동적 객체 분해를 이용한 분할 기반의 공간 합병 조인의 개선

(An Improvement of Partition-Based Spatial Merge Join using Dynamic Object Decomposition)

최 용 진 † 이 용 주 †† 박 호 현 † 이 성 진 ††† 정 진 완 ††††
(Yong-Jin Choi) (Yong-Ju Lee) (Ho-Hyun Park) (Sung-Jin Lee) (Chin-Wan Chung)

요 약 공간 객체를 분할하는 비용이 크기 때문에, 기존의 객체 분할 기법들은 공간 조인 도중에 동적으로 객체를 분할하지 않는다. 본 논문은 객체 분할을 고려하지 않은 기존의 PBSM(Partition Based Spatial Merge-Join)에 적용될 수 있는 수정된 객체 분할 기법을 제시한다. 실세계 데이타의 한가지 특성은 객체들의 크기가 큰 차이를 보인다는 것이다. 본 논문에서는 이러한 특성을 바탕으로 전체 비용에 큰 영향을 미치는 적은 수의 큰 객체들만을 대상으로 분할을 적용한다. 이러한 수정된 객체 분할 기법은 공간 조인 도중에 큰 부담이 되었던 객체의 분할 비용을 크게 줄이면서 효율적인 여과-정제 단계를 수행한다. 여러 가지 실험 결과, 제안된 방법을 사용한 PBSM은 기존의 PBSM보다 월등히 향상된 성능을 보인다.

Abstract Traditional object decomposition techniques do not decompose spatial objects dynamically during spatial joins, because the object decomposition is very expensive. In this paper, we propose a modified object decomposition technique that can be applied in PBSM(Partition Based Spatial Merge-Join). In real-life data, there are much differences among the sizes of objects. We decompose only large objects with great effects on spatial joins. This technique decreases the decomposition cost of objects during spatial joins and enables efficient filter-refinement steps. Experiments show that the PBSM used with our proposed method performs significantly better than the traditional PBSM.

1. 서 론

공간 객체 관리에 대한 연구가 지리 정보 시스템(geographic information system : GIS), 컴퓨터 지원 설계(computer aided design : CAD) 등의 응용 분야에서 최근 많이 이루어 졌다. 복잡한 공간 객체들에 대한 중요한 질의 처리 분야로 크게 공간 선택과 공간 조인으로 나누어진다. 특히 공간 조인은 비용이 비싸기 때문에, 효율적인 공간 조인 처리를 위해서 많은 연구들이 진행되었다. 공간 조인은 공간 색인을 기반으로 하는 방법[1,2]과 공간 색인 사용 없이 처리하는 방법[3,4,5,6]이 있다. 복잡한 공간 질의에서의 중간 결과들에 대하여 공간 조인을 수행할 경우나, 병렬 환경에서 객체들이 동적으로 분할되어 여러 처리기에서 공간 조인을 수행할 경우에는 이미 구축된 색인을 이용할 수 없다. 색인을 이용할 수 없는 상황에서의 효율적인 공간 조인을 위하여 PBSM(Partition Based Spatial Merge-Join)[3]과 공간 해쉬 조인[4,5,6]을 기반으로 한 연구들이 사용될 수 있다. 그러나, 두 방법 모두 여과 단계에서 최소 경계 사각형(minimum bounding rectangle : MBR)만을 사용하기 때문에 객체의 근사화 정도가 너무 빈약하다. 이것은 여과 단계에서의 많은 후보 객체 쌍을 유발시킨다. 공간 조인의 전체적인 비용을 고려해 볼 때, 정제

· 본 논문은 과학기술부의 핵심 S/W 기술개발사업과 한국과학재단의 특정기초연구과제(과제번호 99-2-315-001-3)로부터 지원 받은 연구결과임.
† 비 회 원 : 한국과학기술원 전자전산학과
omni@islab.kaist.ac.kr
hhpark@islab.kaist.ac.kr
†† 정 회 원 : 상주대학교 컴퓨터공학과 교수
yongju@samback.sangju.ac.kr
††† 비 회 원 : (주)성수정보기술연구소 연구원
sjlee@islab.kaist.ac.kr
†††† 종신회원 : 한국과학기술원 전자전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 1999년 6월 22일
심사완료 : 2000년 2월 9일

단계의 비용이 많은 비중을 차지하기 때문에 효율적인 여과 단계는 중요한 의미를 갖는다. 근사의 정확도를 향상시키기 위해 기존의 객체 분할 방법[7,8]을 적용할 수 있다. 이러한 객체 분할 기법은 객체를 더욱 정확히 표현하여 향상된 여과 단계를 수행하고 분할된 부객체들로 정제 단계를 수행하여 전체적으로 많은 성능 향상을 가져왔다. 그러나, 기존의 객체 분할 기법들은 복잡한 객체들을 모두 분할 대상으로 하였다. 이처럼 모든 객체를 분할 대상으로 할 경우, 그 자체만으로도 큰 비용 부담이 된다. 따라서 공간 조인 도중에 객체를 분할하기 위해서는 분할 대상이 되는 객체 수를 줄일 필요가 있으며, 객체의 분할 단계도 제어할 필요가 있다.

본 논문에서는 객체 분할을 고려하지 않은 기존의 PBSM에 객체 분할 방식을 도입한 것이다. 기존의 객체 분할 방식은 수정 없이 PBSM에 적용될 수 없지만, 우리는 실세계 데이타의 특성을 고려하여 수정된 객체 분할 방식을 기존의 PBSM에 적용하였다. 전체적인 공간 조인 비용에 큰 영향을 미치는 적은 수지만 큰 객체들만을 분할하도록 기존의 객체 분할 방식을 수정하였다. 이러한 수정된 객체 분할 방식은 공간 조인 도중의 객체 분할 비용을 크게 줄이면서 효율적인 여과-정제 단계를 수행한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로써 PBSM과 객체 분할 기법을 기술하였다. 3장에서는 PBSM에 적용된 수정된 객체 분할 기법을 기술하였다. 4장에서는 기존의 PBSM과 본 논문에서 제안한 수정된 PBSM을 비교 분석하였다. 마지막으로 5장에서는 결론을 기술하였다.

2. 관련 연구

2.1 PBSM

복잡한 질의에서 중간 결과들에 대한 조인, 병렬 환경에서 객체들을 여러 처리기에 분할하여 조인을 수행할 경우, 미리 구축된 색인을 이용하지 못하게 된다. 이러한 상황에서 중간 결과가 메모리 크기보다 클 경우, 단순한 중첩 조인 방법이나 색인을 실시간에 구축하여 조인하는 방법이 가능하다. 그러나, 이러한 방법보다 중간 결과들을 분할하여 조인하는 방법인 PBSM 또는 공간 해쉬 조인이 일반적으로 더 향상된 성능을 보인다[3,4].

PBSM은 객체들을 구획(partition)에 할당하는 단계와 주 메모리의 크기보다 작게 조정된 두 클래스의 대응되는 구획들을 조인하는 단계로 이루어져 있다. 공간 조인을 위하여 구획 할당 단계에서는 먼저 두 클래스에서 같은 영역에 포함된 객체들을 주 메모리에 올려놓고 연산할 수 있도록 구획 수를 결정한다. 객체의 MBR이 겹치는 모든 구획에 그 객체의 식별자와 MBR을 할당한다. 구획 할당 단계에서 모든 객체들의 할당이 끝나면 조인 단계가 수행된다. 두 클래스의 모든 대응되는 구획끼리 조인을 수행한다. 객체가 여러 구획에 할당될 수 있기 때문에, 구획들간의 조인이 끝나면 식별자 쌍을 정렬하여 중복되는 식별자 쌍을 제거해야 한다. 이렇게 여과 단계가 끝나면, 정제 단계를 거쳐 최종 결과를 얻는다. 그러나, 여과 단계에서 MBR만을 사용하기 때문에 객체의 근사화 정도가 너무 빈약하여 많은 후보 객체 쌍을 유발시킨다. 공간 조인의 전체적인 비용을 고려해 볼 때, 정제 단계의 비용이 많은 비중을 차지하기 때문에 효율적인 여과 단계는 중요한 의미를 갖는다.

2.2 객체 분할 기법

최근에는 여과 단계뿐만 아니라, 정제 단계를 효율적으로 수행할 수 있는 객체 분할 기법이 소개되었다. 객체 분할은 복잡한 하나의 객체를 단순한 여러 부 객체들로 분할한 것이다. 이러한 객체 분할 기법은 객체를 더욱 정확히 근사화하여 후보 객체의 쌍을 많이 줄였으며, 원래의 객체를 대신하여 분할된 부 객체들로 정제 단계를 수행하기 때문에 많은 비용이 요구되는 기하 연산 비용을 감소시켰다.

최근 연구된 객체 분할의 예로서 사다리꼴 분할 기법을 개선한 DMBR(Decomposed Minimum Bounding Rectangle) 방식[8]을 소개한다. 하나의 객체를 사다리꼴 형태의 여러 부 객체들로 분할하여 TR*-tree를 구성한 객체 분할 기법[7]은 공간 질의 처리에서 우수한 성능을 보여 객체 분할의 중요성을 인식하게 하였다. DMBR 기법에서는 사다리꼴 분할 기법보다 약 70% 절약된 저장 공간이 사용되었다. 이것은 적은 메모리를 요구하기 때문에, 공간 조인에 있어서 좀더 나은 성능을 갖는 객체 분할 기법으로 제시되었다. 본 논문에서 제시된 객체 분할 기법이 기본적으로 DMBR 방식을 수정한 형태이기 때문에, 여기서 간단히 DMBR 객체 분할 방식을 소개한다. DMBR 방식은 객체의 MBR이 절반으로 분할 되도록 y축과 x축을 번갈아 가면서 분할하여 부객체들을 반복적으로 생성하는 것이다. 분할은 부객체의 크기가 어떤 한계치에 도달할 때까지 이루어진다. 이 논문에서 언급하는 객체의 크기는 객체의 MBR 크기를 나타낸다. 그림1은 한계치가 원래 객체의 1/4 크기일 때, 객체 분할 과정을 보인 것이다. 첫 번째 분할 단계에서는 객체의 MBR이 절반으로 분할 되도록 y축으로 분할이 이루어진다. 생성된 2개의 부객체들 중에서 왼쪽 부객체는 원래 객체 크기의 1/4보다 크기 때문에 분할

이 계속 이루어지고, 원래 객체 크기의 1/4보다 작은 오른쪽 부객체에 대한 분할은 더 이상 이루어지지 않는다. 두 번째 분할 단계에서는 왼쪽 부객체의 MBR이 절반으로 분할 되도록 x축으로 분할이 이루어진다. 이렇게 생성된 2개의 부객체들은 모두 원래 객체 크기의 1/4보다 작기 때문에, 더 이상 분할이 이루어지지 않는다.

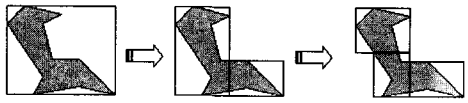

그림 1 DMBR 객체 분할 방식

실세계 데이타는 객체의 크기가 증가할수록 객체의 복잡도도 증가하여 큰 객체일수록 많이 분할하는 것이 적절하지만, DMBR 방식은 분할 대상이 되는 객체들은 모두 동일한 분할 단계까지만 분할하는 단점이 있다.

3. 동적 객체 분할

이 장에서는 공간 조인 도중에 객체 분할이 적용될 수 있다는 근거를 제시하기 위하여 실세계 데이타의 특성을 먼저 기술하고, 이러한 특성을 반영한 수정된 객체 분할 기법을 소개한다.

3.1 실세계 데이타의 특성

본 논문에서 다루고 있는 실세계 데이타는 여러 문헌에서 가장 많이 인용되고 있는 Tiger/Line 데이터이다 [3,4,7,9,10,11,12]. 실세계 데이타의 특성을 분석하기 위하여 캘리포니아주로부터 다각형 형태인 통계적 영역을 의미하는 CTBNA 객체를 채택하였다. 그림2는 Tiger/Line 데이타의 캘리포니아주를 나타낸다. 객체 수는 5200여 개이며, 객체 당 평균점의 수는 136개이다. 제시

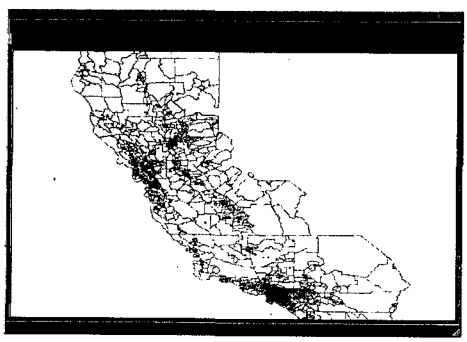

그림 2 Tiger/Line 데이타의 캘리포니아주

된 여러 가지 통계들은 다른 지역에서도 동일한 특성을 갖는다.

객체의 MBR(minx,miny,maxx,maxy)로부터 MBR의 xl=maxx-minx, yl=maxy-miny라고 하자. 그리고 객체의 크기는 MBR의 면적(xl*yl), 객체의 복잡도는 객체가 포함하는 꼭지점의 수라고 하자. 객체들은 MBR 면적에 따른 히스토그램의 구간으로 분류되었고, 히스토그램의 구간은 인접한 구간에 속한 객체들의 평균 MBR 크기가 4배 증가되도록 설정되었다. 객체 크기에 따른 객체의 복잡도에 관한 통계, 객체 크기에 따른 객체의 수에 대한 통계의 2가지 형태로 실세계 데이타의 특성을 분석하였다. 그리고 객체 크기에 따른 객체의 수에 대한 통계를 이용하여 객체 크기에 따른 조인 기대치를 제시하였다.

그림3는 객체 크기에 따른 객체 수를 보여주고 있다. 실세계 데이타의 특성은 객체의 크기가 증가할수록 객체의 복잡도는 증가하며, 객체 크기에 대한 차이가 크다는 점이다.

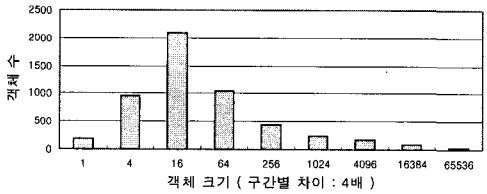

그림 3 객체 크기에 따른 객체 수 (Tiger/Line)

공간 조인 시, 하나의 공간 객체가 다른 공간 클래스의 공간 객체들과의 여과 단계의 결과인 후보 조인 쌍이 될 수 있는 확률을 고려해 볼 때, 상대적으로 큰 객체는 작은 객체보다 많은 후보 조인 쌍이 생길 수 있음을 알 수 있다. 그림3의 객체 크기에 따른 객체 수에 대한 통계 정보를 이용하여 객체 크기에 따른 조인 기대치를 구할 수 있다.

R	:	클래스 R
R^i	:	MBR 면적으로 분류한 히스토그램의 구간 i에 속한 R의 서브클래스
$\|R^i\|$	:	R^i의 객체 수
R^i_{avgxl} (or R^i_{avgyl})	:	R^i의 평균 MBR의 xl (또는 yl)

표 1 심볼의 정의

전체 조인공간을 1*1이라고 가정하자. 표1은 조인 기대치를 나타내기 위한 심볼들의 정의다.

정리1) 두 MBR s, t에서 s_{xl}, t_{xl}은 각각 s, t의 xl이고, s_{yl}, t_{yl}은 각각 s, t의 yl이라고 하자. s, t의 겹칠 확률은 $(s_{xl}+t_{xl})*(s_{yl}+t_{yl})$이다.

증명) 참고문헌 [13]을 참조.

클래스 P와 Q의 공간 조인을 고려해 보자. P^i에 대한 객체 수와 평균 MBR의 정보가 주어졌고, Q^i에 대한 객체 수와 평균 MBR의 정보가 주어졌다고 가정하자. (단, $1 \le j \le n$, n은 구간의 수이다)

정리2) P^i와 Q의 조인 기대치는 P^i와 Q에 속한 각 객체들의 MBR간 겹칠 확률의 합으로, $\sum_{j=1}^{n}\{|P^i|*|Q^j|*(P^i_{avgxl}+Q^j_{avgxl})*(P^i_{avgyl}+Q^j_{avgyl})\}$로 나타낼 수 있다. (단, n은 구간의 수이다)

증명) P^i와 Q에 속한 각 객체들의 MBR간의 겹칠 확률의 합은 각각 P^i와 Q에 속한 객체들간의 겹칠 확률과 이러한 관계가 성립될 수 있는 경우의 수로 고려할 수 있다. $P^i \bowtie Q$는 $\sum_{j=1}^{n}(P^i \bowtie Q^j)$의 형태와 같이, P^i와 Q의 조인 기대치는 P^i와 Q^j의 조인 기대치의 합이다($1 \le j \le n$). 각각 P^i와 Q^j에 속한 한 객체들간의 겹칠 확률은 평균 MBR간의 겹칠 확률로 고려될 수 있으며, 정리1과 같은 형태로 나타낼 수 있다. P^i와 Q^j에 속한 객체들의 MBR간 겹칠 경우의 수는 각각 P^i와 Q^j에 속한 객체 수의 곱이다. 결국, P^i와 Q의 조인 기대치는 $\sum_{j=1}^{n}\{|P^i|*|Q^j|*(P^i_{avgxl}+Q^j_{avgxl})*(P^i_{avgyl}+Q^j_{avgyl})\}$이다.

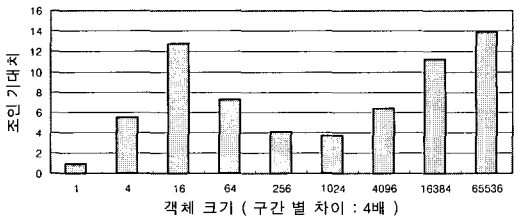

그림 4 객체 크기에 따른 조인 기대치

그림4는 동일한 캘리포니아주의 데이타로 자신(self)-조인하여 얻은 조인 기대치의 결과이다. 그림4는 의미하는 것은 객체 수는 적지만 큰 객체들의 조인 기대치가 높다는 사실이다. 큰 객체와 작은 객체간의 plane-sweep 알고리즘[7]을 이용한 정제 단계에서의 비용을 고려해 볼 때, 큰 객체가 차지하는 비용이 작은 객체가 차지하는 비용에 비해 훨씬 크다. 이 두 가지 사실은 적은 수지만 큰 객체들이 총 공간 조인 비용 면에서 큰 요소가 됨을 의미한다. 따라서 적은 수지만 큰 객체들만을 대상으로 효율적인 정책을 고려해도 전체 성능에 큰 변수가 될 수 있다. 우리는 효율적인 공간 조인을 위해 여과-정제 단계를 모두 이용한 기존의 객체 분할 기법을 고려하였다. 하지만 본 연구에서는 실세계 데이타의 특성을 고려하여 큰 객체들만을 분할하기 때문에, 공간 조인 도중에 부담이 되었던 객체 분할 비용을 크게 줄이면서 전체 성능에 큰 효과를 얻을 수 있다.

그림5는 Tiger/Line 데이타와 다른 실세계 데이타의 객체 크기에 대한 객체 수를 보인 것이다. 이 실세계 데이타는 문헌에서 두 번째로 많이 인용되는 Sequoia 데이타[3,14,15,16]이며, 객체 수는 7900여 개, 객체 당 평균점의 수는 34개이다. Tiger/Line 데이타와 동일하게 Sequoia 데이타 또한 큰 객체들을 포함한다.

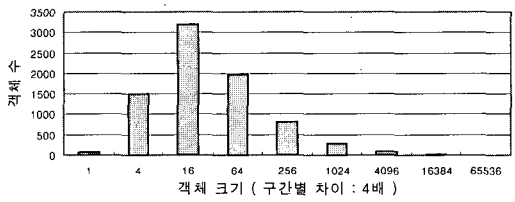

그림 5 객체 크기에 따른 객체 수 (Sequoia)

3.2 동적 객체 분할 기법

3.2.1 수정된 객체 분할 기법

이 논문에서 제시한 수정된 DMBR 방식은 크게 세 가지 측면으로 DMBR 방식을 수정하였다. 첫째로, 객체 분할을 전체적인 공간 조인 비용 면에서 크게 작용하는 적은 수의 큰 객체들로 한정하였다. 반면에, DMBR 방식은 복잡한 객체를 모두 분할 대상으로 간주한다. 둘째로, 분할 대상이 되는 객체들도 상대적으로 큰 객체를 많이 분할한다. 3.1에서 기술한 바와 같이 실세계 데이타는 객체의 크기가 증가할수록 객체의 복잡도도 증가하기 때문에, 큰 객체일수록 많이 분할하는 것은 적절한 정책이다. 반면에, DMBR 방식은 분할 대상이 되는 객체들은 모두 동일한 분할 단계까지만 분할한다. 그림4는 두 가지 수정된 형태가 반영된 것을 나타낸다. 셋째로, 분할 형태는 객체의 MBR을 정확히 이등분하는 방식으로만 분할하는 간단한 정책을 취하였다. 반면에, DMBR 방식은 객체를 분할하는 단계에서 반드시 2개의 부객체가 생성되도록 제약을 두었다. 객체의 MBR을 정확히 이등분하여 3개 이상의 부객체가 생성될 경우, 추가 비

용을 들여 2개의 부객체가 되도록 한다. 3.2.3에서 설명하게 될 부객체들이 구획(partition)에 할당되는 방식을 살펴보면, DMBR방식처럼 추가의 비용을 들여 공간 조인 도중에 객체를 분할할 필요성이 없다. 그림6은 세 번째의 차이점을 나타낸다.

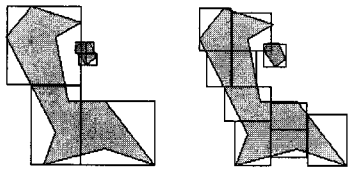

그림 6 수정된 DMBR 방식 (1)

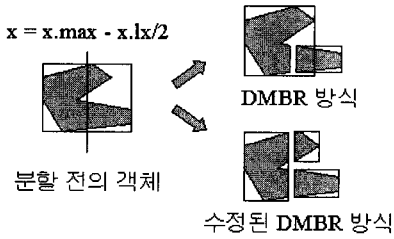

그림 6 수정된 DMBR 방식 (2)

세가지 측면에서 수정된 DMBR 방식은 전체적인 공간 조인 비용 면에서 큰 영향을 미치는 큰 객체들일수록 근사화의 질을 더욱 높여 여과-정제의 효율을 크게 한다. 소수의 큰 객체들만을 대상으로 간단한 분할 정책을 취하였기 때문에, 공간 조인 도중에 적은 부담으로 객체 분할을 적용시켜 큰 효과를 얻을 수 있다.

```
procedure decompose(IP,OP,XL,YL)
for each polygon of IP do
   if MBR.xl > XL and MBR.yl > YL and !rectangle then
       if MBR.xl > MBR.yl then
           decompose polygon to x = MBR.maxx - MBR.xl/2;
           IP1 := decomposed polygons;
           decompose(IP1,OP1,XL,YL);
       else
           decompose polygon to y = MBR.maxy - MBR.yl/2;
           IP1 := decomposed polygons;
           decompose(IP1,OP1,XL,YL);
   else
       OP1 := polygon;

   OP := union(OP,OP1);
end
```

그림 7 수정된 객체 분할 알고리즘

그림7은 수정된 객체 분할 알고리즘이다. PBSM은 조인하기 전에 객체의 수를 이용하여 구획의 수를 결정한다. 이것은 조인 전에 그 상황에 맞는 객체들에 대한 기본적인 통계 정보를 동적으로 얻을 수 있음을 의미한다. 객체 분할의 기준이 되는 XL과 YL은 조인 전에 동적으로 얻어진 기본적인 통계 정보(객체의 평균 크기)를 이용하여 구할 수 있다. 4장에서 설명될 여러 가지 실험 결과로 인해, 객체 분할의 기준은 객체의 평균 크기의 4배가 되도록 고려할 수 있다. 조인 순간에 영향을 미치는 객체들의 기본 통계 정보(객체의 평균 크기)를 이용한 XL과 YL을 사용하기 때문에, 계속적인 객체의 삽입, 삭제, 갱신 등으로 공간 데이타베이스가 변경되었을지라도 그 상황에 맞는 객체 분할 기준을 채택할 수 있다. 입력에 해당되는 IP는 분할 대상이 되는 객체를 의미하며, 결과에 해당되는 OP는 분할된 부 객체들을 의미한다.

3.2.2 PBSM에 적용된 객체 분할 기법

PBSM은 크게 객체를 구획(partition)에 할당하는 분배 단계와 서로 같은 구획에 할당된 객체들끼리의 조인 단계로 이루어져 있다. 객체가 구획에 할당되는 방식은 객체의 MBR과 겹치는 모든 구획에 객체의 OID와 MBR의 정보를 할당하는 형태로 이루어진다[3]. 이와 마찬가지로, 객체 분할로 생성된 부객체들도 동일한 방식으로 구획에 할당될 수 있다. 그림8은 하나의 객체가 구획에 할당되는 기존의 방식을 나타낸 것이며, 그림9는 큰 객체의 분할로 생성된 여러 부객체들이 구획에 할당되는 방식을 나타낸 것이다.

객체 분할이 고려되지 않은 기존의 방식대로 객체를 구획에 할당할 경우, 그림8과 같이 객체는 구획 Part0, Part1, Part2에 모두 할당된다. 객체 분할이 고려된 방식에서의 부객체들을 구획에 할당할 경우, 그림9와 같이 부객체들이 구획 Part1, Part2에만 할당되고 분할 Part0

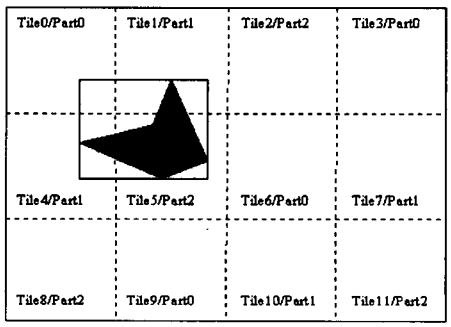

그림 8 객체의 구획 할당 방식

에는 할당되지 않는다. 이와 같이 객체 분할 개념이 이용되면 객체에 대한 근사화 정도의 질을 높여 Part0에서 생기는 불필요한 정제 연산을 줄이게 된다. 또한, Part1, Part2에서 필요로 하는 정제 연산은 원래의 객체를 대신하여 각 구획에 할당된 부객체로 정제 연산을 수행하기 때문에 그만큼 정제 연산의 비용을 줄이게 된다.

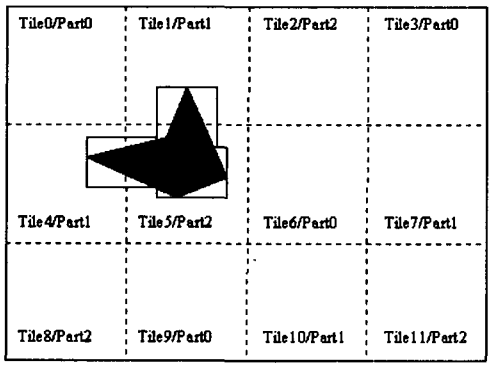

그림 9 분할된 부객체들의 구획 할당 방식

객체 분할이 고려될 경우, 객체의 구획 할당에 필요한 정보는 객체의 OID, 부객체의OID, 그리고 부객체의 MBR이 된다. 객체 분할로 생성된 부객체의 좌표 정보는 다른 파일에 저장되어 정제 연산이 필요할 때에 부객체의 OID를 통하여 제어될 수 있다.

4. 성능 분석

실험을 위하여 사용된 H/W는 메인 메모리 384MB인 Sun Ultra 2이며, 운영체제는 Solaris 2.5.2이다. 저장시스템으로는 SHORE[17]를 이용하였다. 실험을 위한 파라미터인 버퍼의 크기는 2MB, 8MB, 24MB 그리고 48MB를 고려하였다. 구획(partitions) 수는 PBSM[3]의 구획 수를 구하는 식을 적용하여 각 버퍼 크기에 맞게 결정하였다. PBSM(4.3절)에서 언급된 바와 같이 타일(tiles) 수의 변화는 총 실행 시간에 미세한 영향을 미친다. 따라서, 타일의 수는 PBSM에서 채택한 방식과 동일하게 1024로 고정시켜 버퍼의 크기에 상관없이 실험하였다.

Tiger/Line 데이타를 실험 데이타로 선정하였으며, 워싱턴주, 오레곤주, 캘리포니아주, 네바다주, 유타주, 아리조나주, 콜로라도주, 그리고 뉴멕시코주의 8개 주를 고려하였다. 표2는 2개의 주 또는 4개의 주를 하나로 묶은 3지역에 대한 객체 수와 평균점의 수를 나타낸다.

표 2 3지역의 객체 수와 평균 점의 수

3지역	객체 수	평균점의 수
a	1629	304
b	3048	173
c	1127	298

표3은 실험을 위하여 메모리 로딩이 불가능하도록 조정한 3개 클래스의 객체 수와 평균점의 수를 나타내다. 즉, 표2의 3지역을 각각 40배 하였으며, 또한 중복된 객체들이 겹치지 않도록 객체들의 좌표를 이동하여 클래스를 구성하였다.

표 3 실험 데이타 (Tiger/Line)

클래스	객체 수	평균점의 수	총 크기
A	65160	304	302MB
B	121920	173	322MB
C	45080	298	205MB

성능 평가는 객체 분할 기법을 고려하지 않은 기존의 PBSM과 수정된 객체 분할 기법을 고려한 PBSM의 비교로 이루어졌다. 수정된 객체 분할 기법이 여과 단계에서만 고려된 경우와 여과-정제의 두 단계에 모두 고려된 경우로 나누어 비교되었다. 객체 분할을 여과 단계에서만 고려한 경우는 분할된 부객체들의 MBR만을 저장하고 부객체들의 좌표 정보는 저장하지 않아 정제 단계에서 원래의 객체(original object)를 그대로 이용한 것이다. 여과-정제의 두 단계를 모두 고려한 경우는 분할된 부객체들의 MBR을 저장할 뿐만 아니라 부객체들의 좌표 정보도 임시로 저장하여 정제 단계에서 분할된 부객체들(decomposed object)의 좌표 정보를 이용한 것이다. 공간 조인의 술어로 intersect(또는 non-disjoint)를 적용하였다. 객체 분할 알고리즘에 적용될 객체의 분할 기준은 조인에 참가하는 두 클래스에 포함된 객체들의 평균 xl, yl의 배수로 하였다. 실험은 각 클래스의 조합 형태인 3가지로 이루어졌다. 분할 기준이 되는 부객체의 크기는 객체의 49배 크기부터 객체의 근사화 정도를 점점 높여 1/16(0.0625)배 크기까지 분할되도록 고려하였다.

그림10~그림12은 버퍼 크기가 8MB인 공간 조인의 총 비용을 나타낸 실험 결과다. 기존의 PBSM에 객체 분할이 단계별로 고려된 성능을 나타낸다. 3가지 실험 결과에서 모두 분할 기준이 되는 부객체의 크기가 평균 객체 크기의 4배로 분할될 때, 가장 좋은 성능을 보였다. 이러한 실험적 근거를 바탕으로 가장 좋은 성능을 얻기 위해서 기존의 PBSM에 수정된 DMBR 방식에

적용될 수 있는 분할 기준이 되는 부객체의 크기를 평균 객체 크기의 4배로 선택할 수 있다. 일반적으로 객체 분할을 여과 단계에만 적용한 경우보다 여과-정제 단계에 모두 적용한 경우가 향상된 성능 결과를 나타낸다. 이러한 이유는 비록 부객체들을 정제 단계에서 이용하기 위해 임시로 저장하는 추가 비용은 들지만, 정제 단계에서 부객체를 이용한 기하연산의 비용이 크게 감소하기 때문이다. 그러나, 분할 기준이 되는 부객체의 크기가 0.0625배일 때는 오히려 성능이 나빠졌다. 이러한 이유의 주원인은 부객체의 크기를 작게 채택할수록 객체 분할의 비용이 크게 증가하기 때문이다. 다른 분할 기준들과는 달리 0.0625배의 분할 기준에서의 비용은 객체 분할을 여과-정제 단계에서 모두 고려한 경우보다 여과 단계만을 고려한 경우가 크게 나타났다. 분할된 부객체를 이용한 기하연산의 이점보다 부객체들의 정보를 임시로 저장하는 단점이 커지기 때문이다.

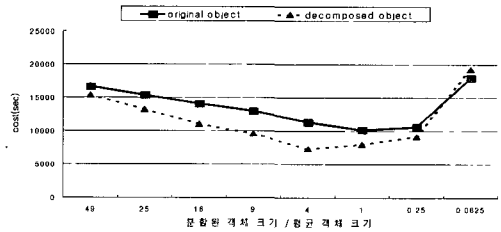

그림 10 A와 B의 공간 조인 결과

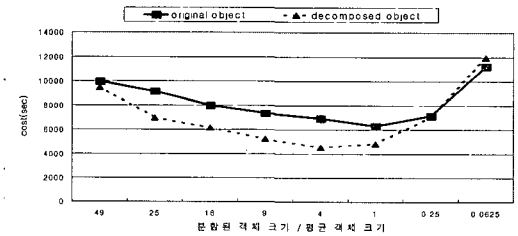

그림 11 B와 C의 공간 조인 결과

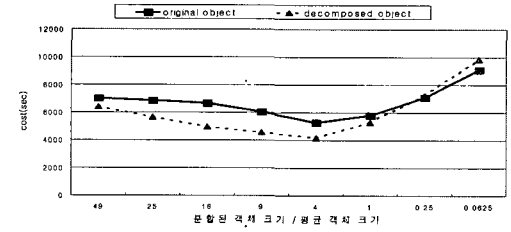

그림 12 A와 C의 공간 조인 결과

그림13는 버퍼 크기 8MB에서 기존의 PBSM과 수정된 PBSM의 공간 조인 결과를 비교한 것이다. 수정된 PBSM에는 가장 좋은 성능 향상을 보이는 수정된 DMBR방식을 적용시킨 것이다. 즉, 분할된 부객체들을 임시로 저장하여 정제 단계에서 고려한 경우와 분할 기준이 되는 부객체 크기가 평균 객체의 4배가 되도록 고려한 경우이다. 수정된 PBSM은 기존의 PBSM과 비교하여 약 50~60%의 성능 향상을 보인다.

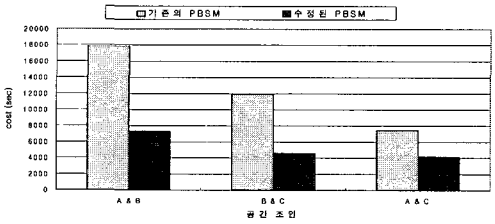

그림 13 기존의 PBSM과 수정된 PBSM의 공간 조인 결과

그림14는 클래스 A와 C의 조인에서 각 분할 기준에 따른 객체 분할과 분할된 부객체들을 임시로 저장하는 비용만을 나타낸 것이다. 분할 대상 수가 조금씩 증가되어도 공간 조인 도중의 객체 분할 비용은 아주 크게 증가되는 것으로 나타났다. 이러한 이유의 부객체의 크기를 작게 채택할수록 객체 분할의 비용이 크게 증가하며, 부객체들의 정보를 임시로 저장하는 비용이 크게 증가하기 때문이다. 이것은 많은 분할을 요구하는 기존의 객체 분할 기법들이 수정 없이 PBSM에 채택될 수 없음을 나타낸다.

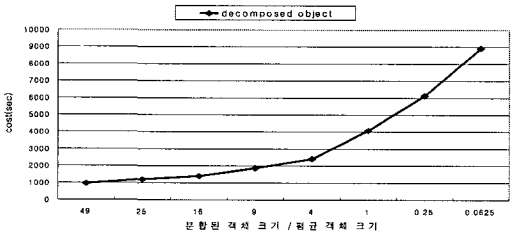

그림 14 A와 C의 공간 조인의 동적 객체 분할 비용

그림15는 버퍼의 크기에 따른 기존의 PBSM과 수정된 PBSM의 공간 조인 결과를 비교한 것이다. 수정된 PBSM에는 가장 좋은 성능 향상을 보이는 수정된 DMBR방식을 적용시킨 것이다. 고려된 버퍼 크기는

2MB, 8MB, 24MB, 48MB이다. PBSM의 구획 수를 구하는 식을 적용하여 각 버퍼 크기에 따른 구획 수를 결정하였으며, 그 결과에 따른 공간 조인 결과를 보인 것이다. PBSM[3]에서 버퍼의 효과는 크지 않다는 언급과 마찬가지로 우리의 실험에서도 버퍼의 효과는 크지 않았으며, B와 C, A와 C의 공간 조인에서도 버퍼의 효과는 크지 않았다.

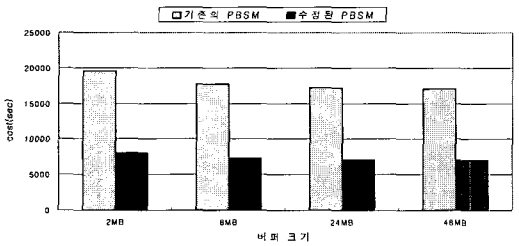

그림 15 A와 B의 공간 조인 결과

5.결론

본 논문에서는 객체 분할을 고려하지 않은 기존의 PBSM에 적용될 수 있는 수정된 DMBR 객체 분할 기법을 제시했다. 기존의 PBSM은 여과 단계에서 객체의 가장 단순한 표현 형태인 MBR을 이용하였기 때문에, 비싼 기하 연산을 필요로 하는 정제 단계에 부담을 주었다. 효율적인 여과-정제 단계를 위해 근사의 정확도를 향상시킨 객체 분할 기법이 연구되었다. 그러나, 객체 분할을 고려하지 않은 기존의 PBSM에 객체를 공간 조인 도중에 분할하는 것은 비용 면에서 큰 부담이 되어, 기존의 객체 분할 기법을 수정 없이 PBSM에 도입할 수 없다. 우리는 실세계 데이타의 특성을 고려하여 전체적인 공간 조인 비용에 큰 영향을 미치는 적은 수지만 큰 객체들만을 분할하도록 기존의 객체 분할 방식을 수정하였다. 이러한 수정된 객체 분할 방식은 공간 조인 도중의 객체 분할 비용을 크게 줄이면서 효율적인 여과-정제 단계를 수행한다. 기존의 PBSM과 수정된 DMBR 방식이 고려된 PBSM의 성능을 비교한 결과, 제안된 방식이 고려된 PBSM이 약 50~60%의 향상된 결과를 보였다.

참 고 문 헌

[1] Beckmann N, "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles," Proceedings of ACM SIGMOD, pp. 322-331, 1990.

[2] T. Brinkhoff, H. P. Kriegel, and B,. Seeger, "Efficient Processing of Spatial Joins Using R-trees," Proceedings of ACM SIGMOD, pp. 237-246, 1993.

[3] Jignesh M. Patel and David J. DeWitt, "Partition Based Spatial-Merge Join," Proceedings of ACM SIGMOD, pp. 259-270, 1996.

[4] M. L. Lo and C. V. Ravishankar, "Spatial Hash-Joins," Proceedings of ACM SIGMOD, pp. 247-258, 1996.

[5] M. L. Lo and C. V. Ravishankar, "Generating Seeded Trees from Data Sets," The Fourth International Symposium on Large Spatial Databases, pp. 328-347, 1995.

[6] M. L. Lo and C. V. Ravishankar, "Spatial Joins Using Seeded Tree," Proceedings of ACM SIGMOD, pp. 209-220, 1994.

[7] T. Brinkhoff, H. P. Kriegel , R. Schneider, and B. Seeger, "Multi-Step Processing of Spatial Joins," Proceedings of ACM SIGMOD, pp. 197-208, 1994.

[8] Y. J. Lee, H. H. Park, N. H. Hong, and C. W. Chung, "Spatial Query Processing Using Object Decomposition Method," Proceedings of Information and Knowledge Management(CIKM), pp. 53-66, 1996.

[9] G. Zimbrao and J. M. Souza, "A Raster Approximation for the processing of Spatial Joins," Porceedings of VLDB, pp. 558-569, 1998.

[10] B. of Census, "Tiger/lines precensus files : 1994 technical documentation, Technical report," Bureau of Census, Washington, DC, 1994.

[11] N. Koudas and K. C. Sevcik, "Size Separation Spatial Join," Processings of ACM SIGMOD, pp. 324-335, 1997.

[12] L. Arge, O. Procoiuc and S. Ramaswamy, "Scalable Sweeping Spatial Join," Proceedings of VLDB, pp. 570-581, 1998.

[13] I. Kamel, C. Faloutsos, "On Packing R-trees," Proceedings of Information and Knowledge Management(CIKM), pp. 490-499, 1993.

[14] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith, "The Sequoia 2000 Benchmark," Proceedings of ACM SIGMOD, pp. 2-11, 1993.

[15] M. J. Carey, D. J. DeWitt, M. J. Franklin, N. E. Hall, M. L. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. G. Tsatalos, S. J. White, and M. J. Zwilling, "Shoring Up Persistent Applications," Proceedings of ACM SIGMOD, pp. 383-394, 1994.

최 용 진

1997년 성균관대학교 정보공학과(학사). 1999년 한국과학기술원 전산학과(석사). 1999년 ~ 현재 한국과학기술원 전자전산학과 박사과정. 관심분야는 공간 데이타베이스, 시공간 데이타베이스

이 용 주

1983년 울산공과대학 졸업. 1985년 한국과학기술원 산업공학과 정보검색 전공(석사). 1997년 한국과학기술원 정보 및 통신공학과 컴퓨터공학 전공(박사). 1985년 ~ 1989년 시스템공학연구소 연구원. 1987년 ~ 1988년 일본 IBM TRL 연구소 연구원. 1989년 ~ 1994년 삼보컴퓨터 근무. 1997년 ~ 1998년 한국과학기술원 Post Doc.. 1998년 ~ 현재 상주대학교 컴퓨터공학과 전임강사. 관심분야는 공간 데이타베이스, GIS, 정보검색

박 호 현

1987년 서울대학교 계산통계학과(학사). 1995년 한국과학기술원 정보 및 통신공학과(석사). 1995년 ~ 현재 한국과학기술원 전자전산학과 박사과정. 1987년 ~ 1993년 삼성전자 통신개발실. 1993년 ~ 현재 삼성전자 학술파견팀. 관심분야는 GIS, 질의처리, 멀티미디어, 컴퓨터 네트워크

이 성 진

1984년 경북대학교 전자공학과(학사). 1985년 경북대학교 전자공학과(석사). 1986년 ~ 1992년 (주)금성전선연구소. 1993년 ~ 현재 한국과학기술원 정보통신및통신공학과 박사과정. 1994년 ~ 현재 (주)성수정보기술 연구소. 관심분야는 분산시스템, 객체지향데이타베이스, GIS

정 진 완

1973년 서울대학교 공과대학 전기공학과(학사). 1983년 University of Michigan 컴퓨터공학과(박사). 1983년 ~ 1987년 미국 GM연구소 선임연구원. 1987년 ~ 1993년 미국 GM연구소 책임연구원. 1993년 ~ 1996년 한국과학기술원 정보및통신공학과 부교수. 1996년 ~ 현재 한국과학기술원 전자전산학과 부교수. 관심분야는 GIS, 객체지향 데이타베이스, 멀티미디어 데이타베이스, 분산 데이타베이스, CIM