3차 저장 장치의 장착을 위한 MIDAS-II의 확장

(Extending MIDAS-II for Incorporating Tertiary Storage Devices)

김 영 성 † 강 현 철 †† 김 준 †††
(YoungSung Kim) (Hyunchul Kang) (June Kim)

요 약 MIDAS-II는 한국전자통신연구원에서 개발한 바다 DBMS의 저장 시스템이다. 본 논문에서는, MIDAS-II가 대용량 멀티미디어 데이타 서버의 저장 시스템으로 기능하도록 광 디스크 쥬크박스 또는 테이프 라이브러리와 같은 3차 저장 장치를 효율적으로 장착하기 위한 확장에 대하여 기술하였다. 복수개의 저장 매체 (platter, 예를 들어 디스크 또는 카트리지 테이프)로 구성된 3차 저장 장치용 볼륨 구조를 위하여 기존 MIDAS-II 디스크 볼륨 구조를 확장하여 3차 저장 장치 볼륨의 용량을 Tera 바이트급으로 확대하였다. 그리고 3차 저장 장치에 저장된 LOB(Large Object) 데이타를 효율적으로 처리하기 위하여 기존 MIDAS-II LOB 구조를 확장하였다. 또한 MIDAS-II 공유 메모리 구조, 프로세스 구조, 유틸리티 등을 확장하였고, 기존 응용 프로그램 운용에 변화를 주지 않기 위하여 MIDAS-II API 함수들의 프로토타입을 그대로 유지한 채 그 내부 기능만 확장하였다. 또한 3차 저장 장치로부터의 데이타 검색 성능을 평가하여 본 논문의 3차 저장 장치 장착을 위한 MIDAS-II 확장의 적정성을 확인하였다. 이상의 확장 및 성능 평가는 모두 SunOS 5.4 환경에서 수행되었다.

Abstract MIDAS-II is the storage system for BADA DBMS developed at ETRI. This paper describes the extension of MIDAS-II for incorporating the tertiary storage device such as an optical disk jukebox or a tape library, enabling MIDAS-II to function as a storage system of the data server that stores a massive amount of multimedia data. The MIDAS-II disk volume structure is extended to efficiently function as a volume for the tertiary storage device with multiple platters, which can store huge amount of data of the order of tera bytes. The storage structure of the LOB is changed to efficiently manage the LOB data in the tertiary storage device. The data structures of the shared memory, the process structure, and the utilities in MIDAS-II are also extended to efficiently incorporating the tertiary storage device. The functionalities of each MIDAS-II API function are expanded to handle the tertiary storage device, while the prototypes of those functions are intact in order not to affect the existing application programs. The performance evaluation shows that the extended MIDAS-II works effectively with the tertiary storage device. All these extensions and the performance evaluation are conducted in the SunOS 5.4 environment.

· 본 논문은 한국전자통신연구원에서 수행 중인 "인터넷 멀티미디어 문서 DBMS 개발" 사업의 위탁 결과 일부입니다.
† 학생회원 : 중앙대학교 컴퓨터공학과
yskim@rose.cse.cau.ac.kr
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
hckang@rose.cse.cau.ac.kr
††† 종신회원 : 한국전자통신연구원 인터넷서비스연구부 연구원
june@bada1.etri.re.kr
논문접수 : 1998년 11월 17일
심사완료 : 1999년 5월 26일

1. 서 론

정보화 시대를 맞이하여 대용량 데이타 서비스가 보편화되고 있으며 이를 위한 멀티미디어 데이타 서버의 저장 용량 확대가 요구되고 있다. 기존 DBMS에서 보조 기억 장치인 디스크만으로 대량의 멀티미디어 데이타, 위성 데이타, 공학 및 통계 데이타 등을 효율적으로 저장하기에는 한계가 있다. 이에 성능은 떨어지지만 가격이 싼 광 디스크 쥬크박스(optical disk jukebox) 또

는 테이프 라이브러리(tape library)와 같은 3차 저장 장치의 도입이 요청되고 있다[1][2][3][4][5]. 현재까지 3차 저장 장치는 백업(backup)이나 보관(archival) 용도를 위해서 주로 쓰여 왔지만, 앞으로는 온라인 저장 장치로서 데이타 서버 저장 시스템의 한 요소가 되어야 할 것이다.

MIDAS-II (Multiuser Index-based Data Access System II)[19]는 한국전자통신연구원에서 개발한 바다 DBMS의 저장 시스템이다. 본 논문은 디스크 위주의 MIDAS-II 저장 공간을 3차 저장 장치로까지 확대하기 위한 MIDAS-II의 확장 (설계, 구현 및 성능 평가)에 관한 것이다. 3차 저장 장치를 활용할 수 있도록 시스템을 확장하는 경우에는 대용량과 저성능이라는 3차 저장 장치의 특징을 고려해야 한다. 3차 저장 장치를 사용하는 많은 응용들은 데이타의 초기 운용 과정에서는 수정이 발생할 수 있지만, 특정 시점 이후에는 그 데이타를 읽기 전용으로 사용한다. 이러한 응용의 예로는 은행의 COLD 시스템, 보험사의 보험 처리 시스템, 통신망 상에서의 멀티미디어 데이타 서비스 등이 있다. 본 논문의 MIDAS-II 확장은 이와 같은 응용들에게 3차 저장 장치에 대한 데이타의 온라인 저장 및 검색 기능을 제공하기 위한 것이다. 본 확장은 SunOS 5.4 환경에서 구현되었고, 확장된 MIDAS-II에 대한 성능 평가를 통하여 3차 저장 장치 장착을 위한 본 논문의 확장이 적정함을 확인하였다.

본 논문의 구성은 다음과 같다. 먼저 2절에서 3차 저장 장치를 장착하여 활용하기 위하여 MIDAS-II를 확장하는 데 있어서의 접근 방안을 기술한다. 3절에서 MIDAS-II 확장 내용을 기술하고, 4절에서 확장된 MIDAS-II의 성능 평가 결과를 기술한다. 그리고 5절에서 관련 연구를 살펴본 후, 마지막으로 6절에서 결론 및 향후 연구 내용을 언급한다.

2. 접근 방안

3차 저장 장치는 디스크에 비해 대용량이지만 성능은 떨어진다. 디스크 기반인 MIDAS-II에서 이러한 3차 저장 장치를 단순히 주변 장치로 장착하여 활용하는 것은 비효율적이다. 따라서 3차 저장 장치의 특성을 고려하여 MIDAS-II를 확장해야 한다. 본 절에서는 MIDAS-II 확장 시 기술적 쟁점, 가정 및 설계 원칙에 대해 기술한다.

2.1 확장 시 기술적 쟁점

본 절에서는 MIDAS-II에서 3차 저장 장치를 활용할 경우, 고려해야 할 기술적 쟁점에 대해 기술한다.

- 3차 저장 장치는 디스크에 비해 성능이 상당히 떨어진다. 특히 3차 저장 장치의 성능 특성은 쓰기 연산인 경우에 더욱 두드러지게 나타난다. 따라서, 3차 저장 장치에 저장된 데이타에 대한 변경을 허용할지를 결정해야 한다. 즉, 3차 저장 장치에 디스크와 같이 수시로 변하는 데이타를 저장할 것인지, 아니면 성능을 고려하여 읽기 전용 데이타만 저장할 것인지를 결정해야 한다.
- 3차 저장 장치는 대용량 확보를 위해 사용된다. 좀 더 많은 대용량 확보를 위해서는 광 디스크 쥬크박스 또는 테이프 라이브러리와 같이 복수 개의 저장 매체(platter, 예를 들어 디스크 또는 카트리지 테이프)로 구성된 3차 저장 장치를 지원할 수 있어야 한다. 3차 저장 장치의 각 저장 매체는 3차 저장 장치에 개별적으로 삽입/제거 가능하다. 이 경우 각 저장 매체에 대한 효율적 관리 방안이 필요하다. 예를 들어, 사용자가 원하는 데이타를 저장하고 있는 저장 매체가 3차 저장 장치에서 제거되어 있는 경우, 해당 저장 매체를 삽입만 하면 그 데이타를 접근할 수 있어야 한다.
- 기존 MIDAS-II에서 데이타의 논리적 저장 단위인 화일을 저장하는 디스크 영역을 볼륨이라 부른다. 3차 저장 장치를 장착하여 활용할 경우, 기존 디스크 볼륨 외에 3차 저장 장치 영역을 중심으로 하는 새로운 볼륨 구조가 필요하다.
- 3차 저장 장치에 대한 입출력을 직접 메모리 버퍼를 이용하여 수행하는 것은 메모리 버퍼의 용량이 3차 저장 장치에 비해 매우 작기 때문에 비효율적이다. 따라서 3차 저장 장치를 활용할 경우에는 메모리 버퍼보다 성능은 떨어지지만 용량이 크고 비휘발성인 디스크 캐쉬를 사용한다. 디스크 캐쉬의 캐쉬 단위로 메모리 버퍼와 같이 페이지를 사용하는 것은 3차 저장 장치에 대한 접근 비용이 비싸기 때문에 비효율적이다. 따라서 디스크 캐쉬의 적당한 캐쉬 단위를 결정해야 한다. 그리고, 디스크 캐쉬의 구현 방법 및 교체 알고리즘에 대한 고려도 필요하다.
- 3차 저장 장치에 저장되는 데이타로 용량이 큰 텍스트 문서, 이미지, 비디오 등의 LOB 데이타를 들 수 있다. MIDAS-II에서는 LOB 데이타의 빠른 접근을 위한 인덱스 부분과 실제 LOB 데이타를 저장하는 데이타 부분을 디스크의 동일한 화일에 저장한다. 3차 저장 장치의 LOB에 대해서는, 이들이 동일 화일에 저장되어 있는 경우 디스크 캐쉬 및 입출력을 효율적으로 수행할 수 없다. 따라서 3차 저장 장치에

저장되는 LOB 데이타의 저장 구조에 대한 변화가 필요하다.

- 3차 저장 장치 접근 비용을 줄이기 위해 3차 저장 장치 입출력 스케줄링이 사용된다. 3차 저장 장치 입출력 스케줄링 및 실제 입출력 수행을 구현하는 방법으로 다음 두가지를 고려할 수 있다. (1) 각 작업을 담당하는 라이브러리 모듈을 추가하는 방법과 (2) 각 작업을 담당하는 프로세스를 추가하는 방법이다. 이것은 시스템 구조에 적합하도록 선택되어야 한다. 또한 디스크 캐쉬와 3차 저장 장치 사이의 입출력 단위를 결정해야 한다.
- 디스크 위주의 기존 MIDAS-II에서 운용되던 응용 프로그램과 3차 저장 장치가 장착된 MIDAS-II 간의 호환성 제공 여부를 결정해야 한다. 호환성을 제공하지 않을 경우에는 기존 MIDAS-II 응용 프로그램에 대한 수정이 필요하다. 호환성을 제공하도록 MIDAS-II를 확장하는 경우에는 MIDAS-II의 디스크 볼륨 구조와 MIDAS-II API 함수의 프로토타입을 그대로 유지해야 한다.
- 3차 저장 장치 운용 및 조작과 관련하여 다양한 기능의 유틸리티가 필요하다. 따라서 어떤 기능을 수행하는, 어떤 유틸리티를, 어떤 방법으로 구현할 지를 결정해야 한다.

2.2 가정 및 설계 원칙

본 절에서는 3차 저장 장치를 효율적으로 활용할 수 있도록 MIDAS-II를 확장하는 데 있어, 본 논문의 가정 및 설계 원칙에 대해 기술한다. 먼저 본 논문에서 가정한 사항들은 다음과 같다.

- 3차 저장 장치로는 Tera 바이트 급의 대용량 제공을 위하여 테이프 라이브러리 또는 광 디스크 쥬크박스와 같이 복수 개의 저장 매체로 구성된 장치를 대상으로 한다.
- 3차 저장 장치에 저장될 데이타에 대한 생성은 디스크에서만 이루어지며, 디스크에서 생성 완료되어 향후 읽기 전용으로 사용할 데이타만이 3차 저장 장치에 저장된다. 즉, 데이타의 생성 과정에는 데이타에 대한 수정이 허용되지만, 생성이 완료된 이후에는 그 데이타에 대해 검색만 수행하는 응용을 대상으로 한다. 이와 같은 응용의 예로는 멀티미디어 데이타 서비스 응용을 들 수 있다.
- 3차 저장 장치에 저장되는 데이타(즉, MIDAS-II 화일)의 크기는 하나의 저장 매체 용량을 초과하지 않는다. 즉, MIDAS-II 화일은 두 개의 저장 매체에 나뉘어 저장되지 않고 하나의 저장 매체에 저장된다. 이것은 3차 저장 장치를 구성하는 저장 매체의 용량이 대략 수 GB에 달하기 때문에 그리 큰 제약은 아니라고 생각된다.

이들 가정을 바탕으로, MIDAS-II 확장 시 준수하고자 하는 설계 원칙은 다음과 같다.

- 3차 저장 장치를 구성하는 각 저장 매체가 데이타 저장 매체로서의 독립성을 확보하도록 한다. 이를 위해 각 저장 매체에는 사용자 데이타 및 그에 대한 메타 데이타를 함께 저장한다. 저장 매체가 독립성을 지님으로써 다음과 같은 잇점을 얻을 수 있다. (1) 각 저장 매체의 내용만으로 그에 저장된 데이타에 접근할 수 있다. (2) 3차 저장 장치에 개별적으로 삽입/제거되는 각 저장 매체를 효율적으로 관리할 수 있다. (3) 3차 저장 장치에 동시에 장착될 수 있는 저장 매체의 총 수보다 많은 갯수의 저장 매체에 데이타를 저장할 수 있다. (4) 고장 복구 시 저장 매체들에 저장된 메타 데이타를 참조하여 3차 저장 장치 볼륨을 재구성할 수 있다.
- 3차 저장 장치의 각 저장 매체로서 재기록이 가능한 (rewritable) 저장 매체보다 가격이 싼 WORM (Write Once Read Many) 매체도 사용할 수 있도록 한다. WORM 매체에는 데이타의 추가만 허용되므로 저장 매체의 독립성 확보를 위해 저장 매체에 저장되는 메타 데이타의 종류와 그 저장 방법을 신중히 선택해야 한다.
- 3차 저장 장치 볼륨에 대하여 데이타 입력(저장)과 검색을 동시에 지원할 수 있도록 한다.
- 응용 프로그램에게 기존 MIDAS-II API를 아무 변화없이 그대로 제공하도록 한다. 이것은 공유 메모리와 디스크로 구성된 기존 MIDAS-II 저장 계층 구조 하에서의 데이타 처리 및 관리 기능을 그대로 유지하면서 이에 3차 저장 장치와의 연동 기능이 부가되는 형태로 확장하는 것을 의미한다. 이로 인해 기존 MIDAS-II의 디스크 볼륨을 사용하던 응용 프로그램에게 아무런 영향을 미치지 않고 3차 저장 장치를 장착하여 활용할 수 있다.

3. MIDAS-II 확장

본 절에서는 2절에서 살펴본 사항들을 바탕으로 수행된 MIDAS-II 확장에 대해 기술한다. MIDAS-II에 3차 저장 장치를 장착하기 위해 확장된 사항들은 볼륨 구조, LOB 화일 구조, 프로세스 구조, 공유 메모리 구조, API

함수 기능, 유틸리티 등이다.

3.1 MIDAS-II 볼륨 구조 확장

디스크 기반의 기존 MIDAS-II의 볼륨은 루트 볼륨과 일반 디스크 볼륨으로 나눌 수 있다. 루트 볼륨은 MIDAS-II 볼륨 중 제일 먼저 생성되는 볼륨이며, 일반 디스크 볼륨과는 달리 MIDAS-II의 모든 볼륨에 대한 정보와 고장 복구에 사용되는 마스터 레코드를 가진다. MIDAS-II의 각 볼륨은 하나 이상의 세그먼트(segment)로 구성되며, 각 세그먼트는 연속된 페이지의 집합인 익스텐트(extent)들로 구성된다. 그리고 MIDAS-II 화일은 논리적인 데이타 저장 단위로서 이들 익스텐트들의 집합으로 구성되며, 소속 페이지들의 이중 링크 리스트(doubly-linked list) 구조로 되어 있다. MIDAS-II의 각 볼륨에는 실제 사용자 데이타를 저장하는 데이타 화일 이외에도 볼륨 관리에 필요한 메타 데이타인 카탈로그 화일이 저장된다. MIDAS-II의 데이타 화일의 종류는 일련의 레코드들을 저장하는 일반 화일, 일반 화일의 한 LOB 필드에 속하는 LOB 데이타들을 저장하는 LOB 화일, 일반 화일에 대한 인덱스를 저장하는 인덱스 화일 등이다. 본 논문에서는 특별한 설명이 없는 한 루트 볼륨과 일반 디스크 볼륨을 통칭하여 '디스크 볼륨'이라 부르며, MIDAS-II 데이타 화일을 간단히 '화일'이라 부르도록 한다.

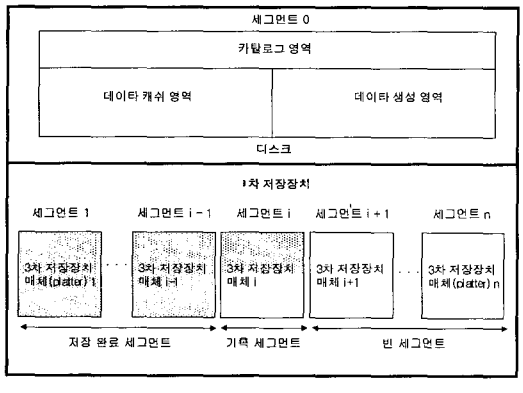


그림 1 3차 저장 장치 볼륨의 구성

n개의 3차 저장 장치 저장 매체 ($n \geq 1$)를 포함하는 확장된 MIDAS-II의 3차 저장 장치 볼륨은, 디스크에 상주하는 첫번째 세그먼트(segment 0)와 각각 하나의 3차 저장 장치 저장 매체에 대응되는 나머지 세그먼트들(segment 1 ~ segment n)로 구성된다 (그림 1 참조). 본 논문에서는 3차 저장 장치 볼륨의 첫번째 세그먼트를 디스크 상주 세그먼트(disk-resident segment)라 부르고, 3차 저장 장치 저장 매체에 대응되는 세그먼트를 저장 매체 세그먼트(platter segment)라 부르도록 한다.

디스크 상주 세그먼트는 저장되는 데이타의 종류에 따라 카탈로그 영역, 데이타 캐쉬 영역, 그리고 데이타 생성 영역으로 나뉘어 사용된다. 카탈로그 영역은 3차 저장 장치 볼륨을 관리하는 데 필요한 카탈로그 정보를 저장하는 데 사용되며, 데이타 캐쉬 영역은 저장 매체 세그먼트에 저장되어 있는 읽기 전용 데이타를 캐쉬하여 저장하는 영역이다. 그리고, 데이타 생성 영역은 화일 초기화 및 아직 생성이 완료되지 않은 화일에 대한 데이타 검색, 추가, 삽입, 수정, 삭제 등의 연산을 수행하는 영역이다.

저장 매체 세그먼트는 디스크 상주 세그먼트의 데이타 생성 영역에서 생성이 완료된 화일들을 읽기 전용으로 사용하기 위해 저장하는 영역이다. 저장 매체 독립성 확보를 위하여 저장 매체 세그먼트에는 화일이 저장될 때 메타 데이타(저장 매체 세그먼트 구성 정보, 화일 접근/구성 정보, LOB 인덱스 화일인 경우 LOB 카탈로그 정보(3.2절 참조) 등)가 함께 저장된다. 저장 매체 세그먼트 구성 정보는 맨 처음 데이타가 저장될 때 한번만 저장되며, 나머지 화일 관련 정보는 해당 화일이 저장될 때 같이 저장된다. 저장 매체 세그먼트에 저장되는 메타 데이타는 기존 MIDAS-II에서는 사용되지 않는 화일의 첫번째 페이지와 마지막 페이지에 저장된다. 이와 같이 함으로써 3차 저장 장치의 저장 매체로 WORM 매체도 지원하면서 저장 매체 독립성을 확보할 수 있다.

저장 매체 세그먼트에 생성이 완료된 화일이 저장될 때는 저장 매체 세그먼트를 순차적으로 하나씩 채워 나간다. 즉, 한 저장 매체 세그먼트에 생성이 완료된 화일을 저장할 수 있는 충분한 가용 공간이 남아 있지 않을 때까지 해당 저장 매체 세그먼트를 채운 후, 비로소 다음 저장 매체 세그먼트에 데이타 저장을 시작한다. 따라서, 저장 매체 세그먼트는 대응되는 저장 매체의 데이타 저장 상태에 따라 다음 3종류로 나뉜다.

(1) 저장 완료 세그먼트 (full segment) : 화일의 저장이 완료되어 더 이상의 저장 가용 공간이 충분히 남아 있지 않은 것으로서 읽기 전용으로만 접근된다.

(2) 기록 세그먼트 (recording segment) : 디스크 상주 세그먼트에서 생성 완료된 화일을 3차 저장 장치로 이동시킬 때 이를 말미에 추가(append)하는 세그먼트로서, 생성 완료된 화일을 저장할 공간이 부족하면 저장 완료 세그먼트가 된다.

(3) 빈 세그먼트 (empty segment) : 아직 화일이 하나도 저장되지 않은 것으로서, 기록 세그먼트가 저장 완료 세그먼트로 되면, 기록 세그먼트 다음의 빈 세그먼트가 새로운 기록 세그먼트가 된다.

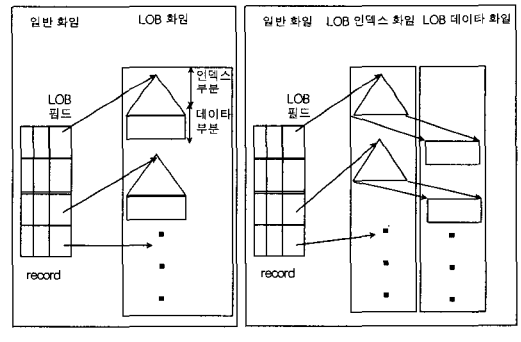
저장 매체 세그먼트에 존재하는 데이타에 대한 접근은 그 데이타를 데이타 캐쉬 영역으로 캐쉬해 온 후 수행된다. 본 시스템에서는 3차 저장 장치의 성능 특성을 고려하여 캐쉬 단위로 페이지가 아닌 익스텐트를 사용하였다. 하지만, 데이타 생성 영역에서 저장 매체 세그먼트로의 데이타 이동은 화일 단위로 이루어 진다. MIDAS-II에서는 화일을 구성하는 페이지들이 이중 리스트로 모두 연결되어 있다. 따라서, 화일 내 어떤 페이지가 다른 곳으로 이동될 경우에는 페이지들 간의 이중 리스트 구조를 유지하기 위해서 전후 페이지 내에 저장되어 있는 포인터를 변환해 주어야 한다. WORM 매체를 사용하면서 화일의 이중 리스트 구조를 유지하기 위해서는 화일 전체가 한꺼번에 이동되어야 한다.

디스크 상주 세그먼트의 용량이 저장 매체 세그먼트들에 비해 매우 작기 때문에 디스크 상주 세그먼트에 데이타를 캐쉬할 공간이나 생성할 공간이 없을 경우에는 디스크 상주 세그먼트의 익스텐트에 대한 교체가 필요하다. 본 논문에서는 익스텐트 교체 알고리즘으로 LRU(Least Recently Used) 알고리즘을 사용하였다. LRU 알고리즘의 교체 대상이 될 수 있는 익스텐트는 (1) 데이타 캐쉬 영역에 해당하는 익스텐트와 (2) 데이타 생성 영역에 해당하는 익스텐트들 중 생성 완료된 화일의 익스텐트로 제한된다. 익스텐트 교체는 다음과 같이 수행된다.

```
ReplaceExtent()
{
  victimExtent ← LRU 알고리즘을 이용하여 교체 대상으로
  선택된 익스텐트;
  if (victimExtent가 데이타 캐쉬 영역에 해당하는 익스텐트) {
    /* 저장 매체 세그먼트에 존재하는 데이타이므로 */
    /* 익스텐트 내 데이타를 무시 */
  }
  else { /* 생성 완료된 화일의 익스텐트일 경우 */
    if (기록 세그먼트의 잔여 공간 < 해당 화일의 용량)    {
      기록 세그먼트 번호 ← 다음 세그먼트 번호;    /* 즉
      빈 세그먼트 */
      화일의 첫번째 페이지 내에 세그먼트 구성 정보 기록;
    }
    화일의 첫번째 페이지 내에 화일 접근/구성 정보 기록;
    if (화일 종류 == LOB 인덱스 화일) {
    화일의 첫번째/마지막 페이지 내에 LOB 카탈로그 정보
    기록;
    }
    기록 세그먼트 내 화일 저장 영역 할당;
    화일 내 모든 페이지 포인터 변환;
    기록 세그먼트의 할당된 영역으로 해당 화일 이동;
  }
  victimExtent를 빈 익스텐트로 표시;
}
```

3.2 MIDAS-II LOB 화일 구조 확장

MIDAS-II는 LOB을 지원하도록 확장된 바 있는데 [20], LOB이 저장되는 공간은 기존 디스크 볼륨이었다. MIDAS-II의 LOB 구조는 LOB 데이타의 바이트 옵셋(offset)을 키로 하는 B^+ 트리 인덱스와 이 인덱스를 통해 접근되는 실제 LOB 데이타의 짝으로 구성되어 있다. 일반 화일 내 각 레코드의 LOB 필드에 대하여 하나의 B^+ 트리 인덱스/LOB 데이타 짝이 존재하며, 하나의 LOB 컬럼 전체에 대하여 하나의 LOB 화일이 할당된다. 즉, MIDAS-II에서 LOB 화일은 여러 B^+ 트리 인덱스/LOB 데이타 짝의 집합이라 할 수 있다 (그림 2a 참조).



(a) 확장 전 (b) 확장 후

그림 2 확장 전/후의 LOB 화일 구조 비교

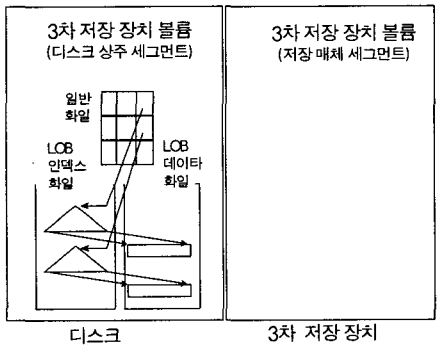
본 절에서는 3차 저장 장치 장착을 위한 MIDAS-II의 확장에서 3차 저장 장치에 저장되는 LOB 화일의 효율적 처리를 위한 확장에 대해 기술한다. 확장된 MIDAS-II에서 LOB 화일의 유형은 다음과 같은 세 종류로 구분된다.

- 기존 MIDAS-II의 LOB 화일 : 3차 저장 장치 볼륨이 아닌 기존 디스크 볼륨에 저장되는 LOB 데이타를 위한 LOB 화일
- LOB 인덱스 화일 : 3차 저장 장치 볼륨에 저장되는 LOB 데이타에 대한 B^+ 트리 인덱스를 저장한 화일
- LOB 데이타 화일 : 3차 저장 장치 볼륨에 LOB 데이타를 저장하기 위한 LOB 저장 구조 중 B^+ 트리 인덱스를 제외한 잎 노드로만 구성된 실제 LOB 데이타를 저장한 화일

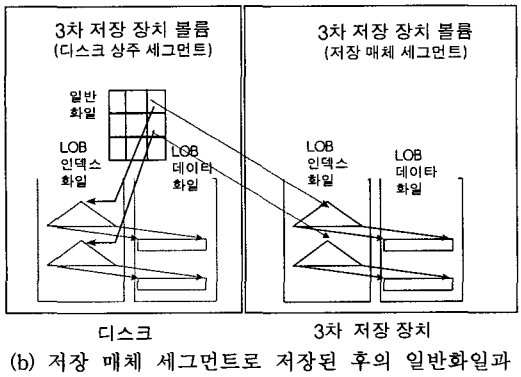
첫번째 LOB 화일은, 3차 저장 장치 볼륨이 아닌 기존 MIDAS-II의 볼륨 구조를 가지는 디스크 볼륨에 저장되는 LOB 화일로서 기존 MIDAS-II의 LOB 화일 구조를 그대로 유지하면서, MIDAS-II의 LOB 관련 API 함수에 의해 기존과 동일하게 처리된다.

두번째와 세번째의 LOB 화일은 3차 저장 장치 볼륨에 저장되는 LOB 화일로서 기존 MIDAS-II의 LOB 화일 구조에서 데이타 부분과 인덱스 부분을 분리하여 각각 LOB 데이타 화일과 LOB 인덱스 화일로 정의한 것이다. 이와 같이 분리함으로써, 빈번하고 빠른 접근이 필요한 LOB 인덱스 화일은 주로 디스크 상주 세그먼트의 데이타 캐쉬 영역에 상주하도록 하고, 대용량인 LOB 데이타 화일은 3차 저장 장치 영역에 저장하는 것이 가능하다.

(그림 2)는 LOB 화일의 확장 전/후의 구조를 나타낸 것이다. (그림 2)와 같이 일반 화일의 LOB 레코드 필드



(a) 생성 시 일반화일과 LOB화일 간의 관계



(b) 저장 매체 세그먼트로 저장된 후의 일반화일과 LOB화일 간의 관계

그림 3 LOB 화일의 생성 시와 교체 후의 루트 페이지의 이동에 따른 문제

는 기존 MIDAS-II에서처럼 LOB 인덱스 화일의 루트 페이지의 식별자를 저장하고, LOB 인덱스 화일의 잎 노드들은 LOB 데이타 화일의 페이지 식별자를 저장하고 있음을 알 수 있다.

3차 저장 장치 볼륨의 디스크 상주 세그먼트에 생성되었던 LOB 인덱스 화일이 저장 매체 세그먼트로 옮겨졌다는 것은 LOB 인덱스 화일의 루트 페이지가 생성 당시와는 다르다는 것을 의미한다. 따라서, 일반 화일 내 레코드의 LOB 필드에 저장되어 있는 LOB 인덱스 화일의 루트 페이지 식별자는 의미를 잃게 된다 (그림 3 참조). 본 논문에서는 이 문제를 해결하기 위하여 LOB 인덱스 화일의 익스텐트 사상 정보를 저장하는 LOB 카탈로그 화일을 신설하여 사용하였다. LOB 카탈로그 화일에 저장되는 익스텐트 사상 정보는, (1) LOB 인덱스 화일 식별자, (2) 생성 당시 디스크 상주 세그먼트의 익스텐트 번호, (3) 저장 매체 세그먼트 번호, (4) 저장 매체 세그먼트의 익스텐트 번호 등으로 구성된다. 응용 프로그램이 일반 화일의 특정 레코드에 대하여 해당 LOB 필드에 저장되어 있는 LOB 인덱스 화일의 루트 페이지 식별자를 이용하여 LOB 데이타에 대한 접근을 요청한 경우,

(1) 해당 LOB 인덱스 화일이 저장 매체 세그먼트로 이동되었을 경우, LOB 카탈로그 화일을 참조하여 이동된 루트 페이지 식별자를 구하여 접근한다.

(2) 이동되지 않은 경우, 주어진 루트 페이지 식별자로 접근한다.

3.3 MIDAS-II 프로세스 구조 확장

기존 MIDAS-II에서는 응용 프로그램과 MIDAS-II 라이브러리가 하나의 MIDAS-II 응용 프로세스를 구성한다. MIDAS-II는 모든 MIDAS-II 프로세스가 공유할 필요가 있는 정보들을 공유 메모리를 통하여 제공한다. 확장된 MIDAS-II에는 3차 저장 장치 입출력 스케줄링을 위한 스케줄러 프로세스(scheduler process)와 3차 저장 장치에 대한 익스텐트 단위의 입출력을 실제로 수행하는 입출력 프로세스(I/O process)가 추가되었다. 본 논문에서는 빠른 구현을 위하여 간단한 FIFO(First In First Out)를 3차 저장 장치 입출력 스케줄링 기법으로 사용하였다. 하지만, 효율적인 3차 저장 장치 입출력을 위하여 입출력 스케줄링 기법을 향후 확장할 계획이다. 입출력 스케줄링 기법의 대체는 스케줄러 프로세스 내 해당 모듈만 수정하면 되도록 하였다.

(그림 4)는 확장된 MIDAS-II의 프로세스들 간의 상호작용을 나타낸 것이다. 각 MIDAS-II 응용 프로세스

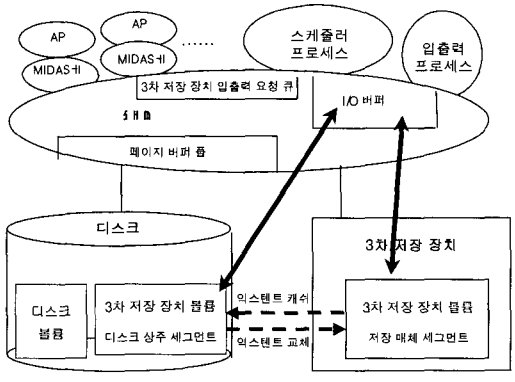


그림 4 3차 저장 장치 입출력을 위해 확장된 MIDAS-II의 프로세스 구조

는 공유 메모리에 존재하는 페이지 버퍼 풀(page buffer pool)을 통하여 디스크에 접근한다. 접근하고자 하는 데이타가 3차 저장 장치 볼륨의 디스크 상주 세그먼트에 존재하지 않고 저장 매체 세그먼트에 존재하는 경우에는, 해당 페이지를 포함하는 익스텐트에 대한 입출력 요청을 3차 저장 장치 입출력 요청 큐(tertiary I/O request queue)에 저장한 후 해당 입출력이 완료될 때까지 기다린다. 스케줄러 프로세스는 입출력 스케줄링 알고리즘에 의해 3차 저장 장치 입출력 요청 큐에 존재하는 입출력 요청들의 수행 순서를 결정한다. 그리고 나서 제일 먼저 수행되어야 할 입출력을 입출력 프로세스에게 요청한다. 입출력 프로세스는 스케줄러 프로세스로부터 받은 입출력 요청을 입출력 버퍼(I/O buffer)를 통하여 수행한 후 스케줄러 프로세스에게 수행 완료를 알린다. 스케줄러 프로세스는 수행 완료된 입출력을 요청한 MIDAS-II 응용 프로세스에게 입출력 수행 완료를 알린 후, 다음에 수행될 입출력을 결정한다. 스케줄러 프로세스와 입출력 프로세스가 수행하는 데 필요한 정보는 3차 저장 장치 상태 정보, 3차 저장 장치 볼륨의 각 세그먼트 정보, 요청된 3차 저장 장치 입출력에 대한 정보, 3차 저장 장치 비용 파라미터 등이다 (표 1 참조).

3.4 MIDAS-II 공유 메모리 구조 확장

MIDAS-II의 공유 메모리는 공유 메모리 헤더(SHMHEAD : shared memory header)와 공유 메모리 헤더가 가리키는 각종 시스템 테이블들로 구성된다. 공유 메모리의 정보는 공유 메모리 관리, 트랜잭션 관리, 버퍼 관리, 디스크 볼륨 관리, 화일 관리, 인덱스 화일 관리, 잠금 관리, 회복 관리 등에 사용된다.

확장된 MIDAS-II에서는 기존 정보 중 디스크 볼륨

표 1 스케줄러 프로세스와 입출력 프로세스가 사용하는 정보

구 분	주 요 내 용
3차 저장 장치 상태 정보	- 3차 저장 장치의 각 드라이브에 적재되어 있는 저장 매체 (즉, 세그먼트) - 3차 저장 장치의 각 드라이브 헤드 위치
3차 저장 장치 볼륨의 각 세그먼트 정보	- 3차 저장 장치 볼륨의 각 세그먼트가 저장된 저장 매체 - 세그먼트 구성 정보
요청된 3차 저장 장치 입출력 정보	- 요청한 프로세스 식별자 - 요청한 페이지 식별자
3차 저장 장치 비용 파라미터	- (표 2) 참조

관리 정보를 확장하였으며, 3차 저장 장치 볼륨 관리 정보, 3차 저장 장치 익스텐트 버퍼 관리 정보, 3차 저장 장치 입출력 관리 정보 등을 추가하였다. 확장된 MIDAS-II에서 확장 또는 추가된 시스템 테이블의 종류는 다음과 같다.

- 디스크 볼륨 관리 정보 중 MIDAS-II의 모든 볼륨에 대한 기본 정보를 저장하고 있는 시스템 테이블인 MIDAS 테이블에 3차 저장 장치 볼륨의 개수를 저장하는 필드를 추가하였다.
- 3차 저장 장치 볼륨 관리 정보를 저장하기 위해 3차 저장 장치 볼륨 테이블(tertiary volume table)과 3차 저장 장치 세그먼트 테이블(tertiary segment table)을 추가하였다. 3차 저장 장치 볼륨 테이블은 3차 저장 장치 볼륨 구성 및 상태 정보를 저장하는 필드들로 구성되어 있고, 3차 저장 장치 세그먼트 테이블은 세그먼트 구성 정보를 저장하는 필드들로 구성되어 있다.
- 3차 저장 장치 익스텐트 버퍼(디스크 상주 세그먼트의 각 익스텐트들) 관리를 위하여 익스텐트 사상 정보 테이블(extent mapping information table), 참조 익스텐트 테이블(referenced extent table) 그리고 참조 익스텐트 파티션(referenced extent partition)을 추가하였다. 익스텐트 사상 정보 테이블은 3차 저장 장치 볼륨의 디스크 상주 세그먼트의 각 익스텐트가 어느 저장 매체 세그먼트의 어느 익스텐트를 캐쉬한 상태인 지를 나타내는 필드로 구성되어 있다. 그리고 참조 익스텐트 테이블과 참조 익스텐트 파티션은 익스텐트 교체 대상 선정 시 LRU 리스트를 구성하고 있다.
- 3차 저장 장치 입출력 관리를 위해서는 3차 저장 장치 입출력 요청 큐, 3차 저장 장치 입출력 버퍼 그리

고 3차 저장 장치 정보 테이블(tertiary device information table) 등을 추가하였다. 3차 저장 장치 입출력 요청 큐는 MIDAS-II 응용 프로세스가 요청한 입출력 정보를 저장하는 것으로서 MIDAS-II 응용 프로세스 식별자, 요청된 익스텐트 번호, 입출력 수행 상태 등을 나타내는 필드들로 구성되어 있다. 3차 저장 장치 입출력 버퍼는 익스텐트 크기의 버퍼 2개로 구성되어 있다. 그리고 3차 저장 장치 정보 테이블은 3차 저장 장치 드라이브 수, 3차 저장 장치 상태 정보(표 1 참조) 등을 저장하는 필드들로 구성되어 있다.

3.5 MIDAS-II API 함수 기능 확장

본 절에서는 각 MIDAS-II API 함수의 기능 확장을 기술한다. 본 논문에서는 함수 프로토타입의 변화없이 3차 저장 장치 지원을 위한 각 함수의 내부 기능만을 확장하였다. MIDAS-II API 함수는 크게 일반 화일 관련 함수와 LOB 화일 관련 함수로 구성된다. 확장된 MIDAS-II의 모든 API 함수는 그 함수의 수행 대상이 되는 데이타가 디스크 볼륨에 존재하는 경우에는 기존 MIDAS-II에서의 수행 과정을 그대로 수행하면 된다. 하지만, MIDAS-II API 함수의 수행 대상이 되는 데이타가 3차 저장 장치 볼륨에 존재하는 경우에는 각각 다음과 같이 수행되도록 한다.

(1) 일반 화일 관련 API 함수의 확장

- midas_createfile : MIDAS-II 볼륨에 화일을 생성하는 데 사용하는 함수로서, 3차 저장 장치 볼륨에 화일을 생성하고자 하는 경우에는 3차 저장 장치 볼륨의 디스크 상주 세그먼트에 생성한다.
- midas_destroyfile : MIDAS-II 볼륨에서 화일을 삭제하는 데 사용하는 함수로서, 삭제하고자 하는 화일이 3차 저장 장치 볼륨에 존재하는 경우에는 그 화일이 생성 중인 경우만 가능하도록 하며, 그렇지 않은 경우는 에러로 처리한다.
- midas_opencursor : 화일을 열어서 커서 할당 및 초기화를 수행한 후 할당된 커서를 리턴해주는 함수로서, 해당 화일이 3차 저장 장치 볼륨에 존재하는 경우에는 먼저 디스크 상주 세그먼트에 그 화일이 존재하는 지를 살펴본 후, 존재하지 않는 경우에는 그 화일의 첫번째 페이지가 속하는 익스텐트를 디스크 상주 세그먼트로 캐쉬해 온 후 커서를 연다. 그리고, midas_opencursor의 모드가 읽기(READ)인 경우에는 해당 화일이 어느 세그먼트에 존재하든지 상관없지만, 모드가 쓰기(WRITE)인 경우에는 화일이 생성 중인 경우에만 가능하며, 저장 매체 세그먼트에 저장된 화일인 경우에는 에러로 처리한다.
- midas_readcursor : 할당된 커서를 이용하여 화일 내 레코드를 읽는 함수이다. 이 함수의 대상이 되는 레코드가 디스크 상주 세그먼트에 존재하지 않는 경우에는 저장 매체 세그먼트로부터 데이타 캐쉬 영역으로 캐쉬해 온 후 접근한다.
- midas_addcursor : 할당된 커서를 이용하여 화일에 레코드를 삽입하는 함수이다. 이 함수의 대상이 되는 화일이 데이타 생성 영역에 존재하는 경우에만 가능하며, 그 외의 영역에 존재하는 화일인 경우에는 midas_opencursor 함수에서 에러가 발생한다. 3차 저장 장치 볼륨의 디스크 상주 세그먼트에서 생성 중이던 화일이 생성 완료되었음을 알리는 경우에는, midas_addcursor의 인자(argument)들 중 Length 인자에 특별한 값을 전달하여 수행한다. 이것은 해당 화일의 모든 익스텐트를 교체 가능하도록 표시함으로써 화일이 향후에는 읽기 전용으로 사용되며, 따라서 저장 매체 세그먼트로 옮겨질 수 있다는 것을 나타낸다.
- midas_deletecursor, midas_updatecursor : 할당된 커서를 이용하여 커서가 가리키는 레코드를 삭제하거나 수정하는 함수이다. 이 함수의 대상이 되는 화일이 데이타 생성 영역에 존재하는 경우에만 가능하며, 그 외의 영역에 존재하는 화일인 경우에는 midas_opencursor 함수에서 에러가 발생한다.

(2) LOB 화일 관련 API 함수의 확장

- midas_createLOB : MIDAS-II 볼륨에 LOB 화일을 생성하는 데 사용하는 함수이다. 3차 저장 장치 볼륨에 LOB 화일을 생성하고자 하는 경우에는 3차 저장 장치 볼륨의 디스크 상주 세그먼트에 LOB 인덱스 화일과 LOB 데이타 화일을 분리하여 생성한다.
- midas_destroyLOB : MIDAS-II 볼륨 내 LOB 화일을 삭제하는 데 사용하는 함수이다. 삭제의 대상이 되는 LOB 화일이 3차 저장 장치 볼륨에 존재하는 경우에는 LOB 인덱스 화일과 LOB 데이타 화일이 생성 중인 경우에만 허용되며, LOB 인덱스 화일과 LOB 데이타 화일을 모두 삭제한다.
- midas_readLOB_file : LOB 화일로부터 원하는 LOB 데이타를 읽어 들이는 데 사용하는 함수이다. 이 함수의 대상이 되는 LOB 인덱스 화일이 3차 저장 장치 볼륨 내 디스크 상주 세그먼트가 아닌 저장 매체 세그먼트에 존재하는 경우에는 LOB 카탈로그 화일을 참조하여 수행되도록 하며 (3.2절 참조), 디스

크 상주 세그먼트에 존재하지 않을 경우에는 데이타 캐쉬 영역으로 캐쉬해 온 후 접근한다.

- midas_addLOB_file : LOB 화일에 LOB 데이타를 삽입하는 데 사용하는 함수이다. 이 함수의 대상이 되는 LOB 인덱스 화일과 LOB 데이타 화일이 생성 중인 경우에만 허용되며, 생성 완료된 화일인 경우에는 에러로 처리한다. 또한, 3차 저장 장치 볼륨의 디스크 상주 세그먼트에서 생성 중이던 LOB 인덱스 화일과 LOB 데이타 화일이 생성 완료되었음을 알리는 경우에는, midas_addLOB_file의 인자들 중 Length 인자에 특별한 값을 전달하여 수행한다. 이것은 해당 LOB 화일들의 모든 익스텐트를 교체 가능하도록 표시하여 LOB 화일들이 향후에는 읽기 전용으로 사용되며, 따라서 저장 매체 세그먼트로 옮겨질 수 있다는 것을 나타낸다.
- midas_deleteLOB_file, midas_updateLOB_file : LOB 화일 내 특정 LOB 데이타를 삭제하거나 수정하는 데 사용하는 함수이다. 이 함수들은 대상이 되는 LOB 인덱스 화일과 LOB 데이타 화일이 데이타 생성 영역에서 생성 중인 경우에만 허용되며, 그 이외의 경우는 에러로 처리한다.

3.6 MIDAS-II 유틸리티 확장

MIDAS-II에 3차 저장 장치를 장착하여 활용하기 위해서는 MIDAS-II에서 제공하는 유틸리티들에 대한 확장도 필요하다. MIDAS-II 유틸리티 확장은 크게 기존의 유틸리티 확장과 새로운 유틸리티 추가로 나눌 수 있다. 기존 유틸리티 중 3차 저장 장치를 활용하는 경우에는 다음과 같이 't' 옵션 (t : tertiary를 의미)을 명시하여 수행하도록 하였다.

- midasinit [-t] : MIDAS-II 프로세스들이 사용할 공유 메모리를 할당하여 초기화하는 유틸리티이다. 't' 옵션은 3차 저장 장치 지원을 위해 확장된 공유 메모리를 사용하고자 하는 경우에 사용한다.
- midasfinal [-t] : 공유 메모리를 해제하는 유틸리티로서, 'midasinit t'를 이용하여 확장된 공유 메모리를 할당하였을 경우에는 't' 옵션을 사용해야 한다.
- createvolume [-t] : MIDAS-II 볼륨을 생성시키는 유틸리티로서, 't' 옵션은 3차 저장 장치 볼륨을 생성하고자 하는 경우에 사용한다. 'createvolume t'는 데이타 캐쉬, 생성, 삽입, 수정, 삭제, 검색 등의 목적으로 사용되는 디스크 상주 세그먼트 1개, 기록 세그먼트 1개 그리고 다수의 빈 세그먼트를 생성한다. 이들 세그먼트들 중 실제로 생성되는 세그먼트는 디스크 상주 세그먼트뿐이며, 나머지 세그먼트는 실제로 생성시키지 않고 디스크 상주 세그먼트에 저장되는 카탈로그 정보만 초기화한다.
- addsegment [-t] : MIDAS-II 볼륨에 세그먼트를 추가하는 유틸리티로서, 't' 옵션은 3차 저장 장치 볼륨의 빈 세그먼트를 추가하는 경우에 사용한다.

확장된 MIDAS-II에 새로이 추가되는 유틸리티의 종류는 다음과 같다.

- rebuildtertiaryvolume : 매체 고장(media failure)으로 인해 MIDAS-II의 디스크가 파괴되어 완전히 복구할 수 없는 경우에는 3차 저장 장치 볼륨의 디스크 상주 세그먼트를 상실하게 된다. 이 경우 저장 매체 세그먼트에 저장된 메타 데이타를 이용하여 디스크 상주 세그먼트를 재생성시킴으로써 3차 저장 장치 볼륨을 재구성한다. 이 유틸리티는 기존 MIDAS-II 고장 회복 기능과 관련하여 구현되어야 하므로, 본 논문에서는 구현하지 않았다.
- addplatter, removeplatter : 3차 저장 장치 볼륨의 해당 세그먼트에 대응되는 저장 매체(platter)를 3차 저장 장치에 삽입하거나, 3차 저장 장치로부터 제거한다. 이 유틸리티의 기능은 디스크 상주 세그먼트에 저장되어 있는 카탈로그 정보에 해당 저장 매체 세그먼트가 삽입되었거나, 제거되었다는 사실을 기록하는 것이다. 이로써 현재 3차 저장 장치에 어떤 세그먼트들이 존재하며, 어떤 세그먼트들이 존재하지 않는 지에 대한 정보의 일관성을 유지할 수 있다. 3차 저장 장치에 동시에 장착할 수 있는 총 저장 매체 수보다 많은 수의 저장 매체 세그먼트를 생성하여 활용하기 위해서는 이 유틸리티가 필수적이다.

4. 성능 평가

본 절에서는 본 논문에서 확장한 MIDAS-II의 3차 저장 장치 볼륨에 저장되어 있는 일반 화일 및 LOB 화일의 검색 성능을 평가한 결과를 기술한다. 본 성능 평가의 목적은, 본 논문에서 확장 구현한 MIDAS-II에 3차 저장 장치를 장착하였을 경우, 3차 저장 장치에 대한 접근을 통한 데이타 검색 및 처리의 성능이 적절한 수준으로 나타나는 지를 평가하기 위한 것이다. 즉, 본 논문에서 수행한 MIDAS-II 확장 구현의 적정성을 평가하기 위한 것이다. 본 성능 평가에서는 3차 저장 장치를 실제 장치 대신 Sony사의 광 디스크 쥬크박스 [6]를 디스크에서 시뮬레이션하였다.

4.1 성능 평가 개요

본 성능 평가는 디스크 볼륨과 3차 저장 장치 볼륨에 각각 저장되어 있는 동일한 크기의 데이타(레코드의 집합 또는 LOB 데이타)를 MIDAS-II API 함수를 이용하여 검색하는 데 걸리는 응답 시간을 각각 측정, 비교함으로써 수행하였다. 일반 화일에 대해서는 화일 내 레코드 전체를 순차적으로 검색하였고, LOB 데이타는 LOB 화일 내의 한 LOB 데이타 바이트 스트림을 모두 검색하였다.

디스크 볼륨에 저장되어 있는 데이타에 대해서는 확장 전 MIDAS-II에서와 동일한 성능을 나타낼 것이다. 하지만, 해당 데이타가 3차 저장 장치 볼륨에 저장되어 있는 경우에는 다른 성능을 나타낼 것이다. 본 성능 평가에서는 먼저 검색 대상 데이타가 기존 디스크 볼륨에 저장되어 있는 경우의 응답 시간을 측정하고, 3차 저장 장치 볼륨의 데이타에 대해서는 다음과 같이 응답 시간을 측정하였다.

3차 저장 장치 볼륨에 저장된 데이타는 디스크 상주 세그먼트의 캐쉬 영역에 존재할 수도 있으며, 캐쉬에는 없고 저장 매체 세그먼트에만 존재할 수도 있다. 따라서 먼저 검색 대상 데이타가 모두 저장 매체 세그먼트에만 존재할 경우 (즉, 최악의 경우로서 캐쉬 적중률이 0%인 경우)의 검색 응답 시간과 검색 대상 데이타가 모두 디스크 상주 세그먼트에 캐쉬되어 있을 경우(즉, 최선의 경우로서 캐쉬 적중률이 100%인 경우)의 검색 응답 시간을 측정하였다. 그리고 캐쉬 적중률이 0%보다 크고 100%보다 작은 경우는 그러한 상황을 인위적으로 만들기가 어려운 관계로, 캐쉬 적중률 0%인 경우와 100%인 경우 양자의 응답 시간으로부터 계산 (interpolation)을 통해 구하였다. 즉, 3차 저장 장치 볼륨에 존재하는 데이타가 디스크 상주 세그먼트에 캐쉬되어 있을 확률 (즉, 디스크 캐쉬 적중률)에 따른 검색 응답 시간을 다음과 같이 계산하였다.

```
검색 응답 시간 (cache hit ratio) =
        (cache_hit_ratio × all_hit_time) +
        (1-cache_hit_ratio)×
            (all_miss_time-all_miss_switch_seek_time) +
        all_miss_switch_seek_time
        (0<cache_hit_ratio<1)
```

여기에서, cache_hit_ratio는 0에서 1사이의 임의의 디스크 캐쉬 적중률을 나타낸다. all_hit_time은 검색 대상 데이타가 모두 디스크 상주 세그먼트에 캐쉬되어 있을 경우의 검색 응답 시간을 나타내고, all_miss_time은 검색 대상 데이타가 모두 저장 매체 세그먼트에만 존재할 경우의 검색 응답 시간을 나타낸다. 그리고, all_miss_switch_seek_time은 검색 대상 데이타가 모두 저장 매체 세그먼트에만 존재할 경우의 검색 응답 시간 중, 3차 저장 장치의 매체 교체 시간 (platter switch time)과 원하는 데이타가 존재하는 위치로 헤드를 옮기는 시간 (seek time)의 합을 나타낸다.

all_miss_switch_seek_time항이 위 계산 공식에서와 같이 사용되는 이유는 다음과 같다. MIDAS-II 응용 프로그램은 원하는 데이타가 캐쉬되어 있지 않은 경우에 3차 저장 장치의 저장 매체 세그먼트로부터 해당 데이타를 디스크 상주 세그먼트로 캐쉬해 오도록 스케줄러 프로세스에게 3차 저장 장치 입출력을 요청하게 된다. 스케줄러 프로세스에게 요청하는 3차 저장 장치 입출력 요청 횟수는 디스크 캐쉬 적중률에 반비례한다. 임의의 디스크 캐쉬 적중률에 따른 검색 응답 시간 중 3차 저장 장치 입출력을 위해 응용 프로그램이 기다리는 시간은, 입출력 요청 횟수에 상관없이 거의 일정한 시간 (매체 교체 시간((platter switch time), 헤드 이동 시간 (seek time) 등)과 나머지 입출력 요청 횟수에 비례하는 시간 (데이타 전송 시간 (data transfer time), 3차 저장 장치 입출력을 위한 스케줄러 프로세스 및 입출력 프로세스의 수행 시간 등)으로 구성된다. 즉, all_miss_switch_seek_time은 디스크 캐쉬 적중률에 상관없이 거의 일정하므로 상수로 취급하여 위 공식에서와 같이 사용하였다.

확장된 MIDAS-II는 3차 저장 장치를 디스크에서 시뮬레이션했기 때문에 3차 저장 장치 접근 시 실제 시간을 측정하는 것은 불가능하다. 하지만, 본 논문에서는 (표 2)와 같은 3차 저장 장치 관련 비용 파라미터 (Sony사의 광 디스크 쥬크박스의 경우[6])를 사용하여 3차 저장 장치 접근 시간을 계산하였다. 그리고, 성능 평가를 위하여 사용하는 3차 저장 장치 볼륨 및 볼륨 내 세그먼트 갯수, 크기, 구성, 그리고 3차 저장 장치 볼륨 내에 저장되는 일반 화일과 LOB 화일의 크기 등은 (표 3)과 (표 4)의 값을 사용하였다.

본 성능 평가의 일반 화일 검색 성능 및 LOB 화일 검색 성능 측정은 각 화일 또는 LOB 데이타 크기를 임의의 순서대로 검색하도록 한 후 각 크기에 대하여 5번의 검색 시간을 측정하여 평균한 것이다.

4.2 성능 평가 결과 및 분석

본 성능 평가는 일반 화일 검색 성능 및 LOB 화일 검색 성능을 측정한 것이다. (그림 5) ~ (그림 7)은 디스크 볼륨에 저장되어 있는 일반 화일 검색 시 그 응답 시간과 3차 저장 장치 볼륨에 저장되어 있는 일반 화일 검색 시 디스크 캐쉬 적중률 변화에 따른 응답 시간을

표 2 3차 저장 장치 접근 비용 파라미터 설정 값

파라미터	설정 값
Switch Time (sec)	8 sec
Transfer Rate (MB/sec)	0. 8 MB/sec
Seek Start (sec)	0.5 sec

표 3 일반 화일을 위한 볼륨 및 화일 관련 파라미터 설정 값

파라미터	설정 값
Page Size (KB)	4 KB
# of Extent (per Segment)	190
Size of Extent (page)	5 pages
# of Segment	3
Size of Record (byte)	88 bytes
Size of File (record)	1000, 10000, 20000 records

표 4 LOB 화일을 위한 볼륨 및 화일 관련 파라미터 설정 값

파라미터	설정 값
Page Size (KB)	4 KB
# of Extent (per Segment)	330
Size of Extent (page)	64 pages
# of Segment	3
Size of LOB Data (MB)	1, 5, 10, 20 MB

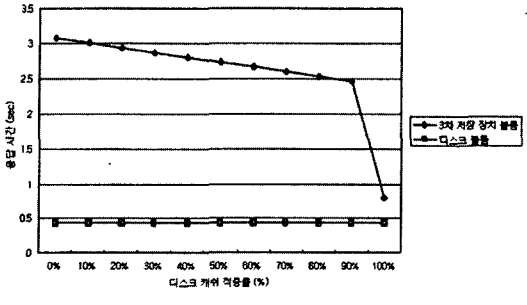


그림 5 일반 화일 접근 시 응답 시간 : 레코드 1000개

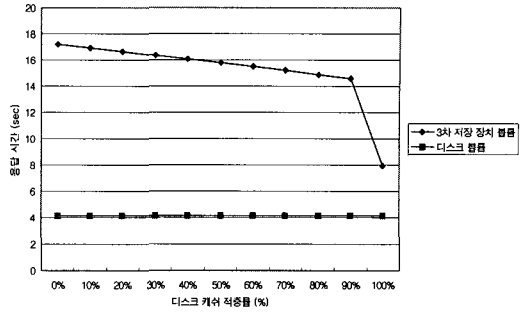


그림 6 일반 화일 접근 시 응답 시간 : 레코드 10000개

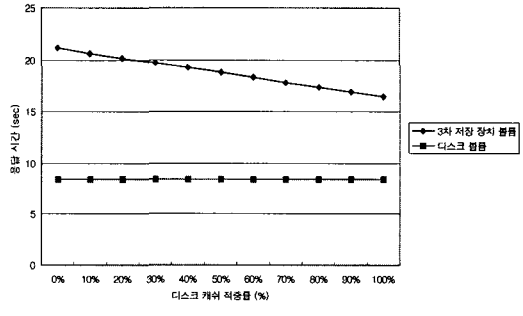


그림 7 일반 화일 접근 시 응답 시간 : 레코드 20000개

나타낸 것이다. (그림 5)는 화일에 1,000 개의 레코드가 있을 때 이들을 순차적으로 모두 검색해 내는 데 걸린 시간을 나타내고 있으며, (그림 6)과 (그림 7)은 각각 화일 내 레코드의 수가 10,000 개와 20,000 개인 경우의 성능을 나타낸 것이다. 한편, (그림 8) ~ (그림 11)은 각각 1M, 5M, 10M, 20M 크기의 LOB 데이타를 검색하는 경우의 응답 시간을 나타낸 것들이다.

그림에서와 같이 디스크 캐쉬 적중률이 높아짐에 따라 3차 저장 장치 볼륨에 존재하는 데이타에 대한 검색 응답 시간이 디스크 볼륨에 존재하는 데이타에 대한 검색 응답 시간에 근접함을 알 수 있다. 특히, 일반 화일의 경우보다는 LOB 화일의 경우에 그 근접도가 현저하였다. 그 이유는 다음과 같다. 확장된 MIDAS-II에서 3차 저장 장치 볼륨 내 원하는 페이지에 대한 접근은, 해당 페이지가 디스크 상주 세그먼트에 캐쉬되어 있는 지의 여부를 살펴본 후 (캐쉬되어 있지 않으면 캐쉬해 온 후) 페이지 식별자를 디스크 상주 세그먼트 내 페이지를 가리키도록 변환한 후 해당 페이지를 접근하게 된다. 일반 화일의 경우에는 페이지 캐쉬 여부 확인과 페이지

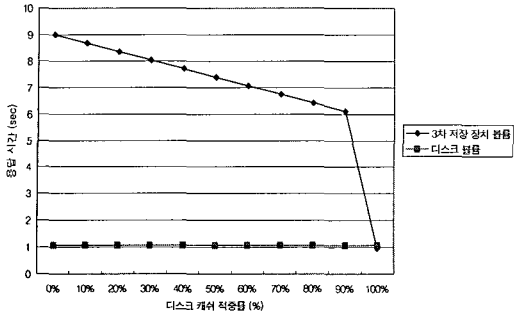


그림 8 LOB 화일 접근 시 응답 시간 : 1M의 LOB 데이타

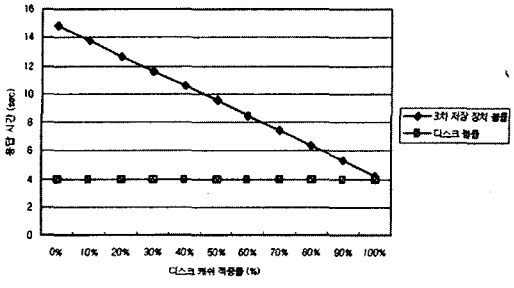


그림 9 LOB 화일 접근 시 응답 시간 : 5M의 LOB 데이타

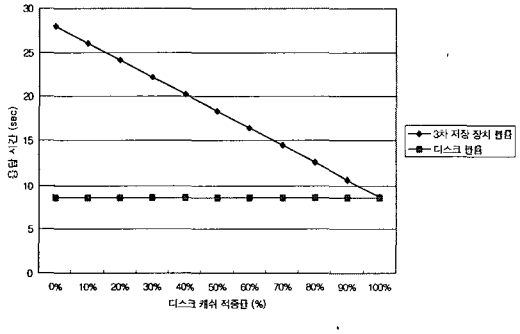


그림 10 LOB 화일 접근 시 응답 시간 : 10M의 LOB 데이타

식별자 변환 작업이 페이지 내의 각 레코드마다 한번씩 수행되어야 하므로 매우 빈번하게 발생한다. 따라서, 디스크 캐쉬 적중률이 100%가 되더라도 디스크 볼륨에 존재하는 데이타에 대한 검색 응답 시간과 완전하게 일치하지 않는다. 반면 LOB 화일의 경우에는 페이지 캐쉬 여부 확인과 페이지 식별자 변환 작업이 페이지 당

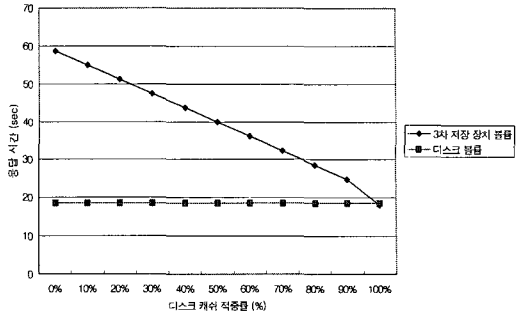


그림 11 LOB 화일 접근 시 응답 시간 : 20M의 LOB 데이타

표 5 3차 저장 장치 볼륨 검색 성능 (디스크 볼륨 검색 성능을 1로 정규화)

캐쉬적중률 / 데이타 크기	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1000개	7.04	6.89	6.73	6.58	6.42	6.27	6.11	5.96	5.80	5.65	1.82
10000개	4.16	4.09	4.02	3.95	3.88	3.81	3.74	3.67	3.60	3.52	1.91
20000개	2.53	2.48	2.42	2.36	2.31	2.25	2.20	2.14	2.09	2.03	1.97
1 MB	8.44	8.14	7.84	7.54	7.23	6.93	6.63	6.33	6.03	5.73	0.92
5 MB	3.70	3.43	3.17	2.91	2.64	2.38	2.12	1.85	1.59	1.33	1.06
10 MB	3.25	3.03	2.81	2.58	2.36	2.13	1.91	1.69	1.46	1.24	1.01
20 MB	3.17	2.97	2.76	2.56	2.36	2.16	1.95	1.75	1.55	1.35	0.97

* 화일 크기 : 일반 화일의 경우는 레코드의 갯수를 의미하고, LOB 화일의 경우는 LOB 데이타의 크기를 의미함

한번만 발생한다. 따라서 3차 저장 장치 볼륨에는 작은 크기의 레코드들로 구성된 일반 화일보다는 대용량의 LOB 화일을 저장하는 것이 검색 성능 면에서 더 효율적임을 알 수 있다. 그리고 일반 화일 및 LOB 화일 양자 모두 그 크기가 커짐에 따라 디스크 볼륨으로부터의 검색 성능 대 3차 저장 장치로부터의 검색 성능의 비율이 점차 작아지는 것으로 나타났다. 이는 3차 저장 장치 접근 시 높은 지연 시간 등의 오버헤드 때문에 한 번 접근 시 많은 양의 데이타를 검색하는 것이 효율적이라는 것을 나타내고 있다. 따라서 3차 저장 장치 볼륨에는 크기가 작은 화일보다는 크기가 큰 화일을 저장하는 것이 더 효율적이다.

(그림 5) ~ (그림 11)에 나타낸 3차 저장 장치 볼륨에 대한 검색 응답 시간을, 디스크 볼륨에 대한 응답 시간을 1로 정규화하여 나타내면 (표 5)와 같다. 일반 화일인 경우의 3차 저장 장치 볼륨에 대한 응답 시간은

디스크 볼륨에 대한 응답 시간의 약 1.82~7.04 배 정도로 나타났으며, LOB 화일인 경우에는 약 0.92~8.44 배 정도로 나타났다. 디스크 캐쉬 적중률이 0%일 때, 일반 화일의 경우에는 3차 저장 장치 볼륨에 대한 검색이 디스크 볼륨인 경우에 비하여 약 2.53~7.04 배 정도 성능이 떨어졌으며, LOB 화일의 경우에는 약 3.17~8.44 배 정도 성능이 떨어졌다. 그리고 디스크 캐쉬 적중률이 100%일 때에는, 일반 화일의 경우에는 약 1.82~1.97 배 정도 성능이 떨어지며, LOB 화일의 경우에는 0.92~1.01 배 정도로 성능이 나타났다.

5. 관련 연구

3차 저장 장치와 관련된 기존 연구로는 3차 저장 장치를 활용하기 위한 요소 기술에 대한 연구와 3차 저장 장치를 활용하기 위한 실제 시스템의 확장에 대한 연구 등이 있다. 전자의 연구로는 3차 저장 장치에 저장된 데이타를 디스크로 캐쉬하여 접근하는 디스크 캐싱 기법[3][6][7][8][9][10][11], 3차 저장 장치에 대한 접근 시간을 감소시키기 위한 입출력 스케줄링 기법[6][7][10][12][28][29][30], 3차 저장 장치에 저장된 데이타의 효율적인 조인 연산 처리 기법[6][9][13][14][15][27], 3차 저장 장치에 최적의 데이타 배치 기법[16], VOD와 같은 응용에서의 활용 방안[17][18] 등의 연구가 진행되고 있다.

3차 저장 장치를 활용하기 위한 실제 시스템의 확장에 대한 기존 연구는 크게 3차 저장 장치를 지원하는 화일 시스템에 대한 연구와 기존 DBMS에서의 3차 저장 장치 지원을 위한 확장 연구로 나눌 수 있다. 먼저 3차 저장 장치를 지원하는 화일 시스템에 대한 연구에는 계층적 저장 관리 시스템 (HSM : Hierarchical Storage Management System)에 대한 연구와 로그 구조 화일 시스템(LFS : Log-structured File System)[21]을 3차 저장 장치에 적용하는 연구가 있다. HSM은 종종 가상 디스크(virtual disk)라는 용어로 사용되며, 3차 저장 장치를 활용하여 디스크를 더 크게 보이도록 해 준다. HSM은 가상 메모리 시스템과 같이 자동으로 자주 접근되는 데이타는 빠른 저장 매체(즉, 디스크)에 유지하고, 거의 접근하지 않는 데이타는 값싼 저장 매체(즉, 3차 저장 장치)에 유지하는 방식이다. HSM은 대용량 데이타의 관리가 편리하다는 장점을 지닌다. 하지만, 데이타 이동 단위가 화일이기 때문에 화일의 일부분만 필요할 경우에는, 3차 저장 장치의 데이타 전송폭(bandwidth)과 디스크 용량을 낭비할 수 있다는 단점이 있다. 계층적 저장 관리 시스템으로 구현된 시스템으로는 유니트리(Unitree)[22][23], 데이타트리(Datatree)[23] 등이 있다. LFS는 새로운 데이타 블록들을 로그(log)라는 순차적 구조로 저장한다. 이러한 순차적 구조를 통하여 쓰기 성능에 향상을 가져 왔으며, 시스템 손상 시 빠른 회복을 지원한다. 로그 구조 화일 시스템을 3차 저장 장치로 확장하여 구현한 시스템에는 LTS[24], Highlight[25], xFS[26] 등이 있다.

3차 저장 장치를 활용하도록 기존 DBMS 또는 저장 시스템을 확장하는 연구는 U.C. Berkeley의 POSTGRES[6][8], U. of Wisconsin의 SHORE[7] 등이 있다. [8]에서는 기존 POSTGRES에 소니 광 쥬크박스(Sony optical jukebox)를 장착하여 활용할 수 있도록 확장하였다. 전송 시간 지연을 보완하기 위한 방안으로 디스크를 3차 저장 장치의 캐쉬 영역으로 사용하였으며, 캐쉬 영역의 데이타 교체 알고리즘으로 LRU를 사용하였다. [6]에서는 [8]에서 확장한 POSTGRES를 이용하여 질의 최적화 및 3차 저장 장치 입출력 스케줄링 기법에 대하여 연구하였다. [7]에서는 LFS를 사용하여 SHORE가 테이프 볼륨을 지원하도록 확장하였다. 이를 위하여 블록 지향 테이프 입출력 드라이버(block-oriented tape I/O driver), 3차 저장 장치 저장 볼륨 관리기(tertiary storage volume manager), 디스크 캐쉬 버퍼 관리기(disk-cache buffer manager), 캐쉬 볼륨 관리기(cache volume manager) 등의 모듈을 추가하였다. POSTGRES에서와 마찬가지로 캐쉬 교체 알고리즘은 구현이 간단한 LRU를 사용하였다.

SHORE 확장 연구에서는 3차 저장 장치로 테이프 라이브러리를 사용하였고, POSTGRES 확장 연구에서는 광 디스크 쥬크박스를 사용하였다. 본 논문에서는 이들 뿐 아니라 복수 개의 저장 매체로 구성된 3차 저장 장치를 대상으로 하였다. SHORE는 WORM 매체를 지원할 수 있지만 LFS를 사용하기 때문에 3차 저장 장치 용량을 낭비할 수 있다. 특히, 데이타에 대한 수정이 빈번히 발생하는 응용은 효율적으로 지원하기 힘들다 POSTGRES도 WORM 매체를 지원하지만 WORM 매체에 저장된 데이타는 수정할 수 없다. 본 논문에서 확장된 MIDAS-II도 WORM 매체에 저장된 데이타를 수정할 수는 없지만, 사용자가 충분히 디스크에서 수정한 후 3차 저장 장치에 저장할 수 있다. 즉, 사용자 의사를 반영하여 데이타를 3차 저장 장치에 저장하도록 함으로써 SHORE와 POSTGRES에서 발생하는 단점을 보완할 수 있다. 또한 본 시스템은 타 시스템과는 달리 각 저장 매체의 독립성을 유지함으로써 저장 매체 관리와 용량 측면에서 장점을 지닌다. 확장된 POSTGRES,

SHORE와 본 논문에서 확장한 MIDAS-II를 비교하면 (표 6)과 같다.

표 6 확장된 MIDAS-II와 타 시스템의 비교

항목 \ 시스템	SHORE[7]	POSTGRES[8]	MIDAS-II
캐쉬 교체 알고리즘	LRU	LRU	LRU
캐쉬 단위	테이프 블록	익스텐트	익스텐트
WORM 매체 지원	O (용량 낭비)	O (수정 못함)	O (수정 못함)
저장 매체 독립성 (저장 매체 개별 삽입/제거)	X	X	O
대상 3차 저장 장치	테이프 라이브러리	광 디스크 쥬크박스	테이프 라이브러리, 광 디스크 쥬크박스 등 복수 개의 저장 매체로 구성된 3차 저장 장치

6. 결론 및 향후 연구

본 논문에서는 한국전자통신연구원에서 개발한 바다 DBMS의 저장 시스템인 MIDAS-II의 저장 공간을 3차 저장 장치로까지 확대하기 위한 MIDAS-II의 확장 설계, 구현 및 성능 평가를 수행하였다.

본 논문에서는 MIDAS-II의 볼륨 구조, LOB 저장 구조, 프로세스 구조, 공유 메모리 구조, API 기능, 유틸리티 등을 확장하여 MIDAS-II가 복수 개의 저장 매체(platter)로 구성된 3차 저장 장치를 장착하여 활용할 수 있도록 하였다. 본 논문에서는 3차 저장 장치 장착 시의 데이타 검색 성능을 평가하여 본 구현의 적정성을 확인하였다. 확장된 MIDAS-II는 다음과 같은 특징을 지닌다.

- 확장된 MIDAS-II는 멀티미디어 서비스 응용과 같이 초기에만 데이타 수정이 발생하고, 특정 시점 이후에는 그 데이타를 읽기 전용으로 사용하는 응용에 적합하다.
- 3차 저장 장치의 각 저장 매체가 저장 매체로서의 독립성을 지닌다.
- 3차 저장 장치의 저장 매체로 WORM 매체도 사용할 수 있다.
- 3차 저장 장치에 대한 온라인 데이타 입력(저장) 및 검색을 동시에 지원한다.
- 기존 MIDAS-II 디스크 볼륨을 사용하던 응용 프로그램에 영향을 주지 않는다.
- 3차 저장 장치에 각 저장 매체를 개별적으로 삽입/제거할 수 있어, 3차 저장 장치에 동시에 장착 가능한 저장 매체의 총 수보다 많은 개수의 저장 매체에 데이타를 저장할 수 있다.

본 논문의 향후 주요 연구 과제로는 다음 내용들을 들 수 있다. 첫째, LOB 화일을 구성하는 LOB 인덱스 화일과 LOB 데이타 화일을 전자는 디스크 볼륨에 후자는 3차 저장 장치 볼륨에 각각 따로 생성하여 LOB 검색의 효율성을 높이는 문제이다. 이를 위해서 현재 MIDAS-II API의 LOB 화일 생성 함수 프로토타입의 확장 또는 새로운 API 함수의 추가가 필요하다. 둘째, LOB을 구성하는 익스텐트들 간의 순차성 (특히, 비디오 또는 오디오와 같은 연속 매체의 경우에)을 고려한 보다 효율적인 3차 저장 장치 입출력 스케줄링 기법에 대한 연구 및 구현이다. 셋째, 확장된 MIDAS-II와 질의 처리기를 연동하여 바다 DBMS의 응용 프로그래밍 환경 (ESQL/C, MS-ODBC 또는 SQL CLI)에서 3차 저장 장치를 활용하도록 하는 연구이다.

참 고 문 헌

[1] M. Carey et al., "Tape Hold Data, Too: Challenges of Tuples on Tertiary Store," Proc. ACM SIGMOD Int'l Conf., 1993, pp. 413-417.

[2] M. Stonebraker, "Managing Persistent Objects in a Multi-Level Store," Proc. ACM SIGMOD Int'l Conf., 1991.

[3] S. Prabhaker et al., "Tertiary Storage: Current Status and Future Trends," Technical Report TRCS96-21, Univ. of California, Santa Barbara, 1996.

[4] C. Mohan, "A Survey of DBMS Research Issues in Supporting Very Large Tables," Proc. Int'l Conf. on Foundations of Data Organization and Algorithms, 1993, pp. 279-300.

[5] P. Selinger, "Predictions and Challenges for Database Systems in the Year 2000," Proc. Int'l Conf. on VLDB, 1993, pp. 667-675.

[6] S. Sarawagi, "Query Processing in Tertiary Memory Databases," Proc. Int'l Conf. on VLDB, 1995, pp. 585-596.

[7] J. Yu and D. Dewitt, "Query Pre-execution and Batching in Paradise: A Two-Pronged Approach to the Efficient Processing of Queries on Tape-Resident Data Sets," Proc. Int'l Conf. on Scientific and Statistical Database Management, 1997.

[8] M. Olson, "Extending the POSTGRES Database System to Manage Tertiary Storage," Master's

thesis, Univ. of California, Berkeley, 1992.
[9] S. Sarawagi and M. Stonebraker, "Single Query Optimization for Tertiary Memory," Sequoia 2000 Techical Reports S2K-94-45, Dec. 1993.
[10] S. Sarawagi, "Database Systems for Efficient Access to Tertiary Memory," IEEE Symposium on Mass Storage Systems, 1995, pp. 120-126.
[11] S. Sarawagi and M. Stonebraker, "Reordering Query Execution in Tertiary Memory Databases," Proc. Int'l Conf. on VLDB, 1996, pp. 156-167.
[12] B. Hiller et al., "Random I/O Scheduling in Online Tertiary Storage Systems," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 1996, pp. 195-204.
[13] J. Myllymaki and M. Livny, "Disk-Tape Joins: Synchronizing Disk and Tape Access," Proc. ACM SIGMETRICS, 1995, pp. 279-290.
[14] J. Myllymaki and M. Livny, "Efficient Buffering for Concurrent Disk and Tape I/O," Proc. Performance, 1996, pp. 453-471.
[15] J. Myllymaki and M. Livny, "Relational Joins for Data on Tertiary Storage," Proc. Int'l Conf. on Data Eng., 1997, pp. 159-168.
[16] S. Christodoulakis et al., "Principles of Optimally Placing Data in Tertiary Storage Libraries," Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp. 236-245.
[17] A. Chervenak et al., "Storage Systems for Movies-on-Demand Video Servers," IEEE Symposium on Mass Storage Systems, 1995.
[18] M. Kienzle et al., "Using Tertiary Storage in Video-On-Demand Servers," Proc. CompCon, 1995, pp. 225-233.
[19] 이진수 외, "MIDAS-II의 설계 및 구현", 한국정보과학회 가을학술발표논문집, 1993, pp. 183-186.
[20] 류은숙 외, "바다 DBMS의 멀티미디어 데이타 지원을 위한 확장", 한국정보과학회 논문지(C), 2권 4호, 1996.12, pp. 339-356.
[21] M. Rosenblum, "The Design and Implementation of a Log-Structured File System," Ph.D. Thesis, University of California, Berkeley, 1992.
[22] F. Kim, "UniTree: A Closer Look At Solving The Data Storage Problem," White Paper, UniTree Software Inc. Available on http://www.unitree.com/WPAPER/wpaper.htm.
[23] F. McClain, "DataTree and UniTree: Software for File and Storage Management," Proc. IEEE Symp. Mass Storage Systems, May 1990.
[24] D. Ford and J. Myllymaki, "A Log-Structured Organization for Tertiary Storage," Proc. IEEE Int'l Conf. on Data Eng., 1996, pp. 20-27.
[25] J. Kohl et al., "HighLight: Using a Log-structured File System for Tertiary Storage Management," Proc. Winter USENIX, 1993, pp. 435-447.
[26] R. Wang and T. Anderson, "xFS: A Wide Area Mass Storage File System," Proc. Fourth Workshop on Workstation Operating Systems. Oct. 1993, pp. 71-78.
[27] A. Kraiss et al., "Tape-Disk Join Strategies under Disk Contention," Proc. 15th Int'l Conf. on Data Eng., Mar. 1999, pp. 552-559.
[28] S. More et al., "Efficiently Sequencing Tape-Resident Jobs," Proc. 18th ACM SIGMOD-SIGACT-SIGART Symp. on PODS, May 1999, pp. 33-40.
[29] O. Sandsta and R. Midtstraum, "Improving the Access Time Performance of Serpentine Tape Drives," Proc. 15th Int'l Conf. on Data Eng., Mar. 1999, pp. 542-551.
[30] B. Hiller et al., "Scheduling and Data Replication to Improve Tape Jukebox Performance," Proc. 15th Int'l Conf. on Data Eng., Mar. 1999, pp. 532-541.

김 영 성
1997년 중앙대학교 컴퓨터공학과 졸업(공학사). 1999년 중앙대학교 대학원 컴퓨터공학과 졸업(공학석사). 1999년 ~ 현재 중앙대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 클라이언트-서버 DBMS, 실시간 데이타베이스, DBMS엔진

강 현 철
1983년 서울대학교 컴퓨터공학과 졸업(공학사). 1985년 Univ. of Maryland at College park, Computer Science(M.S.). 1987년 Univ. of Maryland at College Park, Computer Science(Ph.D.). 1988년 ~ 현재 중앙대학교 컴퓨터공학과 교수. 관심분야는 클라이언트-서버 DBMS, 분산 데이타베이스, 멀티미디어 데이타베이스, 실시간 데이타베이스, 이동 데이타베이스

김 준
1983년 부산대학교 계산통계학과 졸업(학사). 1986년 한국과학기술원 전산학과 졸업(석사). 1986년 2월 ~ 현재 한국전자통신연구원 인터넷서비스연구부 선임연구원. 관심분야는 멀티미디어 DBMS, 저장 시스템, 회복 및 동시성 제어