

# 메인 메모리를 위한 효율적인 공간 인덱스 구조

## An Efficient Spatial Index Structure for Main Memory

이기영\*, 임명재\*\*, 강정진\*\*\*, 김정준\*\*\*\*

Ki-Young Lee, Myung-Jae Lim, Jeong-Jin Kang, Joung-Joon Kim

요 약 최근 실시간 서비스의 요구 사항을 갖는 위치 기반 서비스와 텔레매틱스 서비스를 효율적으로 제공하기 위해서 공간 메인메모리 DBMS에 대한 관심이 급증하고 있다. 이러한 공간 메인 메모리 DBMS에서 기존의 디스크 기반 공간 인덱스들을 메인 메모리에 최적화하기 위해 엔트리 크기를 줄여 캐시 접근 실패를 최소화한 공간 인덱스 구조들이 제안되고 있다. 그러나 엔트리 크기를 줄이기 위하여 부모 노드의 MBR을 기준으로 압축하거나 중복된 MBR을 제거하기 때문에 인덱스 갱신 시 MBR 재구성 비용이 증가하고 인덱스 검색 시 효율이 떨어지는 문제점이 있다. 본 논문에서는 MBR 재구성 비용을 줄이기 위하여 넓은 분포의 경우와 좁은 분포의 경우로 나누어 압축 기준점을 다르게 적용하는 RSMBR(Relative-Sized MBR) 압축 기법을 제시하였다. RSMBR 압축 기법은 넓은 분포일 경우 부모 노드 확장 MBR의 좌하점을 기준으로 압축하고, 좁은 분포일 경우 전체 MBR을 일정 크기의 셀로 나누고 각 셀의 좌하점을 기준으로 압축한다. 또한 인덱스 검색 시 검색 비용을 줄이기 위하여 상대 좌표와 크기를 이용하여 MBR을 압축한다. 마지막으로, 본 논문에서는 실제 데이터를 통한 성능 평가를 수행하여 RSMBR 압축 기법의 우수성도 입증하였다.

**Abstract** Recently there is growing interest in LBS requiring real-time services and the spatial main memory DBMS for efficient Telematics services. In order to optimize existing disk-based spatial indexes of the spatial main memory DBMS in the main memory, spatial index structures have been proposed, which minimize failures in cache access by reducing the entry size. However, because the reduction of entry size requires compression based on the MBR of the parent node or the removal of redundant MBR, the cost of MBR reconstruction increases in index update and the efficiency of search is lowered in index search. Thus, to reduce the cost of MBR reconstruction, this paper proposed the RSMB (relative-sized MBR)compression technique, which applies the base point of compression differently in case of broad distribution and narrow distribution. In case of broad distribution, compression is made based on the left-bottom point of the extended MBR of the parent node, and in case of narrow distribution, the whole MBR is divided into cells of the same size and compression is made based on the left-bottom point of each cell. In addition, MBR was compressed using a relative coordinate and size to reduce the cost of search in index search. Lastly, we evaluated the performance of the proposed RSMBR compression technique using real data, and proved its superiority.

**Key Words :** MBR Compression, Spatial Index, Spatial Data

### I. 서 론

최근 지리 정보 시스템 기술이 발전함에 따라 위치 기

반 서비스, 텔레매틱스, 지능형 교통 시스템 등과 같은 다양한 GIS 응용 분야에서 복잡한 공간 데이터의 신속하고 효율적인 처리가 절실히 요구되고 있다.

이를 위해 전체 데이터베이스를 메인 메모리에 상주시켜 처리하는 공간 메인 메모리 DBMS가 연구 개발되고 있다<sup>[1][2]</sup>. 또한, 이러한 공간 메인 메모리 DBMS에서

\*정회원, 을지대학교 의료산업학부

\*\*중신회원, 을지대학교 의료산업학부(교신저자)

\*\*\*중신회원, 동서대학교 정보통신과

\*\*\*\*정회원, 건국대학교 컴퓨터공학과

접수일자 2009.3.27, 수정일자 2009.4.13

기존의 디스크 기반 공간 인덱스<sup>[3][4]</sup>들을 메인 메모리에 최적화하기 위해 캐시를 고려한 공간 인덱스 구조들이 제안되었다. 이런 인덱스 구조들의 궁극적인 목표는 엔트리 크기를 줄여 팬-아웃을 증가시키고, 캐시 접근 실패를 최소화하여 시스템의 성능을 높이는 것이다<sup>[5][6][7]</sup>.

Rao와 Ross는 메인 메모리 인덱스의 설계에 있어서 캐시 성능의 중요성을 제기하고 이진 탐색 트리나 T-Tree 보다 빠른 검색 성능을 제공하는 CSS-Tree(Cache-Sensitive Search Tree)를 제안하였다<sup>[8]</sup>. 그리고, 이를 확장하여 B+-Tree의 캐시 성능을 향상시키는 CSB+-Tree를 제안하였다<sup>[9]</sup>. 또한, Chen, Gibbons 그리고 Mowry는 캐시 실패로 인한 메모리 지연을 줄이기 위해 선반입을 이용한 PB+-Tree(Prefetching B+-Tree)를 제안하였다<sup>[10]</sup>. 공간 인덱스인 R-Tree에 대해서는 Sitzmann과 Stuckey가 R-Tree의 노드 크기를 캐시 라인 크기에 맞추고, MBR의 불필요한 정보를 제거하여 노드에 보다 많은 정보를 저장하는 pR-Tree(partial R-Tree)를 제안하였다<sup>[11]</sup>. 그리고 Kim과 Cha는 엔트리의 MBR을 압축하여 노드에 보다 많은 엔트리를 포함시킬 수 있도록 하는 CR-Tree(Cache-conscious R-Tree)를 제안하였다<sup>[12]</sup>. 그러나 이러한 구조들은 메인 메모리 인덱스 구조에서 발생할 수 있는 캐시 실패를 줄였지만, 부모 노드 MBR을 기준으로 MBR을 압축하거나 중복된 MBR을 제거하기 때문에 갱신 시 MBR 재구성으로 인해 갱신 성능이 떨어지고 검색 시 정제 단계에 참여하는 객체 수의 증가로 인해 검색 성능이 떨어지는 문제점이 있다.

본 논문에서는 인덱스 갱신 시 MBR 재구성을 위한 추가 갱신 비용을 줄이기 위하여 넓은 분포의 경우와 좁은 분포의 경우로 나누어 MBR 압축 기준점을 다르게 적용하는 RSMBR(Relative-Sized MBR) 압축 기법을 제시하였다.

본 논문은 다음과 같이 구성되어 있다. 1장 서론에 있어서 2장에서는 기존의 캐시를 고려한 공간 인덱스와 MBR 압축 기법의 문제점에 대하여 기술한다. 3장에서는 메인 메모리 공간 인덱스를 위한 효율적인 RSMBR 압축 기법에 대하여 제시한다. 4장에서는 RSMBR을 적용한 RR-tree의 구조와 알고리즘에 대하여 설명한다. 5장에서는 제안된 RSMBR 압축 기술의 여러 가지 상황에서의 테스트를 통해서 우수성을 보인다. 마지막으로 6장에서 결론을 맺는다.

## II. 관련 연구

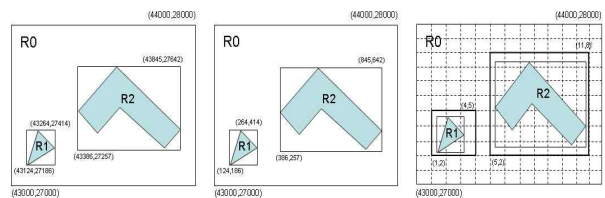
본 장에서는 기존의 캐시를 고려한 공간 인덱스와 MBR 압축 기법의 문제점에 대하여 설명한다.

### 1. 캐시를 고려한 공간 인덱스

CR-Tree(Cache-conscious R-tree)는 R-Tree의 캐시 실패를 줄이기 위해 R-Tree에서 인덱스의 대부분을 차지하는 MBR을 압축하여 압축된 QRMBR(Quantized Relative MBR)을 키로 사용하는 R-Tree이다<sup>[11]</sup>. R-Tree에서 자식 노드의 MBR들은 부모 노드의 MBR에 포함되기 때문에 자식 노드의 MBR을 부모 노드 MBR에 대해 상대적 좌표로 표현할 경우 좌표 값의 크기는 훨씬 작아진다. 그리고 상대적 좌표 값은 부모 노드 MBR의 좌표 값에 따라 최대값이 결정되므로 상대 좌표가 일정한 bit로 표현될 수 있도록 정량화 하면 일정한 크기로 압축된 MBR을 얻을 수 있다. 이러한 CR-tree는 pR-tree 보다 데이터 삽입과 검색에서 더 좋은 성능을 보여준다. 그러나, CR-Tree는 MBR을 QRMBR로 압축하는 과정에서 원래의 MBR과 오차가 발생할 수 있고, 이러한 오차에 의해서 검색 시 정제 단계에 참여하는 객체 수의 증가로 검색 성능이 떨어질 수 있다. 또한, 갱신 시 QRMBR의 재구성을 위한 추가 연산으로 갱신 성능이 나빠지는 단점이 있다.

### 2. MBR 압축 기법

기존의 MBR 압축 기법은 MBR이 커지는 문제와 빈번하게 MBR 압축을 반복해야 하는 문제를 가지고 있다. 그림 1은 절대 좌표 MBR 기법, RMBR(Relative MBR) 압축 기법<sup>[12]</sup>, CR-tree에서 제안한 QRMBR(Quantized Relative MBR) 압축 기법을 보여준다<sup>[12]</sup>. R0는 2개의 자식 노드를 갖는 부모 노드이고 R1, R2는 R0의 자식 노드이다.



(a) 절대 좌표 MBR (b) RMBR (c) QRMBR

그림 1. MBR 압축 기법

Fig 1. MBR Compression Technique

그림 1(a)는 절대 좌표값을 이용한 MBR 표현 기법을 보여준다. 절대 좌표 MBR은 각 X, Y 좌표마다 4byte가 필요하기 때문에 전체 16byte의 저장 공간이 필요하다. 그림 1(b)는 부모 노드 MBR의 좌하점을 기준으로 자식 노드의 MBR을 표현한 RMBR 압축 기법을 보여준다. RMBR은 각 X, Y 좌표마다 2byte가 필요하기 때문에 전체 8byte 저장 공간이 필요하다. 그리고 그림1(c)는 RMBR 좌표값에 정량화 기법을 적용한 QRMBR 압축 기법을 보여준다. QRMBR은 부모 노드의 MBR 크기를 가로 N개, 세로 M개의 그리드로 나누어 자식 노드의 RMBR을 정량화한다. 이렇게 정량화하면 QRMBR은 각 X, Y 좌표 마다 1byte가 필요하므로 전체 4byte의 저장 공간이 필요하다. 그러므로 QRMBR은 절대 좌표 MBR 과 RMBR에 비해 각각 4배와 2배의 압축 효과가 있다. 그러나 QRMBR은 MBR의 커짐 현상이 발생하고 이는 정제 단계에 참여하는 객체 수의 증가를 초래하여 검색 성능이 떨어진다.

그림 2에서와 같이 객체 R3이 삽입될 경우 R1과 R2의 부모 노드 R0의 MBR이 변경되었기 때문에 새로 삽입된 R3뿐만 아니라 R1과 R2도 MBR 압축을 재수행하여야 한다. 또한, 객체 R1이 갱신될 경우에도 R0의 MBR이 변경되기 때문에 R1과 R2의 MBR 압축을 재수행하여야 한다. 이렇게 부모 노드 MBR을 기준으로 MBR을 압축할 경우 부모 노드의 MBR 변경에 민감하기 때문에 빈번한 객체의 삽입과 갱신 시 부모 노드의 MBR 변경에 따른 MBR 재구성으로 인해 갱신 성능이 떨어지고 결국 전체 수행 시간의 증가를 초래한다.

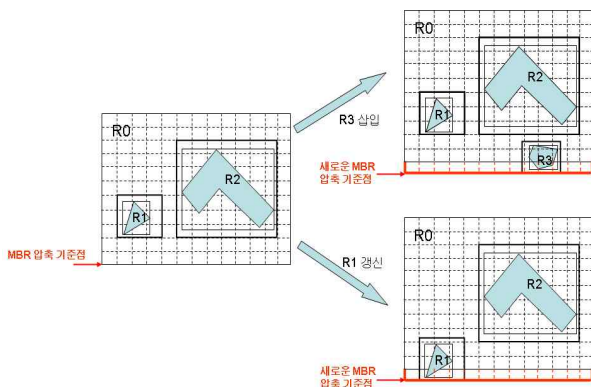


그림 2. MBR 재구성  
Fig 2. MBR Reconstruction

### III. RSMBR(Relative-Sized MBR)

본 장에서는 메인 메모리 공간 인덱스를 위한 효율적인 RSMBR 압축 기법에 대하여 설명한다.

#### 1. RSMBR 압축

본 논문에서는 MBR 압축 시 MBR이 커지는 현상을 제거하기 위하여 MBR의 좌하점은 상대 좌표값으로 표현하고 우상점은 MBR의 크기로 표현하는 RSMBR 압축 기법을 제시한다. 이 기법은 MBR의 정확도를 유지하면서 데이터의 양을 줄일 수 있다. 그림 3은 2차원 공간 데이터에 본 논문에서 제시하는 RSMBR 압축 기법을 적용한 예를 보여준다.

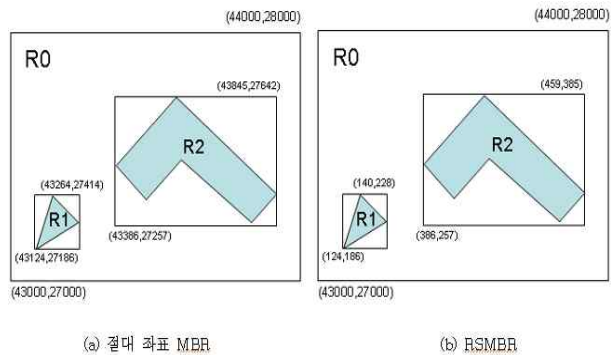


그림 3. MBR 압축 기법  
Fig 3. MBR Compression Technique

그림 3(a)는 절대 좌표 MBR로 표현된 그림이고, 그림 3(b)는 본 논문에서 제안하는 RSMBR 압축 기법으로 표현한 그림이다. 그림 3(b)에서 객체 R1의 MBR 좌하점 상대 좌표값은 (124, 186)이고 MBR의 크기는 X 축으로 140이고 Y 축으로 228이다. 그리고 객체 R2의 MBR 좌하점 상대 좌표값은 (386, 257)이고 MBR의 크기는 X 축으로 459이고 Y 축으로 385이다. 따라서, 객체 R1의 RSMBR은 (124, 186), (140, 228) 이고, 객체 R2의 RSMBR은 (386, 257), (459, 385)이다.

RSMBR 압축 기법에서 각 우상점 좌표값은 길이 플래그와 실제 값을 이용하여 저장된다. 그림 4는 RSMBR 좌표값의 데이터 저장 구조를 보여준다.

2byte	3bit	4-16bit
좌하점의 상대좌표	우상점의 길이 플래그	우상점의 실제 값

그림 4 .RSMBR 좌표 값의 데이터 저장 구조  
Fig 4. Data Structure of RSMBR Coordinates

길이에 대한 플래그는 실제 값이 차지하는 bit의 크기를 나타낸다. 즉, 1, 2, 3, 4는 각각 실제값이 4bit, 8bit, 12bit, 16bit의 크기를 가지고 있다는 것을 의미한다. 그러므로 각 bit 단위 마다 값은 15, 255, 4095, 65535의 크기까지 표현할 수 있다. 예를 들어, 객체 R1의 RSMBR 우상점 좌표값 중 140은 길이 플래그 010과 실제 값 10001100로 표현될 수 있고, 객체 R2의 RSMBR 우상점 좌표값 중 459는 길이 플래그 011과 실제 값 000111001011로 표현될 수 있다. 결론적으로 RSMBR의 각 X, Y 좌표값은 최소 6byte, 최대 10byte로 표현 가능하기 때문에 전체적인 MBR의 저장 공간이 줄어들게 된다.

대부분의 공간 객체의 MBR 크기는 255 이하이기 때문에 1byte로 표현할 수 있다. 따라서, 한 개의 RSMBR이 6byte를 넘는 경우는 그 빈도가 절대적으로 낮다. 이와 같이 RSMBR 압축 기법을 사용했을 때 점 질의 또는 영역 질의와 같은 공간 연산 수행을 위해 압축된 RSMBR을 복구하는 과정에서 다소의 연산이 필요하다. 그러나 RSMBR 압축 기법을 적용하여 압축된 객체들의 RSMBR을 복구하지 않고, 점 질의 또는 영역 질의의 질의 좌표를 객체들의 RSMBR과 동일한 방법으로 압축하여 공간 연산을 수행할 수 있다.

## 2. RSMBR 재구성

본 논문에서는 RSMBR 재구성을 위한 추가 갱신 비용을 줄이기 위하여 넓은 분포의 경우와 좁은 분포의 경우로 나누어 압축 기준점을 다르게 적용한다. 넓은 분포란 전체 MBR의 크기가 일정 크기보다 큰 경우이고 좁은 분포란 전체 MBR 크기가 일정 크기보다 같거나 작은 경우를 말한다. 본 논문에서는 넓은 분포와 좁은 분포를 구분하기 위한 기준 크기를 4byte로 하였다. 그림 5은 넓은 분포일 경우의 RSMBR 재구성 방법을 보여준다.

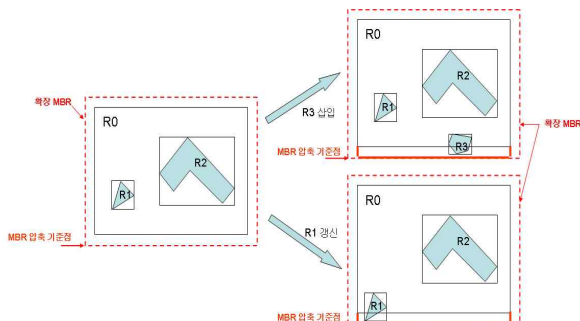


그림 5. 넓은 분포일 경우 RSMBR 재구성 방법  
Fig 5. RSMBR Reconstruction for Board Distribution

그림 5에서와 같이 객체 R3이 삽입될 경우 부모 노드 R0의 MBR이 변경되더라도 R3의 MBR이 R1, R2, R3의 부모 노드인 R0의 확장 MBR에 포함되기 때문에 RSMBR 재구성이 필요 없다. 또한, 객체 R1이 갱신될 경우에도 R1의 MBR이 R0의 확장 MBR에 포함되기 때문에 RSMBR 재구성이 필요 없다. 이렇게 넓은 분포일 경우 부모 노드 확장 MBR의 좌하점을 기준으로 압축하면 부모 노드의 MBR 변화에 둔감하기 때문에 RSMBR 재구성을 위한 추가 갱신 비용을 줄일 수 있게 된다.

그림 6은 좁은 분포일 경우의 RSMBR 재구성 방법을 보여준다.

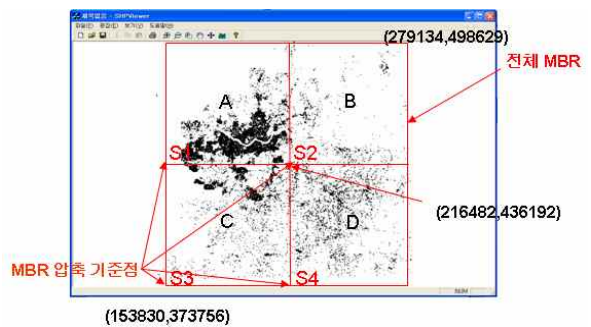


그림 6. 좁은 분포일 경우 RSMBR 재구성 방법  
Fig 6. RSMBR Reconstruction for Narrow Distribution

그림 6에서 전체 MBR의 크기가 일정 크기(4byte) 보다 같거나 작기 때문에 전체 MBR을 2byte의 셀(A, B, C, D)로 나누어 RSMBR 압축 기준점을 적용하였다. 즉, A 셀에 포함되는 객체들은 A 셀의 좌하점인 S1을 기준으로 압축되고 B 셀에 포함되는 객체들은 B 셀의 좌하점인 S2를 기준으로 압축된다. 그리고 C 셀과 D 셀에 포함되는 객체들도 각각 C 셀과 D 셀의 좌하점인 S3과 S4를 기준으로 압축된다. 이렇게 좁은 분포일 경우 전체 MBR을 2byte 단위의 셀로 나누고 각 셀의 좌하점을 RSMBR 압축 기준점으로 적용하면 RSMBR 재구성을 위한 추가 갱신 비용을 줄일 수 있게 된다.

## IV. RR-Tree(RSMBR R-Tree)

본 장에서는 RSMBR을 적용한 공간 인덱스인 RR-tree의 구조와 알고리즘에 대하여 설명한다.

### 1. RR-tree의 구조

그림 7에서 (a)는 RR-tree의 전체 노드 구조를 나타내며 P는 노드의 레벨을 나타내고, N은 엔트리 개수를

나타낸다. Base MBR은 각 엔트리의 압축 기준 MBR이다. 엔트리는 중간 노드와 단말 노드에 따라 두 가지 형태가 있다. 그림 7에서 (b)와 (c)는 각각 중간 노드와 단말 노드의 엔트리 형태를 나타낸다.

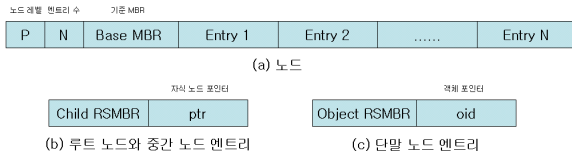


그림 7. RR-tree의 노드구조  
Fig 7. instruct of RR-tree

캐시 활용 도를 높이기 위해서 노드의 크기는 캐시 블록에 맞도록 엔트리의 개수를 제한한다. 즉, 캐시 블록의 크기에 비례해 노드 크기가 결정된다. 그리고 각각의 엔트리에는 중간 노드의 경우는 자식 노드의 RSMBR만 저장하고 단말 노드에는 객체의 RSMBR을 저장한다. 즉, ptr은 이 노드의 자식 노드에 대한 포인터이고, child RSMBR은 p에 의해 참조되는 자식 노드들을 완전히 포함하고 있는 RSMBR이며 oid는 실제 공간 객체를 참조하는 식별자이고 object RSMBR은 객체의 실제 RSMBR이다.

### 2. RSMBR 압축 알고리즘

본 절에서는 RSMBR 압축 알고리즘에 대하여 설명한다. RSMBR 압축 알고리즘은 좌하점이 상대 좌표를 갖고 우상점이 MBR의 크기를 갖는 RSMBR을 구한다. 그림 8은 RSMBR 압축 알고리즘을 보여준다.

```

입력
SL_MBR : 기준 MBR
Oid_MBR : 객체의 MBR
출력
RSMBR : 압축된 RSMBR
함수
Make_RSMBR : 좌하점은 상대 좌표값을 갖고 우상점은 MBR의 크기를 갖는 RSMBR을 구함

Algorithm -----
Begin
// RSMBR의 좌하점을 구함
RSMBR.LeftBottom = Oid_MBR.LeftBottom - SL_MBR.LeftBottom
// RSMBR의 우상점을 구함
RSMBR.RightTop = Oid_MBR.RightTop - Oid_MBR.LeftBottom
// RSMBR 좌표값을 미진수로 변환하여 저장
If (RSMBR.Coordinate.Size <= 15) // 크기가 15 보다 같거나 작을 경우
    RSMBR.Coordinate.Length = "001"
    RSMBR.Coordinate.Value = ConvertBit(RSMBR.Coordinate);
Else If (RSMBR.Coordinate.Size <= 255) // 크기가 255 보다 같거나 작을 경우
    RSMBR.Coordinate.Length = "010"
    RSMBR.Coordinate.Value = ConvertBit(RSMBR.Coordinate);
Else If (RSMBR.Coordinate.Size <= 4095) // 크기가 4095 보다 같거나 작을 경우
    RSMBR.Coordinate.Length = "011"
    RSMBR.Coordinate.Value = ConvertBit(RSMBR.Coordinate);
Else If (RSMBR.Coordinate.Size <= 65535) // 크기가 65535 보다 같거나 작을 경우
    RSMBR.Coordinate.Length = "100"
    RSMBR.Coordinate.Value = ConvertBit(RSMBR.Coordinate);
End if
Return RSMBR
End Begin
    
```

그림 8. RSMBR 압축 알고리즘  
Fig. 8. RSMBR Compression Algorithm

## V. 성능 평가 및 분석

본 장에서는 절대 좌표 MBR 기법, QRMBR 압축 기법, RSMBR 압축 기법 간의 성능 평가 및 분석에 대하여 설명한다.

### 1. 성능 평가 환경

성능 평가는 CPU 펜티엄 III 800MHz, 메모리 512 MB, 캐시 메모리(L1 : 32Byte 캐쉬 블록, L2 : 64Byte 캐쉬 블록), 그리고 Redhat 9.0 환경에서 수행되었다. 테스트 데이터로는 약 37MB 크기의 2차원 공간 데이터인 서울시 전체 건물 데이터를 사용하였다. 서울시 전체 건물 데이터는 249,115개의 공간 객체로 구성되어 있고, 각 객체는 최소 8개 최대 8,174개의 포인터 수를 가진다.

본 논문에서는 절대 좌표 MBR, CR-tree에서 제안한 QRMBR, 그리고 본 논문에서 제안한 RSMBR을 모두 공간 인덱스인 R-tree에 적용하여 성능을 비교 분석하였다. 즉, 성능 평가에서는 절대 좌표 MBR을 사용하는 R-tree, QRMBR을 사용하는 R-tree인 CR-tree, RSMBR을 적용한 R-tree인 RR-tree(RSMBR R-tree)를 비교 분석하였다. RR-tree는 R-tree에서 각 노드가 엔트리에 절대 좌표 MBR 대신 RSMBR을 사용한다는 것을 제외하고는 R-tree와 동일하다.

### 2. 인덱스 크기

그림 9는 전체 37MB 테스트 데이터 삽입 시 생성되는 R-tree, CR-tree, RR-tree에 대한 인덱스 크기를 비교한 것을 보여준다.

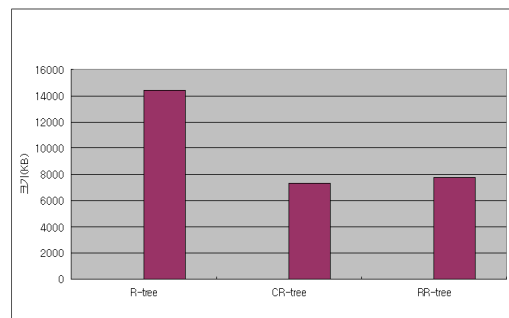


그림 9. 인덱스 크기  
Fig 9. Index Size

그림 9에서 보여주는 바와 같이 각각의 공간 인덱스에서 인덱스의 크기는 RR-tree가 CR-tree 보다 5~10% 성

능이 저하되었지만 R-tree 보다 45~50% 성능이 향상된 것을 알 수 있다. 이것은 RSMBR이 QRMBR 보다 약간 압축률이 떨어지기 때문에 인덱스의 크기가 다소 커진다는 것을 의미한다.

### 3. 데이터 갱신

그림 10은 갱신 대상 객체의 수가 각각 전체의 10%, 20%, 30%, 40%, 50%일 때 R-tree, CR-tree, RR-tree에 대해서 데이터 갱신 시간을 비교한 것을 보여준다.

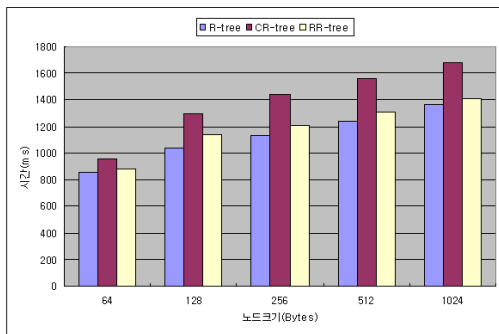


그림 10. 데이터 갱신 시간  
Fig 10. Data Update Time

그림 10에서 보여주는 바와 같이 각각의 공간 인덱스에서 데이터 갱신 시간은 RR-tree가 R-tree 보다 1~5% 성능이 저하되었지만 CR-tree 보다 20~25% 성능이 향상된 것을 알 수 있다. 이것은 RSMBR이 QRMBR 보다 MBR 압축 기준점 변화에 둔감하기 때문에 데이터 갱신 시 MBR 재구성을 더 적게 수행하면서 MBR 압축을 수행한다는 것을 의미한다.

그림 11은 노드 크기에 따른 갱신 시간을 비교한 것을 보여준다.

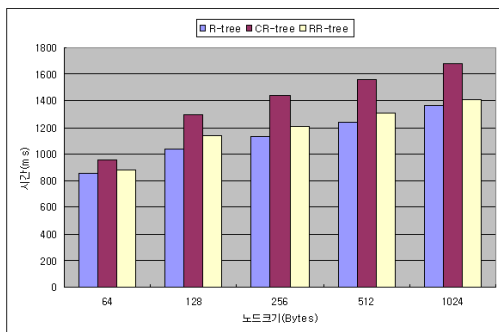


그림 11. 노드 크기에 따른 데이터 갱신 시간  
Fig 11. Data Update Time by Node's Size

그림 11에서 보여주는 바와 같이 노드 크기가 증가함에 따라 갱신 시간이 증가하는 것을 알 수 있다. 이것은 노드 크기가 증가하게 되면 노드 당 엔트리의 수가 급격히 증가하므로 MBR 압축 기준점 변경시 재구성해야 하는 엔트리의 수가 많아져 MBR 압축 재구성 비용이 증가하게 된다는 것을 의미한다.

데이터 갱신 시간은 MBR 기준점 변화 횟수와 갱신되는 노드의 수 그리고 갱신 되어야 할 엔트리 수로 구성된다. MBR 기준점 변화가 발생하면 새로운 MBR 기준점으로 해당하는 노드들의 엔트리들에 대해 MBR 압축 재구성을 수행하여야 한다. N<sub>mbr-change</sub>를 MBR 압축 기준점 변화가 발생한 노드수, E<sub>entry-update</sub>를 노드당 갱신 되어야 하는 엔트리 수, M<sub>mbr-compute</sub>를 엔트리 MBR 재구성 시간이라고 할 때 데이터 갱신 시간 T<sub>update-time</sub>은 다음과 같이 표현할 수 있다.

$$T_{update-time} = N_{mbr-change} \times E_{entry-update} \times M_{mbr-compute}$$

따라서, 데이터 갱신 시간을 최소화하기 위해서는 MBR 압축 기준점 변화에 둔감하게 하여 MBR 재구성을 최소화하여야 한다.

### 4. 데이터 검색

그림 12는 질의 영역이 각각 전체의 10%, 20%, 30%, 40%, 50%일 때 R-tree, CR-tree, RR-tree에 대해서 데이터 검색 시간을 비교한 것을 보여준 것이다.

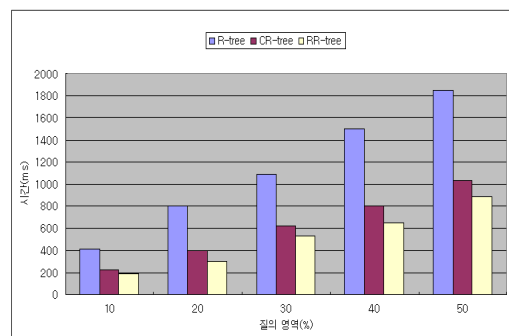


그림 12. 질의 영역에 따른 데이터 검색 시간  
Fig 12. Data Search Time By Query Scope

그림 12에서 보여주는 바와 같이 각각의 공간 인덱스에서 데이터 검색 시간은 RR-tree가 R-tree 보다 50~55% 성능이 향상되고 CR-tree 보다 10~15% 성능이

향상된 것을 알 수 있다. 이것은 RSMBR이 MBR 압축 과정에서 발생하는 MBR 커짐 현상을 제거하였기 때문에 QRMBR 보다 데이터 검색 시 비교 연산을 더 적게 수행하면서 데이터 검색을 수행한다는 것을 의미한다.

## VI. 결 론

최근 공간 인덱스의 메인 메모리 최적화를 위해 캐시 성능을 개선하기 위한 구조 및 알고리즘에 관한 연구가 많은 연구가들에 의해 다양한 방법으로 진행되어 왔다. 대표적인 방법으로 자식 노드의 포인터를 제거하거나 MBR을 압축하는 방법이 있다. 그러나 이러한 방법들은 갱신 시 엔트리 정보를 재구성하는 추가 연산으로 인해 갱신 성능이 떨어지고, MBR 압축 과정에서 MBR이 커져 정제 단계에 참여하는 객체 수가 증가하기 때문에 검색 성능이 떨어지는 문제가 있다.

본 논문은 이러한 문제점을 개선하기 위해 새로운 RSMBR 압축 기법을 제안하였다. 제안된 RSMBR 압축 기법은 넓은 분포의 경우와 좁은 분포의 경우로 나누어 압축 기준점을 다르게 적용함으로써 MBR 재구성을 위한 추가 비용을 줄였고, 상대 좌표와 크기를 이용하여 MBR을 압축함으로써 인덱스 검색 시 정제 단계에서 참여하는 객체 수 증가를 줄였다. 또한, 성능 평가를 수행하여 RSMBR 압축 기법이 기존의 다른 MBR 압축 기법 보다 우수함을 보였다. 즉, 인덱스 크기 측면에서 다소 성능이 떨어졌지만 데이터 삽입, 갱신, 검색에서 다른 MBR 압축 기법 보다 뛰어난 결과를 보였다.

## 참 고 문 헌

- [1] Yun, J. K., Kim, J. J., Hong, D. S., and Han, K. J., "Development of an Embedded Spatial MMBMS for Spatial Mobile Devices," Proc. of the 5th International Workshop on Web and Wireless Geographical Information Systems, 2005, pp.1-10.
- [2] Kim, J. J., Hong, D. S., Kang, H. K., and Han, K. J., "TMOM: A Moving Object Main Memory-Based DBMS for Telematics Services," Proc. of the 6th International Symposium on Web and Wireless Geographical Information Systems, 2006, pp.259-268.
- [3] Guttman, A., "R-Trees: a Dynamic Index Structure for Spatial Searching," Proc. of the ACM SIGMOD Conference, 1984, pp.47-54.
- [4] Mindaugas, P., Simonas, S., and Christian S., "Indexing the Past, Present, and Anticipated Future Positions of Moving Objects," Proc. of the ACM Transactions on Database Systems, 2006, pp.255-298.
- [5] Chen, S., Gibbons, P. B., Mowry, T. C., and Valentin, G., "Fractal Prefetching B+-Trees : Optimizing Both Cache and Disk Performances," Proc. of the ACM SIGMOD Conference, 2002, pp.157-168.
- [6] Bohannon, P., McIlroy, P., and Rastogi, R., "Main-Memory Index Structures with Fixed-Size Partial Keys," Proc. of the ACM SIGMOD Conference, 2001, pp.163-174.
- [7] Boncz, P., Manegold, S., and Kersten, M., "Database Architecture Optimized for the New Bottleneck : Memory Access," Proc. of the International Conference on VLDB, 1999, pp.54-65.
- [8] Rao, J., and Ross, K. A., "Making B+-Trees Cache Conscious in Main Memory," Proc. of the ACM SIGMOD Conference, 2000, pp.475-486.
- [9] Zhou, J., and Ross, K. A., "Buffering Accesses of Memory-Resident Index Structures," Proc. of the International Conference on VLDB, 2003, pp.405-416.
- [10] Chen, S., Gibbons, P. B., and Mowry, T. C., "Improving Index Performance through Prefetching," Proc. of th ACM SIGMOD Conference, 2001, pp.235-246.
- [11] Kim, K. H., Cha, S. H., and Kwon, K. J., "Optimizing Multidimensional Index Tree for Main Memory Access," Proc. of the ACM SIGMOD Conference, 2001, pp.139-150.
- [12] Goldstein, J., Ramakrishnan, R., and Shaft, U.,

“Compressing Relations and Indexes,” Proc. of the IEEE Conference on Data Engineering, 1998, pp.370-379.

저자 소개

이 기 영(정회원)



- 1984년 숭실대학교 전자계산학과학사 졸업.
- 1988년 건국대학교 대학원 컴퓨터공학과 석사 졸업.
- 2005년 건국대학교 대학원 컴퓨터공학과 박사 졸업
- 1984년 ~ 1991년 한국해양연구원 연구원

- 1996년 ~ 1998년 한국컴퓨터정보학회 이사 및 서울동부지 회장
  - 1991년 ~ 현재 을지대학교 의료산업학부 부교수
- <주관심분야 : 공간 데이터베이스, GIS, LBS, USN, 텔레매틱스 등>

임 명 재(중신회원) :교신저자



- 1998년 중앙대학교 대학원 컴퓨터공학과(공학박사)
  - 1992년 3월 ~ 현재 을지대학교 의료산업학부 부교수
- <주관심분야 : S/W공학, CBD 방법론, HCI 등>

강 정 진(중신회원)



- 2009 현재 동서울대학 정보통신과 교수
- 2009 현재 (사)한국인터넷방송통신TV학회 회장
- 2009 현재 (사)한국방송공학회 상임이사

<주관심분야 : 통방융합/Mobile IPTV, RFID/USN, 이동무선 통신/RF회로 및 소자, GIS/LBS/Telematics/GPS, Ultrafast Microwave Photonics>

김 정 준(정회원)



- 2003년 건국대학교 컴퓨터공학과 학사 졸업.
  - 2005년 건국대학교 대학원 컴퓨터공학과 석사 졸업.
  - 2005년 ~ 현재 건국대학교 대학원 컴퓨터공학과 박사 수료.
- <주관심분야 : 공간 메인 메모리 데이터베이스, GIS, LBS, 텔레매틱스, USN>