[1]지리 정보 시스템에서 대용량의 복잡한 객체를 위한 저장 관리

황병연 김병욱
가톨릭대학교 전산학과 (주)APAC 연구소

Storage Management for Large Complex Objects in Geographic Information Systems

Byungyeon Hwang Byungwook Kim

요약

본 연구에서는 지리 정보 시스템(GIS: geographic information system)에서 주로 사용되는 공간 객체를 위한 색인 방법에 대해서 과거의 연구를 토대로 분류를 해보고 이로부터 새로이 제안하는 색인 방법인 MAX (Multi-Attribute indeXing scheme)에 대해서 상세히 기술한다. 또한 MAX의 여러 연산을 위한 알고리즘을 제시하고, 알고리즘의 우수성을 제시한다.

이미 성능 평가를 통해 어느 정도의 성능을 기대할 수 있으며, 이를 실제 시스템에 구현한다면 상당한 성능을 가진 지리 정보 시스템을 구축할 수 있을 것이다. 특히 MAX는 이 기법이 가지는 B 트리의 확장성으로 쉽게 구현할 수 있는 구조를 가지게 된다.

1. 이 논문은 1996학년도 가톨릭대학교 교비 연구비에 의하여 연구되었음.

1. GIS를 위한 다중 키 색인 기법의 개요

데이타베이스 시스템에 저장되는 객체 혹은 레코드의 구조는 점차 복잡해지고 있으며, 또한 이를 구별하기 위한 키는 과거의 단일 필드(field)에 의하지 않고 몇 개의 키를 조합시키는 방식으로 변화되고 있다. 이러한 레코드 구조의 복잡성과 복합키의 존재는 데이타베이스 시스템의 기본 골격을 이루는 색인 관리에 상당한 영향을 미친다. 특히 여러 개의 키에 의한 (위도와 경도등 좌표 값에 의한) 다중 키 검색을 주로 사용하는 GIS의 경우, 색인 관리가 제대로 이루어지지 않을 경우 시스템의 성능은 상당히 저하될 것이다.

색인에 의한 객체 검색 방법은 이미 오래 전부터 사용되어 오던 전통적인 데이타베이스 관리 시스템의 기본 사항으로, 색인은 주로 B 트리와 그 변형[Come 79]에 의해 이루어져 왔다. 현재 운영되고 있는 대부분의 데이타베이스 관리 시스템에서 채택하고 있는 색인 방법은 B^+ 트리이며, 간혹 해싱(hashing) 방법을 사용하는 경우도 있다[Otoo 84]. 이러한 색인 방법은 모두 단일 키를 가정하고 있으며, 다중 키에 대해서는 키를 단순히 연결하여 붙여서 단일 키로 사상시켜 색인에 적용하였다. 그러나 이러한 방법은 다중 키에 의한 검색 효율을 저하시키는 요인이 되기 때문에, 이에 대한 보완으로 다중 키를 단순히 연결하지 않고 색다른 순서에 의해 다시 조합하여 단일 키로 사상시키는 방법이 사용되고 있다[Oren 82, Oren 86]. 이 방법은 비교적 효율적인 검색을 보장하지만, 다중 키를 단일 키로 사상시키는 단계에서 키의 순서가 원래의 순서와 차이가 남으로써 순차적인 검색에 대해서 좋지 않은 성능을 보이게 된다.

앞 절에서 설명한 다중 키 색인 방법을 보완하기 위한 색인 기법으로 많은 연구가 있어 왔으며[Bent 75, Bent 79, Free 87, Gutt 84, Niev 84, Robi 81, Sell 87], 이들 연구에서는 주로 다중 키에 대해서 어떠한 사상을 거치지 않고 직접 색인에 적용하였다. 이러한 연구들의 특징은 다중 키에 의한 데이타의 검색과 삽입, 삭제, 갱신 연산에 대한 효율적인 알고리즘을 제시하고 있으며, 데이타베이스로 구성된 데이타의 특성에 따라 성능이 상대적으로 많은 차이를 보이는 것

이다. 예를 들어 그리드 화일[Niev 84]의 경우 데이타가 균등하게 분포될 경우 상당히 좋은 성능을 보장하지만, 대부분의 실제 데이타가 그렇듯이 데이타가 전체 도메인의 일부분에 결집되어 분포되는 경우에 현저한 성능 저하를 가져오게 된다. 한편, KDB 트리[Robi 81]의 경우 B 트리의 특징을 그대로 살릴 수 있는 장점을 가지고 있지만 이 기법 또한 데이타의 분포가 심하게 불균형한 경우에 상당한 성능 저하를 가져오게 되는 구조적인 단점을 가지고 있다.

앞에서 설명한 기존의 색인 방법은 점(point) 데이타에 대한 것으로, 이들 색인 방법에서는 범위를 가지는 데이타에 대해서는 적용이 불가능하다. 범위 데이타란 이미지(image)나 벡터 그래픽과 같은 다차원의 범위를 가지는 비정형적 데이타를 말한다. 이러한 범위 데이타에 대해서는 R 트리[Gutt 84], R^+ 트리[Sell 87], 그리고 R^* 트리[Beck 90] 등의 색인 기법이 제안되어 있다. 실제로 R 트리는 GIS 데이타베이스를 구축하는데 자주 사용되고 있으며, R^* 트리는 R 트리의 단점을 보완한 최근의 범위 데이타용 색인 기법이다. 본 연구에서는 연구의 범위를 점 데이타로 한정하며, 범위 데이타에 대해서는 추후의 연구 과제로 남겨둔다.

앞절에서 언급한 기존의 색인 기법은 단순히 다중 키를 위한 색인 방법을 알고리즘으로 제안했을 뿐 실제 시스템에 적용 가능한 구현 방법이라든지 기존 시스템을 확장하는 방안에 대해서는 거의 제안된 바가 없다. 구현 방법과 시스템의 확장 방안은 실제 시스템을 구축하는 데 있어서 상당히 중요한 요소가 되므로 이에 대한 언급을 구체적으로 할 필요가 있다. 또한 이들 색인 기법들은 나름대로의 장점을 가지고 있지만, 상황에 따라 상대적으로 많은 단점을 가지고 있다. 본 연구에서는 기존의 다중 키 색인에 대한 연구를 문제점 위주로 분석해 보고 이에 대한 대안으로 새로운 다중 키 색인 방법을 제안하고, 데이타베이스 관리 시스템을 이루는 저장 관리자에 본 색인 방법을 적용하여 GIS 데이타베이스용 저장 관리 시스템 구축을 위한 기법에 대해서 구체적 구현 방안을 제시한다.

2. 기존의 다중 키 색인 방법

지금까지 연구되었던 다중 키 색인 방법은 다중 키의 공간을 여러 개의 부분 공간으로 나누는 초평면(hyperplane)의 특성에 따라 다음과 같은 7가지 형태로 분류할 수 있다. 여기서 초평면의 특성이란 초평면의 차원, 축에 대한 상대적 벡터, 그리고 초평면의 개수이고, n개의 다중 키 공간을 가정한다.

- 클래스 C1 (축에 직교하는 1차원의 1개 초평면)
- 클래스 C2 (축에 평행하는 n차원의 1개 초평면)
- 클래스 C3 (축에 평행하는 n차원의 $2n$개 초평면)
- 클래스 C4 (곡선형 n차원 1개 초평면)
- 클래스 C5 (축에 대하여 일정치 않은 n차원의 1개 초평면)
- 클래스 C6 (축에 대하여 일정치 않은 n차원의 m개수의 초평면)
- 클래스 C7 (축에 대하여 직교하는 n차원의 n개 초평면)

각 클래스에 대한 기존의 다중 키 색인에 관한 연구들은 다음과 같은 표로서 작성될 수 있다. 이들 각각에 대한 자세한 비교는 다음 절에서 다루기로 한다.

표 1. 다중 키 색인 방법의 분류

클래스	초평면의 특성			색인 방법
	차원	개수	벡터	
(C1)	1	1	1차원	B 트리
(C2)	n	1	축에 평행	KDB 트리, 그리드 화일
(C3)	n	$2n$	축에 평행	R 트리, R^+ 트리, R^* 트리
(C4)	n	1	곡면	*MAIN*[Kim 93]
(C5)	n	1	불규칙	기존의 연구 없음
(C6)	n	가변	불규칙	
(C7)	n	n	n개의 축에 직교	

표 1에서 보듯이 클래스 (C5), (C6), (C7)은 아직까지 연구된 바가 없으며, 따라서 이 클래스에 대한 자세한 검토가 중요하다.

2.1 클래스 (C1): 1차원으로의 사상

클래스 (C1)에 속한 색인 방법은 2장 서두에서 설명한 것과 같이 기존의 B 트리와 같은 1차원의 색인 방법을 그대로 사용하는 구성이 가장 손쉬운 색인 방법이다. 즉, n개의 키 값을 가지고 색인을 구성하는 것을 n차원의 공간에 점을 위치시키는 것으로 생각할 때, 클래스 C1의 경우, n차원의 점을 1차원으로 사상시켜서 마치 수직선상에 점을 위치시키는 것으로 생각할 수 있다. 이 방법이 안고 있는 문제점은 n차원에서 모여 있던 점들의 집합이 수직선상에 사상 될 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-순서[Oren 86]와 같이 사상되는 점의 수직선상의 순서를 좀더 좋은 방향으로 변경하는 방법이 있다.

2.2 클래스 (C2): 축에 평행한 초평면

클래스 (C1)과는 다르게 클래스 (C2)의 경우 초평면의 차원이 n차원이며, 따라서 (C1)에서 발생되는 문제를 해결할 수 있는 색인 방법이다. 여기에는 대표적으로 KDB 트리와 그리드 화일 등이 속한다. 하지만 KDB 트리와 그리드 화일은 클래스 (C1)과는 다른 문제를 가지고 있다. 이들 색인 방법에서 발생하는 문제는 보통의 데이타 분포에서는 나타나지 않으며, 데이타가 일부분에 집중될 경우 심각하게 나타난다.

KDB 트리는 일반적인 B 트리를 다 차원으로 확장한 균형 트리이며, B 트

리의 특성을 대부분 그대로 가지고 있다. 그러나 B 트리가 상향적으로 성장하는 반면, KDB 트리의 경우는 상향과 하향 모두 성장하게 된다. 이러한 이유 때문에 데이타의 분포가 균일하지 못한 경우 상당히 많은 하향 성장을 가져오게 된다. 따라서 실제 사용되는 데이타는 균일한 분포일 가능성이 적으므로 KDB 트리를 실제 데이타에 적용하여 색인을 구성하는 것은 시스템의 성능을 상당히 저하시킬 수 있다.

그리드 화일 색인 방법은 KDB 트리와 조금 다른 문제점을 가지고 있다. 그리드 화일은 KDB 트리와는 다른 공간 분할 방법을 사용하고 있다. KDB 트리는 공간을 분할할 때 분할되는 부분 공간의 어느 한 차원을 기준으로 그 부분 공간만 분할하는 반면 그리드 화일은 그 부분 공간뿐만 아니라 초평면에 평행하는 축으로 다른 부분 공간까지도 분할한다. 이렇게 함으로써 불필요한 많은 부분 공간들이 새로이 생성되고 이러한 부분 공간은 시스템의 성능을 크게 저하시킬 수 있다.

2.3 클래스 (C3): 축에 평행한 2*n*개의 초평면

축에 평행하게 초평면이 구성될 수 있는 또 다른 색인 방법이 있다. 여기에서는 클래스 (C2)하고는 다르게, 초평면의 개수가 $2n$으로 상당히 많은 것을 알 수 있다. 초평면의 수가 많기 때문에 공간상의 점을 여러 부분 공간에 나누어 넣은 것은 효과적으로 할 수 있다. 하지만, 그러기 위한 입출력 시간과 계산 시간은 색인의 구성 시간을 크게 할 수 있기 때문에, 전체적인 시스템의 성능은 크게 향상되지 못한다.

클래스 (C3)에 속하는 색인 방법으로는 R트리와 이에 대한 변형으로써 R^+ 트리와 R^* 트리가 있다. R 트리는 원래 공간상의 점을 색인 하기 위한 기법이라기보다 공간상의 범위를 가지는 데이타 (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 2*n*개의 초평면을 단위로 트리를 구성하고 있다. 즉, 2차원의 경우 직사각형을 이루는 4개의 초평면으로 구성된 부분 공간상에 점들을 사상시키는 형태로 색인을 구성한다. 2*n*개의 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.

R^+ 트리는 R 트리와 비슷한 구조를 가지고 있으며, 초평면의 구성 또한 거의 비슷하다. 다만, R 트리가 겹치는 부분 공간을 허용하는 반면 R^+ 트리는 겹치는 부분 공간을 전혀 허용하지 않는다. 겹치는 부분 공간을 배제함으로써 검색에 있어서 상당한 이득을 얻을 수 있으나, 이렇게 부분 공간을 유지하기 위한 초평면의 관리와 그 알고리즘에 있어서 완벽하지 못하고 계산 시간 또한 R 트리와 비교해서 좋지 못하다는 단점을 가지고 있다.

R^* 트리는 근본적으로 R 트리와 같은 구조이며, 단지 초평면을 유지 관리하는 알고리즘이 다를 뿐이다. R 트리의 가장 큰 특징은 부분 공간들 간의 겹침을 최소한으로 유지한다는 것으로 R 트리와 기본적인 알고리즘은 동일하지만 부분 공간의 분할에 있어서는 많은 차이를 보인다. 현재 R 트리의 변형된 형태의 색인 기법들 중에서 가장 우수한 기법으로 볼 수 있으나, 저장 공간의 측면과 알고리즘의 복잡성을 볼 때, 이 기법은 공간상의 점에 대한 색인보다는 정지 화상과 같은 범위를 가지는 복잡한 데이타에 대한 색인으로서 가치를 부여할 수 있다.

2.4 클래스 (C4): 곡면의 초평면

초평면의 형태가 항상 평면일 필요는 없다고 생각할 때, 구성될 수 있는 색

인 기법으로 MAIN을 들 수 있다. 이 색인 기법은 공간상의 어느 한 곳을 원점으로 선택해서 그 곳으로 부터의 거리에 따라 여러 부분 공간으로 분할해 가는 것으로 어느 일정한 범위 거리 내에 있는 모든 점들을 포함하는 부분 공간을 구성해서 유지 관리하는 것이 이 색인 기법의 핵심이다. 다음은 원점으로부터 거리를 계산하기 위한 간단한 산술 식이다. 여기서 R은 각 키 애트리뷰트의 값, 즉 n차원 공간상의 각 축에 대한 값이 된다.

$$R^2 = R_1^2 + R_2^2 + \ldots + R_n^2$$

이 기법의 장점은 간단한 계산을 통해 부분 공간을 관리하는 것이 가능하다는 것인데, 이 계산 방법은 R의 값이 정수 혹은 소수점을 가지는 실수일 경우에 쉽게 적용할 수 있지만, R의 값이 문자열과 같은 수치가 아닌 경우는 적용하기 어려워진다. 따라서 실제 상황에 적용하기에 적합하지 못하다.

2.5 클래스 (C5): 일정치 않은 초평면

초평면의 벡터가 어느 축에도 평행하거나 직교하지 않은 경우, 이를 색인 기법으로 발전시킨 연구는 아직까지 없었다. 이 클래스의 경우 초평면은 하나로 구성되어 있고 그 벡터가 일정치 않기 때문에 초평면을 표현하는 것이 상당히 어렵다는 것이다. 이 때문에, 이를 색인 방법으로 발전시키기는 어렵다.

2.6 클래스 (C6): 일정치 않은 가변 개수의 초평면

초평면의 벡터뿐만 아니라 초평면의 개수 또한 가변인 경우 가장 좋은 색인 기법으로 생각될 수 있다. 왜냐하면, 가장 최적의 부분 공간을 구성할 수 있으며 불필요한 부분 공간이 최소화될 수 있기 때문이다. 그러나 클래스 (C5)처럼 초평면을 표현하기가 거의 불가능하고, 그 개수 또한 가변적이기 때문에 색인 기

법으로 발전시키는 것은 생각해 볼 수 없다. 이 클래스 또한 아직까지 연구된 바가 없는 것으로 다음에 기술할 클래스 (C7)로 대치될 수 있다.

2.7 클래스 (C7): 직교하는 초평면

초평면을 각각의 축에 대해 직교하도록 구성하게 되면 *MAX*와 같은 색인 방법을 유도할 수 있다. 이 클래스는 초평면의 차원이 n차원이고 개수 또한 n개로 구성되며, 축에 대해 직교하는 비교적 단순한 구조를 가지게 된다. 다음과 같은 그림을 가지고 생각을 해보면, 쉽게 짐작을 할 수 있다.

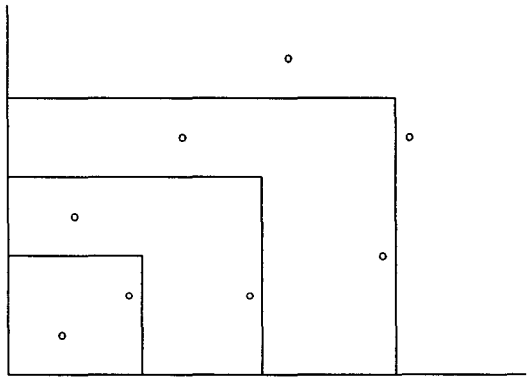

그림 1. *MAX*의 공간 분할 방식

그림 1을 보면 전체 공간은 직교하는 2개의 초평면에 의해서 4개의 부분 공간으로 구성되어 있다. 여기서는 2차원의 공간상에 8개의 점이 구성되어 있는 것을 가정하였으며, 각 부분 공간마다 2개씩의 점을 허용하는 것을 가정하였다.

2.8 다중 키 색인 방법을 위한 모델의 선택

2.1절에서 2.7절까지 다중 키 색인을 위한 기존의 기법과 아직 제안되지 않은 몇 개의 모델에 대해서 간단히 살펴본 결과 각각의 색인 기법을 위한 모델들

은 나름대로의 장점과 단점을 가지고 있으며, 어느 기법이 가장 우수하다고 말할 수는 없다. 이는 데이타의 분포와 양에 따라 달라 질 수 있으며, 그 환경에 따른 변화가 상당히 크다고 할 수 있다. 본 연구에서는 *MAX*를 구현하여 실험해 본 결과 데이타의 분포도나 양에 따른 변화가 가장 적은 것을 확인하였다. 이에 본 연구에서는 *MAX*를 다중 키 색인 기법으로 구현하여 저장 관리의 일부분을 담당하도록 한다. *MAX*를 구현하기 앞서 이에 대한 기본 연산과 데이타베이스를 검색, 갱신하기 위한 데이타 처리용 연산에 대한 알고리즘에 대해서 다음 절에 상세히 기술한다.

3. *MAX*를 위한 데이타 처리용 연산

*MAX*에 대한 기본 연산으로는 한 트리 노드 내에서 원하는 데이타를 찾기 위한 탐색 알고리즘(MABS: Multi-Attribute Binary Search)과 범람된 노드를 분할하는 노드 분할 알고리즘(RESP: Region Split), 수준 이하의 노드를 병합하는 병합 알고리즘(REMG: Region Merge)이 있다. 이들 알고리즘은 B 트리의 것을 그대로 사용해도 무방하며, 다만 키의 개수를 하나가 아닌 n개로 확장할 필요가 있다. 그 외의 나머지 것은 그대로 사용할 수 있다. 데이타 처리용 연산으로는 검색, 삽입, 삭제 연산이 있다. 이들 연산에 대해서는 3.1절에서 3.3절까지 상세히 기술한다.

3.1 검색 연산

검색 연산은 다중 키를 입력으로 받아서 그에 해당하는 데이타를 검색하는 것이다. 여기서 검색 대상은 다중 키의 하한선과 상한선 범위에 존재하는 모든 데이타를 말한다. 다음은 검색 알고리즘을 가상 프로그래밍 언어로 작성한 것이다. 알고리즘의 대부분이 B 트리와 흡사하므로 기존의 B 트리 위에서 구현하는 것은 별로 어렵지 않을 것이다.

Algorithm Multi-Attribute Search(Input: *root*, K; Output: *tuples*)

Input: A root of multi-attribute index tree, *root*
Given multi-attribute list (query) $K = (K_1, K_2, ..., K_n)$.
Let $(K_1, K_2, ..., K_n)$ be $R_n(K)$.

Output: Tuples that satisfy input query, *tuples*

Method:

MS1. [Search Root Node]
Read the root node from disk. If root is leaf then perform binary search with $R_n(K)$ within this node. Call the Binary Search algorithm with argument root and $R_n(K)$. If the binary search is successful then return results.

MS2. [Search Internal Node]

If *root* is internal node then search last location in the node where $R_n(K)$ is strictly-less-than hyper-region in the location. Let the location be i.

MS3. [Search Child]

Search the ith child node with the same search algorithm. Call Multi-Attribute Search recursively with child node.

End of Algorithm

3.2 삽입 연산

삽입 연산은 어느 한 노드에 데이타가 범람하는 경우에 그 노드를 분리하여, 트리를 계속 성장시키는 것이 핵심이다. 이러한 연산을 하기 위해서는 반드시 범람 노드 분할이라는 알고리즘이 필요한데, 이것은 B 트리의 분할 알고리즘과 거의 비슷하며, 다만 키의 개수를 여러 개로 확장 시켜 주는 것이 필요하다. 다음은 삽입 연산의 상세한 알고리즘이다.

Algorithm Multi-Attribute Insert(Input: *root*, K)

Input: A root of ***MAX***, *root*
Given multi-attribute list K = (K_1, K_2, ..., K_n).
Let (K_1, K_2, ..., K_n) be $R_n(K)$.

Output: None, the index structure is modified.

Method:

IN1. [Search Insert Location in Leaf Node]

Search insertion location of leaf node using Multi-Attribute Search algorithm. Let the location be R. R is a set of regions in node structure.

IN2. [Insert into Leaf Node]

Insert $R_n(K)$ into R. As a result from insertion, if the node is overflown then go to step IN3. Otherwise, overwrite the node R to the disk. Terminate insert algorithm.

IN3. [Split Node]

Split leaf node R into R_1 and R_2 using Split algorithm. Now, R_1 and R_2 is newly created and the R's entries is inserted. Write the node R_1 and R_2. Insert the disk location of R_1 and R_2 into the parent node of R by calling recursively Multi-Attribute Insert algorithm.

End of Algorithm

3.3 삭제 연산

삭제 알고리즘은 어느 한 노드 내에 데이타가 미리 정한 수준 이하로 떨어질 경우에 그 주변 노드와 병합해 나가며, 트리의 높이를 줄이는 알고리즘이 핵심이 되며, 노드 병합 알고리즘은 B 트리와 비슷하기 때문에 기존의 B 트리를 확장하는 것은 별로 어렵지 않다. 다만, 다중 키에서 삭제 연산의 경우 삭제되는

데이타는 미리 정해진 일부 범위가 된다는 것이 B 트리와 다른 점이다.

Algorithm Multi-Attribute Delete(Input: *root*, K)

Input: A root of MAX, *root*
Given multi-attribute list K = (K_1, K_2, ..., K_n).
Let (K_1, K_2, ..., K_n) be $R_n(K)$.

Output: None, the index structure is modified.

Method:

DL1. [Search Delete Location in Leaf Node]

Search deletion location of leaf node using Multi-Attribute Search algorithm. Let the location be R. R is a set of regions in node structure.

DL2. [Delete from Leaf Node]

Delete $R_n(K)$ from R. As a result from deletion, if the node is underflown then go to step DL3. Otherwise, overwrite the node R to the disk. Terminate delete algorithm.

DL3. [Merge Node]

Merge node R and adjacent node R' into the node R using Merge algorithm shown in previous section if merge is possible. Overwrite the node R to the disk. Delete from the location R' in the parent node of R by calling recursively Multi-Attribute Delete algorithm.

End of Algorithm

---

4. 결론 및 앞으로의 연구 방향

앞에서 설명한 다중 키를 위한 색인 방법들은 공간 객체의 좌표 값을 키로 하는 공간 객체용 색인으로 적용될 수 있다. 즉, 지도 데이타와 같은 2차원의 공

간 객체는 X축(위도)과 Y축(경도)에 의한 다중 키 데이타라고 할 수 있다. 또한 3차원의 공간 객체는 X축, Y축, Z축으로 이루어진 다중 키 데이타라고 할 수 있다. 이들 공간 객체는 그 표현 방법이 래스터 이미지나 벡터 형태로 될 수 있으며, 이 표현 방법은 공간 객체를 색인하는 것과는 무관하다. 다시 말해서, 색인 시스템과 공간 객체의 실제 저장 방식은 서로 독립적이라는 것이다. 그러므로 기존 시스템에서 어떠한 저장 방법을 사용하든지 그 시스템에 색인 방법을 추가로 구축하는 것은 그 시스템의 하부 구조에 큰 영향을 미치지 않는다. 특히 본 연구에서 제안한 MAX 색인 방법은 기존의 B 트리를 다중 키 형태로 변형하는 것에 불과하므로 구축 시간 또한 크게 줄일 수 있는 장점이 있다.

본 연구에서 제안한 다중 키 색인 방법이 GIS에 적용될 경우 GIS의 성능은 크게 향상될 것이며, MAX를 채용한 하부 저장 시스템은 일반적인 GIS의 하부 저장 시스템으로 자리잡을 수 있다. 다만 색인의 특성상 점 데이타에 한정되어 있고, 범위 데이타에 대해서는 적용할 수 없는 단점이 있으나 범위 데이타에 대해서는 이미 R^* 트리와 같은 상당히 좋은 기법이 제안되어 있으므로 이와 결합하여 시스템을 구축할 경우 상당히 성능이 우수한 GIS를 구축할 수 있다.

참고 문헌

[Beck 90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proceedings of ACM SIGMOD Conference, 1990, pp. 322-331.

[Bent 75] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," Communications of ACM, Vol. 18, No. 9, 1975, pp. 509-517.

[Bent 79] J. L. Bentley, "Multidimensional Binary Search Trees in Database Applications," IEEE Transactions on Software Engineering, Vol. SE-6, No. 4, 1979, pp. 333-340.

[Come 79] D. Comer, "The Ubiquitous B-Tree," ACM Computing Surveys, Vol. 11, No. 2, June 1979, pp. 121-137.

[Free 87] M. Freeston, "The BANG File: A New Kind of Grid File," Proceedings of ACM SIGMOD Conference, 1987, pp. 260-269.

[Gutt 84] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proceedings of ACM SIGMOD Conference, 1984, pp. 47-57.

[Kim 93] B. Kim and S. Moon, "A Flexible Indexing Structure Supporting Multi-Attribute Database Applications: MAIN," Journal of Microprocessing and Microprogramming, Vol. 39, 1993, pp. 177-180.

[Niev 84] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The Grid File: An Adaptable, Symmetric Multikey File Structure," ACM Transactions on Database Systems, Vol. 9, No. 1, March 1984, pp. 38-71.

[Oren 82] J. A. Orenstein, "Multidimensional Tries Used for Associative Searching," Information Processing Letters," Vol. 14, No. 4, 1982, pp. 150-157.

[Oren 86] J. A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System," Proceedings of ACM SIGMOD Conference, 1986, pp. 326-336.

[Otoo 84] E. J. Otoo, "A Mapping Function for the Directory of a Multidimensional Extendible Hashing," Proceedings of Very Large Databases Conference, 1984, pp. 493-506.

[Robi 81] J. T. Robinson, "The KDB-Tree: A Search Structure for Large Multidimensional Dynamic Indexes," Proceedings of ACM SIGMOD Conference, 1981, pp. 10-18.

[Sell 87] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R^+-Tree: A Dynamic Index for Multi-dimensional Objects," Proceedings of Very Large Databases Conference, 1987, pp. 507-518.