모바일 압축 데이터 관리 시스템(CDMS)를 위한 데이터 가용 크기 변경 기법*

장진근° 신영재 이정화 손진현
한양대학교 컴퓨터공학과
jang798@database.hanyang.ac.kr, youngjae@database.hanyang.ac.kr,
jwlee@database.hanyang.ac.kr, jhson@hanyang.ac.kr

Technique about Data Capacity Adaptation for the Mobile Compressed Data Management System

Jinkun Jang° Youngjae Shin Jeongwha Lee Jin Hyun Son
Department of Computer Science and Engineering, Hanyang University in Ansan

요 약

휴대용 정보기기는 정보의 디지털화로 인해 많아지고 있는 디지털 정보를 처리 및 저장해야 되는 상황이 되었다. 따라서 휴대용 정보기기에서는 우수한 디지털 정보를 효과적으로 관리하기 위해 모바일 DBMS을 사용하게 되었고, 저장장치로는 플래시 메모리를 사용하고 있다. 플래시 메모리는 일반 디스크보다 고비용이기 때문에 데이터를 보다 효율적으로 저장하기 위하여 압축 데이터 관리 시스템(CDMS)과 같은 압축을 사용한 관리도 사용되고 있다. 하지만 압축을 사용하여 저장공간을 효율적으로 사용하기 위한 CDMS의 연구는 데이터베이스 파일 크기를 고정하여 관리하였다. 하지만 이것은 실제사용에 있어서는 실용적이지 못하다. 따라서 본 논문에서는 이러한 CDMS를 보완하여 데이터베이스 파일 크기를 동적으로 변환하는 기법을 제시한다.

1. 서론

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 있으며, 이로 인해 휴대용 정보기기에서는 보다 많은 정보의 생성, 처리, 저장이 가능한 다양한 응용프로그램의 사용을 요구 받고 있다[1, 2]. 이러한 환경의 변화로 휴대용 정보기기에서는 많은 데이터의 효율적인 관리 시스템이 필요로 하게 되었다. 따라서 DBMS의 필요성이 휴대용 정보기기에까지 이르고 있다. 세계적인 분석기관인 IDC (International Data Corporation)社는 2004년도에 64%이상의 휴대용 정보기기의 어플리케이션이 임베디드 데이터베이스 관리 시스템이 필요하다고 보고하였다[3].

기존 컴퓨팅 환경에서는 가격이 저렴하고 확장성이 용이한 하드디스크를 데이터 저장장치로 사용함으로 저장공간에 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 모바일 정보기기에서는 이러한 단점을 보완할 수 있는 저전력으로 장시간의 구동이 가능하며, 부피가 작고 경량으로 소음과 진동이 없으며 물리적인 충격에 강해 휴대가 용이한 플래시 메모리를 보조기억장치로 사용한다[4]. 이 플래시 메모리는 NOR형 플래시 메모리와 NAND형 플래시 메모리로 나뉜다. 모바일 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]. 이러한 필요로 인해 효율적인 데이터 압축 관리 시스템인 CDMS(Compressed Data Management System)가 제안되었다[6, 7].

하지만 이 효율적인 압축 데이터 관리 시스템(CDMS)은 DBMS가 관리하는 데이터의 효율적인 처리 및 관리를 제공하지만 몇 가지 문제를 가지고 있다. 그 문제 중 하나가 데이터베이스의 크기를 동적으로 변경할 수 없다는 것이다. 따라서 본 논문에서는 이러한 CDMS의 문제점을 확인하고 그것에 대한 해결방안을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서는 CDMS에 대한 연구를 살펴보고, 3장에서 CDMS의 문제점을 통해 본 연구의 필요성을 설명한다. 4장에서는 CDMS의 문제

* 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-20135-0).

점을 해결할 수 있는 데이터 가용크기 동적 변경 기법의 구조와 기능을 설명하고 5장에서 실험을 통하여 평가한다. 마지막으로 6장에서는 논문의 결론을 맺는다.

2. 관련 연구

이 장에서는 CDMS의 전체적인 구조 및 관리기법을 알아본다. 2.1절에서는 CDMS의 전체적인 구조를[6, 7] 기술하고, 2.2절과 2.3에서 Dynamic 관리기법[7]과 Static 관리기법[6]을 각각 기술한다.

2.1 CDMS 구조

CDMS는 기존 DBMS의 변경 최소화를 위해 DBMS의 스토리지 관리자와 파일 시스템 사이에 들어가게 된다.

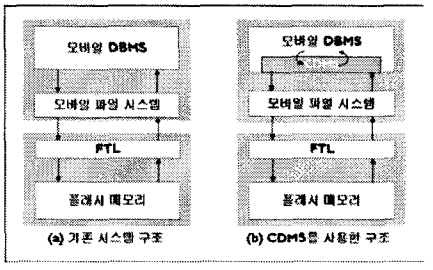

그림 1. 시스템 구조

그림 1은 기존의 시스템 구조와 CDMS를 사용한 구조를 나타낸다. (a)에서 보는 바와 같이 기존의 시스템은 모바일 DBMS에서 데이터베이스 파일을 관리하기 위해, 데이터를 읽고 쓰는 것에 대한 요청을 파일 시스템에 직접 하게 된다[5]. 하지만 이 논문에서 제안하는 CDMS를 이용한 새로운 구조는 (b)와 같이 DBMS의 데이터베이스 파일에 대한 접근 및 처리요청을 CDMS에 의해 이루어지도록 한다. 모바일 DBMS는 기존의 파일시스템에 요청한 것과 동일하게 CDMS에게 요청하면 CDMS는 추가적인 처리를 자체적으로 함으로써 압축을 적용시켜 파일 시스템과 데이터 교환을 하여 DBMS의 요청을 처리한다. 따라서 모바일 DBMS의 데이터베이스 파일은 압축된 블록들로 구성되지만 DBMS는 압축되지 않은 데이터베이스 파일에 접근하는 방식과 동일한 방식으로 CDMS에 요청하면 된다.

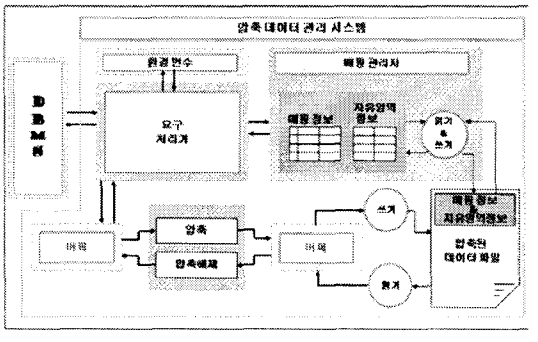

그림 2. CDMS 구조

CDMS의 전체적인 구조도인 그림 2에 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 데이터의 무결성을 보장하기 위한 처리로 인해 발생한다. 구체적으로 설명하면 압축된 블록에 포함된 데이터가 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 기존위치가 아닌 파일 내부의 새로운 공간에 저장되어야 하고, 압축된 블록이 존재하던 기존위치는 새로운 압축 블록의 쓰기를 위해 저장 가능한 공간으로 표시되어야한다. 이렇게 파일내부에 발생하는 저장가능공간의 정보를 자유영역정보로 가지고 있게 된다.

2.2 Dynamic 관리기법

CDMS의 압축 데이터 관리기법 중 하나인 Dynamic 관리기법은 압축된 데이터의 크기에 따라 처리하는 방식이다. 이 Dynamic 관리기법은 세부적으로 2가지로 다시 나눠진다. 먼저 첫 번째는 압축된 블록이 나눠지지 않고 한곳에 저장되는 Dynamic Continuous Writing 방식이다. 이 방식은 CDMS가 데이터 쓰기의 요청을 받았을 때, 압축된 데이터를 저장하기 위한 공간으로 블록 전체가 모두 들어갈 수 있는 크기의 공간이면서, 또한 외부단편화를 줄이기 위해 다른 공간보다 잘 맞는 공간을 선택하여 데이터를 쓰게 된다. 두 번째는 빈 공간이 압축된 블록보다 작더라도 앞쪽의 빈 공간부터 채우고 나머지는 또 다른 빈 공간에 넣어서 나눠진 블록을 링크로 연결하는 Dynamic Minimizing Data File Space 방식이다. 이 방식은 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 빈공간에 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 데이터의 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.

2.3 Static 관리기법

CDMS의 다른 하나의 관리기법은 Static 관리기법이다. 이 Static 관리기법은 각각 다른 크기의 빈공간이 무분별하게 생기는 것을 막기 위한 방법으로 가상의 슬롯을 이용하고 있다. 이러한 가상의 슬롯을 이용함으로 데이터 파일을 논리적으로 고정된 슬롯 크기로 나누어 압축된 데이터를 저장하게 된다. 즉, 파일의 내부를 같은 크기의 슬롯으로 논리적으로 분할하여 압축된 블록을 저장할 때 이 슬롯에 저장하는 방식을 사용한다. Static 관리기법을 사용함으로 슬롯 안에 내부적인 빈공간이 생기지만 데이터의 처리가 슬롯단위로 가능해져서 관리가 편리하면서 효율적이게 된다. Static 기법 또한 세부적으로 2가지로 나눠진다. 첫 번째는 압축된 블록이 떨어져서 저장되지 않고, 연속되는 슬롯에 저장하는 Static Continuous Writing 방식이다. 이 방식은 CDMS가 데이터 쓰기의 요청을 받았을 때 그 데이터의 압축된 블록이 슬롯의 크기보다 큰 경우 두 개 이상의 슬롯을 사용하게 되는데 이 때 무조건 연속되는 슬롯을 찾아 쓰게 된다. 연속되는 슬롯을 찾을 때는 파일의 제일 앞쪽부터 찾게 된다. 두 번째는 빈 슬롯이 압축된 블록보다 작더라도 앞쪽의 빈 슬롯부터 채우는 Static Minimizing Data File Space 방식이다. 이 방식은 Static Continuous Writing 방식과 다르게 데이터 쓰기 요청을 받았을 때 앞쪽부터 빈 공간을 차례로 채워가는 방식이다. 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. 이 때 나눠진 데이터는 링크정보로 연결하게 된다.

3. CDMS의 문제점

기존의 CDMS는 매핑정보와 자유영역정보를 저장하기 위해 Mapping 테이블과 Free Space 테이블을 사용한다. 이것은 읽기 작업을 원활하게 하기 위해 메인 메모리(휘발성 메모리)에 저장된다. 하지만 매핑정보와 자유영역정보를 유지하기 위해서는 비휘발성 메모리에 저장하여야 하기 때문에 CDMS가 운영하는 파일에 저장되지 않으면 안 된다. 이러한 이유로 CDMS가 운영하는 데이터 파일에는 Mapping 테이블과 Free Space 테이블 등의 헤더정보를 유지하고 있다.

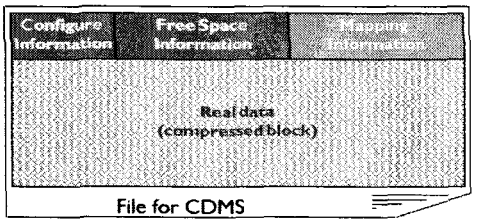

그림 3. CDMS가 운영하는 파일의 구성

그림 3은 CDMS가 운영하는 데이터 파일의 구성을 보여주는 그림이다. 이 그림에서 볼 수 있듯이 CDMS가 운영하는 데이터 파일 안에 자유영역정보와 매핑정보가 존재한다. 이것은 Free Space 테이블과 Mapping 테이블을 파일에 저장한 것이다. 이러한 구조에서 CDMS가 데이터베이스의 크기를 동적으로 변경 하는 것은 불가능하다. 즉 파일에서는 모든 정보가 순차적으로 저장되기 때문에 중간의 어떤 정보의 길이를 늘리고 줄이는 것이 쉽지 않다. 따라서 기존 연구에서는 DBMS가 관리하는 파일의 최대 크기가 정해져 있다고 가정하여 동적 변경을 고려하지 않았다.

이러한 CDMS의 운영 방식은 운영환경이 동적으로 변화하는 곳에서는 DBMS 사용에 제약을 줄 수 있다. 다시 말해 기기 내에 DBMS의 사용을 많이 하는 프로그램이 설치되어서 많은 데이터가 저장될 수 있다. 하지만 이때 기존 CDMS의 운영방식은 데이터를 저장할 공간이 없다면 저장공간의 부족을 사용자에게 통지하고 데이터베이스에 있는 다른 데이터를 지워서 공간을 확보해 줄 것을 요청할 것이다. 따라서 본 논문에서는 CDMS가 보다 유동적인 운영이 가능하도록 CDMS가 운영하는 데이터 파일이 처리하는 데이터베이스의 최대 크기를 동적으로 변경할 수 있는 기법을 제안한다.

4. 데이터 가용크기 동적 변경 기법

이 장에서는 기존 CDMS의 운영 방식으로 인해 데이터 가용크기가 고정되어 있던 문제점을 해결하는 데이터 가용크기 동적 변경 기법에 대해 기술한다.

데이터 가용크기 동적 변경은 헤더정보의 동적 변경으로 가능하게 된다. 이때 변경되는 것은 단순히 내용의 변화뿐 아니라 크기의 변화도 포함한다. 따라서 본 논문에서는 헤더정보와 헤더크기의 동적 변경을 위한 기법을 제안한다. 또한 어떠한 비정상 종료에도 데이터 무결성을 지키면서 데이터 가용크기 변경을 수행하도록 고려하였다.

데이터 가용크기의 변경은 크기가 늘어나는 경우와 줄어드는 경우 두 경우를 고려해야 한다. 먼저 데이터베이스의 최대 크기가 줄어드는 경우는 헤더크기를 줄이지 않고, 줄었다는 정보를 유지함으로써 해결한다. 따라서 이것을 위해 2개의 환경변수가 저장되게 된다. 저장되게 되는 환경변수는 실제로 줄어든 매핑정보와 자유영역정보의 크기를 줄지 않은 것으로 여기게 하기 위해 매핑정보의 시작점 주소(Map_add)과 Data영역의 시작점의 주소(Data_add)가 된다. 이 두 개의 환경변수를 통해 매핑정보와 자유영역정보가 줄어든 만큼 공간 낭비는 있지만 이것은 극히 작고, 헤더정보의 변경으로 생길 수 있는 시스템의 저하 및 데이터베이스의 손상을 미연에 방지하게 된다. 그림 4는 데이터 가용크기가 줄어드는 경우를 그림으로 나타낸 것이다.

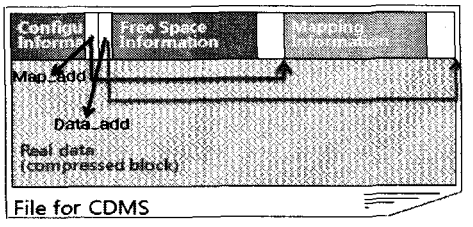

그림 4. 데이터 가용크기가 줄어드는 경우

다음으로는 데이터 가용크기가 늘어나는 경우는 헤더 정보의 크기가 늘어날 수 있도록 데이터 영역의 앞쪽 부분을 헤더정보 영역으로 변경 시켜 주어야 한다. 그림 4에서 나타나 있듯이 변경시키기 위해서 헤더정보가 늘어나는 크기만큼 앞부분의 데이터를 파일의 끝으로 이동시켜 데이터가 위치해 있던 부분은 헤더정보가 사용할 수 있게 되었다. 하지만 공간만 비워줌으로 데이터 가용크기 변경이 수행되는 것은 아니다. 헤더가 늘어난 만큼 데이터 영역의 시작점이 변경되었기 때문이다. 따라서 데이터 영역의 논리적인 주소체계로 구성된 매핑정보와 자유영역정보의 변경이 불가피하게 된다. 따라서 이러한 데이터 영역의 주소의 변화를 반영하기 위해 본 논문에서는 2가지 방법을 제안한다. 그 첫 번째 방법으로 매핑정보와 자유영역정보의 주소를 모두 바꾸는 주소변경정책을 4.1절에서 서술하고, 매핑정보와 자유영역정보의 주소를 변경하지 않고 변경되었다는 정보를 따로 가지고 있음으로 똑같은 효과를 낼 수 있는 주소변환정책을 4.2절에서 서술한다. 그리고 4.3절에서 이러한 두 정책을 비교하여 차이점 서술한다.

4.1 주소변경정책

데이터베이스의 최대 크기를 변경하기 위해서는 헤더정보의 변경이 필요하다. 특히 데이터베이스의 최대크기가 늘어날 경우 헤더정보가 증가, 데이터영역 앞부분을 사용하게 된다. 또한 데이터영역의 시작점이 이동되게 되어, 데이터 영역의 논리적인 위치정보로 자신의 위치를 알리고 있는 각각의 데이터는 자신의 위치정보를 변경된 데이터 영역의 논리주소로 변경하여야 한다.

여기서는 그 변경방법 중 매핑정보와 자유영역정보의 모든 주소정보를 변경된 논리주소로 바꾸는 주소교체정책을 기술한다. 다음은 주소교체정책의 절차를 의사코드로 나타낸 것이다.

```
단계 1 : 데이터 가용크기 변경의 증가 또는 감소 확인
  1) 크기 감소시
     감소영역에 데이터가 존재하면 비정상 종료(오류 반환)
     새로운 데이터 가용크기를 파일의 환경변수에 기록 후
     정상 종료
  2) 크기 증가시 이후 과정을 수행

단계 2 : 헤더정보 증가 크기 계산 및 데이터 영역의 이동
  1) 증가 크기
     = (DB최대크기 증가량 × 페이지에 해당되는 매핑정보 크기 × 2) / 페이지 크기
  2) 증가 크기에 해당되는 데이터들을 파일 끝으로 이동

단계 3 : 이동된 데이터 영역에 맞는 주소정보로 헤더정보 갱신
  1) 갱신된 헤더정보의 일부를 파일 끝에 기록
     ① 새로운 DATA_MAX_SIZE 기록
     ② 매핑정보와 자유영역정보에 포함된 주소의 갱신 및 크기
        변경후 기록
     ③ 앞쪽의 헤더정보에 있는 매핑정보 시작위치를 새로운 매핑
        정보 위치로 변경
  2) 파일 뒤쪽에 기록된 내용을 다시 앞 헤더정보로 이동
     ① 새로운 DATA_MAX_SIZE 갱신
     ② 변경된 매핑정보와 자유영역정보로 헤더정보 갱신
     ③ 매핑정보 시작위치를 갱신한 매핑정보 위치로 이동
```

절차에서 볼 수 있듯이 처음 데이터 가용크기의 증가 및 감소를 확인한 후 감소는 데이터 가용크기 정보 환경변수에 기록하는 것으로 수행을 완료하게 된다. 증가는 헤더정보의 증가 크기를 계산하여 데이터 영역의 앞부분을 헤더영역으로 사용해야 된다. 이 때 헤더정보 중 늘어나게 되는 정보는 매핑정보와 자유영역정보가 된다. 그리고 매핑정보와 자유영역정보는 데이터의 시작과 끝을 나타내는 주소정보의 집합으로 같은 크기를 가지게 된다. 따라서 단계 2의 1)에서 보인 식을 사용하여 헤더정보의 증가 크기를 계산할 수 있다. 이 후 계산된 크기에 해당되는 데이터를 파일의 끝으로 옮기게 되어 헤더영역이 늘어날 수 있는 공간을 확보한다. 이렇게 확보된 영역은 헤더영역으로 편입시키고, 기존 데이터 영역은 헤더영역만큼 뒤쪽으로 이동하게 된다. 데이터 영역의 이동으로 기존 데이터에 대한 논리주소 값 또한 변경되어야 한다. 따라서 이것을 처리하기 위한 과정이 단계 3이 된다. 단계 3은 처리과정 중에 비정상 종료의 위험을 막기 위해 파일 뒤쪽에 변경된 정보를 쓴 후에 다시 옮겨오게 된다.(그림 5) 또한 단계 3에서 나타나 있는 절차의 순서도 비정상 종료의 위험을 막기 위한 방법이다. 비정상 종료의 대처는 특히 환경변수로 추가한 매핑정보의 시작점 주소가 큰 역할을 하게 된다. 이것은 뒤쪽에 쓰는 변경된 헤더정보가 이상없이 완벽하게 쓰여졌다는 것을 알려주고 다 시 앞쪽으로 이동할 때 또한 이상없이 이동되었다는 것을 알려주게 되어 헤더정보가 원자성을 띄도록 하게 한다[8].

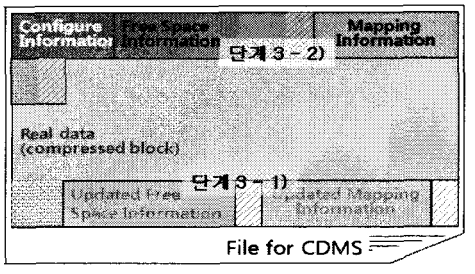

그림 5. 완벽한 헤더정보 갱신을 위한 방법

이러한 일련의 과정으로 헤더정보의 크기를 늘리고, 헤더정보에 있는 매핑정보와 자유영역정보를 변경했지만 CDMS의 관리방법 중 Minimizing의 경우 각 데이터가 가지는 Link_flag까지 변경하지 못하는 문제가 있다. 따라서 이 주소변경정책은 CDMS의 Continuous 방식만을 고려한다.

주소변경정책으로 데이터 가용크기 변경을 하지 못하는 Minimizing 방식의 경우는 주소변환정책을 사용하여 변경을 수행할 수 있다. 주소변환정책은 다음절에서 상세하게 기술한다.

4.2 주소변환정책

주소변환정책은 기본적으로 주소변경정책을 기본으로 한다. 차이점은 매핑정보와 자유영역정보를 주소변경정책에서는 모든 주소정보를 변경했지만 주소변환정책에서는 정보를 직접 고치는 것이 아니라 주소가 이동하였다

는 정보를 유지하면서 추후 주소 연산 시 고려하는 것이다. 따라서 환경변수에 주소 이동에 대한 정보를 가지고 있어야 한다. 이런 방식을 사용함으로 CDMS의 Miniming 방식이 주소정보를 매핑정보와 자유영역정보 외에도 데이터 자체에 링크정보로 가지고 있는 특성까지 모두 고려하게 된다. 다음은 주소변환정책 절차이다.

```
단계 1 : 데이터 가용크기 변경의 증가 또는 감소 확인
  1) 크기 감소시
     감소영역에 데이터가 존재하면 비정상 종료(오류 반환)
     새로운 데이터 가용크기를 파일의 환경변수에 기록 후
     정상 종료
  2) 크기 증가시 이후 과정을 수행

단계 2 : 헤더정보 증가 크기 계산 및 데이터 영역의 이동
  1) 증가 크기
     = (DB최대크기 증가량 × 페이지에 해당되는 매핑정보 크기 × 2) / 페이지 크기
  2) 증가 크기에 해당되는 데이터들을 파일 끝으로 이동

단계 3 : 이동된 데이터 영역에 맞는 주소정보로 헤더정보 갱신
  1) 갱신된 헤더정보의 일부를 파일 끝에 기록
     ① 새로운 DATA_MAX_SIZE 기록
     ② 매핑정보와 자유영역정보 크기 변경 후 기록
     ③ 증가 크기에 대한 정보를 기록한다.
     ④ 앞쪽의 헤더정보에 있는 매핑정보 시작위치를 새로운 매핑
        정보 위치로 변경
  2) 파일 뒤쪽에 기록된 내용을 다시 앞 헤더정보로 이동
     ① 새로운 DATA_MAX_SIZE 갱신
     ② 늘어난 매핑정보와 자유영역정보로 헤더정보 갱신
     ③ 증가 크기에 대한 정보를 갱신한다.
     ③ 매핑정보 시작위치를 갱신한 매핑정보 위치로 이동
```

절차에서 볼 수 있듯이 크기 감소에 따른 처리나 증가 크기에 대한 계산과 데이터의 이동은 주소변경정책과 동일한 것을 볼 수 있다. 즉 단계 2까지는 주소변경정책과 차이점이 없고 단계 3에서 다른 방식으로 처리되는 것을 볼 수 있다. 주소변환정책은 단계 3에서 증가 크기에 대한 정보를 유지하기 위해 증가 크기를 완전하게 저장하기 위한 작업이 추가된다. 이렇게 저장된 증가 크기에 대한 정보를 통해 추후 주소 연산시 사용하게 된다.

주소변환정책의 수행과정에서 비정상 종료에 의한 문제를 방지하기 위해 주소변경정책과 동일하게 매핑정보 시작위치 주소를 사용하였다. 따라서 증가크기에 대한 정보를 포함한 헤더정보가 원자성을 띄게 된다[8].

4.3 데이터 가용크기 동적 변경 기법 비교

데이터 가용크기 동적 변경 기법의 두 가지 정책인 주소변경정책과 주소변환정책은 주소 처리의 차이로 인해 고유의 특성을 가지게 된다. 본 절에서는 그러한 특성을 알아보고 두 정책의 차이점을 확인한다.

주소변경정책은 이름에서도 알 수 있듯이 주소를 변경한다. 따라서 매핑정보와 자유영역정보에 포함된 모든 주소 정보가 변경된다. 모든 주소 정보가 이동된 데이터 영역에 맞게 모두 변경되기 때문에 매핑정보와 자유영역정보의 주소가 모두 독립적으로 사용할 수 있다. 하지만 이로 인해 변경에 대한 오버헤드가 발생할 수 있다. 또한 하나의 데이터를 여러 개로 나누어 저장하고 각 데이터의 링크 정보를 데이터 자체 내에 유지하고 있는 Minimizing Data File Space 경우 주소변경정책을 사용하여 모든 링크정보를 찾아 수정하는 것은 심각한 시스템 오버헤드가 발생할 수 있게 된다.

반면에 주소변환정책은 주소 정보를 변경하지 않고 변환하여 사용하기 때문에 Minimizing 방식의 링크정보를 포함하여 모든 주소 정보 변경으로 발생할 수 있는 오버헤드가 없다. 하지만 데이터 가용크기로 증가된 헤더정보의 증가 크기 정보에 의존성이 강해 증가 크기 정보가 손실되거나 잘못된 연산으로 변경되었을 때 이후 정보는 쓰지 못하게 된다.

이러한 각각의 특성으로 인해 주소변경정책은 Continuous 방식에 주소변환정책은 Minimizing 방식에 적합하다. 표 1는 위에 설명한 각 정책의 장단점을 표로 나타낸 것이다.

구분	주소변경정책	주소변환정책
장점	주소 정보가 독립적 Continuous 방식에 적합	주소 변경에 따른 오버헤드 없음 Minimizing 방식에 적합
단점	주소 변경에 따른 오버헤드 발생 Minimizing 방식에 부적합	주소 정보가 추가적인 정보에 의존적

표 1. 데이터 가용크기 변경 정책 비교

5. 실험

이 장에서는 본 논문이 제안하는 데이터 가용크기 동적 변경 기법에 대한 검증을 위한 실험의 결과를 기술하고 이 기법의 소요시간 분석을 통해 이 기법의 특성과 각 정책의 특성을 살펴본다. 이러한 일련의 실험을 위하여 실험 환경은 이엠웨어(주)의 MobileLite-LINUX 환경을 사용하였다. 이 환경은 기존 CDMS 개발에 사용되었던 환경과 동일한 환경이다.

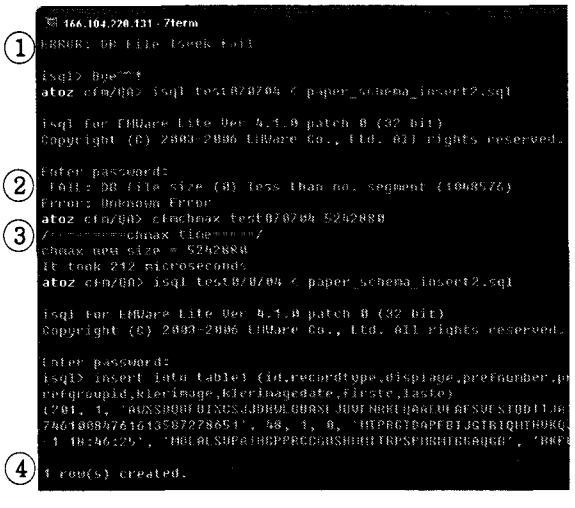

그림 6. 데이터 가용크기 동적 변경의 동작

그림 6에 나타나 있는 실험 결과는 본 논문이 제안하는 데이터 가용크기 동적 변경 기법이 구동되는 상황과 구동 성공 여부를 살펴본 것이다. 이 실험을 위해 DBMS에서 Insert문을 사용하여 계속적으로 데이터베이스에 데이터를 입력하였다. 그림 6의 ①, ②에서 보이듯이 저장 공간을 모두 사용하여 부족하게 되면 DBMS는 에러를 발생하게 된다. 이 때 그림 6의 ③에서 본 논문이 제안하는 기법으로 데이터 가용크기를 늘려주게 된다. 그럼 DBMS가 관리하는 데이터베이스의 용량이 커졌기 때문에 그림 6의 ④에서 알 수 있듯이 추가적으로 데이터를 문제없이 추가할 수 있게 된다. 따라서 이 실험을 통해 본 논문에서 제안하는 기법의 필요 환경과 기법의 검증을 할 수 있었다.

다음 실험은 이 데이터 가용크기 동적 변경의 소요시간을 확인해 본 결과이다.

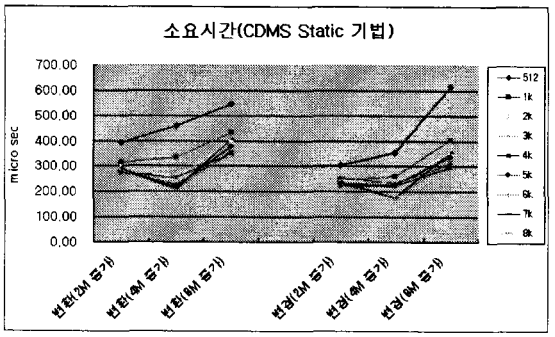

그림 7. DB 최대크기 동적 변경의 소요시간

그림 7에서 볼 수 있듯이 데이터 가용크기 동적 변경은 변화되는 크기의 차이가 크면 클수록 시간이 더 걸리는 것을 알 수 있다. 이것은 앞서 설명했듯이 크기가 늘어나는 만큼에 해당되는 Mapping 테이블과 Free Space 테이블도 늘어나기 때문에 늘어나는 부분에 포함되어 있던 기존의 데이터를 뒤로 옮긴다. 이때 옮기는 양이 변화되는 크기와 비례하기 때문에 시간 또한 영향을 받게 된다. 또한 변환이 변경보다 전반적으로 많은 시간이 걸리는 것을 알 수 있다. 이것은 CDMS의 Minimizing기법을 고려한 교환 정책의 특성상 링크되어 있는 쪼개진 데이터까지 같이 이동하기 때문이다. 또한 슬롯이 작아질 경우 늘어나는 영역에 걸치는 데이터가 많아지기 때문에 이동량이 많아져서 슬롯이 작을수록 시간이 많이 걸리게 된다.

위의 실험들을 통해 본 논문에서 제안하는 데이터 가용크기 변경을 위한 기법에 대한 검증을 할 수 있었다. 또한 실험을 통해 이 기법의 특성을 파악할 수 있었다. 이것은 앞서 기술한 데이터 가용크기 변경을 수행하기 위한 여러 가지 기술적인 부분들의 특성으로 볼 수 있다.

6. 결론

플래시 메모리의 사용으로 인하여 저장 공간의 제약을 받는 모바일 정보기기에서 저장 공간의 효율적인 관리가 필요하다. 또한 모바일 정보기기의 발전으로 기기가 처리해야 되는 정보는 많아지고 있어 DBMS의 사용이 불가피 하게 되었다. 따라서 Mobile DBMS를 위한 효율적인 압축 데이터 관리 시스템(CDMS)이 연구되어 왔다. 하지만 이 시스템은 아직까지 실용적이고 유동적인 관리가 이뤄지기 위해서는 많은 문제점을 가지고 있다. 그 중 하나의 문제점은 데이터 가용크기의 동적 변경을 하지 못하는 것이다. 따라서 본 논문에서는 데이터 가용크기의 동적 변경을 가능하게 하는 기법을 제안하였다. 이 기법은 기존 CDMS의 모든 기법을 만족할 수 있도록 2가지 정책을 사용하여 설계되었다. 본 논문에서 제안된 이 기법은 또한 실험을 통하여 검증하고 특성을 알아보았다.

현대사회는 응용프로그램같은 소프트웨어에서부터 하드웨어에 이르기까지 많은 기술들이 유동적으로 관리되는 방향으로 발전되고 있다. 이것은 한 기술로 여러 환경에서 최적화된 성능을 내기 위함이다. 본 논문이 제안하는 데이터 가용크기의 동적 변경 기법을 사용함으로 CDMS는 환경에 맞는 최적의 데이터베이스 크기로 조절이 가능하게 되었다.

7. 참고문헌

[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, 2004

[2] Mark A. Roth , Scott J. Van Horn "Database compression", ACM SIGMOD Record, Vol.22 No.3, p. 31-39, 1993

[3] S.W. Byun, C. B. Roh, M. H. Jung, "Flash-Based Two Phase Locking Scheme for Portable Computing Devices", The Korea Database Society : Journal of Information Technology Applications & Management vol 12, no 4, 2005

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

[5] 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.

[6] 신영재, 황진호, 박정업, 김학수, 이승미, 손진현 "Mobile DBMS를 위한 효율적인 압축 데이터 관리 시스템", 2006년도 한국정보과학회 가을 학술대회, 33권, 2(C)호, pp.6-11, 세종대학교, 20 Oct, 2006

[7] 신영재, 장진근, 이정화, 김학수, 박찬희, 손진현 "모바일 DBMS를 위한 효율적인 압축 데이터 동적 관리 시스템", 제 27회 한국정보처리학회 춘계학술발표대회, 14권, 1호, pp.42-45, 경원대학교, 12 May, 2007

[8] Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Database Systems: The Complete Book, Prentice Hall PTR, Upper Saddle River, NJ, 2002