다차원 범위 질의를 위한 순차 색인 기법
(A Sequential Indexing Method for Multidimensional Range Queries)

차 광 호 †
(Guang-Ho Cha)

요 약 이 논문은 다차원 범위 질의를 위한 순차 색인 기법인 세그먼트-페이지 색인(SP-색인)이라는 새로운 색인 기법을 제안한다. SP-색인의 목표는 (1) 다차원 색인 기법에서의 범위 질의의 성능 향상, (2) 과도한 색인의 재구성 없이 색인의 클러스터링이라는 두 가지로 요약된다. 오랜 동안의 데이타베이스 연구 결과로 다양한 다차원 색인 기법이 개발 되었지만, 대부분의 연구가 데이타 레벨의 클러스터링에 초점을 맞추었고, 색인 자체의 클러스터링에는 거의 관심을 두지 않았다. 따라서 대부분의 관련된 색인 노드가 디스크에 분산되고, 질의 처리 시에 많은 무작위 디스크 접근이 발생한다. SP-색인은 관련된 노드를 연속적인 디스크 페이지로 구성되는 하나의 세그먼트에 저장하여 노드들의 분산을 피하고, 세그먼트 내에서의 순차 접근을 통해 질의 처리 성능을 높인다. 실험 결과에 따르면 SP-색인은 페이지 기반의 전통적인 색인 기법에 비해 수행 시간 면에서 수 배의 성능 향상을 보이고, 단순히 큰 페이지를 사용에 따른 디스크 대역폭 낭비를 줄인다.

키워드 : 다차원 색인, 순차 디스크 접근, 색인 클러스터링, SP-색인

Abstract This paper presents a new sequential indexing method called *segment-page indexing* (*SP-indexing*) for multidimensional range queries. The design objectives of SP-indexing are twofold: (1) improving the range query performance of multidimensional indexing methods (MIMs) and (2) providing a compromise between optimal index clustering and the full index reorganization overhead. Although more than ten years of database research has resulted in a great variety of MIMs, most efforts have focused on *data-level clustering* and there has been less attempt to cluster *indexes*. As a result, most relevant index nodes are widely scattered on a disk and many random disk accesses are required during the search. SP-indexing avoids such scattering by storing the relevant nodes contiguously in a *segment* that contains a sequence of contiguous disk pages and improves performance by offering sequential access within a segment. Experimental results demonstrate that SP-indexing improves query performance up to several times compared with traditional MIMs using small disk pages with respect to total elapsed time and it reduces waste of disk bandwidth due to the use of simple large pages.

Key words : multidimensional index, sequential disk access, index clustering, SP-indexing

1. 서 론

오랜 동안의 데이타베이스 연구에서 다양한 다차원 색인 기법(multidimensional indexing methods: MIM)이 개발되었고, 이들 MIM은 디스크의 페이지 구조를 고려하여 설계되었다. 그러나 이러한 페이지 기반의 MIM은 많은 작은 색인 페이지를 접근해야 하기 때문에 대용량 검색에 적합하지 못하다. 그러나 대용량 검색을 위해 단순히 페이지 크기를 확장하는 것은 점 질의(point query)같은 무작위 디스크 접근을 요구하는 질의나 저용량 질의의 경우엔 디스크 대역폭의 낭비가 심하다. 본 논문은 많은 페이지를 하나씩 무작위 접근하지 않고, 관련된 페이지들을 순차적으로 접근함으로써 질의 성능을 개선하는 색인 기법을 제시한다.

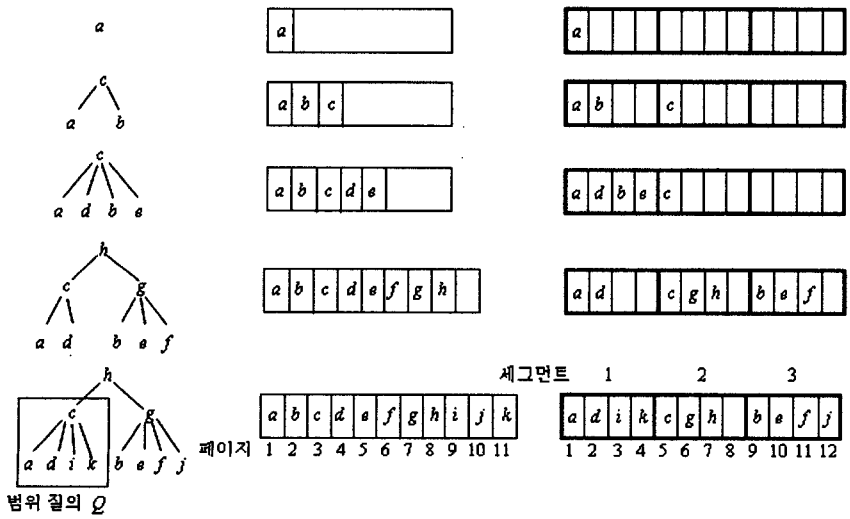
전통적인 4 KB의 색인 페이지는 큰 다차원 질의를 효율적으로 처리하기에는 너무 작다. 최근의 연구 [1]에 따르면, 현재의 장치에서는 8 KB에서 32 KB까지의 페이지 크기가 4 KB의 페이지보다 우월하다고 나와 있다.

· 본 연구는 숙명여자대학교 2004년도 교내연구비 지원에 의해 수행되었음
† 종신회원 : 숙명여자대학교 멀티미디어과학과
ghcha@sookmyung.ac.kr
논문접수 : 2004년 9월 16일
심사완료 : 2005년 5월 27일

그러나, 단순히 큰 페이지는 많은 디스크 대역폭을 낭비할 수 있다. 또한, 전통적인 MIM은 동적인 페이지 할당으로 인해 색인 페이지들이 디스크 상에 산재함으로써 질의 처리 시에 많은 무작위 접근을 해야 한다. 많은 임의 디스크 접근으로 인한 성능 저하를 막기 위해서는 관련된 색인 노드들을 클러스터링해야 하지만 기존의 MIM은 데이타의 클러스터링만 고려했지 색인의 클러스터링을 고려하지 않았다. 또한, 동적 색인 클러스터링의 경우 색인 재구성에 드는 오버헤드가 매우 크다.

본 논문에서는 기존 MIM의 약점을 해결하기 위해 세그먼트-페이지 색인(segment-page indexing: SP-색인)이라는 새로운 색인 기법을 제안한다. SP-색인은 세그먼트(segment)의 개념에 기초한다. SP-색인은 하나의 디스크는 세그먼트들의 모임으로 분할된다고 간주한다. 각 세그먼트는 디스크 상에서 f 개의 연속적인 페이지들의 집합으로 구성되고, 이 세그먼트가 SP-색인의 클러스터링 단위이다. 한 세그먼트에 있는 모든 디스크 페이지는 한 번의 디스크 접근으로 읽을 수 있으므로 디스크 시작과 탐색 시간을 절약할 수 있다. SP-색인에서는 모든 디스크 페이지가 세그먼트 번호와 페이지 번호의 쌍으로 주소가 할당된다. 이 주소 할당으로 인해 세그먼트뿐만이 아니라 페이지도 디스크 접근의 단위로 사용될 수 있다. 즉, 무작위 접근이 필요하거나 질의의 범위가 작을 경우는 세그먼트가 아니라 페이지 기반의 디스크 접근이 가능하다.

그림 1은 전통적인 MIM과 SP-색인에서 색인 노드를 생성하고, 해당 디스크 페이지를 할당하는 색인 생성 과정을 나타내는 한 예이다. 간단히 하기 위해, 모든 노드의 크기는 같고, 중간(nonleaf) 노드는 자식을 가리키는 4 개의 링크를 가질 수 있고, 각 세그먼트는 4 개의 페이지로 구성된다고 가정한다. 그림 1(a)는 노드 분할 순서에 따라 차례로 색인 노드가 할당되어 색인 트리가 성장하는 모습을 나타낸다. 처음에, 하나의 루트 노드 a가 한 세그먼트의 하나의 페이지, 예를 들어 (세그먼트 1, 페이지 1)에 저장된다. 데이타베이스가 성장함에 따라, 노드 a가 범람하게 되면, 새로운 노드 b를 할당한 다음 범람 노드에 있던 데이타를 노드 a와 b에 분산시킨다. 이제 노드가 두 개가 되므로 부모 노드가 필요하게 되어 새로운 노드 c를 할당하고 그것을 노드 a와 b의 부모 노드로 만든다. 그림 1의 (b)와 (c)는 각각 전통적인 색인 기법에서의 페이지 기반 디스크 할당 모습과 SP-색인에서의 세그먼트 기반 디스크 할당 모습을 보여준다. 그림 1(b)에서는 관련된 색인 노드들이 디스크 상에 넓게 분포한다, 예를 들어 부모 노드로 c를 갖는 노드 a, d, i, k가 각각 페이지 1, 4, 9, 11에 분산되어 있다. 반면에, 그림 1(c)의 SP-색인에서는 관련된 노드 a, d, i, k가 세그먼트 1에 클러스터되어 있다. 이 예는 SP-색인이 한 세그먼트에 색인 노드들을 클러스터링하는 효과를 갖고, 클러스터된 노드에 대한 순차 접근을 제공할 수 있음을 보여준다. SP-색인은 중간 노드와 리프 노드를 별도의 세그먼트에 저장한다. 즉, SP-색인은 중간 노드와 리프 노드를 위한 두 종류의 세그먼트를 유지한다. 그림 1(a)에서 범위 질의 Q가 중간 노드 c가 점유하는 영역과 겹친다고 가정하면 검색 중에 노



(a) 색인 구조 (b) 페이지 기반 디스크 할당 (c) 세그먼트 기반 디스크할당

그림 1 색인 트리의 성장과 디스크 상에서의 페이지 및 세그먼트 할당

드 *h, c, a, d, i, k*를 차례로 접근해야 한다. 전통적인 색인 기법에서는 일반적으로 접근해야 하는 노드 수만큼의 무작위 디스크 접근이 필요하다. 그러나 만약 노드가 세그먼트에 클러스터되어 있다면 무작위 접근의 수는 줄어들 것이다. 앞의 예에서는 세그먼트 1, 2를 위해 두 번의 접근만 하면 된다. 무작위로 디스크를 접근하는 회수를 줄여서 디스크 입출력 비용을 줄이는 것이 SP-색인의 핵심이다.

2. 관련 연구

지금까지 순차 디스크 접근을 위한 몇 가지 기법들이 제안되어왔다. 그러나 대부분의 연구가 데이타 레벨의 클러스터링에 초점을 맞추었고, 색인 자체의 클러스터링이나 순차 접근에는 거의 관심을 두지 않았다. 그러나 실제로 색인의 크기도 데이타 자체만큼 크므로 색인에 대한 클러스터링도 중요하다.

Brinkhoff와 Kriegel[2]은 공간적으로 인접한 객체를 나타내는 데이타 페이지의 집합을 연속적인 디스크 페이지로 저장하는 전역(global) 클러스터링의 문제를 연구했다. SP-색인과 Brinkhoff와 Kriegel의 전역 클러스터링은 몇 가지 개념이 비슷하지만 설계 목적이 다르다. SP-색인의 초점은 색인 레벨에서의 클러스터링인 반면에 전역 클러스터링의 초점은 데이타 레벨에서의 클러스터링이다. 따라서, SP-색인 기법은 색인 구조의 설계에서 고려해야 하는 반면에, 전역 클러스터링은 색인 구조와 직접적인 관련이 없다.

가상 저장 접근 기법(Virtual Storage Access Method: VSAM)[3]은 키에 의해 직접 데이타를 접근하거나 키에 따라 정의된 순서로 차례로 데이타를 접근하도록 설계되었다. VSAM 데이타 집합은 각각이 여러 제어 간격(Control Intervals: CINVs)으로 구성되는 여러 제어 영역(Control Areas: CAs)으로 구성되고, 각 제어 간격은 많은 레코드로 구성된다. 한 제어 영역은 보통 디스크의 한 실린더가 된다. VSAM에서 레코드의 순차 검색은 제어 간격을 차례로 도출하여 수행할 수 있다. VSAM의 제어 영역과 제어 간격은 각각 SP-색인의 세그먼트와 페이지의 개념과 비슷하다. 그러나 전역 클러스터링의 경우처럼, 제어 영역과 제어 간격은 데이타에 관한 디스크 공간을 관리하는데 사용되고 SP-색인의 세그먼트와 페이지는 색인에 관한 개념이다.

세그먼트의 개념은, B-트리의 또 다른 형태로써 연속적인 페이지 집합을 수용하고 다중 페이지 접근을 허용하는 SB-트리[4]와 bounded disorder(BD) 접근 기법[5,6]의 다중 페이지 구역(multi-page block)의 개념과도 비슷하다. 그러나 다중 페이지 구역의 아이디어는 데이타 공간의 차원이 증가함에 따라 많은 디스크 대역폭을 낭비할 수 있기 때문에 다차원 색인 기법에는 사용되지 않았다. 그러나 데이타 공간의 차원이 높아질수록 밀도가 희박해져서 검색 공간이 크게 확장되므로 세그먼트 읽기와 같은 다중 페이지 디스크 접근은 실제로 더 필요하다. 또한 모든 데이타 객체와 색인 노드 간에 일차원적인 순서를 갖고 있는 B-트리에서 사용되는 다중 페이지 구역과 달리 MIM에서는 한 세그먼트 내의 색인 노드들이 일 차원적 순서를 갖고 있지 않아서 B-트리의 다중 페이지 구역보다 MIM에서는 세그먼트의 설계와 관리가 어렵다.

세그먼트의 개념은 X-트리[7]의 슈퍼노드(super-node)의 개념과도 비슷하다. 슈퍼노드는 일반 페이지의 크기를 확장한 노드이다. 작은 여러 개의 페이지로 구성되는 세그먼트와 달리 슈퍼노드는 중간 노드의 분할을 피하기 위해 설계된 가변 크기의 큰 노드이다. 따라서 X-트리에서는 점 질의나 범위 질의에 관계없이 항상 큰 슈퍼노드를 읽는다. 반면에 SP-색인에서는 질의의 형태에 따라 세그먼트와 페이지 단위로 선택적으로 읽을 수 있다. 또한 슈퍼노드는 효율적인 중간 노드 색인 구조를 유지하기 위해 색인 트리의 중간 노드에만 적용된다는 점에서 SP-색인과 다르다.

[8]의 적응적 색인 구조는 데이타 분포와 질의 분포를 고려하여 색인의 노드 크기를 가변적으로 사용하는 색인 기법이며, [9]의 DABS-트리 역시 가변적인 크기의 색인 노드를 사용하는 색인 구조이다. 이 두 방법은 순차 접근에 의한 효율적인 범위 질의 처리 성능과 적절한 동적 유지 비용의 타협을 위한 색인 페이지의 클러스터링이라는 본 논문의 목적과는 다소 다른 개발 목표를 갖고 있다. 이들은 최적화하고자 하는 비용 모델을 사용하여 노드(페이지) 크기를 결정하지만 노드의 동적 유지 비용이 크고, 디스크 분할(fragmentation) 정도가 크기 때문에 주기적으로 쓰레기 수집을 포함하는 디스크 재구성을 해야 한다.

[10]의 형제 노드 클러스터링의 개념은 본 논문의 SP-색인과 매우 비슷하며, 효율적인 검색 성능과 적절한 동적 유지 비용의 타협안이라는 동일한 개발 목표를 갖고 있다. 차이점은 SP-색인의 경우 각 세그먼트([10]의 경우 chunk) 내에 유지하는 색인 페이지가 세그먼트의 처음부터 연속적으로 채워진다. 따라서 디스크 접근 시에 전체 세그먼트를 다 읽을 필요 없이 세그먼트 내에 실제로 존재하는 페이지 수 만큼만 순차적으로 읽을 수 있어서 디스크 대역폭을 절약한다. 반면에 형제 노드 클러스터링의 chunk 내에 유지하는 노드들은 해당 chunk 내에 산재하기 때문에 순차 접근을 위해서는 항상 chunk 전체를 읽어야 하므로 디스크 대역폭의 낭비를 가져온다.

3. SP-색인 기법

SP-색인의 효과를 증명하기 위해 본 논문에서는 LSD-트리[11,12]에 SP-색인 기법을 적용하였고, 그렇게 생성된 색인 구조를 SP-트리라고 부른다. SP-트리는 d-차원의 점 데이타를 색인하기 위한 다차원 색인 구조이다. 그러나 실제로 SP-색인 기법은 LSD-트리, R-트리 등을 포함하여 대부분의 MIM에 적용할 수 있다.

3.1 SP-트리의 구조

SP-트리는 디스크가 세그먼트들의 모임으로 구성된다고 간주하고, 세그먼트를 중간 세그먼트와 리프 세그먼트로 구분한다. 중간 세그먼트는 색인 트리의 중간 노드를 수용하고 리프 세그먼트는 리프 노드를 포함한다. 두 종류의 세그먼트로 구분하는 것은 그것이 색인 구조의 설계를 단순화시키고 색인을 메모리에 상주시키기 용이하게 하기 때문이다. 본 논문에서 중간 세그먼트를 n-세그먼트, 리프 세그먼트를 l-세그먼트라 부른다.

각 세그먼트는 디스크 상에 f 개의 연속적인 페이지들의 집합으로 구성된다. 세그먼트의 첫 번째 페이지부터 마지막 페이지까지 1, 2, ..., f 의 번호를 붙인다. 세그먼트는 다음의 성질을 갖는다:

- 세그먼트는 디스크의 연속적인 페이지 상에 놓이는 f 개의 노드의 집합으로 구성된다. f 를 세그먼트의 팬아웃(fanout)이라 부른다.
- 한 세그먼트에 있는K, $1 \le K \le f$, 개의 노드는 세그먼트의 처음부터 연속적으로 채워진다.
- SP-트리는 세그먼트에서 한번에 전체 f 개의 노드를 읽는 것이 아니라 실제로 데이타가 존재하는 K 개의 노드만 읽는다. 따라서 디스크 대역폭을 절약한다.
- 세그먼트 번호와 페이지 번호의 쌍으로 하나의 노드를 직접 접근할 수도 있고, 세그먼트 번호를 통해 한 세그먼트 내의 연속적인 K 개의 페이지를 순차적으로 접근할 수도 있다.
- SP-트리의 모든 노드는 세그먼트 내에 존재한다.
- 리프 노드는 l-세그먼트에, 중간 노드는 n-세그먼트에 존재한다.

그림 2는 SP-트리에서 세그먼트 내의 페이지의 구성을 보여준다. 루트 노드는 레벨 1에 있고, 레벨 j-1은 레벨 j의 부모 레벨이다. 레벨 j-1에 있는 노드의 내부 구성은 점선 사각형 내부에 나타나 있고, 이것은 LSD-트리의 구성과 동일하다. SP-트리는 실제로 이진 트리이다. 중간 노드의 i-번째 요소는 <lptr i, dim i, pos i, rptr i> 형태를 취한다. 여기서 lptr i와rptr i는 하위 레벨에 있는 노드나 같은 레벨의 다른 요소를 가리키는 포인터이다. (dim i, pos i)는 분할하는 차원(split dimension)과 해당 차원에서의 분할 위치(split position)를 나타낸다. 예를 들어, 그림 3(b)는 데이타 공간이 첫 번째 차원의 위치 60, 즉, (분할 차원 = 1, 분할 위치 = 60)에서 둘로 분할됨을 나타낸다. 각 노드는 그 노드에 현재 있는 요소의 갯수를 나타내는 정보를 유지한다. 또한, 특정한 세그먼트 내에서 현재 사용되고 있는 페이지의 갯수를 나타내는 정보를 색인 헤더에 유지한다. 리프 노드는 색인된 데이타 객체에 대한 정보를 포함한다.

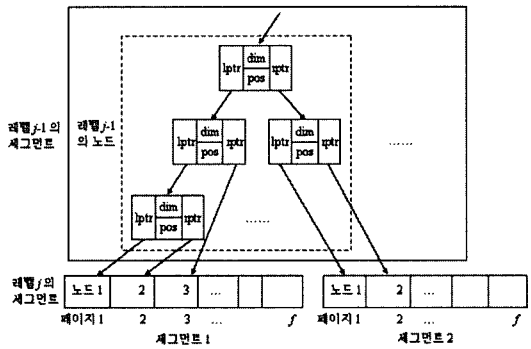


그림 2 세그먼트 내의 연속적인 페이지들

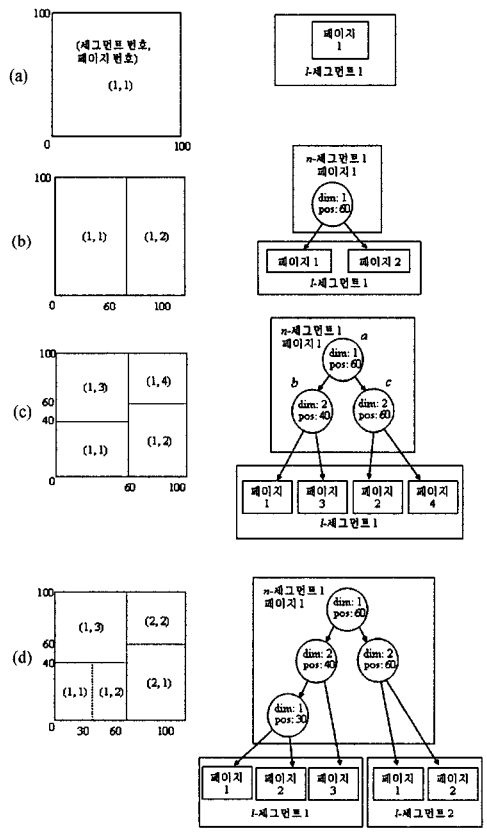


그림 3 SP-트리의 성장 과정

3.2 SP-트리의 생성

SP-트리는 계층적 구조를 갖고, 데이타 공간을 두 개의 중복 없는 셀로 분할한다. 각 셀에 대해 그 셀에 포함되는 모든 객체를 저장하는 하나의 데이타 페이지가 할당된다. 이러한 의미에서 본 논문에서는 셀을 디렉토리 영역이라 부른다. 그림 3의 연속적인 부분은 SP-트리가 성장하는 모습과 노드들이 세그먼트 내에서 클러스터링되는 모습을 보여준다. 첫번째 데이타가 삽입될 때 SP-트리의 첫번째 노드를 위해 l-세그먼트의 페이지 하나를 할당한다. 이 노드는 리프 이면서 루트가 된다. 그림 3의 왼편에 그림들은 2-차원 데이타 공간을 분할하는 과정을 보여준다. 각 차원은 0에서 100까지의 범위를 갖고 있다고 가정하고, 디렉토리 영역 상의 한 쌍의 수는 (l-세그먼트 번호, 페이지 번호)를 나타낸다. 새로운 데이타의 삽입이 노드 분할을 일으킬 때까지 이 노드에 데이타의 삽입이 계속되고, 분할이 일어날 때는 이 노드는 l-세그먼트 1의 페이지 1과 2를 점유하는 두 개의 리프 노드로 분할된다. 이 때, 두 리프노드의 부모 노드를 만들기 위해 n-세그먼트가 할당되고, n-세그먼트의 첫번째 페이지가 새로운 루트 노드로 지정된다. 이제 루트 노드는 하나의 분리자(separator)와 두 개의 포인터를 포함한다. 그림 3(b)에서, 분할은 차원 1의 위치 60에서 이루어졌다. 노드 분할이 일어날 때마다, SP-트리는 분할되는 노드를 포함하는 세그먼트에서 비어있는 페이지를 찾는다. 찾은 페이지는 그 세그먼트에 K 개의 페이지가 이미 존재할 때, K+1로 번호가 붙여진다. SP-트리는 색인 헤더에 각 세그먼트에서 현재 점유된 페이지의 갯수를 나타내는 정보를 유지한다. 만약 비어있는 페이지가 없다면, 세그먼트 분할이 일어난다. 새로운 세그먼트 S' 을 할당하고, 분리되는 노드를 포함하는 범람 세그먼트 S를 메모리로 읽어 들여서 전체 f +1 개의 노드를 두 개의 세그먼트 S와 S'에 분배한다.

3.3 세그먼트 분할

SP-트리를 LSD-트리와 구분하는 중요한 부분은 세그먼트 분할이다. 제일 먼저, SP-트리는 범람 세그먼트 S에 대해 (직접 또는 간접적으로) 루트 역할을 하는 중간 노드 u를 찾는다. 이 노드 u를 범람 세그먼트 S의 제어 노드라고 부른다. 제어 노드는 좌우 링크에 의해 하위의 세그먼트에 속하는 모든 페이지를 커버하고, 그 분리자는 세그먼트를 양분하기 위한 차원과 위치를 갖고 있다. 예를 들어, 그림 3(c)에서 만약 새로운 요소가 셀 (1, 1)에 삽입되고 그것이 l-세그먼트 1을 범람시킨다면, SP-트리는 l-세그먼트 1에 대한 제어 노드를 찾는다. 이 경우, 제어 노드는 n-세그먼트 1에 있는 노드 a 이다. SP-트리는 루트에서부터 새로운 데이타가 삽입되는 해당 노드까지의 접근 경로를 유지하므로 한 세그먼트의 제어 노드를 찾는 것은 간단하다. SP-트리의 루트에서부터 범람 세그먼트의 바로 위에 있는 부모 노드까지 찾은 다음, 현재의 중간 노드의 분리자의 분할 차원과 분할 위치에 의해 범람 세그먼트를 양분할 수 있는지 검사한다. 만약 그것이 가능하다면 해당 중간 노드를 그 세그먼트의 제어 노드로 선택할 수 있고, 그 세그먼트를 분리한다. 제어 노드의 오른쪽 자식에 속하는 페이지들은 새로운 세그먼트 S'의 처음 위치부터 다시 할당되고, 나머지 페이지들은 범람 세그먼트 S의 앞 부분의 페이지에 들어가도록 앞으로 이동시킨다. 이렇게 함으로써 같은 제어 노드 하의 페이지들은 같은 세그먼트 내에 모이게 된다.

그림 4는 중간 페이지 분할의 영향을 보여주고 있다. (a)에서, 하위 레벨 세그먼트 S'에서의 페이지 분리로 인해 새로운 색인 요소를 중간 페이지 P에 삽입하려고 한다. 페이지 P가 이미 꽉 찬 상태라면 새로운 데이타를 삽입할 때 P를 분할해야 한다. 페이지 P가 범람하면 SP-트리는 새로운 페이지 P'을 할당하고, 범람 페이지 P에서 지역(local) 루트 u를 제외하고 새로 들어온 색인 요소를 포함한 모든 요소들을 두 페이지 P와 P'에 분할한다. (b)에서 보듯이 분할 후에는 P에 있던 지역 루트 u는 부모 페이지로 올라가고, 두 페이지 P와 P'은 각각 u의 왼쪽과 오른쪽 부분트리에 속하던 요소들을 포함한다.

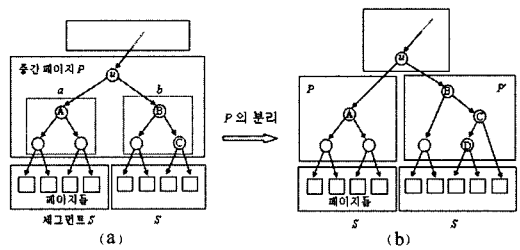


그림 4 중간 페이지 분할의 영향

SP-트리는 이진 트리이고, 중간 페이지에 있는 지역 루트 요소의 왼쪽과 오른쪽 부트리에 있는 요소들이 그들이 가리키는 페이지가 전부 같은 세그먼트에 있지 않는 한 각기 다른 세그먼트에 있는 페이지를 가리킨다. 예를 들면, (a)에서 부트리 a와 b에 있는 요소들은 각기 다른 세그먼트 S와 S'에 있는 페이지를 가리킨다. (b)에 나타난 것처럼, 페이지 P가 분리될 때 같은 세그먼트에 있는 페이지를 가리키는 색인 요소들은 같은 페이지에 들어간다. 따라서 SP-트리에서 한 세그먼트에 속하는 페이지들의 참조(references)는 동일한 중간 페이지에 저장된다.

SP-트리에서는 상위 레벨의 중간 페이지의 분할로 인해 현재 채워지지 않은 하위 레벨의 세그먼트가 분할되지 않는다. SP-트리에서는 중간 페이지의 팬아웃(보통 수 백)이 세그먼트의 팬아웃(보통 수 십)보다 훨씬 크다. 즉, 한 중간 색인 페이지 아래에 일반적으로 많은 세그먼트가 존재한다. 따라서 꽉 찬 상위 레벨의 중간 페이지의 제어 노드(즉, 지역 루트 노드)의 왼쪽과 오른쪽 포인터들이 가리키는 세그먼트들은 그림 4(a)에서 보듯이 이미 다른 세그먼트이다. 따라서 상위 레벨 페이지의 분할 뒤에 같은 중간 페이지에 있는 한 세그먼트에 속하는 페이지에 대한 참조들을 저장하기 위해 하위 레벨 세그먼트가 분할되는 경우는 없다.

4. 성능 평가

SP-색인 기법의 실제적인 효과를 입증하기 위해, 본 논문에서는 4-KB 페이지들로 구성된 128-KB 세그먼트를 사용하는 SP-트리를 구현하였다. SP-트리에 대해 광범위한 실험을 수행하였고, 그 결과를 다차원 색인 구조인 LSD-트리와 비교하였다. SP-트리는 SP-색인 기법이 적용된 LSD-트리이다. 그러나 본 실험에서 LSD-트리의 페이징(paging) 알고리즘은 구현하지 않았고 따라서 모든 색인 노드는 디스크에 저장된다. 모든 실험은 512 MB의 인텔 펜티엄 III 1.2 GHz 프로세서 상의 마이크로소프트 윈도우 XP 프로페셔널 파일 시스템 하에서 수행되었다. 모든 실험은 캐쉬하지 않은 디스크 상에서, 즉, 각각의 질의를 수행하기 전에 메모리 버퍼를 비운 상태에서 수행되었다.

4.1 실험 환경

실험을 위해 아래와 같은 두 종류의 합성 데이타 집합을 사용하였다:

- 무작위(random) 분포를 따르는 이차원 무작위 데이타 집합.
- Zipf의 법칙[13]에 따라 편향 분포를 따르는 이차원 편향(skewed) 데이타 집합: 본 논문에서는 [14]의 Zipf 분포를 사용하였다.
- IBM QBIC[15]의 13,724 개 이미지 집합: 이미지의 특성을 나타내는 특성 벡터를 얻기 위해 이미지 히스토그램의 세가지 통계학적 색상 모멘트를 사용하였다. 일반적으로 한 이미지의 히스토그램 빈(bin)이 포함하는 화소의 갯수는 대부분 매우 적고, 일부의 빈에 대부분의 화소 갯수가 집중된다. 따라서 본 실험에서는 가장 큰 빈도 수를 포함하는 32 개의 히스토그램 빈만 사용한다. 한 이미지 히스토그램의 세 개의 색상 모멘트는 다음과 같다:

$$\mu_i = \frac{1}{n}\sum_{j=1}^{k} f_{ij} v_{ij},\ i = 1,2,3$$

$$\sigma_i = \left(\frac{1}{n}\sum_{j=1}^{k} f_{ij}(v_{ij}-\mu_i)^2\right)^{\frac{1}{2}},\ i = 1,2,3$$

$$\alpha_i = \frac{\frac{1}{n}\sum_{j=1}^{k} f_{ij}(v_{ij}-\mu_i)^3}{\sigma_i^3}$$

위의 수식에서 첨자 i는 RGB 색상 모델의 i-번째 색상 요소, 즉, 빨강, 초록, 파랑 중 하나를 나타낸다. v_{ij}는 i-번째 색상 요소에서 j-번째 빈의 색상 요소의 값이다. f_{ij} 는 v_{ij}의 빈도수이다. k 는 전체 빈의 갯수, 즉, 32, 이다. n은 히스토그램에서 전체 화소의 갯수이다. 첫 번째 모멘트 μ_i는 각 색상 요소의 평균 농도를 정의한다. 두 번째 모멘트 σ_i는 상대적인 질감을 나타내는데 사용하는 대비를 나타내는 값이다. 세 번째 모멘트 α_i는 히스토그램의 편향도(skewness)를 나타내는 값이다. 따라서 사용하는 특성 벡터는 9 차원 데이타가 된다.

13,724 개의 이미지 특성 벡터는 본 연구의 실험을 위해서는 적은 수이므로 본 논문에서는 원래 이미지 특성 벡터 집합 고유의 데이타 분포를 유지하면서 100 만 개의 특성 벡터를 갖는 데이타 집합으로 확장하였다. 새로운 이미지 벡터 v를 생성하기 위해, 원래 데이타 집합에서 임의의 벡터 u를 선정하고 u가 속한 클러스터 c를 찾은 다음, 클러스터 c에서 두 개의 벡터를 임의로 선정하고 각 차원 i, $0 \le i \le 8$, 에서 두 벡터의 평균을 계산하고, 그 것을 새로운 벡터 v로 저장하였다.

각 데이타 집합은 중복 없이 100 만개의 데이타 점들로 구성된다. 모든 실험에서, 4 KB와 128 KB의 두 종류의 페이지를 사용하였고, 128-KB 세그먼트는 32 개의 4 KB 페이지들로 구성된다. 범의 질의의 성능을 측정하기 위해 다섯 그룹의 범위 질의를 생성하였다. 다섯 그룹의 범위의 모양은 정사각형 형태를 취하고, 그 크기는 각각 전체 데이타 공간의 0.001%, 0.01%, 0.1%, 1%, 10%를 차지하도록 하였으며, 그 중심은 데이타 공간에서 무작위 분포를 하도록 하였다. 각 실험에서 무작위로 생성한 100 개의 범위 질의를 수행하였고 그 결과를 평균하였다.

4.2 색인의 크기 및 저장 효율도

표 1은 무작위(random), Zipf, 실제(real) 데이타 집합에 대한 색인 파일의 크기를 보여준다. 표 1에서, SP(128), LSD(128), LSD(4)는 각각 4-KB 페이지들로 구성된 128-KB 세그먼트를 사용하는 SP-트리, 128-KB 페이지를 사용하는 LSD-트리, 4-KB 페이지를 사

표 1 SP-트리와 LSD-트리의 색인 화일 크기

	SP (128)	LSD (128)	LSD (4)
Random	18.75 MB	16.71 MB	16.64 MB
Zipf	20.04 MB	16.71 MB	16.83 MB
Real	76.88 MB	65.66 MB	56.56 MB

표 2 SP-트리와 LSD-트리의 저장 효율

	SP-트리 (128-KB 세그먼트와 4-KB 페이지)				LSD-트리 (128-KB 페이지)	
	페이지 저장 효율	세그먼트 저장 효율	4-KB 페이지의 갯수	128-KB 세그먼트의 갯수	페이지 저장 효율	128-KB 페이지의 갯수
Random	70.8 %	88.7 %	4143	146	70.9 %	129
Zipf	70.0 %	84.0 %	4191	156	70.9 %	129
Real	68.4 %	87.0 %	18512	665	59.5 %	514

용하는 LSD-트리를 나타낸다. SP-트리의 세그먼트가 채워지는 정도(fullness)가 100%가 아니기 때문에 SP-트리의 색인 파일 크기는 LSD-트리의 색인 화일 크기보다 다소 크다. 표 2는 SP-트리와 LSD-트리의 저장 효율도(storage utilization)를 보여준다. SP-트리의 경우, 모든 데이타 집합에서 세그먼트 저장 효율도가 80%를 넘고, 페이지 저장 효율도는 70% 부근이다. LSD-트리의 경우, 합성한 데이타 집합의 페이지 저장 효율도는 70% 부근이나, 실제 데이타 집합의 페이지 저장 효율도는 70%에 훨씬 못 미친다.

4.3 범위 질의

표 3, 4, 5는 각각 범위 질의의 크기가 전체 데이타 공간의 0.001% - 10%까지 일 때, 무작위(random), Zipf, 실제 데이타 집합에 대한 평균 디스크 접근 횟수를 나타낸다. 128-KB 세그먼트를 사용하는 SP-트리와 128-KB 페이지를 사용하는 LSD-트리의 디스크 접근 횟수가 4-KB 페이지를 사용하는 LSD-트리의 디스크 접근 횟수보다 훨씬 적다. 128-KB 세그먼트를 사용하는 SP-트리와 128-KB 페이지를 사용하는 LSD-트리의 디스크 접근 횟수에서는 큰 차이를 보이지 않는다.

그림 5-7은 SP-트리와 LSD-트리에 대한 질의 수행 시간 실험의 결과이다. 각 그래프에서 세로 축의 눈금은

표 3 무작위 데이타 집합에서의 평균 디스크 접근 횟수

	SP (128 KB)	LSD (128KB)	LSD (4KB)
0.001 %	3.09	3.09	3.38
0.01 %	3.22	3.23	4.67
0.1 %	4.05	3.79	11.13
1 %	6.65	6.61	57.30
10 %	24.45	23.45	460.65

표 4 Zipf 데이타 집합에서의 평균 디스크 접근 횟수

	SP (128 KB)	LSD (128KB)	LSD (4KB)
0.001 %	3.04	3.06	3.30
0.01 %	3.14	3.14	4.26
0.1 %	3.54	3.53	8.48
1 %	5.13	5.07	32.96
10 %	12.61	12.04	182.186

표 5 실제 데이타 집합에서의 평균 디스크 접근 횟수

	SP (128 KB)	LSD (128KB)	LSD (4KB)
0.001 %	4.44	4.45	24.16
0.01 %	4.79	4.78	36.06
0.1 %	6.04	5.99	78.49
1 %	11.47	10.41	267.19
10 %	49.93	41.46	1314.86

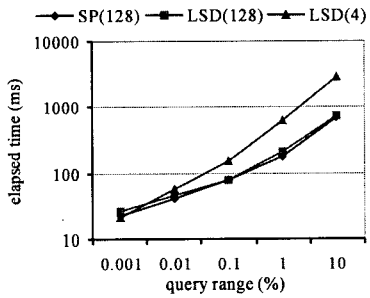


그림 5 무작위 데이타 집합에 대한 질의 성능

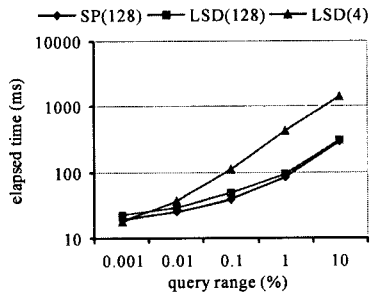


그림 6 편향 데이타 집합에 대한 질의 성능

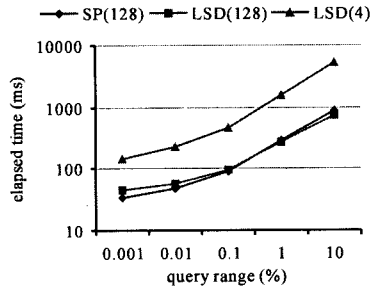


그림 7 실제 데이타 집합에 대한 질의 성능

로그 함수 형태이다. 대부분의 경우에서, 128-KB 세그먼트를 사용하는 SP-트리와 128-KB 페이지를 사용하는 LSD-트리의 성능이 4-KB 페이지를 사용하는 LSD-트리의 성능보다 훨씬 우수하다. 128-KB 세그먼트를 사용하는 SP-트리와 128-KB 페이지를 사용하는 LSD-트리를 비교할 때, 질의의 범의가 전체 데이타 공간의 10% 이상과 같이 아주 큰 범위의 질의를 제외하곤, SP-트리가 LSD-트리보다 약간 우수하다. 이 결과는 SP-트리가 큰 페이지를 사용하는 LSD-트리와 비교할 때, 디스크 대역폭을 절약하기 때문이다. 128-KB 페이지를 사용하는 LSD-트리의 성능이 SP-트리에 비해 크게 떨어지지는 않지만, 질의의 범위가 작거나 점(point) 질의 같은 무작위 질의의 경우에는 성능이 아주 나쁘다. 이것은 단순히 큰 페이지를 사용하는 것은 다양한 형태의 질의에 대해 만족할만한 성능을 나타낼 수 없음을 의미한다. 따라서 세그먼트와 페이지에 의한 디스크 접근을 둘 다 지원하여 상황에 따라 선택적으로 사용할 수 있도록 하는 것이 바람직하다.

4.4 점(무작위 접근) 질의

본 논문에서는 무작위 디스크 접근에서의 SP-색인의 성능을 검토하기 위해 점 질의 같은 무작위 접근 질의에 대한 실험을 수행하였다. 무작위 질의의 경우에 SP-트리는 페이지 기반의 디스크 접근을 수행한다. 그림 8은 점 질의에 대한 수행 시간 실험 결과를 나타낸다. 예상한대로, 128-KB 페이지를 사용하는 LSD-트리의 성능이 가장 나쁘고, 이것은 디스크 대역폭의 낭비가 심한 결과에 기인한다. 실험 결과에서 SP-트리와 4-KB 페이지를 사용하는 LSD-트리 간에 점 질의의 성능 차이는 거의 없음을 보여주는데, 이것은 SP-트리의 높이가 LSD-트리의 높이보다 크지 않음을 의미한다.

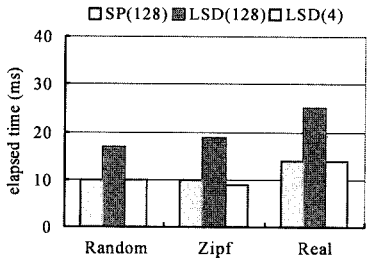


그림 8 점(무작위) 질의 성능

4.5 삽입 및 삭제

표 6은 SP-트리와 LSD-트리에서 삽입, 삭제 시의 평균 디스크 접근 횟수를 나타낸다. SP-트리의 경우, 삽입과 삭제는 4-KB 페이지 단위로 이루어 진다. 따라서, 128-KB 페이지를 사용하는 LSD-트리의 디스크 접근 횟수는 SP-트리와 4-KB 페이지를 사용하는 LSD-트리의 디스크 접근 횟수보다 적다. 그러나, 128-KB 페이지를 사용하는 LSD-트리의 경우, 삽입과 삭제에 드는 실제 처리 시간은 디스크 대역폭의 과도한 사용으로 인해 SP-트리와 4-KB 페이지를 사용하는 LSD-트리의 경우보다 훨씬 길다. 그림 9는 SP-트리와 128-KB 페이지를 사용하는 LSD-트리, 4-KB 페이지를 사용하는 LSD-트리에서의 색인 형성에 드는 실제 시간을 나타낸 것이다. 100 만개의 데이타를 비어있는 초기 색인 구조에 삽입하였다. SP-트리와 4-KB 페이지를 사용하는 LSD-트리를 형성하는데 걸린 시간은 거의 비슷하다. 반면에, 128-KB 페이지를 사용하는 LSD-트리의 형성 시간은 굉장히 큼을 알 수 있다. 이것은 삽입과 삭제에 드는 SP-트리의 추가 비용은 4-KB 페이지를 사용하는 LSD-트리에 비해 무시할 수 있을 정도로 작음을 알 수 있다.

표 6 삽입, 삭제 시의 평균 디스크 접근 횟수

	삽입			삭제		
	LSD (128KB)	LSD (4KB)	SP (128 KB)	LSD (128KB)	LSD (4KB)	SP (128 KB)
무작위	3.02	3.94	3.94	3.01	3.93	3.93
Zipf	3.01	3.95	3.95	3.01	3.92	3.92
실제	3.03	4.01	4.01	3.02	3.99	3.99

5. 결 론

본 논문에서는 색인의 클러스터링을 통해 다차원 색인에서의 범위 질의의 성능을 높이면서도 점 질의 같은 임의 접근 질의의 성능은 그대로 유지하는 새로운 형태의 색인 기술인 SP-색인 기법을 제시하였다. 실험 결과를 통해서 SP-색인은 범위 질의의 성능에서 전통적인 색인 기법의 성능을 수 배까지 향상시킬 수 있음을 보였고, SP-색인의 형성과 관리에 드는 오버헤드는 전통적인 색인과 거의 차이가 없음을 확인하였다. 이것은 다수의 페이지에 대한 순차 접근을 통해 디스크의 접근 시작 시간을 줄이고, 데이타의 삽입과 삭제에는 페이지 단위의 접근을 하는데 기인한다. 아울러, 점 질의와 같은 무작위 접근 질의의 경우에도 페이지 단위로 디스크 접근이 가능함으로써 전통적인 색인과 성능 차이가 없음을 확인하였다. 또한, 범위 질의의 성능 향상을 위해 단순히 큰 페이지를 사용하는 색인의 경우에는 디스크 대역폭의 많은 낭비로 인해 작은 크기의 범위 질의나 점 질의와 같은 무작위 질의, 또 색인 형성과 관리에 드는 비용이 매우 큼을 볼 수 있었다. 아울러, SP-색인은 하나의 세그먼트 내에 다수의 색인 페이지를 연속적으로 저장함으로써 최적의 색인 클러스터링을 수행하는

부담과 온라인으로 색인을 재구성하는 오버헤드를 줄일 수 있다. 따라서, 많은 다차원 색인 기법의 설계 시에 SP-색인 기법을 적용하면 색인의 성능 향상을 기대할 수 있고, 최적의 색인 클러스터링에 대한 간단한 대안이 될 수 있을 것이다.

참 고 문 헌

[1] Gray, J. and Graefe, G., "The Five-Minute Rule Ten Years Later, and Other Computer Storage Rules of Thumb," *ACM SIGMOD Record*, Vol. 26, No. 4, pp. 63-68, 1997.

[2] Brinkhoff, T. and Kriegel, H.-P., "The Impact of Global Clustering on Spatial Database Systems," *Proc. of VLDB Conf.*, pp. 168-179, 1994, Morgan Kaufmann Pub., Orlando, USA.

[3] Keehn, D.G. and Lacy, J.O., "VSAM data set design parameters," *IBM Systems Journal*, Vol. 3, pp. 186-212, 1974.

[4] O'Neil, P.E., "The SB-tree: An Index-Sequential Structure for High-Performance Sequential Access," *Acta Informatica*, Vol. 29, p. 241-265, 1992.

[5] Litwin, W. and Lomet, D.B., "The Bounded Disorder Access Method," *Proc. of ICDE*, pp. 38-48, 1986.

[6] Lomet, D.B., "A Simple Bounded Disorder File Organization with Good Performance," *ACM TODS*, Vol. 13, No. 4, 525-551, 1988.

[7] Berchtold, S., Keim, D.A., and Kriegel, H.-P., "The X-tree: An Index Structure for High-Dimensional Data," *Proc. of VLDB Conf.*, pp. 28-39, 1996.

[8] Tao, Y. and Papadias, D. "Adaptive Index Structures," *Proc. of VLDB Conf.*, pp. 418-429, 2002.

[9] Böhm, C. and Kriegel, H.-P., "Dynamically Optimizing High-Dimensional Index Structures," *Proc. of EDBT Conf.*, pp. 36-50, 2000.

[10] 김기홍, 차상균, "효율적 공간 질의 처리를 위한 트리 구조 공간 색인의 형제 노드 클러스터링", 정보과학회 논문지(B), 제26권 제4호, pp. 487-499, 1999. 4.

[11] Henrich, A., "The LSD^h-tree: An Access Structure for Feature Vectors," *Proc. of ICDE*, pp. 362-369, 1998.

[12] Henrich, A., Six, H.-W., and Widmayer, P. "The LSD-tree: spatial access to multidimensional point and non-point objects," *Proc. of ICDE*, pp. 44-53, 1989.

[13] Knuth, D., *The Art of Computer Programming: Sorting and Searching*, Vol. 3, 1973, Addison Wesley, MA, USA.

[14] Lee, J.-H., Kim, D.-H., and Chung, C.-W., "Multidimensional Selectivity Estimation Using Compressed Histogram Information," *Proc. of ACM SIGMOD*, pp. 205-214, 1999.

[15] Flickner, M. et al., "Query by image and video content: the QBIC system," *IEEE Computer*, vol. 28, pp. 23-32, 1995.

차 광 호

1984년 부산대학교 계산통계학과 졸업(학사). 1989년 한국과학기술원 전산학과 졸업(석사). 1997년 한국과학기술원 정보및통신공학과 졸업(박사). 1986년~1987년 삼성전자(연구원). 1989년~1997년 데이콤(선임연구원). 1997년~2002년 동명정보대학교(조교수). 1999년~2000년 IBM Almaden Research Center(visiting scientist). 2002년~현재 숙명여대(부교수). 관심분야는 멀티미디어 데이타베이스, 바이오 정보시스템, 스트림 데이타베이스