Journal of the Korea Academia-Industrial cooperation Society
Vol. 12, No. 5 pp. 2383-2389, 2011

DOI: 10.5762/KAIS.2011.12.5.2383

공간 연속질의 처리에서 영역 기반의 저장 구조를 이용한 효율적인 디스크 접근 방법

정원일[1*]
[1]호서대학교 정보보호학과

Efficient Disk Access Method Using Region Storage Structure in Spatial Continuous Query Processing

Weonil Chung[1*]
[1]**Dept. of Information Security Engineering, Hoseo University**

요 약 유비쿼터스 응용은 실시간으로 입력되는 데이터 스트림과 저장된 공간 데이터를 동시에 처리하는 이중적인 공간 연속 질의 처리 기술이 요구된다. 이러한 공간 연속 질의 처리에서는 대용량 공간 데이터에 대한 디스크 접근 비용을 최소화가 요구되나 기존 공간색인 기법은 논리적 인접성을 공간 데이터의 물리적인 인접성을 보장할 수 없으므로 공간 데이터 탐색에 있어 비용이 증가한다. 또한 데이터 인접성 보장을 위한 공간 순서화 기법의 경우에도 빈번하게 접근되는 질의 공간 영역에 대한 클러스터링을 고려하지 않고 있다. 본 논문에서는 이중적인 공간 연속질의 처리에서 공간 데이터의 효율적인 접근을 위한 영역 기반 저장 구조를 제안한다. 제안 기법에서는 영역을 기반으로 데이터를 인접하게 저장하고 사용자 질의를 영역 기반으로 그룹 처리함으로써 질의 처리 비용을 감소시킬 수 있다.

Abstract Ubiquitous applications require hybrid continuous query processing which processes both on-line data stream and spatial data in the disk. In the hybrid continuous spatial query processing, disk access costs for the high-volume spatial data should be minimized. However, previous indexing methods cannot reduce the disk seek time, because it is difficult that the data are stored in contiguity with others. Also, existing methods for the space-filling curve considering data cluster have the problem which does not cluster available data for queries. Therefore, we propose the region storage structure for efficient data access in hybrid continues spatial query processing. This paper shows that there is an obvious improvement of query processing costs through the contiguous data storing method and the group processing for user queries based on the region storage structure.

Key Words : Spatial Continuous Query, Spatial Data Stream, Data Clustering

1. 서론

유비쿼터스 환경을 기반으로 도시정보시스템, 교통정보시스템, 재난재해정보시스템 등의 다양한 응용들이 활성화되고 있다[1]. 이러한 유비쿼터스 응용들은 스마트폰, 텔레메틱스 단말 등으로부터 생성되는 위치 및 시간 등의 정보를 포함하는 실시간 데이터 스트림과 정적으로 사전 구축된 공간데이터를 동시에 처리하는 이중적인 공간 연속 질의 처리 기술이 요구된다. 공간 연속 질의에서 처리되는 공간 데이터는 변화가 빈번하지 않지만 대용량 데이터라는 특성으로 인해 모든 공간 데이터를 메모리에 상주시킬 수 없으므로 질의 처리시 많은 디스크 접근 비용이 요구된다. 이러한 공간데이터의 효율적인 처리를 위해 R-tree, R*-Tree, R+-tree, Quad tree등의 공간 색인 기법이나 Z-order, Hilbert-curve 등의 공간 순서화 방법이 연구되었다[2,3,4,5,10]. 기존 공간 색인 기법은 디스크에 저장된 공간데이터에 대한 위치 정보를 색인으로 구성하여 디스크 입출력 비용을 줄이는 기법이다. 그러나 공간

"이 논문은 2010년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임"(2010-0073)
*교신저자 : 정원일(wnchung@hoseo.edu)
접수일 11년 03월 17일 수정일 (1차 11년 04월 15일, 2차 11년 04월 27일) 게재확정일 11년 05월 12일

데이터는 삽입되는 순서에 따라 디스크에 저장되기 때문에 논리적으로 인접한 공간 데이터의 물리적 인접성을 보장할 수 없으므로 질의 처리를 위한 데이터 접근시 탐색 시간 증대가 필연적이다. 공간 순서화 방법은 논리적으로 인접한 공간 데이터를 디스크에서 물리적으로 인접한 곳에 저장하는 방법으로, 일차원 데이터를 키의 순서대로 디스크에 저장하는 경우에는 효과적인 클러스터링 방법일 수 있으나, 공간 데이터와 같은 다차원 데이터를 특성을 반영하기가 용이하지 않을 뿐 아니라 질의에서 접근이 빈번한 공간 영역에 대한 클러스터링이 이루어지지 않고, 고정된 순서에 의해 클러스터링되는 문제점을 갖는다.

공간 연속 질의 처리를 위한 다른 연구로 샘플링을 통한 데이터 축소기법이 있다[6,7]. 데이터 축소 기법은 디스크에 저장된 공간 데이터를 레벨에 따라 서로 다른 비율로 축소 저장하고, 시스템 처리 능력에 따라 레벨을 선택하여 질의를 처리한다. 이 기법은 질의 처리에 이용되는 데이터를 감소시켜 디스크 입출력 비용과 탐색 시간을 줄인다. 그러나 데이터 축소 기법은 빠른 디스크 입출력을 위한 데이터 축소 과정에서 데이터의 정확성을 보장하지 못하는 단점을 갖는다.

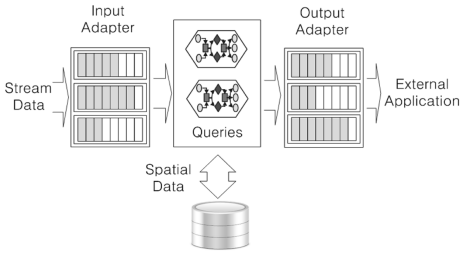
본 논문에서는 이러한 공간 연속질의 처리 환경에서 색인 구축 과정에서 데이터의 인접성을 보장하고 인접 공간 영역을 접근하는 질의들에 대한 클러스터링을 제공하는 영역 기반의 저장 구조를 이용하여 효과적인 디스크 접근 방법을 제안한다. 제안 기법은 정적 질의 특성을 고려한 클러스터링 기법을 이용하여 접근 빈도가 높은 질의 영역에 대한 신속한 디스크 접근을 제공한다. 또한, 디스크 캐싱을 이용한 파일 처리를 통해 캐시 데이터를 질의 중첩에 따라 인접성을 제공하여 질의 처리 비용을 최소화하며, 서로 다른 영역에 대한 겹치는 공간 데이터의 중첩 처리를 위해 무게중심을 이용하여 최적의 영역에서 유지할 수 있는 방법을 제공한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로 공간 연속질의 처리, 공간데이터 색인, 공간 클러스터링, 그리고 데이터 축소 기법에 대해 소개한다. 3장에서는 영역 기반의 효율적인 공간 데이터 접근을 위한 저장 구조에 대해 기술한다. 4장에서는 제안 기법에 대한 성능 평가를 수행하고, 마지막으로 5장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련연구

2.1 공간 연속 질의 처리

공간 연속 질의는 입력 데이터 스트림과 디스크에 저장된 공간 데이터를 연계하여 처리함으로써 다양한 응용서비스에 이용된다. 그림 1은 데이터 스트림과 공간 데이터를 융합 처리하는 공간 연속 질의 처리를 보여준다. Input Adapter는 입력되는 스트림 데이터를 관리하며, 입력되는 스트림 데이터는 디스크에 저장된 공간데이터와 결합되어 질의 처리에 이용된다. 질의 처리가 끝난 데이터는 Output Adapter를 통하여 응용서비스에 전달된다. 이때 디스크에 저장된 공간데이터는 효율적인 공간 질의 처리를 위해 신속한 데이터 접근이 요구된다.



[그림 1] 공간 연속 질의 처리

2.2 공간데이터 색인 기법

디스크에 저장된 공간 데이터의 처리를 위한 공간 색인 기법인 R트리는 검색 수행시 공간영역의 겹침이 많을수록 접근하는 노드의 수가 증가하여 검색시간이 늘어난다[2]. 이러한 공간영역의 겹침을 줄이기 위한 방법으로 확장된 R*트리, R+트리 등이 연구되었다[3,4].

기존의 공간데이터 색인 기법의 문제점은 데이터를 저장할 때 논리적인 데이터의 인접성을 물리적으로 인접시키지 못한다는 점이다. 물리적 인접성이 유지된 데이터는 함께 참조될 가능성이 높은 데이터를 디스크상에 인접하게 저장하여 질의 처리시 디스크 탐색 시간 및 로딩 시간을 줄일 수 있으며, 이러한 물리적 인접성 보장을 위한 데이터 저장구조가 요구된다.

2.3 공간 클러스터링 순서화 기법

공간 클러스터링이란 영역 질의를 보다 효율적으로 처리하기 위해 공간상에서 인접한 데이터를 디스크에서 물리적으로 인접한 곳에 저장하는 방법이다[8,9]. 이러한 공간 클러스터링에서 공간데이터와 같은 다차원 데이터는 일차원에 가까운 특성을 가지는 디스크에 순서대로 저장하기가 용이하지 않는 단점이 있다. 이런 문제를 해결하기 위한 기존 기법으로 Z-order(N-order), Hilbert-curve 등의 공간 순서화 방법이 연구되었다[4,5].

Z-order는 클러스터링을 수행할 때 해당 영역에 대해

문자 'Z'를 재귀적으로 반복하여 2차원 공간의 모든 영역을 지나도록 한다. Z-order 곡선이 지나는 영역을 순차적으로 정렬하는 Z-order 기법을 사용하여 셀들을 1차원으로 정렬한 것으로, Z-order 순서에 의해 디스크에 저장되면 디스크 탐색시간을 줄일 수 있다[5].

그러나 Z-order에서는 고정된 순서에 의해 클러스터링을 수행함으로써 접근 빈도가 높은 영역을 대상으로 하는 유사 질의들에 대한 처리를 고려하지 못하는 문제가 있다.

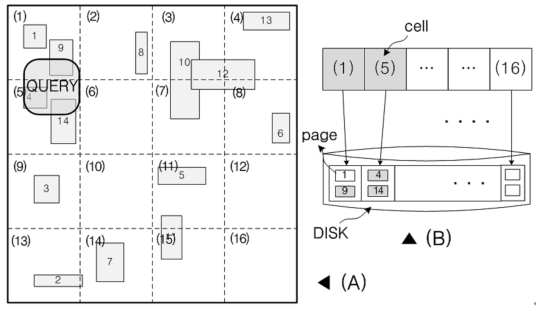
2.4 데이터 축소 기법

데이터 축소 기법은 공간 연속 질의 처리에서 방대한 양의 데이터를 효율적으로 처리하기 위한 기법으로, 질의 처리에 이용되는 데이터를 정해진 레벨에 따라 축소하여 샘플링이나 집계 연산 처리에서 효과적인 방법으로 이용된다. 이 기법은 데이터 크기를 줄임으로써 질의 처리시 디스크 입출력 비용을 줄일 수 있으나, 데이터 축소 과정을 거친 데이터로 인해 질의 처리 결과의 정확성은 떨어지게 될 뿐 아니라 공간 데이터 특성이 반영되지 않아 공간 연속 질의 처리에 적용하기 어렵다.

3. 영역 기반의 저장 구조

공간 데이터 처리를 위한 디스크 접근 비용을 줄이기 위한 영역 기반의 저장구조는 그림 2-(A)와 같은 영역을 이용하며, 그림 2에서는 4X4 영역을 이용한다. 영역으로 분할된 각각의 공간이 셀이 되고, 각각의 셀은 고정된 새로운 영역의 MBR(Miminum Boundary Rectangle)을 가지며, 셀의 MBR 영역 범위 안에 속하는 공간데이터를 관리하게 된다. 또한 하나의 셀은 하나의 페이지 공간을 할당 받아 공간데이터를 저장한다.

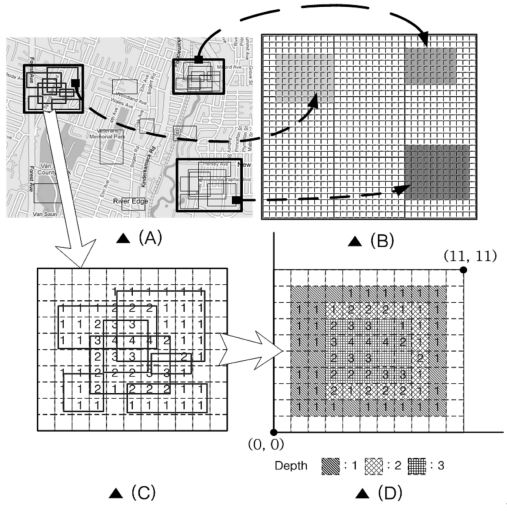
제안한 저장구조에서 위와 같은 질의가 발생한 경우 질의처리 과정은 다음과 같다. 첫 번째, 질의의 MBR 조건에 해당하는 셀을 검색한다. 다음으로, 검색된 셀에 해당하는 공간데이터를 검색한다. 질의 처리결과 그림 2-(B)에서와 같이 (1), (5) 두 개의 셀이 검색되며, 해당 질의는 두 개의 페이지 검색만으로 해당 질의를 처리할 수 있다. 또한 그림 2처럼 (1), (5)번 셀이 클러스터링되어 인접하게 저장되는 경우 기존 R-tree 기법에 비하여 데이터의 고속로딩이 가능하다. 이처럼 질의처리는 공간데이터의 클러스터링 방법과 저장 방법에 따라 데이터 탐색시간 및 데이터 로딩 시간에 차이를 가진다.



[그림 2] 영역 기반의 저장구조

3.1 질의 특성을 반영한 클러스터링 기법

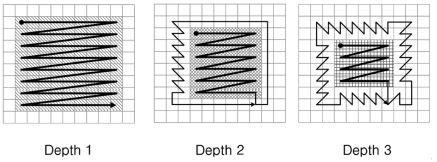
공간영역은 질의 형태에 따라 공통된 영역에 발생되거나, 비슷한 지역의 데이터를 요구하는 경향이 있다. 특히 정적 질의에서 공통적으로 이용되거나 자주 이용되는 영역의 공간데이터는 디스크 상에 인접하게 저장하여 질의 처리시 데이터의 빠른 검색과 고속로딩을 가능하게 한다. 본 절에서는 정적 질의의 그룹화를 통해 질의에 자주 이용되는 공간데이터를 디스크상에 인접하게 저장하는 클러스터링 기법을 제안한다.



[그림 3] 질의 특성이 반영된 클러스터링

그림 3은 정적 질의의 특성을 이용한 클러스터링 방법으로 질의에 의해 자주 이용되는 영역을 그룹화하여 해당 영역에 반영한다. 이때 Depth는 질의 중첩도와 중요도를 나타내며, 모든 질의는 중첩도에 따라 Depth를 정의하였다. 그림 3의 (A)는 실제 지도상에서 질의가 그룹화되는 과정을 보여주며, (B)는 그룹화된 질의를 해당 영역에 반영한 것이다. (C)는 질의 중첩도에 따른 그룹화 과정을 보여주며, 질의의 중첩도에 따라 서로 다른 Depth를 가진

다. 질의 그룹화 결과 (D)와 같이 표현될 수 있다. Depth에 따라 구분된 영역은 사용자가 정한 Depth에 따라 서로 다른 공간 순서화 과정을 거치게 된다. 그림 3-(D)에서 전체 영역을 (0, 0) ~ (11, 11)로 가정하였을 때 Depth에 따른 영역의 크기는 Depth 1: (1, 1) ~ (10, 10), Depth 2: (3, 2) ~ (9, 9), Depth 3: (3, 3) ~ (8, 8)이다.

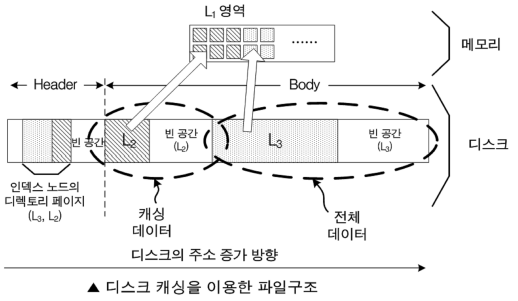


[그림 4] Depth에 따른 공간 순서화 과정

그림 4는 사용자가 정의한 Depth 크기에 따른 공간 순서화 과정이다. 공간 순서화 방법은 기존 공간 채우기 기법을 이용하여 정렬하며, Depth 크기에 따라 서로 다른 형태를 갖게 된다. 공간 순서화 과정은 Depth에 따라 서로 다른 형태를 갖는다. Depth가 지정되지 않은 경우는 기존 순서화 방법과 동일하다. 하지만 Depth가 지정된 경우는 질의 중첩도에 따라 순서화 과정이 변하게 된다. Depth가 1인 경우는 입력된 등록된 질의영역에 대하여 우선적으로 정렬한 후 나머지 영역을 정렬한다. Depth가 2 또는 그 이상인 경우는 Depth 1인 경우와 마찬가지로 해당 Depth 영역을 우선적으로 정렬한 후 나머지 영역을 정렬한다. 이렇게 정렬된 공간데이터는 질의 처리에 자주 이용되는 데이터는 디스크상 앞쪽에 위치되어 빠른 데이터 호출이 가능하다. 또한 질의 처리에 함께 이용되는 데이터가 인접하게 저장되어 데이터 검색시간 또한 감소하게 된다.

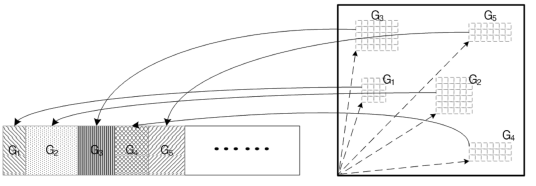
3.2 디스크 캐싱을 이용한 파일구조

디스크에 저장된 데이터의 고속 로딩을 위해 디스크 캐싱을 이용한 파일 구조는 아래 그림 5와 같다.



[그림 5] 디스크 캐싱을 이용한 파일 구조

제안 구조는 페이지의 저장 위치 정보를 포함하는 Header부와 실제 공간데이터가 저장되는 Body부로 구성된다. Header는 Body에 저장된 공간데이터를 관리하기 위한 디렉토리 정보를 저장한다. Header의 디렉토리는 성능 최적화를 위해 역 정렬되어 저장되는데, 이는 자주 이용되는 캐싱된 데이터와 디렉토리 정보를 인접하게 위치하도록 저장하여 데이터 검색시간을 줄일 수 있다. Body는 클러스터링에 의해 캐싱된 축소 데이터(L2)와 축소 데이터(L3)의 저장이 이루어지며, L1은 Body의 성능 최적화를 위해 메모리에 유지하여 데이터 입출력에 사용한다. 이때 할당되는 L2 와 L3에 저장된 데이터의 크기는 클러스터링 형태에 따라 변하게 되므로 모든 데이터를 저장할 수 있는 충분한 공간을 고정으로 할당하여 사용한다.



[그림 6] 클러스터링된 데이터 저장과정

그림 5에서의 캐싱데이터(L2)는 그림 6-(A)와 같이 표현될 수 있다. 그림 6-(A)의 G1 – G5는 클러스터링에 의해 생성된 축소 데이터가 저장된 모습을 나타낸다. 캐싱영역의 데이터 저장순서는 해당 영역의 시작점과 클러스터링 영역의 거리를 기준으로 그림 6의 (A-B)를 통해 확인할 수 있다. 캐싱영역은 빈 공간을 유지하여 추가적인 데이터 삽입에 이용된다. 제안 캐싱기법의 디스크 공간은 연속 공간 할당기법을 이용한다.

3.3 중첩데이터 저장을 위한 무게중심 참조방법

무게중심 참조방법에서는 중첩 데이터에 대한 디스크 접근 비용을 최소화하기 위해 중첩 데이터 저장을 위한 최적 페이지를 선정한다. 공간 데이터가 다각형일 경우 무게중심을 계산하기 위해 다각형을 삼각형으로 나누고 각 삼각형의 무게중심의 평균 위치를 구한다. 이때 각각의 삼각형은 서로 다른 크기를 가질 수 있으므로 무게중심의 평균 위치를 구할 때는 넓이를 가중치로 하는 평균을 이용한다.

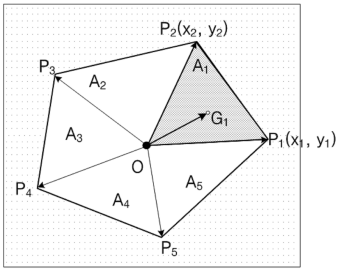
[수식 1] $A_i = \frac{x_i y_{i+1} - x_{i+1} y_i}{2}$

[수식 2] $G_i = (\frac{x_i + x_{i+1}}{3}, \frac{y_i + y_{i+1}}{3})$

[수식 3]

$$G = \frac{\Sigma A_i G_i}{A} = (\Sigma \frac{(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)}{6A}, \Sigma \frac{(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)}{6A})$$

그림 7의 오각형을 A1~A5의 삼각형으로 분할한 경우 각 삼각형 영역의 넓이 Ai([수식 1])와 무게중심 Gi([수식 2])를 계산하고, [수식 3]을 적용하여 전체의 무게 중심을 산출한다.



[그림 7] 다각형의 무게중심

그림 6의 (A)에서와 같이 공간데이터 '3'이 중첩되어 위치하는 경우 무게중심 계산법에 의해 계산하면, 그림 6의 (B)와 같이 저장된다. 이와 같은 결과는 많은 영역을 포함하는 셀에 공간데이터가 저장되다. 이는 질의처리에 이용될 확률이 높은 셀에 할당되며, 질의 처리시 발생하는 디스크 입출력 비용을 줄일 수 있다.

4. 성능 평가

4.1 실험환경

실험에 사용된 시스템 환경은 CPU Pentium 3.0GHz, 메모리 2GB, 운영체제 Window XP SP3를 기반으로 MS Visual Studio 2005로 제작 되었으며, 데이터 생성기를 이용하여 21,153 개의 공간데이터를 랜덤으로 생성했다. 또한 공간데이터 저장 구조는 8KB(8192 Byte) 크기의 페이지 기반 저장 구조를 모델로 사용하여 공간데이터를 저장했다. 제안된 영역 기반의 저장 구조는 50X50 개의 영역으로 구성했다.

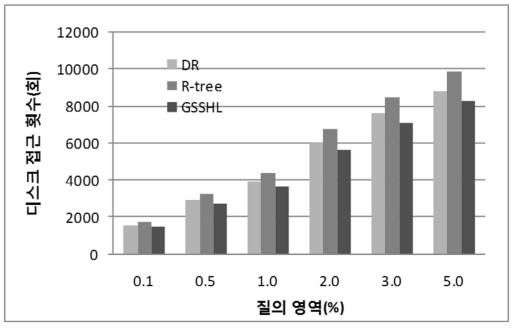
4.2 성능 분석

성능 분석은 기존기법과 제안기법의 비교 분석을 위해 디스크 입출력 횟수, 탐색 시간, 정확도를 비교 평가한다. 평가에서는 범위 질의를 이용하여 평가하였다. 만약, 생성기를 이용하여 생성된 공간객체를 Gi, 질의 영역 Q라고 할 때 Intersects(Gi, Q) 연산이 수행됨을 알 수 있다[11]. 질의 영역은 전체 영역의 0.1% - 5.0%를 기준으로 평가하였으며, 질의 처리에 따라 이용되는 데이터 개수는 밀집 지역과 비 밀집 지역에 따라 질의 영역을 바꾸어 가며 25회씩 수행한 평균값을 기준으로 했다. 표 1은 전체 영역에 따른 질의 영역의 비율, 질의 영역에 따른 평균 데이터 개수, 전체 데이터 개수를 기준으로 질의 처리에 이용되는 데이터의 비율을 나타낸다.

[표 1] 질의 영역에 따른 데이터 개수 및 비율

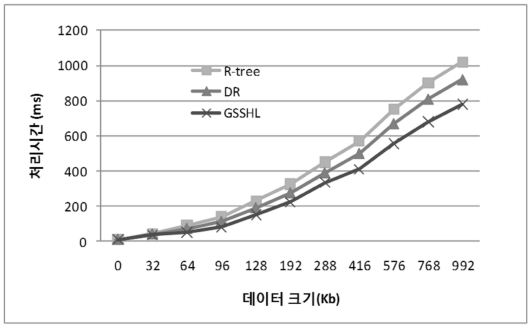
질의 영역	평균 데이터 개수	데이터 비율
0.1%	440.8	2.2%
0.5%	932.3	4.6%
1.0%	1279.5	6.3%
2.0%	1753.2	8.7%
3.0%	1923.2	9.6%
5.0%	2035.7	10.1%

첫 번째 성능평가로 질의 영역에 따른 디스크 접근 횟수를 평가했다. 그림 8은 제안기법의 성능평가 결과이다. 비교 대상으로 제안기법(GSSHL: The Grid Storage Structure for High-performance Loading), R-tree, 데이터 축소기법(DR: Data Reduction)을 이용했다. 성능평가 결과 GSSHL, DR, R-tree 순으로 디스크 접근 횟수가 적은 것으로 나타났다. DR기법이 R-tree에 비해 빠른 성능평가 결과를 보인 이유는 데이터를 축소 저장하여 전체 디스크 탐색 시간을 줄일 수 있기 때문이다. 그러나 DR 기법은 정확성 보장이 어려우며 공간 연속 질의처리에서 이용되기 어렵다. 성능평가 결과 질의 특성을 반영한 클러스터링 기법(GSSHL)의 빠른 질의처리가 가능함을 확인할 수 있다.



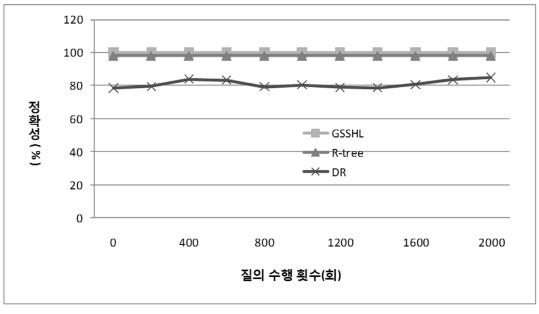
[그림 8] 질의영역에 따른 디스크 접근 횟수

또 다른 성능평가로 제안기법과 기존기법의 질의 처리 시간을 비교했다. 이때 DR 기법은 25%의 비율로 데이터를 축소하여 평가하였으며, 결과는 그림 9를 통해 확인할 수 있다. 그림 9는 데이터 크기에 따른 질의 처리시간의 결과를 보여주며, 평가 결과 크기가 증가함에 따라 GSSHL, DR, R-tree 순으로 빠른 처리 결과를 보였다.



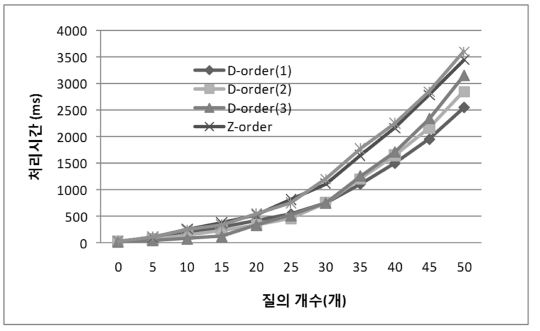
[그림 9] 데이터 크기에 따른 질의처리시간

그림 8-9의 성능분석 결과 DR 기법이 R-tree에 비하여 빠른 질의처리 시간을 보여주는데, 이는 원본 데이터가 감소되어 전체적인 질의 처리 시간이 감소하였음을 나타낸다. 그러나 그림 10을 통해 정확성을 확인한 결과 DR 기법은 정확성에 문제가 나타나고 있음을 확인할 수 있다. 데이터 정확성 평가는 질의 수행 횟수에 따라 정확성을 평가했다. DR 기법의 데이터 축소 비율은 25%로 고정하여 평가했다. 그림 10은 질의 수행 횟수에 따른 정확성 평가 결과를 보여준다. 성능분석결과 제안기법과 R-tree는 100% 정확한 처리결과를 보인 반면 DR 기법은 15%~20%의 데이터 손실이 발생하였음을 알 수 있다.



[그림 10] 질의 수행횟수에 따른 정확도

그림 11은 본 논문에서 제안한 Depth에 따른 공간 순서화 기법(D-order)과 기존 공간 순서화 기법(Z-order, N-order)의 질의 처리시간을 비교하였다. D-order 기법은 Depth의 레벨(1~3)에 따라 3가지를 함께 평가 하였다. 본 성능평가는 정적 질의에 대한 성능평가로 미리 50개의 질의를 등록한 후 실제 연산이 이루어지는 질의 개수를 증가시켜가며 평가 하였다. 성능평가 결과 D-order 기법을 이용하는 경우 기존 기법보다 빠른 질의 처리가 가능하다는 것을 확인할 수 있다.



[그림 11] 공간 순서화 기법에 따른 질의처리시간

제안한 D-order 기법은 Depth 레벨에 따라 서로 다른 결과를 보인다. 레벨이 1인 경우는 모든 질의영역에 대하여 우선적으로 정렬하기 때문에 처리되는 질의개수가 많아 질수록 빠른 처리결과를 보인다. 하지만 처리되는 질의 개수가 작은 경우는 레벨이 올라갈수록 빠른 처리 결과를 보인다. 성능평가 결과에서는 질의 개수가 20개 이하인 경우 레벨 3이 가장 빠르며, 20~30개인 경우는 레벨 2가 가장 빠르다는 것을 알 수 있다. 이러한 결과는 특정 개수 이하의 질의 처리시에는 레벨이 높을수록 빠른 결과를 가져오지만, 이용되는 질의가 많은 경우 레벨이 낮을수록 빠른 결과를 보인다.

5. 결론

본 논문에서는 공간 연속 질의 처리과정에서 발생하는 디스크 입출력 비용과 디스크 탐색 시간을 줄이기 위한 저장 구조를 제안하였다. 제안된 저장구조는 정적 질의 특성이 반영된 클러스터링 기법을 통하여 자주 이용되는 영역에 대해 빠른 검색과 데이터 고속로딩이 가능하다. 클러스터된 데이터의 저장을 위해 제안한 파일 구조는 클러스터링된 데이터를 중복 저장하여 데이터 고속로딩을 가능하게 하였으며, 질의에 자주 이용되는 데이터 영역을 디렉토리 정보와 인접하게 저장하여 데이터 탐색 시간을 줄일 수 있다. 또한 중첩 데이터의 효율적인 저장을 위해 제안된 무게중심 참조 기법은 최적의 공간에 데이터를 저장하여 불필요한 페이지 탐색을 줄일 수 있었

다. 향후 연구로는 클러스터링 영역 구축에 있어 공간 질의 처리 시스템과의 연계를 위한 접근 방법에 대한 시스템 명세가 필요하다. 또한, 클러스터링 영역의 변화에 따른 클러스터 구축시간 단축을 위한 연구 및 질의영역에 따른 클러스터링 수행시 동적 질의 처리에 따른 부하를 방지하기 위한 추가 연구가 필요하다.

참고문헌

[1] 이충호, 안경환, 이문수, 김주완, "u-GIS 공간정보 기술 동향", 전자통신동향분석, ETRI, 2007.

[2] Antomn Guttman, "R-tree: a dynamic index structure for spatial searching", In Proceeding of ACM-SIGMOD, 1984.

[3] T. Sellis, N. Roussopoulos and C. Faloutsos, "The R+-tree: A dynamic index for multi-dimensional objects", in Proc. of the 13th VLDB Conf., pp. 507-518, 1987.

[4] N. Beckmann and H. Kriegel, "The R*-tree: An efficient and robust access method for points and rectangles," in Proc. of ACM SIGMOD Conf, pp. 322-331, 1990.

[5] Oliver Gunther, "The design of the cell tree: An object-oriented index structure for geometric databases", in Proc. of the IEEE Conf. on Data Engineering, pp. 598-605, 1989.

[6] H. Bronnimann, B. Chen, M. Dash, P. Hass, Y. Qiao and P. Scheuermann, Bronimann et al. "Efficient Data-Reduction Methods for On-Line Association Rule Dicovery", NGDM'02, pp. 190-208, 2004.

[7] Gongde G et al, "Data Reduction Based on Spatial Partitioning", Springer-Verlag Berlin Heidelberg, 2001.

[8] K. V. 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.

[9] C. Jermaine, E. Omiecinski, W. G. Yee, "The partitioned exponential file for database storage management", The VLDB Journal, 2007.

[10] 조대수, 홍봉희., "정적 공간 데이타베이스에서 페이지 순서화를 위한 비용 모델", 정보과학회논문지, 컴퓨팅의 실체 제 6권 제 4호, 2000.

[11] OGC, The OpenGIS Simple Feature Specification for SQL Revision 1.1, 1999.

정 원 일(Weonil Chung) **[정회원]**

- 1998년 2월 : 인하대학교 전자계산공학과(공학사)
- 2004년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2004년 7월 ~ 2006년 7월 : 한국전자통신연구원 선임연구원
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 교수

<관심분야>
데이터스트림, 이동객체, 시스템보안