NoD서비스용 멀티미디어 데이터의 효율적인 저장 및 검색을 위한 하부저장 관리자의 설계 및 구현

진 기 성[†] · 정 재 욱[††] · 장 재 우[†††]

요 약

최근 NoD (News-on-Demand)에 대한 사용자의 요구가 증가함에 따라, 이러한 요구를 충족시키기 위한 많은 연구가 진행 중에 있다. 그러나 뉴스 비디오 데이터는 짧은 생명주기, 앵커를 중심으로 한 주기적 변화로 인해 기존의 비디오 저장방식을 그대로 적용하기가 어렵다. 이를 위해, 본 논문에서는 NoD 서비스용 멀티미디어 데이터의 효율적인 저장 및 검색을 위한 하부저장 관리자를 설계 및 구현한다. 하부 저장 관리자는 멀티미디어 객체 지체의 비디오 스트림 데이터와 색인 정보를 효율적으로 저장하고 관리하며, 텍스트 기반 검색을 위해 역화일 기법과 고차원 특징 벡터의 색인을 위해 X-트리 구조를 제공한다. 아울러 하부 저장 관리자는 계층적인 뉴스 비디오 객체로부터 추출된 부가적인 정보들을 효율적으로 저장할 수 있는 저장 API (Application Program Interface)와 커서를 통한 사용자의 편리한 검색을 지원하는 검색 API를 제공한다. 마지막으로 하부 저장 관리자는 SHORE(Scalable Heterogeneous Object REpository) 저장 시스템을 이용하여 UNIX 운영체제 하에서 표준 C++ 언어를 사용하여 구현하였다.

Design and Implementation of a Low-level Storage Manager for Efficient Storage and Retrieval of Multimedia Data in NoD Services

Ki-Sung Jin[†] · Jae-Wuk Jung[††] · Jae-Woo Chang[†††]

ABSTRACT

Recently as the user request on NoD (News-on-Demand) is largely increasing, there are a lot of researches to fulfill it. However, because of short life-cycle of news video data and periodical change of video data depending on anchor, it is difficult to apply the conventional video storage techniques to NoD applications directly. For this, we design and implement low-level storage manager for efficient storage and retrieval of multimedia data in NoD Services. Our low-level storage manager not only efficiently stores video stream data of news video itself, but also handles its index information. It provides an inverted file method for efficient text-based retrieval and an X-tree index structure for high-dimensional feature vectors. In addition, our low-level storage manager provides some application program interfaces (APIs) for storing video objects itself and index information extracted from hierarchical news video and some APIs for retrieving video objects easily by using cursors. Finally we implement our low-level storage manager based on SHORE (Scalable Heterogeneous Object REpository) storage system by using a standard C++ language under UNIX operating system.

1. 서 론

최근 멀티미디어 산업이 정보 사회의 핵심 산업으로 부상하면서 멀티미디어 데이터에 대한 사용자의 요구가 증가하고, 이에 따라 멀티미디어 데이터베이스에 관련된 연구들이 국내외적으로 활발하게 진행되고 있다. 멀티미디어 데이터는 주로 텍스트, 정지 영상, 음성, 오디오, 비디오 등 그 형태가 매우 다양하며, 특히 비디오는 데이터의 양이 방대할 뿐만 아니라 시간과 공간 등의 정보를 포함하기 때문에 저작, 편집, 검색이 매우 어렵다. 일반적으로, 비디오를 기반으로 하는 멀티미디어 응용 서비스는 다양한 종류의 기술을 요구한다. 첫째, 비디오 데이터를 계층적, 구조적인 형태로

※ 본 연구는 한국 과학 재단의 97 목적 기초 과제(97-0100-0101-3)의 연구비에 의해 수행되었음
† 준 회 원 : 전북대학교 대학원 컴퓨터공학과
†† 준 회 원 : 도림위즈(주) 근무
††† 종신회원 : 전북대학교 컴퓨터공학과 교수
논문심사 : 1999년 10월 23일 심사완료 : 2000년 3월 10일

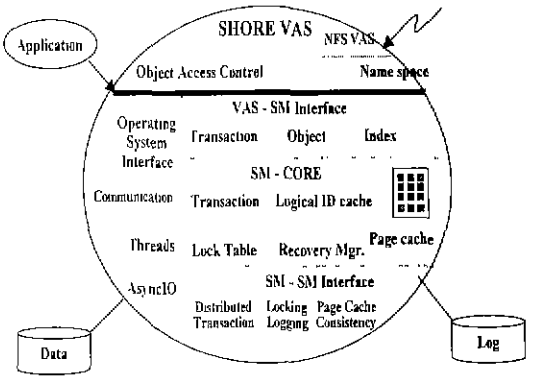
표현할 수 있는 비디오 파싱(Parsing) 기술이 필요하다[1, 2, 3]. 둘째, 비디오 파싱을 통해 나온 각각의 객체들과 그 객체에서 추출된 부가적인 정보를 효율적으로 저장 및 관리하기 위한 비디오 압축 및 저장 기술이 요구된다[4]. 셋째, 비디오 파싱을 통해 얻은 각각의 객체들에 대한 다양한 특징 정보(예를 들면, 대표 프레임에서 추출한 색상, 질감, 형태 정보)를 이용한 비디오 색인 기술이 필요하다. 마지막으로, 사용자의 다양한 검색을 지원하며 편리하게 검색 결과를 표현할 수 있는 사용자 인터페이스 기술이 요구된다[5]. 한편, 비디오에 대한 다양한 검색을 지원하는 비디오 검색 및 비디오 색인에 대한 연구가 다수 수행되어져 왔으나, 비디오의 효율적인 저장 구조에 대한 연구는 많지 않은 실정이다.

본 논문에서는 NoD 서비스용 멀티미디어 데이터의 효율적인 저장 및 검색을 위한 하부저장 관리자를 설계 및 구현한다. 이를 위해 먼저 NoD 서비스용 뉴스 비디오 객체를 여러 개의 하위 객체들로 분류하고 각각의 객체에 대해서 내용-기반 검색을 위한 특징 정보들을 명시한다. 또한, 비디오 파싱 결과로부터 추출된 각각의 객체들의 특징 정보들과 각 객체의 물리적인 비디오 스트림 데이터를 보다 효율적으로 저장하기 위하여, NoD 서비스용 멀티미디어 데이터의 하부저장 관리자를 미국 위스콘신 대학에서 개발한 SHORE (Scalable Heterogeneous Object REpository) 저장 시스템에 기반하여 구현한다. 아울러 구현된 하부 저장 관리자는 텍스트 기반 검색을 위해 역화일 기법과 고차원 특징 벡터의 색인을 위해 X-트리 구조를 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 기존의 SHORE 저장 시스템을 다룬다. 3장에서는 기존의 SHORE 저장 시스템에 텍스트 기반 검색을 위해 역화일 기법과 고차원 특징 벡터의 색인을 위해 X-트리 구조를 추가한 확장된 SHORE 저장 시스템을 설명한다. 4장에서는 NoD 서비스용 멀티미디어 데이터의 하부저장 관리자를 3장에서 제시된 확장된 SHORE 저장 시스템을 기반으로 설계 및 구현한다. 아울러 5장에서는 NoD 서비스용 멀티미디어 데이터의 하부저장 관리자의 구현 및 성능평가를 제시하고, 마지막으로 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. SHORE 저장 시스템

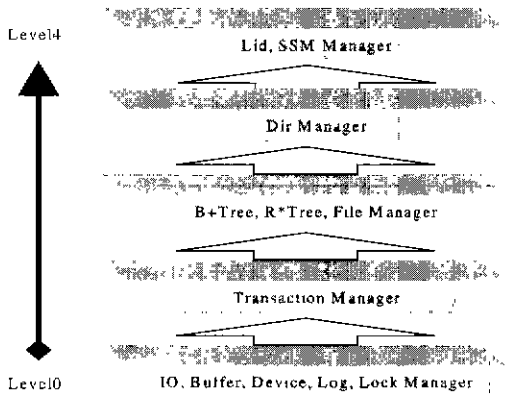
SHORE(Scalable Heterogeneous Object REpository)는 미국의 위스콘신 대학에서 개발한 지속성 객체 저장 시스템으로 기존의 OODB(Object-Oriented DataBase)에서의 단점인 파일 시스템과의 결합의 어려움, 트랜잭션(transaction) 관리의 어려움, Peer-to-Peer 구조의 부적절성 등을 해결하기 위해 개발되었다. SHORE는 객체 지향 데이터베이스 기술과 파일 시스템 기술을 합성하여, 기존의 UNIX 파일 시스템 기반 응용 프로그램을 쉽게 접목시킬 수 있는 장점을 가진다. SHORE는 EXODUS[6]라는 저장 관리자를 확장한 것으로 (그림 1)과 같이 크게 SVAS(Shore Value-Added Server)와 SSM(Shore Storage Manager)으로 구성되어 있다. SVAS 는 시스템 간에 객체들을 서로 원활하게 통신할 수 있도록 관리하며, SSM는 실제 시스템에서 생성되고 처리되는 객체를 관리한다.



(그림 1) SHORE 저장 시스템의 전체적인 구성도

SSM은 객체에 대한 저장 구조로 디바이스(device), 볼륨(volume), 파일, 가변길이 레코드를 제공하고 있으며, 인덱스로는 B+-트리와 R*-트리를 지원한다. 디바이스에는 하나의 볼륨만을 생성할 수 있다. 볼륨은 파일과 인덱스 저장 구조의 집합이며, 이들의 정보를 목록화하기 위해 root index라 부르는 B+-트리 인덱스가 포함된다. 파일은 레코드의 집합체이며, 레코드는 최대 4-GB의 크기 제한을 가진다. 이외에 로그-지향 회복 메커니즘, 동시성 제어, 그리고 파일과 페이지-단위 잠금을 지원한다. 또한 SSM은 (그림 2)와 같이 5단계의 계층으로 이루어져 각 계층마다 독립적인 역할 수행을 하는 객체지향 기법을 사용한다. 이때 시스템의 동작은 제일 하위 계층인 Level 0 에서 부터 Level 4 까지 각각의 독립적인 기능들을 상속받아 수행된다. 보다 세부적인 사항을 살펴보면 Level 0에서는 시스템의 하위 저장 부분을 담당하는 Device, I/O, Buffer 와 데이터베이스의 회복기법인 Log, Lock 관리자를 담당하고, Level 1에서는 트랜잭션에 관한 관리자를 제공한다.

Level 2에서는 파일과 B-트리, R*트리에 대한 관리자를 제공해주며, Level 3은 디렉토리에 관한 기능을 수행한다. 마지막으로 가장 상위 레벨인 Level 4에서는 논리적인 ID에 관한 관리자와 SSM을 위한 실제 사용자 레벨의 API(Application Programming Interface)를 제공해준다.



(그림 2) SSM 계층적 구성도

또한 SHORE를 이용하여 구현된 시스템으로는 GIS 응용을 위해 개발된 PARADISE[7]와 Cornell 대학에서 수행한 PREDATOR[8]가 있다. PARADISE 는 지속성 객체(persistent object) 관리자를 위해 SHORE를 기반으로 2-D 지도상에 공간적인 속성 객체들을 이용한 Client-Server 구조로 구현되었고, PREDATOR는 객체-관계(Object-Relational) DBMS 로써 SHORE의 SVAS를 이용하여 구현한 시스템이다.

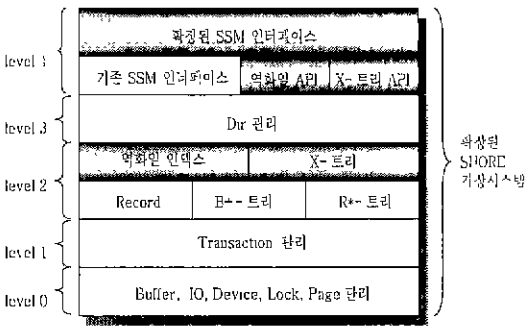
3. 확장된 SHORE 저장 시스템

본 논문에서는 NoD 서비스용 멀티미디어 데이터의 효율적인 저장 및 검색을 위한 하부저장 관리자를 설계하기 위해서, 먼저 텍스트 검색을 위해 역화일 기법과 고차원 특징 벡터 색인을 위해 X-트리 구조를 제공하는 확장된 SHORE 저장 시스템을 구축한다.

3.1 X-트리 고차원 색인 구조

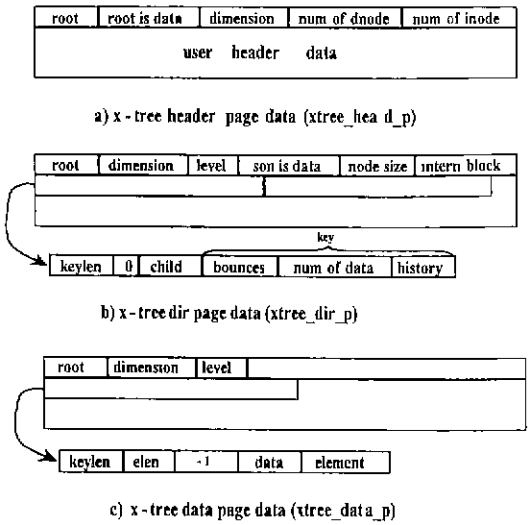
대용량의 멀티미디어 데이터를 다루는데 있어서 주된 문제는 검색의 효율성으로, 이를 위한 적절한 색인 기법이 제공되어야 한다. 그 동안 R*-트리를 비롯한 다수의 다차원 색인 기법들에 대한 연구가 진행되어 왔으나, 차원의 수가 증가함에 따라 시간 또는 공간 N 요구량이 지수적으로 증가되어 색인 구조로서 효력을 상실하게 된다 따라서 고차원 특징벡터의 색인을 위한 다수의 기법이 제안되었으며, 이는 LSD^h-트리, VAM-kd 트리, SS-트리, SR-트리, Pyramid-트리, TV-트리, X-트리 등이다. 한편 X-트리는 겹침영역을 최소화하여 검색 성능을 향상시킨 방법으로, 이들 고차원 색인 기법들 가운데서 우수한 성능을 나타낸다[9] 즉, X-트리는 디렉토리에서의 겹침 영역을 피하기 위해 분할 알고리즘과 수퍼 노드 개념을 이용한다. X-트리 구조는 페이지 노드 분할 시 겹침 영역이 최소가 되지 못할 때는 분할하지 않고, 노드의 크기가 가변적으로 확장될 수 있는 슈퍼노드를 사용한다 X-트리 구조는 고차원으로 갈수록 겹침 영역이 증가되기 때문에 기억공간이 절약되고 빠른 접근이 가능한 선형적인 디렉토리 구조를 사용한다. 노드는 크게 MBR(Minimum Bounding Rectangle) 정보를 갖는 중간 노드와 실제 특징 벡터 데이터를 갖는 데이터 노드로 구분된다

한편 SSM(Shore Storage Manager)은 레벨 2 에서 인덱스 관리를 위한 B-트리 관리자와, 다차원 색인을 위한 R*-트리 관리자 등으로 이루어져 있다. 이 색인 관리자들은 모두 저장구조의 제일 하위단계인 레벨 0로부터 상속받아 구현되고, 특히 R*-트리는 다차원 특징벡터 색인을 지원하는 기법으로서 2D 응용을 위한 GIS등에 적합한 구조이다 그러나 대용량 비디오 데이터와 같은 멀티미디어 객체를 저장하기 위해서는 고차원 색인기법이 요구된다 이를위해 본 논문에서는 고차원 색인에 적합한 X-트리를 R*-트리와 동일한 레벨(level) 인 SHORE 저장 시스템의 레벨 2에서 구현하며, X-트리의 각 페이지들이 가장 하위 레벨인 레벨 0의 Page 관리자로부터 상속받아 구축되도록 한다. X-트리를 통합하여 확장된 SHORE 저장 시스템의 구조는 (그림 3)과 같다



(그림 3) 확장된 SHORE 저장 시스템의 구조

구현 방법은 각 관리자들이 독립적으로 수행되는 SHORE의 특성을 이용하기 위해 X-트리 관리자 클래스인 xtree_m을 정의하고, 이 xtree_m 클래스는 특징 벡터들의 저장 및 검색에 대한 메소드들을 제공해 준다. 아울러 xtree_m 클래스의 페이지를 관리하는 xtree_head_p 클래스, xtree_data_p 클래스, xtree_dir_p 클래스를 이용하여 설계하고, 이 각각의 페이지 관리 클래스들은 SHORE에서 제공하는 페이지 관리자인 page_p 클래스의 상속을 받아 수행한다. xtree_head_p 클래스는 X-트리의 헤더로서 루트 페이지 식별자, 부모페이지가 데이터 페이지인지의 여부, 차원의 수, 데이터 노드의 수와 중간 노드의 수 등 X-트리를 처리하기 위한 기본적인 정보들을 관리한다. xtree_data_p 클래스는 X-트리의 데이터 노드를 저장 및 관리한다. 루트 페이지 식별자, 차원, 노드의 레벨이 페이지의 헤더정보로서 유지되며, 특징 벡터에 관한 정보와 그에 따른 엘리먼트 정보들이 페이지의 데이터로 저장되고 관리한다. xtree_dir_p 클래스는 X-트리의 중간 노드를 저장 및 관리한다 루트 페이지 식별자, 차원 노드 레벨, 자식 노드가 데이터 노드인지의 여부 등이 헤더 정보로 구성되어 있다. 또한 수퍼노드의 경우 여러 페이지가 하나의 노드로 구성되어 있으므로 연관된 페이지의 식별자들을 유지해야 한다 중간노드의 데이터로는 MBR (Minimum Bounding Rectangle)정보와 포함하는 특징 벡터의 수, 자식노드의 분할 위치를 나타내는 정보 및 자식 노드의 페이지 식별자가 저장된다. 페이지의 데이터 정보는 (그림 4)와 같다.



(그림 4) 페이지 데이터 구조 정보

한편 저장된 고차원 특징 벡터들의 검색을 위해 본 논문에서는 기존의 SHORE에서 사용된 스캔 개념을 X-트리에 결합하여, 레벨 4에서 확장된 검색을 위한 scan_xd_i클래스를 구현한다. scan_xd_i 클래스는 포인트 질의, 범위 질의, k-nn 질의를 위한 메소드를 지원해주며, scan_xd_i 클래스의 생성자에 의해 질의 방법과 질의 벡터를 지정한다. 생성자는 다음과 같다.

```
scan_xd_i::scan_xd_i(const lvid_t& lvid, const serial_t
                     & stid,mdim_t::xt_cmp_t c, const
                     mdim_t& box, int k);
enum mdim_t::xt_cmp_t { t_point, t_k_near, t_range };
```

lvid는 볼륨 식별자, stid는 X-트리 식별자이고 c는 질의의 방법을 정하는 매개변수로 t_point는 포인트 질의를 나타내며, t_range는 범위 질의를 나타낸다. t_k_near는 k-최근접 질의이며, k 값을 통해 찾고자 하는 검색의 수를 결정한다 box는 질의 특징벡터가 된다. 생성자를 통해 검색된 결과는 scan_xd_i클래스의 next() 메쏘드를 통해 사용자에게 전달되며 다음 검색 결과로 커서를 움직인다 next() 메쏘드는 다음과 같다.

```
scan_xd_i::next(mdim_t& key, void* el, smsize_t&
                elen, bool& eof) ;
```

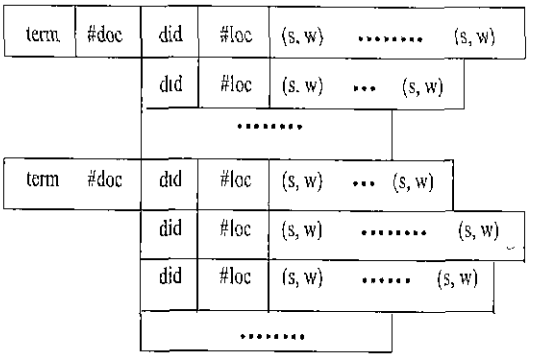
next() 메소드는 key를 통해 검색된 특징 벡터 정보를 반환하며, el은 엘리먼트 정보, elen은 엘리먼트 길이, 그리고 eof는 다음 검색 결과의 유무 정보를 반환한다.

기존 SHORE저장 시스템은 사용자의 접근을 위한 API로서 레벨 4의 ss_m클래스를 통해 SSM 인터페이스를 제공하고 있다. 따라서 본 논문에서는 고차원 특징 벡터의 저장 및 검색을 위해 다음의 X-트리 API를 포함하는 확장된 SSM 인터페이스를 사용자에게 제공한다.

- X-트리 생성
  - create_xd_index(),
- X-트리 삭제
  - destroy_xd_index(),
- X-트리에 특징벡터 삽입
  - create_xd_assoc(),
- X-트리의 특징벡터 삭제
  - destroy_xd_assoc(),
- X-트리를 통한 특징벡터 검색
  - find_xd_assoc(),

3.2 역화일 기법

텍스트 문서를 기반으로 정보 검색을 수행하기 위해 확장된 SHORE저장 시스템은 역화일 기법을 제공해야 한다. 즉 역화일 기법을 통해 뉴스 비디오의 기사 객체에 속하는 기사 내용 부분과 샷 객체의 텍스트 캡션 정보를 저장할 수 있다. 역화일 기법은 내부적으로 단어색인 파일, 후위(posting)화일, 그리고 문서화일로 구성되며, 각 파일은 SHORE 의 SSM를 기반으로 구현된다. 후위 화일의 구조는 색인어의 출현 위치정보를 제공한다. 단어에 대한 key값으로 term을 가지게 되며 각 단어가 출현하는 문서의 출현빈도인 #doc(number of documents)와 문서의 식별자(identifier)로써의 did, 그리고 위치정보를 지닌다. 위치정보는 (S, W)의 쌍으로 단어가 문서 내에서 몇 번째 문단과, 문단 내에서 몇 번째 단어인지를 표시하게 된다. 이에 따른 후위 파일의 구조는 (그림 5)와 같다



(그림 5) 역화일 관리자의 후위 파일 구조

역화일 구조를 통합하기 위해 SSM의 레벨 2에 invertedfile_m 클래스를 설계한다. 단어색인 파일로는 SSM에서 기본적으로 제공하는 B--트리를 사용하며, 포스팅 파일을 위해서는 4GB까지 지원되고, 추가 갱신이 가능한 레코드를 가지는 SSM 파일을 이용한다 따라서 invertedfile_m 클래스는 B+-트리를 관리하는 btree_m 클래스와 파일을 관리하는 file_m 클래스를 이용하여 구현한다. 파일 관리자에 의해 저장되는 레코드의 구조는 색인어 정보, 출현 문서의 수를 가지게 되며, 엘리먼트(문서) 정보와 그 출현 위치들이 된다. 또한 B+-트리 관리자는 색인어에 관한 인덱스를 구성하며, 이들의 엘리먼트 정보는 파일 관리자에 의해 생성된 레코드의 식별자가 된다. 역화일의 확장된 검색을 위해 scan_invt_i 클래스를 구현한다 scan_invt_i 클래스는 메쏘드등을 이용하여 색인어의 정확 검색과 우절단 검색을 지원한다 scan_invt_i 클래스의 생성자에 의해 질의 방법과 질의 벡터를 지정하게 된다 생성자는 다음과 같다

```
scan_invt_i::scan_invt_i(const lvid_t& lvid, const
                         invtserial_t& invtid, cmp_invt_t c,
                         const vec_t& term)
enum cmp_invt_t { notrunc, right }
```

lvid 는 볼륨 식별자이며, invtid은 역화일 구조의 식별자이다. term은 찾고하는 색인어를 나타내며 c를 통해 이 색인어의 검색조건을 나타낸다 nontruc는 정확 검색을, right는 우절단(term*) 검색을 나타낸다 생성자에 의해 검색된 결과는 curr() 메쏘드를 통해 사용자에게 반환되며, next() 메쏘드를 통해 커서를 다음 검색 결과로 이동한다.

역화일 인덱스 API는 역화일 인덱스 파일 자체에 관련된 인터페이스와 검색을 위한 커서에 관련된 인터페이스로 구분된다. 역화일 인덱스 파일 자체에 관련된 인터페이스는 역화일 인덱스 파일을 생성 및 파기를 수행하며, 구성된 역화일 인덱스 파일에서 원하는 색인어를 삭제하거나 원하는 문서를 삭제하는 작업을 수행하게 된다. 검색과 관련된 인터페이스는 역화일 인덱스를 개방하여 검색을 위한 커서를 할당받아 색인 정보를 입력, 출력하는 작업을 수행한다 아울러 텍스트의 효과적인 저장 및 검색을 위해 다음의 역화일 API를 포함하는 확장된 SSM 인터페이스를 사용자에게 제공한다.

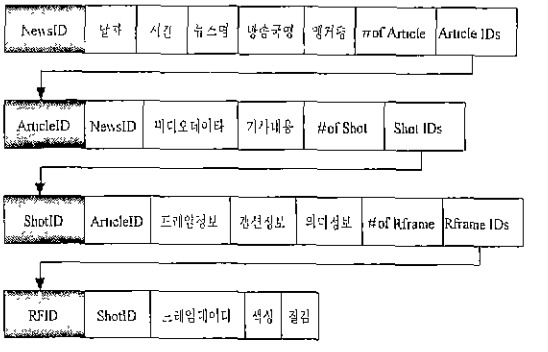
- 역화일 인덱스의 생성
  - create_invt_idx,
- 역화일 인덱스의 삭제
  - destroy_invt_idx
- 역화일 인덱스의 정보 삽입
  - create_invt_assoc,
- 역화일 인덱스 정보의 삭제
  - destroy_invt_assoc
- 역화일을 통한 색인 정보의 검색
  - find_invt_assoc

4. NoD 서비스용 멀티미디어 데이터의 하부저장 관리자

뉴스 비디오 데이터는 다른 종류의 비디오 데이터와는 달리 시간적으로 앵커 장면을 중심으로 일련의 뉴스 기사들이 반복적으로 구성되며, 하나의 뉴스 기사 내에서는 여러 번의 장면 전환이 이루어지는 특징을 지닌다. 따라서, 본 논문에서는 이러한 특징에 기반하여 뉴스 객체를 크게 4가지 형태의 하위 객체들로 분류한다. 첫째, 비디오 스트림의 가장 기본 단위는 프레임(frame)으로 한다. 둘째, 일련의 프레임의 모임을 샷(shot)이라고 명명하며, 장면간의 경계가 뚜렷한 장면 전환을 의미한다. 그리고 비디오 데이타에서만 나타나는 성질인 fixed, panning, tilting, zooming 등과 같은 카메라 동작에 의해서도 샷을 구분하며, 하나의 샷에서는 반드시 하나 이상의 대표 프레임(representative frame)을 부여한다. 셋째, 일련의 샷의 모임을 기사(article)라고 하며, 이 객체는 뉴스 비디오 데이타의 특징인 앵커 프레임을 기준으로 나뉜다. 마지막으로, NoD 서비스용 멀티미디어 객체 중에서 가장 큰 단위인 일련의 기사의 모임을 뉴스(news)라고 한다.

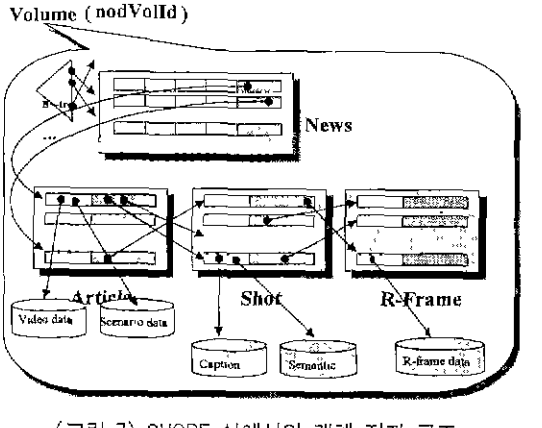
4.1 뉴스 비디오 객체 저장 구조

뉴스 비디오 객체(News)는 각각 뉴스 식별자와 그에 해당하는 카탈로그 정보(속성 정보) 즉 방송국명(예를 들면, 'KBS', 'MBC', 'SBS'), 뉴스명(예를 들면, '9시 스포츠 뉴스'), 앵커맨, 방송된 날짜와 시간(예를 들면, '1998년 12월 17일 9시') 등을 포함하며, 해당 뉴스 객체에 속하는 다수의 기사 객체를 가리키는 기사 객체 식별자들을 유지한다. 둘째, 기사 객체(Article)는 자신이 포함하고 있는 상위 객체인 뉴스 객체의 식별자를 저장하며, 기사 객체의 물리적인 비디오 스트림 데이타, 해당 기사의 텍스트 시나리오 정보인 기사 내용 그리고 해당 기사 객체에 포함되는 다수의 샷 객체를 가리키는 샷 객체 식별자를 포함한다. 셋째, 샷 객체(Shot)는 기사 객체의 마찬가지로 해당 샷 객체를 포함하는 기사 객체의 식별자와 샷 객체의 프레임 정보 즉, 기사 객체에서 해당 샷 객체가 시작하는 시작 프레임 번호(Start Frame Number)와 마지막 프레임 번호(End Frame Number)를 가진다. 그리고 해당 샷의 프레임 자막으로부터 자동으로 추출한 텍스트 캡션 정보, 샷 객체의 의미를 나타내는 의미(Semantic) 정보(예를 들면, 농구 스포츠 뉴스에서 '덩크슛'이라는 의미 정보), 마지막으로 샷 객체를 대표하는 다수의 대표프레임(Rframe)에 대한 식별자를 저장한다. 마지막으로, 대표 프레임 객체는 샷 객체 식별자, 대표 프레임 데이터(즉, 정지 영상 이미지), 대표 프레임의 의미를 나타내는 의미 정보, 해당 대표 프레임 이미지로부터 추출한 특징 벡터 정보(예를 들면, 색상, 질감, 구도 등)를 포함한다. 따라서, 본 논문에서 제안한 NoD 서비스를 위한 뉴스 비디오 객체의 저장 구조는 (그림 6)과 같다.



(그림 6) NoD 서비스용 뉴스 비디오 객체 저장 구조

하나의 뉴스 비디오 객체는 확장된 SHORE 저장 시스템의 SSM에서 제공하는 화일을 이용하여 별도의 영역에 객체별 데이타베이스를 유지하도록 한다. 즉, 최상위 객체인 뉴스 객체들을 위한 뉴스 파일, 아티클 화일, 샷 파일, 대표프레임 파일로 나누어 저장한다. 그리고 각각의 객체에 속하는 부가 정보인 물리적인 비디오 스트림 데이타, 텍스트 정보, 의미 정보, 그리고 특징 벡터 정보 또한 서로 다른 영역에 각각의 데이타베이스를 갖는다. 이러한 뉴스 비디오 저장 구조는 (그림 7)과 같다.



(그림 7) SHORE 상에서의 객체 저장 구조

저장 API는 크게 초기화 인터페이스, 볼륨 생성 및 삭제 인터페이스, 개방 및 폐쇄 인터페이스, 뉴스 객체 색인 정보 중의 단순 속성 정보를 위한 B--트리 인덱스 생성, 파기 그리고 삽입 인터페이스, 데이터를 삽입하는 삽입 인터페이스, 마지막으로 상위 객체와 하위 객체를 연결하는 매핑 인터페이스로 분류된다.

- 초기화
  - snod_init, snod_final
- 볼륨 생성 및 파기
  - snod-createvol, snod-destroyvol
- 개방 및 폐쇄
  - snod-open, snod-close
- B+-트리 인덱스 생성, 파기 및 삽입
  - snod-createbtidx, snod-destroybtidx, snod-addbtidx
- 삽입
  - snod-addnewsinfo, snod-addarticleinfo, snod-addshotinfo, snod-addrframeinfo
- 상위 객체와 하위 객체간의 매핑
  - snod-maptoarticle, snod-maptoshot, snod-maptorframe

먼저, 사용자가 뉴스 비디오 색인기를 통해 색인된 각각의 정보를 하부 저장 구조의 객체 저장 관리자를 통해 저장하기 위해서는 시스템의 환경을 설정하는 snod_init을 설정하여야 한다. 그런 후에, snod_createvol를 통해 볼륨를 생성하고, snod_open를 이용하여 저장 관리자를 개방한다. 삽입할 각각의 뉴스 객체, 기사 객체, 샷 객체, 대표 프레임 객체들에 대한 비디오 스트림 데이터와 부가적인 정보들은 삽입 API를 이용하여 저장한다. 마지막으로, 객체 간의 매핑 인터페이스를 이용하여 상위 객체와 하위 객체들을 연결한다 즉, 하나의 뉴스 객체는 다수의 기사 객체들을 포함하고 있기 때문에, 뉴스 객체의 저장 구조에는 기사 객체들에 대한 객체 식별자를 저장하기 위한 구조를 지닌다. 따라서, 하위 객체에 해당하는 기사 객체들이 실제로 저장된 후 생성되는 기사 객체 식별자를 상위 객체인 뉴스 객체의 저장 구조에 저장함으로써, 상위 객체인 뉴스 객체와 하위 객체인 기사 객체간의 연결을 가능하게 한다. 이를 위해 객체간의 매핑 인터페이스가 필요하다.

4.2 뉴스 비디오 객체 검색

NoD 저장 구조는 뉴스 객체, 기사 객체, 샷 객체, 대표 프레임 객체로 구성되어 있다. 각각의 객체들은 그들이 가지는 고유한 부가적인 정보들을 가지고 있기 때문에 검색을 위한 관리도 각각의 객체들마다 이루어져야 한다 따라서, 객체들에 대한 사용자의 효율적인 검색을 위해 객체 검색 관리자에서는 커서 개념을 도입하여 설계한다. 커서 개념은 검색 조건에 만족하여 검색되어진 객체들에 한해서 커서 동작을 기반으로 사용자가 원하는 정보를 편리하게 검색할 수 있는 개념이다. 따라서, 본 논문에서는 각각의 객체에 대해서 뉴스 커서 관리자, 기사 커서 관리자, 샷 커서 관리자, 대표 프레임 커서 관리자를 나누어 설계한다.

첫째, 뉴스 커서 관리자는 날짜, 시간, 뉴스명, 방송국명, 앵커명에 대한 검색을 하며, 검색된 결과로써 뉴스의 정보, 기사 객체의 수와 기사 객체 식별자들을 검색 인터페이스를 통해 추출한다. 그리고, 뉴스 객체 검색 시 B+-트리 인덱스 생성 여부에 따라 순차 검색 혹은 인덱스 검색이 이루어진다 둘째, 기사 커서 관리자는 기사 객체의 내용에 대한 검색이 이루어지며, 검색된 결과로는 뉴스 객체 식별자, 해당 기사객체의 실제 비디오 스트림 데이타, 텍스트 기사 내용을 저장하고 있는 기사 내용 식별자, 해당 기사 객체에 포함되는 샷 객체의 수, 샷 객체의 식별자 등이 될 수 있다. 셋째, 샷 커서 관리자는 텍스트 캡션 정보와 의미 정보 또한 구현한 역화일 기법을 이용하여 검색이 이루어지며, 커서의 동작에 따라 기사 객체 식별자, 프레임 정보(즉, 시작 프레임 번호, 마지막 프레임 번호), 텍스트 캡션 객체 식별자, 의미 정보 식별자, 해당 샷 객체에 포함되는 대표 프레임의 수, 대표 프레임 식별자들을 검색할 수 있다. 넷째, 대표 프레임 커서 관리자는 의미 정보와 대표 프레임 이미지 전체의 색상과 질감에 대한 특징 벡터 정보를 통해 샷 객체 식별자, 프레임 이미지 데이터 식별자, 의미 정보 식별자들을 접근할 수 있다 특징 벡터 정보는 X-트리의 생성 여부에 따라 순차 검색 혹은 인덱스 검색을 수행한다 검색 API는 크게 커서 개방 및 폐쇄 인터페이스, 커서 설정 인터페이스, 커서를 통한 검색 인터페이스, 그리고 하위 자식 검색(ReadChild) 인터페이스로 분류된다

- 커서 개방
  - snod_opencursor_news, -article, -shot, -rframe
- 커서 폐쇄
  - snod_closecursor_news, -article, -shot, -rframe

- 커서 설정
  - snod_setcursor_news, -article, -shot, -rframe
- 커서를 통한 검색
  - snod_readcursor_news, -article, -shot, -rframe
- 하위 자식(ReadChild) 검색
  - snod_readchildcursor_news, -article, -shot, -rframe

커서 개방 계열 인터페이스는 조건에 대한 검색을 수행 한 후, 각각 커서를 설정한 뒤, 커서 관리 식별자를 반환한다. 커서 폐쇄 계열 인터페이스는 커서 관리 식별자를 시스템에 반환한다. 커서 설정 계열 인터페이스는 객체 식별자를 이용하여 원하는 객체에 커서를 설정한다. 따라서 이 인터페이스를 통해 객체를 직접 접근할 수 있다. 커서를 통한 검색 계열 인터페이스는 검색된 객체들의 전체 정보, 하위 객체의 수와 찾아진 객체의 식별자를 읽기 모드에 따라 얻어오는 함수이고, 하위 자식 검색(ReadChild) 계열 인터페이스는 하위 객체의 식별자를 순차적으로 읽을 수 있는 함수이다.

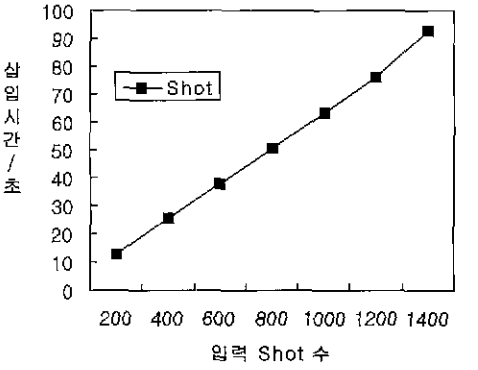
5. 구현 및 성능 평가

4장에서 설계된 저장 관리자는 모든 UNIX 운영체제에서 수행될 수 있도록 표준 C++ 언어를 사용하고, 3장에서 제시한 확장된 SHORE 저장 시스템을 이용하여 구현하였다. 한편 객체지향 기법으로 개발된 SHORE 저장 시스템의 모듈들과 NoD 저장구조의 통합을 위하여 모든 코드를 클래스 형태로 구현하여 향후 SHORE의 새로운 버전에서도 쉽게 통합할 수 있도록 하였다. 또한, 검증 및 성능 평가에 사용할 데이터를 위해 본 논문에서는 1999년 3월 10일부터 1999년 3월 19일까지 10일간 상영된 KBS 스포츠 뉴스 의 비디오를 인덱싱한 결과를 사용하였으며 인덱싱된 데이터 집합은 <표 1>과 같다. 성능평가를 위해 비디오 데이터의 샷을 검색의 대상으로 하였으며, 10일 간의 뉴스 데이터를 이용한 삽입 및 검색성능과, 각 대표프레임에서 추출된 특징벡터를 이용한 삽입 및 검색성능, 아울러 부가 저장공간 측면에서의 성능을 평가한다.

<표 1> NoD 저장 관리자의 검증 데이터

비디오객체 \ 개수	Test Data 수(개)
뉴스	10
아티클	51
샷	1424
억화일을 위한 Term	1637
대표프레임 특징벡터	1424

먼저 뉴스비디오 데이타의 삽입에 대한 성능 평가로는 각 비디오 객체들로부터 추출된 색인어 및 대표프레임 이미지로부터 추출된 15차원의 특징벡터들을 삽입하는데 소요되는 시간을 측정하였다. 각 아티클당 평균 25개의 샷만을 포함하기 때문에 측정 단위로서는 샷이 200개 저장될 때를 기준으로 하였고, 실험에 사용된 비디오 데이터의 저장 및 검색의 측정시간에는 트랜잭션 및 로그 생성시 소요되는 시간이 포함되어 있다.

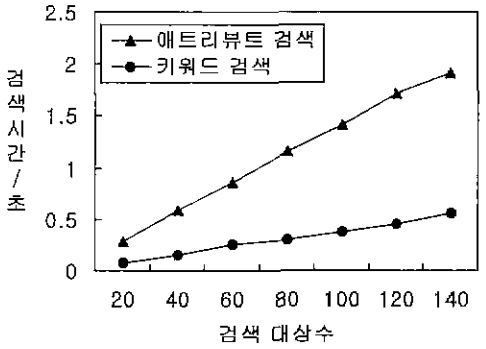


(그림 8) 비디오 객체의 삽입시간

한편 저장된 비디오 데이터의 검색에 대한 성능을 평가하기 위해서 본 논문에서는 다음과 같은 3가지의 항목에 대한 질의를 각각 수행하여 검색 성능을 평가하였다.

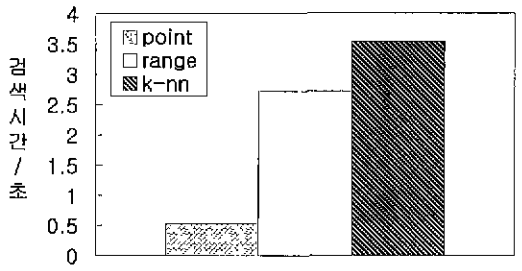
- (애트리뷰트 검색) 뉴스 비디오 데이터의 애트리뷰트를 이용한 검색
- (키워드 검색) 역파일을 사용한 캡션정보의 키워드 검색
- (이미지 내용-기반 검색) X-트리를 사용한 이미지 데이터의 색상 특징벡터 검색

먼저 애트리뷰트 검색의 경우에 일반적인 DBMS에서 사용되는 것과 같은 속성정보를 이용한 검색을 수행한다. 키워드 검색의 경우는 역화일을 이용한 텍스트기반 검색으로서 비디오 파싱시에 추출된 캡션정보를 이용하여 검색을 수행한다. 한편 애트리뷰트 검색과 키워드 검색시에 정확한 성능 평가를 위해 모든 질의에 대해 인덱스 및 커서의 개방 및 폐쇄를 실시하였고, 이에대한 결과는 (그림 9)와 같이 애트리뷰트 검색은 0.5초 이내, 키워드 검색은 2초 이내로 대체로 우수한 성능을 보이고 있다



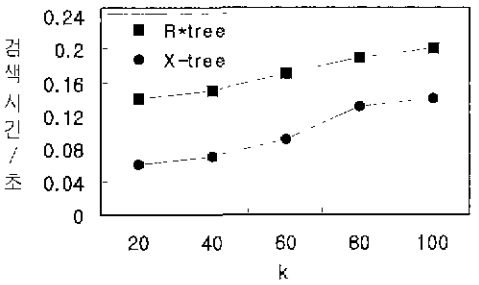
(그림 9) 비디오 객체의 검색시간

또한 이미지 내용-기반 검색성능 측정을 위해, 대표 프레임 이미지 1424개로부터 15차원의 특징벡터를 추출하여 검색성능을 측정한다. 각각의 대표프레임 이미지에 대한 특징벡터의 추출은 한국전자통신연구원(ETRI)에서 제공한 Imageindex 란 프로그램을 사용하여 각각의 이미지에 대해 색상정보를 이용한 15차원의 특징벡터를 추출하였다. 또한 추출된 각각의 특징벡터를 이용하여 포인트 질의, 범위 질의, k-최근접 질의에 대해 Euclidean거리[10]를 사용하여 각각 10개의 질의를 수행한후 그 평균치를 사용하여 성능을 평가한다. 포인트 질의의 경우 주어진 질의벡터와 정확이 일치되는 특징벡터를 검색하는 질의 방법이며, 범위 질의의 경우 질의벡터로부터 주어진 Euclidean거리내에 있는 특징벡터들을 검색하는 방법으로서, 전체 데이터 중에서 1%의 데이터를 검색할 수 있는 값을 범위값으로 사용하였다. 또한 k-최근접 질의는 질의벡터로부터 Euclidean 거리가 가장 가까운 k개의 특징벡터들을 검색하는 방법이며 k-최근접 질의를 위한 k는 10을 사용하고, 이들 각각에 대한 결과는 다음 (그림 10)과 같다. 포인트, 범위 질의는 0.5초 이내로 매우 좋은 성능을 보인 반면, k-최근접 질의의 경우 3.5초로 성능이 다소 떨어짐을 알수있었다. 그 까닭은 k-최근접 질의 처리 알고리즘의 복잡성에 기인한 때문이다



(그림 10) 대표프레임 특징벡터 검색 시간

또한 확장되어진 X-트리의 성능평가를 위해 기존에 알려진 다차원 색인구조인 R*-트리와의 성능을 비교한다. 이를 위해 대표프레임으로부터 추출된 1424개의 특징벡터를 이용하여 k-최근접 질의의 성능을 측정하며, k값으로는 20, 40, 60, 80, 100을 사용한다 (그림 11)은 k-최근접 질의에 대한 두 방법의 성능비교 결과를 보여준다 k가 20일때 R*-트리 0.14초, X-트리 0.06초로 X-트리가 우수한 검색성능을 나타낸다.



(그림 11) 확장된 X-트리와 R*-트리의 검색성능 비교

한편 효율을 검증하기 위해 부가저장 측면의 성능을 측정한다. 이것은 색인되어진 뉴스 비디오 내용의 크기에 대해 역화일에서 사용되어진 후위화일 및 B-트리 인덱스의 비율로서, 순수 문서 데이터 이외에 사용되어진 부가저장 공간비율을 의미한다. 부가 저장 공간에 대한 실험 결과는 <표 2>와 같다.

<표 2> 비디오 객체의 저장시 부가 저장 공간

구 분	크기(byte)
원 데이타	114547
부가 저장공간	152347
부가 저장공간 비율	133.8%

마지막으로 NoD 저장 관리자를 구축하기 위해 확장된 SHORE 저장 시스템을 사용하는 경우와 일반 상용 데이터베이스 시스템을 사용하는 경우를 비교한다 한편 본 논문에서 제안된 확장된 SHORE 하부저장 시스템이 역화일 기법과 X-트리를 사용하여 멀티미디어 데이터의 효율적인 저장 및 검색을 지원하는데 반해, 일반 상업적 데이터베이스 시스템은 애트리뷰트나 부분적으로 텍스트 기반 검색만을 지원하기 때문에 수치적인 성능비교는 불가능한 실정이다. 따라서 본 논문에서는 2가지 접근 방식에 대해 질의처리 및 구현 등의 주요 측면에서의 정성적인 비교를 제시하며, 이는 <표 3>과 같다.

〈표 3〉 SHORE 저장 시스템과 상용 데이터베이스 시스템의 비교

구 분		확장된 SHORE 저장 시스템을 사용하는 경우	상용 데이터베이스 시스템을 사용하는 경우
검색 질의 타입	애트리뷰트 검색	가능	가능
	키워드 검색	역화일을 이용해 검색가능	부분적으로 가능
	내용-기반 검색	X-트리를 이용한 검색가능	불가능
다차원색인구조		X-트리, R트리	R트리 또는 유사
구현의용이성		구현상의 어려움	응용프로그램 제작이 용이

검색질의 타입에 있어서는 확장된 SHORE 저장 시스템을 사용하는 경우, 애트리뷰트 검색, 역화일을 이용한 키워드 검색, X-트리를 이용한 내용-기반 검색을 지원하는데 반해, 일반 데이터베이스 시스템을 사용하는 경우 애트리뷰트 검색은 지원하지만 내용-기반 검색의 지원은 불가능한 실정이다. 한편 시스템 구현 측면에서는 확장된 SHORE 저장 시스템을 사용하는 경우 사용자 인터페이스의 부족으로 구현상의 어려움이 있으나 상용 데이터베이스 시스템을 사용하는 경우 제공되는 다수의 응용프로그램을 통한 구현의 편리성이 있다

6. 결론 및 향후 연구과제

본 논문에서는 NoD 서비스를 위한 뉴스 비디오 멀티미디어 객체를 여러 개의 하위 객체들로 분류한 후, 뉴스 비디오의 파싱 결과로부터 나온 각각의 객체들에 대한 물리적인 비디오 스트림 데이터와 그에 따른 부가적인 색인 정보를 효율적으로 저장하고 관리하기 위한 하부 저장 관리자를 설계 및 구현하였다. 아울러, 이를위해 기존에 SHORE 저장 시스템에서 고차원 특징벡터를 위한 X-트리와, 텍스트 기반 검색을 위한 역화일을 통합하여 확장된 SHORE 저장 시스템을 설계 및 구현하였다 설계한 NoD 서비스용 하부 저장 관리자는 각각의 객체들에서 추출된 부가적인 정보들을 효율적으로 저장할 수 있는 저장 API, 커서를 통한 사용자의 편리한 검색을 지원하는 검색 API, 고차원 특징벡터의 저장 및 검색을 위한 API 그리고 뉴스 비디오 각각의 객체들에서 추출된 텍스트 색인 정보에 대해 빠른 검색을 지원하기 위한 역화일 인덱스 API를 제공한다.

앞으로 향후 연구 과제로는 본 논문에서 제시한 확장된 SHORE 하부저장 시스템을 NoD검색시스템 구축에 적용하여 확장된 SHORE 하부저장 시스템의 효율성을 검증하고, 아울러 News Data를 통해 사용자의 검색을 지원하는 인터페이스를 구축하는 것이다.

참 고 문 헌

[1] V Kobla et al., "Compressed domain video indexing techniques using DCT and motion vector information in MPEG video," in Proc of SPIE-Storage and Retrieval for Image and Video Databases V, San Jose. CA, USA, Vol.3022, pp.200-211. 1997.

[2] H J Zhang, et. al., "Video Parsing, Retrieval and Browsing : An Intergrated and Content-Based Solution," in Proc of ACM Multimedia, 1995

[3] H J Zhang, et. al., "Automatic Parsing and Indexing of News Video," Multimedia Systems. Vol 2, 1995

[4] M Yeung et al., "Video Browsing using Clustering and Scene Transitions on Compressed Sequences," in Proc. of IS&T/SPIE Multimedia Computing and Networking, San Jose, CA, USA, Vol 2417, pp.399-413, 1995.

[5] S Ghandeharizadeh and L. Ramos, "Continuous Retrieval of Multimedia Data Using Parallelism," IEEE Trans Of Knowledge and Data Engineering. Vol 5, No 4, 1993.

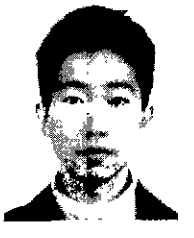
[6] M J Carey, et al., "The architecture of the EXODUS Extensible DMBS," in Proc. VLDB, 1986

[7] DeWitt, D., Luo, J., Patel, J., and Yu, J., Paradise A Parallel Geographic Information System In Proc. of the ACM Workshop on Advances in Geographic Information Systems, November 1993.

[8] Praveen Seshadri, Mark Paskin, PREDATOR : An OR-DBMS with Enhanced Data Types, SIGMOD, 1997

[9] Stefan Berchtold and Daniel A. Keim, High_ Dimensional Index Structures : Database Support for Next Decades Applications, in Tutorial, SIGMOD, 1998

[10] Salton, G., and M McGill, An introduction to Modern Information Retrieval, McGraw-Hill, 1983.



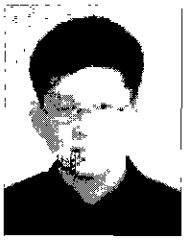
진 기 성

e-mail ksjin@dblab.chonbuk.ac.kr

1999년 전북대학교 컴퓨터공학과 (공학사)

1999년~현재 전북대학교 컴퓨터공학과 석사과정

관심분야 : 멀티미디어 데이터베이스, 멀티미디어 정보검색, 하부저장구조



정 재 욱

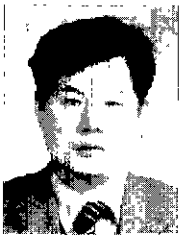
e-mail jujeong@dreamwiz.com

1998년 전북대학교 컴퓨터공학과 (공학사)

2000년 전북대학교 컴퓨터공학과 (공학석사)

2000년~현재 드림위즈㈜ 근무

관심분야 : 멀티미디어 데이터베이스, 멀티미디어 정보 검색



장 재 우

e-mail : jwchang@dblab.chonbuk.ac.kr

1984년 서울대학교 전자계산기공학과(공학사)

1986년 한국과학기술원 전산학과 (공학석사)

1991년 한국과학기술원 전산학과 (공학박사)

1996년~1997년 *Univ of Minnesota*, Visiting Scholar

1991년~현재 전북대학교 컴퓨터공학과 부교수

관심분야 : 멀티미디어 데이타베이스, 멀티미디어 정보 검색, 하부저장구조