韓國 컴퓨터情報學會 論文誌
第11卷 第1號, 2006. 3.

2006-11-1-1-11

준근사를 이용한 공간 인덱스 압축 기법

김 종 완*

A Spatial Index Compression Scheme Using Semi-Approximation

Jong-Wan Kim*

요 약

수 년간 GIS가 발전하면서 위치 속성을 갖는 공간 데이터에 대한 인덱스 연구가 활발히 진행되어 왔다. 특히, R-tree기반의 인덱스들이 많이 연구되어 왔으며, 주된 이슈는 데이터 검색 성능의 향상이다. 본 논문에서는 공간 데이터에 대한 검색성능 향상을 위해 R-tree의 키 값을 압축하는 준근사(Semi-Approximation) 기법을 제안한다. 이 기법의 기본적인 아이디어는 위치 정보를 포함하는 2-차원 공간 데이터에 대한 인덱스를 압축하여 데이터 검색 성능을 향상시키는 것이다. 이 기법은 MBR의 시작 좌표를 상대좌표로 압축하고 끝 좌표는 전체 탐색영역에 대한 양자화(Quantization)를 통해 계산함으로써 MBR의 확장을 QMBR(Quantization of MBR)의 반으로 줄임으로써 노드의 공간 이용률을 높이고 전체적인 탐색 성능을 향상시킨다. 기존에도 인덱스 크기를 줄임으로써 탐색 성능을 향상시키는 방법이 있었지만 본 논문과 같이 양자화의 확장공간을 반으로 축소시키는 연구는 처음이다. 성능평가는 실제 공간 데이터를 기반으로 진행하였으며, 실험결과는 SA 기법이 MBR을 압축하는 기존의 연구보다 향상된 성능을 나타낸다.

Abstract

Over the last several years, studies on spatial index have increased in proportion to the increase in the spatial data. Most of these studies, however, were on the indices based on R-tree by adding or changing some options, and there are a few studies on how to increase the search performance of the spatial data by compressing an MBR. This study was conducted in order to propose a new MBR compression scheme, SA(Semi-Approximation). The basic idea of this paper is the compression of MBRs in a spatial index. Since SA decreases the keys of MBRs, the enlargements of QMBR in half and increases the utilization of nodes, the SA heightens the overall search performance. The study analyzes mathematically the number of node accesses in a 2D space and evaluates the performance of the SA using the real data on location information. The results show that the proposed scheme has increased performance, higher than that of the pre-established schemes on compression of MBR.

▸ Keyword : 공간 인덱스(Spatial Index), 인덱스 압축(Index Compression), 양자화(Quantization)

• 제1저자 : 김종완
• 접수일 : 2006.01.11, 심사완료일 : 2006.02.13
* 고려대학교 컴퓨터학과 박사과정 수료
※ 이 논문은 2005년도 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2005-041-D00665)

I. 서 론

공간 데이터베이스 시스템은 공간 객체를 관리하기 위해 많은 메모리 공간과 처리 시간(processing time)을 필요로 한다. 공간 질의를 효과적으로 처리하기 위해서는 적합한 공간 인덱스와 질의 처리 기술이 있어야 한다. 대부분의 공간 데이터 인덱싱은 일정한 공간 안에서의 가장 작은 값과 큰 값을 범위로 하여 한 영역을 가정하고 객체들을 MBR, MBS와 같은 최소영역으로 인덱싱(indexing)한다.

수 년간 공간 데이터가 증가하면서 공간데이터 인덱스에 대한 연구가 활발히 진행되어 왔으며 특히, R-tree기반의 인덱스들이 많이 연구되어 왔지만 인덱스 크기를 줄임으로써 성능을 향상시키는 방법은 많지 않다. 특히, 본 논문과 같이 양자화에서 나타나는 공간을 반으로 축소시키는 연구는 처음이다.

논문에서는 기존 연구와 구별되는 공간 데이터의 키 값을 압축하는 준근사(Semi-Approximation) 기법을 제안한다. 이 기법의 기본적인 아이디어는 2차원 공간 인덱스에서 MBR을 관리하기 위한 키 값이 인덱스의 대부분을 차지하므로 데이터 객체에 대한 MBR의 좌표를 상대 좌표와 양자화(quantization)를 통해 압축하는 것이다. 이렇게 함으로써 MBR의 확장을 QMBR의 반으로 줄이고 노드의 공간 이용률을 높게 하여 전체적인 탐색성능을 향상시킨다.

논문에서 제안한 SA 기법은 MBR을 구성하는 키를 압축하여 한 노드에 들어가는 엔트리의 수를 증가시키며, 트리의 높이와 질의처리를 위한 노드 방문횟수를 줄인다. 노드 방문횟수의 감소는 디스크I/O의 감소를 가져옴으로써 보다 빠른 질의 결과를 얻게 된다. 인덱스 구조는 각 노드에 하위 노드들에 대한 전체 MBR정보를 갖게 하고 한 노드의 엔트리에는 압축된 MBR정보를 저장한다.

논문에서는 2차원 공간에서의 노드 접근 횟수를 수학적으로 분석하고 위치정보에 대한 실제 데이터를 사용하여 SA 기법의 성능을 분석하였다. 실험결과는 제안 기법이 MBR을 압축하는 기존의 연구보다 성능이 향상되었음을 나타낸다.

논문의 구성은 다음과 같다. 2절에서는 MBR압축 기법에 대한 관련연구를 설명한다. 3절에서는 기존 MBR압축기법과 비교하여 본 논문에서 제안하는 준근사(SA) 기법을 설명하며, 4절에서는 SA 기법에 대한 분석과 함께 실제 데이터를 기반으로 한 다양한 실험결과 통하여 SA가 기존 MBR압축 방법보다 뛰어난 성능이 있음을 보인다. 마지막으로 결론과 향후연구 방향을 5절에서 제시한다.

II. 관련연구

공간 인덱스 구조는 지리적 객체의 위치를 인덱싱 하는데 이용되며, 지금까지 공간 데이터를 관리하기 위해 많은 인덱스가 연구되어왔다[1],[2]. 공간 데이터 인덱싱을 위한 연구는 R-tree[3]를 기반으로 하여 기존 인덱스에 옵션을 설정함으로써 검색 성능을 향상시켰다. 예를 들어, X-tree[4]는 슈퍼노드(super node) 개념을 추가하였으며, SR-tree[5]는 MBR과 MBS(minimum bounding sphere)를 사용하여 인덱스를 구축한다.

다차원 인덱스, 특히 2차원 데이터를 관리하는R-tree의 경우, 인덱스에서 MBR에 대한 정보인 키 값이 전체 인덱스의 80%[5] 정도를 차지하므로 키 값을 압축하면 한 노드에 저장되는 엔트리 수가 증가하며, 검색 성능도 향상된다. 노드의 엔트리가 증가하면 인덱스의 높이가 낮아지며, 이 방법은 차원이 증가할 수록 더 많은 공간이 절약 된다. 공간 인덱스에서 MBR압축 기법을 사용하여 인덱스의 크기를 줄이는 연구로는 상대적 좌표를 이용한 RMBR (Relative representation of MBR), HMBR(Hybrid representation of MBR)[6]과 탐색공간에 대한 양자화에 기반을 둔 QMBR(Quantized representation of MBR)[7], VBR (Virtual Bounding Rectangle)[8]이 있다. 이 연구들의 공통적인 특징은 MBR의 키 값을 줄이는 것이며, 전자는 탐색영역의 특정 좌표로부터 해당 MBR의 변위(offset)을 계산하고, 후자는 탐색영역을 일정한 값에 따라 격자모양으로 분할하는 양자화를 이용한다.

VBR의 개념은 A-tree[8]에서 제안되었으며 이는 QMBR과 같이 탐색영역(search space)를 양자화하여 키 값의 저장단위를 줄여 저장한다. 즉, MBR을 가장 가까운 양자화 단위까지 늘려서 VBR을 구성한다. 그러나 이와 같은 QMBR계열은 실제 MBR이 양자화만큼 커지게 됨으로써 탐색 공간이 늘어나게 되어 객체를 검색할 때 실제 MBR보

다 검색 영역이 증가하게 된다. 결과적으로 다른 MBR과 중복(overlap)이 발생하며 노드 방문 횟수가 증가하여 전체적인 검색 성능이 떨어진다.

RMBR과 HMBR은 키 값을 검색공간의 시작 좌표로부터 상대적인 거리(offset)로 계산하여 키 값이 보다 작은 바이트에 저장되도록 한다. RMBR은 16바이트에 저장되는 키 값이 8바이트로 감소되고 HMBR은 6바이트로 더 줄어든다. 그러나 HMBR의 경우 QMBR의 4바이트보다 더 증가하는 단점이 있다.

III. MBR압축기법

공간 데이터에 대한 객체의 표현은 각 객체를 대표하는 좌표로 구성되며, 2차원 데이터의 경우 왼쪽 아래 좌표와 오른쪽 위 좌표로 구성된 MBR로 표현한다. 이러한 키 값은 인덱스 구조의 대부분을 차지하고 있기 때문에 압축을 통해 각 노드 엔트리의 크기를 줄임으로써 한 노드에 더 많은 엔트리가 저장되도록 한다. 본 절에서는 MBR에 대한 압축 기법인 RMBR, HMBR, QMBR을 소개하고, 논문에서 제안하는 SA기법에 대해 설명한다.

3.1 RMBR, HMBR, QMBR

MBR은 객체 또는 한 객체와 가까운 객체들로 구성되는 가장 작은 사각형이며, 두 대각의 점으로 표현된다. RMBR은 탐색공간을 기준으로 각 엔트리에 저장된 MBR에 대한 상대거리를 계산하여 키 값을 압축한다. 보통 MBR의 한 좌표는 4바이트에 저장되므로 (그림 1)의 R0와 같이 16바이트를 차지한다. 노드에 포함된 각 엔트리의 MBR 키 값을 상대거리로 저장하면 8바이트의 공간을 절약하게 된다. R2의 좌표를 전체 공간에 대한 상대거리로 계산하면 한 좌표의 저장공간은 2바이트로 감소한다. 즉, 노드의 공간 이용률은 더욱 향상된다.

HMBR은 끝좌표를 구할 때 전체 탐색공간을 기준으로 하지 않고 동일한 MBR의 시작점을 기준으로 하여 폭과 높이를 계산한다. 이는 전체 탐색공간의 시작점으로부터 계산된 상대 좌표보다 더 작은 값으로 나타나며 저장공간의 절약을 가져온다. 그러나 RMBR과 HMBR의 경우 키값이 여전히 두 자리 정수의 큰 값으로 나타난다.

키값을 더 작은 정수로 표현하기 위해 탐색공간을 일정한 수로 나누어 n개로 분할(quantization)된 격자(grid) 공간을 이용하여 키 값을 압축하는 기법이 QMBR이다. 이 기법은 저장 공간이 RMBR보다 더 절약된다. 이때, 정수 q는 양자화 레벨이며, MBR을 양자화 단위와 일치하도록 x축, y축 방향으로 확장하여 키 값을 양자화 단위로 대체하면 매우 작은 값으로 MBR을 유지할 수 있다. (그림 3)은 x축, y축을 각각 16×16으로 양자화한 결과를 나타낸다. 양자화 레벨이 256보다 작으면 각 좌표는 1바이트에 저장된다. (그림 1)과 (그림 2)의 결과 키 값을 압축하여 저장할 수 있지만 RMBR, HMBR은 8바이트, 6바이트의 저장 공간이 필요하며 특히, (그림 3)의 QMBR에서는 실제 MBR은 중복되지 않았으나 MBR의 확장에 의해 중복이 발생하여 검색 성능에 영향을 준다. 〈표 1〉은 각 기법의 크기와 특징을 나타낸다.

표 1. 압축기법의 MBR 크기
Table 1. The size of MBR compression schemes

객체	표현방식	크기	특징
MBR	객체의 두 점	16	압축 안함
RMBR	탐색공간의 상대좌표	8	MBR/2
HMBR	끝점은 높이와 폭	6	시작점이 RMBR과 동일
QMBR	탐색공간의 격자분할	4	MBR 확장됨

3.2 Semi-Approximation(SA) of MBR

다차원 공간을 표현하기 위한 n-차원의 사각형은 2n-차원 개의 좌표로 나타낼 수 있으므로 각 좌표를 압축하면 인덱스 저장공간의 절약과 함께 탐색 성능의 향상을 가져온다 [9]. 2차원의 경우 MBR을 구성하는 4개의 좌표로 표현된다. MBR의 준근사 기법에서는 각 노드가 자신의 엔트리들에 대한 전체 MBR을 저장하고 있다. n차원 공간에 대한 MBR은 두 개의 끝점은 (α, β)로 나타낼 수 있으며, 차원에 따라 α = (α1, α2, α3, ···, αn), β = (β1, β2, β3, ···, βn)으로 나타낸다. 이때, $\alpha i \le \beta i$ 이며, $i \in \{1, 2, 3, \cdots, n\}$이다. SA의 기본 아이디어는 α를 상대적인 값으로 계산 하고 β만 양자화로 함으로써 α, β를 모두 양자화하는 QMBR에서 나타나는 거짓탐색영역(false-search region) - 실제 객체가 포함된 영역을 확장하기 때문에 발생하는 객체가 존재하지 않거나 다른 MBR과 겹치는 영역 - 을 반으로 줄임으로써 검색 성능을 향상시키는 것이다. 또한 키 값의 저장 공간을 최소화한다.

【정의 1】 MBR의 상대좌표 표현. 전체 검색공간에 대한 MBR을 M 이라 하면 M의 왼쪽 아래와 오른쪽 위의 두 좌표는 (M.lx, M.ly), (M.rx, M.ry)로 표현된다. 탐색공간에 포함된 한 엔트리 R2는 두 점 (α, β)로 구성되며, 시작점 α 에 대한 상대좌표는 다음과 같다.

$$RM(\alpha) = (|M.lx - \alpha.x|, |M.ly - \alpha.y|)$$

【정의 2】 MBR의 양자화 표현. 정의1을 사용하여 탐색공간 M의 두 점을 (Ms, Me)라 하고 q를 양자화 레벨이라 하자. R2의 끝점 β를 양자화하기 위한 계산 식은 다음과 같이 정의한다. 이때, Qe는 양자화된 MBR의 끝점이다.

$$Q_e(\beta) = \begin{cases} 1, & (\beta = M_s) \\ \lceil ((\beta - M_s)/(M_e - M_s)) \times q \rceil, & (otherwise) \end{cases}$$

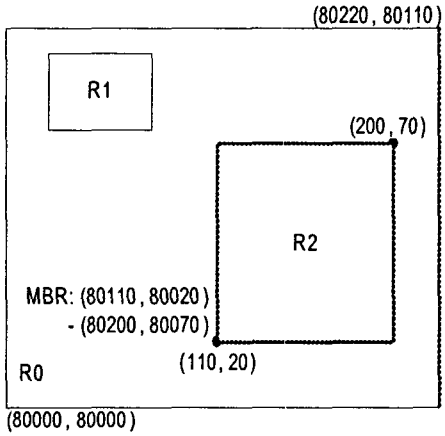


그림 1. 최소경계사각형(MBR)과 상대적 표현
Fig 1. MBR and Relative MBR

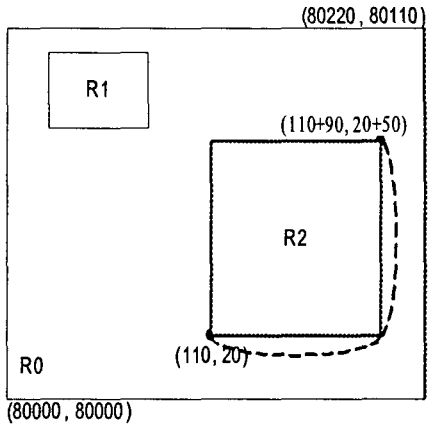


그림 2. MBR의 복합적 표현
Fig 2. Hybrid MBR

끝점인 β는 양자화 레벨에 의해 Qe 가 되며 비트로 변환하여 저장함으로써 저장 공간을 최소화한다. 양자화 레벨은 2n (n=0,2,3, …, 8)으로 하여 최대 1바이트로 표현된다. β는 양자화 레벨 q에 의해 정해지므로 특정 비트열(bit string)로 표현될 수 있다. 즉, 이진수 표현은 (Qe)2 이며 비트열의 길이는 $\lceil \log_2 q \rceil$이다. 끝점으로 계산된 결과는 (Qe - 1)2 로 저장한다. 예를 들면, (그림 4)에서 Qe(β.x)= 15, Qe(β.y)=12이므로 비트열은 두 점에 대한 비트를 연결한 11101011 이며, 결과적으로 R2의 키 값은 5바이트에 저장된다.

SA기법은 (그림 4)에서와 같이 확장되는 공간 즉, 거짓 탐색영역이 빗금 친 부분에만 나타나며, QMBR에서 나타나던 것에 비해 반으로 줄었다. 좌표의 저장공간도 최소 5바이트까지 감소한다. 이때, 양자화 레벨을 증가시키면 비트수도 함께 증가하여 키 값의 저장 바이트가 늘어나지만 여전히 SA의 장점은 유지된다.

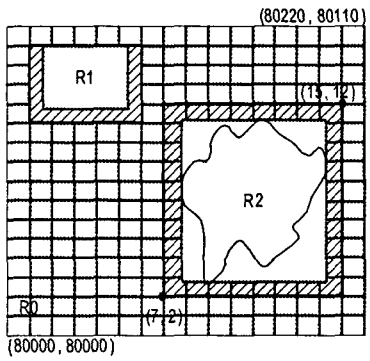


그림 3. MBR의 양자화 표현
Fig 3. Quantized MBR

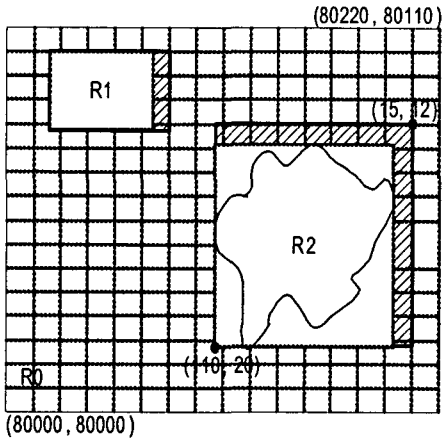


그림 4. MBR의 준근사
Fig 4. Seme-Approximative MBR

3.3 알고리즘

SA기법은 R-tree를 기본으로 하고 있기 때문에 이 절에서는 수정한 부분에 대해서만 설명한다. R-tree의 기본 알고리즘에서 수정이 필요한 부분은 삽입과 검색이다. 새로운 객체의 삽입은 알고리즘1과 같이 읽어 들인 객체를 각 노드에 있는 SA(MBR)과 비교하여 해당 노드를 찾아 삽입한다. 새로운 객체의 삽입은 루트 노드부터 단말 노드(leaf node)까지 이동하여 최소한의 확장을 유지하면서 객체를 포함할 수 있는 노드를 찾는다. 이때, 객체에 대해서도 SA(MBR)을 계산 하고 각 엔트리의 키 값과 비교 하여 삽입한다.

알고리즘1. 객체삽입(Insert): 노드와 객체를 입력받아 트리에 삽입하며, 좌표는 양자화 레벨 q를 인수로 하여 함수 SemiApp_makeMBR를 호출하여 변환한다.

```
-입력:Node n, Object o, QuantizationLevel q
1: Insert(n, o, q){
2:   if first time, Invoke
3:    SemiApp_makeMBR(entire_space,o, q_level);
4:   if(n==root) Compare o.MBR to MBR
5:                              of entire space;
6:    n = root.ptr;
7:  if(n==leaf node) Insert o into the node
8:                          and check overflow;
9:   Return to upper;
10:  Else Compare SA(o.MBR) with all entries;
11:   n = Entriy.ptr;
12:  Insert(n, o, q);
13: }
```

검색 알고리즘은 질의 구간인 Q의 끝점을 양자화하여 각 엔트리의 SA 키 값과 비교한다. 양자화는 알고리즘2의 함수 SemiApp_makeMBR을 호출하여 동일하게 처리된다. 장점은 질의를 변환함으로써 각 엔트리의 MBR값을 원래 값으로 변환하지 않은 채 모든 값과의 비교가 가능하다는 것이다. 따라서, 엔트리의 키 값을 변환하는데 소요되는 연산비용이 발생하지 않는다.

알고리즘2. MBR의 양자화(SemiApp_makeMBR): 객체의 MBR을 양자화 레벨에 맞춰 변환한다.

```
-입력: entireSpace M, object O, quant_level q_level
1: SemiApp_makeMBR(M, O, q_level){
2:   SAs = abs(Ms - Os);
3:   /* In detail, SA.lx = abs(M.lx-O.lx);
4:      SA.ly=abs(M.ly-O.ly); */
5:   If(Oe==Ms) SAe = 1;
6:   Else SAe= Ceiling(q_level * (Oe - Ms)
7:                                / (Me - Ms));
8:   Return SA(O.MBR) to upper;
9:    /* SA(O.MBR) = SAs + SAe */
10: }
```

IV. 분석 및 성능 평가

노드의 이용률은 노드의 크기를 늘리거나 MBR키값을 압축하여 높일 수 있으며, 노드 크기와 이용률을 조절하면 압축 효과는 더욱 증가한다. 본 절에서는 노드 크기, 이용률 그리고 압축에 의한 노드 접근 횟수를 분석 및 실험으로 평가한다.

4.1 노드 크기, 이용률 분석

MBR을 압축하면 트리에서 한 노드의 분기수(fan-out)가 증가하며, 트리의 높이에도 영향을 준다. 증가한 엔트리 수는 노드의 이용률을 높인다. 동일한 이용률을 가정하면 MBR과 RMBR은 2배의 차이가 나타나지만 양자화를 사용한 QMBR에서는 세배이상의 차이를 보인다. (그림 5)는 이용률이 증가할 때 한 노드에 들어가는 엔트리의 수를 분석한 결과이다. 각 기법에서 이용률을 변경하면 노드에 저장되는 엔트리가 지속적으로 증가하는 것을 볼 수 있다.

(그림 6)에서는 노드의 크기를 늘리면 저장되는 엔트리의 수가 늘어며, 키를 압축 하는 것도 노드크기와 함께 엔트리의 수에 영향을 끼치면서 급격하게 증가한다. 물론, (그림 5)와 (그림 6)에서 QMBR이 더 많은 엔트리를 포함하는 것으로 나타났으나 여기에는 QMBR이 갖는 거짓탐색영역을 고려하지 않은 것이며 단순히 노드크기와 이용률에 대한 엔트리의 증가를 보여준다. 거짓탐색영역에 대한 고려는 4.2절에서 다룬다.

노드의 분기수가 증가하면 트리의 높이가 줄어들지만 로그 단위(log scale)로 변하기 때문에 높이의 차이가 크지는 않다. 그러나 한 레벨이 낮아 지면서 나타나는 노드 방문횟수 및 검색 시간의 영향은 크다.

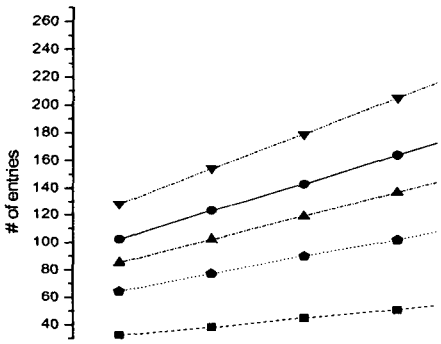


그림 5. 노드 이용률과 엔트리 수
Fig 5. Node utilization and entries

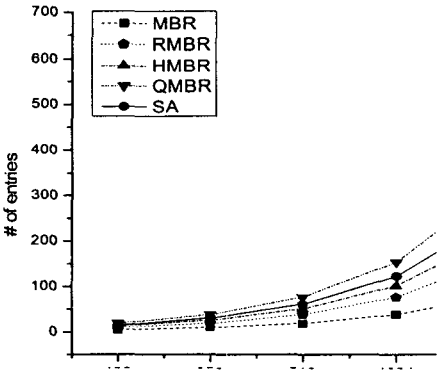


그림 6. 노드 크기와 엔트리 수
Fig 6. Node size and entries

4.2 노드 접근 회수 분석

R-tree에 대한 노드 접근횟수의 수학적 분석은 [5]에 나타난 것을 재 구성한다. 이때, 각 노드가 갖는 MBR이 같은 높이를 갖는다고 가정한다. 높이가 h 인 트리의 한 노드가 차지하는 평균 영역을 ah라하면 전체 노드들에 대한 평균 영역은 1/Mh로 나타낸다. 또한 트리에서 높이 h에 있는 한 노드가 질의영역과 겹칠 확률은 $\left(\sqrt[d]{s}+\sqrt[d]{a_h}\right)^d$ 이며, 이때, d는 차원, s는 질의 영역의 크기이다. 높이 h의 노드들이 질의 영역과 겹치게 되는 영역은 $M_h\left(\sqrt[d]{s}+\sqrt[d]{a_h}\right)^d$ 이며, 다음과 같이 나타낸다.(N: 전체 데이터 객체 수, f: 단말노드의 평균 분기 수(fan-out))

$$\left(1+\sqrt[d]{\left\lceil \frac{N}{f^h} \right\rceil}\cdot s\right)^d \quad \cdots\cdots (1)$$

R-tree의 루트 노드에서 부터 단말 노드까지의 전체 노드 접근횟수는 식(2)와 같이 각 높이에서의 노드들에 대한 합으로 구성된다.

$$1+\sum_{h=1}^{\lceil \log_f N \rceil -1}\left(1+\sqrt[d]{\left\lceil \frac{N}{f^h} \right\rceil}\cdot s\right)^d \quad \cdots\cdots (2)$$

이제 양자화 레벨 q를 적용하면 각 노드는 qd의 양자화 셀을 갖는다. QMBR에서 질의에 대

노드에 대한 접근은 높이 h의 한 노드에 접근한 후 하위 자식 노드에 접근하므로 노드 접근에 대한 확률은 $\left(\sqrt[d]{s}+\sqrt[d]{a_h}/q+\sqrt[d]{a_{h-1}}+\sqrt[d]{a_h}/q\right)^d$ 이다. 전체 노드에 대해 처리되므로 Mh를 곱하고 루트부터 단말 노드까지를 합하면 식(3)과 같다.

$$1+\sum_{h=1}^{\lceil \log_f N \rceil -1}\left(1+\sqrt[d]{\left\lceil \frac{N}{f^h} \right\rceil}\cdot s+\sqrt[d]{\left\lceil \frac{N}{f^{h+1}} \right\rceil}\cdot s/q\right)^d \quad \cdots (3)$$

식(3)은 MBR의 모든 면에 대한 확장을 고려한 것이며, SA에서는 확장이 반으로 감소하므로 식은 다음과 같이 수

정된다. 수식 의해 노드 접근횟수를 분석한 결과는 지면관계상 나타내지 않았으나 (그림 7(b))와 유사한 패턴을 보인다. 분석에서 객체 대상은 1백만 개, 질의 범위는 0.01%로 하고 각 엔트리의 포인터는 4바이트, 엔트리의 MBR크기는 16바이트로 계산했다. RMBR, HMBR, QMBR, SA의 키 값은 각각 8, 6, 4, 5바이트로 설정하였으며, 2차원을 가정하였다.

$$1+\sum_{h=1}^{\lceil \log_f N \rceil-1}\left(\left(1+d\sqrt{\left\lceil \frac{N}{f^h} \right\rceil}\cdot s+d\sqrt{\left\lceil \frac{N}{f^{h+1}} \right\rceil}\cdot s/q\right)/2\right)^d \quad \cdots\cdots (4)$$

4.3 실험 환경

논문에서 제안한 인덱스 구조의 실제적인 성능을 측정하기 위해 MBR과 기존의 압축기법인 RMBR, HMBR, QMBR 그리고 제안한 SA를 비교하였다. MBR은 2차원 인덱스인 R-tree를 실행하여 결과를 받아보았으며, 기존 압축방법들과 SA에서 제안한 알고리즘은 R-tree를 수정하여 구현하였다. 시험을 위한 하드웨어 사양은 펜티엄4 2.6GHz CPU, 1GB메모리이며 플랫폼은 윈도우 XP Professional 이다.

표 2. 실험 파라미터
Table 2. Experiment parameters

파라미터	단위	적용값
노드 크기	바이트	128, 256, 512, 1024
질의범위	%	5, 10, 15, 20, 25, 30
버퍼 크기	바이트	4K
양자화 단위	개	0, 8, 16, 64, 128, 256
노드 엔트리	개	200
데이터 크기	개	62,556개의 위치정보

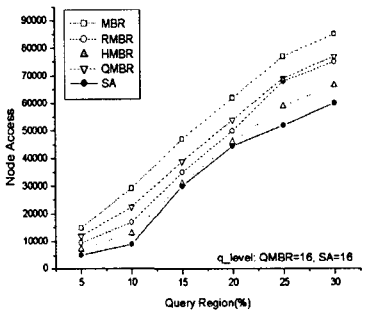
실험용 데이터는 62,556개의 캘리포니아 SEQUOIA[10]에 대한 위치정보를 사용했으며 수행 시간에 대한 성능 측정은 윈도우의 백그라운드 프로세스의 영향을 배제하고자 CSim시뮬레이터[11]를 사용하여 비주얼C++ IDE에서 실행하였다. 실험을 위한 파라미터는 〈표 2〉에 정리되어 있다. 질의 처리에 대한 성능측정은 전체 데이터의 일정범위에 해당하는 영역질의(region query)로 하였으며, 전체 검색공간에 대한 질의 영역의 비율은 5~30%로 하였다.

4.4 실험 결과

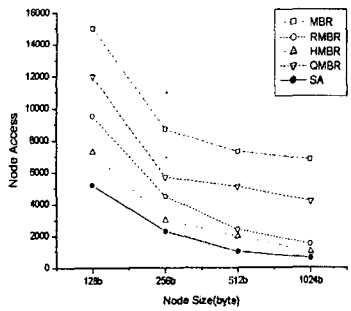
질의 영역에 대한 노드 접근 횟수 측정은 탐색영역에 대한 다양한 비율을 질의 범위로 지정하여 각 영역에 대해 서로 다른 1,000개의 질의를 수행하고 결과를 누적하여 수행하였다. (그림 7(a))에서 질의 영역(%)에 대해 압축하지 않은 MBR의 노드 접근횟수를 기준으로 압축기법들의 횟수가 모두 적게 나타나는데, 이는 MBR의 키 값을 줄임으로써 각 노드의 분기율이 높아지게 되기 때문이며, 결과적으로 노드 접근횟수가 줄어든다. QMBR과 SA 는 양자화 레벨을 모두 16으로 지정하였으며, 이는 SA가 저장할 수 있는 가장 작은 바이트 수이다. QMBR은 레벨이 낮아도 하나의 좌표를 2바이트에 저장하기 때문에 레벨을 동일하게 설정하였다. 실험에서 RMBR과 HMBR이 QMBR보다 좋은 결과를 갖는데, 이는 QMBR이 갖는 거짓탐색영역 때문이다.

노드의 크기를 변화시키면 MBR보다는 키 값을 압축한 기법들이 더 많은 엔트리를 갖게 되어 노드에 접근하는 횟수가 감소하게 된다. (그림 7(b))에서 SA는 키값이 6바이트를 차지하는 HMBR보다 엔트리를 더 포함하게 되어 노드 접근 횟수가 약간 감소하는 것을 볼 수 있다.

(그림 8)에서 보여주는 노드 크기에 따른 탐색시간은 노드 접근 횟수와 유사한 결과로 나타나고 있다. QMBR은 노드에 많은 엔트리가 들어갈수록 거짓탐색영역이 증가하므로 다른 기법보다 성능이 떨어진다. 반면에 RMBR, HMBR은 키 값 공간이 작아진 상태에서 엔트리가 증가하므로 QMBR보다 좋은 성능을 보여준다. 그 이유는 QMBR이 4바이트까지 키 값 크기를 줄였으나 양자화로 인해 발생한 거짓탐색영역 때문에 객체를 검색할 때 R-tree계열의 인덱스에서 나타나는 역진행(back tracking)이 발생하게 되어 실제 검색 시간이 증가하기 때문이다. 즉, 인덱스에서 저장 크기의 감소가 트리구조의 높이를 줄일 수 있지만 MBR의 중복으로 인한 노드 방문 횟수의 증가로 성능의 저하를 가져온다.



(a)



(b)

그림 7. 노드 접근 회수: (a) 질의영역, (b)노드크기
Fig 7. The number of node access: (a) Query region (b) Node size

(그림 9)는 QMBR과 SA를 양자화 크기를 조절하면서 두 기법의 노드 접근 횟수를 누적으로 산출했다. QMBR은 항상 1바이트씩 할당되므로 양자화 레벨에 영향을 적게 받는다. SA는 키 값이 5바이트에 저장되는 q=16에서 급격히 감소하고 점진적으로 증가하지만 QMBR보다 훨씬 적은 횟수를 보인다.

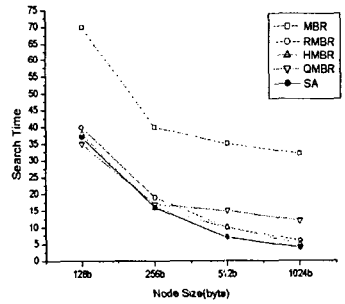


그림 8. 노드크기와 탐색시간
Fig 8. Node size and search time

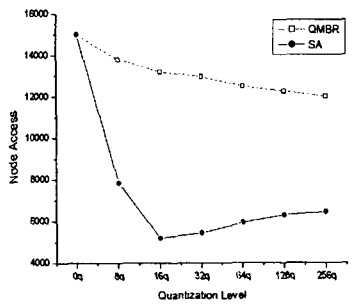


그림 9. 양자화 레벨과 노드접근 회수
Fig 9. Quantization level and node accesses

V. 결 론

논문에서는 공간 데이터베이스 시스템에서 공간 데이터에 대한 인덱스를 구축할 때 MBR압축 기법을 적용한 새로운 기법을 제안했다. 전통적인 공간 인덱스 구조에서는 MBR압축을 통해 인덱스를 구축한 사례가 드물다. 기존의 압축기법도 키 값의 저장공간을 원래 MBR보다 많이 줄였으나 논문에서는 더 작은 공간으로 줄였으며, QMBR이 갖는 확장영역을 반으로 줄여 탐색 성능을 향상시켰다.

본 논문에서는 새로운 MBR 압축기법인 SA를 소개하였으며, 이를 사용하여 MBR의 크기를 줄이는 새로운 시도를 하였다. 성능 평가는 SA기법에 대한 알고리즘을 구현하여 기존 압축방법과 비교하였다.

실험결과에 따르면 SA의 노드 접근 횟수는 질의영역, 노드 크기, 양자화 레벨을 변화시켜가면서 측정한 결과 기존 R-tree와는 명확히 구별되는 결과를 보인다. HMBR과는 근소한 차이를 보이기는 하지만 향상된 결과를 갖는다. 양자화 레벨에 따른 평가에서는 양자화를 사용하는 QMBR과 비교하였으며, QMBR이 갖는 확장공간의 차이로 인해 레벨 16에서 급격한 차이를 보였다. 질의 처리시간도 노드 접근 횟수와 같은 유형을 보였으며, QMBR은 다른 것과 많은 차이를 보였다.

결과적으로 SA는 2차원 공간 인덱스인 R-tree의 MBR보다 좋은 성능을 보였으며, 작은 차이기는 하지만 기존 압축 방법들 보다 향상되었다. 키 값이 인덱스의 대부분을 차지하는 구조에서는 키 값의 압축은 인덱스의 크기뿐 아니라 트리의 높이를 낮춤으로써 검색시간을 감소시킨다. 이때, 압축된 키 값을 원래 키 값으로 변환해야 하지만 논문에서 제안하는 알고리즘을 적용하면 키의 변환 비용을 최소화 할 수 있다.

본 논문의 기법은 모바일 장치와 같이 메모리나 연산 능력에 제약이 있으면서 빠른 검색을 필요로 하는 곳에 적용하면 성능 향상을 가져올 수 있으며, 본 제안의 다음 연구로 진행될 것이다.

참고문헌

[1] 김병곤, 오성균, "다차원 인덱싱 구조에서의 k-근접객체질의 처리 방안," 한국컴퓨터정보학회 논문지, 제10권, 제1호, pp. 87-91, 2005.

[2] 전봉기, "3차원 R-트리를 이용한 이동체 색인에 관한 연구," 한국컴퓨터정보학회 논문지, 제10권, 제4호, pp. 65-75, 2005.

[3] Antonin Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," ACM SIGMOD, pp. 47-57, 1984.

[4] Berchtold S, Keim DA, Kriege H-P, "The X-tree: An index structure for high-dimensional data," Proc 22nd Int. Conf. on VLDB, pp. 28-39, 1996.

[5] Katayama N, Satoh S., "The SR-tree: An index structure for high-dimensional nearest neighbor queries," Proc ACM SIGMOD Int. Conf. on Management of Data, pp. 396-380, 1997.

[6] Jin-Deog Kim, Sang-Ho Moon, Jin-Oh Choi, A Spatial Index Using MBR Compression and Hashing Technique for Mobile Map Service, DASFA2005, LNCS3453, pp. 625-636, 2005.

[7] Kihong Kim, Sang K. Cha, Keunjoo Kwon, "Optimizing Multidimensional Index trees for Main Memory Access," Int. Conf. on ACM SIGMD, pp. 139-150, 2001.

[8] Y. Sakurai, M. Yoshikawa, S. Uemura, H. Kojima, "Spatial indexing of high-dimensional data based on relative approximation," VLDB J. 11, pp. 93-108, 2002.

[9] J. Goldstein, R. Ramakrishnan, and U. Shaft, "Compressing Relations and Indexes," Proceedings of IEEE Conference on Data Engineering, pp. 370-379, 1998.

[10] http://www.rtreeportal.org/spatial.html

[11] Herb Schwetman, "CSIM19: A Powerful Tool for Building System Models," Proceedings of the Winter Simulation Conference, pp. 250-255, 2001.

저 자 소 개

김 종 완

2005. 8 고려대학교 컴퓨터학과 박사과정 수료.

〈관심분야〉 데이터베이스, 모바일 데이터베이스 , 공간 데이터베이스, RFID/USN 데이터관리