Mobile DBMS를 위한 효율적인 압축 데이터 관리 시스템*

신영재, 황진호, 박정업, 김학수, 이승미, 손진현
한양대학교 컴퓨터공학과
{yjshin, jhhwang, jupark, hagsoo, smlee, jhson}@cse.hanyang.ac.kr

Development of the Efficient Compressed Data Management System For Mobile DBMS

Youngjae Shin, Jin-ho Hwang, Jung Up Park, Hak Soo Kim, Jin Hyun Son
Department of Computer Science and Engineering, Hanyang University

요 약

최근 휴대용 정보기기 사용이 보편화되어지고 있는 가운데 정보의 디지털화로 인해 휴대용 정보기기에서 처리되어야 하는 정보가 무수히 많아지고 있다. 이로 인해 휴대용 정보기기에서는 정보들을 효과적으로 관리하기 위해 임베디드 DBMS의 사용이 요구되고 있다. 또한 휴대용 정보기기에서 보편적으로 사용되는 저장장치는 NAND형 플래시 메모리로 단위 공간당 비용이 기존의 하드디스크에 비해 수십배 가량 높아 저장 공간의 효율적인 관리가 요구되고 있다. 따라서 본 논문에서는 플래시메모리를 저장매체로 사용하는 DBMS에서 압축기법을 사용한 효율적인 데이터 관리 시스템을 제안한다. 제안되는 압축 기반 시스템은 저장 공간과 데이터I/O를 줄이며, 데이터I/O를 줄임으로써 DBMS의 성능향상과 플래시 메모리의 수명을 연장시키는 효과를 기대할 수 있다.

1. 서 론

PDA(Personal Digital Assistant), HPC(Hand-held PC), PPC(Pocket PC), 개인용 휴대전화(mobile phone), 스마트폰(Smart Phone)등 현대의 휴대용 정보기기는 기본적인 정보의 처리, 저장 수단으로만 사용하던 것에서 높은 휴대성과 광범위한 정보를 저장, 처리할 수 있는 기기로 바뀌고 있다. 이러한 환경의 변화로 휴대용 정보기기에서는 보다 많은 정보의 생성, 처리, 저장이 가능한 다양한 응용프로그램의 사용을 요구 받고 있다. 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 임베디드 데이터베이스 관리 시스템이 필요하다고 보고했다.

기존 컴퓨팅 환경에서는 가격이 저렴하고 확장성이 용이한 하드디스크를 데이터 저장장치로 사용함으로 저장공간에 대한 비용이 많이 절감되었다. 하지만 높은 휴대성과 내구성, 저 전력 소모를 요구하는 모바일 컴퓨팅 환경에서 전력을 많이 소모하며 크기, 소음, 진동 등의 단점을 가진 하드디스크의 사용이 어렵게 되었다. 따라서 모바일 정보기기에서는 이러한 단점을 보안할 수 있는 저전력으로 장시간의 구동이 가능하며, 부피가 작고 경량으로 소음과 진동이 없으며 물리적인 충격에 강해 휴대가 용이한 플래시 메모리를 보조기억장치로 사용한다. 이 플래시 메모리는 크게 바이트 단위로 I/O를 지원하는 NOR형 플래시 메모리와 페이지 단위의 I/O만을 지원하는 NAND형 플래시 메모리로 나뉜다. 모바일 정보기기에서는 대용량 데이터 저장장치로 주로 사용되는 NAND형 플래시 메모리를 사용한다.

이 NAND형 플래시 메모리는 기존의 하드디스크에 비해 고비용과 데이터 I/O에 따른 수명을 가진다는 단점이 있다. 이로 인해 데이터의 효율적인 관리가 필요하다.

본 논문에서는 현재 가장 보편적으로 모바일 정보기기에서 저장장치로 사용되고 있는 NAND형 플래시 메모리에 데이터베이스의 데이터를 효율적으로 저장하는 방법으로 효율적인 데이터 압축 관리기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 살펴본다. 3장에서는 효율적인 압축 데이터 관리의 필요성에 설명하며 4장에서는 이 필요성을 충족시키는 압축 데이터 관리기법으로 제안하는 CFM을 설명한다.

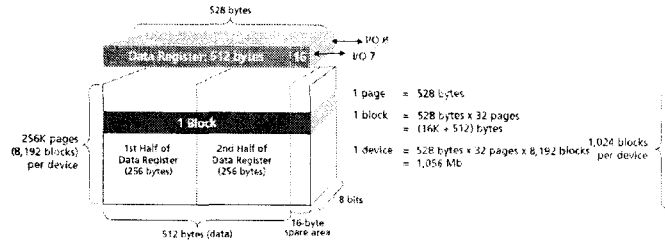
2. 관련 연구

이 장에서는 데이터베이스와 데이터베이스에서 데이터 압축과 플래시메모리의 특성을 알아본다.
이것은 이 논문에서 압축 데이터 관리 시스템에 대한 기본 지식이 된다.

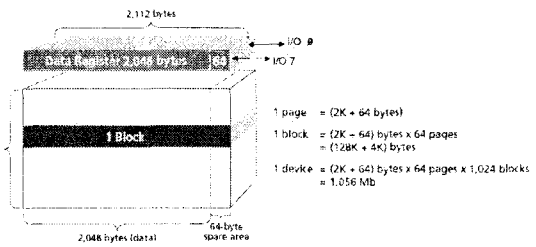
2.1 데이터베이스에서 데이터 압축

압축된 데이터를 사용하여 시스템의 효율을 높이고자 하는 연구는 현재까지 지속적으로 연구되어 왔다. 이와 관련된 연구는 크게 데이터 압축방식과 압축 데이터의 검색으로 나눌 수 있다. 이 중 대부분의 연구는 데이터 압축방식에 대한 연구로써, 대표적으로 초기 Huffman coding을 이용한 방식과 Run-Length coding을 이용한 방식으로 나눌 수 있다. Huffman Coding와 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. 이러한 데이터 압축기법이 데이터베이스 시스템의 성능에 미치는 효과에 대한 연구 또한 진행되었다. 이와 같은 연구에서 데이터 압축기법을 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

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원 사업(IITA-2005-C1090-0502-0016)의 연구결과로 수행되었음



(그림 1) 소블록 NAND형 플래시 메모리(1GB)



(그림 2) 대블록 NAND형 플래시 메모리(1GB)

I/O 횟수의 감소이다. 압축된 데이터는 압축되지 않은 데이터에 비해 동일한 물리적 공간에 보다 많은 데이터를 저장할 수 있게 된다. 즉, 하나의 데이터 블록에 압축된 상태의 보다 많은 데이터가 기록되므로, 일정량의 데이터에 대한 읽기, 쓰기 명령에 의해 일어나는 디스크 I/O의 횟수가 감소하게 되고, 이는 DBMS에서 I/O 병목현상 경감과 대여폭증가의 효과를 의미한다. 이러한 이점에도 불구하고 실제로 데이터 압축방식은 대부분의 고성능의 DBMS에서 전형적으로 사용되고 있지 않다. 상용 DBMS인 DB2에서 이와 같은 압축데이터를 활용하여 시스템의 성능향상과 저장공간 절약의 이득을 보인 연구는 있었다. 이 연구에서 테이블의 열 단위 압축을 사용하며 Ziv-Lempel 압축 알고리즘을 사용해서 비번한 접근이 일어나고 적은 데이터를 담고있는 테이블은 압축하지 않고, 많은 데이터를 담고 있지만 적은 접근이 일어나는 테이블들은 선택적으로 압축하는 기법을 제시하였다. 이러한 기존 연구들을 통해 알 수 있듯이 DBMS에서 데이터 압축을 통해 시스템의 효율을 높이기 위한 연구는 압축 알고리즘과 압축단위로써 테이블의 필드 혹은 열을 사용하는 것에 초점이 맞추어져 있었다. 본 논문에서는 기존 DBMS의 수정을 최소화 할 수 있도록 입출력 page단위를 이용한 데이터 압축 구조를 설계함으로써 특정 DBMS에 비종속적 데이터 압축 구조를 제안하고, 압축된 데이터에 대한 효율적인 관리기법을 제안함으로써 기존 연구와 구별된다.

2.2 플래시메모리의 특성

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 NOR형 플래시 메모리와 NAND형 플래시 메모리의 특성을 수치적으로 비교한 것이다.

	Read (512B)	Write (512B)	Erase	Cost/MB
NOR-type Flash	15㎲	3.5ms	1.2s (128KB)	20~30
NAND-type Flash	36㎲	226㎲	2ms (16KB)	10~20

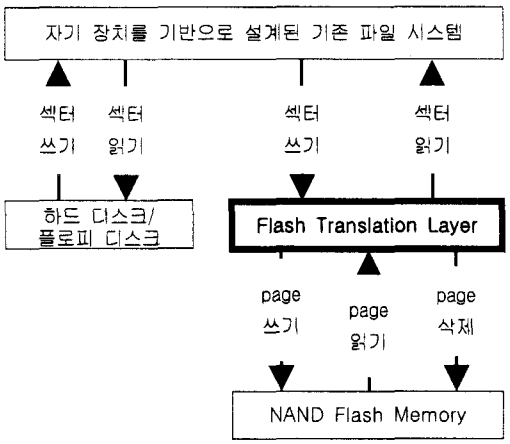
<표 1> NAND형과 NOR형 플래시 메모리의 특성비교

이러한 특성으로 인해 현재 모바일 기기들에 가장 보편적으로 NAND형 플래시 메모리가 사용되고 있다.

NAND형 플래시 메모리는 전력 소모가 작고 충격에 강하여 소형화가 가능하지만, 쓰기위해서는 파일이 쓰여질 위치에 내용이 모두 지워져야 된다. 또한 NAND형 플래시 메모리는 읽기/쓰기는 페이지 단위로 이루어지고, 지우기는 블록 단위로 이뤄진다. 다시 말해 쓰기/읽기 단위와 지우기 단위가 동일하지 않다.

NAND 플래시 메모리는 페이지에 크기에 따라 소블록 NAND형 플래시 메모리와 대블록 NAND형 플래시 메모리로 나눠지는데, 소블록 NAND형 플래시 메모리는 512바이트의 메인영역과 16바이트의 스페어영역으로 페이지가 구성되고, 32개의 페이지가 블록을 구성한다. 대블록 플래시 메모리는 2048바이트의 메인영역과 64바이트의 스페어영역으로 페이지가 구성되고, 64개의 페이지가 블록을 구성한다. (그림 1)과 (그림 2)는 소블록 NAND형 플래시 메모리와 대블록 NAND형 플래시 메모리로 1GB 메모리의 구조를 나타낸 것이다.

위에 설명된 플래시 메모리의 특성에 때문에 기존의 파일시스템을 플래시 메모리에 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 3)에 FTL의 동작을 보였다.



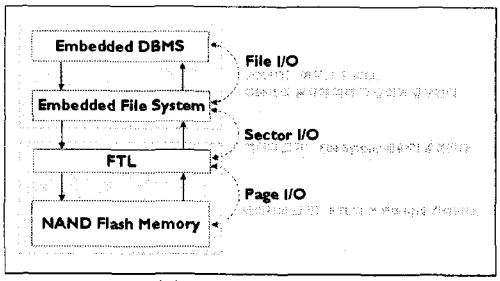
(그림 3) FTL의 동작

FTL은 파일시스템으로부터 오는 읽기/쓰기 요청을 받아서 플래시 메모리에 페이지 읽기/쓰기 작업과 블록 삭제작업 등의 작업을 진행한다.

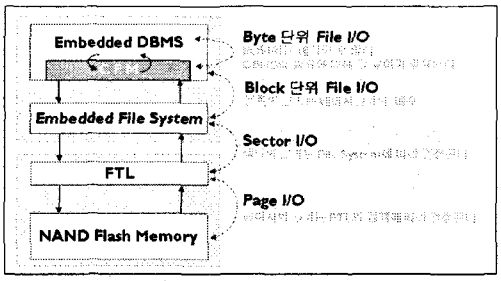
위에서 살펴본 바와 같이 데이터베이스 압축기술과 디스크에서의 압축 데이터 관리기법, 플래시메모리 관련 연구는 있어왔다. 하지만 본 논문에서 제안하는 것과 같이 플래시메모리에 적합한 데이터베이스 압축 관리기법에 대한 연구는 없었다.

3. 데이터 압축 관리의 필요성

모바일 환경에서 DBMS는 기존 DBMS와 달리 저장공간, 데이터 I/O 비용 등의 자원에 대한 제한을 받게 된다. 또한 점차 증가되는 개인정보와 정보기기에서 대용량 정보에 대한 처리를 위하여 저장공간에 대한 효율적 사용이 요구된다. 모바일 환경에서의 저장장치는 플래시 메모리로, 대부분의 휴대용 정보



(a) 기존 시스템 구조



(b) CFM을 사용한 구조

(그림 4) 플래시 메모리를 사용한 정보기기에서 데이터교환

기기에서는 보조기억장치로 기존의 하드디스크를 대신하여 사용한다.

이와 같이 모바일 환경에서 DBMS는 두 가지의 환경적인 문제를 가지게 된다. 첫째는 제한된 메모리(자원) 사용에 따른 저장공간 부족이다. 휴대용 정보기기에서 처리 요구되는 정보의 양이 크게 증가되고 있는 상황에서 DBMS에 의해 관리되는 정보량 또한 크게 증가하고 있다. 하지만 모바일 환경은 자원에 대한 제한으로 인해 많은 양의 데이터의 저장, 처리를 지원할 수 없다. 둘째는 플래시 메모리 사용에 있어 데이터 I/O비용을 감소시켜야 한다. 플래시 메모리는 휴대용 저장장치에 적합한 여러가지 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 I/O는 플래시 메모리의 수명과 직결되는 문제를 야기한다.

이 두 가지 문제를 해결하기 위한 방법으로 본 논문에서는 데이터 압축 관리기법을 제안한다. DBMS에서 처리되는 데이터를 압축하여 관리함으로써 제한된 자원사용에 따른 저장 공간 부족문제와 플래시 메모리에서 많은 데이터 I/O에 의한 성능저하 및 플래시 메모리 수명문제에 대한 효율적인 대처가 가능하다. 또한 압축을 통해 데이터에 대한 암호화로 보안효과를 취할 수 있을 것이다. 본 논문에서 제안하는 데이터 압축관리 기법은 모바일 환경을 위한 것으로 DBMS와 플래쉬 메모리의 파일시스템 사이에서 압축데이터 처리를 위한 구조를 제시한다.

4. 압축 데이터 관리기법

이 장에서는 본 논문이 제안하는 압축 데이터 관리기법에 대해 기술한다. DBMS와 파일시스템 사이에서 사용하는 압축 데이터 관리 시스템으로 CFM을 4.1절에서 제안하고 CFM에서 압축 데이터를 관리하는 기법에 대해 4.2절에서 기술한다.

4.1 CFM 제안

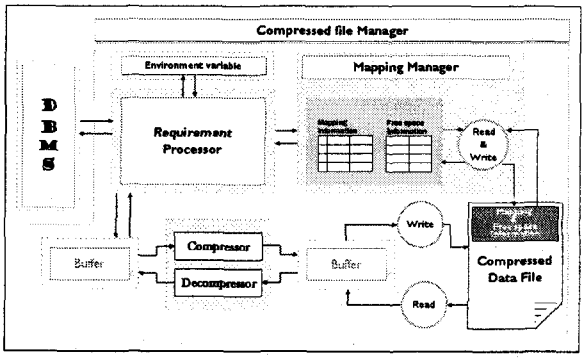
모바일 정보기기에서 발생하는 저장공간 부족문제는 데이터를 압축함으로 해결할 수 있다. 본 논문에서는 데이터 압축기법을 사용하기위해 CFM(Compressed File Manager)을 사용한 새로운 구조를 제안한다.

다음은 본 연구에서 제안하는 CFM의 설계 목표이다.

① DBMS의 스토리지 관리자와 파일 시스템간의 실제 File I/O 단계에서의 데이터 압축을 통해 기존 DBMS의 변경 최소화
② 데이터에 대한 압축 기법 사용으로 플래시 메모리의 효율적인 공간 관리
③ DBMS와 플래시 메모리의 특성을 고려한 처리로 데이터 I/O 비용 최소화로 시스템의 성능향상

기존의 시스템은 (그림 4)에서 보는 바와 같이 임베디드 DBMS에서 데이터베이스를 관리하기 위해 데이터파일을 읽고 쓰는 것에 대한 요청을 파일시스템에 하게 된다. 이때 임베디드 DBMS와 파일시스템 두 계층 사이의 데이터 교환은 파일 I/O를 사용한다. 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. 임베디드 DBMS와 파일시스템 사이의 파일 I/O는 파일포인터를 사용하여 DBMS가 데이터 파일 내의 필요한 데이터에 대한 읽기 쓰기를 하는 것이다. 이러한 처리에서 파일시스템은 해당 데이터가 저장된 논리적 섹터를 찾게 되고 해당 섹터에 대한 처리요청을 FTL에 하게 된다. FTL은 파일시스템이 요청한 논리적 섹터의 데이터에 매칭되는 물리적 페이지를 플래시 메모리에서 찾아 요청에 대한 처리를 하게 된다. 이때 파일시스템의 처리 단위인 섹터와 플래시 메모리에서 실제 데이터 저장단위인 페이지는 같은 크기를 가지게 된다. CFM은 이러한 플래시 메모리를 사용하는 파일시스템의 특성에 맞추어 데이터베이스의 데이터파일을 플래시메모리 Page크기의 배수이면서 DBMS에서 파일I/O의 단위로 사용되는 세그먼트 혹은 페이지의 크기로 기반하여 데이터 블록의 크기를 정의하고 데이터파일을 각 블록으로 나누어 압축, 저장, 관리하게 된다.

이 논문에서 제안하는 CFM을 이용한 새로운 구조는 (그림 4)의 (b)와 같이 DBMS의 파일에 대한 접근, 처리를 CFM에 의해 이루어지 도록하는 것이다. 데이터 압축을 사용하여 데이터파일 내에 압축된 블록들이 존재 하지만 DBMS는 압축되지 않은 데이터파일에 접근하는 방식과 동일하게 데이터파일에 대한 접근요청을 CFM에 할 수 있다. 이러한 요청에 대해 CFM은 압축된 데이터파일내에서 DBMS가 요청한 세그먼트 혹은 페이지크기의 데이터가 압축된 블록을 찾아 해당 블록을 파일시스템에 요청하게 되는 것이다.



(그림 5) CFM을 사용한 구조

DBMS가 요청한 데이터를 압축된 데이터파일에서 찾기 위해 CFM은 데이터파일 내부에 매핑정보를 포함시키고 있다. 이 매핑정보는 논리적 블록번호, 압축된 블록의 파일내 위치값을 가지게 되며, 이러한 매핑정보를 이용해 압축되지 않은 상태의

데이터파일과 압축된 데이터파일의 매핑이 가능해진다. 매핑정보에 포함된 블록번호는 압축되지 않은 상태의 데이터파일을 논리적 블록의 크기로 분할했을 때의 할당된 번호이다. 각 블록들은 압축된 상태로 데이터파일에 존재하게 된다. 이때 압축된 블록이 데이터파일내의 연속적인 공간에 저장되지 못하고 분할되어 저장된 경우 그에 대한 처리를 위해 링크 포인터와 크기값을 사용하게 된다. 데이터 파일내에 하나의 블록이 분할 저장되는 경우는 다음과 같은 이유로 발생하게 된다. 압축된 블록에 포함된 데이터가 DBMS에 의해 갱신되어 다시 쓰기를 수행할 때 CFM에 의해 해당 데이터가 포함된 블록은 다시 압축되어 데이터 파일내에 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 기존위치가 아닌 파일내 새로운 공간에 저장되어야하고, 압축된 블록이 존재하던 기존위치는 새로운 압축 블록의 쓰기를 위해 저장 가능한 공간으로 표시되어야한다. 이렇게 파일내부에 발생하는 저장가능공간의 정보를 CFM은 데이터파일내부의 자유영역정보(Free Space Information)로 가지고 있게 된다. 새롭게 압축된 블록에 대한 쓰기를 진행할 때 이 자유영역정보를 이용해서 처음 발견된 자유영역에 압축된 블록을 저장하게 된다. 이때 자유영역이 새롭게 압축된 블록보다 그 크기가 작은 경우 압축된 블록은 분할되어 저장하게 되며 이때 블록의 끝에는 분할된 다음 데이터가 저장될 위치와 그 크기를 추가하여 데이터 손실을 방지 저장하게 된다. 이렇게 CFM은 매핑정보와 자유영역정보를 활용해 DBMS가 요구하는 데이터에 대한 검색과 저장이 가능하다.

아래 표2는 DBMS의 데이터 읽기 쓰기 명령에 대한 CFM의 기본 처리절차를 나타내었다.

DBMS의 데이터읽기 요청	DBMS의 데이터쓰기 요청
1) 요청된 데이터를 포함하는 압축된 블록 위치 검색 2) 검색된 위치의 블록 읽기 3) 읽어진 블록의 압축해제 4) 데이터 DBMS에 반환	1) 요청된 데이터 압축 2) 저장 가능한 자유영역 검색 3) 검색된 자유영역에 데이터 저장 4) 사용된 자유영역 삭제 5) 매핑정보 갱신 6) 자유영역정보 갱신 (데이터 갱신시)

<표 2> CFM의 읽기/쓰기 기본 처리절차

또한 CFM은 플래시메모리의 페이지크기에 배수이며 DBMS에서 데이터 처리를 위해 설정된 세그먼트 혹은 페이지 크기로 정의한 블록크기로 데이터를 압축하고, 압축 해제를 수행하도록 설계하였다. 이는 CFM이 DBMS의 페이지 혹은 세그먼트 크기로 파일 시스템과의 데이터처리를 수행함에 있어 플래시메모리의 데이터처리 단위인 페이지를 고려함으로써 전체 시스템의 친화성이 높이고, 이러한 친화성의 증가는 전체 성능에 있어 이점으로 작용하기 때문이다. 반면 이러한 방식에서 압축된 데이터가 파일 내에서 분할 저장되어 단편화가 발생될 수 있다는 단점을 가지게 된다. 따라서 이러한 단편화를 최소화 하고 보다 효율적인 관리를 위해서 CFM의 압축데이터 관리 기법 또한 필요하게 되었다.

4.2 CFM의 압축데이터 관리 기법

이 논문에서는 단편화를 막기 위한 방법으로 정적관리기법을 제시하고 있다.

정적관리기법은 데이터베이스 파일을 고정된 슬롯크기로 나누어 압축된 데이터를 저장하는 고정 분할 방식이다. 즉, 파일의 내부를 같은 크기의 논리적 슬롯으로 분할하여 압축된 데이터블록을 저장할 때 이 슬롯에 저장하는 방식을 사용한다.

CFM을 사용함에 있어 DBMS는 압축되지 않은 상태의 파일에 접근하는 기존의 방식을 그대로 사용할 수 있다. DBMS는 데이터파일이 압축되지 않은 상태로 존재한다고 여긴다. DBMS가 압축되지 않은 데이터파일에서 논리적 블록의 처리를 요구할 때 CFM은 매핑정보를 이용하여 실제 데이터가 압축된 Real File에서 해당 블록을 찾아 응답 처리 하게 되는 것이다. CFM의 정적관리기법에서 매핑 정보에는 블록번호, 슬롯번호, 압축된 블록의 사이즈정보가 포함되며, 자유영역정보에는 슬롯번호와 슬롯 개수를 포함하게 된다. 자유영역정보가 없는 경우 새롭게 생성된 압축블록은 파일의 끝에 추가된다. 정적관리 기법에서 전체 성능은 슬롯의 크기를 어떻게 정하느냐에 따라 크게 좌우되어질 것이다. 슬롯크기의 설정은 데이터압축률에 따라 슬롯의 크기를 결정할 수 있는 시스템 변수로 설정할 수 있다. 데이터 압축률은 각 단계별로 구분하여 사용할 수 있으며 압축에 있어서 압축률이 높을수록 압축과 해제에 많은 비용이 소모되므로 압축률과 메모리 효율에 따른 적정 값을 시스템 변수로 설정하여 사용할 수 있도록 설계하였다.

또한 이 정적관리기법으로 슬롯의 크기보다 압축된 블록의 크기가 큰 경우 무조건 붙여서 연속되는 자리에 두는 방식과, 빈 공간이 떨어져 있어도 앞 쪽부터 빈 공간을 채워가는 방식 두 가지로 설계하였다.

(1)에서는 압축된 블록의 크기가 큰 경우 무조건 붙여서 연속되는 자리에 두는 방식인 Continuous Writing에 대해 기술하고, (2)에서는 크기에 관련없이 앞쪽부터 빈 슬롯을 채워가는 방식인 Minimizing Data File Space에 대해 기술한다.

(1) Continuous Writing 방식

Continuous Writing 방식은 CFM이 데이터 쓰기의 요청을 받았을 때 그 데이터의 압축된 블록이 슬롯의 크기보다 큰 경우 두개 이상의 슬롯을 사용하게 되는데 이 때 무조건 연속되는 슬롯을 찾아 쓰게 된다.

다음은 Continuous Writing 방식을 사용한 CFM의 쓰기 절차이다.

```
Step1 : DBMS의 데이터 Page 쓰기요청 및 초기 처리
  1) 쓰기 요청된 페이지의 페이지번호를 계산
     Page Number = file pointer of file descriptor / size of a page
  2) Page관련 Mapping정보가 있는지 확인
     있으면 따로 저장
Step2 : Page의 압축과 압축된 블록의 저장
  1) Page의 압축
  2) 자유영역정보에서 압축된 블록의 크기보다 큰 공간 검색
  3) 찾아진 빈 공간에 압축된 블록 저장
Step3 : 블록관리 정보 갱신
  1) 자유영역정보 update
     자유영역정보에서 사용된 빈 공간 제거
  2) Mapping Information update
     DBMS가 요청한 데이터의 Page#에 해당하는
     slot #와 Size 정보 변경
  3) Step1 -> 2)에서 정보가 있었다면 그 정보를
     자유영역정보에 추가
```

이러한 방식을 사용함으로써 얻어지는 장점으로는 압축된 데이터의 크기가 일정한 경우 매우 효과적이라는 것과 블록 내에 사용되지 않은 부분에 대한 관리를 무시함으로써 세밀한 관리가 필요하지 않아 관리 및 처리가 간편하다는 정적관리기법의 장점과, Page 읽기/쓰기에서 연속되는 슬롯에 읽기/쓰기 때문에 한번의 읽기/쓰기만을 하게 되는 장점이 있다.

(2) Minimizing Data File Space 방식

Minimizing Data File Space 방식은 Continuous Writing 방식과 다르게 데이터 쓰기 요청을 받았을 때 앞쪽부터 빈공간을 차례로 채워가는 방식이다. 연속된 공간에 써야 할 데이터가 연속된 공간을 만난다면 그곳에 연속적으로 쓸 수 있지만 그렇지 않다면 먼저 빈공간의 크기만큼 쓰고 나머지는 다음 빈공간에 쓰게된다.

다음은 Minimizing Data File Space 방식을 사용한 CFM의 쓰기 절차이다.

Step1 : DBMS의 데이터 Page 쓰기요청 및 초기 처리
 1) 쓰기 요청된 페이지의 페이지번호를 계산
  Page Number = $\frac{file\ pointer\ of\ file\ descriptor}{size\ of\ a\ page}$
 2) Page관련 Mapping정보가 있는지 확인
  있으면 따로 저장
Step2 : Page의 압축과 압축된 블록의 저장
 1) Page의 압축
 2) first_location = 자유영역정보에서 빈 슬롯
 3) if (size of compressed block <= size of slot)
  ① LF(Link Flag), LP(Link Pointer)값 "0"으로 설정
  ② 찾아진 빈 슬롯에 (LF+LP+압축된 블록) 저장
  else
  ③ second_location=자유영역정보에서 다음 빈 슬롯
  ④ LF=1, LP=second_location
  ⑤ first_location에
  (LF+LP+슬롯크기만큼 분할된 압축된 블록) 저장
  ⑥ first_location = second_location
  ⑦ goto (Step2 -> 3).)
Step3 : 블록관리 정보 갱신
 1) 자유영역정보 update
  자유영역정보에서 사용된 Slot# 제거
 2) Mapping Information update
  DBMS가 요청한 데이터의 Page#에 해당하는
  slot #와 Size 정보 변경
 3) Step1 -> 2)에서 정보가 있었다면 그 정보를
  자유영역정보에 추가

이러한 방식을 사용함으로써 얻어지는 장점으로는 앞서 설명한 정적관리기법의 장점인 압축된 데이터의 크기가 일정한 경우 매우 효과적이라는 것과 블록 내에 사용되지 않은 부분에 대한 관리를 무시함으로써 세밀한 관리가 필요하지 않아 관리 및 처리가 간편하다는 것과 함께, 앞쪽에 있는 빈 공간부터 채워나감으로써 공간효율성이 높아지는 장점이 있다.

(3) Continuous Writing방식과 Minimizing Data File Space방식의 비교

앞서 본 논문에서 제안한 Continuous Writing방식과 Minimizing Data File Space방식은 장단점을 가진다.
<표 3>은 Continuous Writing방식과 Minimizing Data File Space방식의 장단점을 나타낸 것이다.

<표 3> Continuous Writing방식과 Minimizing Data File Space방식의 장단점 비교

	Continuous Writing방식	Minimizing Data File Space방식
장점	-한번의 파일 읽기/쓰기만 실행한다.	-파일을 효율적으로 사용한다.
단점	-단편화된 연속된 공간은 합병이 필요하다. -사용하지 않는 빈공간이 생길 수 있다	-나눠진 블록의 개수만큼 읽기/쓰기를 실행한다.

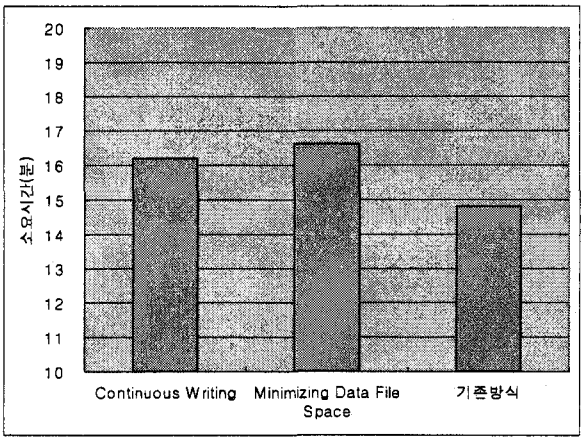
<표 3>에서 Continuous Writing의 장점은 Minimizing Data File Space의 단점이 되고, Minimizing Data File Space의 장점은 Continuous Writing의 단점이 되는 것을 보여준다. 이러한 특성은 환경에 따라 장점을 부각시킬 수 있는 기법을 사용해야 됨을 보여주는 것이다.

Continuous Writing방식은 읽기/쓰기 연산에서 한번의 파일 읽기/쓰기만으로 처리된다. 이것은 파일 읽기/쓰기의 초기화 시간이 있는 것을 감안한다면 시간면에서 좋은 효과를 얻을 수 있을 것이다. 하지만 이 방식은 쓸 블록보다 큰 빈공간을 찾는데, 이것으로 인해 사용하지 않는 빈공간이 생길 수도 있게된다. 그리고 슬롯의 정보가 아니라 빈공간의 정보를 가지고 있어야 되기 때문에 단편화된 연속된 공간은 합병된 빈공간 정보로 가지고 있어야 한다. 이에 반해 Minimizing Data File Space방식은 파일의 앞쪽 빈공간부터 쓰기 때문에 차례로 채워진다. 이것은 파일을 효율적으로 사용할 수 있게 해준다. 하지만 블록이 연속적으로 쓰인다는 보장이 없기 때문에 두개의 슬롯 이상에 걸쳐 있는 블록은 항상 파일 읽기/쓰기를 한번에 할 수 없게 된다.

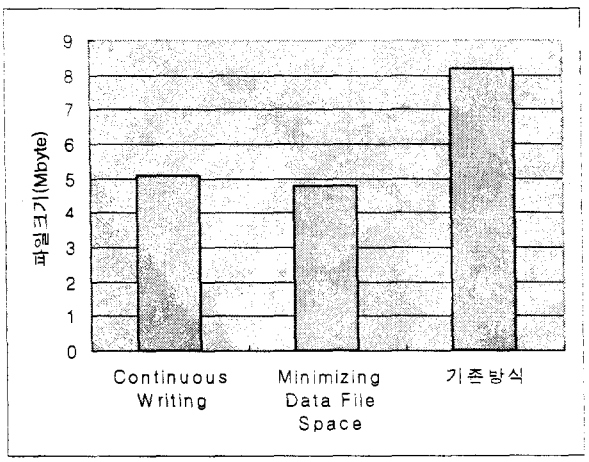
이러한 장단점을 바탕으로 고려해 볼 때 Continuous Writing방식은 파일의 낭비를 감수하고도 좀더 나은 속도를 필요로 하는 환경에 적합하고, Minimizing Data File Space방식은 속도보다는 파일의 효율적인 사용을 필요로 하는 환경에 적합하다.

5. 성능 평가

제안하는 CFM을 평가하기 위하여 이엠웨어(주)의 MobileLite-Linux 환경을 사용하였다. 이 실험을 통해 기존 DBMS와 CFM의 차이를 확인하고자 하였다.
사용한 테스트 시나리오는 MobileLite를 테스트 하기위한 프로그램으로써 여러 가지 SQL문을 돌려서 결과값을 확인하는 프로그램이다. 이 프로그램의 소요시간과 이 프로그램이 실행된 후 발생된 파일을 비교하여 보았다.
그림 6에서 나타내는 것과 같이 기존방식보다 CFM방식이 시간이 조금 더 걸리는 것으로 보여진다. 이것은 CFM방식이 I/O 크기는 줄어들지만 압축시간과 파일 관리시간이 추가적으로 필요하기 때문이다. 하지만 이 시간은 파일 크기에서 50~60%의 압축률을 보이는 것과 비교하면 CFM의 우수함을 알 수 있다.



(그림 6) 기존방식과 제안된 CFM 방식의 소요시간 비교



(그림 7) 기존방식과 제안된 CFM 방식의 파일크기 비교

6. 결론

유비쿼터스 시대를 맞아 모바일 정보기기가 보편화 되어 가고 있는 시점에서 보다 많은 정보를 저장하고 처리할 수 있는 모바일 정보기기의 필요성이 증대되고 있다. 하지만 비용과 크기의 문제로 인해 저장공간을 늘리기에는 한계가 있다. 이러한 요구를 만족시키기 위한 방법으로 본 논문에서는 플래시 메모리에 적합한, 압축을 이용한 DBMS의 효율적인 데이터 관리를 제안하였다. 이 데이터 관리는 또한 Continuous Writing방식과 Minimizing Data File Space방식으로 나누어, 적용분야의 특성에 따라 더 효율적인 관리를 할 수 있도록 설계하였다.

이 방법은 데이터베이스가 압축되지 않은 상태로 저장되는 현재 상황을 고려해 볼 때 획기적인 저장공간의 절약을 가져온다. 이 관리기법은 모바일에서만 국한되지 않고 일반적인 DBMS에도 사용될 수 있다. 이러한 응용은 계속해서 늘어나는 정보를 효율적으로 관리할 수 있게 한다.

또한 이 방법은 디스크 I/O를 줄여준다. 이것은 제한적인 수명을 가지는 플래시 메모리의 수명을 연장시키는 효과를 가져온다.

6. 참 고 문 헌

[1] 임근수, 반효경, 고건, "NAND형 플래시메모리를 위한 플래시 압축 계층의 설계 및 성능평가", 정보과학회논문지 : 시스템 및 이론 제32권 제3-4호, 2005.

[2] 변시우, 노창배, 정명희, "휴대용 정보기기를 위한 플래시 기반 2단계 로킹기법", 한국데이타베이스학회: 정보기술과 데이타베이스 저널 제12권 4호, 2005

[3] Micron Technology, 'Small Block vs. Large Block NAND Flash Devices", Technical Note, TN-29-07, 2006

[4] Balakrishna R. Iyer, David Wilhite, 'Data Compression Support in Databases', Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994

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

[6] L. Felician and A. Gentili, 'A nearly optimal Huffman technique in the microcomputer environment', Inf. Sys. 1987

[7] G.V. Cormack. Data compression on a database system. Communication of the ACM, 28:12, 1985.

[8] D. E. Knuth, "Dynamic Huffman coding", Journal of Algorithms 6, pp163-180, 1985

[9] D. G. Severance, "A practitioner's guide to data base compression", Inf. Sys. 1983

[10] C. A. Lynch and E. B. Brownrigg, "Application of Data Compression to a Large Bibliographic Data Base", Cannes, France, 1981

[11] Peter A. Alsberg "Space and Time Saving Through Large Data Base Compression and Dynamic Restructuring," Proceedings, IEEE 63,8, 1975.

[12] S. S. Ruth and P. J. 'Keutzer, Data compression for business files", Datamation 18, 1972