드내의 엔트리를 나타내는 사각형의 주어진 차원에 대한 좌표 값이 Split_Pos[j]와 중첩하는 사각형의 숫자를 계산해서 배열 OV_count[j]에 저장한다. LO_count[j], UP_count[j], OV_count[j]의 합은 CAP+1이다. 따라서 OV_count[j]의 값은 (CAP+1) - (LO_count[j] + UP_count[j])에 의해 얻어질 수 있다. 각 LO_count[j]와 UP_count[j] (j = 1, 2, 3, ⋯, 2(CAP+1))에 대해 둘 중에 적은 수를 택하여 배열 Temp[j]에 저장한다. Balancing factor m은 분리 이후 두 개의 노드 N과 SP사이에 엔트리 수에 있어 균형을 주기 위해 사용이 된다. 만일 어떤 Temp [j]도 m보다 크거나 같은 갖지 않는다면, m값은 MAX (Temp[j], j= 1, 2, 3, ⋯, 2(CAP+1))로 설정된다. 모든 j에 대하여 Temp[j]의 값이 m보다 크거나 같은 것들 중에서 분리 알고리즘은 최소의 OV_count[j]의 값을 가진 하나의 j를 선택한다. 만일 최소의 OV_count[j]의 값이 하나 이상 존재한다면 분리 position인 Split_Pos[j]에 대해 분리 한 뒤에 노드 N과 SP를 나타내는 색인 사각형의 면적 합이 최소인 j를 선택을 한다. 분리 알고리즘은 balancing factor m이 0이 아닌 이상, 각 차원에 대해서 하나의 분리 position과 그 분리 position에 대해 중첩하는 엔트리의 개수를 얻는다. 만일 balancing factor m이 0이라면, PML-트리의 노드 용량이 증가되어야 한다. Disjoint space decomposition method를 사용하는 공간색인구조의 노드 용량은 최대 엔트리 중첩보다 커야한다. 그렇지 않은 경우 노드의 분리가 불가능하다.

분리 알고리즘은 균형 있는 엔트리 분배를 위해 가장 큰 balancing factor m을 가진 차원을 선택한다. 만일 balancing factorm이 같다면 분리 position에서의 엔트리의 중첩이 최소인 차원을 선택한다. 엔트리의 중첩 역시도 동일한 것이 있다면 분리 이후 두 그룹에 대한 면적을 비교해 최소 값을 주는 차원이 선택된다. 하나의 차원과 분리 position을 선택한 후, 분리 알고리즘은 CAP+1개의 엔트리를 두 개의 노드 N, SP로 분배한다. 분리선과 중첩하는 말단 노드 엔트리는 제거되어 새로이 삽입 알고리즘에 의해 삽입된다. 분리선과 중첩하는 색인 레벨 엔트리들은 선택된 분리 차원과 분리선에 대해 재귀적으로 sub-divide된다. 분리 position보다 좌표 값이 작거나 같은 엔트리들은 노드 N에 저장이 되고, 좌표 값이 크거나 같은 엔트리들은 노드 SP에 저장이 된다.

2.2.3 PML-트리의 검색

PML-트리의 검색 연산은 D(하나의 계층에 있는 트리의 개수 또는 PML-트리에서 사용하는 디스크의 수) 개의 프로세스를 만든다. 각 프로세스 i(i = 1, 2, 3,⋯, D)는 계층에 있는 i번째 트리를 순회하면서 주어진 검색 영역과 중첩하는 모든 공간 객체를 검색한다. 각 프로세스는 서로 독립적으로 질의를 처리한다. PML-트리의 한 트리안에 있는 모든 노드들은 하나의 디스크에 있으므로, 프로세스간의 교신이 전혀 필요치 않다.

2.2.4 PML-트리의 삭제

검색 연산에서와 같이 PML-트리의 삭제 연산은 D개의 프로세스를 만들고, 각 프로세스 i(i = 1, 2, 3, ⋯, D)는 계층에 있는 i번째 트리를 순회하면서 주어진 삭제 영역과 중첩하는 모든 공간 객체를 삭제한다. 만일 말단 노드에서 엔트리가 삭제되었다면 삭제 검색 경로에 있는 모든 엔트리들이 나타내는 색인 사각형들을 조정해야한다. 만일 노드가 삭제의 결과 엔트리의 수가 0이 되면 그 노드는 트리에서 삭제가 된다. 만일 root 노드의 엔트리의 수가 0이 되면 그 트리는 PML-트리의 현재의 계층에서 삭제가 된다. 검색 연산에서와 같이, 프로세스간의 교신이 전혀 필요치 않다. 삭제 연산에서 위에서 언급된 것과 같이 노드가 완전히 빌 때까지 노드를 트리에 존속시키는 방법과 노드내의 엔트리가 일정한 개수 이하로 줄어들면 그 노드를 트리에서 삭제를 하고 노드에 남은 모든 엔트리들을 재 삽입시키는 두 가지 방법이 있다. 공간색인구조의 응용분야의 특성에 따라 적절한 방법이 사용되어야 한다. PML-트리 에서는 모든 사각형들이 상위 레벨 사각형들에 의해 완전히 포함되기 때문에 R^+-트리 에서와 같이 다중의 삭제 경로를 택할 필요가 없다.

3. 성능 비교

이장에서는 병렬처리 공간색인구조인 PML-트리와

MXR-트리의 성능이 다양한 테스트 데이터들을 사용하여 비교된다. 영역 질의를 처리하기 위한 처리시간이 비교되고 각 색인구조들의 공간 활용도 역시 비교가 된다.

3.1 공간색인구조의 구현

PML-트리는 본 논문에서 제시한 두 가지 object distribution heuristic에 대해 주어진 알고리즘에 따라 구현된다. Object distribution heuristic AC와 RC를 사용해 구현된 PML-트리를 각각 PAC-트리와 PRC-트리라 한다. MXR-트리의 object distribution heuristic 중에 proximity index heuristic이 가장 우수한 성능을 가진 것으로 알려졌으므로 proximity index heuristic에 기초하여 [10]에서 주어진 알고리즘에 따라 구현되었다. MXR-트리의 기본 structure는 R-트리의 유사하다. 따라서 R-트리의 분리 알고리즘가운데 가장 우수한 성능을 가진 것으로 알려진 quadratic 분리 알고리즘이 MXR-트리의 분리 알고리즘으로 사용이 되었다. MXR-트리의 minimum node fill factor는 노드분리 성능을 극대화하기 위해 노드 용량의 40%가 사용이 된다. MXR-트리의 검색과 삭제 연산에서는 프로세스간의 교신을 위해 큐(queue)가 사용되는 디스크의 수만큼 필요하다. 이들 큐를 사용해 프로세스는 엑세스할 노드의 포인터를 일시적으로 저장하는데, 한 번에 하나의 프로세스만이 하나의 큐에 포인터를 삽입할 수 있도록 하기 위해 semaphore 연산이 사용이 된다. 모든 큐는 반드시 공유기억공간을 사용해야만 한다. MXR-트리의 모든 엔트리와 노드는 디스크 ID가 필요하고 이는 MXR-트리의 공간활용도를 낮추는 하나의 요인이 된다.

3.2 테스트 데이터

4개의 테스트 데이터(VLSI 데이터, Tiger/Line™ 데이터, randomly distributed 데이터, uniformly distributed 데이터)를 각각 V, T, R, U라 한다. 4개의 테스트 데이터는 <표 1>에서 4가지의 데이터 특성으로 표시된다. 객체의 밀도는 아래와 같이 정의된다 :

$$D = \frac{data\ object\text{의 면적 합}}{dataspace\text{의 면적}} \times 100$$

테스트 데이터 R은 2차원 공간 객체를 가상의 데이터 공간에 난수발생기를 사용해 만들어진 좌표위치에 배치한 것이다. 테스트 데이터 U는 6가지의 크기의 정사각형을 가상의 2차원 공간상에 일정한 거리를 두고 배치한 것이다. 테스트 데이터 V 는 VLSI layout 시스템(Magic)에 의해 만들어진 것이다. VLSI design layout은 여러 개의 마스크 계층상에 위치하는 트랜지스터와 그것들을 연결하는 시그널와이어를 나타내는 많은 수의 객체들로 구성이 되어있다. 대부분의 design layout들은 manhattan 또는 rectilinear geometry를 사용하므로, 이들 geometry들은 2차원 축에 평행한 사각형들로 표현이 된다. 테스트 데이터 T는 Tiger 데이터베이스 시스템의 Tiger/file에서 얻어진 것이다. Tiger database 시스템은 미국 Census Bureau map[7] 상의 점, 선, 면을 나타내는 정보를 제공한다. 이 테스트 데이터의 평균 밀도는 약 0.25로 대단히 낮지만 특정한 지역의 밀도는 60을 초과하는 고 밀도이다. VLSI 와 Tiger 응용에서 사용되는 질의들은 거의 모두가 영역 질의의 변형이므로 본 논문에서는 공간색인구조의 성능을 측정하기 위해서 영역 질의를 사용한다.

〈표 1〉 4개의 테스트 데이터의 특성

데이터	객체수	밀도	객체 크기비율	데이터 공간
V	4085	0.34	0.0002 - 0.4525%	1320 x 1768
T	41058	0.25	0.0000 - 0.3479%	533552 x 349200
R	30000	4.35	0.0025 - 0.140%	3000 x 3000
U	30000	5.38	0.0011 - 0.040%	3000 x 3000

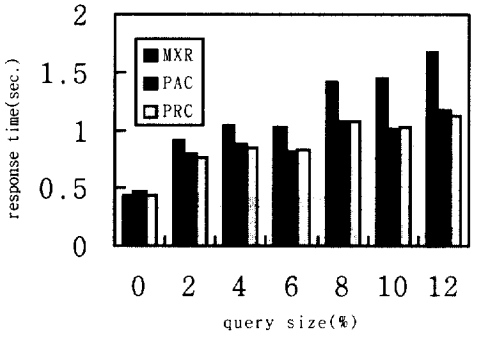
3.3 테스트 결과

모든 프로그램은 24개의 프로세서를 사용하는 Sequent Symmetry S81상에서 실행되었다. 이 실험에서 페이지의 크기는 1Kbyte이고 4개의 디스크가 병렬처리를 위해서 사용이 되었다. 질의 사각형은 데이터 공간상에서 영역 질의를 처리하기 위해 검색할 영역을 나타낸다. 질의 사각형의 범위는 데이터 공간의 0%~12%(0%는 한 점을 나타냄)이며 각 검색 범위에 대해 500개의 질의 사각형이 만들어지고, 검색된 후, 평균 반응시간이 얻어진다.

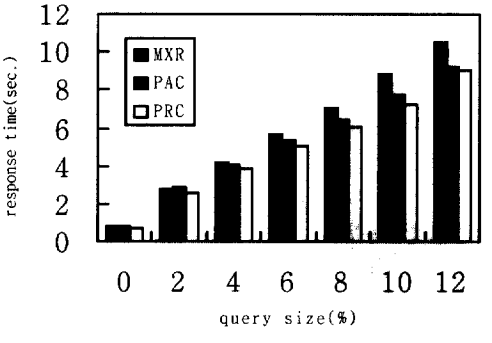
(그림 5a), (그림 5b), (그림 5c), (그림 5d)는 3개의 공간색인구조의 4가지 테스트 데이터에서의 평균 반응시간을 보여준다. 테스트 데이터 R의 아주 작은 크기의 영역 질의(예 : 크기 0인 점 질의)인 경우를 제외하고 대부분의 경우에서 PAC-트리와 PRC-트리는 MXR-트리보다 빠른 질의 반응시간을 보여준다. 이는 MXR-트리의 R-트리에서 계승한 구조적인 문제(다중 검색

경로)와 검색연산 시의 프로세스사이의 교신에 기인한다고 하겠다. (그림 6)은 3개의 공간색인구조에 의해 사용되는 메모리 활용도를 보여준다. PAC-트리가 가장 적은 메모리를 사용하고 MXR-트리가 가장 많은 메모리를 사용한다. 공간활용도는 아래의 식에 의해 구해질 수 있다 :

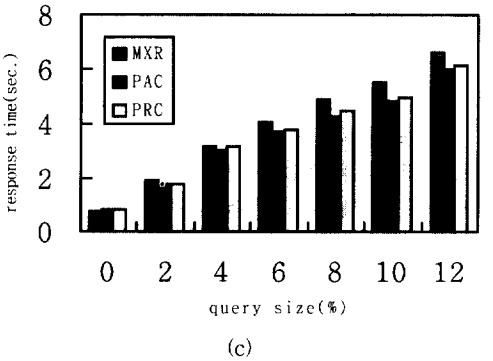
$$\text{공간활용도}(\%) = \frac{\textit{unique entry}\text{의 수}}{\text{전체 }\textit{node}\text{의 수} \times \textit{CAP}} \times 100$$



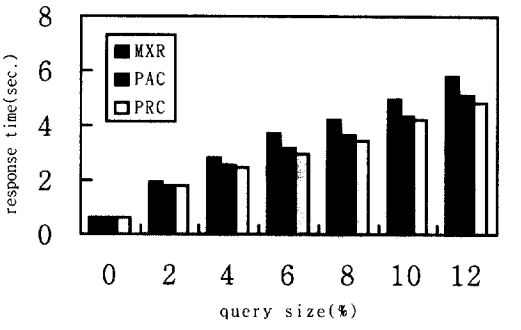
(a)



(b)

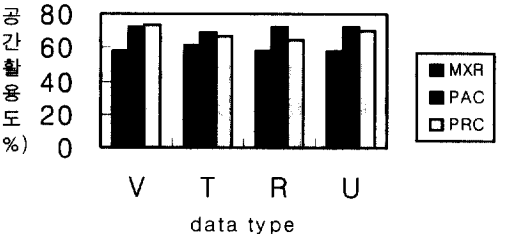


(c)



(d)

(그림 5) 4가지 테스트 데이터 (a)V, (b)T, (c)R, (d)U에 대한 PAC-트리, PRC-트리, MXR-트리의 평균 질의 반응시간 대 질의 크기



(그림 6) 각 트리의 공간활용도

4. 결 론

본 논문에서 새로운 병렬처리 공간색인구조인 PML-트리가 소개되었다. PML-트리는 여러 개의 디스크를 사용하여 질의 연산에 따르는 디스크 I/O를 병렬 처리함으로써 질의 처리를 향상시켰다. 공간 객체를 디스크 상에 균일하게 분배하기 위해서 PML-트리는 두 가지 object distribution heuristic과 함께 여러 연산 알고리즘을 제시하였다. PML-트리는 여러 개의 데이터 공간에 공간 객체를 disjoint space decomposition과 하위 레벨 사각형의 완전포함 특성을 유지하면서 분배함으로써 질의 처리속도를 향상시키고 말단 노드의 불필요한 엔트리의 중복을 제거하였다. 여러 테스트 데이터을 사용하여, PML-트리와 MXR-트리의 성능이 비교되었다. 대부분의 검색 영역에 대하여 PML-트리는 MXR-트리보다 우수한 질의 처리시간을 보여준다. 또한 PML-트리의 공간활용도에서도 MXR-트리보다 우수하며, 모든 테스트 데이터에 대해 매우 안정된 성능을 보인다. PML-트리는 공간 데이터베이스를 위한 효

율적인 공간색인구조로서 사용될 것으로 기대된다. 현재 삽입 연산을 가속화하기 위한 병렬삽입연산에 대한 연구가 진행중이다.

참 고 문 헌

[1] Banerjee, J. and Kim, W., Supporting VLSI geometry operation in a Database System, Proc. of the IEEE 2nd International Conf. on Data Engineering, pp.409-415, 1986.

[2] Bang, K. S. and Lu, Huizhu, A Simulation on an Index Structure for the Spatial object, Proc. of the International Simulation Technology Conf.(SIMTEC '92), November, pp.178-183, 1992.

[3] Bang, K. S. and Lu, Huizhu, An Application of the multi-R tree to the VLSI circuit layout design, Proc. 9th International Conf. on System Engineering, University of Nevada Las Vegas, pp.295-299, 1993.

[4] Bang, K. S. and Lu, Huizhu, SMR-tree : an efficient Index Structure for Spatial Databases, Proc. of the 1995 ACM Symposium on applied Computing, Nashville, February, pp.46-50, 1995.

[5] Bang, K. S. and Lu, Huizhu, An Efficient Index Structure for Spatial Databases, Journal of Database Management, Vol.7, No.3, Summer, pp.3-15, 1996.

[6] Beckmann, N and Kriegel, H. P., The R*-tree : An Efficient and Robust Access Method for Points and Rectangles, ACM SIGMOD, pp.322-331, 1990.

[7] Bureau of the Census, Tiger/Line File, 1992 Technical Documentation, Bureau of the Census, Washington, DC, 1993.

[8] Gunther, O., The Design of the Cell tree : An Object Oriented Index Structure for Geometric Databases, IEEE 5th International Conference on Data Engineering, pp. 598-605, 1989.

[9] Güttman, A., R-trees : A Dynamic Index Structure for Spatial Searching, Proc. of the ACM SIGMOD, pp.47-57, 1984.

[10] Kamel, I. and Faloutsos, C., Parallel R-tree, ACM SIGMOD, pp.195-204, 1992.

[11] Hoel, E. G., and Samet, H., A Qualitative Comparison tudy of Data Structures for Large Segment Databases, ACM SIGMOD, pp.205-214, 1992.

[12] Lomet, D. B., A Review of Recent Work on Multiple attribute Access Methods, ACM SIGMOD Record, Vol.21, No.3, pp.56-63, 1992.

[13] Oosterom, P. V. and Den, Bos, J. V., An Object-Oriented Approach to the Design of Geographic Information systems, Computers & Graphics, Vol. 13, No.4, pp.409-418, 1989.

[14] Osawa, Y. and Sakauchi, M., A New Tree Type Data Structure with Homogeneous Node Suitable for a Very Large Spatial Databases, Proc. of the IEEE 6th International Conference on Data Engineering, pp.296-303, 1990.

[15] Sellis, T., Roussopoulos, T. and Faloutsos, C., R^+-tree : A Dynamic Index for Multi-dimensional objects, Proc. of the 13th VLBD Conference, pp. 507-518, 1987.

[16] Soffer, A. and Samet. H., Pictorial query by image similarity, IEEE Proceedings of the 13th International conf. on Pattern Recognition, pp.114-119, 1996

방 갑 산

e-mail : ksbang@ice.hansung.ac.kr

1987년 중앙대학교 화학과 졸업 (학사)

1992년 (미) Oklahoma State University Computer Science (석사)

1995년 (미) Oklahoma State University Computer Science (박사)

1996년~1997년 삼성SDS 선임연구원

1997년~현재 한성대학교 정보시스템공학과 조교수

관심분야 : 공간자료구조, 공간데이터베이스, 지리정보시스템, 정보시각화