비정형 데이터를 위한 다차원 색인구조

송석일[○*] 피준일[*] 이석희[**] 유재수[*] 조기형[*]
* 충 북 대 학 교 정 보 통 신 공 학 과
** 동 아 방 송 대 학 인 터 넷 방 송 과
{prince, pji}@netdb.chungbuk.ac.kr seoklee@dab-c.ac.kr
{yjs, khjoe}@cbucc.chungbuk.ac.kr

A Multi-Dimensional Index Structure for Unformatted Data

Seok-Il Song[○*] Jun Il Pee[*] Seok Hee Lee[**] Jae Soo Yoo[*] Ki Hyung Joe[*]
* Dept. of Computer & Communication, Chungbuk National University
** Dept. of Internet Broadcasting, Dong-Ah Broadcasting College

요 약

최근 이미지나 멀티미디어 데이터와 같은 비정형 데이터의 검색을 보다 효과적으로 수행하기 위한 연구가 활발하게 진행되어 왔다. 비정형 데이터를 검색하기 위해서는 비정형 데이터를 다차원의 특징 벡터로 변환하고, 그것을 다차원 색인구조를 이용해 색인한다. 따라서 이러한 비정형 데이터를 효율적으로 색인 할 수 있는 다차원 색인구조가 요구되고 있다. 이 논문에서는 데이터를 벡터 근사치로 표현한 후 이를 트리 형태로 구성하여 검색의 효율을 높이는 다차원 데이터를 위한 색인구조 VA(Vector Approximate)-트리를 제안한다. 이 논문에서 제안하는 VA-트리는 VA-파일과 K-D-B-트리 구조를 기반으로 하고 있다. VA-트리는 적은 비트를 이용하여 다차원 공간을 표현하기 위해 노드내의 모든 정보를 비트로 표현한다. 중간노드의 비트 형태 엔트리는 하위노드에 포함된 정보를 의미하고 있어 탐색을 효율적으로 수행할 수 있도록 한다. 실험을 통한 성능평가를 수행하여 제안된 색인구조의 우수함을 보인다.

1. 서론

최근 대용량 네트워크 기술과 멀티미디어 처리기술의 발달로 사용자가 활용하고자 하는 음성, 영상, 동영상과 같은 비 정형 데이터의 양이 점차 증가함에 따라 이들의 효율적인 저장과 재사용이 중요한 문제로 대두되었다. 21세기의 모든 정보 통신망 사이에 주고받는 정보는 음성, 영상, 동영상과 같은 비 정형 데이터 요소가 포함될 것이며, 이에 따른 검색, 접근, 갱신이 이루어져야 한다. 이를 효과적으로 처리하기 위해서는 비정형 데이터가 데이터베이스에 의해 체계적으로 관리되어야 하며, 특히 필요한 데이터를 손쉽게 효율적으로 검색할 수 있는 방법이 있어야 한다.

이 논문은 데이터를 벡터 근사치로 표현한 후 이를 트리 형태로 구성하여 검색의 효율을 높이는 다차원 데이터를 위한 색인구조 VA(Vector Approximate)-트리를 제안한다. 이 논문에서 제안하는 VA-트리는 VA-파일과 K-D-B-트리 구조를 기반으로 하고 있다. VA-트리는 적은 비트를 이용하여 다차원 공간을 표현하기 위해 노드내의 모든 정보를 비트로 표현한다. 중간노드의 비트 형태 엔트리는 하위노드에 포함된 정보를 의미하고 있어 탐색을 효율적으로 수행할 수 있도록 한다. 실험을 통한 성능평가를 수행하여 기존의 방법보다 제안된 방법이 다차원 데이터에 대한 검색 성능이 우수함을 보인다.

이 논문의 구성은 다음과 같다. 제2장의 관련 연구에서는 기존에 제시된 다차원 색인구조들에 대해서 기술하였다. 제3장은 새로운 다차원 색인구조로써 제안하고 있는 VA-트리를 기술한다. 먼저, VA-트리의 설계에서 고려하고 있는 설계 특징과 제안한 VA-트리의 알고리즘에 대해서 기술하였다. 그리고 제4장은 제안한 VA-트리의 성능을 기존의 색인구조와 비교하고 분석한 결과를 기술한다. 마지막 제5장의 결론에서는 논문의 성과와 향후 연구 방향을 제시한다.

2. 관련 연구

최근 중요한 응용들로 주목받고 있는 지리 정보 시스템, 캐드 데이터베이스, 의료 데이터베이스, 내용기반 이미지 검색 시스템, 멀티미디어 데이터 베이스 등에서 다차원 색인구조는 매우 중요한 비중을 차지한다. 다차원 색인구조의 역할은 대용량의 다차원의 특징을 갖는 데이터들을 색인하여 사용자가 원하는 데이터를 효과적으로 검색할 수 있게 한다. 다차원 색인구조의 중요성은 이미 여러 해 전에 인식되었고 이에 대한 연구가 매우 활발히 진행되어 왔다.

현재 제안된 다차원 색인 구조들은 R*-트리[1], TV-트리, X-트리[2], SS-트리, SR-트리와 같은 데이터 분할을 사용하는 색인 구조와 KDB-트리[3], hB-트리, LSDh-트리, BANG 파일, GRID 파일과 같이 공간분할을 사용하는 색인 구조들이 있다. 또한, 이들의 혼합 형태의 색인구조로 Hybrid-트리가 존재하며 이 외에도 LS(Locality Sensitive)해싱기법을 사용하는 색인구조와 VA-파일[4]와 IQ-트리처럼 요약 기법을 사용하는 색인구조도 존재한다.

3. VA-Tree : 새로운 다차원 색인구조

1) 설계 개요

VA-트리는 특징벡터를 벡터 근사치로 매핑시킨 후, 이 벡터 근사치를 이용하여 VA-트리를 구성한다. K-D-B트리 기반의 다른 색인 구조처럼 노드간 겹침은 없다. 노드는 중간노드와 단말노드로 구분된다. 중간노드 구조는 [하위노드를 포함하는 영역 비트, 하위노드에 대한 포인터], 단말노드 구조는 [벡터 근사치, 실제 데이터 레코드에 대한 포인터]로 구성된다. 중간노드는 하위 중간노드 또는 단말노드를 포함하는 영역 정보를 비트 형태로 가지고 있다. VA-트리에서는 유사도 탐색시 최소거리 경계만을 사용한다. 중간노드의 영역 비트 정보만을 보고

질의 포인트에서 중간노드까지의 최소 거리 경계를 계산한다. 또한 단말노드의 벡터 근사치를 이용하여 질의 포인트에서 벡터 근사치까지의 최소 거리 경계를 계산한다. 이러한 최소 거리 경계를 이용하여 효과적인 가지치기를 수행할 수 있으므로 빠른 검색 성능을 보인다.

2) VA-Tree의 알고리즘

(1) 삽입 알고리즘

새로운 특징벡터를 삽입하기 위하여 새 특징벡터를 벡터 근사치로 매핑시킨다. 벡터 근사치를 포함하는 가지들을 선택하면서 탐색하여 단말노드에 도달하면 그 노드에 벡터 근사치를 삽입한다. 삽입시 단말노드에 오버플로우가 발생되면 단말노드 분할 알고리즘에 따라 분할하고, 중간노드의 오버플로우는 중간노드 분할 알고리즘에 의해 처리한다.

```
Procedure Insert

Start procedure

1 입력 특징벡터(O)를 벡터근사치(V)로 변환
2 Leafnode := Newent를 삽입할 단말 노드를
  찾는다(LocateNode 호출);
3 If (LeafNode.entnum == OVERFLOW )
4         분할을 수행한다 ( SplitNode 호출 );
5 else
6         LeafNode에 NewEnt를 삽입;
7         LeafNode.EntNum++;
8 End if

End procedure
```

그림 1 삽입 알고리즘

(그림1)에서 1행은 특징벡터(O)를 벡터 근사치(V)로 변환한다. 2행은 벡터 근사치를 삽입할 단말노드를 LocateNode함수를 이용하여 검색한다. 3~7행은 새로운 벡터 근사치를 삽입한 단말노드에 오버플로우가 발생하면 분할을 수행하기 위해 SplitNode 함수를 호출하고, 그렇지 않은 경우에는 단말노드에 새로운 벡터 근사치를 삽입한 후 노드의 엔트리 수를 하나 증가시킨다.

(2) 검색 알고리즘

VA-트리에서 유사도 검색은 다른 색인 구조와 마찬가지로 루트부터 시작하여 단말노드까지 각 노드별로 엔트리들을 검사하여 수행하는데, 이 논문에서는 [5]에서 사용한 다단계 최근접 점 탐색 방법을 이용한다. 질의 포인터에서 가장 가까운 K개의 유사 객체를 찾는 K-근접 질의 검색을 생각해보면, 우선 질의 포인트를 벡터 근사치로 매핑시킨다. 트리를 순회하면서 중간노드의 비트 정보만을 이용하여 질의 포인트에서 중간노드까지의 최소 거리 경계와 질의 포인트에서 단말노드내의 벡터 근사치까지의 최소 거리 경계를 계산하여 효과적으로 가지치기할 수 있다. 거리가 가까운 벡터 근사치를 가진 실제 데이터들을 읽어와 실제 거리를 계산한다. K번째 실제 거리보다 가까운 근사 거리가 없을 때 탐색을 끝내고 K개를 반환한다.

유사도 검색을 위해 (그림 2)에서와 같이 후보집합, 결과집합 우선 순위 큐 두개를 사용한다. 후보집합에는 질의 포인트에서 벡터 근사치나 중간노드까지의 근사 거리 순으로 정렬되고, 결과집합에는 질의 포인트에서 실제 데이터까지의 거리 순으로 정렬된다. 결과집합의 k번째 거리를 나타내는 K번째 거리 값(result_k_dist)은 초기치 ∞에서 점점 작아진다. 루트부터 트리를 순회하면서 질의 포인트에서 중간노드까지의 거리를 계산하여 후보집합에 넣는다. 3행은 질의 포인트부터 지금까지 방문한 K번째 데이터까지의 실제거리가 후보집합내의 거리보다 작으면 탐색을 끝내고 결과집합에서 k개를 반환한다. 4~6행은 후보집합에서 읽은 데이터가 중간노드이면 그 중간노드내 엔트리까지의 근사 거리를 계산하여 후보집합에 저장한다. 7~9행은 후보집합에서 꺼낸 데이터가 단말노드이면 그 단말노드내의 벡터 근사치까지의 거리를 계산하여 후보집합에 저장한다. 10~12행은 후보집합에서 읽은 데이터가 벡터 근사치이면 질의 포인트와의 거리를 계산하여 결과집합에 저장하고, k번째 거리 값을 갱신한다.

```
Procedure Search

Start procedure

1  결과집합, 후보집합, K번째 거리 = ∞
2  질의포인트 ==> 비트근사치로 변환
3  while ( K번째 거리 > 후보집합 첫번째 거리)
4      만일 후보집합 첫번째가 중간노드라면
5          중간노드내의 엔트리까지의 거리 계산
6          K번째 거리보다 작은 엔트리를 후보집합에 삽입
7      만일 후보집합 첫번째가 단말노드라면
8          단말노드내의 엔트리까지의 거리 계산
9          K번째 거리보다 작은 엔트리를 후보집합에 삽입
10     만일 후보집합 첫번째가 비트근사치라면
11         질의에서 비트근사치까지의 거리 계산
12         결과집합에 삽입
13         K번째 거리를 갱신
14 결과집합에서 K개를 반환

End procedure
```

그림 2 검색 알고리즘

4. 실험 및 결과분석

이 절에서는 제안된 VA-트리를 구현하여 실험한 성능평가 결과를 기술한다. 이 논문에 제시된 모든 실험결과는 Solaris 2.7 운영 체제에 Sun Enterprise 3000, 메인메모리 1GB를 장착하고 있는 시스템을 이용하여 C언어로 구현하였다. 컴파일러는 gcc 2.7.1.이다.

이 논문에서 비교평가 대상으로 사용된 기존의 색인구조는 논문의 저자가 제공해준 원시 프로그램의 내용을 수정하지 않고 사용하였다. 한 차원의 벡터 데이터를 표현하기 위한 비트수는 VA-파일에서 4~5비트일 때 가장 좋은 성능을 보이므로 성능 비교를 위하여 4비트로 하였다. 실험데이터는 동영상에서 각각의 프레임을 하나의 이미지로 간주하고 여기에서 추출된 칼라 정보 값을 9개의 수치로 표현한 실제 데이터와 난수발생기를 이용하여 균일 분포를 갖는 실수형 데이터 집합을 만들어 사용하였다. 모든 실험 데이터와 생성되는 인덱스는 같은 디스크 내에 존재하고, 색인을 위해 사용되는 페이지의 크기는 다양한 환경에서의 성능측정을 위해 4Kbytes, 8Kbytes, 16Kbytes, 32Kbytes로 변경하면서 실험하였다.

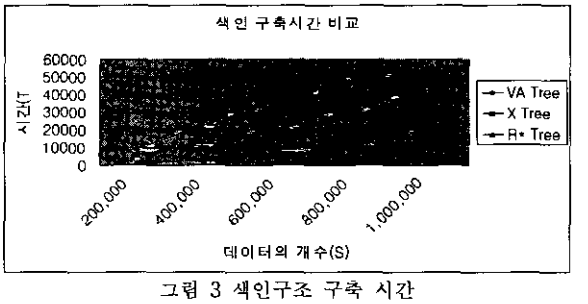

그림 3 색인구조 구축 시간

(그림 3)은 VA-트리와 R*-트리, X-트리의 삽입 시간을 비교한 결과이다. (그림 3)은 노드크기를 32kB, 데이터 차원을 10차원, 데이터 개수를 200,000개에서 1,000,000개까지 늘려가면서 색인을 구성할 때 걸리는 시간을 측정하였다. 측정 결과 VA-트리를 구축하는데 가장 적은 시간이 소요됨을 확인할 수 있었다. 특히, 데이터 개수가 늘어남에 따라 색인 구축시간의 차이가 점점 커지는 것을 확인할 수 있었다. 삽입 알고리즘은 가장 복잡한 단계를 거친다. 그러므로 삽입 시간이 빠르다는 것은 삭제 알고리즘과 같은 다른 연산의 속도가 빠르다는 것을 의미한다고 볼 수 있다.

(그림 4)과 (그림 5)는 균일 분포 데이터 집합들을 이용하여 VA-트리와 VA-파일, R*-트리, X-트리에서 K-NN 질의를 수행한 경우에 대한 검색 성능을 비교한 것이다. Fig. 3.27은 노드크기를 4kB, 데이터 개수를 100,000개, 차원을 4에서 20까지 늘려가면서 색인을 구성한 후, 주어진 질의 데이터에 대해서 10개의 근접 데이터를 100번 검색했을 때 접근되는 평균 접근 노드 수를 측정한 결과이다. 모든 경우에 있어서 기존의 색인구조보다 우수한 검색성능을 나타냄을 알 수 있다. VA-트리는 벡터 근사치를 이용하여 색인을 구성하기 때문에 생성되는 노드의 수가 기존의 색인구조에 비해 현저하게 적고 중간노드의 영역정보가 오버랩이 없다. 이런 이유로 접근하지 않는 노드의 수가 많아지게 되는 것이다.

(그림 5)는 노드크기를 4kB, 차원을 10차원으로 고정시키고, 데이터 개수를 100,000개에서 1,000,000개까지 늘려가면서 색인을 구성한 후, 주어진 질의 데이터에 대해서 10개의 근접 데이터를 100번 검색했을 때 접근되는 평균 접근 노드 수를 측정한 결과이다. VA-파일은 모든 비트 데이터를 순차 검색하기 때문에 데이터가 많아질수록 검색시간이 매우 증가한다. 반면 VA-트리에서는 데이터 집합이 대용량이라도 접근하는 노드수가 거의 일정한 것을 알 수 있다. 이 결과에 따르면 VA-트리는 기존의 색인구조에 비해 성능이 우수함을 보인다.

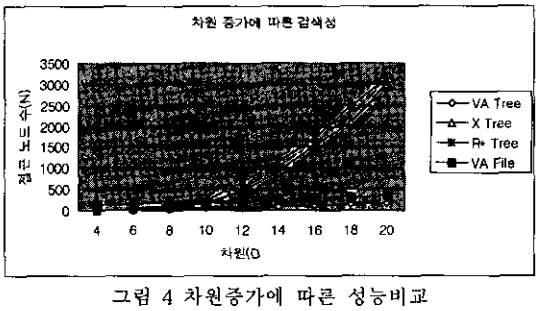

그림 4 차원증가에 따른 성능비교

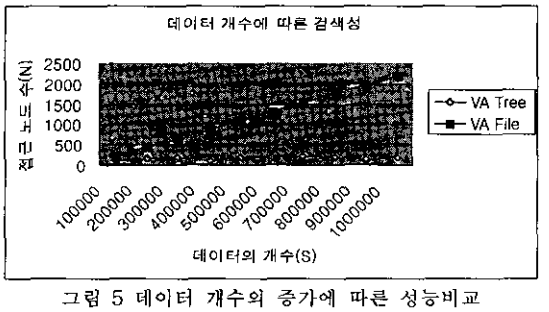

그림 5 데이터 개수의 증가에 따른 성능비교

(그림 6)은 노드크기를 4kB, 차원을 9차원, 데이터 개수를 300,000개의 실제 데이터를 이용해 색인을 구성한 후, 주어진 질의 데이터에 대해서 10개의 근접 데이터를 100번 검색했을 때 접근되는 평균 접근 노드 수를 측정한 결과이다. 데이터는 동영상의 각각 장면에서 장면을 특징지우는 색상정보를 9개의 수치 데이터로 추출하여 사용하였다. 실제 데이터에 대한 성능 측정 결과에서도 VA-트리는 기존의 색인구조에 비해 성능이 우수함을 보였다.

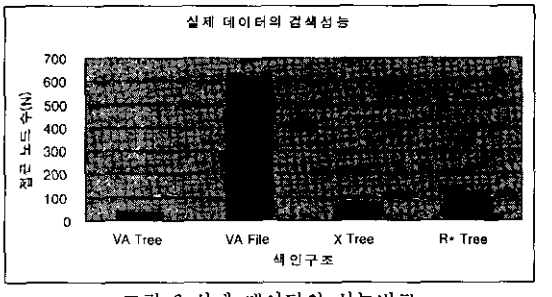

그림 6 실제 데이터의 성능비교

5. 결론

제안하는 색인구조는 차원의 증가에 따른 특징 벡터의 증가를 최대한 줄이기 위해 특징 벡터를 비트열 형태의 벡터 근사치로 표현한 후 이를 이용하여 색인을 트리 형태로 구성한다. 비트열 형태의 벡터 근사치로 표현함으로써 단말노드와 중간노드에서 특징벡터를 표현하는 엔트리의 크기가 작아지는 효과를 얻을 수 있다. 이로써 차원이 증가함에 따라 이를 표현하는 특징벡터의 크기가 증가되는 문제점을 해결하였다. 또한 공간을 기준으로 분할을 수행함으로써 노드간의 겹침영역이 전혀 없으면서 상위노드의 분할이 하위노드에 영향을 미치지 않도록 하는 방법을 제안하였다. 그리고 중간노드에서 분할된 공간을 표현할 때 실제 데이터가 존재하는 영역만을 알 수 있도록 하한값과 상한값을 의미하는 벡터근사치로 표현함으로서 검색시에 필요없는 노드를 접근할 수 있는 가능성을 최소로 줄였다. 실험을 통한 성능평가를 수행하여 기존의 방법보다 제안된 방법이 다차원 데이터에 대한 검색 성능이 우수함을 증명하였다.

향후 연구방향으로 제안된 색인구조를 멀티미디어 데이터베이스 시스템, 이미지 검색 시스템과 같은 실제 응용에서 사용될 수 있도록 하는 연구가 필요하다. 제안된 색인 구조가 실제로 상용 DBMS에 통합되어 사용되기 위해서는 그 색인 구조에 맞는 동시성 제어 및 회복 기법이 필수적이다.

6. 참고문헌

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

[2] S.Berchtold, D.A.Keim, H-P.Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," Very Large Data Bases(VLDB'96), Proc. 22nd., pp. 28-39, 1996.

[3] J.T.Robinson, "The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes." ACM SIGMOD, pp. 10-18, Apr. 1981.

[4] Weber R., Scheck H.-J, Blott S., " A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces ", 1998

[5] Thomas Seidl, Hans-Peter Kriegel, "Optimal Multi-Step K-Nearest Neighbor Search", ACM SIGMOD, pp. 154-165, June 1998