불균등 데이타 분포에 적합한 트리 구조

김수현*, 김병곤*, 이재호**, 임해철*
*홍익대학교 컴퓨터공학과
**인천교육대학교 컴퓨터교육과

Efficient Tree structure for Ununiformed Data

Soo-Hyun Kim*, Byung-Gon Kim*, Jae-Ho Lee**, Hae-Chull Lim*
*Dept. of Computer Engineering, Hong Ik University,
**Dept. of Computer Education, Inchon National University of Education

요 약

공간 데이타를 표현하는 색인 구조에 관하여 여러 연구가 진행되어 왔다 그 중 R-트리 계열의 트리들은 최소 겹침과 최소 영역 증가를 기준으로 최소한계영역(Minimum Bounding Rectangle)을 생성하여 공간을 할당한다. 그러나 R-트리 기반의 트리들은 데이터 분포가 균등한 경우에서는 공간 할당이 적합하게 이루어지는데 반하여 데이타 분포가 어느 특정 영역에 밀집되어 있는 도메인의 경우에는 공간영역을 비효율적으로 할당하는 문제점이 있다. 본 논문에서는 불균등하고 비대칭적인 데이타 분포에 적합한 트리 구조를 제안하였다 제안된 트리 구조는 R*-트리를 기반으로 하였으며, 불균등하고 비대칭적인 데이타의 특징을 반영하여 단말 노드 구조를 설계하고 밀집영역에 대한 클러스터링을 반영하도록 삽입 및 삭제 루틴을 변형하였다

1. 서 론

데이타는 분포 특징에 따라 균등분포 데이타(uniform data)와 불균등 분포 데이타(ununiform data, skewed data)로 나눌 수 있다. 균등분포 데이타는 크기가 같거나 비슷한 데이타들이 전체 데이타 영역에 고르게 분포되어 있는 경우를 뜻하고, 불균등분포 데이타는 다양한 크기로 구성되어 있는 데이터들의 분포상태가 어느 한쪽으로 편중되어 있어 데이타가 밀집되어 있는 부분(dense region)과 희소 부분(sparse region)으로 이루어진 경우를 뜻한다

공간 데이타 인덱싱은 영역분할방법에 따라 R-트리 계열과 같이 영역 겹침을 허용하는 트리들과 K-D-B트리, 사분트리와 같이 영역겹침을 허용하지 않는 트리들로 구분된다

R-트리 계열의 인덱싱은 데이터 공간을 최소한개영역으로 묶어 줌으로써 B-트리와 같이 높이균형을 유지하는 트리이다. R-트리 계열의 노드분할은 최소한계영역의 최소증가와 영역간 겹침의 최소화를 기준으로 두 개의 새로운 영역을 동적으로 정의하여 공간영역을 분할한다 그러나 R-트리는 영역겹침을 허용하므로 색인영역의 중복으로 인해 검색속도를 저하시키는 단점이 있다 실세계 데이타와 같이 데이타의 분포가 불균등할수록 부적합 영역할당의 문제점이 나타난다. 부적합 영역할당이란 거리가 가까운 데이터들을 동일한 영역의 노드에 삽입하지 못하고 거리가 먼, 즉 상관도가 낮은 다른 노드에 삽입하여 영역할당이 부적절하게 이루어짐을 뜻한다 부적합 영역할당은 질의처리 시에 불필요한 노드의 검색을 증가시키므로 질의 처리 성능을 저하시킨다. 부적합 영역할당의 발생 원인은 공간이용도를 높이기 위해 노드 분할 시, 노드 당 최소엔트리 개수를 지정하고 있기 때문이다. 데이타 분포가 불균등할수록 밀집영역의 데이타들은 거리가 먼 최소영역의 데이타들과 묶이게 되어 부적합영역할당의 문제가 커진다.

본 논문에서는 데이타 분포도를 이용하여 밀집영역을 구분한 불균등 분포 데이타에 적합한 트리 구조를 제안하였다.

본 연구는 한국과학재단 특정기초연구과제(과제번호'98-0102-09-01-3)의 지원을 받았음

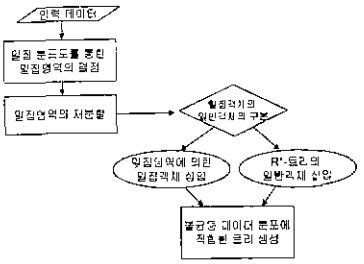
2. 관련 연구

클러스터링이 효율적으로 수행된 공간 인덱스는 공간상의 인접한 데이타들을 같은 노드에 삽입함으로써 데이타 검색 시, 디스크 접근 횟수를 줄일 수 있다. 그러나 R-트리 계열의 인덱싱 기법은 새로운 영역을 정의하거나 노드 분열을 위해 영역을 분할할 때 최소한계영역의 최소증가만을 기준으로 하기때문에 영역 겹침의 문제를 발생시킨다. 영역 겹침은 탐색영역을 증가시키므로 질의 시 방문하는 노드 수를 증가시킨다[1]. R*-트리는 영역 겹침의 문제를 해결하기 위해 최소한계 영역이 겹치지 않도록 중간노드에서의 노드간 겹침을 없애고 여러 영역에 걸쳐있는 객체들을 해당 단말노드에 저장하므로 데이타 중복의 문제가 있다[2] R*-트리의 영역할당은 영역 넓이의 최소화뿐만 아니라 영역둘레길이의 최소화와 겹침의 최소화를 고려하고, 노드 분열 시 강제 재삽입을 통해서 영역 할당을 재조정한다. 따라서 R-트리 보다는 클러스터링 효과가 높지만 불균등 데이타 분포에 있어서는 클러스터링의 효과가 떨어진다[3] Hilbert R-트리는 공간을 채우는 Hilbert 곡선을 사용하여 클러스터링 한다[4]. 그러나 Hilbert 곡선을 사용하여 클러스터링하는 것은 전체 데이타 공간이 커질수록 Hilbert 번수를 계산하는 오버헤드가 증가하는 문제점이 있다. 또 다른 문제점으로 공간이용도 측면에서 R-트리 계열의 공간 이용도는 70% 정도로 적은 편이다. 정적 데이타베이스 환경에서는 공간 이용도를 높이고자 R-트리를 재구성하는 R-트리 압축에 관한 연구가 진행되고 있다[5] 그 중에서도 Hilbert R-트리의 압축 기법이 클러스터링 성능과 공간이용도에 있어서 가장 우수하나, 기존의 연구는 불균등 분포 데이타를 반영하지 못했다 불균형 분포 데이타는 희소영역보다 밀집영역에 데이타들이 집중되어 있기 때문에 클러스터링의 중요성이 더욱 커진다. 따라서 데이타들을 효율적으로 클러스터링하여 영역질의, 최근접질의, 공간조인등을 수행할 때 방문하는 노드 수를 최소화할 수 있다

3. 밀집영역 정보를 포함하는 트리 구조

3.1 트리 생성 단계

본 연구에서는 불균등 데이타 분포 클러스터링에 적합한 트리를 구축하기 위하여 밀집 영역 정보를 이용한다 <그림1>은 트리 생성 과정을 단계별로 표현한 것이다 우선, 밀집영역에 대한 클러스터링을 수행하기 위해 입력 데이타의 전체 데이타 공간에서 밀집영역을 구별하는 과정이 필요하다 밀집영역을 결정한 후, 밀집영역내의 데이타 개수가 기준 이상이면 밀집영역을 재분할하여 각 영역별로 정해진 개수를 넘지 않도록 하부영역으로 나눈다 다음은 트리 생성을 위한 데이타 입력과정으로, 트리에 객체 입력 시, 밀집영역 내 객체의 삽입은 객체가 속한 하부영역을 단말노드에 엔트리로 삽입한다 밀집영역 이외의 객체 삽입은 R*-트리와 동일하다



<그림1> 트리 생성 단계도

3.2 밀집 영역 결정

데이타 전체 영역 중에서 밀집도가 높은 영역은 데이타 분포도를 이용하여 결정한다[6]. 데이타 분포도는 전체 영역을 동일한 면적의 셀로 일정하게 나누어, 각 셀 당 존재하는 데이타의 개수나 해당 데이타가 차지하는 면적을 히스토그램으로 나타낸 것이다.

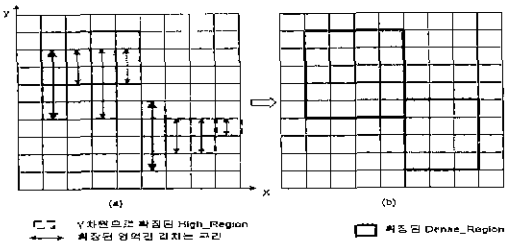
밀집영역은 셀 당 데이타 비율이 다른 셀보다 높은 셀들의 집합으로 정의한다 본 논문은 포인터 데이타와 단일 크기의 사각 데이터만을 고려하였으며, 다면체 데이타의 경우도 동일한 방식으로 확장할 수 있다

전체 데이타 개수가 N개, 전체 셀의 개수가 m개일 때, N = km (k>=1인 상수)으로 정의한다. 각 셀에 포함되는 데이타 개수는 nk(0<=n<=m)개로, n의 편차가 작을 때는 균등분포이고, 편차가 클 때에는 불균등분포라고 정의한다. 셀에 속하는 데이터의 개수가 mk/2개 이상이면 밀집도가 높은 셀(High_cell)이라고 정의하고, 밀집도가 높은 셀들의 모임을 밀집영역이라고 정의한다 밀집영역을 결정하는 방법은 밀집도가 높은 셀들의 확장을 통해서 이루어진다 다음은 밀집영역을 결정하는 과정이다.

```
1단계 : 한 차원을 기준으로 인접한 high_cell들을 확장한다.
        단, x차원으로 n개, y차원으로 m개의 셀이 있다.
        High_region은 한 축으로 확장된 인접한 밀집셀들의 집합
        No는 High_region의 인덱스로 전역변수이다.(No=0)
for (i=0, i<n, i++) // i는 x 차원의 인덱스
  for (j=0, j<m; j++) // j는  y 차원의 인덱스
    if ( cell[i][j]가 High_cell )
       High_Region(No)에 cell[i][j]를 포함
       if (cell[i][j+1]이 High_cell이 아니면)
         No = No + 1
       endif
    endif

2단계 : 인접한 High_Region에 대한 통합을 수행한다.
        High_Region()들을 y축에 사영(projection)시켜 얻은
        구간에 대한 정보를 projec()에 저장한다
        단, dNo은 projec의 인덱스이다 (dNo =0)
        Dense_Region은 조건을 만족하는 인접한 High_Region
        들의 집합
for( i=0; i<No; i++)
  High_Region(i)를 Dense_Region(dNo)에 포함시킨다
  if (projec[i]와 projec[i+1]이 1/3이상 겹치지 않으면)
    dNo = dNo + 1.
  endif
```

<그림2>의 회색 셀은 밀집도가 높은 셀을 나타낸다. <그림2-(a)>는 알고리즘의 1단계를 y축 방향으로 확장하여 수행한 후 생성된 8개의 High_Region들을 나타낸 것이다 <그림2-(b)>는 1단계에서 생성된 High_Region들에 대하여 2단계 알고리즘을 수행하여 생성한 최종적인 두 개의 Dense_Region들을 나타낸 것이다

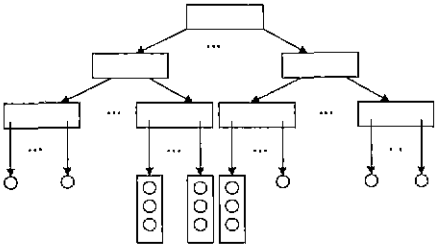


<그림2> 밀집영역 결정

3.3 트리 구조 및 생성

밀집영역내 객체들의 삽입과 밀집영역이외의 객체들 삽입을 구분하여 트리를 생성한다 데이타 분포도를 이용하여 결정된 밀집영역내의 객체들은 객체단위로 삽입하지 않고 객체가 포함된 영역단위로 삽입한다

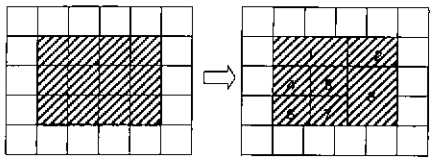
밀집영역이외의 객체들은 R*-트리의 객체 삽입과 동일하게 객체가 삽입될 단말노드를 찾아 단말노드의 영역정보에 객체의 최소한계영역을 지정하고, 단말노드의 포인터가 객체를 지시하도록 한다 그러나 밀집영역내의 객체들은 객체별로 삽입되는 것이 아니라, 객체가 속한 밀집영역의 영역 정보를 단말노드의 영역정보에 저장하고, 단말노드의 포인터가 밀집영역의 객체집합을 지시하도록 한다.



<그림3> 트리 구조

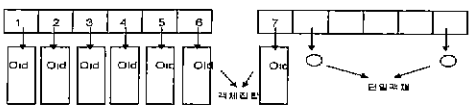
<그림3>은 R*-트리를 기반으로 불균등 데이타분포에 적합하도록 확장된 트리 구조이다. 그림에서 보듯이, 단말노드의 포인터는 일반객체를 가리키는 경우와 객체집합을 가리키는 경우로 나뉜다. 객체집합이란 밀집영역내의 객체들의 모임이다 이러한 구조는 R*-트리의 삽입방식을 따르면서도 밀집영역내의 객체들을 밀집영역으로 묶어서 삽입하므로, 밀집영역내의 객체들이 외부의 일반객체들과 묶여 다른 노드에 삽입되는 현상을 막아준다. 단말노드가 가리키는 객체집합의 개수는 제한되기 때문

에, 다음과 같이 밀집영역을 재분할하여 일정한 개수의 객체를 지니는 하부영역으로 나눈다



<그림4> 밀집영역내의 분할

<그림4>의 왼쪽 그림은 밀집도가 높은 셀들을 묶어 밀집영역을 결정한 것이다 밀집영역내의 모든 객체는 하나의 객체집합으로 포함될 수 없고, 밀집영역을 재분할 할 경우, 밀집 영역 내에서도 각 셀마다 데이타의 개수가 동일하지 않으므로, 데이타 비율에 따라 사분 트리의 영역 분할 방식으로 재분할한다. 재분할된 영역정보는 단밀 노드의 영역정보로 사용된다 <그림5>는 <그림4>의 오른쪽 그림의 하부 영역1, , 7의 영역정보를 단밀노드의 영역정보로 사용하고, 단말노드의 포인터는 각 영역에 속하는 객체 집합을 나타낸다.



<그림5> 단밀노드의 구조

3.4 삽입, 삭제 및 질의

객체의 삽입은 R*-트리와 동일하게 수행된다. 단, 삽입할 객체가 밀집영역내의 객체일 경우, 해당 단말노드를 찾아서 단말노드가 가리키는 객체집합에 객체정보만을 삽입한다 그러나 트리가 생성된 이후에 계속되는 추가 삽입에 의하여 새로운 밀집영역이 생겨날 수 있다. 이 때, 새로운 밀집영역의 생성을 파악하기 위해, 트리 생성 시의 생성된 분포도를 이용한다 트리 생성 이후 삽입된 객체의 수가 일정 개수에 도달할 때마다 새로운 밀집영역의 생성유무를 체크한다 새로운 밀집영역이 생성되면, 이전에 삽입된 밀집 영역에 해당하는 객체들을 트리에서 삭제한 후, 영역별로 묶어 영역에 의한 삽입으로 재 삽입하고, 단말노드의 포인터는 해당 객체집합을 가리킨다 기존 밀집영역에 객체 삽입이 일어나면, 객체집합이 오버플로우 되는 경우가 빈번한다. 이 때는 오버플로우 된 영역에 해당하는 객체와 영역정보를 트리에서 삭제한 후, 오버플로우 영역을 데이타 비율에 따라 재분할하고, 각 객체집합을 영역에 의해 재 삽입한다 마찬가지로 단말노드의 포인터는 해당 객체집합을 가리킨다

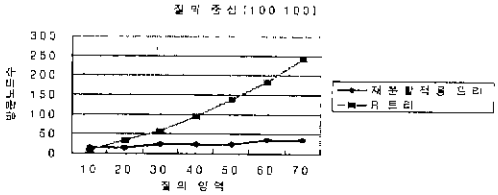
객체의 삭제 역시 R*-트리와 동일하게 수행한다 단, 밀집영역내의 객체 삭제는 객체가 속한 영역과 일치하는 단말노드를 찾아 단말노드가 가리키는 객체 집합 중에서 해당 객체만을 삭제하고, 단말노드의 영역정보는 그대로 유지한다. 그러나 동일한 밀집영역의 객체들의 삭제가 계속되어 밀집영역내의 객체가 존재하지 않으면 하부영역정보를 단말노드에서 삭제하고, 삭제된 영역정보의 변화를 루트 노드까지 긴파하여 트리를 재조정한다

질의의 수행은 R*-트리 기반의 트리들과 동일하게 수행하지만 단말노드의 포인터가 하나의 객체가 아닌 객체집합을 가리키는 경우 객체집합에서 선형탐색을 수행하여 질의조건에 만족하는 객체를 찾는다.

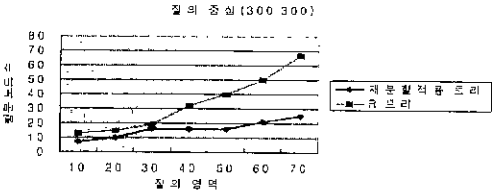
4. 성능평가

본 논문에서는 1~1000 범위의 2차원 데이타 영역에서, 단일 면적 '1'을 갖는 정사각형 객체 2000개 중, 2/5에 해당하는 800개의 객체가 1~300사이의 불균등분포영역에 집중되어 있는 데이터에 대하여 성능평가를 수행하였다

분포도를 통하여 밀집영역을 결정하고, 밀집영역을 144개의 하부영역으로 재분할한 후, 본 논문에서 제안한 트리와 R*-트리를 비교한 것이다 성능 평가 기준은 영역질의 수행시에 방문한 노드 수이다. <그림6>과 <그림7>은 질의 중심이 불균등분포 영역내 또는 경계에 해당되는 경우 제안한 트리가 기존 R*-트리보다 작은 수의 노드를 방문함을 보여준다.



<그림6> 질의 중심 (100, 100)인 영역질의



<그림7> 질의 중심 (300, 300)인 영역질의

5. 결 론

본 논문에서는 불균등 데이터 분포에 적합한 트리구조를 구축하기 위하여 전체 데이터 영역에 대한 밀집영역을 파악하여 데이터 삽입 시, 이를 고려하여 효율적인 질의처리가 가능하도록 하였다 밀집영역은 일정한 데이터 개수를 지니는 하부영역으로 세분하여, 밀집영역에 해당되지 않는 일반 객체와 구분하여 처리함으로써, 성능을 향상시켰다.

성능 평가에서는 질의 중심이 불균등분포 영역내 또는 경계에 해당되는 경우, 제안한 트리가 기존 R*-트리보다 적은 수의 노드를 방문함을 보였다

참고문헌

[1] A. Guttman, R-trees: a dynamic index structure for special searching, ACM SIGMOD, pp 47-57, 1984

[2] N Beckmann, H-P Kriegel, R Schneider, and B Seeger, The r*-tree: an efficient and robust access method for points and rectangles, ACM SIGMOD, pp 322-331, 1990.

[3] T Sellis, N. Roussopoulos, and Christos Faloutos, The R +-Tree A Dynamic Index for Multi-Dimensional Objects, VLDB, pp 507-518, 1987

[4] Ibrahim Kamel Christos Faloutsos, Hilbert R-Tree An Improved R-tree using Fractals, VLDB, pp.500-509, 1994

[5] Ibrahim Kamel and Christos Faloutsos, On Packing R-trees, Conference on Information and Knowledge Management (CIKM), 1993

[6] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Ken Sevik, and Torsten Suel, Optimal Histograms with Quality Guarantees, VLDB, 1998