공간 데이터베이스에서 효율적인 여과를 위한 격자 분할 색인 기법

박 정 민*, 김 성 희*, 이 순 조**, 배 해 영*

* 인하대학교 전자계산공학과

** 서원대학교 컴퓨터정보통신공학부

mycomman@netsgo.com

Grid Decomposition Indexing Method for Efficient Filtering in Spatial Database

Jeong-Min Park*, Sung-Hee Kim*, Soon-Jo Lee**, Hae-Young Bae*

* Dept. of Computer Science, Inha University

** School of Computer, Information & Communication, Seowon University

Abstract

고비용의 공간 연산을 수행해야 하는 공간 질의 처리는 여과-정제의 2단계 처리가 일반적이다. 그러나, 2단계 색인 방법은 여과율이 좋지 못한 단점이 있으므로, 최근 다단계 여과 과정이 많이 연구되고 있다. 다단계 여과 과정은 1차 여과된 객체에 대하여 더욱 정밀한 필터를 적용함으로써 후보 객체 수를 줄이는 방법으로 접근하고 있으나, 여러 번의 여과 단계를 거치므로 수행 시간이 길어지고 추가 정보 유지로 인한 저장 공간 낭비 등의 단점이 있다.

본 논문에서는 전체 공간 영역을 격자로 분할하고, 객체를 격자 위에 구성하는 2단계의 공간 색인 방법을 제안한다. 제안된 색인 방법은 Dead Space의 크기를 줄이고, 한 번의 여과 과정으로 높은 여과율을 갖는다.

1. 서론

공간 데이터베이스 시스템에서는 대용량의 공간 데이터를 효율적으로 관리, 유지하는 부분이 중요하다.[1,2] 특히, 공간 객체를 대상으로 하는 공간 연산은 기존의 문자열, 수치 데이터에 대한 연산과는 달리, 고비용의 연산을 필요로 한다. 따라서, 공간 색인을 이용하여 적은 연산으로 후보 객체를 찾아내는 여과 과정과, 후보 객체에 대해서만 공간 연산을 적용하여 결과 객체를 찾아 내는 정제 과정의 2단계 질의 처리 과정이 연구되었다.[1,2] 현재 가장 널리 이용되는 공간 색인의 종류에는 R-tree 계열이 있으나, R-tree 계열 색인은 여과 과정에서 최소 경계 사각형(MBR)을 사용하므로, 선 및 임의 형태를 갖는 공간 객체의 경우 Dead Space가 크게 발생하여, 여과율이 떨어진다.[1,2,7] 이러한 문제점을 극복하기 위하여, 최근 다단계 여과 방법들이 연구되고 있다.[1,2,3,8] 다단계 여과는 1차 여과된 객체들에 대하여 추가적인 여과 단계를 적용함으로써, 후보 객체를 최소화하는 방법이다. 그러나, 다단계 여과 방법은 여러 단계의 여과 방법을 수행함으로 인하여 수행 시간이 길어지고, 다단계 여과를 위한 추가적인 정보 때문에 저장 공간이 낭비되는 단점이 있다.[7,8]

본 논문에서는 전체 공간 영역을 격자로 분할하고 공간 객체를 격자 위에 구성함으로써, 객체의 실제 형태에 유사한 근사 데이터를 색인으로 유지하는 방법을 제안한다. 본 논문에서 제안한 방법은 Dead Space의 크기를 줄일 수 있으므로 여과율을 크게 향상시킬 수 있고, 한 번의 여과로 여과 과정이 종료되므로 다단계 여과 기법보다 더 빠르게 수행된다.

본 논문의 구성은 다음과 같다. 2장에서는 공간 질의 처리에 이용되는 색인과 다단계 여과에 대한 관련 연구를 살펴 본다. 3장에서는 제안하는 색인의 구조와 알고리즘 대하여 기술한다. 마지막으로 4장에서는 본 연구의 결론에 대하여 언급한다.

* 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임.

2. 관련 연구

2.1 여과-정제를 이용한 2단계 질의 처리 방법

실제 객체를 대상으로 질의에 대한 결과를 얻어내는 과정은 매우 많은 비용이 필요하므로, 공간 질의를 효과적으로 처리하기 위하여 여과 단계와 정제단계를 사용한다. 여과 단계에서는 객체를 근사 데이터로 투영하여, 근사 데이터가 질의를 만족하는 객체를 후보 객체로 삼는다. 정제 단계에서는 여과 단계에서 선택된 후보 객체들이 실제로 질의 조건을 만족하는가를 확인하기 위해서 실제의 기하 데이터를 이용하여 질의 연산을 수행한다.[1,2]

2.1.1 여과 단계

객체를 실제 기하 데이터가 아닌, 객체와 유사한 형태의 근사 데이터로 투영한다. 근사 데이터는 실제 데이터보다 단순한 형태로 이루어져 있으며, 따라서 적은 연산과 빠른 시간 내에 질의 조건에 부합 여부를 판단할 수 있다. 여과 단계에서 후보 객체로 선택된 객체는 질의에 부합하는 가능성이 높다는 의미이며, 이 후보 객체들이 모두 질의 결과가 되는 것은 아니다. 따라서, 여과 단계의 목표는 근사 데이터를 가능한 실제 데이터와 유사하게 구성함으로써 여과 효율을 극대화시키는 것이다.[1,2]

2.1.2 정제 단계

정제 단계는 이전 단계에서 여과된 후보 객체에 대하여, 실제로 질의 영역과 겹치는 지를 판단하는 단계이다. 정제 단계는 공간 객체의 기하 데이터를 가지고 처리를 해야 하므로, 여과 단계에 비하여 매우 많은 비용이 요구된다.[1,2]

여과-정제 방법은 다양한 종류의 공간 질의 처리에 적용할 수 있으며, 구현이 용이하다는 장점 때문에 많은 연구가 이루어지고 있다.[7,8]

2.2 다단계 여과

다단계 여과 기법은 1차 여과 과정의 효율이 좋지 못하기 때문에, 여과율을 높이기 위한 보완 방법으로 연구되었다. 다단계 여과에서 사용할 수 있는 여러 가지 여과 방법이 연구되었는데, 크게 분류하여 근사 기법과, 분할 기법이 있다.[1,2,3,8]

2.2.1 근사 기법

근사 기법은 근사하는 방법에 따라, 보존적 근사(conservative approximations), 일반적 근사(generalizing approximations), 진보적 근사(progressive approximations)로 구분할 수 있다. 실제 데이터의 모든 영역이, 근사한 영역에 포함될 때 보존적 근사방법이라고 하며, 대표적인 예로 MBR이 있다. 일반적 근사는 공간 객체를 대강의 형태로 근사하는 방법으로, 공간 객체의 기하학적 특성을 잃어버려 원래 공간 객체와의 위상관계가 정확하지 않기 때문에 2차 여과 장치로는 적당하지 않다. 진보적 근사방법은 근사한 영역이, 실제 데이터의 영역에 포함되는 경우이며, 이 방법은 여과 과정의 연산이 복잡하다는 단점이 있다.[3,8]

2.2.2 분할 기법

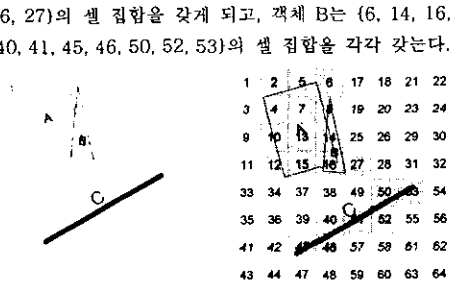
분할 기법은 공간 객체의 MBR을 분할하는 MBR 분할 기법과, 공간 객체를 여러 개로 분할하는 객체 분할 기법으로 나누어 진다. MBR 분할 기법은 전체 MBR을 여러 개의 작은 MBR로 분할하여, 객체와 유사한 모양의 작은 MBR을 여러 개 유지하는 방법이다. 이 방법은 여과효율을 높일 수 있으나, 공간 객체의 형태가 복잡해 질수록, MBR의 분할 횟수가 늘어나고, 많은 양의 저장 공간을 필요로 하며, 연산의 대상이 늘어난다는 단점이 있다. 다른 방법으로 객체 분할 기법은 복잡한 형태의 공간 객체를 단순한 여러 개의 객체로 분할시키는 방법으로, 근사의 정밀도는 높일 수 있지만[1,2], 공간 질의 처리시 연산이 다수의 분할된 공간객체에 대하여 이루어져야 한다는 것과, 원래 공간객체에 대한 정보가 요구될 때 공간객체를 재구성해야 하는 단점이 있다.[7,8]

3. 격자 분할 색인 기법(GDIM)

기존의 다단계 여과기법의 단점을 극복하고, 1단계의 여과 연산만으로 효율적인 여과율을 얻어내기 위하여 격자 분할 색인 기법(GDIM : Grid Decomposition Indexing Method)을 제안한다. 격자 분할 색인 기법은 전체 공간 영역을 격자 영역으로 분할한 후, 실제 데이터와 최대한 유사하게 근사 데이터를 투영할 수 있도록 하였다.

3.1 색인 구조

격자 분할 색인은 먼저 전체 공간 영역을 격자 형태로 분할을 한다. 이 전체 영역을 색인 영역(Index Region)이라고 하고, 색인 영역은 분할된 작은 사각형의 집합으로 이루어진다. 이 작은 사각형 영역 1개를 셀 영역(Cell Region)이라고 하고, 각 셀 영역은 색인 영역에서 유일한 주소를 갖도록, Z-Order[4]에 의해서 주소가 매겨진다. 하나의 셀 영역에 중첩되는 여러 객체가 올 수 있으므로, 각 셀 영역은 객체들의 리스트를 갖는다. 객체는 여러 개의 셀 영역들의 집합으로 표현될 수 있다. 예를 들어 [그림 1]의 경우에, 객체 A는 {2, 5, 6, 4, 7, 8, 10, 13, 14, 12, 15, 16, 27}의 셀 집합을 갖게 되고, 객체 B는 {6, 14, 16, 27}, 객체 C는 {40, 41, 45, 46, 50, 52, 53}의 셀 집합을 각각 갖는다.



(a)실제 데이터 (b)격자 근사

그림 1. 실제 데이터와 격자 근사

3.1.1 버켓

버켓은 객체들의 리스트를 저장하는 저장 장소가 된다. 하나의 셀 영역은 하나의 버켓을 지정하고 있으며, 버켓의 자료 구조는 [그림 2]와 같다.

```
Record Bucket
  Count : 현재 버켓 안에 저장된 객체 수를 저장
  Object[] : array [1..MAX_BUCKET_SIZE] of Object ID
End Record
```

그림 2. 버켓의 자료 구조

3.1.2 디렉토리

각각의 셀 영역은 셀 주소와 버켓 포인터를 갖는다. 디렉토리는 셀들의 집합 구조이다. 각각의 셀은 버켓에 대한 포인터를 가지고 있다. 셀과 디렉토리에 대한 자료 구조는 [그림 3]과 같다. 각각의 셀은 자신만의 유일한 번호를 가지고 있으므로, 셀 번호가 곧 Directory 내에서 셀을 지정하는 주소 역할을 하게 된다.

```
Record Cell
  BucketPointer : 이 셀이 지정하는 버켓의 포인터
End Record

Record Directory
  Directory[] : array [1..MAX_CELL_NUM] of Cell
End Record
```

그림 3. 셀과 디렉토리의 자료 구조

[그림 4]는 [그림 1(b)]에 대한 색인 구조를, 일부만 가시적으로 도시화한 것이다.

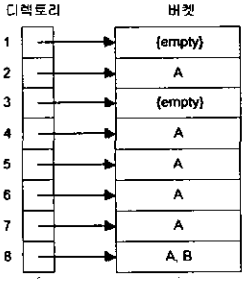


그림 4. 디렉토리와 버켓 구조

3.2 알고리즘

본 절에서는 격자 분할 색인 기법을 이용한 검색 연산과, 삽입 연산에 대해서 기술한다.

3.2.1 Tessellation

공간 객체를 입력으로 받아, 그 공간 객체가 차지하고 있는 셀의 리스트를 반환한다. 이 함수는 검색, 삽입, 삭제, 갱신 등 색인 내의 대부분의 연산에 이용된다.

<알고리즘 1. Tessellation of GDIM>

```
Input : 객체
Output : 셀 리스트
Procedure Tessellation
Begin
  Set CL to Cell List;
  For each All Cell
    If Current Cell is overlapped 객체
      CL := CL union Current Cell;
    End of If
```

End of For
return CL;
End

3.2.2 검색

질의 영역을 Tessellation 과정을 거쳐서, 셀 리스트를 얻어낸다. 각각의 셀에 대한 셀 번호는, 디렉토리 내에서 색인으로 사용되므로, 디렉토리와 셀 번호를 사용해, 버켓에 대한 포인터를 얻어낸다. 버켓 내에 저장된 모든 객체 들을 가져와서 임시 저장 장소에 저장하고, 이 과정은 셀 리스트 내의 모든 셀에 대하여 반복된다. 임시 저장 장소에 저장된 모든 객체들의 합집합이 검색 연산의 결과로 반환된다. [알고리즘 2]는 검색 연산의 알고리즘을 나타낸다.

<알고리즘 2. Search of GDIM>
Input : 질의 영역
Output : 객체 리스트
Procedure ***Search***
Begin
Set CL to Cell List;
Set OL, TempOL to Object List;
Set BP to Bucket Pointer;
/ 질의 영역의 셀 리스트를 얻어내는 과정 */*
CL := Tessellation(질의 영역);
/ 셀 리스트 내의 모든 셀에 대하여 반복연산 */*
For each Cell in CL
/ 버켓 포인터를 얻어낸다. */*
BP := bucket referenced by
Directory[Cell].BucketPointer;
/ 버켓 내의 모든 객체를 가져온다. */*
TempOL := Retrieve All Object in BP;
/ 임시 저장 장소의 객체들을 합집합 */*
OL := OL union TempOL;
End of For
/ 결과를 반환한다. */*
return OL;
End

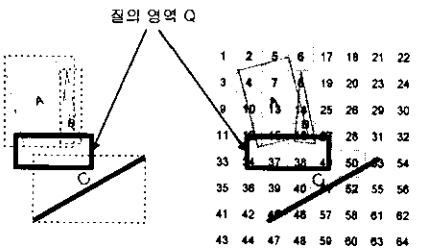
3.2.3 삽입

삽입하고자 하는 객체를 Tessellation 과정을 거쳐서, 셀 리스트를 얻어낸다. 위에서와 마찬가지의 방법으로 디렉토리와 셀 번호를 사용해, 버켓에 대한 포인터를 얻어낸다. 버켓 내에 삽입하고자 하는 객체의 ID를 삽입하고, 이 과정은 셀 리스트 내의 모든 셀에 대하여 반복된다. [알고리즘 3]은 삽입 연산의 알고리즘을 나타낸다.

<알고리즘 3. Insert of GDIM>
Input : 삽입 객체, 객체 ID
Procedure ***Insert***
Begin
Set CL to Cell List
Set BP to Bucket Pointer;
/ 삽입 객체에 대한 셀 리스트를 얻어내는 과정 */*
CL := Tessellation(삽입 객체);
/ 셀 리스트 내의 모든 셀에 대하여 반복 연산 */*
For each Cell in CL
/ 버켓 포인터를 얻어낸다. */*
BP := bucket referenced by
Directory[Cell].BucketPointer;
/ 버켓 내에 객체 ID를 삽입한다. */*
Insert 객체 ID into Bucket pointed by BP;
End of For
End

3.3 MBR 근사와, 격자 근사의 여과 효과 비교

다음 [그림 5]와 같이 동일 질의 영역 Q에 대하여, R-tree 계열의 (a)MBR 근사와, (b)격자 근사에 대하여 여과 효과를 비교한다.



(a) MBR 근사 (b) 격자 근사

그림 5. MBR 근사와 격자 근사에서의 질의 예

[그림 5]에서 굵은 선으로 주어진 부분이 질의 영역이다. (a)의 경우에는 객체 A, B, C의 MBR이 질의 영역 Q와 교차하므로, A, B, C 모두가 후보 객체로 선정된다. 반면, (b)의 경우는, 객체 A, B가 차지하는 셀이 질의 영역 Q와 교차하므로, A, B는 후보 객체로 선정되지만, 객체 C는 질의 영역과 교차되는 셀이 존재하지 않으므로, 객체 C는 후보 객체로 선택되지 않고, 여과된다.

4. 결론

본 논문에서는 격자 분할 영역을 이용한 공간 색인 기법을 제안하였다. 전체 공간 영역은 작은 셀 단위로 분할되며, 각각의 객체는 객체의 형태에 따라 셀 집합을 갖는다. 따라서, 객체를 근사하였을 경우 Dead Space를 크게 감소시켜 높은 여과율을 나타낸다. 또한, 한 번의 여과 과정만이 필요하기 때문에, 다단계 여과와 비교하여 수행 속도와 저장 공간 사용에 더 우수한 성능을 나타낸다.

참고문헌

[1] T. Brinkhoff, H.-P. Kriegel, B. Seeger, "Efficient Processing of Spatial Joins Using R-trees" Proc. Intl. Conf. on Management of Data, ACM SIGMOD Conference 1993, pp 237-246

[2] T. Brinkhoff, H.-P. Kriegel, R. Schneider, "Efficient Spatial Query Processing in Geographic Database Systems" IEEE Data Eng. Bulletin, Vol 16, No. 3, 1993, pp 10-15

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

[4] J.A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System" ACM SIGMOD Conference 1986, pp 326-336

[5] R.H. Guting, "An Introduction to Spatial Database Systems" VLDB Journal, Vol 3, No. 4, October 1994, pp 357-399

[6] H. Samet, "The quadtree and related hierarchical data structures" ACM Computing Surveys 16, 1984, pp 187-260

[7] 김성희, "선형 공간 객체의 효율적인 색인을 위한 기하학적 특성을 이용한 MBR 분할 기법," 공학석사학위논문, 1999

[8] 유치영, "효율적인 공간 질의 처리를 위한 그리드를 갖는 확장된 R-Tree," 공학석사학위논문, 1999