고정 그리드 파일의 객체 및 셀 클러스터링 알고리즘

(Object and Cell Clustering Algorithms of the Fixed Grid File)

조 대 수 † 유 진 영 †† 홍 봉 희 †††

(DaeSoo Cho) (JinYoung Yoo) (BongHee Hong)

요 약 공간 데이타베이스에서 효율적으로 공간 질의를 처리하기 위해서는 클러스터링을 통해서 디스크 접근 비용을 줄이는 것이 필요하다. 이 논문은 공간 지역성에 기반을 둔 여러 가지 클러스터링 알고리즘을 제안하고 실험을 통해 제안한 클러스터링 알고리즘의 성능을 평가하였다. 이 논문에서 제안하는 클러스터링 알고리즘은 객체 클러스터링 알고리즘과 셀 클러스터링 알고리즘으로 나뉜다. 객체 클러스터링 알고리즘은 정규 분할 공간 색인 구조에서 영역 분할 선과 겹치는 객체들의 저장 위치를 결정하는데 사용된다. 셀 클러스터링 알고리즘은 클러스터를 만들기 위해 정규 분할된 영역들을 그룹화하는데 사용된다.

실험 결과 객체 클러스터링 알고리즘에서는 객체간의 거리를 이용한 경우에 대체로 좋은 성능을 보였지만, 버퍼 크기가 커지거나 데이타가 희박한 영역의 질의에 있어서는 알고리즘 별로 성능의 차이는 거의 없었다. 셀 클러스터링 알고리즘에 대한 실험에서는 이 논문에서 제안한 클러스터링 알고리즘은 N-순서화 기법에 의한 클러스터링 알고리즘에 비해 우수한 성능을 보였다. 특히 중복 참조도를 이용한 경우와 셀의 무게 중심간 거리를 이용한 방법이 가장 우수하였다.

Abstract For efficiently processing spatial queries in a spatial database, it is required to reduce the cost of disk accesses by clustering spatial data on one-dimensional space of disk. In this paper, we propose several clustering algorithms based on spatial locality in order to store spatial data in the fixed grid file, and evaluate the performance of each of newly proposed clustering algorithms. The clustering scheme consists of two steps: object clustering and cell clustering. The object clustering algorithms are used to determine to assign an object, which overlap with two or more adjacent grid-cells, to a specific grid cell. The cell clustering algorithms are used to make clusters by grouping a set of logically related grid-cells.

Experimental results show that the object clustering algorithm using the object distance performs better than the others. However, if the buffer size becomes more larger and also the range queries are carried out in the sparse area, there is little difference in performance among the object clustering algorithms. The cell clustering algorithms proposed in this paper show better performance than the N-order-based clustering algorithm. In particular, the cell clustering algorithms using the number of duplicated references and the distance between the cell gravities are best.

1. 서 론

지리정보시스템(Geographic Information System: GIS)은 대규모의 공간정보를 다루기 때문에 사용자 정의 질의에 대한 질의 처리 속도가 느린 문제점을 가진다. 지금까지 질의 처리 속도를 개선하기 위해 다양한 공간 색인 기법과 데이타 클러스터링 방법들이 연구되었다[1, 2, 3, 6, 7, 8, 10, 11]. 이 논문에서는 [7]을 기반으로 정규분할 공간 색인 구조를 가지는 공간 데이타베이스에서 효율적인 공간 질의 처리를 위한 데이타 클러스터링 알고리즘을 제시한다.

† 학생회원 : 부산대학교 컴퓨터공학과
dsio@hyowon.pusan.ac.kr
†† 비 회 원 : 삼성 SDS(주) 컨설팅사업부 연구원
jyyoo@samsung.co.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@hyowon.pusan.ac.kr
논문접수 : 2000년 2월 21일
심사완료 : 2000년 11월 13일

데이타베이스에서 클러스터링(clustering)의 정의는 논리적으로 인접한 데이타를 데이타 페이지 내에서 물리적으로 인접하게 저장하는 것이다. GIS에서 논리적으로 인접하다는 것은 객체들이 공간적으로 인접하다는 것을 의미한다. 즉, GIS에서 클러스터링은 서로 인접한 객체들을 디스크 내에 인접하게 저장하는 것으로 정의한다.

클러스터링의 목적은 질의 처리 시간을 구성하는 CPU 시간과 디스크 I/O 시간 중에서 I/O 시간을 최소화하는데 있다[14]. 디스크 I/O 시간은 탐색시간과 전송시간으로 이루어지며 탐색시간이 전송시간에 비해 많은 시간을 차지한다. 클러스터링은 탐색시간을 최소화하여 질의 처리 속도를 높이기 위해 사용된다. 지금까지 R-Tree 계열 색인[2,4], Quad-Tree 색인[3], 그리고 Grid-File 형태의 색인[1,7]들을 이용한 클러스터링 방법들이 연구되었으나, 정규분할 공간 색인을 이용한 클러스터링 알고리즘에 대한 연구는 거의 이루어지지 않고 있다. 이 논문에서는 정규분할 공간 색인으로 고정 그리드 파일[10]을 사용한다. 고정 그리드 파일은 객체의 공간적 위치와는 무관하게 영역을 분할(이 논문에서는 분할된 영역을 셀이라 부른다)하는 특성을 가진다.

고정 그리드 파일 공간 색인은 다음과 같은 특징이 있다. 첫째, R-tree계열과 같이 객체 중심의 영역 분할 방식을 사용하지 않기 때문에, 둘 이상의 셀에 중첩되는 객체들이 발생한다. 이러한 객체들은 분할해서 저장하는 방법, 중복 참조하는 방법, 그리고 중복 저장하는 방법으로 처리 될 수 있다. 공간 객체를 분할 할 경우에는 분할된 객체를 결합(composition)하는데 많은 비용이 발생하고, 중복 저장하는 방법에서는 저장 장소의 낭비가 발생하기 때문에[1] 이 논문에서는 객체를 중복 참조하는 방법을 사용한다. 중복 참조란 객체는 실제로 디스크에 한번만 저장되지만 그 객체와 중첩되는 모든 셀들에서 객체의 저장 위치를 가지고 있음을 의미한다. 중복 참조 방식에서 객체의 저장 위치를 결정하는 것은 디스크 I/O 횟수에 영향을 준다.

예를 들어, 그림 1에서 공간 객체 C는 셀1, 셀3에 중첩되고 있다. 이 때, C 객체는 그림 1.(a)와 같이 A 객체, B객체와 동일한 페이지에 저장될 수도 있고, 그림 1.(b)와 같이 D객체, E객체와 동일한 페이지에 저장될 수 있다. 각각의 경우에 주어진 질의 영역(A, B, C객체를 요청)에 대한 질의 수행을 위해서 그림 1.(a)는 *page-1*을, 그림 1.(b)는 *page-1*, *page-N*을 읽기 위해 디스크 I/O가 발생한다. 즉, 중복 객체의 저장 위치에 따라서 서로 다른 디스크 I/O 횟수가 발생함을 알 수

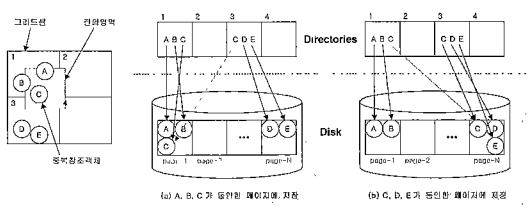


그림 1 중복 참조 예제 (C객체가 중복 참조됨)

있다. 따라서 디스크 I/O가 적게 발생할 수 있도록 중복 객체의 저장위치를 결정해야 한다.

두 번째 특징으로는 고정 그리드 파일 공간 색인을 구성하는 모든 셀들은 계층 구조를 갖지 않는다. 따라서 색인 구조만으로는 셀간에 그룹화가 이루어질 수 없다. 그룹화는 클러스터링을 위해서 필요한 기본 연산으로 하나의 클러스터를 구성하기 위해 여러 개의 셀을 묶는 것을 의미한다. 반면에 R-tree 계열[12,13,14]이나 Quad-tree계열[15]의 색인은 계층 구조를 갖기 때문에, 임의의 중간 노드를 자신의 조상으로 가지는 말단 노드들로 그룹화가 가능하다. 셀간의 그룹화는 디스크 탐색 시간에 큰 영향을 준다.

예를 들어, 그림 2에서는 주어진 그리드 파일에서 각 셀들을 (a)Row-Order에 의한 셀 그룹화 방법과 (b)N-Order에 의한 셀 그룹화 방법을 보이고 있다. 각각의 경우 주어진 질의 영역(셀1, 셀2, 셀5, 셀6을 요청함)에 대한 질의 수행을 위해서 요청되는 디스크 페이지의 개수는 동일(중복 참조 객체를 고려하지 않는다면 *page-1*, *page-2*, *page-5*, *page-6*을 요청)하다. 그러나 페이지의 그룹화 방법에 따라서 디스크의 탐색시간이 서로 다르게 된다. 그림 2.(a)의 경우 2번의 디스크 탐색이 발생하지만, 그림 2.(b)의 경우에는 *page-1*, *page-2*, *page-5*, *page-6*이 모두 동일한 클러스터에 포함되기 때문에 1번의 디스크 탐색만이 발생함을 알

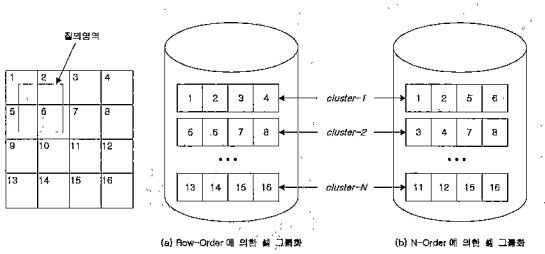


그림 2 셀 그룹화 예제
(페이지 4개를 하나의 클러스터로 생성)

수 있다. 따라서 디스크 탐색시간을 최소화하여 전체 질의 처리 시간을 줄이기 위해서는 공간 데이타의 클러스터링이 필요하며, 고정 그리드 파일을 공간 색인으로 사용할 경우에 클러스터링을 위해서 여러 셀들을 그룹화하기 위한 규칙이 정의되어야 한다.

이 논문에서는 객체의 중복참조에 따른 디스크 I/O 횟수를 줄이기 위해서 객체의 무게중심과 같은 공간 정보를 이용하여 중복참조 객체의 저장 위치를 결정하는 것을 객체 클러스터링(Object Clustering)으로 정의하고, 객체 클러스터링을 위한 알고리즘을 제시하고 성능을 비교 분석한다. 객체 클러스터링 알고리즘으로는 (1) 무게중심을 이용한 알고리즘, (2) 최단거리를 이용한 알고리즘, 그리고 (3)무게중심간 거리를 이용한 알고리즘을 제시한다. 또한 셀간의 공간적 근접성을 반영하여 서로 인접한 셀들을 클러스터(cluster)라는 단위로 묶어 클러스터 단위로 적재하는 것을 셀 클러스터링(Cell Clustering)으로 정의하고, 셀 클러스터링을 위한 알고리즘을 제시하고 성능을 비교 분석한다. 셀 클러스터링 알고리즘으로는 (1)셀의 무게중심(정의7 참조)간 거리를 이용한 알고리즘, (2)중복 참조도(정의10 참조)를 이용한 알고리즘, (3)셀간의 인력(정의11 참조)을 이용한 알고리즘, 그리고 (4)포함 공간 객체(정의3 참조)의 개수를 이용한 알고리즘을 제시한다.

이 논문의 구성은 다음과 같다. 2장에서 클러스터링에 대한 관련 연구를 살펴보고, 3장에서는 이 논문에서 다루려는 고정 그리드 파일을 이용한 공간 데이타 저장 구조에 대해서 설명한다. 이 논문에서 제시하는 객체 클러스터링 알고리즘과 셀 클러스터링 알고리즘은 각각 4장과 5장에서 기술한다. 그리고 6장에서는 이 논문에서 제시한 클러스터링 알고리즘에 대한 성능 비교 실험의 결과를 기술하고, 마지막으로 7장에서 결론 및 향후 연구과제에 대해서 기술한다.

2. 관련연구

공간 데이타의 클러스터링에 대한 연구는 모두 공간 인덱스의 종류에 의존적이다. 사분 트리(Quad-Tree) 또는 R-tree 계열의 계층적 구조를 가지는 공간 인덱스를 이용한 클러스터링에 대한 연구[2, 3]가 있다. 계층적 구조의 공간 인덱스는 트리 형태로 구성되어 있다. 말단 노드에는 공간적으로 인접한 공간 객체들이 할당되며, 중간 노드들은 또한 공간적으로 인접한 말단 노드들이 할당된다. 따라서 특정한 위치의 중간 노드에 포함된 모든 공간 객체들을 하나의 클러스터로 구성하게 되면, 이들 공간 인덱스의 특성상 공간적으로 인접한 공간 객체들이 동일한 클러스터에 포함되게 된다. 이 방법에서는 인접한 공간 객체들로 구성된 클러스터를 디스크 I/O의 기본 단위로 사용하기 때문에 디스크 접근 비용을 크게 줄일 수 있다. 그러나, 이 논문의 적용 대상인 정규분할 공간 색인은 평면 구조를 가지기 때문에 이 방법들은 그대로 적용하거나 일부 수정하여 적용하기 어렵다.

비정규분할 공간 색인인 그리드 파일을 이용한 클러스터링에 대한 연구[1]에서는 하나의 셀 내에 포함된 공간 객체들을 하나의 클러스터에 저장한다. 즉, 셀과 클러스터는 일대일 관계를 가지며, 셀 내에 포함된 공간 객체들의 크기의 합이 셀마다 서로 다르기 때문에 가변 길이 클러스터를 사용한다. 이 방법에서는 둘 이상의 셀에 중첩되는 공간 객체는 분할하여 분할된 공간 객체를 각각의 셀에 저장하는 방식을 사용하기 때문에, 중복 참조 객체는 발생하지 않는다. 이 연구의 핵심은 질의의 이력(query history)에 따라 셀 분할선을 변경하는 것이다.

셀 분할선을 변경하는 이유는 다음과 같다. 첫째, 셀 크기가 너무 작은 경우에는 공간 객체의 분할이 많이 발생하기 때문에 분할된 객체의 결합을 위해서 많은 비용이 발생하기 때문에 공간 객체의 분할이 적게 발생하도록 셀의 크기를 크게 할 필요가 있다. 둘째, 셀 크기가 너무 큰 경우에는 하나의 클러스터 내에 많은 수의 공간 객체가 포함되므로 질의에 필요한 객체 외에 불필요한 객체(false drop)가 발생하여 계산 비용이 증가하기 때문에 불필요한 객체의 수를 줄이기 위해서 셀의 크기를 줄일 필요가 있다. 그런데 두 가지 이유는 서로 상반된 결과를 요구하기 때문에 이 연구에서는 객체 결합을 위한 비용과 불필요한 객체로 인한 비용이 서로 균형을 이루도록 셀 분할선을 재조정하였으며, 이 경우에 가장 좋은 성능을 보임을 보였다.

예를 들면, 그림 3에서 주어진 질의에 대해 셀 분할선의 개수에 따라서 질의1과 질의2를 처리하기 위해서 필요한 클러스터의 개수 및 평균 질의 처리비용은 표 1과 같다. 9x8의 경우가 10x8의 경우보다 필요한 클러스터의 수는 적지만 질의처리시간이 많이 걸린 이유는 셀의 크기가 커져서 질의 수행에 불필요한 객체로 인해 많은 처리 시간이 걸렸기 때문이다. 반면에 9x9의 경우가 10x8의 경우보다 많은 시간이 필요한 이유는 셀의 크기가 작아서 객체의 분할이 많이 발생하여 객체 결합으로

표 1 셀 분할선 조정에 따른 질의 수행 결과 [1]

	9x9		11x9		9x8		10x8	
	질의1	질의2	질의1	질의2	질의1	질의2	질의1	질의2
클러스터개수 (개)	6	4	10	4	3	1	4	1
평균 질의처리시간 (msec)	1107		833		883		667	

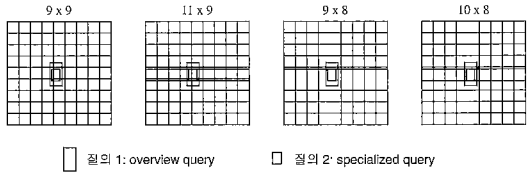


그림 3 질의 이력을 기반으로 셀 분할선 재조정[1]

많은 처리 시간이 걸렸기 때문이다.

질의 이력을 기반으로 한 셀 분할선 재조정은 기존의 질의와 유사한 질의가 계속적으로 반복 수행될 경우에, 즉, 질의의 이력이 앞으로의 질의를 잘 반영하는 경우에는 효과적이나, 분할선 재조정에 따라 클러스터를 재구성하는데 많은 시간이 걸리는 단점을 가진다. 이 방법은 공간 객체를 분할해서 저장하였기 때문에 중복 참조 객체가 발생하지 않고, 하나의 셀을 하나의 클러스터로 할당했기 때문에, 이 논문에서 다루려는 객체 및 셀 클러스터링과는 무관하다.

정규분할 공간 색인을 이용한 클러스터링에 대한 연구에서 객체 클러스터링과 유사한 연구는 없으며, 셀 클러스터링과 유사한 연구로는 공간 채우기 곡선을 이용한 순서화 기법[6]에 대한 연구가 있다. 공간 채우기 곡선은 2차원 공간 데이타를 1차원으로 순서화 시키는 방법으로 Hilbert 곡선, N 곡선, Grad-Code 기반의 곡선, 열-우선(row-prime) 곡선 등이 있다. 예를 들면, N 곡선은 그림 4.(a)와 같이 문자 'N'을 재귀적으로 사용하여 2차원 공간의 모든 영역을 지나게 만든 곡선이다. 그림 4.(b)는 N 곡선이 지나는 영역을 순차적으로 정렬하는 N-순서화 기법을 사용하면 셀들을 1차원으로 정렬한 것으로, 이 순서대로 디스크에 순차적으로 저장하면 디스크 탐색시간을 줄일 수 있다[6]. N-순서화 기법에 따라 정렬된 셀들을 순서대로 그림 4.(c)와 같이 클러스터 단위(3이라고 가정)로 묶으면, 이 논문에서 제안하는 셀 클러스터링 알고리즘의 대안으로 사용될 수 있다. 그러나, 이 방법은 고정된 순서에 의해 하나의 클러스터를 구성하는 셀들의 선택되며, 셀간의 공간 관련성이 충분히 반영되지 못하는 단점을 갖는다. 예를 들어, 그림 4.(c)의 셀14와 셀3은 공간적으로 서로 인접하지 않으나 동일한 클러스터에 할당되고 있다. 즉, 함께 접근될 확률이 적은 셀이 동일한 클러스터에 할당됨으로써 디스크 접근 비용이 커지는 문제점을 갖는다.

그림 4 N-순서화 기법과 셀 클러스터링

이 논문은 정규분할 공간 색인에서 셀 분할선에 의해 중복 참조되는 객체의 저장위치의 결정하기 위한 객체 클러스터링과 공간적으로 서로 인접한 셀들을 선택하여 클러스터를 생성하는 셀 클러스터링 문제를 다룬다. 이 문제는 아직 연구된 바 없으며, 공간 채우기 곡선을 이용한 셀 클러스터링 알고리즘 고려할 수 있으나, 이 방법은 셀간의 공간 관련성을 충분히 반영하기 어려운 문제점을 가지고 있다.

3. 고정 그리드 파일을 이용한 공간 객체 저장 구조

고정 그리드 파일 색인은 셀 별로 공간 객체의 저장 위치를 저장하고 있는 디렉토리들로 구성된다. 각 디렉토리는 고정 그리드 파일을 구성하는 셀에 대해 일대일 관계를 갖는다. 디렉토리에는 자신이 대표하는 셀 내에 포함된 공간 객체의 대략적인 표현(approximations)과 정확한 표현(exact representations)에 대한 참조(디스크 상의 저장 위치에 대한 포인터)를 갖는다. 공간 객체에 대한 대략적인 표현은 최소 경계 사각형(MBR)을 의미한다. 고정 그리드 파일을 이용한 공간 객체 저장 구조는 디스크에 공간 객체를 저장하는 방법에 따라서 그림 5와 같이 객체 기반 저장 구조 모델, 페이지 기반 저장 구조 모델, 확장된 페이지 기반 저장 구조 모델, 그리고 클러스터 기반 저장 구조 모델로 표현될 수 있다.

그림 5.(a)는 객체 기반 저장 구조 모델을 나타내고 있다. 이 모델에서는 디스크에 모든 공간 객체들이 연속

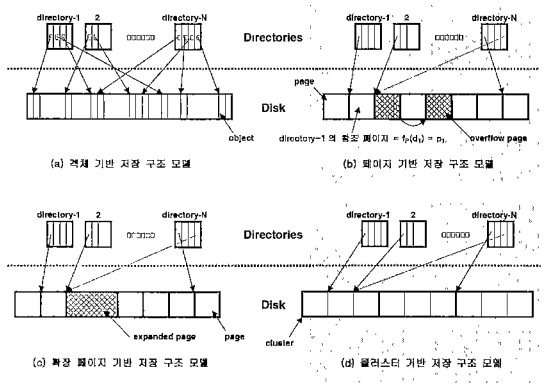


그림 5 고정 그리드 파일을 이용한 공간 객체 저장 구조

적으로 저장되며, 공간 색인의 디렉토리에는 디스크에서 공간 객체의 정확한 표현을 저장하고 있는 위치를 각각의 객체별로 참조하게 된다. 객체 기반 저장 구조 모델은 객체별로 디스크 접근이 발생하므로 디스크 접근 횟수가 많아지며, 요청된 공간 객체들이 디스크에서 떨어져 저장된 경우에, 디스크 탐색시간이 증가하는 단점을 가지므로, 여러 공간 객체들이 동시에 요구되는 응용에는 부적합하다.

그림 5.(b)는 페이지 기반 저장 구조 모델을 나타내고 있다. 이 모델에서는 객체 기반 저장 구조 모델과 달리 디스크를 고정 크기의 페이지 단위로 나누어서 공간 객체를 저장한다. 하나의 페이지 내에는 모두 동일한 셀에 포함된 객체들이 저장된다. 디렉토리에서는 디스크내의 하나의 페이지 주소를 참조하며, 이 주소의 페이지를 해당 디렉토리의 참조 페이지라 한다. 만약에 디스크내의 페이지가 자신을 참조하는 디렉토리 내의 모든 공간 데이타를 저장하지 못하는 경우에는 오버플로어 페이지를 가지게 된다. 예를 들어, 그림 5.(b)는 디렉토리 2번의 참조 페이지가 오버플로어 페이지를 가지고 있음을 나타내고 있다. 참조 페이지와 오버플로어 페이지는 디스크상에서 떨어져서 저장될 수 있다. 다음은 페이지 기반 저장 구조 모델을 표현하기 위한 정의이며, 그림 6에는 2x2로 전체 지도 영역을 분할한 경우 셀 집합, 디렉토리 집합, 페이지 집합과 매핑 함수의 예를 보이고 있다.

[정의 1] 전체 지도 영역를 MxN으로 분할할 때 각각의 셀은 그리드 파일 색인에서 하나의 디렉토리로 표현된다. 즉, 셀 집합 $C = \{ c_i \mid 0 \le i < MxN \}$의 포함된 셀 c_i는 그리드 파일 색인을 구성하는 디렉토리 집합 $D = \{ d_i \mid 0 \le i < MxN \}$에 포함된 디렉토리 d_i과 일대일 관계가 성립하며 함수 f_D로 정의된다.

$f_D : C \rightarrow D$,

즉, $f_D(c_i) = d_i$이다.

[정의 2] 셀 c_i에 포함된 공간 객체의 대략적인 표현과 정확한 표현이 저장되어 있는 곳의 참조는 디렉토리 d_i 내에 저장되어 있고, 공간 객체의 정확한 표현은 디스크내의 페이지 집합 $P = \{ p_i \mid 0 \le i < MxN \}$내의 페이지 p_i에 저장된다. 디렉토리 d_i와 페이지 p_i간에는 일대일 관계가 성립하며 함수 f_P로 정의되며, 페이지 p_i를 셀 c_i 또는 디렉토리 d_i의 참조 페이지라 한다.

$f_P : D \rightarrow P$,

즉, $f_P(d_i) = p_i$이다.

디렉토리는 자신에게 포함된 모든 객체에 대해서 객체의 정확한 표현이 저장되어 있는 위치를 참조 페이지의 시작 위치로부터의 상대적인 주소로 가진다. 그런데 다른 디렉토리가 참조하는 페이지에 저장된 중복 참조 객체의 경우에는 상대적인 주소만으로 공간 객체의 정확한 표현을 얻을 수 없다. 이 경우에는 중복 참조 객체가 저장된 페이지를 참조하는 디렉토리도 함께 기록함으로써, 상대적인 주소의 기준이 되는 페이지의 시작 위치를 얻을 수 있다. 그림 5.(b)에서 N번째 디렉토리에 포함된 공간 객체들은 대부분 실선 화살표가 가리키는 참조 페이지에 저장되어 있다. 그러나 2번째 디렉토리의 참조 페이지에 저장되어 있는 중복 참조 객체에 대해서는 디렉토리 2를 통해서 상대적인 주소의 시작 위치(그림에서 점선 화살표가 나타내고 있다)를 얻는다.

페이지 기반 저장 구조 모델은 객체 기반 저장 구조 모델에 비해서 셀 내에 포함된 공간 객체들은 디스크내에 페이지 단위로 클러스터링이 되는 장점을 있다. 반면에 데이타 파일 내의 페이지는 고정 크기를 가지므로 일부 사용되지 않는 영역이 발생하기 때문에, 객체 기반 저장 구조 모델보다 데이타 파일의 크기가 크다는 단점을 갖는다.

확장된 페이지 기반 저장 구조는 페이지 기반 저장 구조에서 발생하는 오버플로어 페이지를 제거하기 위해 그림 5.(c)와 같은 구조를 갖는다. 확장된 페이지 기반 저장 구조는 기본적으로는 페이지 기반 저장 구조와 동일하나 하나의 페이지가 자신을 참조하는 셀에 포함된 모든 공간 객체를 저장할 수 없는 경우에 모두 저장할 수 있을 만큼의 페이지를 디스크상의 연속된 공간에 할당해서 저장한다. 따라서 페이지 기반 저장 구조에서 발생한 오버플로어 페이지를 위한 추가의 디스크 탐색이 이루어지지 않는 장점을 가진다. 이 논문에서는 객체 클러스터링 알고리즘의 성능 평가 실험을 위해 페이지 기반 저장 구조 모델과 확장된 페이지 기반 저장 구조 모델을 사용한다.

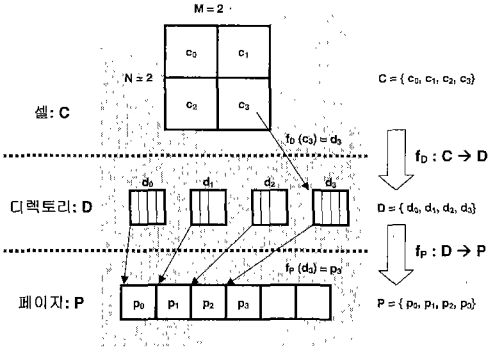


그림 6 [정의1]과 [정의2]의 예

마지막으로 클러스터 기반 저장 구조 모델은 디스크를 일정한 크기의 클러스터로 나누어 관리하는 방식이다. 클러스터는 여러 개의 페이지로 구성된다. 디렉토리에서는 페이지 기반 저장 구조 모델과 마찬가지로 참조 페이지를 가진다. 그러나 디스크 I/O는 클러스터 단위로 이루어지기 때문에, 자신이 참조한 페이지 이외에 다른 페이지도 함께 메모리로 전송된다. 만약 함께 전송된 다른 페이지가 메모리 버퍼에서 비워지기 전에 다음 질의에서 사용된다면 페이지 기반 저장 구조 모델에 비해 디스크 I/O를 줄일 수 있지만, 그렇지 않은 경우에는 메모리 버퍼를 쓸모 없이 사용하기 때문에 디스크 I/O가 증가할 수도 있다[1]. 따라서 동일한 클러스터에 참조 페이지를 갖는 셀들을 결정하는 것이 매우 중요하다. 이 모델은 5장에서 제시하는 셀 클러스터링 알고리즘을 위한 저장 구조로 사용한다.

4. 객체 클러스터링

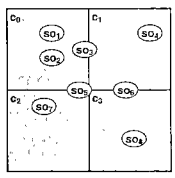
객체 클러스터링은 서로 인접한 공간 객체를 디스크 상에 물리적으로 서로 인접하게 저장하는 것이다. 이 논문에서는 공간 색인으로 고정 그리드 파일을 사용하기 때문에 각 그리드 셀에 포함된 공간 객체들을 디스크 내에 페이지 단위로 저장하는 페이지 기반 저장 구조 모델을 따른다. 그런데, 정규 분할 공간 색인의 경우 둘 이상의 셀에 의해 중복 참조되는 공간 객체는 어떤 디렉토리의 참조 페이지에 저장해야 하는가를 결정해야 한다. 다음은 중복 참조되는 객체에 대한 객체 클러스터링 알고리즘을 위한 정의이며, 그림 7에는 8개의 공간 객체에 대해서 포함 공간 객체 집합, 중첩 공간 객체 집합, 중첩 셀 집합, 포함 공간 객체 함수, 중첩 공간 객체 함수의 예를 보이고 있다.

[정의 3] SO는 전체 지도에 포함된 모든 공간 객체의 집합이며, 포함 공간 객체 집합 $SO^w = \bigcup_{i=0}^{M\times N-1} SO_i^w$ 와 중첩 공간 객체 집합 $SO^o = \bigcup_{i=0}^{M\times N-1} SO_i^o$ 의 합집합이다.

SO_i^w = { so_j | $so_j \in SO$, *Is-Within*(c_i, so_j) }, *Is-Within*(A, B)는 B가 A에 완전히 포함되는 가를 검사하는 함수이다.

SO_i^o = { so_j | $so_j \in SO$, *Is-Overlapped*(c_i, so_j) }, *Is-Overlapped*(A, B)는 A와 B가 어느 한 쪽에 포함되지 않으면서 중첩되는가를 검사하는 함수이다.

[정의 4] 공간 객체 $so_j \in SO^o$와 중첩되는 셀들은 중첩 셀 집합 OC_j = { c_i | *Is-Overlapped*(c_i, so_j) }으로 정의하며, 중첩 셀 집합은 $\forall so_j \in SO^o$에 대해서 하나씩 존재한다.



$SO = SO_0^w + SO_1^w + SO_2^w + SO_3^w + SO_0^o + SO_1^o + SO_2^o + SO_3^o$
$= \{so_1, so_2, so_3, so_4, so_5, so_6, so_7, so_8\}$

$SO_0^w = \{so_1, so_2\}$ $SO_0^o = \{so_3, so_5\}$

$SO_1^w = \{so_4\}$ $SO_1^o = \{so_3, so_5, so_8\}$

$SO_2^w = \{so_7\}$ $SO_2^o = \{so_8\}$

$SO_3^w = \{so_8\}$ $SO_3^o = \{so_5, so_8\}$

$OC_2 = \{c_0, c_1\}$, $OC_5 = \{c_0, c_1, c_2, c_3\}$, $OC_6 = \{c_1, c_3\}$

$f_{SO}^w(so_1) = c_1$ $f_{SO}^o(so_3) = OC_3 = \{c_1, c_2\}$

그림 7 [정의3], [정의4]과 [정의5]의 예

[정의 5] 포함 공간 객체 함수 f_{SO^w}는 공간 객체 $so_j \in SO^w$에 대해서 so_j를 포함하는 셀을 구하는 함수이고, 중첩 공간 객체 함수 f_{SO^o}는 공간 객체 $so_j \in SO^o$에 대해서 so_j에 중첩되는 중첩 셀 집합을 구하는 함수로 정의한다.

f_{SO^w}: SO → C, f_{SO^o} : SO → OC,

즉, $f_{SO^w}(so_j) = c_i$이며, $f_{SO^o}(so_j) = OC_i$이다.

이 논문에서 제안하는 객체 클러스터링 알고리즘에서는 SO^w에 속한 공간 객체 so_j는 데이타 페이지 $f_P(f_D(f_{SO^w}(so_j)))$에 저장된다. 객체 클러스터링 알고리즘의 핵심은 SO^o에 속한 객체들(즉, 중복 참조를 갖는 공간 객체들)을 어디에 저장할 것인가를 결정하는 것이다. 예를 들어, 공간 객체 $so_i \in SO^o$를 중복 참조하는 셀 집합 OC_i가 {c_{23}, c_{24}, c_{33}}일 때, so_j를 데이타 페이지 {p_{23}, p_{24}, p_{33}}중에서 어디에 저장해야 할 것인지를 결정해야 한다. 중복 참조 객체의 저장 위치 선택 기준은 이 논문에서 제시하는 여러 가지 객체 클러스터링 알고리즘에 따라 다르게 적용된다.

[정의 6] 객체 클러스터링 결합력(CCO: Clustering Coherence of Object)은 중복 참조 객체와 중첩 셀과의 결합 정도를 측정하는 척도로써, 중복 참조 객체에 대한 저장 위치 선택의 기준이 된다. 공간 객체 $so_j \in SO^o$와 셀 $c_i \in OC_j$간에는 객체 클러스터링 결합력 CCO_{ij}가 존재하며, CCO 값은 함수 f_{CCO}에 의해 맵핑된다.

f_{CCO} : SO^o, OC → CCO,

즉, $f_{CCO}(c_i, so_j) = CCO_{ij}$이다.

정의3~정의6를 이용한 객체 클러스터링 알고리즘은 그림 8과 같다. 객체 클러스터링 알고리즘의 입력은 전체 지도에 포함된 모든 공간 객체 집합 SO이다. 우선 정의3과 정의4에 따라 SO로부터 포함 공간 객체 집합 SO^w, 중첩 공간 객체 집합 SO^o, 그리고 중첩 셀 집합 OC를 생성(그림 8.(a))한 다음에 SO에 속한 모든 공간 객체를 클러스터링하여 데이타 파일 내에 페이지 단위

로 저장한다. 이 때, 공간 객체 so가 SO^w에 속한 객체인지(그림 8.(b)), 아니면 SO^o에 속한 객체인지(그림 8.(c))에 따라서 다른 처리 과정을 거친다. SO^w에 속한 공간 객체는 정의2에 따라 자신을 포함하고 있는 셀을 대표하는 디렉토리의 참조 페이지에 저장된다. SO^o에 속한 공간 객체의 경우 자신과 중첩되는 셀이 둘 이상이므로 공간 객체 결합력에 따라서 저장될 페이지가 결정된다. 따라서 SO^o에 속한 공간 객체는 대한 중첩 셀 집합에 속한 해당 셀(그림 8.(d))과의 CCO를 계산(그림 8.(e))하여 CCO가 가장 큰 셀에 할당된 페이지에 저장된다.

```
void  object_cluerstering (SpatialObjectSet SO)
{
      SpatialObject so,
      GridCell c;
      CellNode cn;
      DataPage p;
      float maxCCO, CCO;
      generate SO^w, SO^o, and OC; ..... (a)
      for (each so in SO) {
            if (so in SO^w) { ..... (b)
                  c = f_SO^w (so);
                  cn = f_cn(c);
                  p = f_p(cn);
                  store so to p;
                  update cn by so and p;
      } /* of for */
      for (each so in SO) {
            if (so in SO^o) { ..... (c)
                  maxCCO = 0;
                  for (each c in f_SO^o (so)) { ..... (d)
                        CCO = f_CCO(c, so); ..... (e)
                        if (CCO > maxCCO) {
                              maxCCO = CCO;
                              cn = f_cn(c),
                              p = f_p(cn);
                        } /* of if */
                  } /* of for */
                  store so to p;
                  for (each c in f_SO^o (so)) update f_cn(c) by so and p;
            } /* of if-else */
      } /* of for */
}; /* of object_cluerstering */
```

그림 8 객체 클러스터링 알고리즘

4.1 무게중심을 이용한 객체 클러스터링 알고리즘

이 알고리즘에서 f_{CCO}는 매개 변수로 사용된 셀과 공간 객체에 대해서 객체의 무게 중심이 셀에 포함된 경우에는 1을, 포함되지 않을 경우에는 0을 반환한다. 따라서 공간 객체의 무게중심을 이용하여 중복 참조 객체의 물리적인 저장 위치가 결정된다. 즉, 중복 참조 객체는 자신의 무게 중심을 포함하는 중첩 셀의 참조 페이지에 저장된다. 객체의 무게 중심은 객체의 유형에 따라서 다음과 같이 계산된다.

- 점 객체: 자신의 좌표
- 선 객체: 선을 이등분하는 좌표
- 면 객체: 면 객체의 가장 자리를 구성하는 모든 좌표의 평균값

[예 1] 그림 9.(a)에서 객체SO_1의 중첩 셀 집합 OC_1는 {c_0, c_1, c_4, c_5}이다. 이때 객체 SO_1와 각 중첩 셀과의 객체 클러스터링 결합력을 계산하면, $f_{CCO}(c_5, SO_1)$가 가장 큰 값을 가진다. 따라서 중복 참조 객체인 SO_1는 셀 c_5의 참조 페이지인 p_5에 물리적으로 저장된다. 그림 9.(a)에서 표현된 나머지 중복 참조 객체인 SO_2, SO_3에 대해서도 객체 SO_1와 같은 방법을 사용하면 객체 SO_2는 페이지 p_{11}, 객체 SO_3는 페이지 p_{13}에 각각 저장되어진다.

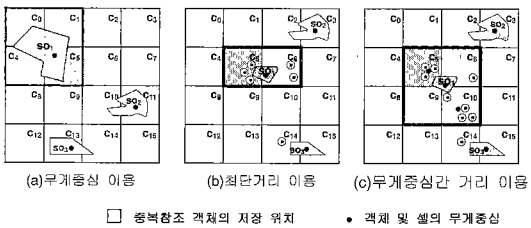


그림 9 객체 클러스터링 예제

4.2 최단거리를 이용한 객체 클러스터링 알고리즘

무게 중심을 이용한 방법은 공간 객체의 분포에 대한 정보를 고려하지 않고 단순히 공간 객체의 무게 중심만을 이용하기 때문에 다음과 같은 문제점을 갖는다. 그림 9.(b)에서 공간 객체 SO_1는 무게 중심은 셀 c_6에 존재한다. 그러나, GIS에서 공간 데이타간에 존재하는 공간 지역성(Spatial Locality)을 고려한다면, SO_1는 셀 c_5에 포함된 공간 객체들과 함께 접근될 확률이 높다.

최단거리를 이용한 방법에서 공간 객체간의 지역성을 이용하여 중복 참조 객체의 물리적 저장 위치가 결정된다. 객체의 저장 위치를 결정하는 f_{CCO}의 결과는 다음과 같다.

(1) 매개 변수로 사용된 셀이 가지는 포함 공간 객체와

매개 변수로 사용된 객체간의 최단 거리의 역을 반환한다.

(2) 매개 변수로 사용된 셀이 포함 공간 객체를 가지지 않을 경우에는 매개 변수로 사용된 객체의 무게 중심이 셀에 포함될 경우에 0을, 포함되지 않을 경우에는 -1을 반환한다.

[예 2] 그림 9.(b)에서 객체 so_1의 중첩 셀 집합 OC_1는 { c_5, c_6 }이다. 객체 so_1를 기준으로 최단거리를 이용하여 각 셀의 CCO를 계산하면 $f_{CCO}(c_5, so_1)$가 $f_{CCO}(c_6, so_1)$보다 크게 나타난다. 따라서 중복 참조 객체 so_1는 셀 c_5의 참조 페이지 p_5에 물리적으로 저장되어진다. 중복 참조 객체 so_2는 중첩 셀 집합 내에 포함된 객체가 없기 때문에 객체의 무게 중심을 포함하고 있는 셀 c_2의 참조 페이지에 저장된다. 반면에 so_3는 객체의 무게 중심은 셀 c_{15}에 있지만, $f_{CCO}(c_{14}, so_3)$가 $f_{CCO}(c_{15}, so_3)$보다 크기 때문에 셀 c_{14}의 참조 페이지에 저장된다.

4.3 무게중심간 거리를 이용한 객체 클러스터링 알고리즘

최단 거리를 이용한 객체 클러스터링 알고리즘의 CCO 계산에서는 중복 참조 객체와 가장 가까운 객체와의 거리만으로 공간 지역성을 평가하였다. 이 방법을 그림 9.(c)의 공간 객체 so_1에 적용하면 가장 가까운 공간 객체를 포함하고 있는 c_{10}이 선택된다. 그런데 셀 내의 공간 객체의 분포를 살펴보면 c_{10}보다 c_5에 포함된 객체와의 공간 지역성이 크다고 할 수 있다. 이러한 문제점은 공간 지역성을 평가하는데 하나의 공간 객체만을 고려하였기 때문에 발생하였다.

무게중심간 거리를 이용한 객체 클러스터링 알고리즘에서는 다음에서 정의하고 있는 셀의 무게 중심을 이용하여 공간 지역성을 평가한다. 즉, 중복 참조 객체의 중첩 셀 집합 중에서 중복 참조 객체의 무게 중심과 가장 가까운 무게 중심을 갖는 셀이 공간 지역성이 가장 크다고 평가된다. f_{CCO}는 매개 변수로 사용된 셀과 객체의 무게 중심간의 역을 반환한다. 만약 셀에 포함 공간 객체가 존재하지 않아서 셀의 무게 중심이 없는 경우에는 0을 반환한다.

[정의 7] 셀의 무게 중심은 셀이 포함하고 있는 모든 포함 공간 객체의 무게 중심의 평균으로 정의한다. 단, 포함 공간 객체를 포함하지 않는 셀의 무게 중심은 존재하지 않는다.

[예 3] 그림 9.(c)에서 객체 so_1의 중첩 셀 집합 OC_1는 { c_5, c_6, c_9, c_{10} }이다. 이때 객체 so_1와 각 중첩 셀과의 객체 클러스터링 결합력은 객체의 무게 중심과 셀의 무게 중심의 거리가 가장 가까운 c_5에 대해 가장 큰 값을 나타낸다. 따라서 중복 참조 객체인 so_1는 셀 c_5의 참조 페이지 p_5에 물리적으로 저장되어진다. 반면에, 동일한 환경에서 무게 중심을 이용한 방법에서는 c_6이, 최단 거리를 이용한 방법에서는 c_{10}이 가장 CCO가 크게 나타난다.

5. 셀 클러스터링

셀 클러스터링은 서로 인접한 셀들을 클러스터라는 단위로 묶어 디스크 상에 물리적으로 서로 인접하게 저장하는 것이다. 셀 클러스터링 알고리즘의 핵심은 (1)클러스터로 묶이는 셀들을 선택하는 기준과 (2)정해진 기준에 따라 셀들을 클러스터로 묶는 방법이다. 이 논문에서는 셀들을 선택하는 기준으로 다음에 정의한 셀 클러스터링 결합력을 이용한다. 그림 10은 3x3으로 전체 지도 영역을 분할한 경우에 9개의 셀에 대해서 인접 셀 쌍과 인접 셀 쌍 집합, 셀 클러스터링 결합력 함수의 예를 보이고 있다.

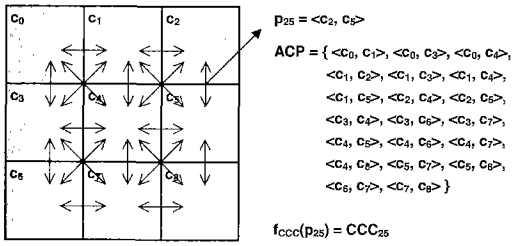


그림 10 [정의8]과 [정의9]의 예

[정의 8] 셀 $c_i \in C$와 셀 $c_j \in C$가 서로 한 점 이상에서 만날 때, c_j는 c_i의 인접 셀이라 한다. 가장 자리에 있는 셀을 제외한 모든 셀은 8개의 인접 셀을 갖는다. p_{ij} = $<c_i, c_j>$를 인접 셀 쌍이라 하고, 인접 셀 쌍 집합 ACP(Adjacent Cell Pairs)는 다음과 같이 정의된다. (단, i<j임)

ACP = { p_{ij} | i<j, $c_i \in C$, $c_j \in C$, *Is_Adjacent*(c_i, c_j) }, *Is_Adjacent*(A, B)는 A와 B가 서로 인접 셀인가를 검사하는 함수이다.

[정의 9] 서로 인접한 셀 간에는 셀 클러스터링 결합력(CCC: Clustering Coherence of Cell)이 존재하며, 두 셀 간의 공간 지역성이 클수록 셀 클러스터링 결합력이 커지는 특성을 갖는다. 셀 c_i와 자신의 인접 셀 c_j간에는 셀 클러스터링 결합력 CCC_{ij}가 존재하며, CCC 값은 함수 f_{CCC}에 의해 맵핑된다.

f_{CCC} : ACP → CCC,

즉, $f_{CCO}(p_{ij})$ = CCC_{ij}이다.

이 논문에서 제시하는 셀 클러스터링 결합력을 이용한 셀 클러스터링 알고리즘은 다음과 같다.

I. 클러스터 생성 준비 단계

I.1 인접 셀 쌍 집합(ACP)을 구한다.그림 10과 같이 3x3의 고정 그리드 파일에서 ACP는 20개의 인접 셀 쌍으로 구성된다.

I.2 각각의 인접 셀 쌍에 대한 CCC를 계산한다. CCC 값은 셀 클러스터링 알고리즘(5.1~5.4절 참조)에 따라서 다르게 계산된다.

I.3 인접 셀 쌍을 CCC 값에 따라 내림차순으로 정렬한다

II. 클러스터 생성 단계

II.1 ACP에서 CCC값이 가장 큰 항목 p_{ij}을 선택해 ACP에서 제거한다. 더 이상의 항목이 없으면 종료한다. (그림 11.(e)는 클러스터 생성 단계가 종료한 상태이다.)

II.2 c_i와 c_j를 연결할 경우 연결된 셀들의 전체 참조 페이지의 크기가 클러스터 크기보다 크면 II.1로 돌아간다.

II.3 c_i와 c_j를 링크로 연결하고 II.1로 돌아간다. (그림 11.(d)는 링크를 네 번 연결한 상태이다.)

III. 클러스터 저장 단계

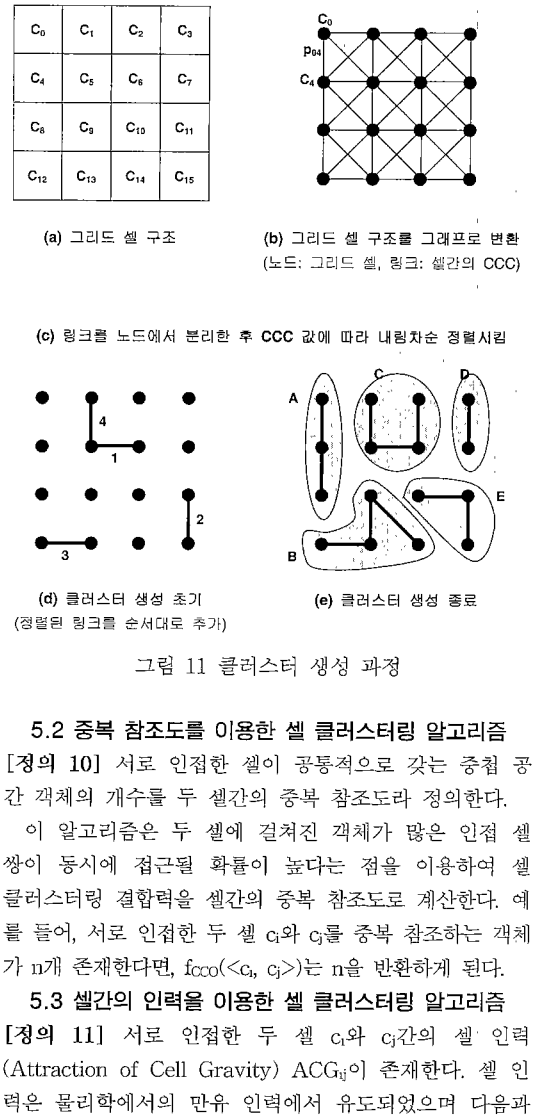
III.1 전체 그리드 셀에 대해서 N순서화 기법에 따라 셀을 하나씩 선택한다. 더 이상 선택할 셀이 없으면 종료한다.즉, 이 논문에서는 생성된 여러 클러스터간의 저장 순서는 N 순서화 기법을 따른다. 그림 11.(e)에서는 5개의 클러스터가 생성되었으며, 클러스터간의 저장 순서는 N 곡선($c_0 \rightarrow c_4 \rightarrow c_1 \rightarrow c_5 \rightarrow c_8 \rightarrow c_{12} \rightarrow c_9 \rightarrow c_{13} \rightarrow c_2 \rightarrow c_6 \rightarrow c_3 \rightarrow c_7 \rightarrow c_{10} \rightarrow c_{14} \rightarrow c_{11} \rightarrow c_{15}$)에 따라서 A, C, B, D, E가 된다.

III.2 선택된 셀을 포함하고 있는 클러스터가 디스크에 이미 저장된 경우라면 III-1로 돌아간다.
즉, N곡선의 순서에 따라서 처음 선택된 셀 c_0를 포함한 클러스터 A를 디스크에 저장한 후에 다음에 셀 c_4 또는 셀 c_8을 방문했을 때는 클러스터 A를 반복 저장할 필요가 없다.

III.3 선택된 셀을 포함하고 있는 클러스터를 디스크에 저장하고 III.1로 돌아간다.

5.1 셀의 무게중심간 거리를 이용한 셀 클러스터링 알고리즘

이 알고리즘에서 f_{CCC}는 셀의 무게 중심(정의7 참조)간의 거리를 이용한다. 즉, 매개 변수로 사용된 두 셀의 무게 중심간의 거리의 역을 반환하여 무게 중심이 가까운 인접 셀 쌍의 셀 클러스터링 결합력을 높게 만든다. 그리고 매개 변수로 사용된 셀 중에 포함 공간 객체를 포함하지 않는 셀이 하나 포함된 경우에는 0을 반환하고, 둘을 포함한 경우에는 -1을 반환하여 셀 클러스터링 결합력을 최소화한다. 이 방법은 셀의 무게 중심이 셀에 포함된 공간 객체의 개수와 분포를 충분히 반영하지 못하는 단점이 갖는다.



(a) 그리드 셀 구조

(b) 그리드 셀 구조를 그래프로 변환 (노드: 그리드 셀, 링크: 셀간의 CCC)

(c) 링크를 노드에서 분리한 후 CCC 값에 따라 내림차순 정렬시킴

(d) 클러스터 생성 초기 (정렬된 링크를 순서대로 추가)

(e) 클러스터 생성 종료

그림 11 클러스터 생성 과정

5.2 중복 참조도를 이용한 셀 클러스터링 알고리즘

[정의 10] 서로 인접한 셀이 공통적으로 갖는 중첩 공간 객체의 개수를 두 셀간의 중복 참조도라 정의한다.

이 알고리즘은 두 셀에 걸쳐진 객체가 많은 인접 셀 쌍이 동시에 접근될 확률이 높다는 점을 이용하여 셀 클러스터링 결합력을 셀간의 중복 참조도로 계산한다. 예를 들어, 서로 인접한 두 셀 c_i와 c_j를 중복 참조하는 객체가 n개 존재한다면, $f_{CCO}(<c_i, c_j>)$는 n을 반환하게 된다.

5.3 셀간의 인력을 이용한 셀 클러스터링 알고리즘

[정의 11] 서로 인접한 두 셀 c_i와 c_j간의 셀 인력(Attraction of Cell Gravity) ACG_{ij}이 존재한다. 셀 인력은 물리학에서의 만유 인력에서 유도되었으며 다음과 같이 정의한다. 즉, 만유 인력 공식에서 물체의 질량은 셀에 포함된 객체의 면적의 합으로, 물체간의 거리는 셀의 무게 중심간 거리로 변경시켜 적용한다.

$$ACG_{ij} = k\frac{A_iA_j}{d^2}$$

단, k는 상수, d는 두 셀의 무게 중심간 거리(*distance*(*gravity*(c_i), *gravity*(c_j)))이고, A_i는 c_i에 속한 모든 공간 객체의 면적의 합이다.

이 알고리즘에서는 셀 클러스터링 결합력으로 셀 인력을 사용한다. 따라서 셀 인력이 큰 인접 셀이 동일한 클러스터에 포함되게 된다. 셀 인력은 클러스터를 만들기 위해 인접 셀 쌍을 정렬하는 과정에서 상대적인 수치로 이용되기 때문에 상수 k는 제거될 수 있다. 또한 공간 객체의 크기가 일정한 경우에는 객체 면적의 합 대신에 객체 수가 사용될 수 있다.

셀간의 인력 계산을 위해서 객체의 면적 계산이 필요하지만, 점 객체 및 선 객체의 경우 면적을 가지지 않기 때문에 이 알고리즘을 적용하기 어렵다. 이 경우에도 공간 객체의 면적 대신에 객체의 수가 사용되어야 한다.

5.4 포함 공간 객체의 개수를 이용한 셀 클러스터링 알고리즘

이 알고리즘에서 셀 클러스터링 결합력은 인접 셀 쌍이 가지는 포함 공간 객체의 개수의 합으로 계산한다. 예를 들어, 서로 인접한 두 셀 c_i와 c_j에 각각 m개와 n개의 포함 공간 객체가 존재한다면, $f_{CCO}(<c_i, c_j>)$는 $(m+n)$을 반환하게 된다.

이 알고리즘은 앞서 언급했던 여러 알고리즘과는 달리 인접 셀 쌍간의 공간적인 관련성을 전혀 고려하지 않고 단순히 공간 객체의 개수라는 정보만을 이용하고 있다. 이 논문에서 이 알고리즘을 제시하는 이유는 막연히 알고리즘의 추가를 목적으로 하는 것이 아니라, 인접 셀 쌍간의 공간 관련성을 고려한 알고리즘과 그렇지 않은 알고리즘간의 차이점을 알아보기 위해서이다.

이 논문에서 제시하는 클러스터 생성 방법은 기본적으로 인접 셀 쌍이라는 공간적으로 인접한 두 셀을 사용한다. 따라서 이 알고리즘이 CCC 값을 계산하는 과정에서 공간 관련성 정보를 이용하지 않은 방법일지라도 클러스터링을 통한 성능 향상을 기대할 수 있다. 6장에서는 실험을 통해서 N-순서화 기법을 이용한 클러스터 생성 방법(N-순서화 기법에 따라 셀을 정렬한 후에 클러스터의 크기에 맞게 셀들을 나눔)과의 차이를 통해서 이 논문에서 제시하는 클러스터 생성 방법의 성능을 입증한다.

6. 실험 평가

6.1 실험 환경

이 절에서는 실제 실험을 통해서 4장과 5장에서 제시한 클러스터링 알고리즘들의 성능을 평가한다. 표 2는 이 실험에서 사용할 클러스터링 방법론에 대한 표기법이다. 평가를 위한 테스트 데이타는 미국의 Geological Survey's Geographic Information Retrieval and Analysis System (GIRAS)로부터 추출한 Sequoia 2000 storage benchmark를 위한 다각형 데이타를 기반으로 편집하였다. 전체 17048개의 객체로 구성되어 있으며 4K 크기의 페이지 기반 저장 구조 모델을 사용하여 고정 그리드 파일 색인을 생성하였을 때, 색인 파일의 크기는 534K이며, 데이타 파일의 크기는 28,146K이다.

표 2 클러스터링 방법론 표기 방법

클러스터링 방법론	표기
객체의 무게중심(Gravity)을 이용한 객체 클러스터링	O(G)
무게 중심간 거리(Distance)를 이용한 객체 클러스터링	O(D)
최단 거리(Shortest)를 이용한 객체 클러스터링	O(S)
임의(Random) 객체 클러스터링	O(R)
셀의 무게중심간 거리(Distance)를 이용한 셀 클러스터링	C(D)
중복 참조도(Multiple reference)를 이용한 셀 클러스터링	C(M)
셀간의 인력(Attraction)을 이용한 셀 클러스터링	C(A)
포함 공간 객체의 개수(Count)를 이용한 셀 클러스터링	C(C)
N-순서화 기법을 이용한 셀 클러스터링	C(N)

이 논문에서 제안한 여러 클러스터링 알고리즘의 성능 평가를 위해 다각형의 면 객체만으로 실험한 이유는 다음과 같다. 2차원 공간 데이타베이스에서 데이타 형태는 점, 선, 면의 세 가지 종류가 있다. 이 중에서 점 객체는 작은 크기의 면 객체로 고려할 수 있으므로, 유사한 결과가 예상된다. 선 객체는 면 객체에 비해서 한 쪽 방향으로 길며, 더 많은 셀에 중첩이 될 확률이 크다는 특징이 있다. 그러나, 대부분의 GIS에서 선 객체는 세크먼트라는 단위로 분할되어 관리/저장되고 있으며, 각각의 세그먼트는 약간 긴 모양의 면 객체로 고려할 수 있으므로, 유사한 결과가 예상된다.

생성한 고정 그리드 색인 파일은 50x50개의 셀로 구성되며 노드 사용률은 표 3과 같이 객체 클러스터링 알고리즘에 따라 달라진다. 객체 클러스터링 알고리즘에 따라 저장 위치가 결정되어야 하는 중복 참조 객체의 수는 652개로 전체의 3.8%에 해당한다.

표 3 50x50 그리드 색인 파일에서 알고리즘 별 노드 사용률

	O(G)	O(D)	O(S)	O(R)
노드 사용수	869	858	858	888
노드 사용률	35%	34%	34%	36%

표 4 질의 집합의 특성

	평균객체수 (밀집 영역)	평균객체수 (희박 영역)	평균객체수 (전체)
0.10%	652.2	82.6	367.4
0.25%	1110.7	113.7	612.2
0.50%	1325.9	228.9	777.4
1.00%	1821.3	311.2	1066.3
4.00%	3477.2	440.9	1959.0
9.00%	5208.1	843.4	3025.7

실험에 사용할 질의는 영역 질의로써 주어진 최소 경계 사각형(MBR)과 교차하거나 포함된 객체를 검색하는 것이다. 질의 집합은 질의 영역의 크기가 0.1%, 0.25%, 0.5%, 1%, 4%, 9%인 경우와 질의 영역내의 데이타가 밀집한 영역과 데이타가 희박한 영역으로 총 12가지로 구성되며 질의 집합의 특성은 표 4와 같다.

실험 결과는 각각의 질의 집합에 대해서 20개의 질의 영역으로 실험한 결과를 합하여 나타낸다. 실험 결과로 측정할 데이타는 질의를 처리하기 위해서 발생한 디스크 접근 횟수이다. 질의 시간은 디스크 접근 시간만을 디스크 접근 횟수와 전송 데이타의 크기로 계산해서 사용하며, 계산 시간은 고려 대상에서 제외한다.

디스크 I/O 시간은 디스크 탐색 시간, 디스크 회전 지연 시간과 데이타 전송 시간으로 구성된다. 이 논문에서는 시간의 기준 단위로써 4K 데이타를 전송하는 시간을 사용하며 T라고 정의한다. 만약 디스크 탐색 시간과 회전 지연 시간의 합이 T의 C배이고 크기가 (4S)K인 데이타를 N번 전송한다면, 전체 디스크 접근 시간은 다음과 같이 계산된다.

- 디스크 접근 시간 = 디스크 접근 횟수(탐색시간+회전지연시간+데이타전송시간) = N (C + S)

예를 들어 C가 10일 때, 4K 데이타를 8번 읽은 경우에는 8(10+1)으로 80(T)만큼의 시간이, 32K 데이타를 1번 읽은 경우에는 1(10+8)으로 18(T)만큼이 시간이 필요하다. 즉, 동일한 크기의 데이타를 전송하더라도 전송하는 횟수와 데이타 크기에 따라 전체 디스크 접근 시간은 매우 큰 차이를 보인다. 이것은 데이타 전송시간이 탐색시간과 회전지연시간보다 매우 적기 때문이다.

표 5에는 [11]를 바탕으로 여러 하드디스크에 대한 C 값을 계산한 결과를 보이고 있다. 고속의 하드디스크일수록 C값이 크게 나타남을 알 수 있다. C값이 클수록 클러스터링으로 인한 디스크 접근 시간은 더 줄어들게 된다. 이 논문에서는 Western Digital Caviar AC2850를 기준으로 C 값을 10으로 하여 디스크 접근 시간을 계산한다. 이 값은 [9]에서 사용한 수치이다.

표 5 하드디스크의 특성

하드 디스크 종류	탐색시간	회전지연	전송시간 (4KB/msec)	C
Segate Cheetah 9	8	3	0.282	38.96
Western Digital Caviar AC22100	12	6	1.465	12.29
Western Digital Caviar AC2850	10	6 6	1.693	9.80

질의 수행을 위한 디스크 접근 횟수는 메모리 버퍼의 크기에 크게 의존한다. 이 논문에서는 다양한 실험을 위해서 크기가 32K에서 1024K까지의 메모리 버퍼를 사용한다. 메모리 버퍼에서의 페이지 교체 전략은 LRU (Least Recently Used)를 사용하였다. LRU는 현재 메모리 버퍼가 차서 새로운 페이지를 버퍼에 추가할 수 없는 경우에 가장 오랜 시간 동안 사용되지 안았던 페이지를 선택하여 버퍼에서 제거한 후에 디스크로부터 새로 읽어온 페이지를 버퍼에 추가하는 방식이다.

6.2 객체 클러스터링 실험 결과

이 절에서는 중복 참조 객체의 저장 위치를 결정하는 방법에 따른 질의 처리 성능을 질의 영역의 크기에 따라, 질의 영역의 특성에 따라, 메모리 버퍼의 크기에 따라 비교 평가한다. 공간 객체 저장 구조 모델로는 페이지 기반 모델을 사용한다. 비교 평가의 대상이 되는 알고리즘은 4장에서 제시한 3가지 객체 클러스터링 방법론과 중복 객체의 저장 위치를 임으로 결정하는 방법론이다.

그림 12는 32K 메모리 버퍼를 사용할 때, 질의 영역의 크기에 따라 질의 처리에 필요한 디스크 접근 횟수를 측정한 결과를 보여준다. 디스크 접근 횟수는 데이타 파일에서 하나의 페이지를 메모리 버퍼로 읽어온 횟수를 의미한다. 실험 결과 O(S)가 질의 영역 크기가 모든 경우에 우수한 성능을 나타내고 있다. 4.3절에서 O(D) 알고리즘은 O(S) 방법의 문제점을 개선하기 위한 방법

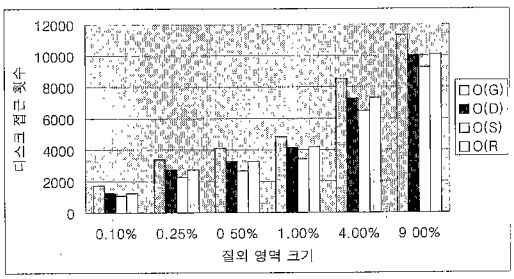


그림 12 질의 영역의 크기에 따른 객체 클러스터링 성능 평가(32K버퍼사용)

으로 제안하였으나, 실험 결과 데이타 희박 지역에서 약간의 좋은 성능을 보였을 뿐, 데이타 밀집 지역에서는 오히려 더 나쁜 성능을 보였다. 그 이유는 다음과 같이 설명될 수 있다. O(D) 알고리즘을 제안할 때 제시한 O(S) 알고리즘의 문제점은 중복 참조 객체와 중첩 셀간의 공간 지역성을 평가할 때, 가장 가까운 객체만 고려할 뿐, 셀 내의 모든 공간 객체의 분포를 고려하지 못한다는 것이다. 즉, O(S) 알고리즘은 중첩 셀 내의 대부분의 공간 객체가 중복 참조 객체와 멀리 떨어져 있더라도 하나의 객체만이 중복 참조 객체의 근처에 있는 경우에 공간 지역성이 크다는 평가를 받는 문제점을 가지고 있다. O(D) 알고리즘은 이 문제를 해결하기 위해서 중첩 셀 내의 모든 공간 객체의 무게 중심과 중복 참조 객체의 무게 중심간의 거리를 고려하여 공간 지역성을 평가하였다. 그런데 데이타가 밀집한 지역에서는 O(S) 알고리즘의 문제점이 크게 발생하지 않는다. 왜냐하면 데이타 밀집 지역에서는 셀 내의 많은 객체들이 밀집해 있으므로, 중첩 셀 내에 중복 참조 객체와 인접한 소수의 객체와 멀리 떨어진 다수의 객체가 포함될 확률이 적기 때문이다.

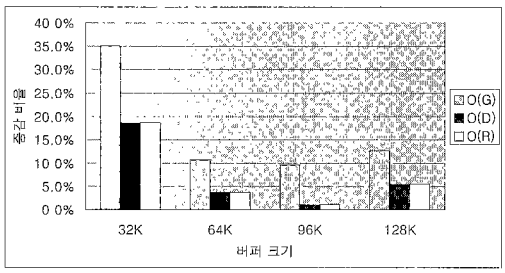
표 6의 O(S)를 사용한 경우의 디스크 접근 횟수를 기준으로 다른 알고리즘을 사용한 경우에 디스크 접근 횟수의 증감 비율을 질의 영역의 특성과 크기에 따라서 보여준다. 이 데이타는 메모리 버퍼 크기를 32K(즉, 페이지 8개를 메모리에서 가질 수 있음)로 하여 실험한 결과이다. 데이타 밀집 지역인 경우에 디스크 접근 횟수의 차이가 데이타 희박 영역인 경우보다 더 큼을 알 수 있다. 왜냐하면 데이타 희박 지역의 경우 데이타 밀집지역에 비해 중복 참조 객체의 수도 상대적으로 적기 때문에, 중복 객체의 저장 위치 결정 방법에 따른 성능의 차이가 줄어들기 때문이다.

표 6 질의 영역의 특성에 따른 디스크 접근 횟수 증감 비율 (O(S)를 기준, 32K버퍼사용)

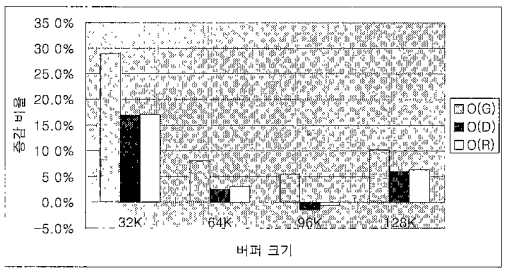
	질의 영역 크기	O(G)	O(D)	O(R)
데이터 밀집 지역	0.10%	38.0%	12.3%	12.3%
	0.25%	35.4%	19.3%	19.3%
	0.50%	37.7%	20.1%	20.1%
	1.00%	32.5%	19.9%	19.9%
	4.00%	27.2%	12.7%	12.9%
	9.00%	21.5%	9.5%	9.7%
데이터 희박 지역	0.10%	6.2%	0.0%	0.0%
	0.25%	4.3%	-0.6%	0.6%
	0.50%	2.3%	3.2%	5.1%
	1.00%	-1.2%	-5.7%	-3.6%
	4.00%	2.5%	0.8%	2.9%
	9.00%	3.2%	0.7%	2.8%

디스크 접근 횟수는 메모리 버퍼의 크기에 따라 달라진다. 다음 실험은 메모리 버퍼의 크기 가 각각 32K, 64K, 96K, 128K일 때, 각 알고리즘 별로 디스크 접근 횟수를 측정한다. 그림 13은 O(S)에 대한 다른 알고리즘의 디스크 접근 횟수의 증감 비율을 메모리 버퍼 크기의 변화에 따라 보여준다. 사용한 질의 영역의 크기는 전체 지도 크기의 0.5%와 1.0%이다.

중복 객체의 저장 위치를 결정하기 위한 객체 클러스터링 방법론에 대한 실험에서 가장 중요하게 비교해야 하는 것은 이 논문에서 제안한 알고리즘 중에서 가장 좋은 성능을 보이는 방법과 임의 위치에 중복 객체를 저장한 방법과의 성능 차이이다. 그림 13의 실험 결과를 살펴보면, O(S)가 O(R)에 비해 메모리 버퍼 크기가 32K인 경우에는 15% 이상의 성능 향상을 가져왔지만 메모리 버퍼가 32K보다 큰 경우에는 5%내외의 성능 향상을 가져왔음을 알 수 있다. 특히, 96K 크기의 버퍼를 사용하고, 질의 영역이 1.0%인 경우에는 O(R)이 더 적게 디스크 접근을 했음을 알 수 있다. 결론적으로 객체 클러스터링(특히 O(S))은 메모리 버퍼가 적은 경우에는 많은 성능 향상이 기대되지만, 메모리 버퍼가 큰 경우에는 O(R)에 비해 많은 성능 향상을 가져오지 못한다.



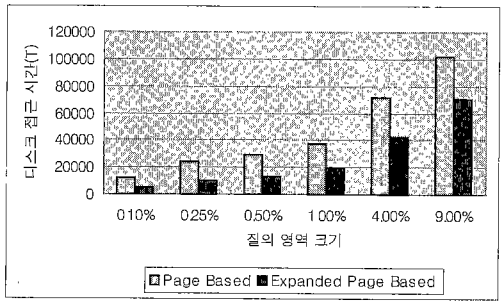
(a) 질의 영역 크기가 0.50%인 경우



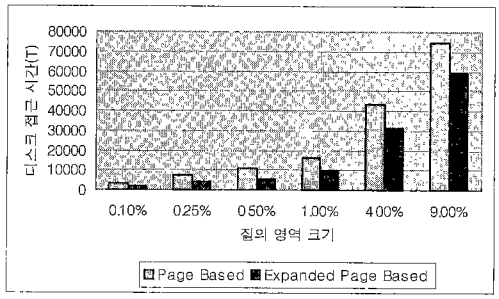
(b) 질의 영역 크기가 1.00%인 경우

그림 13 버퍼 크기에 따른 디스크 접근 횟수 증감 비율 (O(S)를 기준)

마지막 객체 클러스터링 실험은 페이지 기반 저장 구조 모델과 확장된 페이지 기반 저장 구조 모델의 성능 평가이다. 저장 구조 모델이 다른 경우에는 한번의 디스크 접근으로 전송되는 데이타 크기가 서로 다르기 때문에 디스크 접근 횟수만으로는 두 모델의 성능을 비교할 수 없다. 따라서 그림 14은 객체 클러스터링 알고리즘으로 O(S)를 사용한 경우에 두 저장 구조 모델간의 질의 접근 시간(T)을 비교하여 보여주고 있다. 실험 결과 확장된 기반 저장 구조 모델이 페이지 기반 저장 구조 모델에 비해 매우 우수한 성능을 나타냄을 알 수 있다. 버퍼 크기가 32K인 경우에는 질의 영역의 크기에 따라 30.3%~60.2%, 버퍼 크기가 128K인 경우에는 20.8%~49.2%의 성능 개선이 되었다.



(a) 데이타 버퍼 크기가 32K인 경우



(b) 데이타 버퍼 크기가 128K인 경우

그림 14 페이지 기반 저장 구조 모델과 확장된 페이지 기반 저장 구조 모델의 비교(O(S) 사용)

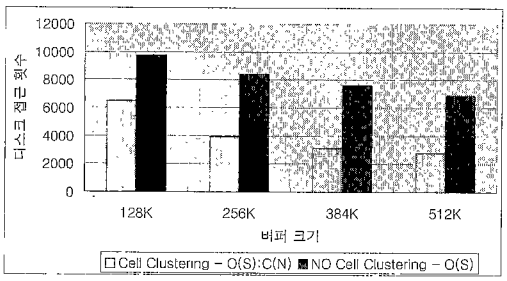
6.3 셀 클러스터링 실험 결과

이 절에서는 클러스터 기반 저장 구조 모델을 사용하여 동일한 클러스터내에 참조 페이지를 가지는 셀을 선택하는 방법에 따른 질의 처리 성능을 질의 영역의 크기에 따라, 질의 영역의 특성에 따라, 메모리 버퍼의 크기에 따라, 클러스터의 크기에 따라 비교 평가한다. 비교 평가의 대상이 되는 알고리즘은 5장에서 제시한 4가지 셀 클러스터링 알고리즘과 N-순서화 기법을 이용한 클러스터링 알고리즘으로 다음과 같은 특징을 갖는다.

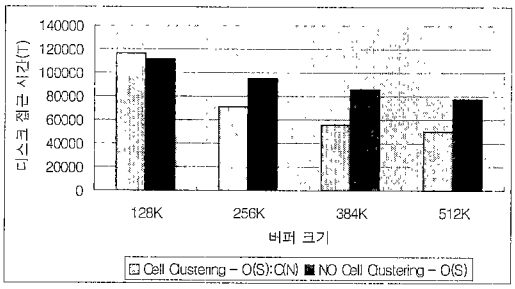
표 7 셀 클러스터링 알고리즘의 특징

알고리즘	동일한 클러스터에 포함된 모든 셀들의 연결성	셀 선택 기준
C(D), C(M), C(A)	모두 연결되어 있음	인접 셀 쌍간의 공간 관련성 (무게 중심, 중복 참조도, 셀 인력)
C(C)		포함 공간 객체의 수
C(N)	서로 떨어져 있을 수도 있음	N-순서화에 따른 순서 정보

이 논문에서 제안하는 셀 클러스터링의 성능을 비교하기 이전에 셀 클러스터링의 효과를 셀 클러스터링을 한 경우와 안 한 경우의 비교 실험을 통해서 살펴보겠다. 중복 객체의 저장위치를 결정하는 객체 클러스터링은 O(S)를 사용하였으며, 클러스터의 크기는 32K를 사용하였다. 이 실험에서 셀 클러스터링은 N-순서화 기법에 따라 셀을 정렬한 후에 클러스터의 크기에 맞게 셀들을 나누는 C(N)을 사용한다. 그림 15은 디스크 접근 횟수와 디스크 접근 시간의 차이를 버퍼 크기에 따라 보이고 있다. 셀 클러스터링에서 디스크 접근 횟수는 데이타 파일에서 하나의 클러스터를 메모리 버퍼로 읽어온 횟수를 의미한다.



(a) 디스크 접근 횟수 비교



(b) 디스크 접근 시간 비교

그림 15 셀 클러스터링을 한 경우와 안 한 경우의 비교 (32K 클러스터 사용, 전체 질의 집합)

이 실험에서의 디스크 접근 횟수와 디스크 접근 시간은 셀 클러스터링을 한 경우에는 클러스터 기반 저장 구조 모델에서 측정하였고, 셀 클러스터링을 하지 않은 경우에는 확장된 페이지 기반 저장 구조 모델에서 측정하였다. 실험 결과 128K 버퍼를 사용할 경우에는 디스크 접근 시간이 클러스터를 안 한 경우가 더 적지만, 버퍼 크기가 커질수록 셀 클러스터링을 한 경우의 성능이 훨씬 좋아짐을 알 수 있다. 디스크 접근 시간은 256K, 384K, 512K 버퍼를 사용한 경우에 각각 25%, 34.8%, 35.6% 단축되었다.

작은 크기의 버퍼를 사용한 경우에 셀 클러스터링을 한 경우의 성능이 더 나쁜 이유는 셀 클러스터링으로 인한 이득이 LRU 버퍼에 의한 이득보다 적기 때문이다. 즉, 동일한 클러스터에 포함된 페이지들 중에서 아직 사용되지 않은 페이지가 버퍼의 많은 공간을 차지하고 있기 때문에 버퍼에 페이지 없음(page fault)이 자주 발생하기 때문이다. 그러나 버퍼 크기가 커지게 되면 사용되지 않은 페이지가 버퍼에서 제거되지 않고, 나중에 다시 사용될 확률이 커지게 된다.

다음 실험에서는 이 논문에서 제안한 셀 클러스터링 알고리즘과 N-순서화 기법을 이용한 클러스터링 알고리즘의 성능을 비교해 본다. 객체 클러스터링 알고리즘에 따라 중복 객체의 저장 위치가 서로 다르게 결정되기 때문에, 서로 다른 객체 클러스터링을 함께 적용시켜 실험하였다. 그림 16은 객체 클러스터링 방법과 셀 클러스터링 방법을 각각 적용시킨 경우에, 모든 질의 집합의 질의 수행 성능을 128K 버퍼를 사용하여 실험한 결과이다.

이 실험을 통해서 증명된 사실은 다음과 같다. 첫째, 이 논문에서 제시한 셀 클러스터링 알고리즘이 N-순서화 기법을 이용한 클러스터링 알고리즘에 비해서 우수한 성능을 나타낸다. 특히 C(C)의 경우 인접 셀 쌍간의 공간 관련성을 고려하여 셀 클러스터링 결합력(CCC)을 계산하지는 않았지만, 클러스터를 생성하는 과정에서 서로 인접한 셀들만을 대상으로 하였기 때문에 클러스터를 구성하는 모든 셀들은 서로 연결되어 있다. 이 방법은 동일한 클러스터에 포함된 셀들이 서로 떨어져 있을 수 있는 N-순서화 기법을 이용한 클러스터 생성 방법에 비해 훨씬 효과적이다. 둘째, 단순히 포함 공간 객체의 개수 정보만을 이용한 C(C)보다는 인접 셀 쌍 간의 공간 관련성을 이용한 C(A), C(D), C(M)이 더 우수한 성능을 나타낸다. 세째, O(S)를 함께 사용한 경우가 전반적으로 좋은 성능을 나타내고 있으나, 객체 클러스터링의 결과가 전체 성능에 큰 영향을 주지는 않는다. 따라서 이 절의 다음에 기술하는 모든 실험에서 중복 개체의 저장 위치는 O(S)를 사용하여 결정한다.

그림 17는 128K 메모리 버퍼를 사용할 때, 질의 영역의 크기에 따라 질의 처리에 필요한 디스크 접근 횟수를 측정한 결과를 보여준다. 실험결과는 질의 영역의 크기가 전체의 0.10%, 0.25%, 0.50%, 1.00% 일 때에는 C(M)이 가장 좋은 성능을 보이고 있지만, 질의 영역의 크기가 4.00%, 9.00%인 경우에는 C(D)가 가장 좋은 성능을 보인다.

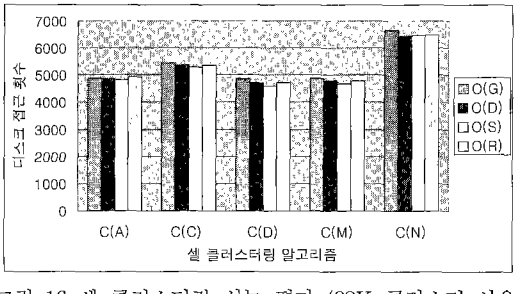


그림 16 셀 클러스터링 성능 평가 (32K 클러스터 사용, 128K 버퍼 사용, 전체 질의 집합)

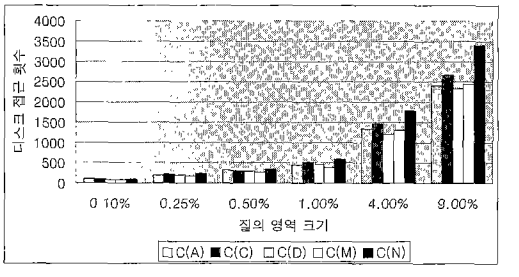


그림 17 질의 영역의 크기에 따른 셀 클러스터링 성능 평가 (32K 클러스터 사용, 128K 버퍼사용)

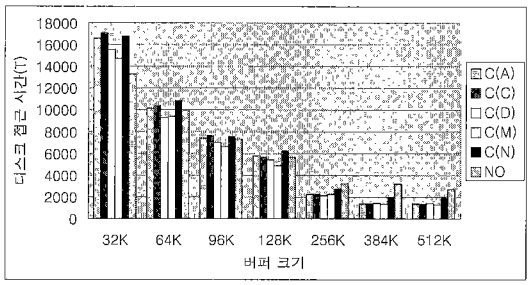
표 8 질의 영역의 특성에 따른 디스크 접근 횟수 증감 비율 (32K 클러스터 사용, C(M)를 기준, 128K버퍼사용)

	질의 영역 크기	C(A)	C(C)	C(D)	C(N)
데이터 밀집 지역	0.10%	27.5%	25.0%	9.6%	14.3%
	0.25%	18.8%	24.7%	14.9%	26.3%
	0.50%	20.5%	18.3%	13.4%	25.1%
	1.00%	16.4%	24.9%	18.0%	34.8%
	4.00%	1.5%	14.1%	-6.7%	25.8%
	9.00%	-2.6%	9.6%	-4.0%	23.7%
데이터 희박 지역	0.10%	-7.7%	-16.7%	-16.7%	26.3%
	0.25%	-6.9%	8.8%	6.1%	27.9%
	0.50%	-6.5%	-8.2%	-4.8%	5.7%
	1.00%	7.4%	21.4%	12.0%	29.6%
	4.00%	5.9%	-4.1%	-14.4%	30.7%
	9.00%	1.1%	4.2%	-6.9%	41.6%

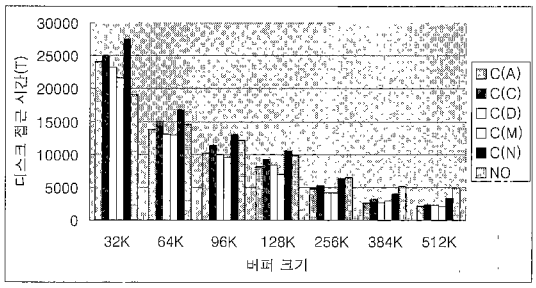
표 8은 C(M)을 사용한 경우의 디스크 접근 횟수를 기준으로 다른 알고리즘을 사용한 경우에 디스크 접근 횟수의 증감 비율을 질의 영역의 특성과 크기에 따라서 보여준다. 실험에서 4가지 경우의 질의 영역의 크기에서 가장 좋은 성능을 보인 C(M)를 기준으로 실험 결과를 정리하였다. 데이타가 밀집한 지역의 질의 집합에 대해서는 C(M)가 대부분의 경우에 있어 좋은 성능을 나타내고 있다. 특히 N-순서화 기법을 이용한 C(N)에 비해서는 질의 영역의 크기에 따라14.3%~34.8%의 성능이 개선되었다. 그러나, 데이타가 희박한 지역에서는 C(M)가 C(D)보다 더 성능이 나쁘게 나타나고 있으며, C(A)와 C(C)와는 비슷한 성능을 나타내고 있다.

지금까지의 실험에서는 128K 크기의 버퍼를 사용하였을 때, 전체적으로 C(M)의 성능이 가장 우수함을 보이고 있다. 그런데, 디스크 접근 시간은 메모리 버퍼의 크기에 따라 달라질 수 있으므로, 버퍼 크기에 따른 성능을 비교해 볼 필요가 있다. 그림 18은 버퍼 크기에 따라 각각의 셀 클러스터링 알고리즘을 적용한 경우와 셀 클러스터링 알고리즘을 적용시키지 않은 경우에 대해서 디스크 접근 시간을 측정한 결과를 보여준다. 클러스터의 크기는 32K를 사용하였다.

이 실험을 통해서 다음의 사실들을 알 수 있다. 첫째,



(a) 질의 영역 크기가 0.50%인 경우



(b) 질의 영역 크기가 1.00%인 경우

그림 18 버퍼 크기에 따른 디스크 접근 시간 비교 (32K 클러스터 사용)

이 논문에서 제안하는 클러스터 생성 방법을 이용한 클러스터링 알고리즘들은 버퍼 크기가 클러스터 크기의 두 배인 64K 일 때에도 셀 클러스터링으로 인해 디스크 접근 시간이 줄어듦을 알 수 있다. 이것은 클러스터가 지역성이 높은 페이지들로 구성되었음을 의미한다. 둘째, 버퍼 크기와 질의 영역의 크기에 따라 차이는 있지만 전체적으로 C(M)와 C(D)의 성능이 우수함을 알 수 있다. 셋째 버퍼 크기가 커짐에 따라 이 논문에서 제안한 공간 지역성을 고려한 알고리즘인 C(A), C(D), C(M)간의 성능 차이는 2.0% 이내로 줄어들지만, C(C)와 C(N)는 그림 19와 같이 많은 차이를 보이고 있다.

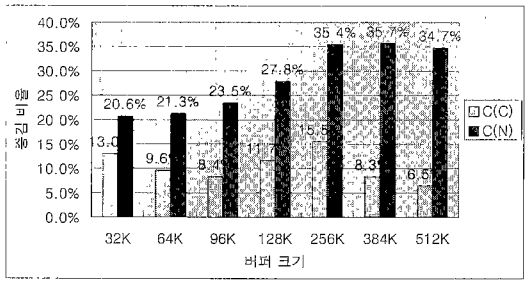


그림 19 버퍼 크기에 따른 디스크 접근 시간의 증감 비율(32K 클러스터 사용, C(M)을 기준, 전체 질의 영역)

셀 클러스터링에 대한 마지막 실험은 클러스터의 크기에 따른 성능 평가이다. 이전의 실험에서 메모리 버퍼의 크기가 클러스터의 크기에 비해 크지 않은 경우에는 클러스터링으로 인해 디스크 접근 비용이 줄어들지 않음을 알 수 있었다. 따라서 클러스터의 크기와 버퍼 크기에 따라 디스크 접근 시간을 비교해 보아야 한다. 그림 20은 C(M)알고리즘에 대해서 버퍼 크기와 클러스터 크기에 따른 디스크 접근 시간을 보여주고 있다.

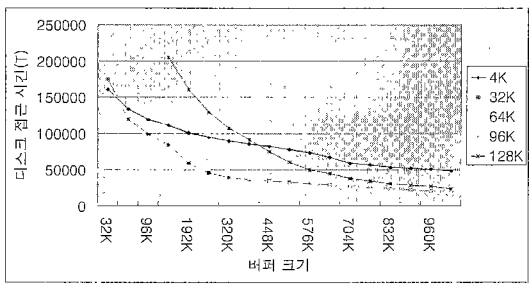


그림 20 클러스터 크기와 버퍼 크기에 따른 디스크 접근 시간 (C(M) 사용, 전체 질의 집합)

실험 결과 버퍼 크기가 32K인 경우에는 클러스터 크기가 4K(즉, 클러스터링을 하지 않은 경우)일 때, 버퍼 크기가 4K~384K인 경우에는 클러스터 크기가 32K일 때, 448K~1024K인 경우에는 클러스터 크기가 64K일 때가 가장 디스크 접근 시간이 적게 나타난다. 이 실험을 통해서 버퍼 크기가 커짐에 따라 클러스터의 크기에 따른 성능의 차이가 계속 줄어 들고 있다는 사실을 알 수 있다. 즉, 버퍼가 충분히 크다면 클러스터 크기는 전체 성능에 큰 영향을 주지 않는다.

7. 결론 및 향후 연구과제

이 논문은 정규분할 공간 색인 구조에서 색인을 이용한 공간 객체 저장 구조 모델을 제시하였으며, 공간 질의 처리 시 디스크 접근 비용을 줄이기 위한 여러 가지 클러스터링 알고리즘을 제안하였다. 그리고 성능 평가 실험을 통해 제안한 알고리즘이 디스크 접근 비용을 줄일 수 있음을 입증하였다.

정규분할 공간 색인 구조에서 발생하는 중복 참조 객체에 대한 저장 위치를 결정하는 객체 클러스터링 알고리즘으로는 무게 중심, 최단거리, 무게 중심간 거리 등을 이용한 알고리즘들을 제안하고 실험을 통해 성능 비교를 하였다. 객체 클러스터링 알고리즘에 대한 실험을 통해 다음과 같은 결론을 얻을 수 있다.

- 최단거리를 이용한 객체 클러스터링 알고리즘과 무게 중심간 거리를 이용한 객체 클러스터링 알고리즘이 대체로 좋은 성능을 나타냈다.
- 메모리 버퍼의 크기가 커지거나 데이타 분포가 희박한 지역에 대한 질의 처리에서는 알고리즘간에 성능 차이가 크지 않았다.
- 동일한 객체 클러스터링 알고리즘을 사용하더라도 공간 객체 저장 구조 모델에 따라서는 성능의 차이가 크게 나타났다. 즉, 확장된 페이지 기반 저장 구조가 페이지 기반 저장 구조에 비해 128K 크기의 버퍼를 사용한 경우 최고 49.2%의 성능이 개선되었다.

논리적으로 서로 인접한 셀들을 클러스터 단위로 묶어서 디스크 상에 물리적으로 인접하게 저장하기 위해서 인접 셀 쌍간의 셀 클러스터링 결합력을 이용한 클러스터 생성 방법을 제시하였다. 그리고 셀 클러스터링 결합력을 계산하는 방식으로 셀 인력, 셀의 무게 중심간 거리, 중복 참조도, 포함 공간 객체의 개수를 이용한 알고리즘을 제안하였다. 셀 클러스터링 알고리즘에 대한 실험을 통해서 다음과 같은 결론을 얻을 수 있다.

- 객체 클러스터링 알고리즘은 셀 클러스터링 알고리즘의 성능에 크게 영향을 주지는 않는다.
- N-순서화 기법을 이용하여 셀 클러스터링을 한 경우가 셀 클러스터링을 안 한 경우보다 35.6%의 성능이 개선되었다. 따라서 셀 클러스터링을 통해서 디스크 접근 시간이 줄어듦이 증명되었다. (32K 클러스터와 512K 버퍼를 사용)
- 이 논문에서 제안한 C(M)을 사용해서 셀 클러스터링을 한 경우가 N-순서화 기법을 이용한 경우보다 34.7%의 성능이 개선되었다. 따라서 공간 지역성을 이용한 방법이 N-순서화 기법을 이용한 방법보다 우수함이 증명되었다. (32K 클러스터와 512K 버퍼를 사용)
- 질의 영역에 따른 실험 결과 전체적으로는 C(M)이 가장 좋은 성능을 나타내었으나, 질의 영역의 크기가 전체 지도 크기의 4.00%이상인 경우와 데이타 희박 지역인 경우에는 C(D)의 성능이 좋은 것으로 나타났다. (32K 클러스터와 128K 버퍼를 사용)
- 버퍼 크기가 커짐에 따라서 이 논문에서 제안한 공간 지역성을 이용한 알고리즘간의 성능 차이는 2% 이내로 줄어들지만 C(C)와 C(N)은 여전히 성능차이가 크게 나타난다.
- 버퍼 크기가 클러스터 크기에 비해 충분히 크다면 클러스터 크기는 전체 성능에 많은 영향을 주지 않는다.

전체적으로 말하면, 정규 분할 공간 색인을 이용하여 공간 데이타를 저장할 경우에는 클러스터 기반 저장 구조에서 중복 참조도를 이용하여 셀 클러스터링을 하는 것이 가장 좋은 성능을 나타낸다. 셀 클러스터링은 하지 않고 중복 참조 객체의 저장 위치만을 결정할 경우에는 최단 거리를 이용하여 객체 클러스터링을 하는 것이 좋다. 이 경우에는 페이지 기반 저장 구조 모델보다는 지역 클러스터 기반 저장 구조를 이용하는 것이 더 효과적이다.

향후 연구는 제안한 여러 가지 알고리즘을 결합하여 보다 나은 성능의 클러스터링 알고리즘을 개발하는 것이다. 이 논문에서 제안한 각각의 알고리즘은 공간 관련성을 계산하기 위해서 하나의 요소(factor)만을 고려하였다. 따라서 각 알고리즘에서 고려한 요소들을 결합한 새로운 알고리즘의 설계가 가능하다. 이러한 알고리즘들에 대해서는 다양한 성능 평가 실험을 통한 성능 검증이 필요하다.

그리고 정규 분할 공간 색인을 가지는 공간 데이타베이스에서 여러 개의 디스크를 사용해서 공간 데이타를 저장하는 경우에 디스크 접근 비용을 최소화하기 위한

연구가 필요하다. 여러 개의 디스크를 사용할 경우에 디스크 접근 비용을 줄이기 위한 전통적인 방법으로는 디-클러스터링이 있다. 디-클러스터링을 하는 방법으로써 하나의 디스크에서 클러스터링을 위해 사용된 알고리즘을 적용시킬 수 있다.

참 고 문 헌

[1] Gisbert Droege, Hans-Jorg Scheck, Query-adaptive data space partitioning using variable-size storage clusters, Advances in Spatial Databases, pp. 337-356, Springer-Verlag, 1993.

[2] Thomas Brinkhoff, Hans-Peter Kriegel, The Impact of Global Clustering on Spatial Database Systems, Proc. VLDB, pp. 168-179, 1994.

[3] 김주형, 김진덕, 홍봉희, 김장수, 확장된 Quad-tree을 이용한 클러스터링에 관한 연구, 한국 정보 과학회 97 봄 학술 발표 논문집 vol. 23, No.1, pp. 35-38, 1997.

[4] Elmasri Navathe, Fundamentals of Database Systems, The Benjamin / Cummings Publishing Company, Inc.

[5] H. V. Jagadish, Linear Clustering of Objects with Multiple Attributes, Proc. ACM SIGMOD, Vol.19, No.4, 1990

[6] K. J. Li, R. Laurini, The Spatial Locality and a Spatial Indexing Method by dynamic clustering in Hypermap system, Advances in Spatial Database, pp. 207-224, Springer-Verlag, 1991

[7] 유진영, 김진덕, 홍봉희, 김장수, 정규분할 공간 색인을 위한 클러스터링 알고리즘, 한국 정보 과학회 98 봄 학술발표논문집 vol25, No.1, pp. 50-52, 1998.

[8] Won Kim, Modern Database Systems, ACM Press, 1995

[9] Brinkhoff T., Horn H., Kriegel H.-p., Schneider R., A Storage and Access Architecture for Efficient Query Processing in Spatial Databases, Proc. 3rd Int. Symp. On Large Spatial Database, Singapore, pp. 357-376, 1993.

[10] Nievergelt J., Hinterberger H., Sevcik K.C., The Grid File: An Adaptable, Symmetric Multikey File Structure, ACM Trans. On Database Systems, Vol. 9, No. 1, pp. 38-71, 1984

[11] Michael J. Folk, Bill Zoellick, Greg Riccardi, File Structures An Object-Oriented Approach with C++, Addison Wesley, pp. 49, 1998.

[12] A Guttman, R-trees: a dynamic index structure for spatial searching, Proc. ACM SIGMOD, pp47-57, June 1984

[13] T. Sellis, N. Roussopoulos, and C. Faloutsos, The r+ tree: a dynamic index for multi dimensional objects, In Proc. 13th International Conference on VLDB, pp507-518, England, September 1987

[14] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, The r*-tree: an efficient and robust access method for points and rectangles, ACM SIGMOD, pp322-331, May 1990

[15] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1989

조 대 수

1995년 2월 부산대학교 컴퓨터공학과 공학사. 1997년 2월 부산대학교 컴퓨터공학과 공학석사. 1997년 3월 ~ 현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 공간데이타베이스, 공간데이타 클러스터링, 인터넷 GIS, GIS 표준화

유 진 영

1997년 2월 경성대학교 컴퓨터공학과 졸업(공학사). 1999년 2월 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2000년 ~ 현재 삼성 SDS(주) 컨설팅 사업부 IT 컨설팅팀 대리. 관심분야는 객체지향 데이타베이스, 지리정보 시스템, 전사적 자원관리 시스템(ERP) 등.

홍 봉 희

1982년 서울대학교 전자계산기공학과 졸업(공학사). 1984년 서울대학교 대학원 전자계산기공학과 졸업(공학석사). 1988년 서울대학교 대학원 전자계산기공학과 졸업(공학박사). 현재 부산대학교 공과대학 컴퓨터공학과 정교수. 관심분야는 병렬공간 DB, 분산공간 DB, 개방형 GIS