동적 비트 할당을 통한 다차원 벡터 근사 트리

Multi-Dimensional Vector Approximation Tree with Dynamic Bit Allocation

복경수*, 허정필**, 유재수*
충북대학교 정보통신공학과*, 매크로임팩트(주) 시스템 소프트웨어연구소**

Kyoung-Soo Bok(ksbok@netdb.chungbuk.ac.kr)*
Jung-Pil Heo(hjungpil@macroimpact.com)**, Jae-Soo Yoo(yjs@cbucc.chungbuk.ac.kr)*

요약

최근 컴퓨팅 환경의 급속한 발전으로 다양한 응용에서 다차원 데이터에 대한 활용이 증가되고 있다. 본 논문에서는 내용 기반 다차원 데이터 검색을 위한 벡터 근사 트리를 제안한다. 제안하는 색인 구조는 공간 분할 방식과 벡터 근사화 기법을 이용하여 영역 정보를 표현하기 때문에 하나의 노드 안에 많은 영역 정보를 저장하여 트리의 높이를 감소시킨다. 또한 다차원의 데이터 공간에 동적인 비트로 할당하여 다차원 색인 구조의 문제점인 "차원의 저주 현상"을 해결한다. 또한 군집화된 데이터에 대해서 효과적인 표현 기법을 제공한다. 자식 노드의 영역 정보는 부모 노드를 기준으로 상대적으로 표현함으로서 좀더 정확한 영역을 표현할 수 있다. 제안하는 색인 구조의 우수성을 보이기 위해 실험을 통해 기존에 제안된 색인 구조와의 비교 분석을 수행한다.

□ 중심어 : | 다차원 데이터 | 차원의 저주 | 벡터 근사 트리 | 유사도 검색 |

Abstract

Recently, It has been increased to use a multi-dimensional data in various applications with a rapid growth of the computing environment. In this paper, we propose the vector approximate tree for content-based retrieval of multi-dimensional data. The proposed index structure reduces the depth of tree by storing the many region information in a node because of representing region information using space partition based method and vector approximation method. Also it efficiently handles "dimensionality curse" that causes a problem of multi-dimensional index structure by assigning the multi-dimensional data space to dynamic bit. And it provides the more correct regions by representing the child region information as the parent region information relatively. We show that our index structure outperforms the existing index structure by various experimental evaluations.

□ Keyword : | Multi-Dimensional Data | Dimensionality Curse | | Vector Approximation Tree | Similarity Search |

이 논문은 2004년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음
접수번호 : #040616-001
접수일자 : 2004년 6월 16일
심사완료일 : 2004년 7월 27일
교신저자 : 유재수, e-mail : yjs@cbucc.chungbuk.ac.kr

I. 서 론

90년 이후 의학 분야, CAD/CAM 시스템, 내용 기반 검색 시스템 등과 같은 다양한 응용 분야에서 다차원 데이터에 대한 사용이 증가되었다. 특히, 내용 기반 검색 시스템은 이미지 또는 비디오에서 다차원의 특징 벡터를 추출하고 추출된 다차원의 특징 벡터와 유사한 객체를 찾는 유사도 검색을 수행한다. 이렇게 추출한 특징 벡터는 수십 차원의 특징 벡터로 구성되어 있기 때문에 이를 통한 유사도 검색을 효과적으로 수행하기 위한 다차원 색인 구조가 필요하다[1].

내용 기반 검색에서 사용되는 일반적인 다차원의 색인 구조는 추출된 다차원의 특징 벡터를 다차원 공간 내에 하나의 점으로 간주하여 색인을 구성한다[2]. 기존에 제안된 색인 구조는 실제 다차원의 데이터를 이용하여 색인을 구성하기 때문에 차원이 증가할수록 노드의 팬아웃(fanout)이 감소되고 색인 구조의 높이를 증가시킬 뿐만 아니라 분할된 영역들 사이에 선별력이 저하되어 검색 성능이 저하되는 문제점이 있다. 특히, k-최근접 검색(K-nearest neighbor search)에 대해서는 특징 벡터의 차원이 증가에 따라 검색 성능이 급속히 저하되는 문제점이 있다. 이러한 문제를 차원의 저주(dimensionality curse) 현상이라고 한다[2, 3].

최근 이러한 차원의 저주 현상을 해결하기 위해 다차원 데이터 공간을 특정 크기 단위로 분할하고 분할된 영역에 비트를 할당하여 근사화된 영역으로 색인을 구성하는 벡터 근사화 기법에 대한 연구들이 진행되고 있다[4, 5, 6, 7]. 이러한 기법의 하나인 VA-파일(Vector Approximation-File)은 각 차원에 특정 비트를 할당하여 전체 데이터 영역을 고정된 크기의 셀로 분할한다[4]. 분할된 각 셀에는 유일한 비트 값을 할당하여 실제 데이터 값에 대한 근사화를 수행한다. 근사화된 데이터 값들은 배열에 저장되고 전체 배열을 순차적으로 검사하면서 검색을 수행한다. 이에 반해, CS-트리(Cell based Signature-Tree)는 특징 벡터 공간을 VA-파일과 유사한 방법으로 분할하고 근사화된 값을 R-트리 기반의 색인 구조로 표현한다[5]. 또한, A-트리(Approximation-Tree)는 CS-트리와 같이 전체 벡터 공간을 분할하는 것이 아니라 R-트리 기반의 계층 구조의 영역을 기준으로 VBR(Virtual Bounding Rectangle)을 표현하고 상대적인 영역 정보 또는 데이터 값을 비트 형태로 표현한다[6, 7]. 이러한 벡터 근사화 색인 구조들은 노드의 팬아웃을 증가시켜 색인 구조의 높이를 감소시키는 장점이 있다. 그러나 전체 영역 또는 상대적인 영역을 표현하기 위해 고정된 비트 값을 할당하기 때문에 근사화된 데이터 영역들에 대한 정확도가 감소된다. 특히, 데이터 분포가 일부 영역에 군집화(Clustering)되어 있을 경우 검색 성능이 현격히 저하된다. 또한 영역의 분포 특성을 고려하지 않고 정해진 비트 값을 할당하기 때문에 동적인 삽입과 삭제가 발생할 경우 검색 성능이 저하된다.

따라서 본 논문에서는 기존에 제안된 벡터 근사 기법의 문제점을 해결하기 위한 ABA-트리(Adaptive Bit Allocation-Tree)라는 새로운 다차원 색인 구조를 제안한다. 제안하는 색인 구조는 공간 분할 방식에 의해 분할을 수행하고 실제 데이터들이 존재하는 영역은 벡터 근사화를 이용하여 표현한다. 제안하는 색인 구조에서 분할된 영역에 대한 명확한 근사화를 수행하기 위해 다차원의 특징 벡터 공간에 대해 고정된 비트를 할당하는 것이 아니라 데이터의 분포 특성에 따라 동적 비트를 할당한다. 또한 색인 구조의 각 계층에 존재하는 영역 정보들은 부모 노드를 기준으로 상대적으로 표현한다. 이를 위해 각 노드에는 상대적 영역을 표현하기 위해 필요한 정보를 노드의 헤더에 표현한다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 기존에 제안된 다차원의 색인 구조의 특징을 분석한다. 분석된 특징을 기반으로 Ⅲ장에서는 본 논문에서 제안하는 색인 구조에 대해 기술하고 Ⅳ장에서는 제안하는 색인 구조에서 수행하는 삽입 및 분할 기법에 대해 기술한다. Ⅴ장에서는 본 논문에서 제안하는 색인 구조의 우수성을 입증하기 위해 성능 평가를 수행한 결과를 기술한다. 마지막으로 Ⅵ장에서는 결론 및 향후 연구에 대해 기술한다.

II. 관련 연구

기존에 제안된 고차원의 색인 구조는 고차원의 데이

터 영역을 분할하는 방법에 따라 공간 분할(space partition) 방식과 데이터 분할(data partition)방식으로 구분한다[2]. 공간 분할 방식은 데이터가 존재하는 전체 벡터 공간을 데이터의 분포 특성을 고려하여 커다란 영역으로 분할하는 것으로 KDB-트리[8], LSD-트리[9], LSDh-트리[10] 등이 있다. 그러나 객체가 존재하는 영역만을 표현하는 것이 아니라 전체 공간을 표현하기 때문에 사장 공간(dead space)이 발생하는 문제점이 있다. 이에 반해 데이터 분할 방식은 객체가 존재하는 최소 영역을 표현하는 방법으로 R-트리[11], R*-트리[12], TV-트리[13], X-트리[3] 등이 있다. 데이터 분할 방식은 전체 데이터 공간에서 실제 데이터 객체가 존재하는 영역만을 표현하기 위해 MBR(Minimum Bounding Rectangular)을 구성하기 때문에 기존의 공간 분할 방식의 단점인 사장 공간을 효과적으로 감소시켰다. 그러나 이러한 색인 구조들은 차원이 증가할수록 노드의 팬아웃(Fanout)이 감소되고 색인 구조의 높이를 증가시킬 뿐만 아니라 분할된 영역들 사이에 겹침이 증가되어 검색 성능이 저하되는 문제점이 있다.

최근 다차원 색인 구조의 문제점인 "차원의 저주" 현상을 극복하기 위해 벡터 근사 기반의 색인 구조에 대한 연구들이 활발히 진행되고 있다. VA-파일(Vector Approximation-File)은 객체 근사화를 기반으로 하는 색인 구조이다. 즉, 전체 벡터 공간상에 존재하는 객체들의 데이터 값을 간단한 비트 값의 형태로 표현하는 기법이다. VA-파일은 벡터 근사화를 통해 생성된 데이터 값을 배열 구조에 저장한다. 이처럼 VA-파일은 배열 구조를 가지고 있기 때문에 10차원 이상에서도 선형 검색보다 빠른 성능을 보이는 특징을 가지고 있다. VA-파일에서 K-최근접(K-Nearest Neighbor) 검색은 배열에 저장된 내용을 순차적으로 접근하는 데이터를 필터링하는 방식을 사용하고 있다. 따라서, K-최근접 검색 과정에서 거의 모든 데이터를 조사한다. 이러한 VA-파일의 검색 성능은 색인 파일의 크기에 많은 영향을 받으며 검색 과정에서 순차적으로 검색을 하기 때문에 다차원의 대용량의 데이터 검색에서는 성능이 저하되는 문제점이 있다. 즉, 기존에 제안되었던 다차원의 색인구조에서는 검색 공간을 줄여가면서 객체를 검색해 가는 기존의 트리 형태의 계층적 색인 구조의 장점이 손실되는 문제가 발생한다.

CS-트리(Cell based Signature-tree)는 하나의 벡터 공간을 일정한 크기의 셀들로 분할하고 분할된 각 셀에 고정된 비트를 할당하여 근사화된 영역 CMBR(Cell based MBR)을 표현한다. CS-트리의 중간 노드에서는 실제 데이터들이 존재하는 MBR을 비트 값으로 표현하기 때문에 하나의 노드 안에 들어가는 정보량은 실제 MBR을 기록하는 것보다 많은 정보를 저장할 수 있다. 이로 인해 노드의 팬아웃을 기존 R-트리보다 증가시킬 수 있고 기존의 다차원 색인 구조보다 색인의 높이를 감소시킬 수 있다는 장점이 있다. 그러나 실제 MBR을 고정된 비트로 표현된 CMBR로 표현하기 때문에 특정 영역에 군집화된 데이터 분포에 대해 검색 성능이 저하되는 문제점이 있다. 또한, 전체 영역에 대해 고정된 비트를 할당하여 근사화를 수행하기 때문에 데이터의 분포 특성에 따라 CMBR들 사이에 선별력이 저하되거나 겹침 영역이 증가될 수 있는 문제점이 있다 .

A-트리(Approximation-tree)는 최근에 제안된 CS-트리와 거의 유사한 색인 구조를 갖는다. A-트리는 CS-트리와 같이 전체 벡터 공간을 셀 단위로 분할하여 하위 영역 정보를 비트 형태로 표현한 구조이다. 그러나 A-트리는 CS-트리와 같이 전제 데이터 공간에 대해 고정된 비트를 할당하는 것이 아니라 자식 노드에 대한 영역은 부모 노드를 기준으로 상대적인 영역 VMBR (Virtual MBR)을 표현한다. 이와 같이 부모 노드를 기준으로 상대적인 영역을 표현하기 때문에 보다 정확한 영역 정보를 표현할 수 있으며 이로 인해 향상된 검색 성능을 제공할 수 있다. 그러나 A-트리는 R-트리 기반 색인 구조에 영역을 표현하는 방법만을 수정한 것이기 때문에 분할된 영역들 사이에 겹침 영역을 발생할 수 있다. 또한 자식 노드에 대한 영역을 표현하기 위해 기준이 되는 부모 노드에 대한 영역이 변경될 경우 자식 노드들을 재구성해야 하는 문제점이 있다.

III. 제안하는 색인 구조의 구조

1. 제안하는 색인 구조의 특징

최근에 제안된 벡터 근사 기반의 색인 구조들은 실제 데이터들이 존재하는 영역을 표현하는데 실제 MBR 정보를 기록하는 것이 아니라 비트 형태로 표현된 근사화된 영역을 표현하여 색인 구조의 팬아웃을 증가시키는 장점이 있다. 그러나 동적인 데이터의 삽입과 삭제로 인해 분할된 영역들 사이에 겹침 영역이 증가되거나 근사화된 영역들 사이에 선별력이 저하되어 검색 성능이 저하되는 문제점이 있다. 제안하는 색인 구조에서는 기존에 제안된 벡터 근사 기반 색인 구조의 문제점을 해결하기 위해 다음과 같은 특징을 고려한다. 첫째, 전체 데이터 공간에 고정된 정적 비트를 할당하여 영역을 표현할 경우 데이터들이 특정 영역에 군집화되어 있을 경우 분할된 영역들 사이에 선별력이 저하되어 검색 성능에 많은 영향을 미친다. 따라서, 제안하는 색인 구조에서는 분할된 영역 내에 존재하는 데이터들의 분포 특성에 따라 동적인 비트를 할당한다. 둘째, 부모 노드를 기준으로 자식 노드의 위치 정보를 상대적으로 표현하는 경우 자식 노드에 대한 영역 정보를 보다 정확하게 표현할 수 있지만 동적인 데이터의 삽입이나 삭제에 의해 기준 영역이 되는 부모 노드가 변경되면 자식 노드들도 변경을 수행해야 한다. 따라서, 제안하는 색인 구조에서는 부모 노드를 기준으로 상대적인 영역을 표현하지만 A-트리와 같이 영역 기반 분할 정책을 수행하는 것이 아니라 공간 분할 방식에 의해 데이터를 삽입한다.

제안하는 색인 구조에서 데이터의 삽입은 공간 분할 방식에 의해 수행되고 실제 데이터들이 존재하는 영역은 데이터 분포 특성에 따라 근사화된 영역을 표현한다. [그림 1]은 제안하는 색인 구조에 의해 분할된 영역을 나타낸 것이다. [그림 1]에서는 S_i는 데이터 공간에 대해 분할을 수행한 위치로 제안하는 색인 구조에서는 공간 분할을 수행한다. 각 S_i는 영역들 사이에 겹침이 발생하지 않는 분할 위치로 단말 노드에서는 실제 데이터들이 존재하는 영역을 기준으로 겹침이 없으면 최적의 영역을 생성할 수 있는 위치를 선택하며 중간 노드에서는 기존에 중간 노드에 존재하는 분할 정보를 이용하여 선택된 분할 위치로 분할된 영역들 사이에 겹침이 없으면서 하위 노드에 대한 연속 분할이 발생하지 않는 위치이다. BM_i는 공간 분할에 의해 생성된 영역에 대해 비트를 할당하여 근사화한 영역으로 BMBR(Bit MBR)이다. 또한, P_i는 실제 다차원의 데이터이다.

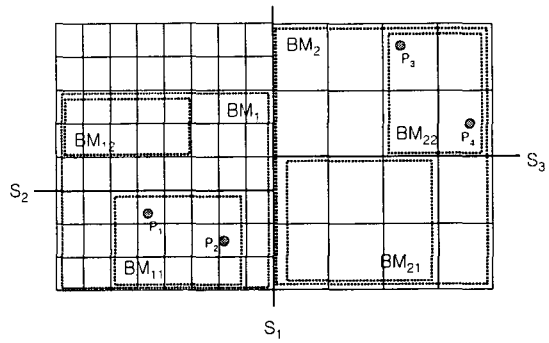


그림 1. 데이터 공간

만약 하나의 노드에 팬아웃이 3이라고 할 때 [그림 1]과 같이 분할된 영역을 색인으로 구성하면 [그림 2]와 같다. 제안하는 색인 구조는 노드에 대한 오버플로우가 발생할 경우 공간 분할을 수행하고 분할된 영역에 대해 실제 데이터들이 존재하는 영역에 비트를 할당하여 근사화된 영역을 표현한다. 또한, 자식 노드에 대한 영역 즉, BMBR은 부모 노드를 기준으로 대적으로 표현한다. 따라서, 제안하는 색인 구조의 각 노드에는 공간 분할을 위해 필요한 정보와 함께 실제 BMBR을 표현하기 위해 필요한 정보를 헤더에 표현해야 한다. [그림 2]의 각 노드에 존재하는 H_i는 노드에 대한 헤더 정보를 나타내고 BM_i는 헤더를 기준으로 표현된 하위 노드에 대한 BMBR을 나타낸다.

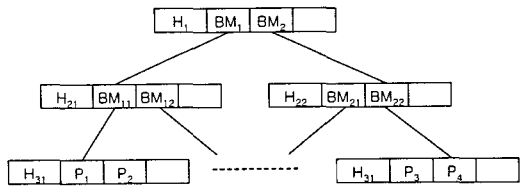


그림 2. 색인 구조의 구성

2. 노드 구조

제안하는 ABA-트리의 중간 노드는 정확한 자식 노드의 위치 정보를 표현하기 위해서 헤더와 엔트리로 구성되어 있다. 중간 노드에 존재하는 헤더는 하위 노드에 대한 영역을 표현하기 위해 $\langle SP, BitNum, PI\rangle$로 구성되어 있다. 이때, SP는 하위 노드에 대한 영역을 표현하기 위해 기준이 되는 영역으로 현재 노드가 공간 분할에 의해 처음 생성되었을 때의 전체 영역 정보에 대한 실제 값을 표현한다. $BitNum$는 SP을 기준으로 분할된 영역에 대한 BMBR을 표현하기 위해 할당된 비트 수를 나타낸다. 마지막 PI는 현재 노드에 포함되어 있는 하위 노드에 대한 영역들이 분할된 정보로 이러한 공간 분할 정보는 분할 차원와 분할 위치를 비트로 표현한다. PI는 중간 노드에 오버플로우가 발생하여 분할을 수행할 경우 KDB-트리 기반의 색인 구조의 문제점인 연속 분할(Cascading Split)을 해결하기 위해 기존에 분할된 정보를 기록하고 이를 통해 공간 분할을 하기 위한 적절한 위치를 찾기 위해 사용된다. [그림 3]은 중간 노드의 헤더 부분에 존재하는 분할 정보 PI을 나타낸 것이다. [그림 3]에서 ①~④는 분할된 순서를 나타낸 것이다. PI는 분할된 정보를 표현하기 위해 분할 차원과 분할 위치를 비트 형태로 표현한다. 2차원의 데이터 공간에 대한 분할 차원을 구별하기 위해서는 하나의 비트를 이용하여 첫 번째 차원으로 분할되었을 경우 0으로 표현하고 두 번째 차원으로 분할되었을 경우에는 1로 표현한다. 또한 분할 위치는 현재 노드에 할당된 $BitNum$에 해당하는 비트를 이용하여 분할 위치 정보를 나타낸다.

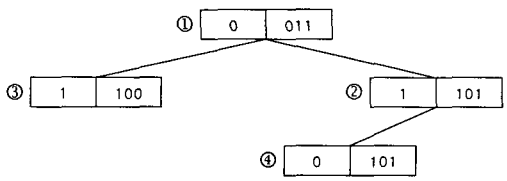


그림 3. 분할 정보

중간 노드에 존재하는 엔트리는 BMBR을 통해 실제 데이터들이 존재하는 영역과 함께 자식 노드를 접근하기 위한 정보를 표현한다. 중간 노드의 엔트리는 $\langle BMBR_i, Ptr_i\rangle$로 표현되며 $BMBR_i$는 중간 노드에 존재하는 헤더를 기준으로 자식 노드를 포함하는 BMBR을 표현한 것으로 분할 과정에서 데이터의 분포 특성에 따라 동적인 비트를 할당하여 표현한다. 또한, Ptr_i는 자식 노드를 접근하기 위한 포인터를 나타낸다.

단말 노드는 실제적인 다차원의 데이터를 저장한다. 단말 노드는 중간 노드와 유사하게 헤더와 엔트리로 구성된다. 단말 노드의 헤더는 SP로 구성되어 있으며 SP는 기존의 공간 분할에 의해 분할된 영역 정보를 나타낸다. 이러한 SP는 단말 노드에 공간 기반 분할을 수행하기 위해 단말 노드에 존재하는 엔트리들이 존재하는 영역을 나타낸다. 즉, 단말 노드에 다차원의 데이터를 포함하고 있다면 실제적인 데이터만 포함된 영역을 나타내기 때문에 공간 분할을 수행하기 위해 기존에 공간 분할에 의해 생성된 영역을 표현한다. 단말 노드의 엔트리 $\langle FV_i, OID_i\rangle$는 실제적인 다차원의 데이터 FV_i와 객체 식별자 OID_i로 구성되어 있다. 다차원의 데이터 FV_i는 다차원의 포인트 데이터 $(P_1, P_2, \ldots, P_n)$을 나타낸다.

Ⅳ. 삽입 및 분할 기법

1. 삽입

ABA-트리는 기존에 제안된 KDB-트리의 특성과 A-트리의 특성을 이용하여 삽입 과정을 수행한다. 데이터의 삽입은 KDB-트리와 유사하게 공간 분할 방식을 이용하여 삽입하기 적절한 위치를 검색하고 실제 데이터가 삽입된 영역은 A-트리와 유사하게 삽입된 영역에 비트를 할당하여 근사화된 영역을 생성한다.

제안하는 색인 구조는 새로운 데이터 NewE가 삽입되어 겹침 영역이 발생하는 단말 노드에 선택하고 선택된 단말 노드에 오버플로우가 발생하는 공간 분할 방식에 의해 분할을 수행한다. [그림 4]는 제안하는 색인 구조의 삽입 알고리즘을 나타낸 것이다. 제안하는 색인 구

조에서 새로운 데이터 NewE를 삽입하기 위해서는 먼저 루트 노드에서부터 삽입하기에 가장 적절한 단일 노드를 선택하기 위해 FindNode( )를 수행한다. FindNode( )는 KDB-트리의 삽입 알고리즘과 유사하게 각 노드에 존재하는 헤더를 이용하여 기존에 분할 영역을 확인한다. 기존에 분할 영역을 통해 새로운 엔트리가 겹침이 없는 삽입이 가능한 영역을 확인하고 실제 노드에 존재하는 엔트리를 통해 자식 노드를 포인터를 획득한다. 이러한 과정은 단말 노드에 도달할 때까지 반복 수행한다. 단말 노드에 도달하면 KDB-트리의 정책을 수행하지 않고 R-트리의 MBR 구성 정책을 적용하여 새로운 객체의 삽입으로 MBR 영역에 대한 변경이 필요하면 이를 계산하여 부모 노드에 반영하기 위한 AdjustBMBR( )를 수행한다. 만약 단말 노드에 NewE를 삽입할 여유 공간이 없는 경우에는 SplitNode( )를 수행하여 분할 작업을 수행한다.

```
Algorithm Insert(NewE : 삽입할 데이터, Node:루트 노드)
{
if(Node==LeafNode){ /* 루트 노드가 단말 노드 */
   /* NewE의 삽입으로 오버플로우 발생하는지 확인 */
   Check=CheckOverflow(Node, NewE); /
   if(Check==Overflow) /* 오버플로우 발생 */
      SplitNode(Node, NewE, NULL);
   else  /* 단말 노드에 새로운 엔트리를 삽입 */
      MBR=InsertEntry(Node, EewE);
}
else{
   /* 탐색 경로를 기록할 스택을 초기화 */
   Stack=InitStack( );
   /* 새로운 엔트리를 삽입할 단말 노드 검색 */
   Leaf=FindeNode(Node, NewE, Stack);
   Check=CheckOverflow(Leaf, NewE);
   if(Check==Overflow)
       /* 단말 노드를 분할 부모 노드에 반영 */
       SplitNode(Leaf, NewEntry, Stack);
   else{
      /* 단말 노드에 새로운 엔트리 삽입 */
      MBR=InsertEntry(Node, EewE);
       /*부모 노드에 BMBR의 변경을 반영 */
       AdjustBMBR(MBR, Stack);
   }
}
}
```

그림 4. 삽입 알고리즘

2. 분할

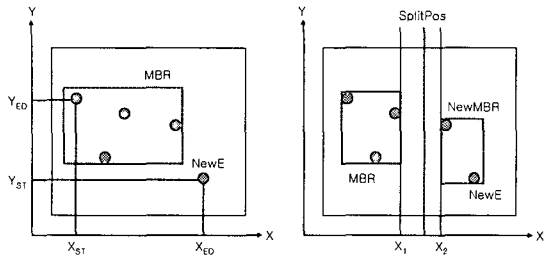
제안하는 색인 구조에서 분할은 분할 영역들 사이에 겹침 영역을 발생시키지 않는 공간 분할 방식을 수행한다. 공간 분할 방식에 의해 분할된 영역은 실제 데이터들이 존재하지 않는 사장 공간(Dead Space) 영역을 포함하고 있기 때문에 검색 성능을 저하시킬 수 있다. 따라서, 제안하는 색인 구조의 분할에 의해 생성된 각 영역에 대해 실제 데이터들이 존재하는 영역에 대해서만 데이터 분포 특성을 고려하여 동적 비트를 할당하여 벡터 근사화를 수행하여 BMBR을 구성한다.

단말 노드에 새로운 데이터를 삽입하여 오버플로우가 발생할 경우 단말 노드에 존재하는 데이터들의 특성에 따라 공간 분할을 수행하고 분할된 영역을 부모 노드에 반영한다. 단말 노드에 대한 분할 위치의 선택은 실제 데이터를 이용하여 데이터의 분포 특성에 따라 실제 BMBR을 생성할 때 최소 영역을 생성하기 위한 위치를 선택한다.

단말 노드에 분할 위치의 선택하기 위해서는 먼저 분할 차원을 선택해야 한다. 분할 차원은 실제 데이터가 존재하는 영역에 대한 데이터 분포 형태를 분석하고 데이터의 분포가 가장 넓은 축을 선택한다. 만약 동일한 분포 길이를 갖는 다수의 차원이 존재할 경우에는 실제 데이터가 존재하는 영역을 기준으로 데이터들의 밀집도가 높은 분할 차원을 설정한다. 데이터의 밀집도 높은 차원을 분할할 경우 적은 비트 수로 BMBR 영역 정보를 효과적으로 생성할 수 있다. 또한, 적은 비트 수만으로도 데이터가 존재하지 않는 사장 공간을 감소시킬 수 있다. 이로 인해 데이터 객체에 대한 검색 수행 과정에서 필요 없이 검색하는 노드의 수를 감소시킬 수 있다. 벡터 공간이 긴 차원을 분할 차원으로 설정할 경우 분할된 영역들 사이에 사장 공간을 제거시킬 수 있으며 BMBR을 정사각형에 가까운 형태로 구성할 수 있어 검색 성능을 향상시킬 수 있다.

[그림 5]는 2차원의 데이터에 대해 단말 노드에서 분할을 수행하는 과정을 나타낸 것이다. 단말 노드에 분할을 수행하기 위해서 먼저 분할 위치를 선택하기 위해서 먼저 분할을 수행할 차원을 선택한다. 분할 차원은 실제 데이터가 존재하는 영역에 대해 데이터의 분포 특성을 판별하기 위해 [그림 5]의 (a)와 같이 X 축과 Y 축으로 정사영을 시켜 실제 데이터들이 존재하는 영역 구간을

판별한다.



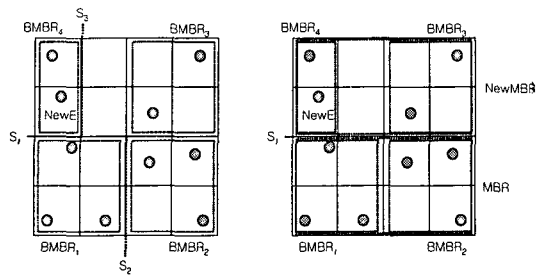
(a) 분할 전 (b) 분할 후

그림 5. 단말 노드 분할

[그림 5]의 (a)에서는 X 축이 Y축에 비해 실제 데이터들이 존재하는 영역의 분포가 크기 때문에 X축을 분할 축으로 선택한다. 분할 축을 선택하면 실제 데이터들의 분포 특성을 고려하여 분할된 영역들 사이에 겹침 영역이 발생하지 않으면서도 BMBR을 구성할 때 최소의 영역을 생성할 수 있는 위치를 선택한다. [그림 5]의 (b)에서 같이 X_1과 X_2 위치 사이에서 분할 축을 선택하는 것이 분할된 영역들 사이에 최소의 영역을 생성할 수 있다. 이러한 분할 축은 부모 노드에서 헤더 부분에 분할 정보로 표현하기에 적절해야 하기 때문에 X_1과 X_2 사이에서 부모 노드의 분할 위치 즉, 부모 노드의 헤더에서 분할 위치를 표현하는데 필요한 비트 수를 최소로 할 수 있는 위치를 선택한다. [그림 5]의 (b)에서는 이러한 위치를 *SplitPos*이라고 한다.

오버플로우가 발생한 단말 노드의 분할 과정이 완료되면 부모 노드에 분할된 정보를 반영하게 된다. 분할로 인하여 부모 노드에 새로 생성된 노드에 대한 정보와 기존에 존재하는 노드에 대한 정보를 반영하는 과정에서 부모 노드에 변경되는 정보를 반영할 여유 공간이 없을 경우에는 중간 노드에 대한 분할 과정을 수행한다.

공간 분할 방식의 색인 구조는 중간 노드의 분할에 의해 하위 노드를 계속적으로 분할하는 연속 분할(cascading split)이 발생할 수 있다. 따라서 중간 노드에 대한 분할 위치를 선택하기 위해서는 연속 분할이 발생하지 않은 위치를 선택해야 한다. 중간 노드의 분할 위치를 선택하기 위해서는 먼저 현재 노드의 공간 분할 영역 SP과 분할 정보 PI를 이용하여 연속 분할을 발생하지 않으면서도 분할 영역을 효과적으로 구성하기 위한 분할 위치를 선택한다. 다차원의 공간 영역에서 대부분의 경우 다차원의 영역에서 연속 분할을 수행하지 않은 차원은 노드에 존재하는 영역을 처음으로 분할한 차원 또는 아직 분할을 수행하지 않은 차원이 될 것이다. 본 논문에서는 중간 노드에 대한 분할 수행하기 위해서 먼저 노드의 헤더 정보에 있는 기존의 분할 정보와 자식 노드에서 새로 분할된 위치 정보를 이용하여 하위 노드에 대한 연속 분할이 발생하지 않으면서 분할된 영역들 사이에 BMBR 영역의 합이 최소가 되는 위치를 선택한다 [그림 6]은 중간 노드의 분할 과정을 나타낸 것이다. [그림 6]의 (a)와 같이 새로운 엔트리 NewE가 삽입되어 기존 $BMBR_3$ 영역이 S_3을 기준으로 $BMBR_3$과 $BMBR_4$로 분할되어 중간 노드에 분할된 영역 $BMBR_4$을 영하는 과정에서 오버플로우가 발생되었다고 가정하자. 중간 노드에 오버플로우가 발생하면 기존 분할 정보 S_1와 S_2 그리고 새로운 분할 S_3을 이용하여 자식 노드에 대한 연속 분할이 발생하지 않는 위치를 선택한다. 만약 분할 정보 S_2와 S_3에 의해 분할을 수행할 경우 기존에 하나의 영역들이 분할되어 자식 노드에 대한 연속 분할을 발생시킬 수 있다. 그러나 S_1는 현재 노드에 대한 분할 위치로 선택할 경우 자식 노드에 대한 연속 분할을 수행하지 않는다. 따라서, [그림 6]의 (b)와 S_1에 위치에 의해 분할을 수행하고 분할된 노드에 대한 MBR 정보를 생성한다. *NewMBR*는 분할로 인해 새로 생성된 노드의 MBR 정보이고 *MBR*는 기존 노드에 대한 MBR 정보이다. 이러한 MBR 정보는 부모 노드에 반영하는 과정에서 부모 노드에 존재하는 헤더를 기준으로 새로운 BMBR 정보로 변경되어 저장된다.



(a) 분할 전 (b) 분할 후

그림 6. 중간 노드 분할

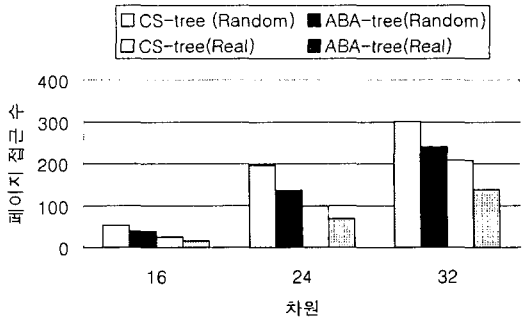


그림 7. 범위 검색에 대한 페이지 접근 수

V. 실험 및 성능 평가

1. 실험 환경

이 장에서는 제안하는 색인 구조와 CS-트리에 대한 성능 평가를 수행한 내용을 기술한다. 실험을 위한 구현 환경은 Pentium-Ⅳ 1.8GHz CPU와 메인 메모리 256MB, 윈도우즈 2000 시스템에서 C언어를 이용하여 구현한다. 성능 평가를 위해 각각 랜덤하게 생성된 10만 개의 랜덤 데이터 집합과 코렐 이미지에서 추출된 실제 데이터 집합을 사용한다. 제안하는 색인 구조의 검색 성능을 평가를 위해 사용되는 파라미터는 [표 1]과 같다.

표 1. 성능 평가를 위한 파라미터들

	파라미터
데이터 수	100,000 개
노드 크기	4 Kbytes
차원	16차원, 24차원, 32차원
데이터의 범위	각 차원은 0~1 사이의 실수 값

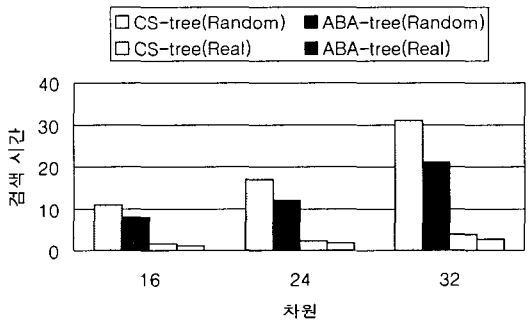


그림 8. 범위 검색에 대한 검색 시간

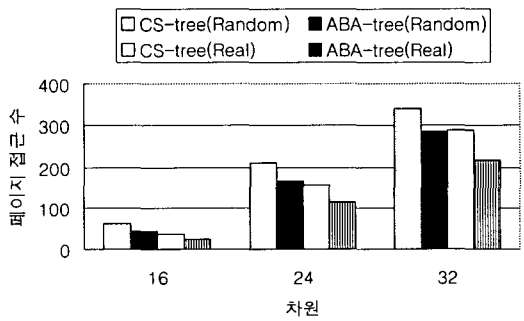


그림 9. K-최근접 검색에 대한 페이지 접근 수

2. 성능 평가

검색 성능을 평가하기 위해서 다차원 데이터의 유사도 검색에 대표적인 범위 검색과 k-최근접 검색을 50번 수행한 평균 값에 따라 페이지 접근 횟수와 검색 시간을 측정한다. [그림 7]과 [그림 8]은 범위 검색에 따른 페이지 접근 횟수와 검색 시간을 나타낸 것이다. [그림 9]와 [그림 10]은 k-최근접 검색에 따른 페이지 접근 횟수와 검색 시간을 나타낸 것이다.

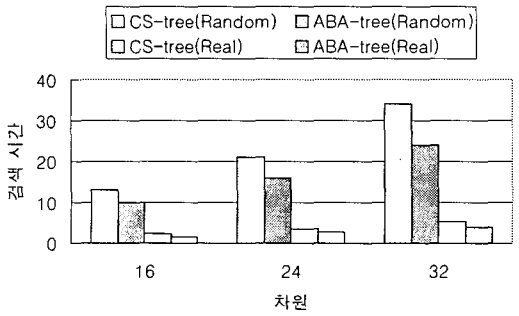


그림 10. K-최근접 검색에 대한 검색 시간

제안하는 색인 구조의 전체적인 검색 성능은 기존에 제안된 CS-트리보다 약 35~40%정도의 성능 향상을 보이고 있다. 제안하는 색인 구조는 공간 분할 방식을 수행하기 때문에 분할된 영역들 사이의 겹침 영역이 없으면서도 실제 데이터들이 존재하는 영역에 대해 근사화를 수행하기 때문에 검색 과정에서 탐색해야 할 노드의 수를 감소시켜 검색 성능을 향상시킨다.

Ⅵ. 결론 및 향후 연구

본 논문에서는 차원의 저주 현상을 해결하기 위한 벡터 근사 기반의 색인 구조를 제안하였다. 제안하는 색인 구조는 기존의 다차원 색인 구조의 문제점을 해결하기 위해 공간 분할 방식에 의해 분할을 수행하고 실제 데이터들이 존재하는 영역에 대해 BMBR이라는 근사화된 영역을 표현하였다. 따라서 분할된 영역들 사이에 겹침 영역이 발행하지 않으면서도 노드의 팬아웃을 증가시킬 수 있다. 또한 보다 정확한 BMBR 영역을 생성하기 위해 하위 노드에 대한 영역은 부모 노드를 기준으로 상대적으로 표현하며 실제 데이터들이 존재하는 영역에 대한 데이터의 분포 상태를 고려하여 동적인 비트를 할당한다. 이로 인해 분할된 영역들 사이에 선별력이 증가시켜 검색 성능을 향상시킬 수 있다. 제안한 색인 구조는 기존에 제안된 CS-트리보다 삽입 성능이 약간 저하되지만 검색 성능이 향상됨을 보였다.

향후 연구 방향으로 제안한 다차원 색인 구조에 대한 유사도 검색을 효과적으로 수행하기 위한 검색 기법과 벌크 연산 기법들에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] V. Gaede and O. Gunther, "Multidimensional Access Methods," ACM Computer Survey, Vol. 30, No. 2, pp.170-231, 1998.

[2] K. Chakrabarti and S. Mehrotra, "The Hybrid Tree : An Index Structure for High Dimensional Feature Spaces," Proc. the 15th International Conference on Data Engineering, pp.440-447, 1999.

[3] S. Berchtold, D. A. Keim and H. P. Kriegel, "The X-Tree : An Index Structure for High-Dimensional Data," Proc. 22nd International Conference on Very Large Data Bases, pp.28-39, 1996.

[4] R. Weber, H. J. Schek and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," Proc. 24rd International Conference on Very Large Data Bases, pp.194-205, 1998.

[5] K. T. Song, H. J. Nam and J. W. Chang, "A cell-based index structure for similarity search in high-dimensional feature spaces," Proc. the 2001 ACM Symposium on Applied Computing, pp.264-268, 2001.

[6] Y. Sakurai, M. Yoshikawa, S. Uemura and H. Kojima, "The A-tree : An Index Structure for High-Dimensional Spaces Using Relative Approximation," Proc. 26th International Conference on Very Large Data Bases, pp.516-526, 2000.

[7] Y. Sakurai, M. Yoshikawa, S. Uemura and H. Kojima, "Spatial indexing of high-

dimensional data based on relative approximation," VLDB Journal, Vol.11, No. 2, pp.93-108, 2002.

[8] J. T. Robinson, "The K-D-B-Tree : A Search Structure For Large Multidi -mensional Dynamic Indexes," Proc. the ACM SIGMOD International Conference on Management of Data, pp.10-18, 1981.

[9] A. Henrich, Hans-Werner Six and Peter Widmayer, "The LSD-tree : Spatial Access to Multidimensional Point and Nonpoint Objects," Proc. the Fifteenth International Conference on Very Large Data Bases, pp.45-53, 1989.

[10] A. Henrich, "The LSDh-tree: An Access Structure for Feature Vectors," Proc. the Fourteenth International Conference on Data Engineering, pp.362-369, 1998.

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

[12] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, "The R*-Tree : An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD International Conference on Management of Data, pp.322-331, 1990.

[13] K. L. Lin, H. V. Jagadish and C. Faloutsos, "The TV-tree-An index structure for high-dimensional data," the VLDB Journal, Vol. 3, No. 4, pp.517-542, 1994.

저 자 소 개

복 경 수(Kyoung-Soo Bok) 정회원

- 1998년 2월 : 충북대학교 수학과 (이학사)
- 2000년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2000년 3월~현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 데이터베이스시스템, 위치 기반 서비스, 이동 객체 데이터베이스, 멀티미디어 검색, 자료 저장 시스템, 다차원 색인 구조 등

허 정 필(Jeong-Pil Heo) 정회원

- 2002년 2월 : 충북대학교 정보통신공학과(공학사)
- 2004년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2004년 3월~현재 : 매크로임팩트(주) 시스템 소프트웨어 연구소 주임 연구원

<관심분야> : 데이터베이스시스템, 멀티미디어 검색, 다차원 색인 구조, 시스템 소프트웨어 등

유 재 수(Jea-Soo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1995년 2월 : 한국과학기술원 전산학과(공학박사)
- 1995년 3월~1996년 8월 : 목포대학교 전산통계학과 전임강사
- 1996년 9월~현재 : 충북대학교 전기전자컴퓨터공학부 부교수

<관심분야> : 데이터베이스시스템, 정보 검색, 멀티미디어 검색, 분산 객체 컴퓨팅 등