The Journal of GIS Association of Korea, Vol.6, No.2, pp.125-132, October 1998

지리정보시스템에서 공간 색인기법에 관한 연구

황병연*

A Study on the Spatial Indexing Scheme in Geographic Information Systems

Byung-Yeon Hwang*

요 약

지리정보시스템을 위한 공간 데이터베이스는 공간 데이터의 특성에 의해 다차원의 대용량 데이터를 다루기 때문에 공간 질의의 I/O 성능이 매우 중요한 역할을 한다. 따라서 본 논문에서는 공간 질의의 I/O 성능을 높이기 위해서 복잡한 공간 객체들을 다루는 대표적인 접근기법들인 Z-변형을 이용한 B 트리, KDB 트리, R 트리, MAX 트리에 대해 기술하였다.

또한, 다양한 실제 데이터와 질의 집합을 사용해서 여러 공간 색인 기법들의 성능을 측정하였다. 벤치마크 실험을 해본 결과, MAX 트리는 삽입, 영역 질의, 공간 조인 등의 연산에 대해 다른 색인 기법들보다 상대적으로 좋은 성능을 나타냈다. MAX 트리는 향후 GIS의 하부 저장시스템을 구성하는 색인기법으로 사용될 것으로 기대된다.

ABSTRACT : The I/O performance for spatial queries is extremely important since the handling of huge amount of multidimensional data is required in spatial databases for geographic information systems. Therefore, we describe representative spatial access methods handling complex spatial objects, z-transform B tree, KDB tree, R tree, MAX tree, to increase I/O performance.

In addition, we measure the performance of spatial indexing schemes by testing against various realistic data and query sets. Results from the benchmark test indicates that MAX outperforms other indexing schemes on insertion, range query, spatial join. MAX tree is expected to use as index scheme organizing storage system of GIS in the future.

1. 서 론

최근에 지리정보시스템(GIS : geographic information systems), VLSI 설계, 멀티미디어 문서 검색 등 여러 개의 키를 갖는 데이터베이스를 이용하는 응용이 점차 늘어나고 있다. GIS에서 데이터베이스는 2차원 지도상에 존재하는 공간 객체들을 저장한다. 각 공간 객체는 도시, 도로, 호수 등과

* 가톨릭대학교 컴퓨터공학과(Dept. of Computer Engineering, The Catholic University of Korea, 43-1, Yokkok 2-Dong, Wonmi-Gu, Puchon, Kyong Ki-Do 420-743, Korea)

같은 특정한 엔티티 클래스에 속하는 것으로 분류될 수 있다. 객체들은 인구, 이름, 용도 등의 비공간 속성과 위치와 같은 공간 속성에 의해 표현된다. 이와 같은 속성을 갖는 데이터베이스를 공간 데이터베이스(spatial database)라고 한다(Gunther & Buchmann, 1990). 공간데이터베이스에서는 효율적으로 공간 객체들을 검색하기 위해 교차(intersection), 포함(contains), 근접(proximity) 질의를 위한 공간 연산자들이 추가된다. 예를 들어 "종로와 교차하는 거리를 탐색하라"와 같은 영역 질의(range query)의 경우 주어진 질의 영역과 겹치는 모든 객체들을 검색하게 된다. 영역 질의의 특별한 경우가 질의 영역이 점으로 줄어드는 점 질의(point query)이다. 이와 같은 공간관계를 다루는 질의를 효율적으로 처리하기 위해 별도의 색인 구조를 사용한다(Medeiros & Pires, 1994). 또한 공간 객체는 공간상에서 크기(extents)를 가지므로 색인 기법은 크기를 처리할 수 있는 구조를 가져야 한다.

GIS 데이터를 처리하기 위한 색인 기법으로 많은 연구가 있어 왔으며(Zou & Salzberg, 1996; Berchtold, 1996), 이들 연구에서는 주로 다중 키에 대해서 어떠한 사상을 거치지 않고 직접 색인에 적용하였다. 이러한 연구들의 특징은 다중 키에 의한 데이터의 검색과 삽입, 삭제, 갱신 연산에 대한 효율적인 알고리즘을 제시하고 있으며, 데이터베이스로 구성된 데이터의 특성에 따라 성능이 상대적으로 많은 차이를 보인다.

본 연구에서는 GIS 데이터를 다루는 대표적 색인 기법인 Z-변형을 이용한 B 트리(Comer, 1979), KDB 트리(Robinson, 1981), R 트리(Guttman, 1984), MAX 트리(황병연 & 김병욱, 1997)에 대해 벤치마크 실험을 통해 성능을 비교한다.

2. 관련 연구

2.1 Z-변형을 이용한 B 트리

기존의 B 트리와 같은 1차원의 색인 방법을 그대로 사용할 수 있는 방법으로 구성이 가장 손쉬운 색인 방법이다. 즉, n개의 키 값을 가지고 색인을 구성하는 것을 n차원의 공간에 점을 위치시키는 것으로 생각할 때, n차원의 점을 1차원으로 사상시켜서 마치 수직선상에 점을 위치시키는 것으로 생각할 수 있다. 이 방법이 안고 있는 문제점은 n차원에서 모여 있던 점들의 집합이 수직선상에 사상될 때, 원래 가지고 있던 특성과는 다르게 점들이 흩어지는 것이다. 이러한 문제점은 색인에 있어서 상당한 단점으로 작용하게 되므로 이러한 색인 방법은 실제로 사용되기 어렵다. 사상시키는 방법으로는 단순히 키를 연결하는 것과 Z-순서와 같이 사상되는 점의 수직선상의 순서를 좀더 좋은 방향으로 변경하는 방법이 있다.

2.2 KDB 트리

KDB 트리는 일반적인 B 트리를 다 차원으로 확장한 균형 트리이며, B 트리의 특성을 대부분 그대로 가지고 있다. 그러나 B 트리가 상향적으로 성장하는 반면, KDB 트리의 경우는 상향과 하향 모두 성장하게 된다. 이러한 이유 때문에 데이터의 분포가 균일하지 못한 경우 상당히 많은 하향 성장을 가져오게 된다. 한편, 실제 사용되는 데이터는 균일한 분포일 가능성이 적으므로 KDB 트리를 실제 데이터에 적용하여 색인을 구성하는 것은 시스템의 성능을 상당히 저하시킬 수 있다.

2.3 R 트리

R 트리는 원래 공간상의 점을 색인하기 위한 기법이라기보다 공간상의 범위를 가지는 데이터(예 : 정지 화상)에 대한 색인 기법이라고 할 수 있다. 하지만 공간상의 점에 대한 색인에 적용할 수도 있기 때문에, 다중 키 색인 방법으로 분류될 수 있다. R 트리는 공간상의 점을 둘러싸는 2n개의 초평면을 단위로 트리를 구성하고 있다. 즉, 2차원의

경우 직사각형을 이루는 4개의 초평면으로 구성된 부분 공간상에 점들을 사상시키는 형태로 색인을 구성한다. 2n개의 초평면이 구성되어야 하므로 여기에 소요되는 저장 공간 또한 상당하다고 볼 수 있다. 또한 초평면을 구성하기 위한 계산 시간은 다른 색인 방법에 비해 상대적으로 상당히 많으며, 알고리즘이 복잡한 관계로 그 구현에 있어서 많은 어려움을 가지고 있다.

2.4 MAX 트리

MAX 트리는 초평면을 각각의 축에 대해 직교하도록 구성한 색인 방법으로 초평면의 차원이 n차원이고 개수 또한 n개로 구성되며, 축에 대해 직교하는 비교적 단순한 구조를 가지게 된다. 전체 공간은 직교하는 2개의 초평면에 의해서 4개의 부분 공간으로 구성되어 있다. MAX 색인 방법은 기존의 B 트리를 다중 키 형태로 변형하는 것에 불과하므로 구축 시간 또한 크게 줄일 수 있는 장점이 있다.

3. 벤치마크 실험 설계

어떤 시스템의 성능을 평가하는 방법은 시뮬레이션 방법과 벤치마크 실험방법으로 분류할 수 있다. 먼저 시뮬레이션의 경우는 완전한 구현은 아니지만 실제 시스템과 흡사한 모형을 바탕으로 실험이 이루어진다. 이를 통해 시스템의 성능을 제대로 측정하기 위해서는 사실에 근접한 다량의 시험 데이터가 필요하다. 또한 성능 평가 후에 사용될 시스템이 실제 환경을 바탕으로 구현되어야 한다. 하지만, 시뮬레이션 결과가 좋다고 하더라도 실험은 가상의 환경을 바탕으로 하였으므로 실제 구현된 시스템에서는 똑같은 성능이 보장되는 것은 아니다.

벤치마크 실험은 실제의 환경에 맞추어 시스템을 완전히 구현한 후에 성능이 측정된다. 따라서 시스템의 성능을 예측하지 못한 상태이기 때문에 벤치마크 실험을 통한 성능 평가는 상당히 위험할 수 있다. 또한 이 실험은 다량의 실제 데이터를 필요로 한다. 가장 좋은 방법은 실제 데이터를 시뮬레이션을 통해 성능평가를 가진 다음 시스템을 구현하고 그 시스템 위에서 벤치마크 실험을 하는 것이다.

두 단계의 성능 측정은 많은 시간을 필요로 하기 때문에, 본 연구에서는 실제 데이터와 유사한 데이터를 가지고 벤치마크 실험을 하였다. 본 연구에서 측정한 성능은 실제 상황을 유사하게 반영하고 있으므로 믿을 만한 결과라고 말할 수 있다.

3.1 데이터의 설계

벤치마크 데이터 세트는 실제 응용을 반영하도록 설계되어야 한다. 특히 공간 데이터베이스 응용에서는 주로 다중 애트리뷰트를 사용한다는 것을 고려해야 한다. 이러한 다중 애트리뷰트 데이터는 점 데이터일 수도 있고 영역 데이터일 수도 있다. 본 연구의 목적이 점 데이터에 있으므로, 여기서는 점 데이터를 기준으로 한다. 본 연구에서 가정하고 있는 데이터의 특징은 다음과 같다.

화일은 제한된 수의 레코드로 구성되어 있으며, 각 레코드는 2개의 키를 가지고 있다. 각 필드의 레코드는 정수나 문자 스트링이고, 키 필드는 수치적으로 서로 비교 가능하다. 데이터베이스는 공간적으로 특정한 영역에 결집될 수 있고 데이터는 특정한 키에 대해 정렬될 필요는 없으며, 다중 색인 방법이 성능향상을 위해 사용된다.

이러한 가정을 바탕으로 2개의 키 필드를 가지는 데이터 화일은 다음과 같은 특징이 있다. 각 키 필드는 범위를 가지는 정수로 구성되며, 두개의 키 모두 [0, 10000)의 범위에 존재한다. 이것은 최대 100,000,000 (10^8)의 점으로 구성되어 있는 공간을 가정한다. 한 레코드의 길이는 255바이트로 가정한다.

결집도는 2차원 공간상의 면적 비율로 결정된다. 결집도가 n일 경우 전체 공간의 10^n부분에 해당하

는 부분 공간에 70%의 데이터가 존재하게 된다. 나머지 30%는 전체 데이터 공간에 공평하게 분포한다고 가정한다. 70%라는 값은 실험을 통해 얻어진 것으로 실제 데이터를 충분히 고려한 것이다.

10만개의 레코드가 실험용으로 사용되지만 때로는 100만개의 레코드가 사용되는 경우도 있다. 그러나 100만개의 레코드는 많은 실험 시간을 요하게 되므로 본 연구에서는 10만개의 레코드를 기준으로 실험을 하였다.

3.2 질의 설계

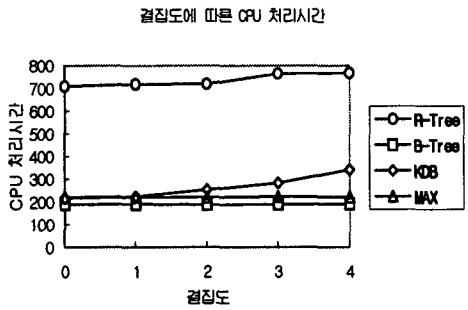
n차원에 대한 질의의 형태는 각 차원에 대해서 구간을 가지는 영역으로 본 연구에서는 5개의 질의 영역을 정의하였다. 전체 데이터 공간의 1%를 차지하는 질의를 '극소형' 질의, 5%를 차지하는 질의를 '소형' 질의, 10%를 차지하는 질의를 '중간형' 질의, 20%를 차지하는 질의를 '대형' 질의, 40%를 차지하는 질의를 '초대형' 질의로 명명하였다.

4. 실험 결과

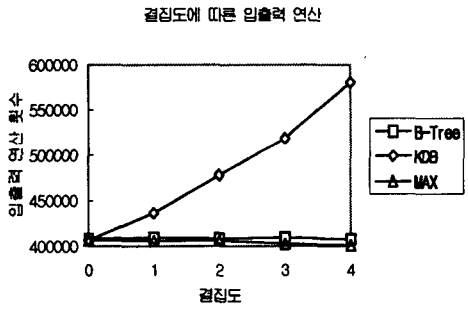
성능 평가는 데이터의 삽입, 영역 검색(질의), 공간 조인, 그리고 삭제 연산에 대해서 실시되었다. 삽입 연산은 5가지의 결집도에 대해서 10개의 레코드를 기준으로 실험이 진행되었다. 삽입 연산에 대해서는 CPU 사용시간, 저장공간 사용량과 이용률, 읽기/쓰기 횟수 등이 측정되었다. 영역 질의와 공간 조인에 대해서는 읽기 연산의 횟수와 잘못된 검색의 횟수로 성능을 측정하였다. 삭제 연산에 대해서는 읽기/쓰기 연산의 횟수로 성능을 측정하였다. 본 연구에서는 이러한 벤치마크 실험을 운영체제 KLE Solaris 2.5에서 운영되는 Axil-Ultima 170 워크스테이션에서 실시하였다.

4.1 삽입 연산

삽입 연산의 성능은 CPU 사용시간, 저장공간 사용량과 이용률, 읽기/쓰기 연산의 횟수로 측정되었다. 측정 횟수는 각기 다른 5개의 결집도를 가진 데이터에 대해서 20번 반복하였으며, 이의 평균 값을 결과로 비교하였다.

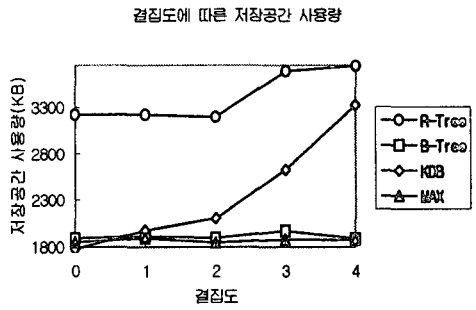


〈Fig. 4.1〉 CPU time versus clustering

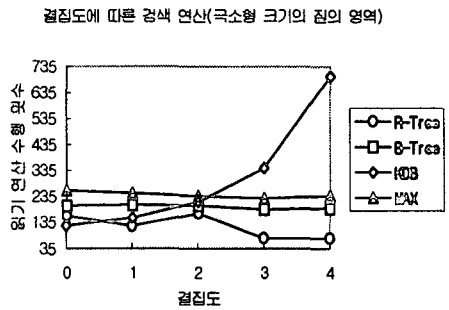


〈Fig. 4.2〉 Number of I/O versus clustering

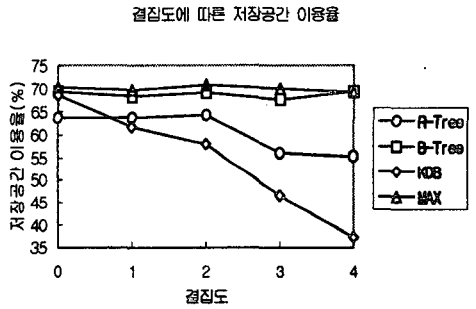
<Fig. 4.1>과 <Fig. 4.2>는 결집도에 따른 CPU 사용 시간과 읽기/쓰기 연산의 횟수를 각 색인 기법별로 보여주고 있다. 그림에서 보는 바와 같이 MAX는 상대적으로 적은 CPU 사용시간과 적은 읽기/쓰기 연산 횟수를 나타내었다. 읽기/쓰기 연산의 횟수는 R 트리의 경우 현격한 차이로 인해 표시하지 않았으며, KDB 트리의 경우 결집도가 증가함에 따라 급격한 성능 저하를 나타내는 것을 볼 수 있다.



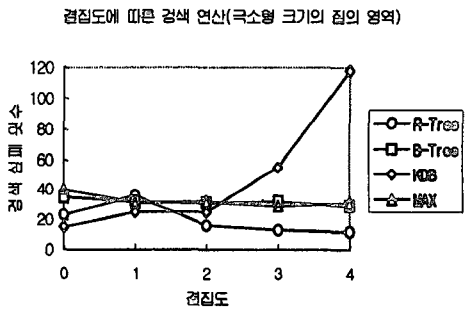
〈Fig. 4.3〉 Storage overhead versus clustering



〈Fig. 4.5〉 Number of read versus clustering(small range query)



〈Fig. 4.4〉 Storage utilization versus clustering



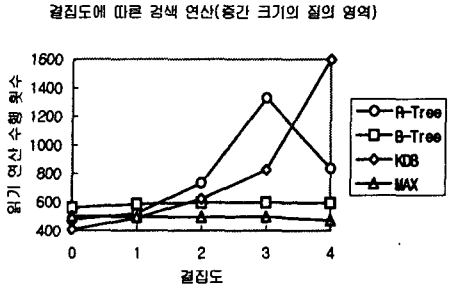
〈Fig. 4.6〉 Number of read fault versus clustering(small range query)

저장공간 사용량과 이용률에 대해서는 <Fig. 4.3>과 <Fig. 4.4>에서 보는 바와 같이 MAX는 결집도와 상관없이 거의 일정하게 유지되는 반면, KDB 트리는 결집도가 증가함에 따라 급격한 성능 저하를 나타내고 있다. R 트리의 경우는 상당히 많은 저장공간을 사용하는 것으로 나타났다.

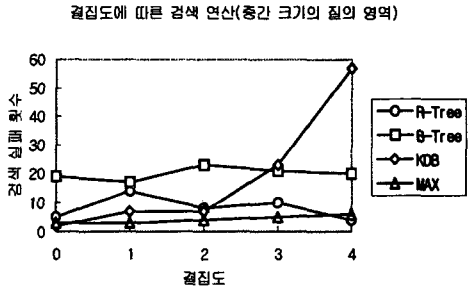
4.2 영역 질의

영역 질의에 대해서는 질의 영역의 크기를 크기별로 5가지로 나누어서 성능을 측정하였다. 성능 측정의 항목으로는 읽기 연산의 횟수와 잘못된 검색의 횟수로 검색 연산에서 가장 중요한 2가지를 고려하였다.

'극소형' 영역 질의에 대해서는 <Fig. 4.5>와 <Fig. 4.6>에서 보는 바와 같이 R 트리가 결집도와는 상관없이 가장 좋은 성능을 나타냈다. KDB 트리의 경우 결집도가 증가함에 따라 성능이 현저히 저하되었다. 영역 질의의 질의 영역 크기가 아주 작을 경우 질의 영역과 겹치는 데이터 공간이 작으므로 R 트리가 가장 유리하다고 할 수 있다.

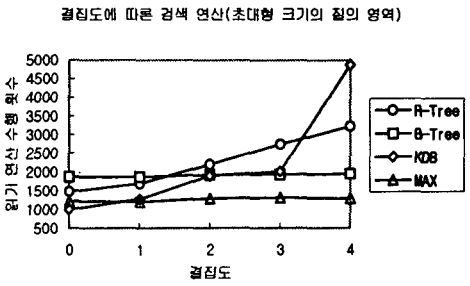


〈Fig. 4.7〉 Number of read versus clustering(medium range query)

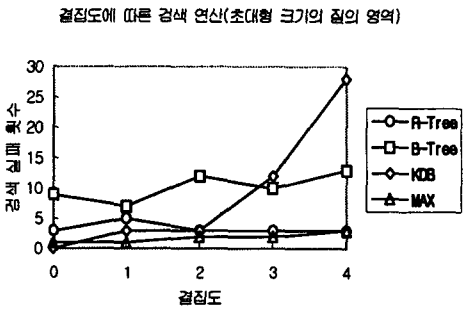


〈Fig. 4.8〉 Number of read fault versus clustering(medium range query)

'중간형' 영역 질의에 대해서는 <Fig. 4.7>과 <Fig. 4.8>에서 보는 바와 같이 결집도와 상관없이 MAX가 가장 좋은 성능을 나타냈다. 반면에 KDB 트리와 R 트리는 결집도에 따라 심한 성능의 변화를 보여주고 있다.



〈Fig. 4.9〉 Number of read versus clustering(large range query)

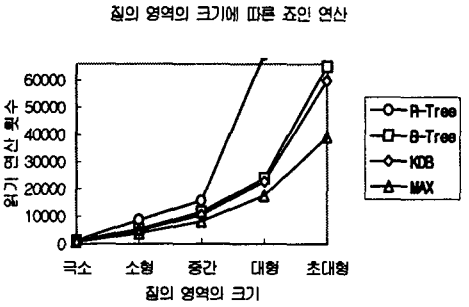


〈Fig. 4.10〉 Number of read fault versus clustering(large range query)

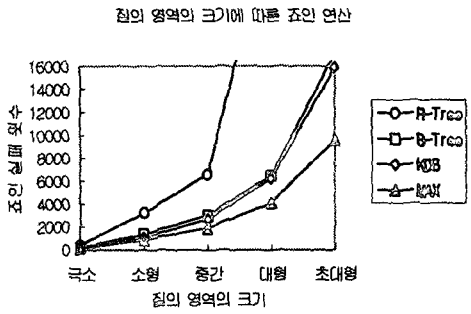
'초대형' 영역 질의에 대해서는 <Fig. 4.9>와 <Fig. 4.10>에서 보는 바와 같이 결집도와 상관없이 MAX가 가장 좋은 성능을 나타냈다. '중간형' 영역 질의와 비슷한 결과를 나타낸 것으로 볼 때 질의 영역의 크기가 클 수록 MAX는 좋은 성능을 나타낸다고 볼 수 있다.

4.3 공간 조인

공간 조인은 두 공간 데이터 집합에 대해서 질의 영역에 해당되는 부분에 공통으로 겹치는 데이터를 검색하는 것이다. 이 연산은 상당히 많은 데이터를 검색할 필요가 있으며, 따라서 전체 데이터베이스 응용의 성능을 크게 좌우한다고 볼 수 있다.



〈Fig. 4.11〉 Number of read versus size of query range(join)

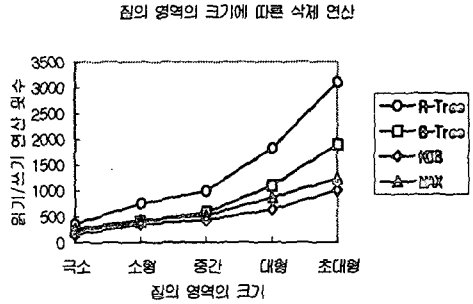


〈Fig. 4.12〉 Number of join fault versus size of query range(join)

본 연구에서는 질의 영역의 크기를 5가지로 변형시키면서 실험을 한 결과 <Fig. 4.11>과 <Fig. 4.12>와 같이 읽기/쓰기 연산과 잘못된 검색 횟수 면에서 MAX가 뛰어난 성능을 나타내었다.

4.4 공간 삭제

공간 상의 일부 부분 공간을 삭제하는 것이 공간 삭제이다. 이 연산은 공간 조인과 더불어 많은 읽기/쓰기 연산을 필요로 하는 중요한 연산이다. 또한 공간 삭제이후 색인 트리를 그대로 유지하여야 하므로 더욱 중요하다고 볼 수 있다.



〈Fig. 4.13〉 Number of I/O versus size of query range(deletion)

이 연산에 대해서는 읽기/쓰기 연산의 횟수로 성능을 측정하였으며, 측정 결과 <Fig. 4.13>에서 보는 바와 같이 MAX는 KDB 트리 다음으로 좋은 성능을 나타냈다. R 트리의 경우는 본래 색인을 구성하기 위한 저장 공간을 많이 차지하게 되므로 좋지 못한 성능을 나타낸다.

5. 결 론

본 연구에서는 GIS 데이터를 다루는 대표적 색인 기법인 Z-변형을 이용한 B 트리, KDB 트리, R 트리, MAX 트리에 대해 특징을 간략히 설명하고, 벤치마크 실험을 통해 성능을 비교하였다. 실험 결과 MAX 트리는 삽입, 삭제, 검색, 조인 연산에서 다른 기법들에 비해 상대적으로 좋은 성능을 나타냈다.

MAX 색인 방법이 GIS에 적용될 경우 시스템의 성능은 향상될 것이며, MAX를 채용한 하부 저장 시스템은 일반적인 GIS의 하부 저장 시스템으로 사용될 수 있을 것이다. 다만 색인의 특성상 점 데이터에 한정되어 있고, 범위 데이터에 대해서는 적용할 수 없는 단점이 있으나 범위 데이터에 대해서는 이미 R* 트리(Beckmann, 1990)와 같은 상당히 좋은 기법이 제안되어 있으므로 이와 결합하여 시스템을 구축할 경우 상당히 성능이 우수한 GIS를 구축할 수 있을 것으로 기대된다. 본 연구에서는 연구의 범위를 점 데이터로 한정하였으며, 범위 데이터를 포함한 다양한 형태의 데이터에 대해서는 앞으로 계속 연구해 나갈 것이다.

참 고 문 헌

황병연, 김병욱, 1997, "지리 정보 데이터베이스에서 대용량의 공간 객체를 위한 저장 관리 시스템에 관한 연구," 한국GIS학회지, 5권 1호, pp.1-10.

Beckmann, N., Kriegel, H. P., Schneider, R.,

and Seeger, B., 1990, "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles," Proc. of ACM SIGMOD Conference, pp.322-331.

Berchtold, S., Keim, D. A., and Kriegel, H. P., 1996, "The X-tree : An Index Structure for High-Dimensional Data," Proc. of Very Large Databases Conference, pp.28-39.

Comer, D., 1979, "The Ubiquitous B-Tree," ACM Computing Surveys, Vol. 11, No. 2, pp.121-137.

Gunther, O. and Buchmann, A., 1990, "Research Issues in Spatial Databases," Proc. of ACM SIGMOD Record, Vol. 19, No. 4, pp.61-68.

Guttman, A., 1984, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. of ACM SIGMOD Conference, pp.47-57.

Medeiros, C. B. and Pires, F., 1994, "Database for GIS," Proc. of ACM SIGMOD Record, Vol. 23, No. 1, pp.107-115.

Robinson, J. T., 1981, "The KDB-Tree: A Search Structure for Large Multidimensional Dynamic Indexes," Proc. of ACM SIGMOD Conference, pp.10-18.

Zou, C. and Salzberg, B., 1996, "On-line Reorganization of Sparsely-populated B+-trees," Proc. of ACM SIGMOD Conference, pp.115-124.