DOI: 10.3745/KIPSTD.2008.15-D.6.733

모바일 DBMS를 위한 압축 데이터 관리 시스템의 기능 고도화

황 진 호[†] · 이 정 화[†] · 김 건 우[††] · 신 영 재[†††] · 손 진 현[††††]

요 약

최근 정보의 디지털화와 휴대용 정보기기의 보편화로 휴대용 정보기기가 업무처리의 중요한 수단이 되었고 처리해야하는 정보의 양 또한 증가하고 있다. 이러한 휴대용 정보기기에서 무수한 디지털 정보를 효과적으로 처리하기 위해 휴대용 정보기기에 적합한 모바일 DBMS의 사용이 요구된다. 또한 휴대용 정보기기에서 데이터 저장장치로서 보편적으로 사용되는 플래시 메모리는 기존의 하드디스크에 비해 저장 비용이 높아 데이터 저장에 있어 공간적 제약을 받으며, 데이터 I/O에 따른 수명을 가짐으로서 사용성에 제약을 받는다. 이러한 이유로 플래시 메모리를 저장매체로 사용하는 DBMS가 보편적으로 사용할 수 있는 압축 데이터 관리 시스템이 연구되고 있다. 하지만 압축 데이터 관리 시스템의 연구는 초기 단계이며 몇 가지 문제점을 가지고 있다. 따라서 본 논문에서는 압축 데이터 관리 시스템의 문제점을 해결하고, 추가적인 관리방식을 제안하여 새로운 압축 데이터 관리 시스템을 제안한다.

키워드 : 휴대용 데이터베이스 시스템, 데이터베이스, 데이터 압축, 플래시메모리, 압축 데이터 관리 시스템

Functional Improvement of the Compressed Data Management System for Mobile DBMS

Jin-Ho Hwang[†] · Jeong Wha Lee[†] · Gun-Woo Kim[††] · Youngjae Shin[†††] · Jin Hyun Son[††††]

ABSTRACT

Recently, mobile computing devices are used popularly. And quantity of information on mobile computing devices is being increased due to digitalization of information. So it needs an embedded DBMS for effective information management. Furthermore, since flash memory having a restriction on the number of partial write cycles is rapidly deployed on mobile computing devices as data storage and is more expensive than the conventional magnetic hard disk, the compressed data management system(CDMS) has been considered as an effective storage management technique for mobile computing devices in previous research. However, the research of CDMS is at the initial stage and has several problems. Hence, in this paper, we present additional storage management methods to solve the problems and improve the effectiveness of the CDMS for embedded DBMS.

Keywords : Mobile DBMS, Database, Data Compression, Flash Memory, Compressed Data Management System

1. 서 론

최근 무선통신의 발달과 휴대용 정보기기의 발달로 업무처리에 있어 휴대용 정보기기의 활용이 급격히 늘어나고 있으며 이러한 정보기기에서 데이터의 접근성을 높이고 보다 많은 정보들을 효율적으로 관리, 처리하기위해 내장형 DBMS (Database Management System)의 채용이 증가하고 있다.

또한, 높은 휴대성과 내구성, 저 전력소모를 요구하는 휴대용 정보기기에서 부피가 작고 경량으로 소음과 진동이 없으며 물리적인 충격에 강한 플래시 메모리는 보편적인 저장장치로 사용된다[1, 4, 7, 13]. 이 플래시 메모리는 기존의 하드디스크에 비해 저장 비용이 높고 데이터 I/O 횟수에 따른 수명을 가진다는 단점으로 인해 효율적인 데이터 관리가 필요하다[3, 8, 9, 10].

이러한 필요로 기존의 하드디스크를 저장장치로 사용하는 DBMS를 위한 압축 데이터 관리 시스템에 대한 연구에 이어 현재 휴대용 정보기기에서 저장장치로 사용되고 있는 NAND형 플래시 메모리를 사용하는 내장형 DBMS를 위해

※ 이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2007-313-D00757). 본 연구(논문)는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다.
† 준 회 원 : 한양대학교 컴퓨터공학과 석사과정
†† 준 회 원 : 한양대학교 컴퓨터공학과 박사과정
††† 정 회 원 : 삼성전자 통신연구소
†††† 종신회원 : 한양대학교 컴퓨터공학과 부교수
논문접수 : 2007년 12월 27일
수 정 일 : 1차 2008년 3월 12일
심사완료 : 2008년 4월 10일

데이터를 효율적으로 저장하는 방법으로 압축 데이터 관리 시스템에 대한 연구가 진행되었다[2, 5, 6, 11, 12, 14, 15]. 하지만 이 연구는 아직까지 초기단계로써 추가적인 연구가 진행되어야 한다.

이전 플래시 메모리 기반 데이터 압축 관리 시스템에 관한 연구 중 내장형 DBMS를 위한CDMS(Compressed Data Management System)라는 압축 데이터 관리 시스템이 제안되었다[2]. 이러한 CDMS는 몇 가지 문제점을 가지고 있다. 본 논문에서는 CDMS가 가지는 압축데이터를 관리하기위해 필요한 관리정보의 양이 관리정책에 따라 많은 메모리의 낭비를 가져올 수 있다는 문제와 모바일 컴퓨팅 환경에서 쉽게 야기될 수 있는 비정상 종료상황에서 데이터 손상 문제를 보완하고 해결하기위한 추가적인 관리방식을 제안한다. 또한, 실험을 통하여 새로운 관리방식을 적용한 시스템의 성능을 비교, 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 기존 연구인 압축 데이터 관리 시스템(CDMS)에 대해 살펴보고, 3장에서는 압축 데이터 관리 시스템의 문제점을 확인하고 개선방법을 제시한다. 4장에서 추가적인 압축 데이터 관리방식을 제안하고 5장에서는 각 관리방식을 실험을 통해 비교한다. 마지막으로 6장에서는 논문의 결론을 맺는다.

2. CDMS

이 장에서는 이 논문이 보완하고자 하는 압축 데이터 관리 시스템(CDMS, Compressed Data Management System)에 대해 기술한다. 먼저 CDMS의 전체적인 구조에 대해 2.1절에서 기술한다. 그리고 CDMS의 관리방식으로 압축된 데이터의 크기에 따라 처리하는 동적 기법과 가상의 슬롯을 이용하는 정적 기법을 2.2절에서 기술한다.

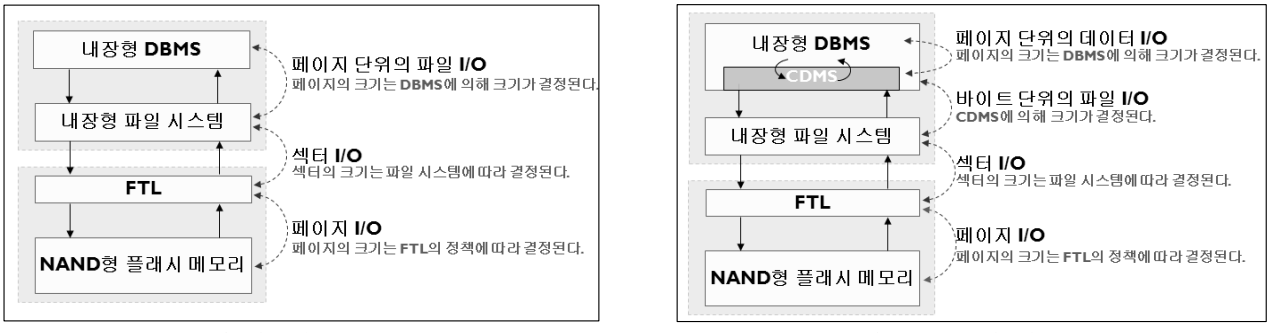
2.1 CDMS 구조

CDMS는 기존 DBMS의 변경 최소화를 위해 DBMS의 스토리지 관리자와 파일 시스템 사이에 위치하게 된다. DBMS는 기존에 파일시스템으로 보내던 데이터를 CDMS에게 보내게 되고 이것을 CDMS가 압축하여 파일시스템에 보내게 된다. (그림 1)은 기존 시스템과 CDMS의 데이터 교환을 나타낸 것이다[2].

(그림 1)의 (가)에서 보는 바와 같이 기존의 시스템은 모바일 DBMS에서 파일을 관리하기 위해 데이터를 읽고 쓰는 것에 대한 요청을 파일시스템에 하게 된다. 이때 모바일 DBMS와 파일시스템 두 계층 사이의 데이터 교환은 DBMS가 정의한 페이지 단위의 파일 I/O를 사용한다. 하지만 CDMS를 이용한 구조는 (그림 1)의 (나)와 같이 DBMS의 데이터 파일에 대한 접근 및 처리를 CDMS에 의해 이루어지도록 한다. 모바일 DBMS는 기존의 파일시스템에 요청한 것과 동일하게 페이지 단위의 데이터 I/O를 사용하여 CDMS에게 요청하면 CDMS는 추가적인 처리를 자체적으로 함으로 압축을 적용시켜 파일 시스템과 바이트 단위의 파일 I/O를 통해 데이터 교환을 하여 DBMS의 요청을 처리한다. 따라서 모바일 DBMS의 데이터 파일은 압축된 블록들로 구성되지만 DBMS는 압축되지 않은 데이터파일에 접근하는 방식과 동일하게 데이터 파일에 대한 접근요청을 CDMS에 할 수 있다.

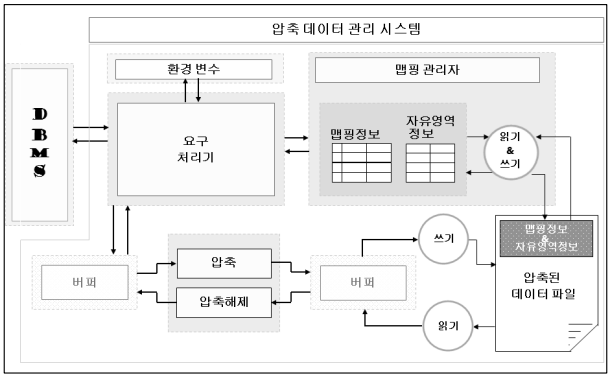
(그림 2)는 CDMS의 전체적인 구조도이다[2]. (그림 2)에 나타나 있는 바와 같이 CDMS는 DBMS가 요청한 데이터를 압축된 데이터 파일에서 찾기 위해 데이터파일 내부에 매핑정보를 포함시키고 있다. 이 매핑정보는 논리적 블록번호, 압축된 블록의 위치나 크기 정보를 가지게 되며, 이러한 매핑정보를 이용해 압축되지 않은 상태의 데이터파일과 압축된 데이터파일의 매핑이 가능해진다. 이때 매핑정보에 포함된 블록번호는 압축되지 않은 상태의 데이터파일을 논리적 블록의 크기(페이지)로 분할했을 때의 할당된 번호이다.

또한 CDMS는 데이터파일 내부에 자유영역정보를 포함시키고 있다. 이 자유영역정보는 다음과 같은 이유로 생기게 된다. 압축된 데이터에 해당하는 페이지가 DBMS에 의해 갱신되어 다시 쓰기를 수행할 때 CDMS에 의해 해당 데이터가 포함된 블록은 다시 압축되어 데이터 파일 내에 저장하게 된다. 이때 다시 압축된 블록은 갱신되기 전 압축된 상태의 블록크기와 동일한 크기를 가진다는 것을 보장 할 수 없다. 이러한 이유로 압축된 블록이 갱신되어 다시기록되어야 할 때는 기존 데이터가 새로운 데이터보다 작을 때는 기존위치가 아닌 파일 내 새로운 공간에 저장되어야 하고, 기존 데이터가 존재하던 공간은 다른 새로운 압축된 블록의 쓰기를 위해 저장 가능한 공간으로 표시되어야 한다. 또한 기존 데이터가 새로운 데이터보다 클 경우는 나머지



(가) 기존 시스템 구조 (나) CDMS를 이용한 구조

(그림 1) 플래시 메모리를 사용한 정보기기에서 데이터교환



(그림 2) CDMS의 구조

부분을 저장 가능한 공간으로 표시되어야 한다. 이렇게 발생된 재사용이 가능한 공간의 표시를 위해 자유영역정보에 빈 공간에 대한 정보를 저장하여 관리하게 된다.

이렇게 구성된 CDMS는 압축된 블록에 대한 쓰기를 진행할 때 자유영역정보를 이용해서 발견된 빈 공간에 압축된 블록을 저장하게 된다. 저장하는 블록은 매핑정보에 추가함으로 CDMS는 매핑정보와 자유영역정보를 활용해 DBMS가 요구하는 데이터에 대한 검색과 저장이 가능하다. 기본적인 DBMS의 데이터 읽기 쓰기 명령에 대한 CDMS의 처리절차는 <표 1>에서 나타난 바와 같다[2].

<표 1>에 나타나 있듯이 데이터읽기 요청 시에는 매핑정보를 이용해 위치를 찾아 읽어 와서 압축해제를 한 후 돌려주게 된다. 또 데이터쓰기 요청 시에는 데이터를 압축하여 쓸 수 있는 빈 공간을 찾아 압축된 데이터를 저장하고 그 정보를 매핑정보에 입력하게 된다. 그리고 그 후 자유영역정보를 갱신하게 된다. 자유영역정보 갱신은 사용된 빈 공간을 삭제하고, 이전 데이터를 바꾸는 작업이면 이전 데이터가 차지하고 있던 자리를 빈 공간으로 자유영역정보에 추가함으로 이루어진다.

2.2 압축데이터 관리기법

CDMS는 압축된 데이터들의 효율적인 관리를 위해 두 가지 관리기법, 동적기법과 정적기법을 제안하였다. 동적 기법은 압축된 데이터의 크기에 따라 동적으로 처리하는 방식이다. 이 방식은 CDMS가 데이터 쓰기의 요청을 받았을 때 그 데이터의 압축된 블록이 저장을 하고자 하는 공간은 무조건 블록 전체가 모두 들어갈 수 있는 크기의 공간이면서, 다른 공간보다 앞쪽의 공간이다. 이 조건에 맞는 빈 공간을 찾아 압축된 블록 전체를 쓰게 된다. 이러한 관리방식은 빈 공간 없이 데이터가 들어갈 수도 있지만 반대로 1바이트라도 모자라면 사용하지 못하게 된다. 따라서 압축된 블록이 들어가지 못해 계속적으로 사용하지 못하는 빈 공간이 생기게 된다. 이로 인해 공간의 낭비와 자유영역정보의 증가를 초래할 수 있다. CDMS의 또 하나의 관리방식은 정적 기법이다. 이 방식은 빈 공간의 증가를 막기 위한 방법으로 가상의 슬롯을 이용하고 있다. 이러한 가상의 슬롯을 이용함으로 데이터 파일을 논리적으로 고정된 슬롯 크기로 나누어 압축된 블록을 저장하게 된다. 즉, 파일의 내부를 같은 크기의 슬롯으로 논리적으로 분할하여 압축된 블록을 저장할 때 이 슬롯에 저장하는 방식을 사용한다. 이 방식을 사용함으로 슬롯 안에 내부적인 빈공간이 생기지만 데이터의 처리가 슬롯단위로 가능해져서 관리가 편리하면서 효율적이게 된다. 정적 기법은 저장할 때 가상의 슬롯 크기로 저장하게 된다. 따라서 갱신할 때 발생하는 빈 공간도 슬롯 크기를 가지게 된다. 이로 인해 정적 기법은 데이터의 입출력 및 빈 공간 관리를 슬롯 형태로 할 수 있게 된다. 하지만 이 기법은 슬롯 안에 사용하지 못하는 빈 공간이 존재하는 등의 단점이 존재한다.

3. CDMS 개선

CDMS는 압축된 데이터를 효율적으로 관리하기 위해 자유영역정보와 매핑정보가 추가적으로 필요하다. 하지만 이 추가적인 정보의 관리에 있어서, 문제가 발생할 소지가 있게 된다.

먼저 자유영역정보의 추가, 삭제가 발생하면서 자유영역정보를 담고 있는 테이블의 무분별한 증대를 가져오게 된다. 이러한 문제를 예방하기 위해 적절한 자유영역정보의 관리가 필요하게 된다.

자유영역정보의 관리를 위해 본 논문에서 제안하는 방법은 첫 번째로, 데이터의 마지막 정보(DATA_END)를 별도로 유지하는 것이다. 이 방법은 자유영역정보 테이블에 저장되어야 할 파일의 마지막 부분에 해당되는 정보를 저장하지 않고, 데이터의 마지막 정보(DATA_END)를 갱신함으로 테이블의 사용을 줄이게 된다. 이것은 CDMS가 갱신할 때, 앞쪽의 자유영역부터 채우는 것을 기본으로 하고 있기 때문에 일반적으로 마지막 부분이 자유영역으로 반환되기 때문에 좋은 효과를 거둘 수 있다. 그리고 본 논문에서 자유영역정보 관리를 위해 제안하는 두 번째 방법은 연속되는 자유영역을 검사하여 통합함으로 추가되는 자유영역이 기존에 있던 자유영역에 연속적인 경우 기존의 자유영역 정보를 갱신하는 방법으로 추가하게 되어 테이블의 사용을 줄이게 한다.

또한 CDMS의 추가정보 처리에는 비정상 종료로 인한

〈표 1〉 CDMS를 사용한 DBMS의 데이터읽기, 쓰기 요청처리 절차

DBMS의 데이터읽기 요청	DBMS의 데이터쓰기 요청
1) 요청된 데이터를 포함하는 압축된 블록의 위치 검색 2) 검색된 위치의 블록 읽기 3) 읽어진 블록의 압축해제 4) 압축해제 된 데이터를 DBMS에 반환	1) 요청된 데이터 압축 2) 저장 가능한 빈 공간 검색 3) 검색된 빈 공간에 압축된 블록 쓰기 4) 매핑정보 갱신 5) 자유영역정보 갱신

〈표 2〉 CDMS를 사용한 DBMS의 데이터쓰기 요청처리 절차

DBMS의 데이터쓰기 요청 처리 절차
1) DBMS의 쓰기 요청된 데이터 압축 2) 자유영역리스트의 첫 자유영역으로부터 압축된 블록 쓰기 3) 사용된 자유영역정보 삭제 4) 매핑정보 갱신 5) 자유영역정보 추가 (데이터 갱신 시에만 발생)

자유영역정보의 손상이 있을 수 있다. 이것은 심각할 경우 데이터 파일의 무결성을 보장하지 못할 수도 있다. 예를 들어 데이터 갱신 시 자유영역정보를 갱신(추가/삭제)한 후 매핑정보를 갱신하는 절차를 따른다면, 자유영역정보를 추가한 후 비정상 종료가 되었을 때 매핑정보에 존재하는 유효한 구역이 자유영역정보에도 동일하게 존재하게 된다. 이때 자유영역정보에 포함되어 있는 구역 정보로 데이터를 쓰고 문제가 발생되어 처리되지 않은 매핑 정보의 구역정보가 다시 자유영역정보가 되었을 때 먼저 쓰인 데이터를 잃게 된다. 따라서 데이터의 무결성을 보장하기 위해 본 논문에서는 자유영역정보의 갱신의 순서를 삭제 후 추가로 정하고, 매핑정보의 갱신을 자유영역정보 삭제 다음에 수행되는 절차를 제안한다. 즉 처리절차는 자유영역정보 삭제, 매핑정보 갱신, 자유영역정보 추가의 순서이다.

<표 2>는 이러한 관리를 반영한 CDMS의 데이터쓰기 요청처리 절차이다.

위에서 설명한 것과 같이 자유영역정보 중 처음 위치부터 데이터 쓰기가 수행되므로 기존방식의 CDMS와 달리 자유영역검색을 위한 시간 소모와 자유영역정보를 담고 있는 테이블이 무한대로 증가하는 문제를 해결할 수 있으며 비정상 종료에 대체하기 위해 자유영역과 매핑정보에 대한 처리 순서가 달라졌음을 보여주고 있다.

이러한 CDMS는 압축된 블록에 대한 쓰기를 진행할 때 자유영역정보를 이용해서 발견된 자유영역에 압축된 블록을 저장하게 된다. 저장하는 블록은 매핑정보에 추가함으로 CDMS는 매핑정보와 자유영역정보를 활용해 DBMS가 요구하는 데이터에 대한 검색과 저장이 가능하다. 또한 CDMS의 데이터쓰기 절차는 데이터 무결성을 보장한다. 다시 말해 어떠한 상황에서 비정상 종료를 하더라도 데이터파일의 손상과 완료되기 전의 데이터를 보호할 수 있다. 이것은 먼저 갱신이 요청된 데이터를 기존의 구역이 아닌 다른 구역에 저장하고 매핑정보와 자유영역정보의 갱신 절차를 통해 같은 구역이 각각의 정보에 중첩되지 않도록 함으로 가능하게 된다.

4. 압축 데이터 관리기법

이 장에서는 압축된 데이터들의 효율적인 관리를 위해 CDMS에 적용된 2가지 관리방식 외에 추가적으로 2가지 관리방식을 소개한다. 여기서 CDMS에서 소개되어진 관리방식을 정적 연결 관리방식, 동적 연결 관리방식이라 명한다. 그리고 본 논문에서 제안하는 관리방식을 정적 분리 관리방식, 동적 분리 관리방식이라 명한다. 먼저 정적 분리 관리방식을 4.1에서 기술하고 그 후 동적 분리 관리방식을 4.2에서 기술한다.

4.1 정적 분리 관리 방식

정적 분리 관리방식은 기존 CDMS에서 제안된 정적 연결 관리방식과 다르게 데이터 쓰기 요청을 받았을 때 앞쪽의 빈 공간을 차례로 데이터를 나눠서라도 채워가는 방식이다. 다시 말해 연속된 공간에 써야 할 데이터가 연속된 공간을 만난다면 그곳에 연속적으로 쓸 수 있지만 그렇지 않다면 먼저 빈공간의 크기만큼 쓰고 나머지는 다음 빈 공간에 쓰게 된다.

다음은 정적 분리 관리방식을 사용한 새로운 CDMS의 쓰기 절차이다.

```
CFM_Write(File Descriptor fp, Buffer buf, Data Size n)
1)  page_number = fp / size of a page   // calculate the page number for a requested page
2)  m_info = Map(page_number) // get the mapping information for a requested page
3)  compressed_data = CompressData(buf, n) // compresse the requested page data
4)  tmp_location =  first_location = CFMFreespace() // get the first free space
5)  while (¬isEmpty(compessed_data))
6)    if (size of compressed_data <= size of a slot)
7)        LF=0, LP= 0 // set link flag and link point to zero
8)       // save LF, LP and compressed_data on first_location
9)       CFMWrite(first_location, LF, LP, compressed_data)
10)      // remove the information of first location from free space table
11)      CFMFreeDel(first_location)
12)      compressed_data = null
13)   else
14)      second_location = CFMFreespace() // gets the next free space in a data file
15)      LF=1, LP=second_location
16)      CFMWrite(first_location, LF, LP, CFMsplit(compressed_data, as size of a slot))
17)      CFMFreeDel(first_location)
18)      first_location = second_location
19)      compressed_data =- CFMSplit(compressed_data, as size of a slot)
20)   end if
21) end while
22) // update mapping information
23) CFMMapping(page_number, tmp_location, size of orignal compressed data)
24) // update free space information
25) if (¬isEmpty(m_infor))
26)   do while (¬(m_infor.slot_number->LF) =0)
27)      CFMFreeAdd(m_infor.slot_number)
28)      m_infor.slot_number = m_infor.slot_number->LP
29)   end while
30) end if
```

이러한 정적 분리 관리방식은 정적 연결 관리방식처럼 압축된 데이터의 크기가 일정한 경우 사용하지 않는 부분이 매우 적어지고, 관리에 있어서 논리적 슬롯방식을 사용하기 때문에 슬롯 내에 사용되지 않은 부분에 대한 관리를 무시하는 등 세밀한 관리가 필요하지 않아 관리 및 처리가 간편하다. 또한 이 방식은 크기에 상관없이 앞쪽에 있는 빈 공간부터 채워나감으로써 공간효율성이 정적 연결 관리방식보다 높아지는 장점이 있다.

4.2 동적 분리 관리방식

동적 분리 관리방식은 압축된 데이터들에 맞는 공간만을 찾다가 빈 공간이 많이 생기는 것을 제거하고자 압축된 데이터를 나눠서 앞쪽의 빈 공간부터 채워 가는 방식이다. 연속된 공간에 써야 할 데이터가 처음에 연속된 공간을 만난다면 그곳에 연속적으로 쓸 수 있지만 그렇지 않다면 먼저 빈공간의 크기만큼 쓰고 나머지는 다음 빈 공간에 쓰게 된다. 이 때 나눠지는 데이터는 나눠진 블록에 대한 정보를 자체적으로 보유하게 된다. 예를 들어 압축된 데이터가 3개의 블록으로 나눠졌다면, 매핑정보에는 첫 블록의 시작위치와 전체 블록의 크기정보를 가지고 있다. 첫 블록은 다음 블록의 시작위치 정보와 다음의 블록 크기를 가지고 있다. 두 번째 블록 또한 다음 블록의 정보를 가지고 있고, 마지막 블록인 세 번째 블록은 압축된 데이터의 전체 크기를 가지고 있게 된다.

다음은 동적 분리 관리방식을 사용한 새로운 CDMS의 쓰기 절차이다.

```
CDMS_Write(File Descriptor fp, Buffer buf, Data Size n)
1)  page_number = fp / size of a page
2)  m_info = Map(page_number)
3)  compressed_data = CompressData(buf, n)
5)  // gets the first free space in a data file with its size
4)  temp_location = first_location = CFMFreespace()
5)  while (¬isEmpty(compessed_data))
6)     if (size of compressed_data <= first_location.size)
7)        // set next block base and  next block size
8)        NB=0, NS= size of compressed_data
9)        CFMWrite(NB, NS, compressed_data, first_location.
base)
10)       CFMFreeDel(first_location)
11)       compressed_data = null
12)       // add remained free space of first_location into free
space table
13)       temp_size = size of compressed_data
14)       CFMFreeAdd(first_location+temp_size, first_location.size
- temp_size)
15)    else
16)          second_location = CFMFreespace() // gets the
next free space in a data file
17)       NB=second_location.base, NS = size of compressed_data
- first_location.size
18)       CFMWrite(NB, NS, CFMSplit(compressed_data,
first_location.size), first_location)
19)       CFMFreeDel(first_location)
20)       first_location = second_location
21)       compressed_data =- CFMSplit(compressed_data, first_
location.size)
22)    end if
23)  end while
24)  // update mapping information
25)  CFMMapping(page_number, temp_location, size of orignal
compressed data)
26)  // update free space information
27)  if (¬isEmpty(m_infor))
28)    do while (¬(m_infor.base->NB) =0)
29)       CFMFreeAdd(m_infor.base, m_infor.size)
30)       m_infor.base = m_infor.base->NB
31)    end while
32)  end if
```

이러한 동적 분리 관리방식을 사용함으로써 파일의 빈 공간을 앞에서부터 차례로 채워 나가기 때문에 빈 공간을 획기적으로 줄일 수 있다. 따라서 파일 용량의 효율적인 관리가 가능함으로 공간효율성이 높아지는 장점이 있다. 하지만 너무 잘게 세분화 될 수 있는 단점이 있다. 동적 연결 관리방식에서는 빈 공간이 1바이트라도 모자라면 사용하지 못해 낭비되고, 동적 분리 관리방식은 너무 잘게 세분화 되어 여러 부분을 읽어야 되어서 성능이 떨어지게 된다.

5. 성능 평가

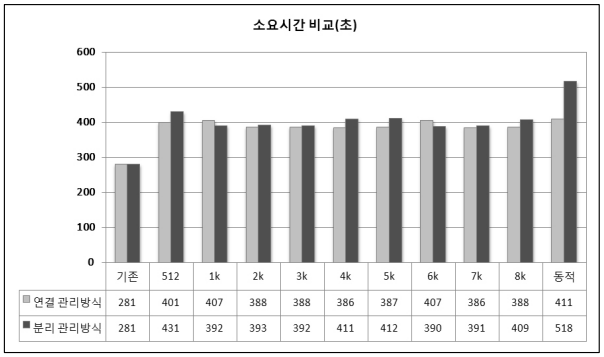
이 장에서는 본 논문이 제안하는 새로운 CDMS의 타당성을 검증한 실험결과 및 분석결과를 제시한다. 이 실험을 위하여 이엠웨어(주)의 MobileLite-LINUX 환경을 사용하였다. 실험을 통해 기존 DBMS와 CDMS를 사용한 DBMS의 차이를 확인하고 나아가 모바일 환경인 플래시 메모리상에서의 결과를 예측 할 수 있도록 하였다. 이 실험은 실제로 상용화 되어 사용되고 있는 모바일 DBMS인 이엠웨어(주)의 MobileLite와 DBMS를 검증하기 위한 테스트 프로그램을 사용하였다. 이 테스트 프로그램은 DBMS에게 많은 질의문을 넣어 실행하면서 그 결과를 검증하는 프로그램이다. 5.1절에서는 실험의 결과와 결과 분석을 기술하고, 5.2절에서는 설계방식과 실험을 바탕으로 각 관리방식 특징을 정리하여 기술한다.

5.1 실험 결과 및 분석

실험은 실제로 운영되고 있는 DBMS인 이엠웨어(주)의 MobileLite에 CDMS를 추가하여 MobileLite Test 프로그램을 실행시켜 도출한 결과이다. 비교 대상은 각 관리방식을 적용한 CDMS가 적용된 시스템과 기존 시스템이다. 여기서 각 관리방식은 정적 연결 관리방식, 정적 분리 관리방식, 동적 연결 관리방식, 동적 분리 관리방식이고 정적기법은 추가적으로 슬롯크기를 512bytes, 1Kbytes, 2Kbytes, 3Kbytes, 4Kbytes, 5Kbytes, 6Kbytes, 7Kbytes, 8Kbytes로 나누어 각각 테스트 한다.

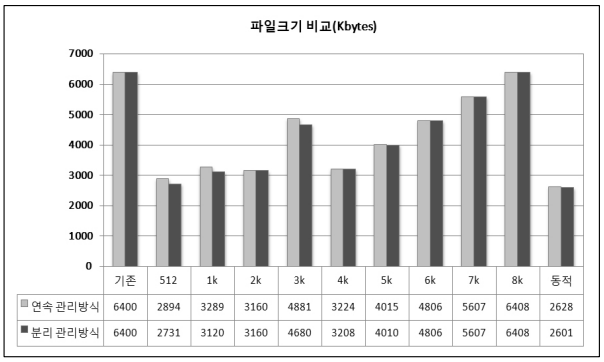
(그림 3)은 각 관리방식마다 Test 프로그램이 완료되는 시간을 측정한 실험이다. 이 실험을 통해 CDMS를 사용한 시스템이 평균적으로 속도 면에서 기존 시스템보다 40%의 추가 시간이 발생하는 것을 알 수 있다. 또한 동적 기법이 정적 기법보다 일반적으로 많은 시간이 소요되는 것을 알 수 있었다. 이것은 CDMS를 사용함으로 데이터에 대한 압축처리 시간과 압축된 데이터의 관리를 위한 관리정보 처리시간에 의해 속도 면에서의 성능 저하가 있음을 보여주고 있다. 또한, 정적 기법은 슬롯방식을 사용함으로 인해 압축된 데이터의 저장과 검색, 관리가 편리하고 간편하기 때문에 동적 기법보다 성능이 향상 되는 것을 보여주는 결과이다.

아래 (그림 4)는 각 관리방식마다 Test 프로그램이 완료된 시점에서의 데이터를 저장하고 있는 파일 크기를 측정한 것이다. 이 실험을 통해 기존방식보다 40%정도 적은 공간을 사용하여 CDMS의 사용이 우수함을 알 수 있다. 또한 3k에서 정적 기법은 다른 비슷한 슬롯 크기보다 나쁜 성능을 보



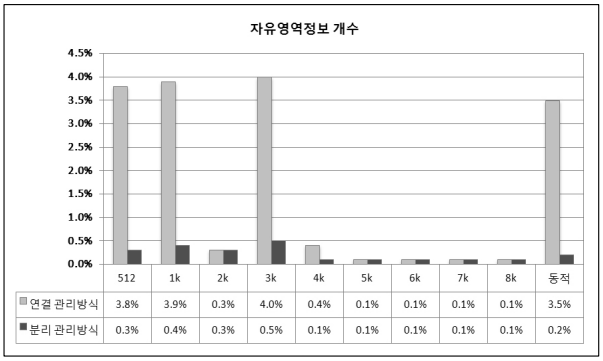
	기존	512	1k	2k	3k	4k	5k	6k	7k	8k	동적
연결 관리방식	281	401	407	388	388	386	387	407	386	388	411
분리 관리방식	281	431	392	393	392	411	412	390	391	409	518

(그림 3) 기존방식과 제안된 CDMS방식의 Test 프로그램 소요 시간 비교



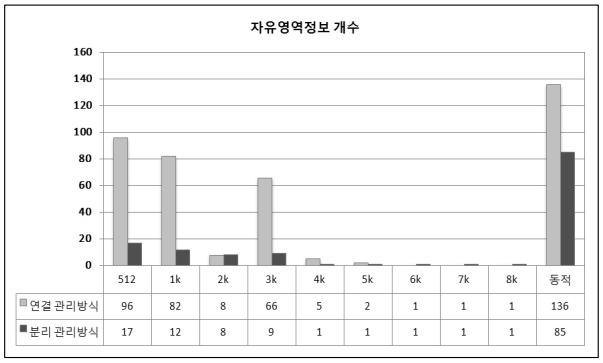
	기존	512	1k	2k	3k	4k	5k	6k	7k	8k	동적
연속 관리방식	6400	2894	3289	3160	4881	3224	4015	4806	5607	6408	2628
분리 관리방식	6400	2731	3120	3160	4680	3208	4010	4806	5607	6408	2601

(그림 4) Test 프로그램 실행 후 기존방식과 제안된 CDMS 방식의 파일크기 비교



	512	1k	2k	3k	4k	5k	6k	7k	8k	동적
연결 관리방식	3.8%	3.9%	0.3%	4.0%	0.4%	0.1%	0.1%	0.1%	0.1%	3.5%
분리 관리방식	0.3%	0.4%	0.3%	0.5%	0.1%	0.1%	0.1%	0.1%	0.1%	0.2%

(그림 5) Test 프로그램 실행 후 생성된 CDMS 관리 파일의 빈 공간 크기 비교



	512	1k	2k	3k	4k	5k	6k	7k	8k	동적
연결 관리방식	96	82	8	66	5	2	1	1	1	136
분리 관리방식	17	12	8	9	1	1	1	1	1	85

(그림 6) Test 프로그램 실행 후 자유영역정보에 빈 공간 정보의 개수

이는데, 이것은 정적 기법에서 잘못된 가상 슬롯 크기 선택이 나쁜 성능을 보일 수 있음을 보여준다. 본 실험에서는 DBMS의 페이지 쓰기 단위는 8Kbytes로 설정하였다. 이는 압축된 페이지들이 평균적으로 3Kbyte가 약간 넘는 크기로 압축되어 3Kbyte크기의 슬롯을 사용한 경우 슬롯의 낭비가 심함을 보여주는 것이다. 경우에 따라서 정적 기법에서는 슬롯 크기 선정이 성능에 중요한 영향을 미칠 수 있음을 확인할 수 있다.

다음으로 (그림 5)는 각 관리방식마다 Test 프로그램이 완료된 시점에서의 데이터를 저장하고 있는 파일 내부에 차지하고 있는 빈 공간을 측정한 것이다. 즉 자유영역정보에 저장되어 있는 빈공간의 합이다. 이 실험을 통해 일반적으로 동적 연결 관리방식이 가장 관리가 나쁜 것을 알 수 있다. 반면에 슬롯을 차지하는 정적 기법은 슬롯에 크기에 따라 다르기는 하지만 일반적으로 관리가 잘 이뤄지는 것을 알 수 있다. 하지만 앞서 보인 (그림 4)에서 빈 공간을 많이 가지는 동적 연결 관리방식보다 정적 기법들의 파일크기가 큰 것을 볼 수 있다. 이것은 정적 기법이 슬롯 안에 자유영역정보에 등록되지 않는 빈 공간을 내포하기 때문이다.

앞서 슬롯을 사용함으로 인해 자유영역정보를 효율적으로 관리할 수 있는 장점이 있을 것을 예상했다. 그것을 확인하기 위하여 테스트 프로그램을 실행 시킨 후의 자유영역정보에 빈 공간 정보 개수를 확인하여 보았다. (그림 6)은 자유영역정보에 저장되어 있는 빈 공간 정보의 개수를 보인 것이다.

(그림 6)을 통하여 정적 기법이 자유영역정보에서 빈 공간의 정보가 적은 것을 알 수 있다. 또 (그림 5)와 (그림 6)을 통해 정적기법의 실험항목 중 6k, 7k, 8k에 해당하는 빈 공간 정보의 개수가 같고 빈 공간이 차지하는 파일내부의 공간비율이 비슷함을 확인할 수 있다. 이는 압축된 데이터가 모두 6Kbyte 이하의 크기로 압축되었기에 6Kbyte 이상의 슬롯크기를 사용한 정적기법에서 동일한 슬롯의 개수가 사용되어 위와 같이 동일한 결과를 낳은 것이다. 이 실험을 통해 정적 기법이 동적 기법에 비해 빈 공간 정보가 효율적으로 관리되는 것을 알 수 있다. 하지만 슬롯의 크기가 작아지면서 완충 작용을 할 수 있는 범위가 줄어들어 빈 공간의 정보가 늘어나는 것을 볼 수 있다.

위의 일련의 실험을 통해 본 논문에서 제안하는 새로운 CDMS을 검증하고 각 관리방식의 특성을 실험을 통해 알아보았다. CDMS를 사용함으로 인해 평균 40%의 공간 절약을 가져오는 것을 확인할 수 있었고 슬롯을 사용함으로 인해 빈 공간의 관리가 효율적으로 이뤄지는 것과 잘못된 슬롯 크기의 선택으로 기존보다 더 나쁜 성능을 보일 수 있다는 것을 확인하였다.

5.2 분석 결과

앞에서 설명한 동적 기법과 정적 기법은 슬롯의 사용유무로 인해 차이가 발생한다. 정적기법은 슬롯을 사용하기 때문에 관리가 쉽고 간단하다. 따라서 데이터의 종류에 따라

압축률의 차이로 크기가 달라지는데, 이때 정적기법은 달라진 크기로 연산하는 것이 아니라, 슬롯으로 연산하게 되어 다른 크기를 가지는 데이터들을 쉽고 간단하게 처리할 수 있다. 하지만 슬롯의 사용으로 인해 슬롯 내부에 빈 공간이 발생한다. 또한 적절한 슬롯의 크기를 설정해야 한다. 반면에 동적 기법은 슬롯을 사용하지 않음으로 슬롯 내부에 발생하는 빈 공간이 없어지므로 크기를 획기적으로 줄일 수 있다. 하지만 동적기법은 각각의 블록에 맞게 바이트 단위로 처리되기 때문에 1바이트의 차이도 허용되지 않아 최악의 경우 갱신되는 데이터가 모두 DATA_END 뒤쪽으로 쓰이게 되어 갱신되기 전 데이터가 저장되어 있는 곳은 모두 빈공간이 되어 데이터 크기 효율이 많이 떨어지게 된다. 각 기법들은 이러한 정적 기법과 동적 기법의 기본 차이를 가지면서, 내부적으로 연결과 분리 2가지 방식을 가지고 있다. 이러한 각 방식들도 각각의 장단점을 가지고 있다. 먼저 연결 방식은 데이터를 연속된 공간에 저장하여서 단일 read 시스템콜로 데이터를 읽어올 수 있게 한다. 반면에 분리 방식은 앞쪽의 빈 공간부터 모두 채워나감으로써 공간효율성이 높다는 장점이 있다. 하지만 나눠진 데이터끼리는 링크 정보를 가지고 있어야 된다는 단점을 가지고 있다.

<표 3>은 위에 설명한 각 관리방식의 장단점을 표로 나타낸 것이다.

<표 3>를 통해서 확실하게 확인할 수 있는 것은 동적 기법의 장점은 정적 기법의 단점이 되고, 정적 기법의 장점은 동적 기법의 단점이 되는 사실이다. 이것은 어떤 특정 관리방식이 다른 관리방식보다 항상 우수하지 않고 사용되는 환경에 따라 우수한 관리방식이 다를 수 있다는 것이다. 그 환경은 동적기법일 경우 빈 공간이 발생하지 않는 상황인 갱신이 잘 일어나지 않고, 한번 쓰기를 실행한 후 읽기만 계속적으로 이용하는 DBMS의 경우나 갱신이 일어나도 똑같은 크기의 압축된 블록이 발생 되는 경우가 될 것이다. 반면에 정적기법은 그 반대의 경우인 갱신이 빈번하게 일어나면서 압축된 블록의 크기가 차이가 있는 경우에 적합한 관리방식이 될 것이다. 다른 크기의 압축된 블록으로 갱신이 빈번하게 일어난다면 빈 공간이 계속적으로 발생하게 될 것이기 때문에 이 환경에서는 슬롯을 사용하여 빈 공간의 관리가 효율적인 정적기법이 권장될 것이다.

6. 결 론

플래시 메모리가 본질적으로 가지는 여러 장점들로 하드디스크를 대체하여 다양한 휴대용 정보기기에서 저장장치로 사용되어지고 있다. 하지만 비용과 크기의 문제로 인해 저장공간을 늘리기에는 한계가 있으며 데이터 I/O 횟수에 따른 수명을 가진다는 단점이 있다. 이러한 단점을 보완하기 위한 방법으로 플래시 메모리를 위한 압축을 이용한 DBMS의 효율적인 데이터 관리가 연구되고 있다. 본 논문에서 이러한 연구의 연장선상으로 데이터 무결성을 지키면서 관리할 수 있는 방법을 제안하였고, 추가적인 관리방식을 제안하였다. 또한 각 관리방식을 비교 분석하여서 적용 환경의 특성에 따라 더 효율적인 관리를 할 수 있도록 기술하였다. 이러한 압축데이터 관리 시스템을 사용함으로 성능 평가를 통해서 검증 하였듯이 데이터베이스가 압축되지 않은 상태로 저장되는 현재 상황을 고려해 볼 때 획기적인 저장 공간의 절약을 가져온다. 또한 저장매체에 써야 되는 데이터를 줄임으로 데이터 I/O의 횟수를 줄여준다. 이것은 제한적인 쓰기 횟수를 가지는 플래시 메모리의 수명을 연장시키는 효과가 있다. 추가적으로 압축을 통해 데이터를 변형하기 때문에 데이터베이스가 압축되지 않은 상태로 저장되어 쉽게 정보가 노출될 수 있는 문제도 해결한다. 또한 CDMS는 모바일 DBMS에만 국한되지 않고 일반적인 DBMS에도 사용될 수 있다. 또한 DBMS에만 국한되지 않고 저장매체로의 입출력이 필요한 다른 응용에도 사용할 수 있다.

정보가 계속적으로 늘어나고 있는 현재, 사용하기 편리하지만 비용과 자기디스크와는 다른 특성을 가지는 플래시 메모리의 등장으로 더 이상 정보를 특별한 관리 없이 저장할 수 없게 되었다. 이러한 문제를 해결하면서 정보의 저장을

〈표 3〉 새로운 CDMS의 각 관리방식 장단점 비교

관리방식	장점	단점
정적 연결	■ 압축된 data의 크기가 비슷한 경우 매우 효과적이다 (vs. 동적) ■ 슬롯을 사용하기 때문에 관리가 잘 됨 (vs. 동적) ■ 연속된 공간에 저장 할 수 있다 (단일 read operation으로 read가능) (vs. 분리)	■ 관리자의 환경설정에 의존적 (Slot Size) (vs. 동적) ■ 내부 단편화 발생 (vs. 동적) ■ 특정 사이즈 이하의 Free Space가 계속해서 사용되지 않고 남을 수 있다 (vs. Minimizing)
정적 분리	■ 압축된 data의 크기가 비슷한 경우매우 효과적이다 (vs. 동적) ■ 슬롯을 사용하기 때문에 관리가 잘 됨. (vs. 동적) ■ 앞쪽 빈 공간부터 모두 채워나감으로써 공간효율성이 높다 (vs. 연결)	■ 관리자의 환경설정에 의존적 (Slot Size) (vs. 동적) ■ 내부 단편화 발생 (vs. 동적) ■ 슬롯 크기보다 큰 data는 분할 저장됨 (분할된 Data는 분리된 read/write필요) (vs. 연결) ■ 추가 링크정보 필요(vs. 연결)
동적 연결	■ 연속된 공간에 저장 할 수 있다 (단일 read operation으로 read가능) (vs. 분리) ■ 내부 단편화가 없다 (vs. 정적) ■ 관리자의 환경설정에 의존적 (SLOT_SIZE 환경변수에 영향 없음) (vs. 정적)	■ 압축된 Data의 미묘한 크기 차이로 균일하지 않는 Free Space가 발생하여 관리가 어렵다 (vs. 정적) ■ 특정 사이즈 이하의 Free Space가 계속해서 사용되지 않고 남을 수 있다 (vs. 분리)
동적 분리	■ 앞쪽 빈 공간부터 모두 채워나감으로써 공간효율성이 높다 (vs. 연결) ■ 내부 단편화가 없다 (vs. 정적) ■ 관리자의 환경설정에 independent (SLOT_SIZE 환경변수에 영향 없음) (vs. 정적)	■ 압축된 Data의 미묘한 크기 차이로 균일하지 않는 Free Space가 발생하여 관리가 어렵다. (vs. 정적) ■ 하나의 데이터가 분할 저장될 수 있다 (분할된 Data는 분리된 read/write필요) (vs. 연결) ■ 추가 링크정보 필요(vs. 연결)

효율적으로 관리하기 위해서는 계속적인 압축 데이터 관리 시스템에 대한 연구가 계속되어야 할 것이다. 본 논문에서 제기한 문제점 외에도 현재 압축 데이터 관리 시스템에서 고정시킨 매핑정보와 자유영역정보 크기의 유동적인 관리 등이 추가적인 연구가 필요한 부분이다.

참 고 문 헌

[1] 변시우, 노창배, 정명희, "휴대용 정보기기를 위한 플래시 기반 2단계 로킹 기법", 한국데이타베이스학회 : 정보기술과 데이타베이스 저널 제12권 제4호, pp.59-70, 2005.

[2] 신영재, 장진근, 이정화, 김학수, 박찬희, 손진현 "모바일 DBMS를 위한 효율적인 압축 데이터 동적 관리 시스템", 제27회 한국정보처리학회 춘계학술발표대회, 14권, 1호, pp.42-45, 경원대학교, 12 May, 2007.

[3] C.-H. Wu, L.-P. Chang, T.-W. Kuo, "An Efficient B-Tree Layer for Flash-Memory Storage Systems," Lecture notes in computer science, Vol.2968, pp.409-430, 2004.

[4] G.-J. Kim, S.-C. Back, H.-S. Lee, H.-D. Lee, M.-J. Joe, "LGeDBMS: A Small DBMS for Embedded System with Flash Memory," In Proceedings of the 32nd International Conference on VLDB, Seoul, Korea, pp.1255-1258, 2006.

[5] G. Graefe, L. Shapiro, "Data Compression and Database Performance," In ACM/IEEE-CS Symp. On Applied Computing, pp.22-27, 1991.

[6] Intel Corporation, "Understanding the Flash Translation Layer(FTL) Specification," APPLICATION NOTE, AP-684, 1998.

[7] J.-U. Kang, H.-S. Jo, J.-S. Kim, J.-W. Lee "A superblock-based flash translation layer for NAND flash memory," Proceedings of the 6th ACM International Conference on Embedded Software(EMSOFT), ACM, seoul, pp.161-170, 22-25 October, 2006.

[8] J. Kim, J.-M. Kim, S.-H. Noh, S.-L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for Compact Flash Systems," IEEE Transactions on Consumer Electronics, Vol. 48, No.2, pp.366-375, 2002.

[9] K.-S. Yim, H.-K. Bahn, K. Koh, "A Flash Compression Layer for SmartMedia Card Systems," IEEE Transactions on Consummer Electronics, Vol.50, No.1, pp.192-197, 2004.

[10] M.-L. Chiang, R.-C. Chang "Cleaning policies in mobile computers using flash memory," Journal of Systems and Software, Vol.48, No.3, pp.213-231, 1999.

[11] M. Kjelso, S. Jones, "Memory Management in Flash-Memory Disks with Data Compression," Lecture Notes in Computer Science, Vol. 986, Springer Verlag, pp. 399-413, 1995.

[12] S. Wells, D. Clay, "Flash Solid-State Drive with 6MB/s Read/Write Channel and Data Compression," In Proceedings of the 40th International Conference on Solid-State Circuits, pp.52-53, 1993.

[13] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee, H.-J. Song, "System Software for Flash Memory: A Survey," Lecture notes in computer science, Vol.4096, pp.394-404, 2006.

[14] Till Westmann, Donald Kossmann, Sven Helmer, Guido Moerkotte, "The implementation and performance of compressed databases," ACM SIGMOD Record, Vol.29 No.3, pp.55-67, 2000.

[15] W. Paul Cockshott, Douglas McGregor, and John Wilson. "High-performance operations using a compressed database architecture," The Computer Journal, Vol.41, No.5, p.283 - 296, 1998.

황 진 호

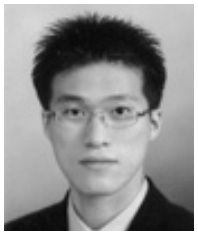
e-mail : jhhwang@database.hanyang.ac.kr
2006년 상지대학교 컴퓨터공학부(학사)
2006년~현 재 한양대학교 컴퓨터공학과 석사과정
관심분야 : 모바일 데이터베이스, 파일 시스템

이 정 화

e-mail : jwlee@database.hanyang.ac.kr
2007년 한양대학교 컴퓨터공학부(학사)
2007년~현 재 한양대학교 컴퓨터공학과 석사과정
관심분야 : 모바일 데이터베이스, 데이터베이스, 데이터 마이닝

김 건 우

e-mail : gwkim@database.hanyang.ac.kr
2006년 호주 뉴캐슬 대학교 컴퓨터공학과(학사)
2007년 호주 뉴캐슬 대학교 정보공학과(석사)
2008년~현 재 한양대학교 컴퓨터공학과 박사과정
관심분야 : 데이터베이스, E-Business, RFID, BPM, Database



신 영 재

e-mail : yj99.shin@samsung.ac.kr
2006년 한양대학교 전자컴퓨터공학부(학사)
2008년 한양대학교 컴퓨터공학과(석사)
2008년 2월~현재 삼성전자 통신연구소 연구원
관심분야 : 모바일 데이터베이스, 데이터 마이닝

손 진 현

e-mail : jhson@hanyang.ac.kr
1996년 서강대학교 전산학과(학사)
1998년 한국과학기술원 전산학과(석사)
2001년 한국과학기술원 전자전산학과(박사)
2001년 9월~2002년 8월 한국과학기술원 전자전산학과 박사후 연구원
2002년 9월~현재 한양대학교 컴퓨터공학과 부교수
관심분야 : 데이터베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템