DOI: 10.3745/KIPSTD.2008.15-D.5.589

모바일 DBMS를 위한 효율적인 압축 데이터 관리 시스템의 개발

신 영 재[†] · 황 진 호[††] · 김 학 수[†††] · 이 승 미[††††] · 손 진 현[†††††]

요 약

최근 휴대용 정보기기 사용이 보편화되어지고, 정보의 디지털화로 인해 휴대용 정보기기에서 처리되어야 하는 정보가 무수히 많아지고 있다. 이로 인해 휴대용 정보기기에서는 정보들을 효과적으로 관리하기 위해 모바일 DBMS의 사용이 요구되고 있다. 또한 휴대용 정보기기에서 보편적으로 사용되는 저장장치는 NAND형 플래시 메모리로 단위 공간당 비용이 기존의 하드디스크에 비해 수십 배 가량 높아 저장 공간의 효율적인 관리가 요구되고 있다. 따라서 본 논문에서는 플래시메모리를 저장매체로 사용하는 모바일 DBMS에서 압축 기법을 사용한 효율적인 데이터 관리 시스템을 제안한다. 제안되는 압축 기반 시스템은 저장 공간의 절약을 가져오고, 데이터 입출력을 줄인다. 이러한 이점은 플래시 메모리의 수명을 연장시키는 효과 또한 기대할 수 있다.

키워드 : 휴대용 데이터베이스 시스템, 데이터베이스, 데이터 압축, 플래시메모리, 플래시 변환계층

Development of the Efficient Compressed Data Management System for Embedded DBMS

Youngjae Shin[†] · Jin-Ho Hwang[††] · Hak Soo Kim[†††] · Seung Mi Lee[††††] · Jin Hyun Son[†††††]

ABSTRACT

Recently, Mobile Computing Devices are used generally. And Information which is processed by Mobile computing devices is increasing. Because information is digitalizing. So Mobile computing Devices demand an Embedded DBMS for efficient management of information. Moreover Mobile computing Devices demand an efficient storage management in NAND-type flash memory because the NAND-type flash memory is using generally in Mobile computing devices and the NAND-type flash memory is more expensive than the magnetic disks. So that in this paper, we present an efficient Compressed Data Management System for the embedded DBMS that is used in flash memory. This proposed system improve the space utilization and extend a lifetime of a flash memory because it decreases the size of data.

Keywords : Mobile DBMS, Database, Data Compression, Flash Memory, Flash Translation Layer

1. 서 론

PDA(개인 휴대 정보 단말기), HPC(핸드헬드 PC), PPC(포켓 PC), 개인용 휴대전화, 스마트폰 등 현대의 휴대용 정보기기는 기본적인 정보의 처리, 저장 수단으로만 사용하던 것에서 높은 휴대성과 광범위한 정보를 저장, 처리할 수 있는 기기로 바뀌고 있다[1]. 이러한 환경의 변화로 휴대용 정보기기에서는 보다 많은 정보의 생성, 처리, 저장이 가능한 다양한 응용프로그램의 사용을 요구 받고 있다[2]. 이로 인해 많은 데이터의 효율적인 관리 시스템이 필요로 하게 되었다. 그래서 휴대용 정보기기에서 정보를 쉽게 접근하여 처리하고 갱신할 수 있도록 구성된 데이터의 집합체인 데이터베이스의 사용이 필요하게 되었다. 세계적인 분석기관인 IDC(International Data Corporation)社는 2004년도에 64%이상의 휴대용 정보기기의 어플리케이션이 모바일 데이터베이스 관리 시스템이 필요하다고 보고하였다[3].

기존 컴퓨팅 환경에서는 가격이 저렴하고 확장성이 용이한 하드디스크를 데이터 저장장치로 사용함으로 저장 공간에 대한 비용이 많이 절감되었다. 하지만 높은 휴대성과 내구성, 저 전력소모를 요구하는 모바일 컴퓨팅 환경에서 전력을 많이 소모하며 크기, 소음, 진동 등의 단점을 가진 하드디스크의 사용이 어렵게 되었다. 따라서 휴대용 정보기기에서는 이러한 단점을 보완할 수 있는 저 전력으로 장시간

※ 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-20135-0)
본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업(IITA-2008-C1090-0801-0031)의 연구결과로 수행되었음.
† 정 회 원 : 삼성전자 통신연구소 연구원
†† 준 회 원 : 한양대학교 컴퓨터공학과 석사과정
††† 준 회 원 : 한양대학교 컴퓨터공학과 박사과정
†††† 정 회 원 : 한양대학교 컴퓨터공학과 BK21 사업팀 연구원
††††† 종신회원 : 한양대학교 컴퓨터공학과 부교수
논문접수 : 2006년 8월 22일
수 정 일 : 1차 2007년 4월 11일, 2차 2007년 9월 20일, 3차 2007년 11월 28일
심사완료 : 2007년 11월 30일

의 구동이 가능하며, 부피가 작고 경량으로 소음과 진동이 없으며 물리적인 충격에 강해 휴대가 용이한 플래시 메모리를 보조기억장치로 사용한다[4]. 이 플래시 메모리는 크게 바이트 단위로 I/O를 지원하는 NOR형 플래시 메모리와 페이지 단위의 I/O만을 지원하는 NAND형 플래시 메모리로 나뉜다. 휴대용 정보기기에서는 NOR형 플래시 메모리보다 비용이 저렴하여 대용량 데이터 저장장치로 사용되는 NAND형 플래시 메모리를 사용한다. 하지만 이 NAND형 플래시 메모리는 기존의 하드디스크에 비해 고비용과 데이터 I/O에 따른 수명을 가진다는 단점이 있다. 이로 인해 데이터의 효율적인 관리가 필요하다[1, 3].

본 논문에서는 현재 가장 보편적으로 휴대용 정보기기에서 저장장치로 사용되고 있는 NAND형 플래시 메모리에 데이터베이스의 데이터를 효율적으로 저장하는 방법으로 효율적인 데이터 압축 관리 시스템인 CDMS(Compressed Data Management System)를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터의 압축과 기존 압축데이터 관리기법, 그리고 플래시 메모리에 대한 관련연구를 살펴본다. 3장에서는 효율적인 압축 데이터 관리 시스템인 CDMS의 필요성에 대해 설명하며 4장에서는 이 필요성을 충족시키는 압축 데이터 관리 시스템, CDMS의 구조와 두 가지의 압축 데이터 관리기법을 설명하고, 각 관리기법을 비교해 본다. 5장에서는 제안하는 CDMS의 관리기법의 성능을 실제 DBMS와 연동하여 평가한다. 마지막으로 6장에서는 논문의 결론을 맺는다.

2. 관련 연구

이 장에서는 데이터베이스에서 데이터의 압축과 기존 압축데이터 관리기법, 그리고 플래시 메모리에 대한 관련연구를 살펴본다. 이것은 이 논문에서 제안하는 압축 데이터 관리 시스템(CDMS)에 대한 기본 지식 및 개발 동기가 된다. 먼저 2.1절에서는 데이터 압축 및 기존 압축데이터 관리기법에 관련해서 알아보고, 2.2절에서는 플래시 메모리의 특성에 대해서 알아본다.

2.1 데이터베이스에서 데이터 압축 및 관리

압축된 데이터를 사용하여 시스템의 효율을 높이고자 하는 연구는 현재까지 지속적으로 연구되어 왔다. 이와 관련된 연구는 크게 데이터 압축방식과 압축 데이터의 처리로 나눌 수 있다. 이 중 대부분의 연구는 데이터 압축방식에 대한 연구로써[5], 대표적으로 초기 Huffman coding을 이용한 방식과 Run-Length coding을 이용한 방식으로 나눌 수 있다. Huffman Coding과 Run-Length coding기법(Fixed-Length)은 엔트로피 기법으로서 압축시킬 대상의 특성을 고려하지 않고 데이터를 단순히 비트 집합 혹은 바이트 집합으로 보고 압축을 하게 된다. 이 기법들은 복원한 데이터가 압축전의 데이터와 완전히 일치하는 무손실(lossless) 기법이다[6, 7]. 이러한 데이터 압축기법이 데이터베이스 시스템의 성능에 미치는 효과에 대한 연구 또한 진행되었다. 이와 같은 연구에서 데이터 압축 기법을 DBMS에서 사용함으로써 일반적으로 다음과 같은 이점을 얻을 수 있음을 공통적으로 제시하고 있다. 첫째, 물리적 디스크공간의 절약이다. 데이터베이스의 데이터를 압축 저장함으로써 데이터의 양을 줄일 수 있고 이로 인해 물리적인 저장소의 디스크 공간을 절약할 수 있게 되는 것이다. 둘째로 디스크 I/O 감소이다. 데이터를 압축하게 되면 기존의 데이터가 차지하는 공간보다 작은 물리적 공간을 차지하게 된다. 이것은 같은 데이터를 읽고자 할 때 스토리지에서 기존보다 적은 부분만 읽어 오면 된다. 또한 같은 크기의 블록을 읽었을 때 압축된 상태의 데이터는 보다 많은 정보가 기록되어 있으므로 일정량의 데이터에 대한 읽기, 쓰기명령에 의해 일어나는 디스크 I/O의 횟수가 감소하게 된다. 이러한 이유로 DBMS는 I/O 병목현상 경감과 대역폭 증가의 효과를 얻을 수 있다[2, 8].

이러한 이점에도 불구하고 실제로 데이터 압축방식은 대부분의 고성능의 DBMS에서 전형적으로 사용되고 있지 않다. 사용되더라도 특정 DBMS에 특화된 방식으로 적용되어 사용되고 있다. 한 예로 상용 DBMS인 DB2에서 이와 같은 압축데이터를 활용하여 시스템의 성능향상과 저장 공간 절약의 효과를 보인 연구가 있었다. 이 연구에서 Ziv-Lempel 압축 알고리즘을 사용해서 테이블의 열 단위로 압축한다. 이때 빈번한 접근이 일어나면서 적은 데이터를 담고 있는 테이블은 압축하지 않고, 많은 데이터를 담고 있지만 적은 접근이 일어나는 테이블들은 선택적으로 압축하는 기법을 제시하였다[6].

또한 한 연구에서는 사전기법을 사용하여 압축하여 추후 질의 처리 시 압축을 풀지 않고 실행하는 방법과 압축된 데이터의 저장방식을 제시하였다[9, 10].

이러한 기존 연구들은 DBMS 내부에서 데이터 압축을 통해 시스템의 효율을 높이기 위한 연구로, 압축 알고리즘과 압축을 사용한 전체 DBMS를 개발 하는 것에 초점이 맞추어져 있었다. 또한 기본 연구들은 압축된 데이터의 저장방식에 대해 약간의 언급은 있었으나 그로인해 생겨나는 빈공간의 처리 등에 관한 최적의 관리방식을 보여주지 못했다.

따라서 본 논문에서는 기존 DBMS의 수정을 최소화 할 수 있도록 DBMS의 입출력단위인 페이지단위를 이용한 데이터 압축 구조를 설계함으로써 특정 DBMS에 비종속적 데이터 압축 구조를 제안하고, 효율적인 압축된 데이터의 관리기법을 제안하여 압축 효과를 증대시키도록 하였다.

2.2 플래시 메모리의 특성

플래시 메모리는 저 전력으로 장시간의 구동이 가능하며, 부피가 작고 가볍다. 또한 소음과 진동이 없으며, 물리적인 충격에 강해 휴대가 용이하다[4].

이러한 플래시 메모리는 설계의 차이로 크게 NOR형 방식과 NAND형 방식으로 나눠진다.

NOR형 플래시 메모리는 읽기 속도는 빠르지만 쓰기 속도가 느려 주로 프로그램 코드용 메모리로 사용된다. NAND형 플래시 메모리는 쓰기 속도가 빠르고 단위 공간당 단가가 낮아 주로 대용량 데이터 저장장치로 사용된다. <표 1>은

〈표 1〉 각종 저장 매체들의 특성비교

종류	읽기(단위)	쓰기(단위)	삭제(단위)	비용/MB
DRAM	60ns(2B) 2.56μs(512B)	60ns(2B) 2.56μs(512B)	-	30~40
NOR형 플래시	150ns(1B) 14.4μs	211μs(1B) 3.53ms(512B)	1.2s(128KB)	20~30
NAND형 플래시	10.2μs(1B) 35.9μs(512B)	201μs(1B) 226μs(512B)	2ms(16KB)	10~20
하드디스크	12.4ms(512B) (average)	12.4ms(512B) (average)	-	1

[Reference]
DRAM: 2-2-2 PC100 SDRAM.
하드디스크: Segate Barracuda ATA II.
NOR형 플래시: Intel 28F128J3A-150.
NAND형 플래시: Samsung K9F5608U0M.

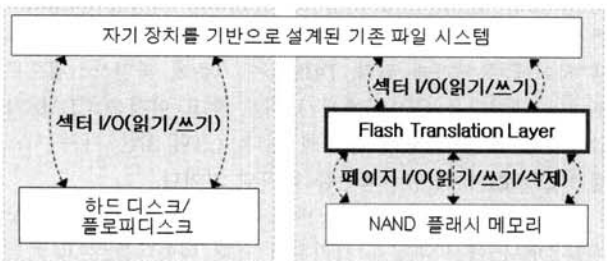
NOR형 플래시 메모리와 NAND형 플래시 메모리, 그리고 다른 저장 매체들의 특성을 수치적으로 비교한 것이다[11].

<표 1>에서는 플래시 메모리가 RAM에 비해 가격이 저렴하고, HDD보다 성능이 뛰어나는 것을 보여준다. 이러한 수치적 특성은 플래시 메모리가 비싸면서 대용량 저장매체로 사용하기 힘든 휘발성인 RAM과 여러 가지 단점으로 모바일 기기에 사용하기 힘든 하드디스크의 단점을 보안한 저장매체인 것을 보여주고 있다. 따라서 이러한 특성으로 인해 현재 모바일 기기들에 가장 보편적으로 NAND형 플래시 메모리가 사용되고 있다.

NAND형 플래시 메모리는 전력 소모가 작고 충격에 강하며 소형화가 가능하지만, 쓰기 위해서는 파일이 쓰일 위치에 내용이 모두 지워져야 된다.(erase-before-write) 또한 NAND형 플래시 메모리는 읽기/쓰기는 페이지 단위로 이루어지고, 삭제는 블록 단위로 이뤄진다. 다시 말해 쓰기/읽기 단위와 삭제 단위가 동일하지 않다[4,12,14]. 이때 삭제는 디스크의 수명을 단축시켜 약 10~100만 번을 수행하면 그 블록을 사용할 수 없게 된다. 이러한 수명에 대한 제약을 해결하기 위해서 I/O를 줄이는 것이 필요하게 된다[1, 3, 15].

또한 플래시 메모리는 위에 설명된 특성 때문에 기존의 파일시스템을 그대로 사용할 수 없다. 그래서 이런 문제를 해결하기 위해 FTL(Flash Translation Layer)을 중간에 위치시켜 플래시 메모리가 하드디스크처럼 여길 수 있도록 하는 역할을 한다. (그림 1)에 FTL의 동작을 보였다.

(그림 1)에서 나타난 바와 같이 파일시스템은 요청받은 파일의 데이터 처리를 위해 FTL에 해당 데이터에 대한 읽기, 쓰기 요청을 하게 되며, 파일시스템과 FTL사이의 데이터 교환 단위는 섹터단위가 된다. 이후 FTL에 의해 실제 데이터가 위치한 물리적 장치인 플래시 메모리에서는 페이지 단위의 읽기 쓰기가 일어난다. 이러한 처리에서 파일시스템은 해당 데이터가 저장된 논리적 섹터를 찾게 되고 해당 섹터에 대한 처리 요청을 FTL에 하게 된다. FTL은 파일시스템이 요청한 논리적 섹터의 데이터에 매칭 되는 물리적 페이지를 플래시 메모리에서 찾아 요청에 대한 처리를 하게 된다. 이때 FTL은 파일시스템으로부터 오는 읽기/쓰기 요청을 읽기/쓰기/삭제 작업을 통해 처리한다[11, 13, 14].



(그림 1) FTL의 동작

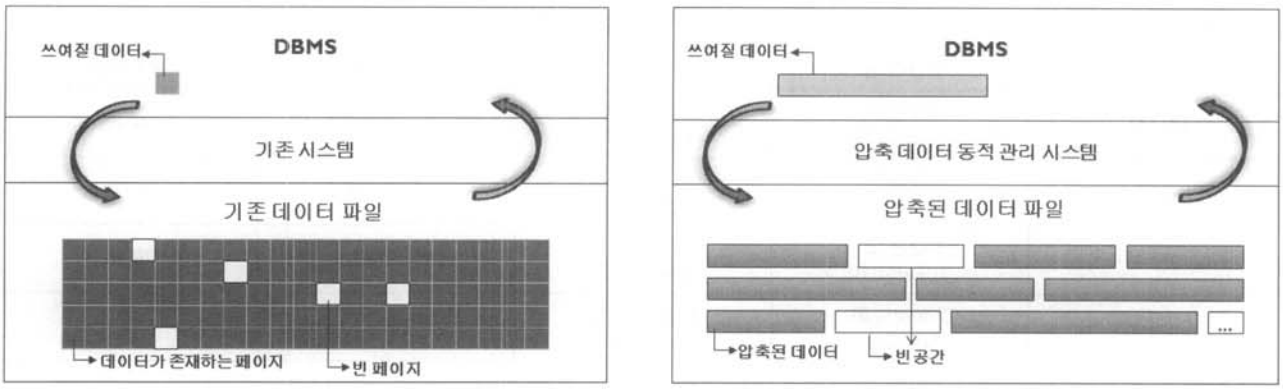
기존 연구 중 FTL 단에서 압축을 시도한 연구가 있었다[1]. 하지만 FTL에서 압축을 시도한다면, JPEG와 MP3같은 압축된 멀티미디어 파일이 들어오게 되면 압축효과는 없으면서도 압축을 시도하게 되어 이중으로 압축하는 문제가 발생한다. 요즘 휴대용 정보기기에서 멀티미디어 파일을 많이 처리/저장하고 있다는 사실에서 이러한 문제는 큰 영향을 준다.

이러한 플래시 메모리의 특성과 연구로 인해 플래시 메모리를 위해서 압축을 사용하는 것이 필요하다는 것을 알게 되었다. 또한 FTL단에서 처리하기에는 많은 문제점이 따르는 것도 알 수 있었다. 따라서 본 논문은 일반적으로 압축되지 않은 텍스트 정보가 저장되는 데이터베이스 파일만을 압축하는 CDMS를 제안하게 된 것이다. CDMS의 필요성은 다시 3장에서 구체적으로 설명한다.

3. 압축 데이터 관리 시스템(CDMS)의 필요성

위 플래시 메모리의 특성에서 살펴본 바와 같이 플래시 메모리는 많은 장점을 가지기 때문에 대부분의 휴대용 정보 기기에서는 보조기억장치로 기존의 하드디스크를 대신하여 플래시 메모리를 사용한다. 하지만 이 플래시 메모리는 자기저장장치인 하드디스크보다 비용이 많이 들고 데이터 I/O로 인해 수명이 제한된다.

따라서 모바일 환경에서 DBMS는 기존의 DBMS와 달리 저장 공간과 데이터 I/O에 따른 제한을 받게 된다. 이에 반해 모바일 환경에서 처리해야 할 정보의 수와 크기는 점차적으로 증가되고 있다. 이러한 이유로 인해 모바일 환경에서는 저장 공간의 효율적인 사용이 무엇보다 우선되어야 한다.



(가) 기존 경우 (나) 압축을 사용하였을 경우

(그림 2) DBMS에서 저장 공간으로 저장

구체적으로 말해, 모바일 환경에서 DBMS는 두 가지의 환경적인 문제를 가지게 된다. 첫째는 제한된 메모리 사용에 따른 저장 공간 부족이다. 휴대용 정보기기에서 처리 요구되는 정보의 양이 크게 증가되고 있는 상황에서 DBMS에 의해 관리되는 정보량 또한 크게 증가하고 있다. 하지만 모바일 환경은 비용에 의한 저장 공간의 제한으로 많은 양의 데이터의 저장, 처리를 지원하는데 어려움이 따른다. 둘째는 플래시 메모리 사용에 있어 데이터 I/O비용을 감소시켜야 한다. 플래시 메모리는 휴대용 저장장치에 적합한 여러 가지 장점을 가지고 있지만 느린 쓰기(기록) 속도와 데이터 I/O에 따른 수명을 가진다는 단점으로 인해 데이터 I/O에 대한 비용 절감을 요구한다. 느린 쓰기속도의 문제는 DBMS의 성능저하를 발생시키며 많은 데이터에 대한 I/O는 플래시 메모리의 수명과 직결되는 문제를 야기한다.

이 두 가지 문제를 해결하기 위한 방법으로 본 논문에서는 데이터 압축을 제안한다. 압축을 사용함으로 저장매체에 쓰거나 읽어야 할 데이터가 줄어들게 되어, 제한된 자원사용에 따른 저장 공간 부족문제와 플래시 메모리에서 많은 데이터 I/O에 의한 성능저하 및 플래시 메모리 수명문제에 대한 효율적인 대처가 가능하다. 또한 추가적으로 압축을 통해 데이터에 대한 암호화로 보안효과를 취할 수 있다. 하지만 압축을 사용함으로 추가적인 관리가 필요하게 된다. (그림 2)에서 보는 바와 같이 기존 경우(그림 2-가)는 정해진 규격(페이지)으로 관리되어지기 때문에 관리가 간편하고 저장 상태에 대한 많은 정보를 가지고 있을 필요가 없다. 하지만 압축을 사용하였을 경우(그림 2-나)는 압축한 원본 데이터의 크기가 각각 달라지어 관리가 복잡하고 저장 상태에 대한 추가적인 정보가 더 필요하게 된다. 따라서 본 논문에서는 이러한 압축된 데이터를 효율적으로 관리하는 관리기법 또한 제안하고 있다.

이러한 데이터 압축 관리 시스템은 일반적인 DBMS에서 파일시스템으로의 저장단위인 페이지(페이지)를 관리하는 방식으로 제안되었다. 따라서 제안되는 관리 시스템은 DBMS와 플래시 메모리의 파일시스템 사이에 위치하게 된다. 그 결과 DBMS의 변경 없이 기존의 DBMS에 쉽게 이용될 수 있는 특성을 가진다. 이것은 다음 장의 CDMS의 구조에서 다시 설명한다.

4. CDMS

이 장에서는 본 논문이 제안하는 압축 데이터 관리 시스템(CDMS, Compressed Data Management System)에 대해 기술한다. 먼저 CDMS의 전체적인 구조에 대해 4.1절에서 기술한다. 그리고 본 논문에서는 CDMS의 세부적인 관리기법으로 압축된 데이터의 크기에 따라 처리하는 동적 기법과 가상의 슬롯을 이용하는 정적 기법을 제안하는데, 이것을 4.2절에서 기술한다. 또한 제안하는 각 관리기법의 장단점 등을 비교한 내용을 4.3절에서 기술한다.

4.1 CDMS 구조

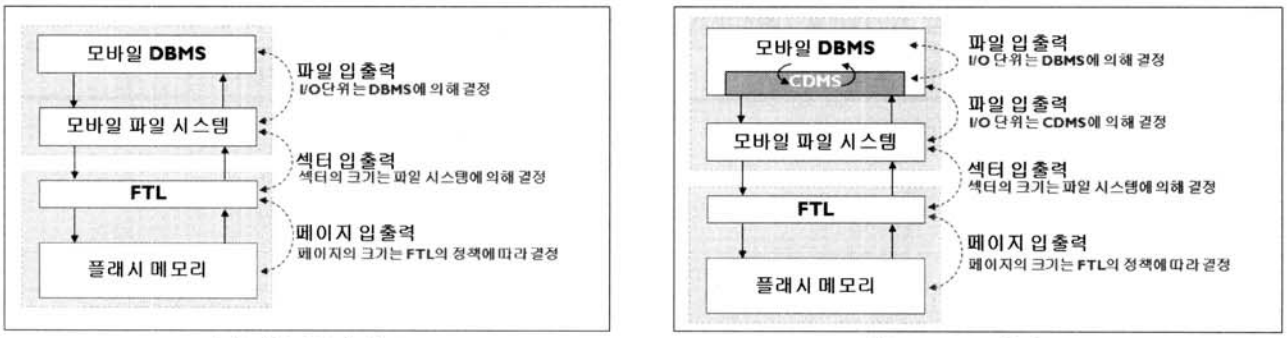
휴대용 정보기기에서 발생하는 저장 공간 부족문제는 데이터를 압축함으로 해결할 수 있다. 하지만 압축된 데이터는 크기가 달라 추가적인 관리가 필요하다. 본 논문에서는 이러한 요구를 만족시키기 위한 압축 데이터 관리 시스템, CDMS(Compressed Data Management System)를 사용한 새로운 구조를 제안한다.

다음은 본 연구에서 제안하는 CDMS의 설계 목표이다.

① DBMS의 데이터 파일이 차지하는 저장 공간의 절약
② 기존 DBMS의 변경 최소화
③ 효율적인 압축된 데이터 관리

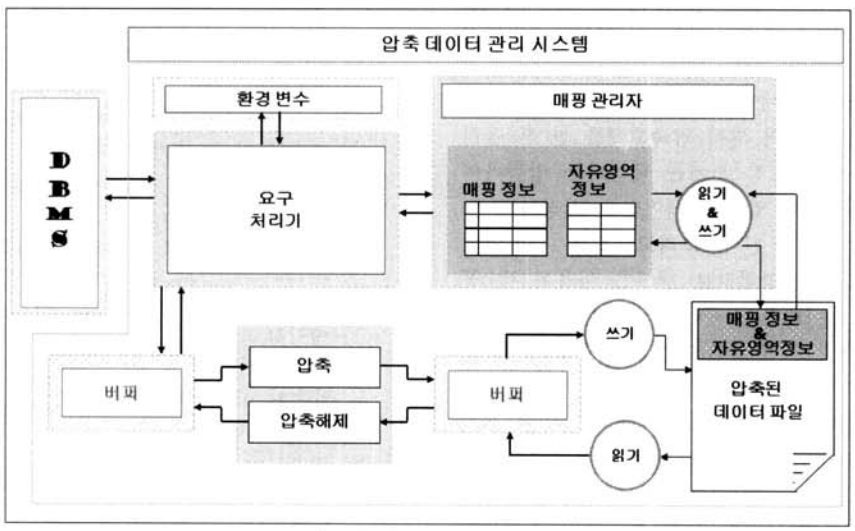
먼저 본 논문이 제안하는 CDMS는 저장 공간의 절약을 위해 압축 기법을 사용한다. 또한 CDMS는 기존 DBMS의 변경 최소화를 위해 DBMS의 스토리지 관리자와 파일 시스템 사이에 들어가게 된다. DBMS는 기존에 파일시스템으로 보내던 데이터를 CDMS에게 보내게 되고 이것을 CDMS가 압축하여 파일시스템에 보내게 된다. (그림 3)은 기존 시스템과 CDMS의 데이터 교환을 나타낸 것이다.

(그림 3)의 (가)에서 보는 바와 같이 기존의 시스템은 모바일 DBMS에서 파일을 관리하기 위해 데이터를 읽고 쓰는 것에 대한 요청을 파일시스템에 하게 된다. 이때 모바일



(가) 기존 시스템 구조 (나) CDMS를 이용한 구조

(그림 3) 플래시 메모리를 사용한 정보기기에서 데이터교환



(그림 4) CDMS의 구조

DBMS와 파일시스템 두 계층 사이의 데이터 교환은 DBMS가 정의한 페이지 단위의 파일 I/O를 사용한다. 하지만 이 논문에서 제안하는 CDMS를 이용한 새로운 구조는 (그림 3)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 4)는 위에서 설명한 CDMS의 자체적인 처리를 할 수 있는 CDMS의 전체적인 구조도이다.

(그림 4)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 자유영역정보(Free Space Information)를 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 다시기록 되어야 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 부분을 저장 가능한 공간으로 표시되어야 한다. 이렇게 발생된 재사용이 가능한 공간의 표시를 위해 자유영역정보에 빈 공간에 대한 정보를 저장하여 관리하게 된다.

〈표 2〉 CDMS를 사용한 DBMS의 데이터읽기, 쓰기 요청처리 절차

DBMS의 데이터읽기 요청	DBMS의 데이터쓰기 요청
1) 요청된 데이터를 포함하는압축된 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

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 2>에서 나타난 바와 같다.

<표 2>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. 하지만 이렇게 저장되는 데이터는 압축률에 따라 크기가 각각 다르다. 크기가 다른 압축된 데이터로 인해 기존 시스템처럼 정해진 규격에 따른 관리가 어렵게 된다. 따라서 크기가 다른 압축된 데이터들의 효율적인 관리를 위해 본 논문에서는 2가지의 관리기법을 제안한다. 이 관리기법을 다음 장에서 설명한다.

단계 1 : DBMS의 데이터 페이지 쓰기요청 및 초기 처리
1) 쓰기 요청된 페이지의 페이지번호를 계산
페이지 번호 = $\frac{file\ pointer\ of\ file\ descriptor}{size\ of\ a\ page}$
2) 페이지관련 매핑정보가 있는지 확인하여 있으면 관련 정보 저장

단계 2 : 페이지의 압축과 압축된 블록의 저장
1) 페이지의 압축
2) 자유영역정보에서 압축된 블록의 크기보다 크면서 가장 앞쪽의 빈 공간 검색
(자유영역정보에 없을 때는 파일 끝부분을 빈 공간으로 검색)
3) 검색된 빈 공간에 압축된 블록 저장

단계 3 : 압축된 블록 관리정보 갱신
1) 매핑정보 갱신
DBMS가 요청한 데이터의 페이지에 해당하는
압축된 데이터의 첫 위치 번호와 압축된 페이지의 크기 정보 변경
2) 자유영역정보 갱신
단계 2에서 자유영역정보에 있는 빈 공간을 사용하였다면 제거
단계 1 → 2)에서 정보가 있었다면 그 정보를 자유영역정보에 추가

4.2 압축데이터 관리기법

이 장에서는 압축된 데이터들의 효율적인 관리를 위해 2개의 관리기법, 동적기법과 정적기법을 제안한다. 먼저 압축된 데이터의 크기에 따라 처리하는 동적 기법을 4.2.1에서 기술하고 그 후 가상의 슬롯을 이용하는 정적 기법을 4.2.2에서 기술한다.

4.2.1 동적 기법

CDMS의 압축 데이터 관리기법 중 하나인 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 동작 방식은 쓰기 작업의 절차를 보면 그 특징을 알 수 있다. 다음은 동적 기법을 사용한 CDMS의 쓰기 절차이다.

동적기법은 위에서 보는 바와 같이 단계 2에서 빈 공간을 찾을 때 압축된 블록의 크기보다 크면서 가장 앞쪽의 빈 공간을 찾는다. 이 때 공간의 단위는 바이트 단위이고 자유영역정보에 빈 공간이 없다면 파일 마지막부분에 이어서 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. 따라서 본 논문에서는 압축된 블록의 관리가 편리하고 효율적일 수 있도록 가상의 슬롯을 사용하는 정적 기법을 제안한다.

4.2.2 정적 기법

본 논문에서 제시하는 또 하나의 관리기법은 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 슬롯단위

단계 1 : DBMS의 데이터 페이지 쓰기요청 및 초기 처리
1) 쓰기 요청된 페이지의 페이지번호를 계산
페이지 번호 = $\frac{file\ pointer\ of\ file\ descriptor}{size\ of\ a\ page}$
2) 페이지관련 매핑정보가 있는지 확인하여 있으면 관련 정보 저장

단계 2 : 페이지의 압축과 압축된 블록의 저장
1) 페이지의 압축
2) 자유영역정보에서 압축된 블록의 크기보다 크면서 앞쪽의 빈 슬롯 검색
(자유영역정보에 없을 때는 파일 끝부분을 빈 슬롯으로 검색)
3) 검색된 빈 슬롯에 압축된 블록 저장

단계 3 : 압축된 블록 관리정보 갱신
1) 매핑정보 갱신
DBMS가 요청한 데이터의 페이지에 해당하는 슬롯 번호와 압축된 블록의 크기 정보 변경
2) 자유영역정보 갱신
단계 2에서 자유영역정보에 있는 빈 슬롯을 사용하였다면 제거
단계 1 → 2)에서 정보가 있었다면 그 정보를 자유영역정보에 추가

로 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.

다음은 정적 기법을 사용한 CDMS의 쓰기 절차이다.

위에서 보는 바와 같이 단계 2에서 동적기법과 다르게 슬롯 형태로 압축된 블록을 저장한다. 따라서 자유영역정보 또한 슬롯형태로 관리가 가능해져서 단계 3의 2)에서 자유영역정보에 추가되는 정보가 슬롯형태의 정보가 된다.

하지만 이 정적기법도 동적 기법의 빈 공간이 많아지는 것을 보완하기 위해 나온 기법이지만 슬롯 안에 사용하지 못하는 빈 공간이 존재하는 등의 단점이 존재한다. 따라서 각각의 특성에 맞게 관리기법을 결정하는 것이 중요하다. 이것을 위해 다음 절에서는 동적기법과 정적기법의 비교를 통해 그 특성을 파악한다.

4.3 압축데이터 관리기법 비교

위에서 설명한 동적기법과 정적기법은 슬롯의 사용유무로 인해 차이가 생긴다. 다시 말해 동적기법은 슬롯을 사용하지 않음으로 인해 슬롯의 사용으로 발생할 수 있는 내부적인 빈 공간이 없다. 또한 슬롯의 크기를 설정해야 할 필요도 없다. 하지만 각각의 블록에 맞게 바이트 단위로 처리되기 때문에 1바이트의 차이도 허용되지 않는다. 또한 가장 앞쪽의 빈 공간부터 채움으로 인해 빈 공간의 크기를 고려하지 않게 된다. 따라서 들어오는 압축된 블록과 빈 공간의 크기 차이가 생기게 된다. 이 때 생기는 빈 공간은 다른 데이터를 넣기에는 너무 작아 계속적으로 사용되지 않게 된다. 이렇게 생기는 빈 공간으로 인해 효율적인 공간 사용에 큰 영향을 끼친다. 또한 이러한 빈 공간의 정보를 유지해야 되는 자유영역정보가 계속적으로 늘어나게 된다. 반면에 정적기법은 슬롯을 사용함으로 인해 이러한 동적기법의 단점을 해소한다. 슬롯이 이러한 문제를 발생하지 않도록 완충작용을 하는 것이다. 그 결과 다른 데이터를 넣기에 너무 작은 빈 공간이 발생하지 않게 되고, 따라서 자유영역정보도 줄어들게 된다. 또한 발생되는 빈 공간도 슬롯단위이기 때문에 1바이트라도 모자라면 못 들어가는 동적기법보다 압축된 블록이 어떤 빈 공간이든지 들어갈 수 있는 확률이 높아진다. 하지만 정적 기법은 적절한 슬롯 크기를 설정해야 되며, 그렇지 않았을 경우 잘못된 슬롯 크기를 선택으로 슬롯 내부에 빈 공간이 생길 수 있다. 이것은 자유영역정보에 기록되지 않는 CDMS가 처리하지 못하는 빈 공간이다.

<표 3>은 위에 설명한 각 관리기법의 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 관리기법이 다른 관리기법보다 항상 우수하지 않고 사용되는 환경에 따라 우수한 관리기법이 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

〈표 3〉 CDMS의 각 관리기법 장단점 비교

관리기법	장점	단점
동적 기법	■ 슬롯 사용으로 생기는 내부 빈 공간이 없음 ■ 슬롯 크기 같은 설정이 불필요	■ 압축된 블록의 미묘한 크기 차이로 균일하지 않는 빈 공간이 발생하여 관리가 어려움 ■ 특정 사이즈 이하의 빈 공간이 계속해서 사용되지 않고 남을 수 있음
정적 기법	■ 슬롯의 사용으로 관리가 잘 됨 ■ 슬롯 크기 이하의 빈 공간은 발생하지 않음	■ 슬롯 크기에 영향을 받음 ■ 슬롯 사용으로 내부 빈 공간 발생

관리기법이 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.

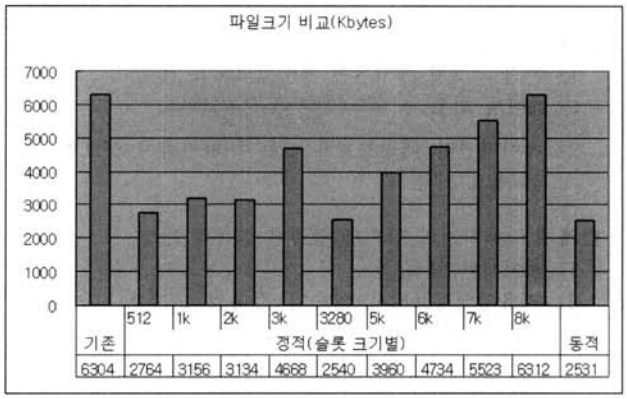
5. 성능 평가

이 장에서는 본 논문이 제안하는 CDMS의 타당성을 검증한 실험결과 및 분석결과를 제시한다. 이 실험을 위하여 실제 상용화 되어 있는 모바일 DBMS의 LINUX 개발 환경을 사용하였다. 실험을 통해 기존 DBMS와 CDMS를 사용한 DBMS의 차이를 확인하고 나아가 모바일 환경인 플래시 메모리상에서의 결과를 예측 할 수 있도록 하였다. 이 실험은 DBMS에서 일어날 수 있는 모든 상황을 확인하기 위해 상용 모바일 DBMS의 개발당시 DBMS의 검증을 위해 사용된 품질 보증(Quality Assurance)을 위한 QA 프로그램을 사용하였다. 이 프로그램은 DBMS가 처리해야 되는 64M바이트에 달하는 1,000여개의 질의 파일을 포함하고 있으며, 발생되는 데이터베이스 파일은 (그림 5)에서 보이듯이 6M바이트의 데이터를 포함한다. 또한 각 질의 파일들은 11,000여개의 테이블을 생성하게 된다. 본 논문은 이러한 QA 프로그램을 실행시킴으로 본 논문이 제안하는 시스템을 검증하였고, QA 프로그램을 실행시키면서 여러가지 결과들을 도출하여 제안하는 시스템의 특성을 확인하였다.

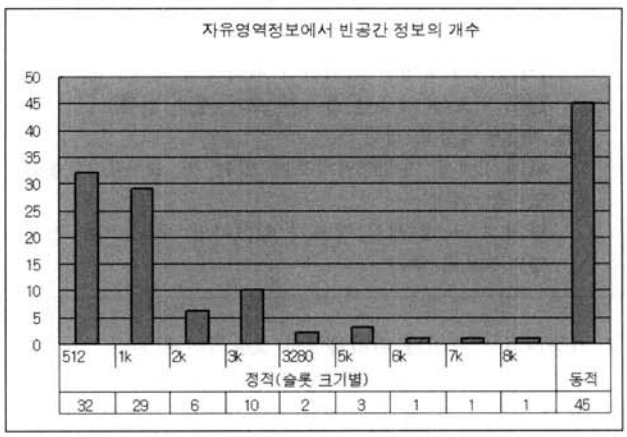
일반적으로 실험은 모바일 DBMS에 CDMS를 추가하여 QA 프로그램을 돌려본 결과와 모바일 DBMS 자체로만 QA 프로그램을 돌려본 결과를 비교하였다. 이때 CDMS는 2가지의 관리기법으로 나누어 실험하였고, 추가적으로 정적기법은 슬롯크기를 512바이트, 1K바이트, 2K바이트, 3K바이트, 3280바이트, 5K바이트, 6K바이트, 7K바이트, 8K바이트로 나누어 각각 실험 하였다. 이때 3280바이트의 슬롯은 적절한 슬롯 크기의 선택이 중요한 것을 확인하기 위해 저장되는 데이터가 대부분 3280바이트로 압축되는 것을 미리 확인하여 4K바이트가 아닌 3280바이트로 설정하였다.

성능에서 CDMS의 목표는 우선적으로 DBMS의 데이터 파일의 크기를 줄이는 것이다. 이와 같은 성능을 확인하기 위해서 본 논문에서는 먼저 QA 프로그램이 완료된 시점에서의 데이터를 저장하고 있는 파일 크기를 측정하였다.

(그림 5)는 계속적인 데이터의 추가가 많이 일어나는 QA 프로그램을 실행시킨 결과를 나타낸 것이다. 이 실험을 통해 CDMS의 사용이 기존방식보다 최대 60%, 평균 40%정도 파일 크기를 줄여 주는 것을 확인할 수 있었다. 그리고 슬롯 내부 빈 공간으로 인해 정적 기법이 동적 기법보다 크기 감소가 적은 것을 알 수 있었다. 또한 정적 기법은 슬롯 크기가 증가 할수록 내부 빈 공간 때문에 파일크기가 계속 증가하는 것을 볼 수 있다. 특히 8k에서 정적 기법은 기존 시스템보다 나쁜 성능을 보이는데, 이것은 정적 기법에서 잘못된 가상 슬롯 크기 선택이 기존 시스템보다 나쁜 성능을 보일 수 있음을 보여준다. 반면에 슬롯 크기가 3280바이트일 때는



(그림 5) QA 프로그램 실행 후 데이터 파일크기 비교



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

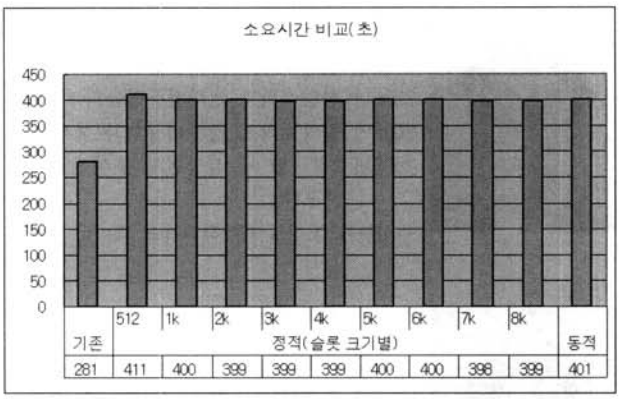
적당한 슬롯크기를 선택함으로 가장 크기가 작은 것을 확인할 수 있었다. 이것은 정적 기법에서 슬롯 크기 선택이 공간 절약측면에서 얼마나 중요한지를 나타내고 있다.

(그림 6)은 자유영역정보에 저장되어 있는 빈 공간 정보의 개수를 보인 것이다. (그림 6)을 통하여 정적 기법의 경우 자유영역정보에서 빈 공간의 정보가 적은 것을 알 수 있었다. 즉 슬롯을 사용함으로 자유영역정보를 효율적으로 관리할 수 있다. 하지만 슬롯의 크기가 작아지면서 완충 작용을 할 수 있는 범위가 줄어들어 빈 공간의 정보가 늘어나는 것을 볼 수 있다.

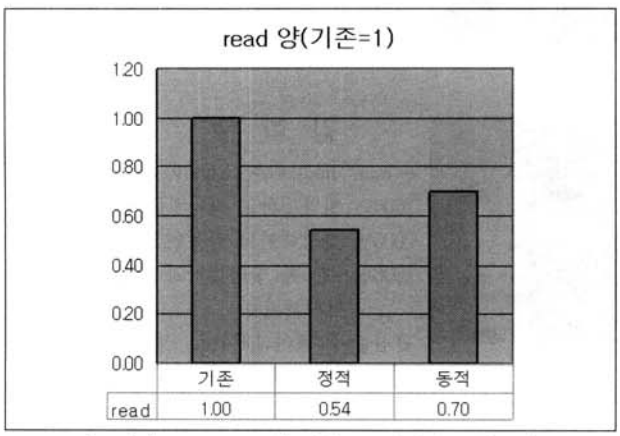
이상의 실험으로 CDMS를 사용함으로 인해 평균 40%의 공간 절약을 가져오는 것을 확인할 수 있었다. 그리고 슬롯을 사용함으로 인해 빈 공간의 관리가 효율적으로 이뤄지는 것과 잘못된 슬롯 크기의 선택으로 기존보다 더 나쁜 성능을 보일 수 있다는 것을 확인하였다.

또 다른 성능에서 CDMS의 목표는 기존 시스템과의 소요시간 차이를 최소화 시키는 것이다. 이와 같은 성능을 평가하기 위하여 본 논문에서는 먼저 QA 프로그램이 완료된 시점까지 걸리는 시간을 측정하였다. (그림 7)은 QA 프로그램이 완료되는 시간을 측정한 결과이다.

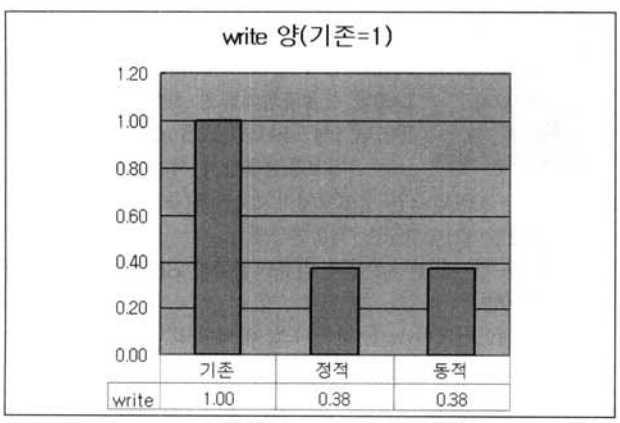
이 실험을 통해 속도 면에서 CDMS를 사용한 시스템이



(그림 7) 기존방식과 제안된 CDMS방식의 QA 프로그램 소요시간 비교



(그림 8) QA 프로그램 실행 중 데이터 I/O(read)



(그림 9) QA 프로그램 실행중 데이터 I/O(write)

압축 연산으로 인해 평균적으로 기존 시스템보다 40%의 추가 시간이 필요한 것을 확인하였다. 이것은 저장 공간의 절약이 최우선되는 목적인 CDMS가 앞서 확인한 바와 같이 최대 60%, 평균 40%의 저장 공간을 줄이는 부담으로써, 압축 알고리즘이 압축률과 소요시간이 비례한다는 특성을 바탕으로 압축률을 줄임으로 시간성능을 향상 시킬 수 있다. 따라서 압축률과 압축시간의 적절한 조절로 사용 환경에 적합한 시스템이 될 수 있다.

(그림 8)과 (그림 9)는 모바일 DBMS의 QA 프로그램 실행시 DBMS와 저장매체 사이의 전체 I/O량을 측정한 것이다. (그림 8)는 저장매체에서 데이터를 읽는 양이고, (그림 9)는 저장매체로 데이터를 쓰는 양이다. 이 실험 결과를 통해 데이터 I/O가 기존 시스템보다 제안하는 CDMS를 사용하였을 때 현저히 낮은 것을 알 수 있다. 따라서 쓰여 질 데이터의 크기를 줄임으로 플래시 메모리의 수명을 연장할 수 있게 된다.

위의 일련의 실험을 통해 본 논문에서 제안하는 CDMS을 검증하였다. CDMS를 사용함으로 공간 절약이 가능하였고, 반면에 추가 소요시간이 필요한 것을 확인하였고, 기타 설계 목적들 또한 검증 하였다.

6. 결 론

유비쿼터스 시대를 맞아 휴대용 정보기기가 보편화 되어 가고 있는 시점에서 보다 많은 정보를 저장하고 처리할 수 있는 휴대용 정보기기의 필요성이 증대되고 있다. 하지만 비용과 크기의 문제로 인해 저장 공간을 늘리기에는 한계가 있다. 이러한 요구를 만족시키기 위한 방법으로 본 논문에서는 플래시 메모리를 위한 압축을 이용한 DBMS의 효율적인 데이터 관리를 제안하였다. 이 데이터 관리는 또한 두 가지 방식으로 나누어, 적용 환경의 특성에 따라 더 효율적인 관리를 할 수 있도록 설계하였다. 성능 평가를 통해서 검증 하였듯이 이 CDMS는 데이터베이스가 압축되지 않은 상태로 저장되는 현재 상황을 고려해 볼 때 획기적인 저장 공간의 절약을 가져온다. 또한 저장매체에 써야 되는 데이터를 줄임으로 디스크 I/O를 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.

하지만 압축 알고리즘의 개선과 무분별하게 커지는 자유영역정보의 효율적인 관리 그리고 비정상 종료에도 데이터의 손실을 막는 것은 추가적인 연구가 필요한 부분이다. 또한 실험을 통해 나타난 매핑정보와 자유영역정보의 갱신을 위해 많은 시간이 소모되는 것도 추가적인 연구가 필요하다.

본 논문에서 제안하는 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 효율적으로 관리하기 위해서는 본 논문이 제안하는 CDMS가 필요하다.

참 고 문 헌

[1] 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.
[2] Mark A. Roth, Scott J. Van Horn "Database compression," ACM SIGMOD Record, Vol.22 No.3, p.31-39, 1993.
[3] 변시우, 노창배, 정명희, "휴대용 정보기기를 위한 플래시 기반 2단계 로킹 기법," 한국데이타베이스학회 : 정보기술과 데이타베이스 저널 제12권 제4호, 2005, pp.59-70.
[4] 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.
[5] 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, Sept., 2000.
[6] Balakrishna R. Iyer, David Wilhite, "Data Compression Support in Databases," Proceedings of the 20th VLDB Conference, Santiago, Chile, pp.695-704, 1994.
[7] G.V. Cormack. "Data compression on a database system," Communication of the ACM, 28:12, pp.1336-1342, 1985.
[8] G. Graefe, L. Shapiro, "Data Compression and Database Performance," In ACM/IEEE-CS Symp. On Applied Computing, pp.22-27, 1991.
[9] W.P. Cockshott, D. McGregor, N. Kotsis, J. Wilson, "Data Compression in Database Systems," dexa, 9th International Workshop on Database and Expert Systems Applications (DEXA'98), p.981, 1998.
[10] 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.
[11] 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.
[12] Intel Corporation, "Understanding the Flash Translation Layer(FTL) Specification," APPLICATION NOTE, AP-684, 1998.
[13] Jeong-Uk Kang, Heeseung Jo, Jin-Soo Kim, Joonwon 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.
[14] Chung, T.-S. ; Park, D.-J. ; Park, S. ; Lee, D.-H. ; Lee, S.-W. ; Song, H.-J. "System Software for Flash Memory: A Survey," Lecture notes in computer science, Vol.4096, pp.394-404, 2006.
[15] 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.

신 영 재

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

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

김 학 수

e-mail: hagsoo@cse.hanyang.ac.kr
2004년 한양대학교 전자컴퓨터공학과(학사)
2006년 한양대학교 컴퓨터공학과(석사)
2006년~현 재 한양대학교 컴퓨터공학과 박사과정
관심분야 : 데이터베이스, 시멘틱 마이닝, e-비즈니스

이 승 미

e-mail : smlee@database.hanyang.ac.kr
1989년 숭실대학교 전산학과(학사)
1991년 한국과학기술원 전산학과(석사)
1999년 한국과학기술원 전산학과(박사)
1991년 1월~1993년 12월 충남대학교 정보통신공학과 시간강사
2000년 3월~2001년 3월 엘엔에치이치코리아(주) 음성언어기술연구소 책임연구원
2006년 3월~현 재 한양대학교 컴퓨터공학과 BK21 사업팀 연구원
관심분야 : 데이터베이스, e-비즈니스, 임베디드 시스템

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