멀티미디어 응용을 위한 SHORE 하부저장 시스템의 확장

정재욱, 장재우
전북대학교 컴퓨터공학과

Extension of SHORE storage system for multimidia applications

JaeWuk Chung, JaeWoo Chang
Dept. of Computer Engineering, Chonbuk National University

요 약

컴퓨터 통신 기술의 급속한 발달로 인해 정지영상, 오디오, 비디오와 같은 다양한 미디어로 구성된 대용량의 멀티미디어 자료를 효율적으로 저장하고 관리할 수 있는 하부 저장 시스템이 필요하다. 이러한 멀티미디어 자료에 대한 내용-기반 검색을 위해 텍스트 기반 검색과 색상 또는 질감과 같은 특징벡터에 기반한 검색이 이루어져야 한다. 본 논문에서는 멀티미디어 응용을 위한 하부저장 시스템을 구현하기 위해 미국 위스콘신 대학에서 개발한 지속성 객체 시스템인 SHORE 를 확장하고자 한다. 텍스트 기반 검색을 위해 역화일 구조를 구현하였으며, 고차원의 특징 벡터의 검색을 위해 X-트리를 통합하였다.

1.서론

최근 하드웨어와 컴퓨터 통신 기술의 급속한 발달로 인해 정지영상, 오디오, 비디오와 같은 다양한 미디어로 구성된 대용량의 멀티미디어 자료를 효율적으로 저장하고 관리해야 하며, 이러한 환경하에서 사용자들은 다양한 형태의 멀티미디어 서비스를 요구하는 추세이다. 이에 따라 멀티미디어 자료에 대한 많은 연구가 진행되었다. 멀티미디어 자료의 효율적인 검색을 위해서는 텍스트 기반 검색과 특징벡터에 기반한 검색이 이루어져야 한다.

이를 위하여 본 논문에서는 미국 위스콘신 대학에서 개발한 SHORE 의 하부 저장 시스템을 확장하였다. SHORE 의 SSM(Shore Storage Management)은 가변 길이 레코드를 관리할 수 있는 파일과 B+-트리 및 2 차원 벡터를 처리할 수 있는 R*-트리 등을 포함하고 있다. 그러나 텍스트의 검색과 고차원 특징벡터를 검색할 수 있는 인덱스를 포함하고 있지 않다. 일반적으로 텍스트 검색을 위한 색인 기법으로는 역화일 구조를 주로 사용한다[3]. 아울러 다차원 색인기법인 R*-트리[2]는 차원이 증가함에 따라 겸침 영역이 증가하여 성능이 급격히 저하하는 문제가 있다. 이를 보완하여 고차원에 적합하도록 제안된 다수의 색인 기법이 있으며, 이 가운데 X-트리는 효율적인 고차원 색인 기법으로 알려져 있다[1].

본 논문에서는 SHORE 의 SSM 이 텍스트 기반 검색을 할 수 있도록 B+-트리와 파일을 이용하여 역화일 구조을 설계 및 구현하였고, 고차원 특징벡터의 검색을 위해서 X-트리를 SSM 에 통합하여 구현하였다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 SHORE 를 소개한다. 제 3 장에서는 X-트리와 역화일 구조를 통합한 SHORE SSM 의 확장에 대해 설명한다. 끝으로 제 4 장에서는 결론을 제시한다.

2. SHORE

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. SHORE 는 객체 지향 데이터베이스 기술과 파일 시스템 기술을 합성하여, 기존의 UNIX 파일 시스템 기반 응용 프로그램을 쉽게 접목 시킬 수 있는 장점을 지닌다.

SHORE 는 EXODUS 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 원할하게 통신할 수 있도록 관리하며, SSM 는 실제 시스템에서 생성되고 처리되는 객체를 관리한다.

SSM 은 5 레벨로 구성되어 있다. 레벨 0 은 Device, IO, Buffer, Lock, Log 에 관한 관리자를 제공한다. 레벨 1 는 트랜잭션에 관한 관리자를 제공해 준다. 레벨 2 에서는 파일과 B+-트리, R*-트리에 대한 관리자를 제공해 주며, 레벨 3 은 디렉토리에 관한 관리자를 제공한다. 레벨 4 는 Logical Id 에 관한 관리자

와 실제 SSM 의 사용자 레벨 API 를 제공해 준다[4][5].

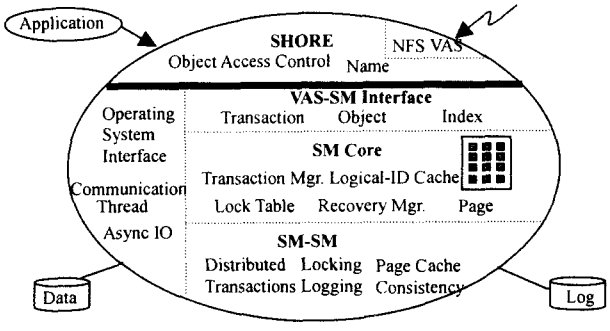

그림 1 SHORE 저장 시스템의 전체적인 구조

3. SHORE SSM 의 확장

본장에서는 멀티미디어 정보검색에 적합한 하부저장 구조를 위해 SHORE SSM 상에서 X-트리와 역화일 구조의 설계 및 구현에 대하여 설명한다.

3.1 X-트리의 설계 및 구현

SSM 은 5 레벨로 구성되어 있다. 여기서 레벨 2 는 B+-트리 관리자, R*-트리 관리자와 파일 관리자등의 색인에 관련된 관리자이며, 이들은 레벨 0 와 1 의 상속을 받아 구현되어 있다. 따라서 색인 관리자에 속하는 X-트리 관리자인 xtree_m 클래스는 레벨 2 에서 구현한다. xtree_m 클래스는 특징 벡터들의 저장, 검색에 대한 메쏘드를 제공해 준다. xtree_m 클래스는 페이지 관리를 해주는 xtree_head_p, xtree_data_p, xtree_dir_p 클래스를 이용하여 구현한다. 각각 페이지 관리 클래스는 page_p 클래스의 상속을 받기 때문에 기본적인 페이지 관리 메쏘드를 이용한다.

xtree_head_p 클래스는 X-트리의 헤더로서 루트 페이지 식별자, 부모페이지가 데이터 페이지인지의 여부, 차원의 수, 데이터 노드 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 저장되며, 이들을 관리하게 된다. 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 특징벡터의 수, 자식 노드의 Split 위치를 나타내는 정보 및 자식 노드의 페이지 식별자가 저장된다. 페이지의 데이터 정보는 그림 2 와 같다.

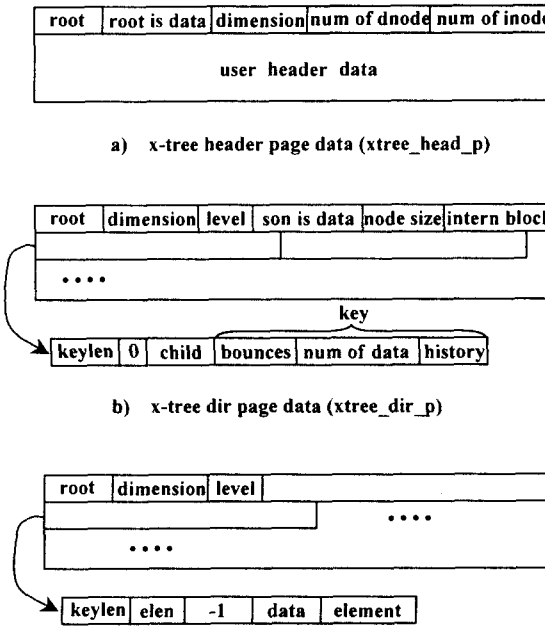

그림 2. 페이지 데이터 구조 정보

레벨 4 의 ss_m 클래스의 메쏘드를 통해 SSM 를 사용자가 사용하게 되므로, X-트리의 사용자 레벨 API 역시 ss_m 클래스의 메쏘드에 확장한다. 확장된 기본적인 메쏘드는 표 1 과 같다.

표 1. X-트리 API

create_xd_index()	X-트리 생성
destory_xd_index()	X-트리 삭제
create_xd_assoc()	X-트리에 특징벡터 삽입
destory_xd_assoc()	X-트리의 특징벡터 삭제
find_xd_assoc()	X-트리를 통한 특징벡터 검색

X-트리의 확장된 검색을 위해 스캔관리를 하는 레벨 4 에 scan_xd_i 클래스를 구현한다. scan_xd_i 클래스는 포인트 질의, 범위 질의, k-nn 질의를 메쏘드를 통해 지원해 준다. scan_xd_i 클래스의 생성자에 의해 질의 방법과 질의 벡터를 지정하게 된다. 생성자는 다음과 같다.

```
scan_xd_i::scan_xd_i(const lvid_t& lvid, cosnt 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 } ;
```

lvid 는 볼륨 식별자, stid 는 X-트리 식별자이고 c 는 질의의 방법을 정하는 매개변수로 t_point 는 포인트 질의를 나타내며,

t_range 는 범위 질의를 나타낸다. t_k_near 는 k-nn 질의이며, k 값을 통해 찾고자 하는 검색의 수를 결정한다. box 는 질의 특징벡터가 된다.

생성자를 통해 검색된 결과는 scan_xd_i 클래스의 next() 메쏘드를 통해 사용자에게 전달 되며 다음 검색 결과로 커서를 움직인다. next() 메쏘드는 다음과 같다.

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

key 를 통해 검색된 특징 벡터 정보를 반환하며, el 은 엘리먼트 정보, elen 은 엘리먼트 길이, 그리고 eof 는 다음 검색 결과의 유무 정보를 반환한다.

3.2 역화일 구조의 설계 및 구현

역화일 구조를 통합하기 위해 SSM 의 레벨 2 에 invertedfile_m 클래스를 설계하였다. 단어색인 파일로는 SSM 에서 기본적으로 제공하는 B+-트리를 이용하였고, 포스팅 파일로는 0~4GB 까지 지원하며, 추가 갱신이 가능한 레코드를 가지는 SSM 파일를 이용한다. 따라서 invertedfile_m 클래스는 B+-트리를 관리하는 btree_m 클래스와 파일를 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. 포스팅 파일의 구조는 그림 3 와 같다. ss_m 클래스의 메쏘드에 확장된 역화일 구조의 사용자 레벨 API 는 표 2 와 같다.

표 2. 역화일 구조 API

create_invt_index()	역화일 생성
destory_invt_index()	역화일 삭제
create_invt_assoc()	역화일에 정보 삽입
destory_invt_assoc()	역화일에 정보 삭제
find_invt_assoc()	역화일을 통한 정보 검색

역화일의 확장된 검색을 위해 scan_invt_i 클래스를 구현하였다. 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, invtid 은 역화일 구조의 식별자이다. term 은 찾고하는 색인어를 나타내며 c 를 통해 이 색인어의 검색조건을 말해준다. nontruc 는 정확 검색은 right 는 우절단 (term*) 검색을 말한다. 생성자에 의해 검색된 결과는 curr() 메쏘드를 통해 사용자에게 반환되며, next() 메쏘드를 통해 커서를 다음 검색 결과로 움직인다.

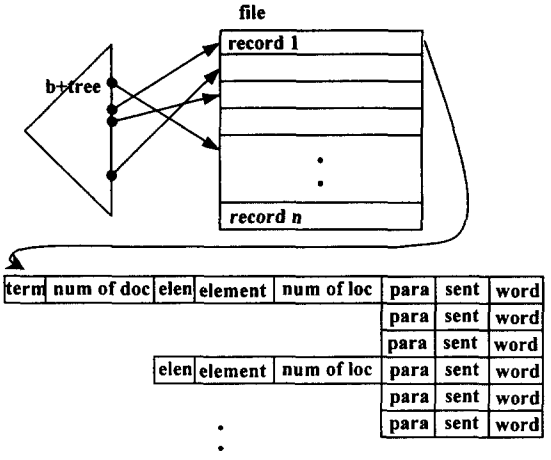

그림 3. 포스팅 파일 구조

5. 결론

대용량의 멀티미디어 정보를 관리하기 위해서는 내용-기반 검색을 지원하는 하부저장 시스템이 필요하다.

본 논문에서는 미국 위스콘신 대학에서 개발한 지속성 객체 시스템인 SHORE 의 하부 저장 구조 시스템에 역화일 구조와 X-트리를 통합함으로써 멀티미디어 정보에 대한 텍스트 기반 검색과 고차원 특징벡터에 대한 내용-기반 검색을 지원한다. 또한 사용자 레벨 API 를 위해 ss_m 클래스를 확장하였고, 확장된 질의 검색을 위해 스캔 관리 클래스를 구현하여, 사용을 용이하게 하였다.

향후 계획으로는 기존 파일 시스템과의 성능 비교와 X-트리의 벌크로드(bulk load)에 대한 구현이 필요하다.

참고 문헌

[1] Berchtold S., Keim D., H-P. Kriegel, "The X-tree: An Index Structure for High Dimension Data", VLDB Journal, Vol. 3, pp. 517-542, 1994.

[2] Beckmann N., Kriegel H.-P, Schneider R., Seeger B., 'Ther R*-tree: An Efficient and Robust Access 메쏘드 for Points and Rectangles', Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, pp.322-331.

[3] W.B. Frakes and R. Baeza-Yates, "Information Retrieval Data Structures&Algorithms," Pretice Hall, 1992

[4] M. J. Carey, et al., "Shoring Up Persistent Applications," in Proc. of ACM SIGMOD, Vol.23, No.2, pp.383-394, 1994

[5] M.J. Carey, et al., "Shoring Up Persistent Applications, "