MiDAS-Ⅲ에서 효율적인 이미지 검색을 위한 CIR-트리 관리기의 설계 및 구현

(Design and Implementation of a CIR-Tree Manager for Efficient Image Retrieval on MiDAS-Ⅲ)

송 석 일 † 이 희 종 †† 이 석 희 †
(Seok Il Song) (Hee Jong Lee) (Seok Hee Lee)
유 재 수 ††† 조 기 형 ††† 유 관 희 ††††
(Jae Soo Yoo) (Ki Hyung Cho) (Kwan-Hee Yoo)

요 약 현대 사회는 이미지 데이타의 홍수라 해도 과언이 아닐 정도로 이미지 데이타는 기하 급수적으로 증가하고 있다. 이렇게 방대한 양으로 증가하는 이미지 데이타를 효과적으로 관리하기 위해서는 이미지 데이타를 위한 고차원 색인구조가 필요하다. 그러나 아직 국내에서는 상용 DBMS(Database Management System)에서 이러한 색인구조를 지원한 예가 없다. 이 논문에서는 고차원 색인구조인 CIR-트리를 국내에서 개발한 바다-Ⅲ DBMS의 하부 저장시스템인 MiDAS-Ⅲ 에서 설계하고 구현하여 보다 효과적으로 이미지 데이타를 관리할 수 있도록 한다. 이 논문에서 구현한 CIR-트리 관리기를 순차검색과의 비교를 통해 성능을 입증한다.

Abstract Nowadays, the amount of image data increase explosively. To manage the large amount of image data efficiently, high-dimensional index structures are necessary. However, as my knowledge none of existing DBMSs supports high-dimensional index structures as access methods of DBMSs. In this paper, we design and implement CIR-Tree as a access method for retrieving image data effectively on the MiDAS-Ⅲ that is the storage subsystem of the BADA-Ⅲ. The implemented CIR-Tree manager shows much better retrieval performance than sequential search in performance evaluation.

1. 서 론

최근 들어 멀티미디어 기술의 발달이 가속화되면서 멀티미디어 데이타에 대한 이용이 폭발적으로 증가해 왔다. 특히, 이미지 데이타에 대한 이용은 여러 분야에 걸쳐 폭넓게 분포되어 있다. 이들 분야에서 사용되는 이미지 데이타의 양은 이미 사람이 관리하기에는 불가능할 정도로 증가해 버렸고 또 현재 증가하고 있다. 따라서, 사용자들이 대용량의 이미지들 중 원하는 이미지를 정확하고 빠르게 검색할 수 있도록 하는 이미지 검색시스템이 요구된다. 이미지 데이타는 기존의 텍스트 데이타에 비해 대용량이라는 특성과 비정형적인 특성을 가지고 있어 기존의 데이타와는 다른 구조의 색인을 필요로 한다.

이미지 데이타를 검색하는 대표적인 방법으로 내용기반 이미지 검색을 들 수 있다. 이미지 데이타는 비정형의 데이타이다. 그러므로 비정형 이미지 데이타를 정형 데이타로 변환을 해 주어야 한다. 이러한 과정을 이미지에 대한 특징 추출 과정이라고 한다. 내용기반 이미지

· 본 연구는 한국과학재단 특정기초연구(과제번호 · 1999-1-303-007-3) 지원으로 수행되었음.
† 비 회 원 : 충북대학교 정보통신공학과
prince@netdb.chungbuk.ac.kr
seoklee@pretty.chungbuk.ac.kr
†† 비 회 원 : (주)넥스텍 연구개발부 연구원
lhj0271@netsgo.com
††† 종신회원 : 충북대학교 전기전자및컴퓨터공학부 교수
yjs@cbucc chungbuk.ac.kr
khjoe@cbucc.chungbuk.ac.kr
†††† 종신회원 : 충북대학교 컴퓨터교육과 교수
khyoo@cbucc.chungbuk.ac.kr
논문접수 : 1999년 10월 20일
심사완료 : 2001년 7월 10일

검색은 자동으로 이미지가 갖는 질감, 색상, 형태 등의 특징벡터를 추출해 내는 특징 추출 과정을 수행한 후, 추출된 특징벡터를 이용하여 이미지를 검색하는 방법을 말한다.[1, 2, 3] 내용기반 이미지검색을 위한 시스템 구조는 그림 1과 같으며, 이미지 특징 추출, 유사성 기반의 검색, 시각 질의를 지원하는 인터페이스, 효율적인 색인구조 등을 지원해야 한다.

이미지에서 얻은 특징벡터는 앞에서 언급한 바와 같이 이미지의 특징에 대한 질감, 색상, 형태 등 많은 수의 차원을 갖기 때문에 내용기반 이미지 검색을 위한 색인 구조는 기존의 색인 구조와는 달리 고차원 데이타에 대해 효율적인 색인기법을 제공해야 한다. 이러한 목적으로 개발된 색인 구조로 R-트리 계열의 구조들을 들 수 있다. 대표적인 R-트리 계열의 색인 구조로는 R-트리[4]를 비롯하여 R*-트리[5], R+-트리[6], X-트리[7], TV-트리[8], SS-트리[9], SR-트리[10] 등이 있다. 이러한 R-트리 계열의 구조들은 계속 발전되어져 왔고 가장 최근의 색인구조로 CIR-트리[11, 12]가 제안되었다.

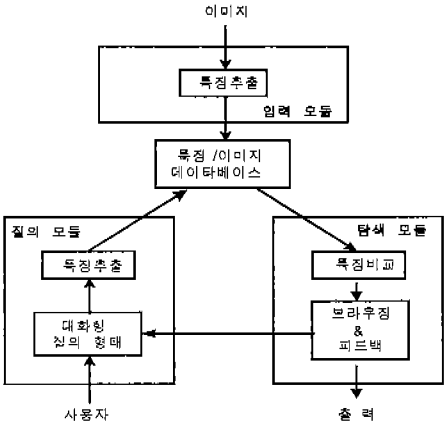

그림 1 내용기반 이미지검색 시스템 구조

기존 이미지 검색시스템들은 일반적으로 화일시스템에서 구현되어 있거나 기존의 DBMS 위에서 사용자 프로그램 수준에서 구현되어있어 전체적인 수행성능이 낮을 수밖에 없다. 따라서, 효율적인 이미지 검색시스템을 구축하기 위해서는 DBMS의 하부 저장 시스템을 내용기반 이미지 검색을 위한 색인구조를 수용할 수 있도록 확장해야 한다. 그러나, 현재 국내에선 내용기반 이미지 검색을 지원하는 상용 데이타베이스 시스템이 구현되어 있지 않다. 이 논문에서는 국내 기술로 개발된 멀티미디어 DBMS인 바다-Ⅲ의 하부 저장 시스템(MiDAS-Ⅲ)가 내용기반 이미지 검색을 효율적으로 지원할 수 있도록 고차원 색인 구조인 CIR-트리를 설계하고 구현한다.

이 논문의 구성은 다음과 같다. 2장은 관련연구로서 CIR-트리의 특징과 MiDAS-Ⅲ의 전체적인 구성에 대해 기술하고, 3장에서는 MiDAS-Ⅲ 기반에서 CIR-트리 관리기를 설계하고 구현한 내용에 대해서 설명한다. 4장에서는 성능평가를 통해 이 논문에서 구현한 CIR-트리 관리기의 타당성을 검증하며 5장에서 결론을 맺는다.

2. 관련연구

CIR(Content based Image Retrieval)-트리는 내용기반 이미지 검색을 지원하기 위해 제안된 고차원 색인구조이다.[10, 11] 이 색인구조는 기존의 R-트리 계열의 고차원 색인구조들의 문제점인 차원이 증가함에 따라 급격히 색인구조의 성능이 떨어지는 차원의 저주현상을 개선하고 있다. CIR-트리의 특징을 크게 3가지로 정리할 수 있다. 첫째, 단말노드를 제외한 모든 노드들이 분별력이 있는 특징만을 선정하여 노드를 구성함으로써 효율적으로 고차원의 이미지 특징들을 수용한다. 둘째, 겹침영역을 최소화하기 위한 노드의 분할 방법을 제공하며 슈퍼노드 개념을 도입했다. 셋째, 이미지 특징들의 군집화를 위해 무게 중심을 이용하는 보다 개선된 재삽입을 수행한다. 그림 2는 CIR-트리의 전체적인 구조이다.

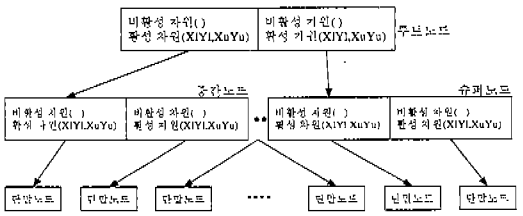

그림 2 CIR-트리의 구조

일반적으로 DBMS는 크게 두 부분으로 나눌 수 있다. 사용자 인터페이스라고 할 수 있는 데이타베이스 언어 처리기 부분과 데이타를 저장하고 트랜잭션을 관리하는 저장시스템이다.[13] 이 중 데이타의 저장 및 트랜잭션 관리의 역할을 하는 것이 바로 MiDAS-Ⅲ 이다. 앞서 언급 한데로 MiDAS-Ⅲ는 바다-Ⅲ의 하부 저장 시스템으로서 데이타의 저장, 접근, 동시성 및 회복 등을 지원한다.[14] 그림 3에서 MiDAS-Ⅲ 의 전체적인 구성을 보여주고 있다. 데이타를 처리하는 부분은 디스크에 대한 데이타의 입출력과 메모리 관리 그리고 색인관리, 커서 관리, IR-Index 관리 등이다.

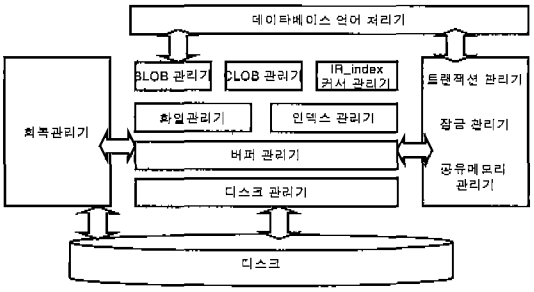

그림 3 MiDAS-Ⅲ 시스템 내부 구성도

트랜잭션 관리는 잠금 관리, 작업 관리, 회복 관리기가 수행한다. MiDAS-Ⅲ에서 사용되는 저장단위로는 페이지(page), 익스텐트(extent), 화일(file), 세그먼트(segment), 볼륨(volume)등이 있다. 페이지는 가장 작은 단위로서 디스크와 공유 메모리 사이의 입출력 단위가 된다. 익스텐트는 물리적으로 인접한 몇 개의 페이지를 묶은 것으로 디스크 영역을 화일에 할당하거나 반환할 때 기준이 되는 단위이다. 화일은 몇 개의 익스텐트로 구성되는 논리적인 개념으로써 레코드들이 들어가는 공간이다. 세그먼트는 물리적인 영역인데 디스크 파티션의 연속된 일부 혹은 전부인 디스크 영역이거나 일반 UNIX 화일이다. 볼륨은 하나 또는 여러 개의 세그먼트로 이루어지는 논리적인 개념으로 각 화일은 임의의 볼륨에 속하게 된다.

3. CIR-트리 관리기의 설계 및 구현

3.1 CIR-트리 관리기의 구조

이해를 돕기 위해 실제적인 CIR-트리 관리기의 구성에 대해 설명하기 전에 먼저 MiDAS-Ⅲ의 계층적 구조와 CIR-트리 관리기가 계층구조의 어느 부분을 차지하는가에 대해서 설명한다. MiDAS-Ⅲ 시스템은 데이타의 입력 및 검색을 위해 커서를 사용한다. 이 커서를 통해서 색인이나 데이타 화일을 접근할 수 있도록 한다. 하지만 CIR-색인 관리기는 질의와 부합하는 문서를 계산하는 단계를 거쳐야 하므로 별도의 커서를 두고 있으며 커서 관리기와 동등한 위치에 존재한다. 그림 4는 MiDAS-Ⅲ에서 CIR-트리 관리기와 다른 관리기들과의 관계를 보여주고 있다. CIR-트리 관리기는 화일 관리기를 통해 데이타 화일에 접근하고 버퍼 관리기와 디스크 관리기들과 상호 작용하며 일을 수행한다.

그림 5는 MiDAS-Ⅲ에서 구현한 CIR-트리의 구성과정을 나타낸다. 이미지 데이타의 특징 벡터는 이미지 검색 엔진에서 이미지 전처리 과정을 거쳐 추출되고 이것

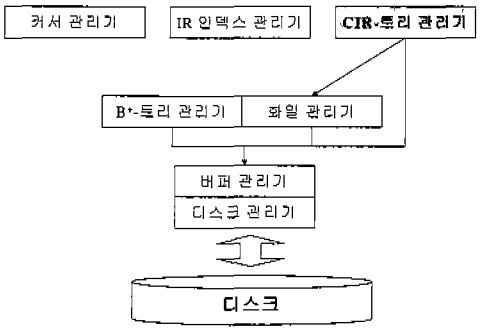

그림 4 다른 관리기와 CIR-트리 관리기의 관계

을 다시 데이타 화일 내 레코드의 한 필드에 저장한다. CIR-트리를 구축하기위한 입력데이타는 이미지 데이타와 특징벡터가 저장되어 있는 데이타 화일의 레코드에 대한 식별자이다. 레코드 식별자와 함께 레코드내의 특징벡터 필드를 설명해 주는 필드설명자가 주어지면 데이타 화일로부터 특징벡터를 추출해내고 이 특징벡터가 레코드 식별자와 함께 CIR-트리의 단말노드 엔트리를 형성하고 이를 트리에 삽입한다. 구현한 CIR-트리에서는 특징벡터를 적절한 정수로 변환하여 사용한다. 특징벡터가 분포할 수 있는 범위와 이미지 데이타의 개수등을 적절히 고려하여 특징벡터를 일정 범위의 정수로 변환하게 된다. 실제 CIR-트리에 삽입되는 것은 이 정수형태로 변환된 특징벡터이다. 검색시에도 같은 방법으로 질의 특징벡터를 정수로 변환하여 검색을 수행한다.

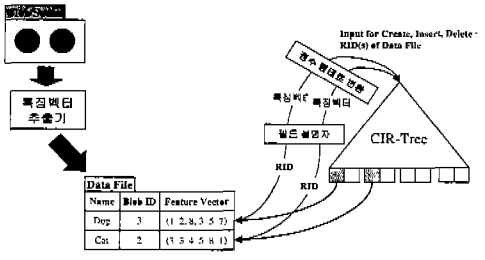

그림 5 MiDAS-Ⅲ에서 CIR-트리의 구성

MiDAS-Ⅲ에서 CIR-트리는 각각 색인설명자를 갖는다. 색인설명자에는 CIR-트리 구축에 사용된 특징벡터의 차원 수, 트리의 깊이, 필드 설명자 등의 정보들이 포함된다. 색인 설명자는 MiDAS-Ⅲ의 색인 카탈로그 화일에 저장되게 되는데 이때 이 색인 설명자의 레코드 식별자가 구축된 CIR-트리를 유일하게 구분해주는 색인 식별자가 된다. 그림 6에서 이를 보여준다. CIR-트리의

개방(open)은 색인식별자로부터 시작한다. 엔트리의 삽입, 삭제, 탐색 시에는 먼저 색인식별자를 통해 색인 카탈로그 화일로부터 색인설명자를 읽어오고 색인설명자를 바탕으로 커서를 생성한다. 커서에는 CIR-트리의 루트노드, 결과 집합, 필드설명자 등을 유지하며 이 정보들은 실제로 탐색을 하거나 새로운 엔트리를 삽입하거나 또는 엔트리를 CIR-트리에서 삭제할 때 사용된다.

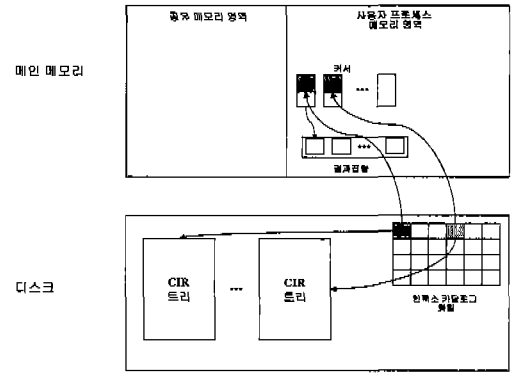

그림 6 CIR-트리 관리기의 구성도

3.2 주요 데이타 구조

CIR-트리의 개방은 색인 카탈로그 화일의 색인설명자가 저장된 레코드식별자로 이루어진 색인식별자로부터 시작한다. 엔트리의 삽입, 삭제, 탐색 시에는 먼저 색인식별자를 통해 색인 카탈로그 화일로부터 색인 설명자를 읽어 오고 색인 설명자를 바탕으로 커서를 생성하게 된다. 커서에는 CIR-트리의 루트노드, 결과 집합, 필드설명자 등을 유지하며 이 정보들은 실제로 탐색을 하거나 새로운 엔트리를 삽입하거나 엔트리를 CIR-트리에서 삭제할 때 사용된다. MiDAS-Ⅲ에서는 색인 구조가 만들어질 때 색인에 대한 설명자를 카탈로그 화일에 기록하는데 CIR-트리의 색인설명자의 구조는 그림 7과 같다.

```
typedef  struct {
   T_FID                BaseFileID;
   T_FID                FileID;
   T_PID                RootPage;
   short                Depth;
   float*               (*transform)();
   T_CIRFIELDDESC       KeyDesc;
   short                TotalDim;
   short                ChosenDim;
   short                ActiveDim;
} T_CIRIDXDESC;
```

그림 7 색인설명자의 구조

색인설명자는 위에서부터 순서적으로 색인이 만들어진 데이타 화일의 식별자, 색인 화일의 식별자, 색인 화일의 루트노드의 페이지 식별자, 트리의 높이, 사용되는 변환함수, 데이타 화일상의 키 설명자, 전체 차원의 개수, 변환함수에 의해 선택되어진 차원의 개수, 활성차원의 개수가 기록이 된다.

페이지는 MiDAS-Ⅲ의 데이타 입출력의 기본 단위이며 디스크 관리기와 버퍼 관리기에서 페이지 단위로 입출력이 이루어진다. MiDAS-Ⅲ에는 현재 B+-트리 색인 관리자를 위한 색인 페이지와 리프 페이지, 화일 관리자를 위한 디렉토리 페이지, 그리고 카탈로그 페이지 종류가 있다. 이러한 페이지들은 각각 사용 목적에 따라 정의되어 있다. CIR-트리 관리기를 위해서 역시 CIR-트리를 위한 페이지 타입이 필요하다. 그림 8은 CIR-트리의 페이지 구조를 보여준다. 페이지의 데이타 타입은 MiDAS-Ⅲ에서 사용하는 데이타 페이지의 기본적인 규칙을 그대로 따르고 있으며 CIR-트리만을 위한 정보가 추가되어 있다. NotUsedMaxRsc 이하의 5개의 데이타 구조는 MiDAS-Ⅲ에서 기본적으로 사용되는 데이타 구조로 NotUsedMaxRsc, FileID, PageLSN는 값은 유지하지만 구현에서는 사용되지 않는다. PageType은 CIR-트리 색인 관리기에서 사용되는 페이지의 형태를 나타내는 식별자이다. CIR-트리 색인 관리기에서 사용되는 페이지의 형태로는 CIRROOT, CIRSUPER, CIRINTERNAL, CIRLEAF의 네 가지 형태가 있다. This Page는 현재 페이지의 식별자를 나타낸다.

Data는 CIR-트리 관리자가 유지하는 실제 엔트리의 데이타를 위한 영역이고 Slot은 엔트리의 위치를 나타내는 역할을 하고 Free는 사용될 수 있는 여유공간의 시작위치를 NofNonsig는 노드에 비 활성차원의 수를 유

```
Typedef      struct {
   char      Data[DSKPAGESIZE-CPFIXED];
   short     Slot[1];
   short     RIDCnt;
   short     Free;
   short     NofNonsig;
   short     Level;
   T_PID     NextSuper;
   T_PID     NextPage;
   T_NSN     PageNSN;
   int       NotUsedMaxRsc;
   T_FID     FileID;
   int       PageType;
   T_PID     ThisPage;
   T_LSN     PageLSN;
} T_CIRPAGE;
```

그림 8 CIR-트리 페이지 구조

지함으로써 페이지에 대한 추가적인 계산을 줄여준다. Level은 CIR-트리 관리기에서 노드가 위치한 레벨 정보를 표시하고, NextSuper는 노드가 슈퍼노드일 경우 다음 노드를 찾아갈 수 있는 포인터를 저장하고 있다.

3.3 CIR-트리의 주요 알고리즘 구현

3.3.1 CIR-트리 생성 알고리즘

CIR-트리 관리기의 색인생성 과정은 그림 9와 같다. 우선 데이타 화일에 색인을 구성할 특징벡터가 존재하는지를 조사하여 데이타 화일이 비어 있다면 빈 색인을 생성한다. 그렇지 않으면 필드설명자를 이용해 데이타 화일로부터 특징벡터를 추출한다. 추출된 특징벡터를 데이타 화일의 레코드식별자와 함께 BulkFile에 저장하게 된다. 이때 만들어지는 BulkFile은 임시 화일로서 작업이 종료되면 자동으로 삭제된다.

생성된 BulkFile을 이용해서 데이타를 하나씩 CIR-트리에 삽입한다. 삽입이 모두 끝나면 색인에 대한 정보를 색인설명자에 유지하고 이르 색인 카탈로그 화일에 저장한 후 색인설명자의 레코드식별자를 색인식별자로 반환하고 색인생성을 종료한다.

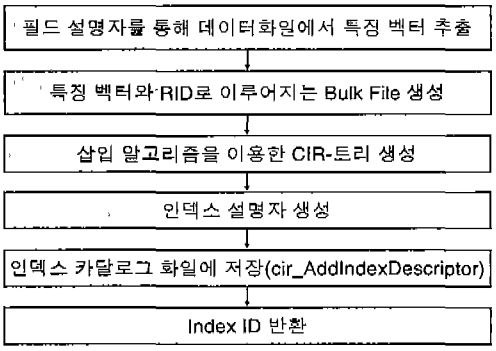

그림 9 CIR-트리 생성과정

3.3.2 삽입 알고리즘

CIR-트리에서 삽입은 우선 삽입할 단말노드까지의 경로를 구하고 삽입된 엔트리를 기록한 다음 상위노드에 변경된 MBR을 반영하는 과정을 루트노드까지 반복한다. 노드분할 시 루트노드 분할의 경우 CIR-트리 색인 설명자의 내용이 변경되는 것을 막기 위해 처음 결정된 루트노드는 변하지 않도록 새로이 두 개의 노드를 할당하고 루트노드를 분할하여 분할된 내용을 두 노드에 나누어 기록한 후, 루트노드에 두 노드의 MBR을 기록해 루트노드의 페이지가 변경되는 것을 막았다.

다음의 그림 10은 삽입 알고리즘에 대한 설명이다. 1

```
알고리즘 cir_Insert
입력 : *NewEntry
{
1:   cir_FindNode;
2:   if ( 여유공간이 없다면 )
3:   {
4:           if ( 부모노드가 루트 )
5:                   노드 분할;
6:           else
7:                   재삽입 수행;
8:   }
9:   else
10:  {
11:          엔트리 삽입;
12:          부모노드의 MBR 변경;
13:  }
14:  return;
}
```

그림 10 삽입 알고리즘

행의 cir_FindNode는 엔트리가 삽입될 단말노드까지의 경로를 구하는 함수이다. 루트노드에서 시작하여 새로운 엔트리가 삽입되어 겹침 영역의 증가가 최소가 되는 것, 비 활성 차원이 가장 많이 같은 MBR, 영역의 증가가 최소, 둘레의 길이가 최소, MBR의 중심으로부터 새로 삽입하려는 엔트리까지의 거리가 최소가 되는 순서로 노드의 엔트리들 중 자식노드를 결정한다. 이러한 방법으로 해당되는 레벨의 노드까지 경로를 구해 내려가게 된다. 2행과 9행은 삽입 대상이 되는 단말노드에 새로운 엔트리를 삽입할 여유 공간이 있는지를 확인한다. 3~8행은 오버플로우인 경우의 처리를 하고 10~13행의 경우 하나의 엔트리를 삽입하는 과정을 보여준다. 3~8행의 경우 부모노드가 루트인 경우를 제외하고는 재삽입을 수행한다. 10~13행의 경우는 하나의 엔트리를 삽입한 후 상위노드에 변경된 MBR을 반영시킨다.

3.3.3 탐색 알고리즘

이 논문에서 구현된 CIR-트리 관리기는 범위탐색과 k-최근접 탐색을 지원한다.[15, 16] 그림 11은 범위탐색, 그림 12는 k-최근접 탐색의 의사코드이다. 범위탐색 알고리즘에선 두 개의 큐를 유지한다. 첫 번째 큐는 탐색경로를 유지하는 큐이고 다른 하나는 탐색 결과를 저장하는 큐이다. 10행에서는 중간노드에서 노드내 엔트리들의 MBR과 주어진 질의와의 거리를 계산하여 주어진 범위내의 엔트리들만을 추출하여 큐에 저장한다. 12 행에서는 단말노드의 경우이며 주어진 범위내의 모든 엔트리들을 결과집합 큐에 저장한다.

최근접 탐색은 k가 1인 경우로 같은 거리에 있는 엔트리가 여러 개이면 같은 거리에 있는 엔트리 전부를

```
알고리즘 cir_RangeSearch
입력 : Query, Distance, RootPage, TotalDim, ResultSet
{
1:    while(1)
2:    {
3:      큐가 비었으면 break;
4:      큐로부터 노드 추출;
5:    for( ;PageId!=NIL ; PageId = cp->NextSuper)
6:      {
7:        for(노드의 엔트리 개수 동안 반복)
8:        {
9:            if(중간노드)
10:             주어진  거리 내의 자식 노드만을 추출
11:           else 단말노드
12:             주어진 거리 내의 엔트리들을 추출;
13:       }
14:     }
15:   }
}
```

그림 11 범위-탐색 알고리즘

결과집합에 포함시킨다. k-최근접 탐색에서는 탐색한 결과 값에서 거리가 가까운 k개를 결과집합에 포함시키고 k번째 엔트리에 같은 값이 여러 개 있다면 같은 값을 갖는 모든 엔트리를 결과집합에 포함시킨다. 그림 12의 15행 이전까지의 과정은 트리를 순회하면서 정렬된 리스트를 구하는 과정이고 15행 이후의 과정은 정렬된

```
알고리즘 cir_KnnSearch
입력 : Query, k, RootPage, TotalDim, ResultSet
{
1:   if(루트노드)
2:   {
3:     for( ; ; CurPid = CirPage->NextSuper)
4:     {
5:       최근접 거리의 초기화
6:       if(중간노드라면)
7:       {
8:         각 엔트리와 질의로부터 거리를 산출;
9:         리스트를 정렬;
10:        각 엔트리의 자식 노드를 탐색
11:      }
12:      else
13:       노드에서  최근접 거리보다 작거나 같은 값만을 추출;
14:    }
15:  }
16:  for( k번째 엔트리까지)
17:  {
18:    결과로부터 하나의 엔트리를 추출;
19:    추출된 엔트리를 결과 큐에 저장;
20:    if(결과의 k번째 엔트리인 경우)
21:      k 번째 엔트리와 같은 값을 가지는 모든 엔트리 추출
22:  }
}
```

그림 12 k-최근접 탐색 알고리즘

리스트로부터 결과 값을 추출하는 과정이다.

3.4 CIR-트리 관리기의 API

3.4.1 색인 생성 API

midas_cir_createindex는 데이타 화일의 식별자, 필드 설명자, 전체 차원의 수, 변환함수에 대한 포인터, 활성 차원의 수, 반환될 색인 식별자가 저장될 주소를 입력으로 받아서 CIR-트리를 생성한 후 생성된 색인식별자를 CIRIndexID로 반환한다. 다음은 midas_cir_createindex의 함수 원형이다.

```
midas_cir_createindex(BaseFileID,FieldDesc,TotalDim,Transform
(),ActiveDim,CIRIndexID)
T_FID                BaseFileID;
                     /* 데이타 화일의 화일식별자 */
T_CIRFIELDDESC       FieldDesc;
                     /* 필드설명자 */
int                  TotalDim;
                     /* 전체 차원의 수 */
char                 (*Transform);
                     /* 변환함수 */
T_CIRIDXID           *CIRIndexID;
                     /* 색인식별자를 받을 주소 */
```

3.4.2 색인 소멸 API

midas_cir_destoryindex는 색인 식별자를 입력받아 색인 식별자가 가리키는 색인을 제거한다. 다음은 midas_cir_destroyindex의 프로토타입이다.

```
midas_cir_destroyindex(CIRIndexID)
T_CIRIDXID           *CIRIndexID;
                     /* 삭제될 색인의 식별자 */
```

3.4.3 커서 개방 API

midas_cir_opencursor는 색인식별자와 읽기와 쓰기모드를 지정하는 플래그를 입력으로 받아 커서를 생성한다. 이 함수를 통해 생성된 커서를 이용하여 CIR-트리 색인에 대한 작업을 수행한다. 다음은 midas_cir_opencursor의 프로토타입이다.

```
midas_cir_opencursor(CIRIndexID, Mode)
T_CIRIDXID           *CIRIndexID;
                     /* 대상 색인의 색인 식별자 */
unsigned char        Mode;
                     /* 색인의 개방 형태 (읽기, 쓰기) */
```

3.4.4 커서 폐쇄 API

midas_closecursor은 커서 식별자를 입력받아 주어진 커서를 닫는다. 다음은 midas_cir_closecursor의 프로토타입이다.

```
midas_cir_closecursor(CIRcsid)
T_CIRCSID            CIRcsid;
                     /* 개방된 CIR색인 커서 식별자 */
```

3.4.5 데이타 삽입 API

midas_cir_addcursor는 커서 식별자와 추가될 데이타의 레코드 식별자를 입력으로 받아 해당 레코드를 색인

에 추가한다. 다음은 midas_cir_addcursor의 프로토타입이다.

```
midas_cir_addcursor(CIRcsid, Rid)
T_CIRCSID        CIRcsid;
                 /* 커서식별자 */
T_RID            Rid;
                 /* 삽입될 데이타의 레코드식별자 */
```

3.4.6 벌크 삽입 API

midas_cir_addbulk는 커서식별자, 레코드식별자의 리스트, 레코드 식별자들이 저장된 화일의 식별자, 삽입될 데이타의 개수를 입력으로 받아 하나 이상의 데이타를 색인에 삽입한다. 다음은 midas_cir_addbulk의 프로토타입이다. 프로토타입에서 보는 것처럼 삽입될 레코드에 대한 식별자들은 리스트형태(*Rid)로 입력될 수도 있고 또는 화일(RidFileID)에 저장되어 입력될 수도 있다.

```
midas_cir_addbulk(CIRcsid, Rid, RidFileID, No_of_Rids)
T_CIRCSID        CIRcsid;
                 /* 커서식별자 */
T_RID            *Rid;
                 /* 레코드식별자의 리스트 */
T_FID            RidFileID;
                 /* 레코드식별자 화일의 식별자 */
int              No_of_Rids;
                 /* 추가될 데이타의 개수 */
```

3.4.7 삭제 API

midas_cir_deletecursor는 커서식별자와 레코드식별자를 입력받아 주어진 레코드를 색인에서 삭제한다. 다음은 midas_cir_deletecursor의 프로토타입이다.

```
midas_cir_deletecursor(CIRcsid, Rid)
T_CIRCSID        CIRcsid;
                 /* 커서 식별자 */
T_RID            Rid;
                 /* 삭제될 레코드의 식별자 */
```

3.4.8 이미지 검색 API

구현한 CIR-트리 관리기에서는 탐색이 두 단계로 이루어진다. 먼저 midas_cir_searchimage를 통해서 질의에 부합하는 특징벡터들을 찾고 midas_cir_getimage를 통해서 원하는 이미지를 거리 또는 RID의 순서로 읽어온다. midas_cir_searchimage는 커서식별자, 질의, 검색된 데이타의 개수를 저장할 주소를 입력받아 질의에 부합하는 데이타들을 검색한다. 이때 질의로 주어지는 구조체에는 질의 특징벡터, 검색방법, 검색방법에 따라 거리(범위 질의), k(K-최근접 질의) 등이 정의되어 있다. 다음은 midas_cir_searchimage의 프로토타입이다.

```
midas_cir_searchimage(CIRcsid, Key_for_PointData, No_of_Rid)
T_CIRCSID        CIRcsid;
                 /* 커서 식별자 */
T_CIR_PointKey   Key_for_PointData;
                 /* 질의를 위한 구조체 */
int              No_of_Rid;
                 /* 검색된 데이타의 수 */
```

3.4.9 이미지 판독 API

midas_cir_getimage는 커서식별자, 결과집합을 저장할 리스트의 주소, 검색할 결과 개수, 정렬방식을 나타내는 플래그를 입력으로 받아서 midas_cir_searchimage를 통해 검색한 결과들 중, 원하는 개수의 이미지만을 거리순서 또는 레코드식별자의 순서로 제공한다. 다음은 midas_cir_getimage의 프로토타입이다.

```
midas_cir_getimage(CIRcsid, Result, No_of_Rid, Navigation)
T_CIRCSID        CIRcsid;
                 /* 커서의 식별자 */
T_CIRRESULT      *Result;
                 /* 결과 집합 */
int              No_of_Rid;
                 /* 검색된 결과의 수 */
int              Navigation;
                 /* 정렬 방식(Rid 순, 거리 순) */
```

4. 성능 평가

CIR-트리와 다른 고차원 색인구조의 비교 결과는 이미 다른 논문들[10, 11]에서 CIR-트리가 가장 우수하다고 제시된 바 있다. 이 논문에서는 CIR-트리 관리기의 평가 기준으로 MiDAS-III의 순차화일을 선정하여, 순차화일에 비교한 CIR-트리의 성능을 측정하였다. 본 논문의 구현에 사용된 시스템은 Solaris 2.6.x 운영 체제에 Sun UltraSPARC-II, 366㎒ CPU 2개를 장착하고 있으며 사용된 컴화일러는 gcc 2.7.1 이다. 이 논문에서 구현한 CIR-트리 관리기에 대한 실험은 정규분포를 가진 실수형 데이타 집합을 이용해 4차원부터 16차원까지의 데이타를 구성해 사용하였다. 활성차원은 5차원을 사용하였다. 성능 측정은 색인 구성 성능과 검색 성능에 대해서 측정하였다. 색인 구성 성능 측정은 구성시간과 구성노드에 대해서 측정하였다. 구성 시간은 CIR_CreateIndex함수의 cir_BuildIndex 함수의 시작 시점부터 종료 시점까지의 시간을 측정하였다. 저장공간에 대한 평가를 위해 색인을 구성하는 노드의 개수를 측정하였다. 즉, 중간노드의 개수와 단말노드의 개수를 측정하였다. 검색 성능에 대한 측정은 범위-탐색과 k-최근접 탐색에 대해 검색 시 순회한 노드의 개수와 검색시간에 대해서 측정하였다.

그림 13은 CIR-트리의 구성시간을 측정한 결과이다. 측정은 4, 8, 16차원에 대해서 동일한 개수(50,000)의 데이타에 대한 색인 구축 시간에 대해서 측정하였다. 그림 14는 색인을 구성하는 페이지(노드)의 개수를 보여주고 있다. 이 실험은 CIR-트리 관리기의 저장공간에 대한 효율성을 측정하기 위해 실행하였다. 그림 15를 보면 차

원이 증가하더라도 CIR-트리의 단말노드에 대한 중간노드의 수의 비율이 매우 작고 변화도 미미한 것을 볼 수 있다. 이것은 색인 구축으로 인한 저장공간에 대한 부담이 적음을 보여준다.

다음은 순차 검색과의 비교를 통해 구현한 CIR-트리 관리기의 타당성을 입증한다. 실험에 사용한 데이타 집합은 10차원 데이타에 활성차원은 5개이며 5만개 데이타를 사용하였다. CIR-트리 관리기의 타당성을 보이기 위해 CIR-트리를 이용해 K-최근접 질의와 범위 질의를 수행할 때 접근한 노드의 수와 검색시간 그리고 순차탐색을 이용해 K-최근접 질의와 범위 질의를 수행할 때 접근한 노드 수와 시간을 각각 비교한다.

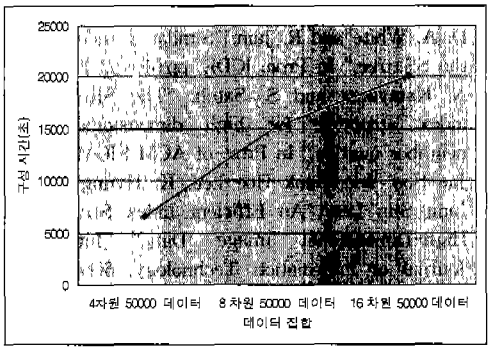

그림 13 색인 구성시간

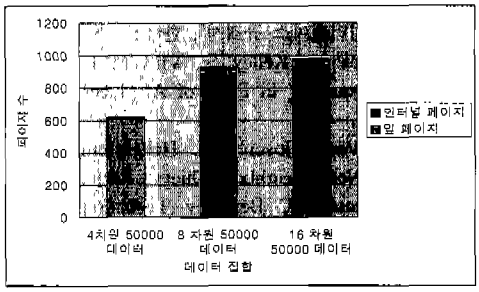

그림 14 노드 구성

그림 15는 K-최근접 질의를 각각 CIR-트리를 통해서 수행했을 때와 순차검색을 이용하여 수행했을 때 접근한 노드 수를 비교한 것이다. CIR-트리를 통해 수행했을 때가 그렇지 않을 때 보다 약 87 % 정도의 성능 향상을 얻을 수 있었다. 그림 15는 같은 상황에서 시간을 가지고 비교한 것이다. 이 경우에는 약 99 % 정도의 성능 향상을 얻는 것을 볼 수 있었다.

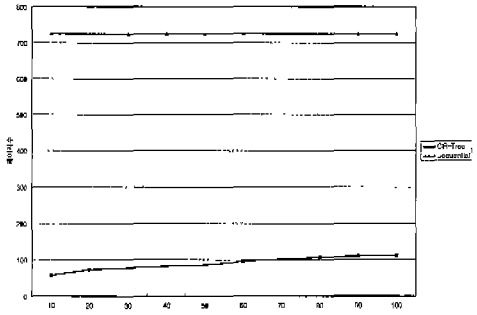

그림 15 K-최근접 질의 (노드수)

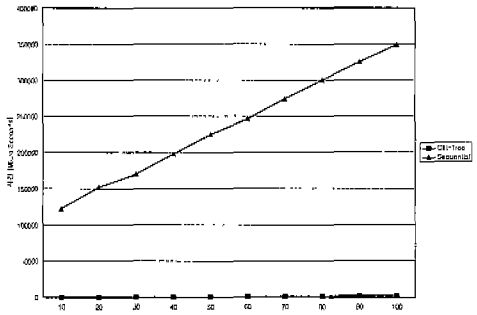

그림 16 K-최근접 질의 (시간)

그림 17과 그림 18은 각각 범위 질의를 수행했을 때 접근한 노드 수와 수행 시간을 비교한 것이다. 이 경우에도 각각 60%와 98%의 성능 개선 효과를 볼 수 있었다. 평균적으로 보면 노드 접근 회수 보다 수행 시간 측면에서 더 높은 성능 개선 효과를 볼 수 있었는데 이유는 순차탐색의 경우에는 접근하는 모든 노드는 모두 처음 접근하는 노드지만 CIR-트리의 경우에는 중간노드에 해당하는 노드들은 이미 접근했던 노드일 확률이 있기 때문이다.

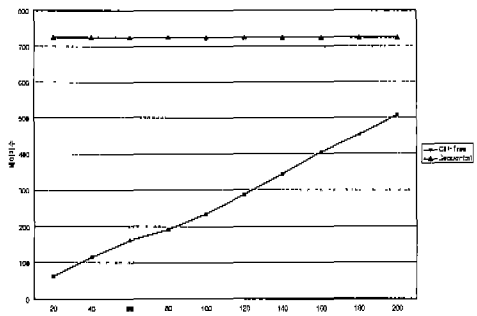

그림 17 범위 질의 (노드수)

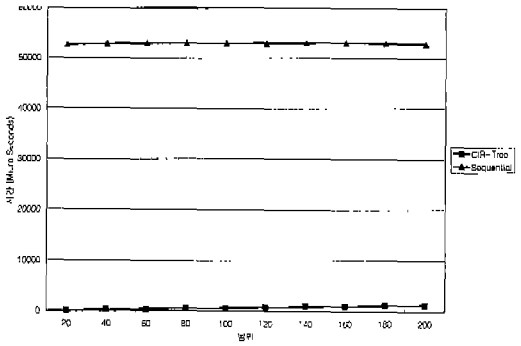

그림 18 범위 질의 (시간)

4. 결 론

이 논문에서는 보다 효율적인 내용기반 이미지 검색을 지원하기 위해 MiDAS-III에서 CIR-트리 관리기를 설계하고 구현하였다. 구현한 CIR-트리 관리기는 순차검색과 비교했을 때 매우 뛰어난 성능을 보였다. 이것은 CIR-트리 관리기가 MiDAS-III에 포함됨으로써 그 이전보다 훨씬 더 효율적인 내용기반 이미지 검색을 지원할 수 있다는 것을 의미한다. 향후 연구에서는 삽입연산 수행 시 재삽입을 수행할 때가 수행하지 않을 때에 비해 수행 시간이 상대적으로 매우 긴 단점이 있는데 이를 보완하고자 한다. 또한, 이 논문에서 구현한 CIR-트리 관리기는 동시성 및 회복이 고려되지 않았는데 향후 연구에서는 회복 및 동시성을 고려한 CIR-트리 관리기를 MiDAS-III에서 구현 할 것이다.

참 고 문 헌

[1] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, "The QBIC Project: Querying Image by Content using Color, Texture, and Shape," In Proc. SPIE(Storage and Retrieval for Image and Video Databases), pp.173-187, February 1993.

[2] D. A. White and R. Jain, "Similarity Indexing : Algorithm and Performance," In Proc. of SPIE (*Storage and Retrieval for Image and Video* Database), pp.62-75, 1996.

[3] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equiz, "Efficient and Effective Querying by Image Content," Journal of Intelligent Information System(JIIS), 3(3), pp.231-262, July 1994.

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

[5] T. Sellis, N. Roussopoulos and C. Faloustos, "The R+-tree: A Dynamic Index for Multi-Dimensional Objects," In Proc. of VLDB, pp.507-518, 1987.

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

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

[8] K. -I .Lin, H. Jagadish and C. Faloutsos, "The TV-tree: An Index Structure for High-Dimensional Data," VLDB Journal, 3(4), pp.517-549, Octorber 1994.

[9] D. A. White and R. Jain, "Similarity Indexing with the SS-tree," In Proc. ICDE, pp.515-523, 1996.

[10] N. Katayama and S. Satoh. "The SR-Tree: An index structure for high dimensional nearest neighbor queries," In Proc. of ACM SIGMOD, 1997.

[11] Jae Soo Yoo, Seok Hee Lee, Ki Hyung Cho and Jang Sun Lee, "An Efficient Index Structure for High-Dimensional Image Data," International Journal of Information Technology, 6(1), pp.1-15, May 2000.

[12] 이석희, 송석일, 유재수, "내용기반 이미지 검색을 위한 고차원 색인구조", 한국정보학회 데이타베이스연구회 논문지, 14(4), pp.53-68, 1998.

[13] 박치항 외 13인, "바다 DBMS를 중심으로 한 데이타베이스 관리 시스템 구조", 한국전자통신연구원, 1997

[14] "마이다스 인터페이스 규격서", 한국전자통신연구원

[15] *Roussopoulos* N., *Kelley* S., *Vinvent* F., "Nearest Neighbor Queries," In Proc. of ACM SIGMOD, pp. 71-79. 1995.

[16] Shuanhu Wang, Joseph M. Hellerstein and Ilya Lipkind. "Near-Neighbor Query Performance in Search Trees." UC Berkeley Tech Report CSD-98-1012, September 1998.

송 석 일

1998년 충북대학교 공과대학 정보통신공학과(공학사). 2000년 충북대학교 정보통신공학과(공학석사). 2000년 ~ 현재 충북대학교 정보통신공학과(박사과정). 관심분야는 고차원 데이타 색인, 정보검색 프로토콜(Z39.50), OODBMS, 저장 시스템, 회복기법, 동시성제어

이 희 중

1998년 한남대학교 전자계산공학과 학사. 2000년 충북대학교 정보통신공학과 석사. 2000년 ~ 현재 (주)넥스텍 연구개발부 근무. 관심분야는 저장구조, 고차원색인구조, XML 저장관리, 검색엔진, 데이타마이닝

이 석 희

1994년 2월 충북대학교 정보통신공학과 졸업(공학사). 1998년 2월 충북대학교 정보통신공학과 대학원 졸업(공학석사). 2000년 2월 충북대학교 정보통신공학과 박사과정수료. 2000년 3월 ~ 현재 동아방송대학 인터넷방송과 전임강사. 관심분야는 멀티미디어 데이타베이스, 정보검색, 영상처리, 영상통신 등

유 재 수

정보과학회논문지 : 컴퓨팅의 실제
제 7 권 제 1 호 참조

조 기 형

정보과학회논문지 : 컴퓨팅의 실제
제 7 권 제 1 호 참조

유 관 희

1985년 전북대학교 전산통계학과 학사. 1988년 한국과학기술원 전산학과 석사. 1995년 한국과학기술원 전산학과 박사. 1988년 ~ 1997년 (주)데이콤 종합연구소 선임연구원. 1997년 ~ 현재 충북대학교 컴퓨터교육과 교수. 관심분야는 컴퓨터 그래픽, 계산기하학, 네트워크 게임, 인공 치아 3차원 모델링 등