産業技術硏究(江原大學校 産業技術硏究所 論文集) 第 16 輯 (1996)
Research Report. The Institute of Industrial Technology, Kangwon Nat'l Univ., Korea, Vol. 16, 1996

Z-순서화 기법을 이용한 계층 그리드 화일의 일괄 구성

김 상 욱*

Batch-Constructing of Multilevel Grid Files Using the Z-ordering Scheme

Sang-Wook Kim*

ABSTRACT

The multilevel grid file(MLGF) is a dynamic multidimensional file organization supporting multi-attribute accesses efficiently. This paper proposes new method for batch-constructing MLGFs. Our method consists of two phases. The first phase begins by relocating all the objects in order that logically adjacent objects in multidimensional domain space are clustered in one-dimensional physical space. For this, our method employs the Z-ordering scheme, which effectively maps multidimensional space into one dimensional space preserving proximity. The second phase paginates the relocated objects and creates leaf level directory entries, each of which corresponds to a object page. Simultaneously, it performs same actions on the directory entries recursively in a bottom-up fashion until the root directory fits in a page. For performance evaluation, we analyze our method in terms of the number of page accesses. The result shows the optimality of our method.

1. 서 론

다차원 동적 화일(multidimensional dynamic file)은 하나 이상의 애트리뷰트를 조건의 대상으로 하는 질의를 효과적으로 처리하기 위한 구조이며, 컴퓨터 지원 설계(computer aided design), 지리 정보 시스템(geographic information systems), 이미지 처리(image processing), 공간 데이타베이스(spatial databases)등의 응용 분야에서 널리 사용된다[Lome92]. 최근 들어, 많은 다차원 동적 화일들이 제안되어 왔으며, 이중 대표적인 것으로는 K-D 트리

* 강원대학교 정보통신공학과 전임강사

[Bent79], K-D-B 트리[Robi81], hB 트리[Lome90], LSD 트리[Henr89], EXCELL[Tamm82], 인터폴레이션을 기반으로 하는 인덱스(interpolation based indexes)[Burk83], 다차원 선형 해싱(multidimensional linear hashing)[Ouks83], 다차원 신장 해싱(multidimensional extendible hashing)[Otoo84], 다차원 디지틀 해싱(multidimensional digital hashing)[Otoo85], 균형 다차원 신장 트리(balanced multidimensional extendible hash tree)[Otoo86], 그리드 화일(grid file)[Niev84], 그리고 계층 그리드 화일(multilevel grid file)[Whan95][Whan91] 등이 있다.

본 논문에서는 다차원 동적 화일을 위한 일괄 구성(batch-construction) 기법에 관하여 논의하고자 한다. 다차원 동적 화일을 위한 일괄 구성이란 대량의 객체들을 대상으로 다차원 동적 화일을 일괄 처리 방식으로 구성하는 것을 의미한다. 기존에 제안된 다차원 동적 화일을 위한 연산 알고리즘들은 주로 하나의 객체를 대상으로 하는 검색, 삽입, 삭제 알고리즘에 국한되어 왔으며, 이러한 일괄 구성에 관해서는 국내외를 막론하고 전혀 논의된 바 없다.

물론, 별도의 일괄 구성 알고리즘의 개발 노력 없이 하나의 객체를 다차원 동적 화일에 삽입하는 기존의 삽입 알고리즘을 반복적으로 적용함으로써 전체 다차원 화일 구조를 구성할 수도 있다. 그러나 이러한 경우, 데이타 공간(data space)[Niev84]상에서의 객체간의 인접성이 전혀 고려되지 않은 상태에서 객체들이 무작위 순으로 다차원 화일 구조내에 삽입된다. 이 결과, 같은 페이지를 디스크로부터 여러번 액세스하게 되므로 다차원 동적 화일 구성을 위한 오버헤드가 크다. 반면, 본 일괄 구성 알고리즘을 적용하는 경우에는 각 페이지를 디스크로부터 한번 액세스할 때, 이 페이지에 저장될 데이타 공간상의 인접한 여러 객체들을 함께 처리하므로 삽입 알고리즘을 반복적으로 적용하는 방식에서 발생하는 오버헤드를 해결할 수 있다.

데이타베이스내에 다차원 동적 화일이 이미 구성되어 있는 상태에서는 삽입 알고리즘을 반복적으로 적용함으로써 새로운 객체들을 다차원 동적 화일에 반영할 수 밖에 없다. 그러나 데이타베이스를 새로 구축하는 시점이나 이미 구축되어 있는 데이타베이스내에 새로운 다차원 동적 화일을 추가하는 경우에는 일괄 구성이 가능하다. 특히, 데이타베이스 구축시에는 수백만에서 수천만에 이르는 방대한 양의 객체들을 대상으로 하므로 효율적인 일괄 구성 알고리즘의 개발은 반드시 필요하다.

본 논문에서는 다차원 동적 화일의 일괄 구성을 위한 최적의 알고리즘을 제안한다. 본 논문에서는 보다 구체적인 서술을 위하여 일괄 구성 알고리즘의 대상으로 이미 제안된 많은 다차원 동적 화일들 중 계층 그리드 화일(multilevel grid file)[Whan85][Whan91]을 선정하였다. 선정을 위하여 (1) 완전 매치 질의(exact match query), 부분 매치 질의(partial match query), 범위 질의(range query) 등 다양한 질의 처리의 효율성, (2) 객체의 삽입 및 삭제가 빈번한 동적 환경으로의 적응성, (3) 디렉토리 구조의 간결성, (4) 저장 공간의 효율성 등을 고려하였다. 계층 그리드 화일의 이러한 특성 및 성능 평가에 관한 자세한 논의는 참고 문헌 [Whan85][Whan91][Kim93][Whan94] 등에 잘 나타나 있다. 그러나 본 알고리즘의 적용이 단지 계층 그리드 화일에 제한되는 것은 아니며, 핵심이 되는 기본 아이디어는 다른 다차원

화일 구조의 일괄 구성을 위해서도 활용될 수 있다.

먼저, 개념적인 설명을 위한 기본 형태의 알고리즘을 제안하고, 디스크 액세스 수의 최소화를 위한 성능 개선 방안을 제시한다. 또한, 실제 구현시 나타나는 이슈와 이에 대한 해결 방안을 제시한다. 성능 분석을 위하여 일괄 구성의 수행시 발생되는 디스크 액세스 수를 분석함으로써 제안된 알고리즘의 최적성을 보인다.

본 논문의 구성은 다음과 같다. 제 2장에서는 논의의 대상이 되는 계층 그리드 화일의 특성에 관하여 간략히 소개한다. 제 3장에서는 일괄 구성을 위한 기본 알고리즘과 구현시의 디스크 액세스 최소화 방안에 관하여 논의 한다. 제 4장에서는 성능 분석으로서 제안된 알고리즘의 수행시 발생되는 디스크 액세스 수를 추정한다. 제 5장에서는 본 논문의 공헌을 요약하고, 향후 연구 방향을 제시한다.

2. 계층 그리드 화일

본 장에서는 제안한 일괄 구성 알고리즘의 대상이 되는 계층 그리드 화일(multilevel grid file)[Whan91]에 대하여 설명하고자 한다. 먼저 제 2.1절에서는 계층 그리드 화일의 동적 특성에 대하여 설명하고, 제 2.2절에서는 구조적 특성에 대하여 언급한다.

2.1. 계층 그리드 화일의 동적 특성

계층 그리드 화일은 다차원 동적 화일의 하나로서 다단계의 디렉토리와 데이타 단계로 구성된다. 각 단계의 디렉토리는 전체 데이타 공간의 분할 상태를 반영하며, 데이타 페이지는 데이타 공간에서 자신과 대응되는 영역의 객체들만을 저장한다.

계층 그리드 화일은 객체가 데이타 공간에 삽입되고 삭제되는 상황에 따라 분할과 병합을 반복함으로써 동적 변화에 적응한다. 데이타 공간내에 객체가 삽입되는 경우, 객체가 속하는 영역을 찾아 그 영역에 할당된 데이타 페이지에 객체를 삽입하게 된다. 이 결과 데이타 페이지의 용량이 초과되면(overflow), 해당 영역은 같은 크기를 갖는 두 영역으로 분할되고 새로운 데이타 페이지가 하나 더 할당된다. 기존의 데이타 페이지에 있던 객체들은 분할된 두 영역의 분할 경계값을 기준으로 각각의 영역에 할당된 두 데이타 페이지에 분산된다. 객체들을 삭제하는 경우에는 삽입 과정의 역에 해당되는 작업이 수행된다.

2.2.계층 그리드 화일의 구조적 특성

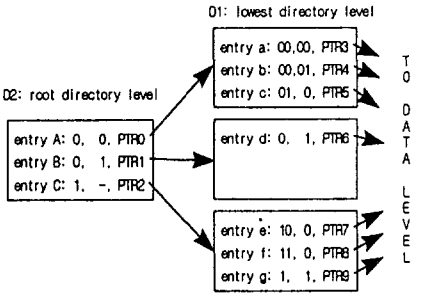
본 절에서는 계층 그리드 화일의 구조적 특성으로서 디렉토리 엔트리의 구조와 디렉토리의 구조에 대하여 언급한다.

디렉토리 엔트리는 리젼 벡터(region vector)와 다음 단계 페이지에 대한 포인터로 구성된다. N개의 애트리뷰트를 갖는 계층 그리드 화일의 리젼 백터는 N개의 해쉬값으로 구성되며, 해당 디렉토리 엔트리가 나타내는 영역의 위치, 모양, 그리고 크기에 대한 정보를 갖는다. 리젼 벡터에서 i번째 해쉬값은 그 디렉토리 엔트리의 영역내에 속하는 모든 객체들의 i번째 애트리뷰트를 해싱하였을 때 나타나는 해쉬값들의 공통 접두부(prefix)가 된다.

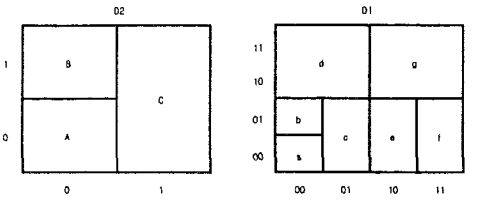
계층 그리드 화일은 디렉토리 구조로서 계층 구조를 취한다. 즉, 최하위 단계 디렉토리 D1내에서 데이타 공간상에서 서로 인접한 영역과 대응되는 엔트리들을 같은 페이지에 저장하고, 이 엔트리들이 표현하는 영역들을 모두 포함하는 큰 영역과 대

응되는 디렉토리 엔트리를 D1의 상위 단계 디렉토리 D2에 유지시킨다. 디렉토리 D2의 엔트리들을 하나의 페이지에 유지할 수 없게 되면, 같은 방식으로 그 상위 단계 디렉토리 D3를 두게되며, 이것은 최상위 단계 디렉토리 엔트리들이 하나의 디렉토리 페이지에 유지될 수 있을 때까지 반복된다. 이러한 구조적 특성은 계층 그리드 화일을 위한 삽입, 삭제 알고리즘 [Whan85][Whan91]에 의하여 자연스럽게 유지된다.

그림 2.1(a)는 두개의 애트리뷰트를 갖는 이단계 계층 그리드 화일의 전체 디렉토리 구조를 표현한 것이다. 그림 2.1(b)는 각각 그림 2.1(a)의 디렉토리 D1과 D2가 나타내는 데이타 공간의 분할 상태를 도면화한 것이다. 최하위 단계 디렉토리 D1을 위한 디렉토리인 동시에 루트 디렉토리인 D2의 한 엔트리 A는 첫번째 애트리뷰트와 두번째 애트리뷰트의 해쉬값의 접두부가 각각 '0', '0'인 객체들이 속한 영역을 나타내며, 이 영역은 D1에서 디렉토리 엔트리 a, b, c가 나타내는 세개의 영역으로 다시 세분된다. 따라서 상위 디렉토리에서 하위 디렉토리로 내려올수록 보다 구체화된 데이타 공간의 분할 상태가 나타난다.



(a) 이단계 디렉토리 구조.



(b) D1과 D2내의 디렉토리 엔트리들이 표현하는 영역들.

그림 2.1. 두개의 애트리뷰트를 갖는 계층 그리드 화일의 예.

3. 일괄 구성 알고리즘

본 장에서는 계층 그리드 화일의 일괄 구성을 위한 알고리즘을 제안한다. 먼저, 제 3.1절에서는 보다 쉬운 개념의 이해를 위하여 기본 형태의 알고리즘을 기술하고, 제 3.2절에서 이 알고리즘의 성능 개선 방안에 대하여 논의 한다.

3.1. 기본 알고리즘

일괄 구성 알고리즘의 핵심 아이디어는 하나의 페이지를 디스크로부터 한번 액세스할 때, 이곳에 저장되어야 할 데이타 공간상에서 인접한 객체들을 함께 처리하는 것이다. 이를 위하여 먼저 데이타 공간상에서 서로 인접한 객체들을 물리적으로도 인접하도록 사전 처리(preprocessing)로서 재배치 작업을 수행한다.

재배치 작업 후 계층 그리드 화일의 구성은 상향식(bottom-up)으로 진행된다. 먼저, 재배치된 데이타 화일내의 객체들을 참조하여 계층 그리드 화일의 데이타 단계를 위한 페이지화 작업을 수행하고, 이를 참조하여 그 상위의 디렉토리 단계들을 위한 페이지화 작업을 차례로 수행한다. 이렇게 하위 단계를 참조하여 상위 단계를 구성하는 작업은 최종 디렉토리 단계가 하

나의 페이지로 구성되는 루트 단계로 될 때까지 반복된다.

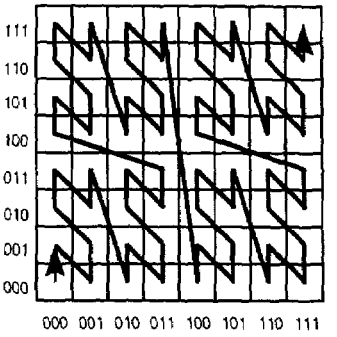
그림 3.1은 계층 그리드 화일의 일괄 구성을 위한 알고리즘을 개략적으로 나타낸 것이다.

1. 재배치 작업을 수행한다
2. 데이타 단계의 페이지화 작업을 수행한다.
3. 디렉토리 단계가 단 하나의 페이지만으로 구성될 때까지,
   해당 디렉토리 단계의 페이지화 작업을 수행한다.

3.1. 계층 그리드 화일의 일괄 구성을 위한 기본 알고리즘.

(1) 재배치 작업

재배치 작업에서 해결해야 하는 가장 큰 문제는 다차원 데이타 공간상에서 인접한 객체들을 일차원 물리적 공간상에서도 인접하도록 변환해야 한다는 것이다. 본 연구에서는 이러한 변환 메카니즘으로서 Z-변환(Z-transformation)[Oren86][Oren88]을 사용한다. Z-변환은 n 차원의 영역[1]을 표현하는 n 개의 k 비트 이진 스트링을 순환적으로 서로 끼우면서 일차원의 영역을 표현하는 k*n 비트의 이진 스트링으로 바꿔주는 함수이다. 예를 들어, bij를 i 번째 차원의 j 번째 비트라 하면, n 차원의 영역은 $(b_{11}b_{12},...,b_{1k}, b_{21}b_{22},...,b_{2k}, ... b_{n1}b_{n2},...,b_{nk})$로 표현된다. 이것을 Z-변환하면, $b_{11}b_{21}...b_{n1}b_{12}b_{22}...b_{n2}$ ..... $b_{1k}b_{2k}...b_{nk}$가 되며, 이를 Z-값(Z-value)이라 한다. 이러한 Z-값들은 사전식 순서(lexicographical order)로 비교가 가능하며, 이러한 순서를 Z-순서(Z-order)라 한다. 예를 들어, 각 차원의 좌표가 세비트의 이진 스트링으로 표현되는 이차원 데이타 공간에서의 Z-순서는 그림 3.2와 같다. 그림에서 나타난 바와 같이 Z-순서에 의한 일차원 공간내에서의 인접성은 다차원 공간에서의 인접성을 잘 반영하고 있음을 볼 수 있다.



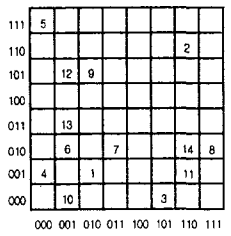
3.2. 이차원 데이타 공간에서의 Z-순서의 예.

재배치 작업을 위하여 각 객체의 Z-값을 대상으로 외부 정렬(external sorting)[Horo93]을 수행한다. 즉, 데이타 화일내의 객체들에 대하여 Z-값을 대상으로 버퍼 페이지 단위의 내부 정렬(internal sorting)을 먼저 수행하고, 그 결과에 대하여 K+1 개의 버퍼 페이지를 이용하는 K 원 병합(K-way merge)[Horo93]을 반복적으로 수행함으로써 전체 객체들을 Z-순서로 정렬시킨다.

그림 3.3은 본 장에서 사용할 예제를 위한 객체의 특성을 나타낸 것이다. 왼쪽의 그림은 데이타 공간상에 존재하는 객체의 분포 특성을 나타낸 것이며, 오른쪽의 그림은 각 객체의 x, y 좌표의 해쉬값과 이를 이용하여 구성한 Z-값을 데이타 화일내에 저장된 순서대로 나타낸 것이다. 그림 3.4는 그림 3.3의 객체들을 대상으로 수행한 재배치 작업의 결과로서 Z-순서로 정렬

1 다차원 데이타 공간상에서는 객체 자체를 표현하는 점도 사실상은 데이타 공간상의 최대 해상도(maximum resolution)를 가지는 한 영역으로 간주할 수 있다. 따라서 본 논문에서는 객체와 대응되는 점도 하나의 영역으로 취급한다.

된 객체들을 저장된 순서대로 나타낸 것이다.



Object	hash values(x, y)	Z-value
1	(010, 001)	001001
2	(110, 110)	111100
3	(101, 000)	100010
4	(000, 001)	000001
5	(000, 111)	010101
6	(001, 010)	000110
7	(011, 010)	001110
8	(111, 010)	101110
9	(010, 101)	011001
10	(001, 000)	000010
11	(110, 001)	101001
12	(001, 101)	010011
13	(001, 011)	000111
14	(110, 010)	101100

그림 3.3. 예제를 위한 데이타.

Object	hash values(x, y)	Z-value
4	(000, 001)	000001
10	(001, 000)	000010
6	(001, 010)	000110
13	(001, 011)	000111
1	(010, 001)	001001
7	(011, 010)	001110
12	(001, 101)	010011
5	(000, 111)	010101
9	(010, 101)	011001
3	(101, 000)	100010
11	(110, 001)	101001
14	(110, 010)	101100
8	(111, 010)	101110
2	(110, 110)	111100

그림 3.4. 재배치 작업후의 객체들의 저장 순서.

(2) 데이타 단계의 페이지화 작업

재배치 작업 후에는 정렬된 객체들을 페이지 단위로 그룹화 하는 데이타 단계를 위한 페이지화 작업이 필요하다. 이 작업에서는 정렬된 객체들을 차례로 액세스하며, 각 데이타 페이지에 속하게 될 객체들을 파악하여 해당 데이타 페이지내에 저장한다. 이 작업의 결과 계층 그리드 화일의 데이타 단계가 완성된다. 한편, 이와 병행하여 데이타 단계 바로 상위에 존재하는 최하위 디렉토리 단계를 위한 디렉토리 엔트리들을 생성한다. 각 엔트리의 리전 벡터는 대응되는 데이타 페이지내의 객체들을 모두 포함하는 영역을 표현하며, 다음 단계 페이지 포인터는 이 데이타 페이지의 식별자를 가리킨다.

세부적인 페이지화 작업 방식은 다음과 같다. 먼저, 직전에 이미 확정된 데이타 페이지와 대응되는 영역에 포함되지 않는 최소의 Z-값이 나타내는 영역에서 시작하여 정렬된 데이타 화일에서 새롭게 나타나는 객체들을 차례로 포함할 수 있도록 영역을 점차 확장시킨다. 영역의 확장은 현재 영역을 기준으로 Z-순서 방향으로 다음에 나타나는 동일한 크기의 영역을 포함하는 방식으로 두배씩 확장된다. 영역의 확장시 다음과 같은 두가지 조건을 모두 만족하지 못하는 경우에는 확장을 중단하고, 확장 직전의 영역내에 속하는 객체들을 해당 데이타 페이지내에 저장함으로써 데이타 페이지를 확정시킨다.

- 조건 1: 확장된 영역내에 포함되는 객체 수는 데이타 페이지의 블러킹 인수 이하여야 한다.
- 조건 2: 확장된 영역은 이미 확정된 페이지들과 대응되는 영역들과 서로 소여야 한다.

조건 (1)은 데이타 페이지의 오버플로우를 방지시키기 위한 것이며, 조건 (2)는 같은 디렉토리 단계에서 나타나는 모든 영역들은 서로 소라는 계층 그리드 화일의 구조적 특성을 만족시키기 위한 것이다. 그림 3.5는 그림 3.4(b)에 나타난 객체들을 차례로 액세스하여 블러킹 인수가 3인 데이타 단계의 페이지화 작업을 수행한 결과이다.

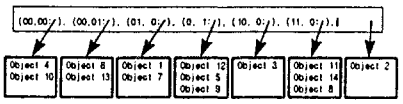


그림 3.5. 데이타 단계의 페이지화 작업의 결과.

(3) 디렉토리 단계의 페이지화 작업

이 작업에서는 직전의 데이타 단계의 페이지화 작업의 결과로 얻은 정렬된 디렉토리 엔트리들을 차례로 액세스함으로써 각

디렉토리 페이지내에 속하게 될 엔트리들을 파악하고, 이들을 해당 페이지내에 저장한다. 이러한 방식은 데이타 단계를 위한 페이지화 방식과 동일하며, 따라서 조건 (1)과 조건 (2)가 영역 확장 여부를 결정하는 기준으로 사용된다. 디렉토리 단계의 페이지화 작업의 결과 계층 그리드 화일의 최하위 디렉토리 단계가 완성되며, 또한 그 상위의 디렉토리 엔트리들도 생성된다. 그림 3.6은 그림 3.5를 이용하여 최하위 디렉토리 단계의 페이지화 작업을 수행한 결과이다.

이러한 내부 단계의 페이지화 작업은 상위 단계로 계속 올라가면서 전체 단계를 하나의 페이지내에 유지할 수 있을 때까지 반복된다. 그림 3.7은 그림 3.6을 이용하여 두번째 하위 디렉토리 단계의 페이지화 작업을 수행한 결과를 나타낸 것이다. 이 단계에서는 전체 엔트리들을 하나의 페이지내에 유지시킬 수 있으므로 모든 작업이 종료된다.

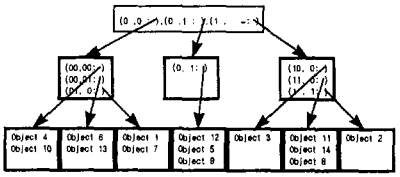


그림 3.6. 첫번째 디렉토리 단계의 페이지화 작업의 결과.

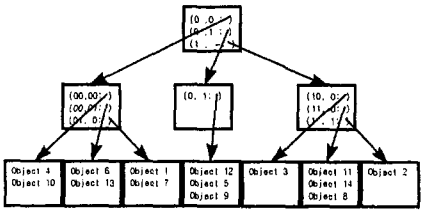


그림 3.7. 두번째 디렉토리 단계의 페이지화 작업의 결과.

3.2. 성능 개선 방안

성능 개선을 위하여 본 논문에서는 각 단계의 페이지화 작업이 완료된 후에야 비로소 상위 단계의 페이지화 작업을 시작하는 기본 알고리즘 대신, 하위 단계의 페이지화 작업과 상위 단계의 페이지화 작업을 유기적으로 연관시켜 모든 단계의 페이지화 작업을 병행하는 방안[Kim95]을 사용한다. 즉, 어떤 단계에서 페이지화가 확정되면, 이 페이지와 대응되는 상위 단계의 엔트리를 생성시켜 이를 상위 단계에 넘겨준다. 상위 단계에서 이 엔트리를 받음으로 인하여 페이지화가 확정되면, 다시 이 페이지와 대응되는 그 상위 단계의 엔트리를 생성시켜 이를 상위 단계에 넘겨 준다. 이러한 작업을 데이타 단계가 모두 페이지화 될 때까지 반복시킨다. 이와 같이 하위 단계의 페이지화 작업과 상위 단계의 페이지화 작업이 병행될 수 있다. 이 결과 재배치 작업의 완료 후에는 계층 그리드 화일에 속하는 페이지 수 만큼만의 디스크 쓰기가 발생한다[Kim95].

4. 성능 분석

본 장에서는 제안된 알고리즘에 대한 성능 분석으로서 객체들이 데이타 공간상에서 균일하게 분포한다는 가정하에 본 알고리즘을 적용하여 계층 그리드 화일을 구성할 때 발생되는 디스크 액세스 수를 추정한다.

먼저, 디스크 액세스 수의 분석을 위하여 필요한 인자를 정의한다. N은 구성의 대상이 되는 객체의 수를 나타낸다. B_{data}와 B_{dir}는 각각 계층 그리드 화일의 데이타 페이지와 디렉토리 페이지의 블러킹 인수로서, 페이지내에 몇개의 객체 혹은 엔트리들을 저장할 수 있는가를 나타낸다. 또한,

K는 재배치 작업에서 사용되는 외부 정렬 내의 K 원 병합을 위한 K 값을 나타낸다.

데이타 화일은 $\lceil(N/B_{data})\rceil$개의 페이지들로 구성된다. 페이지 단위의 내부 정렬과 K 원 병합 작업을 이용하여 전체 객체들을 정렬하는데 필요한 디스크 읽기 및 쓰기 단계 수는 $\lceil\log_K\lceil(N/B_{data})\rceil\rceil$이다 [Horo93]. 여기서 재배치 작업시 데이타 화일에서 객체들을 참조하기 위하여 발생하는 디스크 액세스는 계층 그리드 화일의 구성을 위한 어떤 알고리즘에서도 발생하는 것이므로 이를 비용에서 제외한다. 또한, K 원 병합 작업의 마지막 단계에서 나타나는 결과에 대해서는 쓰기 작업을 하지 않고, 이를 그대로 페이지화 작업의 입력으로 사용할 수 있으므로 이 비용 역시 제외한다. 따라서 디스크 읽기 및 쓰기 단계수는 각각 $\lceil\log_K\lceil(N/B_{data})\rceil\rceil-1$이다. 각 단계에서 디스크 읽기와 쓰기의 두번의 디스크 액세스가 발생하므로 재배치 작업을 위한 총 디스크 액세스 수는 다음과 같다.

$$relocation\ cost = \lceil(N/B_{data})\rceil \times (\lceil\log_K\lceil(N/B_{data})\rceil\rceil - 1) \times 2 \quad (식\ 4.1)$$

데이타 단계와 디렉토리 단계를 위한 페이지화 작업에서는 제 3.2절에서 설명한 성능 개선 방안을 적용하면, 완성된 계층 그리드 화일을 구성하는 페이지 수 만큼의 디스크 쓰기만이 발생된다. 객체들이 데이타 공간상에서 균일하게 분포할 때, 계층 그리드 화일과 같이 데이타 공간을 그리드 형태로 분할하는 자료 구조에서의 페이지 점유율은 ln2(약 0.6931)라 알려져 있다. 본 논문에서는 이를 기반으로 주어진 N에 대하여 일괄 구성 작업의 결과로 완성된 계층 그리드 화일의 페이지의 수를 추정한다. 먼저, 데이타 단계를 위한 페이지의 수는 $\lceil N/(B_{data}\ln 2)\rceil$가 된다. 또한, 이를 이용하여 각 단계 디렉토리를 위한 페이지 수를 구하면, 최하위 단계 D1을 위한 페이지 수는 $\lceil\lceil N/(B_{data}\ln 2)\rceil/(B_{dir}\ln 2)\rceil$, 그 상위 단계 D2를 위한 페이지 수는 $\lceil\lceil\lceil N/(B_{data}\ln 2)\rceil/(B_{dir}\ln 2)\rceil/(B_{dir}\ln 2)\rceil$, ... 이므로 이를 일반화시키면 다음과 같다.

$$pagination\ cost = \lceil N/(B_{data}\ln 2)\rceil + \sum_{x=1}^{\lceil\log_{B_{dir}\ln 2}\lceil N/(B_{data}\ln 2)\rceil\rceil} \lceil(\lceil N/(B_{data}\ln 2)\rceil/(B_{dir}\ln 2)^x)\rceil \quad (식\ 4.2)$$

따라서, 본 알고리즘을 적용하여 계층 그리드 화일을 구성하는 경우의 전체 디스크 액세스 수는 (식 4.1)과 (식 4.2)의 합이므로 다음과 같다.

$$total\ cost = \lceil(N/B_{data})\rceil \times (\lceil\log_K\lceil(N/B_{data})\rceil\rceil - 1) \times 2 + \lceil N/(B_{data}\ln 2)\rceil + \sum_{x=1}^{\lceil\log_{B_{dir}\ln 2}\lceil N/(B_{data}\ln 2)\rceil\rceil} \lceil(\lceil N/(B_{data}\ln 2)\rceil/(B_{dir}\ln 2)^x)\rceil \quad (식\ 4.3)$$

본 알고리즘을 적용하여 계층 그리드 화일을 구성하는 경우, 재배치 작업을 제외하면 계층 그리드 화일에 속하는 페이지 수 만큼만의 디스크 쓰기가 발생한다. 재배치 작업이 일괄 구성을 위한 필수 불가결한 전처리 단계인 것을 감안하면, 제안된 알고리즘은 각 페이지에 대하여 단 한 번의 디스크 액세스만을 요구하므로 일괄 구성을 위한 최적의 알고리즘이라 할 수 있다.

5. 결 론

다차원 동적 화일은 다중 애트리뷰트 액세스를 효과적으로 지원하기 위한 구조이다. 데이타베이스를 새롭게 구축하는 경우에는 매우 방대한 양의 객체들을 대상으로 하므로 다차원 동적 화일을 위한 효율적인 일괄 구성 기법이 요구된다. 본 논문에서는 다차원 동적 화일의 하나인 계층 그리드 화일을 위한 최적의 일괄 구성 알고리즘을 제안하였다. 제안된 일괄 구성 알고리즘에서는 계층 그리드 화일을 구성

하는데 필요한 페이지를 디스크로부터 한 번 액세스할 때, 이곳에 저장될 모든 객체들과 엔트리들을 한꺼번에 처리함으로써 기존의 삽입 알고리즘을 반복적으로 적용하는 경우 같은 페이지를 디스크로부터 여러번 액세스하게 되는 문제점을 해결할 수 있다.

성능 분석을 위하여 본 알고리즘을 적용하여 계층 그리드 화일을 구성할 때 발생되는 디스크 액세스 수를 분석하였다. 분석 결과에 의하면, 일괄 구성을 위하여 반드시 요구되는 재배치 작업을 제외하면, 계층 그리드 화일의 구성에 사용되는 페이지 수 만큼만의 디스크 쓰기가 발생하는 것으로 나타났다. 따라서 각 페이지에 대하여 단 한번의 디스크 액세스만을 요구하므로 제안된 알고리즘은 일괄 구성을 위한 최적의 알고리즘이라 할 수 있다. 현재까지 다차원 동적 화일을 위한 일괄 구성 기법은 국내외를 막론하고 전혀 다루어 진 바 없다. 따라서 본 연구 결과는 다차원 동적 화일을 이용하는 데이타베이스 구축 시의 성능을 크게 개선시킬 수 있을 것으로 확신한다.

향후 연구 방향은 기존의 계층 그리드 화일을 위한 삽입 알고리즘을 반복적으로 적용하는 경우와의 실험적인 비교를 통하여 제안된 알고리즘을 적용하는 경우 얼마만큼의 성능 개선 효과를 가져오는가를 규명하는 것이다.

참 고 문 헌

[Bent79] Bentley, J. L., "Multidimensional Binary Search Trees in Database Applications," *IEEE Trans. Software Engineering*, Vol. SE-5, No. 4, pp. 333-340, July 1979.

[Burk83] Burkhard, W. A., "Interplation-Based Index Maintenance," In *Proc. 2nd ACM Symp. on Principles of Database Systems*, ACM SIGMOD, pp. 79-89, 1983.

[Henr89] Henrich, A., Six, H. W., and Widmyer, P., "The LSD Tree: Spatial Access to Multidimensional Point and Non-Point Objects," In *Proc. 15th Intl. Conf. on Very Large Data Bases*, VLDB, pp. 45-53, 1989.

[Horo93] E. Horowitz, S. Sahni, and S. Anderson-Freed, *Fundamentals of Data Structures in C*, Computer Science Press, 1993.

[Kim93] Kim, S. W. and Whang, K. Y., "Asymptotic Directory Growth of the Multilevel Grid File," In *Proc. Intl. Symp. on Next Generation Database Systems and Their Applications*, pp. 257-264, Fukuoka, Japan, Sept. 1993.

[Kim95] 김 상욱, 황 환규, 황 규영, "B+ 트리를 위한 벌크 로드 알고리즘," 한국 정보과학회 추계학술대회 발표 논문집(A), Vol. 22, No. 2, pp. 243-246, 1995.

[Lome90] Lomet, D. and Salzberge, B., "The hB-Tree: A Multidimensional Indexing Method with Good Guaranteed Performance," *ACM Trans. on Database Systems*, Vol. 15, No. 4, pp. 625-658, Dec. 1990.

[Lome92] Lomet, D., "A Review of Recent Work on Multi-attribute Access Methods," *ACM SIGMOD RECORD*, Vol. 21, No. 3, pp. 5

6-63, Sept. 1992.

[Niev84] Nievergelt, J. et al., "The Grid File: An Adaptable, Symmetric Multikey File Structure," *ACM Trans. on Database Systems,* Vol. 9, No. 1, pp. 38-71, Mar. 1984.

[Oren86] Orenstein, J. A., "Spatial Query Processing in an Object-Oriented Database System," In *Proc. Intl. Conf. on Management of Data,* ACM SIGMOD, pp. 326-336, 1986.

[Oren88] Orenstein, J. A. and Manola, F. A., "PROBE Spatial Data Modeling and Query Processing in an Image Database Application," *IEEE Trans. on Software Engineering*, Vol. 14, No. 6, pp. 611-629, 1988.

[Otoo84] Otoo, E. J., "A Mapping Function for the Directory of a Multidimensional Extendible Hashing," In *Proc. 10th Intl. Conf. on Very Large Data Bases*, pp. 493-506, Aug. 1984.

[Otoo85] Otoo, E. J., "A Multidimensional Digital Hashing Scheme for Files with Composite Keys," In *Proc. Intl. Conf. on Management of Data*, ACM SIGMOD, pp. 214-229, 1985.

[Otoo86] Otoo, E. J., "Balanced Multidimensional Extendible Hash Tree," In *Proc. 6th ACM Symp. on Principles of Database Systems*, ACM SIGMOD, pp. 100-113, 1986.

[Ouks83] Ouksel, M. and Scheuermann, P., "Storage Mapping for Multidimensional Linear Hashing," In *Proc. 3rd ACM Symp. on Principles of Database Systems,* ACM SIGMOD, pp. 90-105, 1983.

[Robi81] Robinson, J. T., "The K-D-B Tree: A Search Structure for Large Multidimensional Dynamic Indexes," In *Proc. Intl. Conf. on Management of Data,* ACM SIGMOD, pp. 10-18, 1981.

[Tamm82] Tamminen, M., "The Extendible Cell Method for Closest Point Problems," *BIT,* Vol. 22, pp. 27-41, 1982.

[Whan85] Whang, K. Y. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC11516, 1985.

[Whan91] Whang, K. Y. and Krishnamurthy, R., "The Multilevel Grid File-A Dynamic Hierarchical Multidimensional File Structure," In *Proc. 2nd Intl. Conf. on Database Systems for Advanced Applications,* pp. 449-459, Apr. 1991.

[Whan94] Whang, K. Y., Kim, S. W., and Wiederhold, G., "Dynamic Maintenance of Data Distribution for Selectivity Estimation," *The VLDB Journal*, Vol. 3, No. 1, pp. 29-51, 1994.

Acknowlegment

본 연구는 한국과학재단 96년도 핵심전문연구과제 "GIS 응용을 위한 다차원 동적 화일의 일괄 구성 및 죠인 처리에 관한 연구"의 연구비를 지원 받았슴.