클러스터링을 이용한 R-Trees 구축방법

차정숙, 이기준
부산대학교 전자계산학과

R-Trees construction using clustering

Jung-Sook Cha, Ki-Joune Li

Department of Computer Science, Pusan National University
e-mail : {chasook, lik}@quantos.cs.pusan.ac.kr

요약

공간 데이터베이스에서 사용되는 데이터는 그 양이 방대하고 복잡하여 이를 효율적으로 저장, 관리하는 색인이 필요하다. 여러 공간 색인 방법들 중에서 R-tree는 삽입과 삭제가 빈번히 발생하는 동적인 환경에서 효율적인 질의 성능을 보이는 것으로 알려져 있다. R-tree는 삽입되는 데이터의 순서에 따라 트리의 구조가 달라질 있는데, 주어진 데이터가 수정이 자주 발생하지 않는다면 데이터 입력 순서를 결정하여 질의 성능이 가장 좋은 트리를 구성할 수 있다. 본 논문에서는 데이터가 자주 수정 되지 않는 환경에서 노드간의 중첩을 가장 최소화 할 수 있는 데이터 입력 순서를 결정하기 위해 클러스터링을 이용한 새로운 방법인 CSR-tree를 제안하고자 한다. CSR-tree는 일반 R*-tree와 hilbert packed R-tree 방법보다 향상된 질의 성능을 보인다

1. 서론

공간 데이터베이스에서 사용되는 공간 데이터는 양이 방대하고 복잡하여 효율적으로 저장, 관리해 줄 수 있는 여러 가지 색인 방법이 제안되어져 왔다. 제안된 여러 방법들 중에서 R-tree와 그 트리의 변형 방법들은 삽입과 삭제가 빈번히 발생하는 동적인 상황에서 좋은 질의 성능을 보이는 것으로 알려져 있다[1, 5, 6, 7]. 만약 삽입, 삭제가 빈번히 발생하지 않는 정적인 데이터에 대해 트리를 구성하고자 한다면 동적인 데이터에 대해 구성되는 트리보다 향상된 질의 성능을 보이는 트리를 구성할 수 있다. 왜냐하면 R-tree 패밀리의 특성은 삽입되는 데이터의 순서에 따라 구성되는 트리의 구조가 달라지고 그 구조에 따라 질의 성능이 달라질 수 있기 때문이다. 여기서 좋은 질의 성능은 디스크 입력, 출력 횟수에 따라 좌우되는 데 이를 줄이는 방법은 기능한 공간적으로 근접해 있는 데이터들을 동일한 노드에 저장하여 디스크 접근 횟수를 줄이는 것에 의존한다.

본 논문에서는 주어진 데이터에 대한 전처리 작업을 하여 공간적으로 근접한 데이터들을 트리에 삽입된 후에도 근접하도록 하는 입력 순서를 찾아주는 방법인 CRS-tree (Clustering and Sorting-tree)를 소개한다. 근접한 데이터들을 찾아주는 방법으로는 공간적으로 가까이 있는 데이터들을 한 클러스터로 묶어주는 클러스터링을 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 실험에서 사용되어지는 R-trees에 대해 간단히 살펴보고 주어진 데이터에 대해 공간 근접성을 반영한 순서로 R-tree를 구성하는 방법에 대해 알아본다. 3장에서는 논문에서 제안하고 있는 클러스터링을 통해서 공간적으로 근접해 있는 데이터들을 분류하여 각각에 대해 정렬하여 R-tree에 삽입하는 방법에 대해서 살펴본다. 그리고 4장에서는 실제 실험에서 제안된 방법의 성능을 평가하고 마지막으로 5장에서는 결론과 향후 연구과제에 대해서 설명한다.

2. 관련 연구

지금까지 대량의 공간 데이터를 효율적으로 저장, 관리하기 위한 여러 가지 공간 색인 방법들이 제안되어졌다. 이러한 방법들은 크게 두 가지 클래스로 분류되어 질 수 있다. 첫번째는 객체의 차원을 변환하는 방법이다. k차원의 객체를 2k차원의 한 점으로 변환하거나 k차원상의 한 점으로 변환하는 방법이다. 1차원으로 변환하기 위해서 Z-곡선, 힐버트 곡선과 같은 공간 포화곡선을 이용하여 변환한다. 예를 들어 2차원의 사각형을 4차원의 한 점으로 변환하여 그 값에 대해 색인한다. 두 번째는 실제 객체에 대한 근사를 구해서 색인하는 방법이다. 실제 데이터에 대한 근사는 사각형, 오각형, 타원등과 같은 객체로 표현이 가능하다. R-tree와 변형 방법들이 이 부류의 대표 색인 방법이며 점, 선, 면과 같은 임의의 객체를 가장 최소로 둘러싸는 최소경계사각형(Minimum Bounding Rectangle)을 이용하여 다차원 공간 데이터들을 색인한다.

본 논문에서는 여러 색인 방법들 중 공간 데이터베이스에서 가장 많이 사용되고 다차원 데이터에 대한 가장 일반적인 색인 구조인 R-trees에 대한 성능을 향상시키는 방법을 제안한다.

2.1. R-trees

R-trees는 다차원 객체들을 색인하기 위해 B-tree를 확장한 것으로 Guttman에 의해 제안되었다[5, 6, 7]. 공간 객체를 최소경계사각형으로 표현하고 그것에 대해 색인을 한다. R-trees의 구조는 단말노드와 비단말노드로 구성된다. 비단말노드는 자식노드를 가르키고 있는 포인터를 가진 엔트리들로 구성되어 있다. 단말노드는 쉽게 데이터가 저장되어 있는 디스크상의 위치를 나타내는 포인터를 가진 엔트리들로 구성되어진다. 동일 레벨에서 최소경계사각형간의 중첩을 허용하는 R-tree의 중첩을 허용하지 않는 R+-tree 방법이 있고 중첩을 가능한 최소한으로 줄이는 R*-tree가 있다. 여러 가지 변형 방법들 중

에서 R*-tree 가 가장 좋은 성능을 보이는 것으로 알려져 있다.

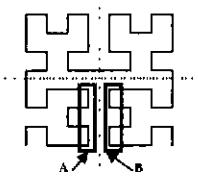
2.2 개선된 R-trees

R-트리와 변형 방법들은 동적인 환경에서 다른 색인 방법들보다 효율적인 삽입과 삭제를 제공한다 이러한 R-트리와 변형 방법들의 특징은 데이터들이 어떠한 순서로 트리에 삽입되느냐에 따라 구성되어지는 트리의 구조와 질의 성능이 많이 달라진다 만약 주어진 데이터가 삽입, 삭제와 같은 수정이 자주 발생하지 않는다면 질의 성능을 향상시키도록 삽입 순서를 결정할 수 있다 즉 데이터에 대해서 R-tree 를 구성할 때 가까이 있는 데이터들은 동일한 노드에 저장하고 전체적으로 노드들의 면적을 최소로 하는 데이터 삽입 순서를 결정하는 것이다

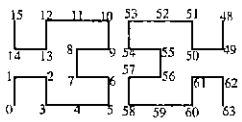
[1]이 이러한 특징을 이용하여 R-tree packing 방법을 제안하였다 이 방법은 삽입과 삭제가 자주 발생하지 않는 정적인 데이타가 주어질 때 공간 활용과 질의 성능면에서 효율적으로 트리를 구성할 수 있도록 하는 방법을 제시하고 있다 데이터에 대한 전처리 작업으로 실제 데이터의 근사인 최소경계사각형에 힐버트 곡선을 이용하여 공간적으로 가까이 있는 데이터들을 연속 값으로 순서를 결정한다. 최소경계사각형에 힐버트값을 구하는 방법은 다음 4 가지 방법을 사용하고 있다 첫번째는 최소경계사각형의 4 개의 꼭 지점을 4 차원의 한 점으로 차원 변환을 하여 그 점에 대해 힐버트값을 계산한다 두번째는 첫번째 방법과 같은 방법으로 최소경계사각형의 중심점과 사각형의 4 면 중 한면을 4 차원의 한 점으로 변환하여 힐버트값을 구한다 세 번째는 각 사각형의 중심점에 대해 힐버트값을 구하는 방법이다 마지막으로 사각형의 중심점에 대해 z-값을 구하는 방법이다. [1]이 제안한 방법은 위의 방법으로 결정된 순서를 정렬하여 R-tree 에 삽입한다.

3. CSR-Tree (Clustering and Sorting-Tree)

2.2 절에서 소개된 것처럼 힐버트 곡선을 이용하여 데이터들의 입력 순서를 결정하는 방법에서는 다음과 같은 문제가 발생한다 힐버트 곡선에서 실제 공간적으로 근접한 데이터를 가까이 있는 것으로 찾지 못하는 경우가 발생할 수 있다.



[그림 3.1] 힐버트 곡선



[그림 3.2] 힐버트 곡선에 대한 힐버트 값

[그림 3.1]은 이러한 문제점을 나타내고 있다. 그림에서 보는 바와 같이 A 부분과 B 부분은 공간적으로 근접하지만 힐버트값은 아주 차이가 많이 나는 것을 볼 수 있다. [그림 3.2]는 [그림 3.1]을 힐버트 값과 함께 자세히 표현한 것이다 이러한 순서로 R-tree 에 삽입되면 가까이 있는 데이터임에도 불구하고 다른 노드에 저장되는 결과를 발생시킨다

본 논문에서 제안하고자 하는 CSR-Tree (Clustering and Sorting R-Tree) 방법은 먼저 데이터에 대해 공간 클러스터링을 한 후, 클러스터 각각에 대해 힐버트 정렬을 하여 이러한 문제점을 해결하고자 한다 클러스터링을 하는 목적은 서로 근접한 데이터들을 하나의 클러스터로 생성하여 가능한 한 동일한 노드에 저장하려는 것이다 그리고 그 클러스터에 대해 힐버트 곡선을 이용하여 정렬하는 경우 A, B 와 같은 지역이 다른 노드에 저장되는 확률은 낮아지므로 성능이 향상 될 수 있다 CSR-tree 의 방법은 클러스터링을 통해 만들어진 각각의 클리스타에 대해 클리스터 중심김을 기준으로 힐버트 값을 구해 정렬을 한다. 클러스터의 중심 점에 대해 정렬이 된 후에 각각의 클러스터내에 속하는 데이터 집합에 대해서도 힐버트 값을 구하여 정렬을 한다 이렇게 두 번의 힐버트값에 대해 정렬을 한 후 그 순서로 R-tree 에 삽입을 하는 방법이다 CSR-Tree 의 특징은 클러스터링을 통해 전체 데이터의 분포를 고려하여 각각의 가까운 데이터간에 정렬을 한다 이러한 순서로 R-tree 에 삽입을 하면 가능한 한 공간적으로 근접한 데이터들은 동일한 노드에 저장을 할 수 있고 결과로 질의 성능을 향상시킬 수 있다. 알고리즘은 다음과 같다.

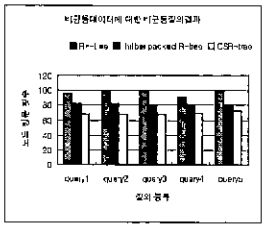
알고리즘 CSR-Tree

```
Input : point set P
// => SMTin 은 주어진 데이터에 대해 TIN 을 생성하여 클리스터
      를 생성하는 함수
Call SMTin(P)
// => n 개의 클러스터 C1, . , Cn 생성한다
// => 각 클러스터 Ci 는 di 개의 데이터를 가진다
for i = 1 to n
  hilbert_sorting( center of i-th cluster Ci )
end for
for i = 1 to n
  for j = 1 to di
    // => Cij 는 i 번째 클러스터의 j 번째 데이터
    hilbert_sorting(Cij)
  end for
end for
save hilbert sorting result to result_file
// => 힐버트 정렬된 순서대로 데이터를 R-tree 에 삽입
insert_Rtree(result_file)
```

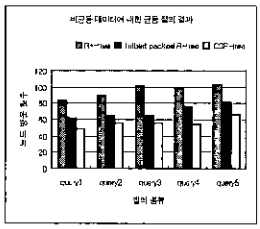
4. 실험 결과

본 논문의 실험에서 사용되는 데이터는 점 객체에 한정한다 벤치마크를 위한 데이터 생성기로 만들어진 인공 점 객체에 대해 균일 분포와 비균등 분포를 생성하여 실험하고 있다 또한 실제 데이터에 대한 실험을 위하여 부산시 지도에서 길 객체를 추출한 데이터를 사용한다. 본 논문에서의 CSR-Tree 의 성능을 평가하기 위해 R*-tree 와 hilbert packed R-tree 에서의 평균 노드 방문 횟수로 비교를 한다

각각 균등 데이터와 비균등 데이터에 대해 R*-tree 에서 1K, 2K 로 페이지 크기를 변화시켰다 질의 영역의 크기를 전체 질의 영역의 0.1%, 0.3%, 0.5%, 1%, 10%로 변화시켜 각각 query1, query2, query3, query4, query5 로 표현하였다 그리고 각 질의 영역 크기에 대하여 질의 분포를 균등과 비균등으로 나누어서 실험하였다.



[그림 4.1] 비균등데이터에 대한 비균등 질의

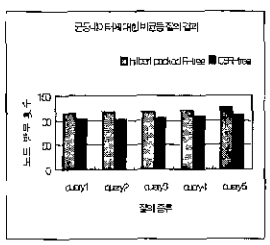


[그림 4.1] 비균등데이터에 대한 균등 질의

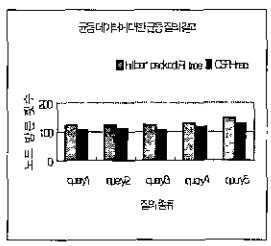
[그림 4.1]은 트리의 페이지 크기를 2K 로 정했을 때, 비균등 데이터에 대한 비균등 분포의 질의를 한 경우 방문된 노드의 평균 횟수이다. [그림 4.2]는 페이지 크기 2K 에 대해 비균등 데이터에 대한 균등 분포의 질의를 했을 때 방문된 노드의 평균 횟수이다 균등 질의 와 비균등 질의 모두 CSR-tree 의 성능이 좋은 것으로 나타난다 Hilbert packed R-tree 는 데이터가 밀집되어 있을수록 가까이 있는 데이터들이 멀리 떨어질 확률이 높아지지만 CSR-tree 에서는 처음에 밀집된 데이터들을 추출하여 그 것에 대해 힐버트 정렬을 하여 그 문제점을 낮추어 줄 수 있기 때문이다

[그림 4.3]은 페이지 크기 2K 에 대해 균등 데이터에 대한 비균등

분포의 질의, [그림 4.4]는 균등 데이터에 대한 균등 분포의 질의를 했을 때 방문된 평균 노드의 횟수를 나타내고 있다. 균등 데이터에서는 일반 R*-tree와 성능면에서 큰 차이가 없으므로 실험에서 제외하였다. 힐버트값에 대해 정렬을 한 후 R*-tree에 삽입하는 경우와 주어진 데이터에 대해 클러스터링을 한 후 각각의 클러스터 내의 데이터에 대해 힐버트 정렬을 하는 경우를 비교하면 데이터의 비균등성이 클수록 본 논문에서 제시하는 방법의 성능 향상을 그래프로 판찰할 수 있다

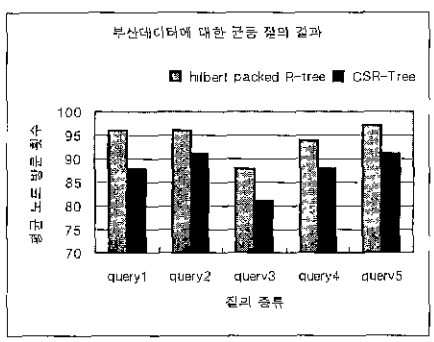


[그림 4.3] 균등데이터에 대한 비균등 질의



[그림 4.4] 균등데이터에 대한 균등 질의

[그림 4.5]는 실제 부산시 지도에서 짐 객체만을 추출하여 균등한 질의를 했을 때 평균 노드 방문 횟수를 나타내고 있다. 실제 데이터에 대해서도 그림에서 보는 바와 같이 CSR-Tree의 평균 노드 방문 횟수가 적은 것을 볼 수 있다.



[그림 4.5] 부산데이터에 대한 균등질의 결과

5. 결론 및 향후 연구 과제

다차원 공간 색인 R-tree와 그 변형 방법들의 특성은 삽입되는 데이터의 순서에 따라 구성되는 트리의 구조와 질의 성능이 영향을 받는다는 것이다. 이러한 특성을 이용하여 주어진 데이터가 삽입, 삭제가 자주 발생하지 않는 정적인 데이터라면 가장 트리를 잘 구성할 수 있도록 데이터의 입력 순서를 결정할 수 있다

[1]에서 제안된 방법은 힐버트 공간 포화 곡선을 이용하여 공간적으로 근접한 데이터들을 가깝게 정렬하여 트리에 삽입하였다. 그러나 이 방법의 문제점은 공간 포화 곡선이 공간적으로 가까이 있는 데이터임에도 불구하고 멀리 있는 것으로 계산되어 다른 노드에 저장이 될 수 있다는 것이다. 즉 데이터에 대한 완전한 근접성을 반영하지 못하는 경우가 발생할 수 있고, 또한 주어진 데이터에 대한 분포를 고려하지 않은 방법이다

본 논문에서는 이러한 문제점을 해결하기 위하여 먼저 데이터에 대해 공간 클러스터링을 함으로써 가까이 있는 데이터 집합을 생성하고 가능한 한 동일한 노드에 저장을 한다. 다음으로 동일한 클러스터에 속하는 데이터에 대하여 힐버트 정렬을 하여 클러스터 내에서 가까운 정도를 반영하여 순서를 정한다. 정해진 순서로 R*-tree에 삽입을 하였다

실험에서 보는 바와 같이 클러스터링을 한 후에 힐버트 정렬을 한 경우가 무작위로 데이터들을 삽입하는 방법과 힐버트 정렬 후에 삽입하는 방법보다 질의에 방문된 횟수가 적음을 알 수 있다. [표 5.1]은 비균등 데이터에 대해서 무작위로 트리에 삽입하는 경우와 힐버트값에 따라 정렬하여 삽입하는 경우에 대비하여 클러스터링을 한 후 힐버트값에 따라 정렬하는 경우의 성능향상 정도를 나타내고 있다. [표 5.2]는 균등 질의에 대한 성능 향상을 비교한 것이다.

향후 연구과제는 정적인 데이터에서의 최적의 R-tree를 구성되고 난 후에 동적으로 삽입과 삭제가 일어나는 경우에서도 최적의 트리 구조를 유지할 수 있도록 하는 문제가 있다. 이는 한 클러스터의 데이터 개수가 한 노드의 페이지 크기보다 클 경우 기본은 분할되어 지정되는 데 이것을 클러스터링을 할 때 클러스터 내에 속하는 데이터 개수에 제한을 두는 방법을 고려해야 한다. 또한 삽입이 발생할 때 최대 엔트리 수를 넘어설 경우 분할을 하는 데 이때 이미 유지되어 있는 공간 근접성을 분할이 발생하여 위배되지 않도록 하는 문제도 고려되어야 한다

비교대상 \ 질의종류	Query1	Query2	Query3	Query4	Query5
HSR-tree vs CSR-tree	20.9%	15.4%	14.1%	28.9%	19.5%
R*-tree vs. CSR-tree	48.1%	53.1%	55.1%	40.6%	35.3%

[표 5.1] 비균등 데이터에 대한 비균등질의

비교대상 \ 질의종류	Query1	Query2	Query3	Query4	Query5
HSR-tree vs CSR-tree	18.1%	17.1%	17.3%	14.8%	9.8%
R*-tree vs CSR-tree	23.5%	23.5%	27.2%	31.7%	25.5%

[표 5.1] 비균등 데이터에 대한 균등질의

6. 참고문헌

[1] I Kamel, C Faloutsos, "On Packing R-trees", In Proc 2nd International Conference on Information and Knowledge Management(CIKM-93), November 1993.

[2] J Bercken, B Seeger and P Widmayer, "A Generic Approach to Bulk Loading Multidimensional Index Structures", VLDB, 1997

[3] L Chen, R Choubey and E A Rundensteiner, "Bulk-Insertions into R-trees", ACM-GIS, 1997

[4] I.S. Kang, T.W Kim and K.J. Lik, "A Spatial Data Mining Method by Delaunay Triangulation", ACM-GIS, Proc of 5th, 1997

[5] N Beckmann, H P Kniegel, R Schneider, and B Seeger, "Ther R*-tree an efficient and robust access method for points and rectangles", ACM SIGMOD, 1990

[6] A Guttman, 'R-trees,a dynamic index structure for spatial searching", Proc ACM SIGMOD, 1984

[7] T. Sellis, N Roussopoulos, and C Foloutsos, "The R+-tree a dynamic index for multi-dimenstional objects", In Proc 13th International conference on VLDB, 1987

[8] S. T Leutenegger. M. A Lopez, and J Edgington, "STR A Simple and Efficient Algorithm for R-Tree Packing", Proc ICDE, 1997

[9] R T Ng and J Han, "Efficient and Effective Clustering Methods for Spatial Data Mining", Proc. VLDB Conference, 1994