The Journal of GIS Association of Korea, Vol.8, No.1, pp.155-170, April 2000

효율적인 여과를 위한 그리드 필터를 갖는 R-Tree의 확장

김재홍*

Extended R-Tree with Grid Filter for Efficient Filtering

Jae-Hong Kim*

요 약

공간 인덱스인 R-Tree를 이용하여 조건에 만족하는 공간 객체를 찾기 위해서 MBR의 비교하는 여과 과정만으로는 부정확한 경우가 있으며 그러한 경우에는 해당 공간 객체를 읽어 조건과 직접 비교하는 과정, 즉 정제 과정이 필요하게 된다. 정제 과정을 수행하기 위해서는 디스크 입출력이 요구되고 고비용 공간 연산을 수행해야 되므로 이는 검색 비용이 커지는 요인이 된다. 그래서, 여과 과정 후의 후보 객체 수를 최소화하기 위한 2단계 여과 기법들이 연구되었지만, 여과 효율이 떨어지거나, 추가로 데이터를 유지해야하거나, 원래 객체의 정보를 잃어버려 객체들을 재구성해야 하는 문제점이 발생한다.

따라서, 본 논문에서는 이차 여과 과정에서 공간 객체의 존재 여부를 저장하는 진위 테이블인 그리드 필터를 이용하여 간단한 논리 연산만으로 공간객체를 검색할 수 있도록 하는 확장된 R-Tree를 제안한다. 그러므로, 그리드 필터를 가지는 확장된 R-Tree는 효율적인 이차 여과 과정을 수행하기 때문에 여과를 위한 연산 비용이 작고, 근사의 질이 높아 여과 효율이 우수하다.

색인어 : 공간 데이터베이스 시스템, R-Tree, 여과 과정, 그리드 필터, 공간 질의

Abstract : When we use R-Tree, a spatial index, to find objects matches some predicate, it often leads to an incorrect result to perform filtering step only with MBR. And, each candidates need to be inspected to conform if it really satisfies with given query, so called, 'refinement step'. In refinement step, we should perform disk I/O and expansive spatial operations which is the cause of increasing retrieval costs. Therefore, to minimize the number of candidates after filtering step, two-phase filtering methods were studied, but there was many problems such as inefficiency of filtering, maintenance of additional informations and reconstruction of data resulted from the loss of original information.

So, in this paper, I propose an Extended R-Tree which provides ability to retrieve spatial objects only with some simple logical operations using Grid Table, truth table storing the information about

the existence of spatial objects, in second filtering step. Consequently, this Extended R-Tree using Grid Filter has low cost of operation for filtering because of efficient second filtering step, and better filtering efficiency caused by high quality of approximation.

key word : spatial database system, R-Tree, filtering, grid filter, spatial query

1. 서 론

공간 데이터베이스 시스템은 기존의 속성 데이터 이외에 방대한 양의 다차원 공간 데이터에 대한 효율적 검색 기법이 요구되므로 B-Tree 계열의 1차원 데이터를 지원하는 색인 기법 뿐만 아니라 다차원 데이터에 대한 효율적이 색인 기법이 요구된다.

R-Tree는 현재 다차원 공간 데이터에 대한 가장 일반적으로 채택되고 있는 색인 구조이다. R-Tree의 질의 처리는 루트 노드에서부터 질의 영역과 겹치거나 포함되는 MBR(Minimum Boundary Rectangle)에 해당되는 노드만 선택하여 자식 노드로 내려간다. 모든 노드에 대하여 이 과정이 반복되어 리프 노드까지 도달하였을 때 결과에 포함될 후보객체가 선택된다. MBR은 공간 객체를 근사한 값이기 때문에 실제 공간 객체가 포함하지 않는 공간을 포함하고 있으며, 공간 질의 처리 수행 과정에서 선택된 공간 객체가 정제 과정 후에 false hit으로 판명되는 경우가 발생한다. False hit은 불필요한 디스크 입출력과 복잡한 연산을 요구하므로 질의 처리의 효율을 높이기 위해서는 false hit을 줄이는 것이 필수적이다[2,3]. False hit을 줄이기 위하여 정제(refinement) 과정 전의 여과(filter) 과정의 성능을 향상시키기 위한 연구들이 많이 진행되고 있다[4,5,6,7].

여과 과정의 성능을 향상시키기 위한 방법으로 공간 객체 근사의 질(quality of approximation)을 높이는 방법들이 제안되었는데 근사하는 방법은 보존적근사 (conservative approximation), 진보적근사(progressivea pproximation),일반적근사 (generalizing approximation)로 나눠진다[5]. 이중 일반적근사는 공간 객체의 위상 관계를 잃어버리기 때문에 공간 질의에서는 사용할 수 없다. 진보적근사 방법은 일반적 근사 방법과 동시에 사용될 때만 의미가 있다. 보존적근사 방법은 MBR을 포함해서 RMBB(rotated minimum bounding box), MBC(minimum bounding circle), MBE(minimum bounding ellipse) 등이 있으며 이들 방법은 간단하지만 여과 효율이 좋지 못하다. 근사의 질을 높이기 위한 방법으로 복잡한 공간 객체를 다수의 간단한 공간 객체로 관리하는 공간 객체 분할 기법들이 연구되어왔다[6,7]. 공간 객체 분할 기법(spatial object decomposition technique)은 여과 효율은 좋으나 원래 객체 전체에 대한 정보를 잃어버리므로 원래 공간 객체에 대한 반전 등이 발생할 때마다 원래 객체를 재구성해야 한다는 부담이 있다[8]. 이외에도 [8]이 제안한 기법과 같이 여과 효율을 높이기 위하여 기존의 근사 기법과 병행하여 사용할 수 있는 기법들이 제안되었으나 공간객체가 복잡해지면 유지해야하는 정보의 양이 커지고 연산의 복잡도가 증가한다는 문제점이 있다.

본 논문에서는 R-Tree를 사용하여 R-Tree의 장점인 공간 객체에 대한 관리가 간편하고 연산이 단순하다는 점을 그대로 유지하면서 MBR이 가지고 있는 낮은 여과 효율을 보완하기 위하여 기존의 여과 기법보다 여과를 위한 연산이 간단하면서 부가적인 정보의 양이 상대적으로 적은 그리드 필터(Grid)를 갖는 확장된 R-Tree를 제안한다.

본 논문에서 제안하는 그리드 필터는 MBR을

* 영동대학교 전자공학부 컴퓨터공학과 교수(Professor at Faculty of Informatin & Electronic Eng./Computer Eng., Youngdong University)

셀(cell)로 분할하고 각 셀과 공간 객체의 포함성을 진위 테이블(Truth Table)로 구성하며 R-Tree의 리프 노드에 MBR과 객체 ID와 더불어 위치하며 R-tree에 데이터가 삽입될 때 구성되며, 삭제될 때 지워진다. 그리드 필터는 R-Tree를 이용한 공간 질의 수행시에 MBR을 이용한 여과가 이루어지고 난 후에 적용되며 검색 영역과 검색 대상인 공간 객체의 그리드 필터와의 겹침 영역을 AND 연산함으로써 정제 단계에 입력될 후보 객체의 수를 줄이게 된다. 즉 겹치는 영역에서 그리드 필터의 셀에 1의 값이 존재하면 MBR에서 겹치는 영역이 실제 영역에서도 겹칠 수 있다는 것을 의미하므로 검색 대상 공간 객체는 후보 객체로 선택된다.

제안된 기법은 R-Tree의 장점을 그대로 유지하고 있으며 R-Tree의 단점인 여과 효율을 개선시켰고, R-Tree에서 부가적으로 가져야할 정보의 양도 기존의 기법보다 작다는 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 공간 질의 처리 효율을 높이기 위한 여러 가지 기법들에 대하여 분석하고, 3장에서는 본 논문에서 제안하는 확장된 R-Tree의 구조 및 그리드 필터의 생성 방법과 분할 조건을 설명하며, 4장에서는 확장된 R-Tree의 질의 처리 구조와 공간 데이터베이스 시스템에서 가장 많이 사용하는 점 질의(point query), 영역 질의(region query)에 대해서 그리드 필터를 이용한 질의 처리 방법을 설명한다. 끝으로 5장에서 본 논문의 결론을 맺는다.

2. 관련연구

본 장에서는 R-Tree의 공간 질의 처리 구조와 단점에 대하여 설명하고 R-Tree를 이용한 공간 질의 처리 효율 향상을 위한 다단계 처리(multi-step processing) 방법에 대하여 알아보고, 질의 처리 효율 향상을 위하여 제안된 필터링 기법들인 공간 객체 근사 기법(approximation)과 공간객체 분할 기법(decomposition)에 대하여 설명한다.

2.1 R-Tree의 공간 질의 처리

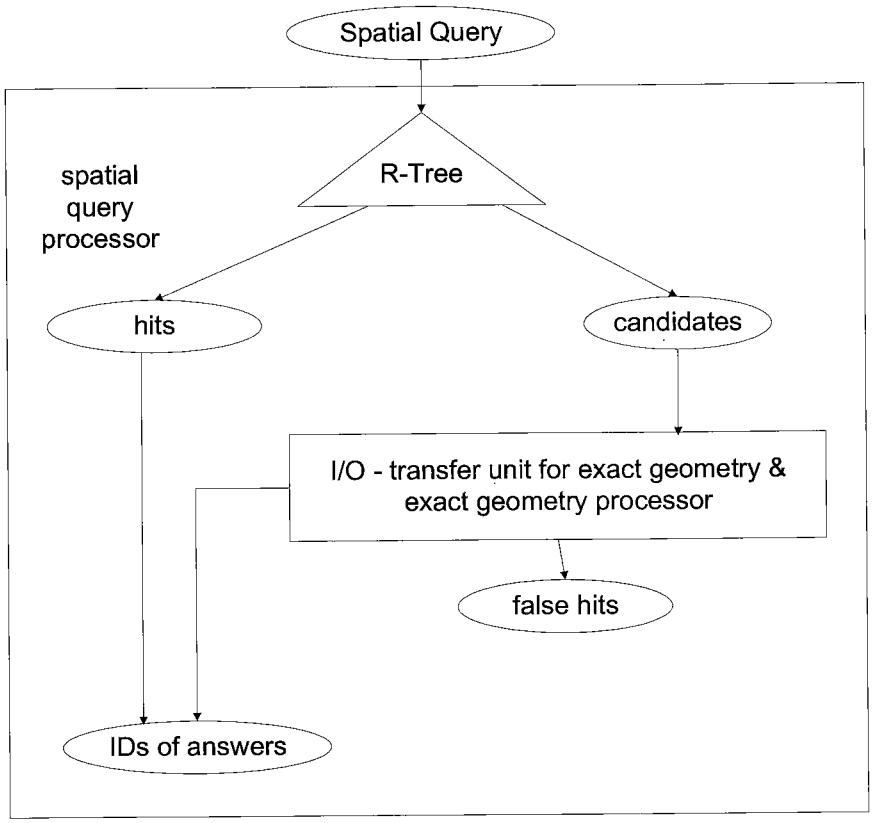
[9]에서 제안된 R-Tree는 공간 객체에 대한 관리가 간편하고 연산이 단순하다는 장점 때문에 많은 공간 데이터베이스 시스템에서 공간 데이터의 색인으로써 이용되고 있다[1]. 하지만 R-Tree는 색인의 키로 근사의 질이 높지 않은 MBR을 사용하기 때문에 공간질의 처리효율이 높지 않다는 단점이 있다.

R-Tree의 공간 질의 처리 과정은 다음과 같이 여과와 정제의 두 단계를 갖는다.

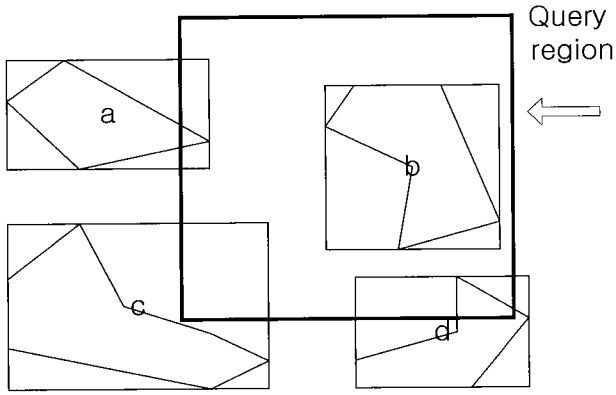
- 여과 단계 : 질의 사격형과 교차하는 모든 객체를 찾는다.
- 정제 단계 : 여과 단계의 결과인 후보 객체들에 대하여 디스크에 저장되어있는 공간 객체를 메모리로 읽어 질의에 만족하는 객체를 찾는다.

위와 같은 기본적인 구조를 바탕으로 한 R-Tree의 공간 질의 처리 과정은 [그림 2-1]과 같다. 공간 질의 처리는 [그림 2-1]에서처럼 R-Tree에 공간 질의가 입력된 후 R-Tree의 루트 노드에서 시작한다. 루트 노드에서 질의 영역과 겹치거나 포함되는 객체를 선택하여 그 객체의 자식 노드로 내려간다. 이 과정은 선택된 자식 노드(child node)가 리프 노드가 될 때까지 반복된다. 이 과정이 여과 과정이며 여과 과정을 통하여 질의에 만족되는 해가 구해질 수도 있지만, R-Tree에서 키 값으로 사용하는 MBR은 공간 객체를 그대로 표현한 것이 아니라 공간 객체를 근사한 것이기 때문에 여과 과정을 통하여 구하여진 대부분의 공간 객체들은 질의를 완전히 만족시키는 결과가 아니라 질의를 만족시킬 수도 있는 후보가 될 뿐이다.

따라서 질의를 만족시키는 해를 구하기 위해서는 여과 과정을 통하여 구해진 후보 객체(candidate object)들의 근사값이 아닌 실제 데이터를 디스크로부터 읽어서 후보 객체들이 정말 질의의 조건에 만족하는가 확인하여야만 한다. 예를 들어 [그림 2-2]와 같이 질의 영역이 주어



[그림 2-1] R-Tree의 질의 처리 구조



[그림 2-2] 선택 오류의 예

졌을 때 "질의 영역과 겹치는 모든 공간 객체를 찾아라" 라는 질의에 대하여 MBR만을 이용한 여과 과정을 거치게 되면 질의 결과에 대한 해에는 공간 객체 a, b, c, d 가 포함되게 되는데, c의 경우 MBR이 공간 객체를 그대로 표현한 값이 아니라 공간 객체를 근사한 근사치라는 특성 때문에 필연적으로 발생하는 dead space가 질의 영역과 겹치는 것이므로 정제 과정을 통하여 실제 데이터를 이용한 확인 과정을 거쳐 결과에서 제거해야 하며, a, d의 경우도 질의에 대한 후보해는 될 수 있지만 MBR만을 이용해서는 필요충분조건을 만족시킬 수가 없기 때문에 정제 과정을 거쳐야 한다. 결과적으로 [그림 2-2]와 같은 경우에는 c 한 개의 false hit이 존재하게 되며 근사정확도(accuracy of approximation) 가 높은 필터를 사용한다면 a와 d가 정제 과정을 거치는 것 또한 방지할 수 있다. 따라서 R-Tree를 이용한 공간 질의의 처리 효율의 향상을 위해서 여과 단계의 효율을 높여 후보 객체를 줄임으로써 복잡한 연산과 디스크 입출력을 요구하는 고비용의 정제 과정에서 처리해야 할 공간 객체의 양을 줄여야만 하며, 이를 해결하기 위한 여러 가지 방법들이 연구되어 왔다[10,11].

2.2 R-Tree의 여과 효율 향상을 위한 공간 질의의 다단계 처리

본 절에서는 여과와 정제의 두 단계로 구성된 R-Tree의 여과 효율 향상을 위한 공간 질의 처리를 위한 다단계 처리 방법에 대하여 논의한다.

2.1절에서 언급한 바와 같이 [9]에서 제안된 R-Tree는 2단계의 질의 처리 구조를 갖는다. 하지만 MBR을 이용한 질의 처리는 간단하고 빠른 검색 시간이 장점이지만 검색 효율이 낮은 단점이 있다. 이는 여과 과정의 효율이 좋지 못해 정제 단계에서 처리해야 하는 데이터의 양이 상대적으로 많기 때문이다.

공간 객체를 벡터 형식으로 표현하면 공간 객체는 이차원 좌표 값들의 연속으로 표현된다. 만약 공간 객체들에 대하여 영역 검색 질의를 수행한다면 가장 많은 비용을 요구하는 작업은 공간 객체를 디스크로부터 메모리로 읽어드리는 작업과 메모리로 읽어드린 공간 객체에 대하여 영역 검색 질의(region retrieval query)를 수행하는 작업이다. 이러한 작업들을 효율적으로 처리하기 위한 방법으로 제안된 것이 공간 질의의 다단계 처리 방법이다 [12]. 이 방법은 다음의 세가지 단계로 구성되어 있다.

- 첫번째 단계 : 공간 질의를 근사값을 가지고 처리하기 위하여 공간 객체의 실제 좌표값을 사용하지 않고 이를 대표하는 MBR과 같은 공간 객체 근사 방법을 사용한다. 이 단계에서는 해당 공간 질의의 해가 되는 공간 객체들과 해가 될 가능성이 있는 후보공간 객체가 모두 결과에 포함되어 있다. 이 단계는 R-Tree에서 지원되고 있다[12].

- 두번째 단계 : 첫번째 단계에서 결과로서 넘겨진 공간 객체들에 대하여 보다 정확한 질의 처리를 수행하기 위하여 좀더 정확한 공간 객체 근사 방법이 적용된다. 따라서 두번째 단계를 거치게 되면 첫번째 단계의 결과 질의의 해로써 적당하지 않는 객체들 일부가 해에서 삭제된다. 결국 첫번째와 두번째 단계를 거치면서 공간 객체를 디스크로부터 읽지 않고도 질의를 수행할 수 있거나 디스크로부터 읽는 공간 객체의 양을 줄일 수 있다.

- 세번째 단계 : 두번째 단계를 통하여 얻어진 질의의 후보 해들에 대하여, 디스크에 저장되어 있는 공간 객체를 메모리로 읽어들여 질의의 해로써 합당한지 여부를 검사한다. 이때 공간 객체들을 검사하는 시간은 공간 객체의 메모리에서의 표현 방식에 따라서 차이가 난다.

공간 객체의 다단계 처리 방식의 2번째 단계에서 사용할 수 있는 여러 가지 필터링 방법들이 연구되어왔는데 이를 크게 분류하면 공간 객체 근사기법과 분할기법으로 나눌 수 있다[3].

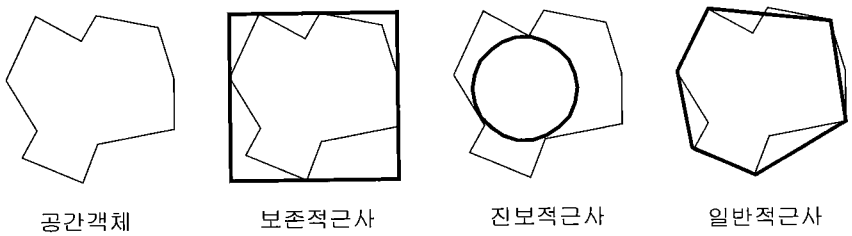
1) 근사 기법

공간 인덱스는 geometry key에 의하여 공간 객체들을 관리하는데 MBR이 가장 많이 쓰이는 geometry key이다. MBR을 사용하면 공간 객체의 모양이 유지되면서도 공간 객체을 네 개의 값으로 표현할 수가 있다. MBR의 또다른 장점으로는 point-in-MBR 같은 연산 수행 시 실행 시간이 매우 짧다는 것이다. 하지만 MBR을 이용하면 여과 속도가 빠른 반면, 그 결과가 정확하지 않다는 단점이 있다. 이는 MBR중의 상당 부분이 실제 공간 객체가 차지하는 영역과 다르기 때문이다. 따라서 MBR을 이용하는 공간 인덱스들은 MBR의 이러한 단점을 보완하기 위하여 다른 종류의 근사 기법들이 연구되었다[13].

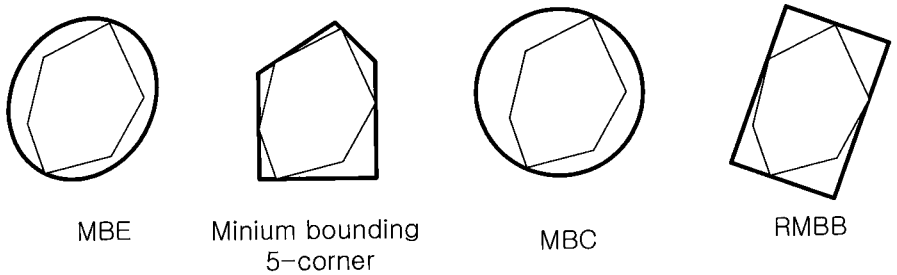
근사 기법은 방법의 차이에 따라 크게 [그림 2-3]과 같이 세가지 종류로 구분할 수 있다[5].

- 보존적 근사(conservative approximations)
- 진보적 근사(progressive approximations)
- 일반적 근사(generalizing approximations)

보존적 근사 방법은 만약 실제 공간 객체가 포함하는 모든 영역이 근사한 영역에 포함될 때 보존적근사 방법이라고 하며, 이에는 MBR을 포함해서 RMBB(rotated minimum bounding box), MBC(minimum bounding circle), MBE(minimum bounding ellipse) 등이 있으며 이들 방법은 간단



[그림 2-3] 근사기법의 분류



[그림 2-4] 기존의 근사 기법

하지만 MBR을 이용할 때 보다 효율이 낮다[5,13,14]. 일반적근사는 공간 객체의 기하학적 특성을 잃어버려 원래 공간 객체와의 위상 관계가 정확하지 않기 때문에 MBR을 보완하는 필터로 이용할 경우 아무런 이득이 없다. 진보적 근사 방법은 근사한 영역이 실제 공간 체계의 영역에 포함될 경우를 말하며 이방법의 연산도 여과 과정의 연산이 복잡한 단점이 있다. [그림 2-4]는 기존의 근사 기법들을 보여주고 있다.

2) 분할 기법

공간 객체 분할 기법은 공간 객체의 MBR을 분할하는 MBR 분할 기법과 공간 객체 자체를 여러 개로 분리하는 공간 객체 분할 기법으로 나눌 수 있다.

MBR 분할 기법은 공간 객체를 한번은 수직선, 한번은 수평선으로 재귀적으로 분할하며, 분할된 객체는 다시 분할된 객체의 MBR로서 표현된다. 이때 전체 객체 집합의 크기에 대한 객체의 MBR 크기와 객체의 MBR에 대한 객체의 영역이 차지하는 비율을 고려하여 공간 객체의 MBR 분할 여부를 결정한다. 공간 객체의 MBR이 분할되는 경우는 분할된 MBR의 크기가 분할 이전의 MBR 크기의 2-7보다 클 경우와 객체의 MBR에 대한 객체의 영역이 차지하는 비율이 75%이하일 경우이다[15]. MBR 분할 기법은 영역 포함 질의와 객체 삭제, 객체 선택 등의 검색 효율을 높일 수 있다[15]. 그러나 공간 객체가 복잡해질수록 MBR의 분할 횟수가 늘어나고 많은 양의 저장 공간을 필요로 한다는 단점이 있다.

공간 객체 분할 기법은 복잡한 모양의 공간 객체를 단순한 여러 개의 공간 객체로 분할시키는 방법이다. 분할한 객체의 모양에 따라 볼록 다각형으로의 분할, 사다리형으로의 분할, 삼각형으로의 분할, 서로 다른 두 개 이상의 타입으로의 분할 등의 방법으로 구분된다[3]. 공간 객체 분할 기법은 복잡한 공간 객체를 다수의 단순한 공간 객체로 분할하여 근사의 정밀도(precision of approximation)는 높일 수 있지만, 공간 질의 처리 시 연산이 다수의 분할된 공간 객체에 대하여 이루어져야 한다는 단점과 분할되기 이전의 공간 객체에 대한 정보가 요구될 때 공간 객체를 재구성해야 하는 단점이 있다.

3. 그리드를 갖는 확장된 R-Tree 제안

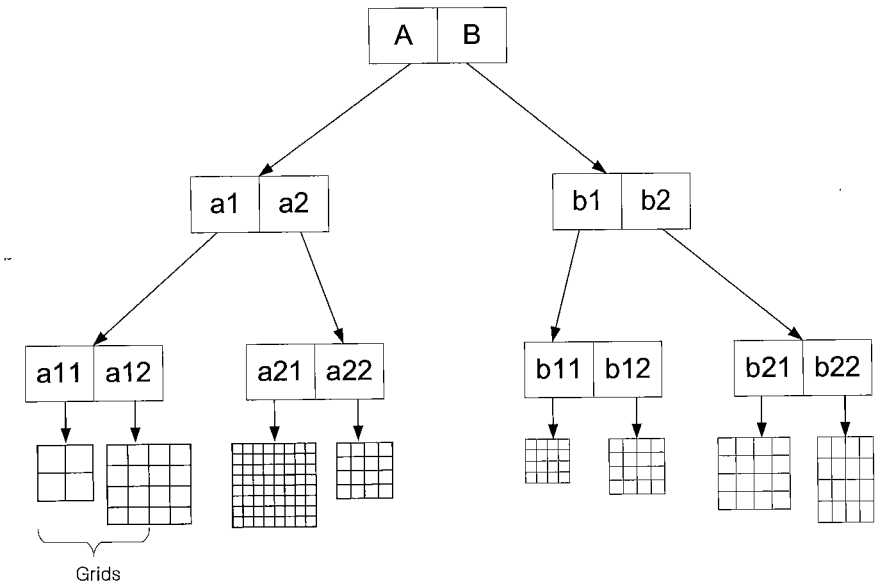
본 장에서는 R-Tree의 MBR 여과 과정의 결과인 후보 객체들의 수를 정제 과정 전에 줄이기 위한 그리드 필터를 제안하며, 제안된 기법을 R-Tree에 적용하기 위하여 그리드 필터를 갖는 확장된 R-Tree를 제안한다.

3.1 R-Tree의 여과 효율 개선 방안

R-Tree는 관리의 간편함과 연산의 단순함으로 많은 공간 데이터베이스 시스템에 도입되었으나, MBR을 키 값으로 사용함으로 인하여 질의 처리의 과정 중 여과 과정의 효율이 낮다는 단점을 가지고 있다. 이는 결국 전체 질의 처리 성능의 저하를 가져오게 된다. 본 논문에서는 이러한 R-Tree의 낮은 여과 성능을 향상시켜 고비용의 정제 과정에서 처리해야 하는 공간 객체의 양을 줄임으로써 전체적인 질의 처리 성능을 향상시키는 다단계 질의 처리 기법 중 두번째 단계의 여과 방법으로 사용될 여과 기법을 제안한다. 제안되는 기법은 R-Tree의 여과 성능을 향상시키면서도, 제안되는 기법으로 인하여 발생하는 추가적인 부하를 최소화시키는 것을 목적으로 한다. 이러한 목적을 만족시키기 위하여, 제안되는 기법은 R-Tree의 여과 성능은 향상시키면서도 여과를 위한 연산이 단순하고 추가적으로 요구되는 데이터의 양도 작아야 하고 관리하기가 간편해야 한다.

여과 효율 성능 향상을 위하여 R-Tree에 다단계 질의 처리 방법을 도입하는 방법으로 새로운

여과 기법을 R-Tree의 리프 노드에 추가하는 방법이 가장 적당하다. 이는 R-Tree의 내부 노드를 변경시키지 않기 때문에 기존 R-Tree의 장점인 관리의 용이함을 유지할 수 있고, 내부 노드에는 추가적인 데이터를 저장하지 않으므로 추가적으로 관리해야 하는 데이터의 양을 최소화시킬 수 있기 때문이다. 따라서 제안되는 기법은 이단계 여과를 위하여 필요한 데이터를 R-Tree의 리프 노드에 추가하게 된다. 이를 반영한 R-Tree는 [그림3-1]과 같은 구조를 갖는다.

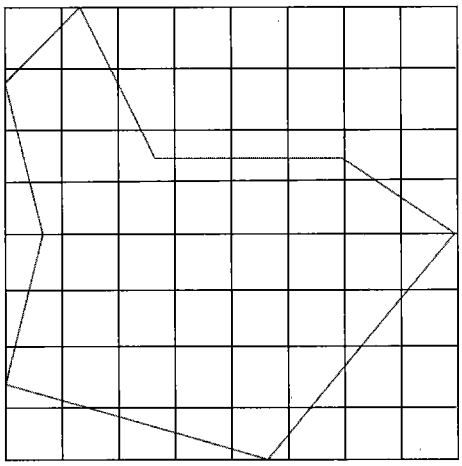


[그림 3-1] 확장된 R-tree의 구조

3.2 효율적인 여과를 위한 그리드 필터

본 절에서는 R-Tree의 질의 처리 효율 향상을 위하여 도입되는 다단계 질의 처리의 두 번째 단계에서 사용되는 그리드 필터를 제안한다. 제안하는 그리드 필터는 공간 색인을 위한 연산이 단순하며 여과 효율이 우수하고 추가적으로 가져야 하는 데이터의 양도 크지 않다는 장점을 갖는다.

본 논문에서 제안하는 그리드 필터는 객체의 MBR과 같은 크기의 사각형을 실험에 의해 정해진 인자에 의한 분할 개수만큼 가로/세로로 나눈 격자 모양을 갖는다. 그리드를 이루는 각 셀들은 셀이 차지하는 영역에 객체가 포함되는지의 여부를 0과 1의 비트 값으로 대표하여 저장하고 있다. 그리드를 나누는 기준인 분할 개수는 공간 객체의 MBR의 면적과 공간 객체 자체의 면적 비율에 의해 정해지게 된다. [그림 3-2]는 MBR에 포함된 공간객체를 표현한 것이며 [그림3-3]은 [그림3-2]의 공간 객체를 표현하는 그리드 필터이다. [그림 3-3]의 그리드 필터의 각 셀들의 값은 공간객체가 셀과 겹치면 1, 겹치지 않으면 0의 값을 갖는다.



[그림 3-2] 공간 객체

1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0
0	1	1	1	1	1	0	0

[그림 3-3] 그리드 필터

1) 그리드 분할 기준

그리드의 분할 기준은 공간 객체의 모양과 밀접한 관계가 있다. 만약 공간 객체의 모양이 복잡한 구조를 가지고 있다면, 그리드는 보다 많은 셀들로 분할되어야 일정 수준의 정밀도를 유지할 것이고, 만약 공간 객체의 모양이 단순하다면 그리드는 적은 수의 셀들로 분할되어도 일정 수준의 정밀도를 유지할 수 있다.

본 논문에서는 그리드의 분할 기준을 결정하기 위하여 공간 객체의 면적과 MBR의 면적의 비율 즉, 공간 객체가 공간 객체를 완전히 포함하는 MBR에서 차지하는 면적의 비율과 그리드를 구성하는 총 셀의 개수에서 값이 1인 셀의 비율을 이용한다. 여기서 MBR의 면적 대 공간 객체 면적의 비율은 MBR이 공간 객체의 실제 모습을 얼마나 정확하게 표현하고 있는지를 의미하며 이는 MBR의 정확도 즉, AOM(Accuracy of MBR)이라고 할 수 있고, 그리드를 구성하는 셀들에서 값이 1인 셀들이 차지하는 비율은 그리드가 공간 객체의 실제 모습을 얼마나 정확하게 표현하고 있는지를 의미한다. 이는 그리드의 정확도, 즉 AOG (Accuracy of the gird)라 할 수 있다. AOG가 AOM에 근접할수록 그리드의 정확도가 높아짐을 의미하며, AOM/AOG 비율이 1에 가까울수록 그리드의 정확도는 높아지게 된다. 하지만 AOM/AOG 비율이 1에 근접하지 않더라도 어느 정도의 분할 이후에는 검색 효율이 더 이상 좋아지지 않는 AOM/AOG 비율의 임계값을 찾을 수 있을 것이다. 본 논문에서는 그리드의 분할 기준을 결정하기 위한 AOM/AOG 비율의 임계 값을 실험을 통하여 구하고자 한다.

2) 그리드의 생성

본 논문에서 제안하는 그리드는 공간 객체의 MBR과 AOM/AOG 비율을 이용하여 생성된다. MBR은 그리드를 구성하는 틀을 만들기 위하여 이용되며 AOM/AOG 비율은 틀을 채울 셀의 개수를 결정하는데 이용된다. 그리드의 분할은 실험에 의하여 구해진 AOM/AOG 비율의 임계값에 의하여 분할 개수가 정해지면 그리드의 각 셀은 분할 개수만큼 분할되며 이때 그리드의 마지막 가로 세로 마지막 셀들의 크기를 제외한 모든 셀의 크기는 동일하게 분할된다. 동일하게 분할하는 이유는 각 셀이 포함하는 영역을 직관적으로 알 수 있기 때

문에 셀의 위치 정보를 저장할 필요가 없어 저장 공간을 줄일 수 있기 때문이다. 그리드의 생성 방법은 [알고리즘 3-1]과 같다.

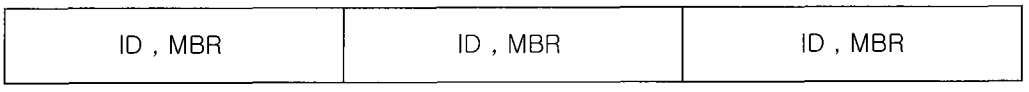
[알고리즘 3-1]은 입력 인자로 공간 객체, 공간 객체의 MBR , 실험에 의해 정해진 최적의 AOM/AOG 비율의 임계값을 받는다. 그리드의 생성은 먼저 그리드의 분할 개수를 결정하기 위하여 AOM을 구하고, MBR을 가로세로로 한번씩 반으로 나누어, 나누어진 셀 내에 공간객체가 포함되는지 검사한 후 그리드의 AOG를 구하여 만약 AOM/AOG 비율이 실험에 의해 정해진 AOM/AOG 비율보다 큰 값을 갖으면 분할을 중지하고 그렇지 않으면 AOM/AOG 비율이 실험에 의해 정해진 AOM/AOG 비 보다 큰 값을 갖을 때까지 분할을 계속한다.

[알고리즘 3-1] 공간 객체에 대한 그리드 생성 알고리즘

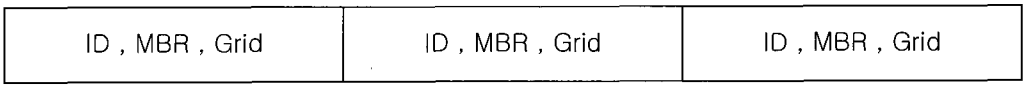
```
Algorithm : BuildGrid
INPUT: 공간 객체, 공간 객체의 MBR, AOM/AOG 비
OUTPUT: 그리드

BuildGrid( 공간 객체 g, 공간 객체의 MBR m , AOM/AOG 비 r ){
Let AOM = GetAOM( g , m );
WHILE( AOM/AOG > r )
{
/* 그리드를 분할 한다. */
DecomposeGrid( Grid , m );
/* 그리드의 각 셀들의 값을 이용하여 AOG를 구한다. */
AOG = ComputeAOG( Grid );
}
}
```

3.3 그리드 필터를 갖는 확장된 R-Tree의 제안

본 절에서는 R-Tree의 질의 처리 효율 향상을 위하여 본 논문에서 제안한 그리드 필터를 R-Tree의 2차 여과기로 갖는 그리드 필터를 갖는 확장된 R-Tree를 제안한다. 본 절에서 제안하는 확장된 R-Tree는 [그림 3-1]에서 살펴본 것처럼 R-Tree의 리프 노드에 그리드 필터에 대한 정보를 포함하고 있다. 확장된 R-Tree에서 리프 노드의 엔트리는 < id , mbr , grid >의 구조를 갖는다. [그림 3-5]는 확장된 R-Tree의 리프 노드를 나타내고 있는 것이며, [그림 3-4]는 내부 노드를 나타내고 있는 것이다. 그림에서 ID는 공간 데이터 베이스에서 공간 객체를 유일하게 식별할 수 있는 식별자를 말하며 MBR은 공간 객



[그림 3-4] 확장된 R-Tree의 내부 노드



[그림 3-5] 확장된 R-Tree의 리프 노드

체를 완전히 포함하는 사각형이고 Gird는 그리드를 가리키는 포인터이다.

[9]에서는 R-Tree를 관리하기 위한 연산들을 제시하고 있다. 이 중 Insertion과 Deletion을 제외한 다음과 같은 알고리즘은 [9]에서 제안된 알고리즘과 같아 별도로 기술하지 않는다.

- CreateTree : 새로운 트리를 생성하고 초기화 한다.
- DestroyTree : 트리의 모든 노드를 삭제하고 트리를 없앤다.
- SplitNode : 노드의 오버플로우(overflow)가 발생할 경우 노드를 분할한다.
- AdjustTree : 리프 노드 L에서 루트 노드까지의 경계 사각형 값을 갱신한다.
- Choose : 새로운 색인 엔트리 E가 위치할 리프 노드를 선택한다.
- Find : 루트 노드가 T인 R-Tree에서, 인덱스 엔트리 E를 포함하는 리프 노드를 찾는다.
- CondenseTree : 노드 삭제 시 엔트리를 재배치하고 경계 사각형 값을 갱신한다.

(1) 생성 알고리즘

그리드 필터는 R-Tree에 새로운 객체에 대한 인덱스를 구축하는 시점에 생성된다. 즉 R-Tree에 새로운 키가 입력될 때 생성되어 R-Tree의 리프 노드에 위치하게 된다. 확장된 R-Tree의 Insertion 알고리즘은 [알고리즘 3-2]와 같다.

[알고리즘 3-2] 공간 객체의 엔트리 추가 알고리즘

```
Algorithm : Insertion
INPUT : MBR , oid

Insertion(MBR , oid)
{
Entry.oid = oid;
/* 객체의 MBR을 구함 */
Entry.MBR = MakeMBR(oid)
/* 공간 객체에 대한 그리드를 생성한다. */
Entry.Grid = BuildGrid(Entry.oid , Entry.MBR);
/* 새로운 엔트리가 위치하게 될 리프 노드를 선택한다. */
Let L = Choose ();
IF( L에 E를 추가할 공간이 있다면 ) Install E;
ELSE SplitNode();
AdjustTree();
}
```

(2) 삭제 알고리즘

그리드 필터는 확장된 R-Tree의 리프 노드에 위치하므로 엔트리가 삭제될 때 같이 삭제된다. 확장된 R-Tree의 삭제 알고리즘은 [알고리즘 3-3]과 같다.

[알고리즘 3-3] 공간 객체의 엔트리 삭제 알고리즘

```
Algorithm : Deletion
INPUT : OID

Deletion( OID )
{
/* OID를 포함하는 리프 노드를 찾는다 */
Let n = FindNode( OID );
IF( n == NULL ) stop;
ELSE
/* 선택된 리프 노드를 제거하고 그리드를 제거한다 */
Remove( E.oid , E.MBR , E.Grid );
CondenseTree( n );
}
```

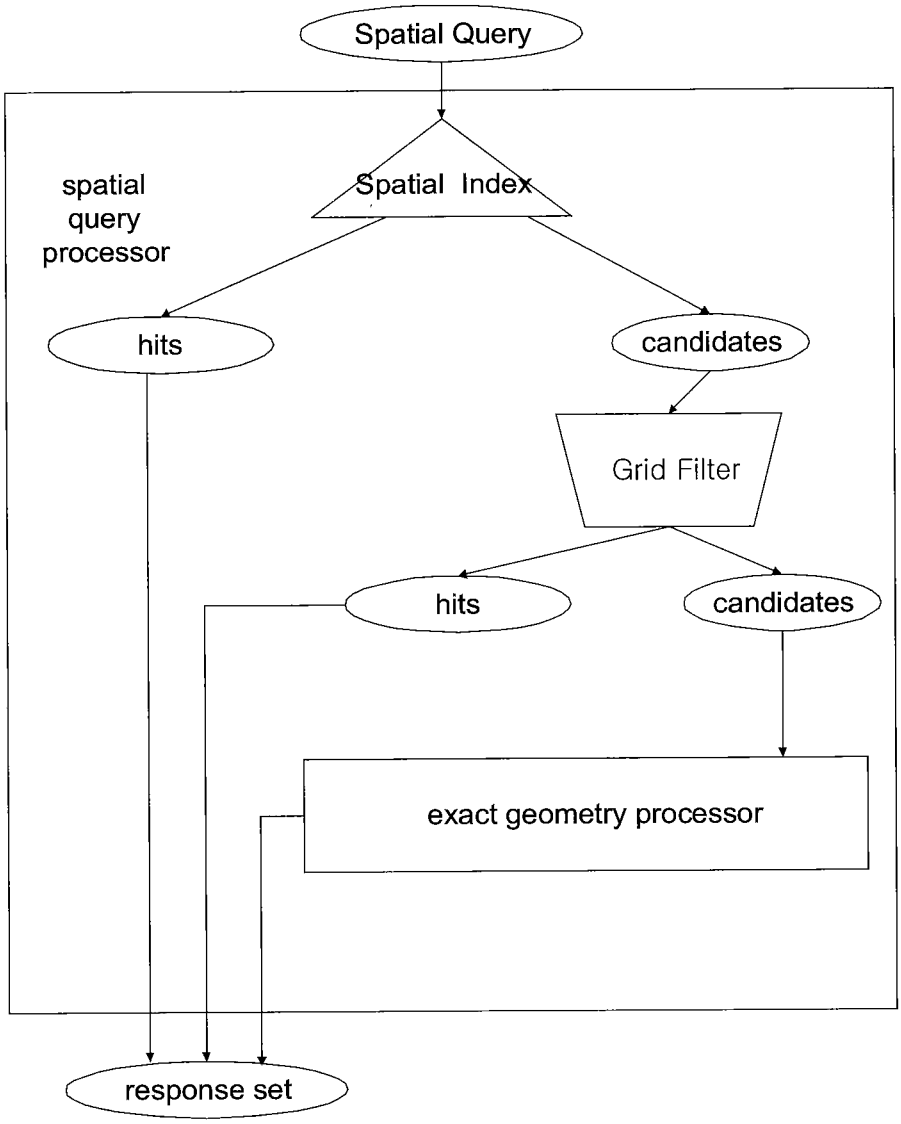
4. 확장된 R-Tree를 이용한 공간 질의 처리

본 장에서는 확장된 R-Tree에서의 질의 처리 구

조를 설명하고, 공간 질의 처리를 위해 선행되어야 하는 공통 영역에 포함되는 셀들을 구하는 방법에 대하여 기술하고, 공간 데이터베이스 시스템에서 빈번히 사용되는 점 질의와 영역 질의에 대하여 제안된 기법의 적용 방법을 설명한다.

4.1 확장된 R-Tree에서의 공간 질의 처리 구조

본 논문에서 제안하는 확장된 R-Tree를 이용한 공간 연산의 처리는 [그림 4-1]과 같은 구조를 갖는다. 그림에서 보듯이 확장된 R-Tree에서는 공간 질의의 처리시에 먼저 MBR을 이용하



[그림 4-1] 확장된 R-Tree에서의 질의 처리 구조

여 첫번째 여과 과정을 거치게 된다. 만약 이때 후보 객체가 존재하지 않는다면 질의 처리 과정은 끝나게 되고 후보 객체가 존재한다면 이 후보 객체들은 그리드 필터로 넘겨지게 된다. 그리드 필터를 이용한 질의 처리는 MBR 검색으로 선택된 후보 객체들에 대하여 그리드 필터를 이용하여 주어진 질의에 대한 각각의 알고리즘을 적용하여 여과 과정을 수행하게 되고 결과로써 hit가 나오면 그 OID를 반환하고, 역시 후보 객체가 나오면 이들을 exact geometry processor에 넘겨 처리하게 된다. 따라서 확장된 R-Tree에서는 두 번에 걸친 여과 과정을 통하여 처리 비용이 많이 드는 exact geometry processor에서의 연산을 줄이게 된다.

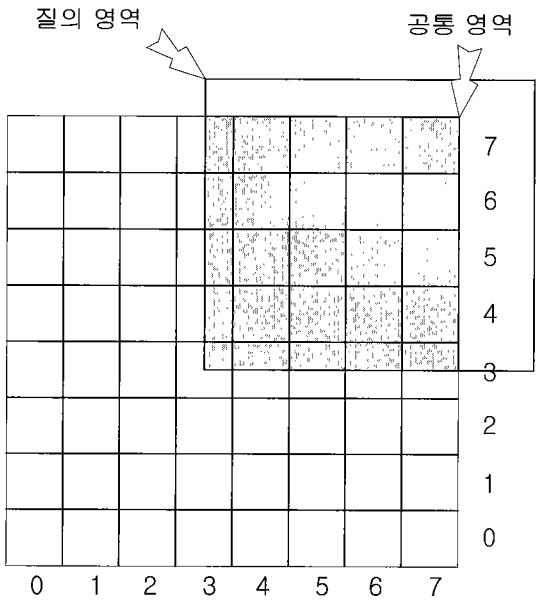
4.2 확장된 R-Tree에서의 공간 질의 처리 방법

본 절에서는 공간데이터베이스 시스템에서 가장 많이 사용하는 공간 질의인 점 질의, 영역 질의에 대하여 제안된 기법을 적용한 알고리즘을 제시한다. 본 논문에서 고려하는 점 질의, 영역 질의에 대한 정의는 다음과 같다[3].

[정의 4-1] 점 질의 : Given a point P $\in$ E2, find all Spatial Objects in the database where P $\in$ Spatial Object

영역 질의 : Given a rectilinear window w $\subseteq$ E2, find all Spatial objects in the database where w $\cap$ Spatial Object $\neq \Phi$

점 질의는 사용자가 화면에 점을 찍었을 때 공간 데이터베이스 내의 공간 객체 중 사용자가 입력한 점을 포함하는 모든 공간 객체를 구하는 질의이다. 영역 질의는 사용자가 화면에 직사각형으로 된 일정 영역을 선택했을 때 공간 데이터베이스 내의 공간 객체 중 사용자가 선택한 영역과 공통 영역을 갖는 공간 객체를 구하는



[그림 4-2] 그리드

질의이다.

1) 공통 영역에 포함되는 셀 검색

그리드 필터를 이용한 공간 질의 처리를 위해서는 먼저 주어진 질의 영역과 그리드 필터가 겹쳐지는 부분을 찾아내어 겹치는 부분에 어떤 셀들이 포함되는지를 구해야 한다. 그리드 필터는 분할 개수에 따라 균등하게 분할되므로 공간 객체의 MBR과 그리드의 분할 개수를 이용하여 주어진 질의 영역과 겹치는 셀들을 구할 수 있다.

[그림 4-2]와 같은 그리드를 갖는 공간 객체의 분할 개수가 8이고 MBR이 (1000, 1100, 3000, 3100)이고 질의 영역의 MBR이 (1900, 2000, 3100, 3200)이라면 공간 객체의 MBR과 질의 영역의 MBR이 겹치는 영역의 MBR은 (1900, 2000, 3000, 3100)이 된다. 이때 공통 영역에 해당하는 셀들을 구하는 방법은 다음과 같다.

셀의 너비 = (3000 - 1000) / 8 = 250
셀의 높이 = (3100 - 1100) / 8 = 250
시작 셀의 LOW = (1900 - 1000) / 250 = 3.6
시작 셀의 COLUMN = (2000 - 1100) / 250 = 3.6
마지막 셀의 LOW = (3000 - 1000) / 250 = 8
마지막 셀의 COLUMN = (3100 - 1100) / 250 = 8
∴ 공통 영역에 포함되는 셀 = 셀3:3 ~ 셀7:7

위의 계산에서 마지막 셀은 다른 셀들과 너비와 높이가 틀릴 수가 있기 때문에 공통 영역에 포함되는 셀을 계산할 때 셀의 분할 개수보다 큰 수가 나오면 이는 마지막 셀을 의미하는 것이다.

2) 질의 처리 알고리즘

[알고리즘 4-1]은 점 질의를 처리하기 위하여 함수 GetLowCellPosition과 GetColumnCellPosition를 이용하여 점과 겹치는 셀의 위치를 구한 후, 셀의 값이 1이면 공간객체의 OID를 반환하고 그렇지 않으면 MBR 여과에서 선택된 후보객체가 false hit임을 반환한다.

[알고리즘 4-1] 점 질의에 대한 그리드 필터 검색 알고리즘

```
Algorithm PointQuery
INPUT : 점의 좌표
OUTPUT : 공간 객체의 OID

PointQuery ( POINT p , 공간 객체의 MBR , GRID)
{
/* 주어진 점이 포함되는 셀의 가로 위치를 구한다. */
Let L = GetLowCellPosition( MBR , p );
/* 주어진 점이 포함되는 셀의 세로 위치를 구한다. */
Let C = GetColumnCellPosition( MBR , p);
/* 해당 셀에 공간 객체가 포함되어 있는지를 검사한다. */
IF GIRD[L][C].bit == 1
RETURN OID;
ELSE RETURN false hit
}
```

[알고리즘 4-2]는 영역 질의를 처리하기 위하여 함수 GetCellsInMBR을 이용하여 질의 영역과 공간 객체의 MBR의 공통 영역을 구한 후, 공통 영역에 포함되는 시작 셀과 마지막 셀을 구하여 선택된 셀들에 대하여 1의 값을 갖는 셀의 존재 유무를 검사한다. 검사 도중 1의 값을 갖는 셀이 발견되면, 이후의 셀에 대한 검사는 하지 않고 공간 객체의 OID를 반환한다. 그렇지 않으면 MBR 여과에서 선택된 후보 객체가 false hit임을 반환한다.

[알고리즘 4-2] 영역 질의에 대한 그리드 필터 검색 알고리즘

```
Algorithm RegionQuery
INPUT : 질의 영역의 MBR
OUTPUT : 공간 객체의 OID

RegionQuery ( 질의 영역의 MBR r , 공간 객체의 MBR o , GRID )
{
/* 질의 영역과 공간 객체의 MBR의 공통 영역을 구한다. */
Let C = GetCommonMBR( r , o );
/* 주어진 영역에 포함되는 셀들의 시작과 마지막 위치를 구한다 */
Let pos = GetCellsInMBR( r , o );
/* 위에서 구해진 모든 시작과 마지막 ROW에 대하여 수행 */
WHILE pos내의 모든 ROW
{
/* 위에서 구해진 모든 시작과 마지막 COLUMN에 대하여 수행 */
WHILE pos내의 모든 COLUMN
{
IF GRID[x][y].bit == 1
RETURN OID
}
}
}
```

5. 결 론

R-Tree는 관리가 간단하고 연산이 쉽다는 장점들 때문에 기존의 공간 데이터베이스 시스템에서 공간 데이터에 대한 인덱스로써 가장 많이 사용되고 있다. 그러나 R-Tree는 기본적으로 MBR이라는 근사도가 낮은 키를 사용하고 있어 공간 질의 처리 시에 효율이 좋지 못한 문제가 있다. 이러한 단점을 보완하기 위하여 MBR 여과와 더불어 다양한 종류의 여과 기법들이 연구되어 왔다. 하지만 이들은 관리가 어렵거나, 연산이 복잡하거나, 여과효율이 낮다는 단점들이 존재한다.

본 논문에서는 R-Tree의 단점인 낮은 여과 효율을 보완하기 위하여 그리드 필터를 갖는 확장된 R-Tree를 제안하였다. 기존의 R-Tree는 여과와 정제의 2 단계 질의 처리 과정을 가지고 있지만, 확장된 R-Tree는 MBR을 이용한 여과 이후에 그리드 필터를 이용한 여과를 도입하였다. 그리드 필터는 비트 단위의 셀들로 구성되어있어 저장 공간이 적게 필요하고, 질의 처리 시 검색 영역과 공간 객체가 공통으로 갖는 영역을 대표하는 셀들만을 비교하는 단순한 연산들로 이루어지므로 여과를 수행하기 위한 비용이 적다.

제안된 기법은 웹 GIS 시스템과 같이 사용자로부터의 점 질의, 영역 질의가 빈번한 시스템에서 좋은 성능을 나타낼 것이다. 향후 연구 과제로는 구현을 통한 AOM/AOG 비율의 임계치 측정과 기존의 기법들과의 응답시간, false hit율 비교를 하는 성능 평가가 필요하며, 그리드 필터의 생성시 소요 시간 단축을 위한 알고리즘 개발에 대한 연구가 필요하다.

참 고 문 헌

[1] T. Brinkhoff, H. Kriegel, and B. Seeger, "Efficient Processing of Spatial Joins Using R-Trees," Proc. ACM SIGMOD Int. Conf. On Management of Data, pp 237-246, 1993.

[2] H. Kriegel, T. Brinkhoff, and R. Schneider, "Efficient Spatial Query Processing in Geographic Database System," Data Engineering Bulletin, vol.16, no.3, pp.10-15, 1993.

[3] T. Brinkhoff, et. al., "A Storage and Access Architecture of Efficient Query Processing in Spatial Database Systems," Proc. 3rd Int. Symp. On Large Spatial Databases, Lecture Notes in computer Science, vol.692,

Springer-Verlag, pp.357-376, 1993.

[4] J. Orenstein, "Redundancy in Spatial Databases," Proc. ACM SIGMOD Int. Conf. On Management of Data, pp.294-305, 1989.

[5] R. Schneider, "A Storage and Access Structure for Spatial Database systems," Ph.D. thesis, Institute for computer Science, University of Munich. 1992.

[6] H. Kriegel, H. Horn and M. Schiwietz, " The Performance of Object Decomposition Techniques for Spatial Query Processing," Data structures and Efficient Algorithms, Lecture Note in Computer Science, vol.594, Springer-Verlag, pp.104-123, 1992.

[7] 김성희, "선형 공간 객체의 효율적인 색인을 위한 기하학적 특성을 이용한 MBR분할 기법," 석사학위논문, 인하대학교, 1999.

[8] 김영란, "False hit을 개선을 위한 분할된 MBR을 갖는 R-Tree의 확장," 석사학위논문, 인하대, 1998.

[9] A. Guttman, "R-Trees : An Dynamic Index Structure for Spatial Searching," Proc. Of the ACM SIGMOD Conf. on Management of Data, pp.47-57, 1984.

[10] H. Kriegel, R. Schneider, and T. Brinkhoff, "Potentials for Improveing Query Processing in Spatial Database Systems," Proceedings of 9th Journals Bases de Donnes Avance, 1993.

[11] T. Brinkhoff, H Kriegel, and B. Seeger, "Parallel Processing of Spatial Joins Using R-Trees," Proceedings of 12th International Conference on Data Engineering(ICDE'96), 1996.

[12] T. Brinkhoff, et. al., "Multi-Step Processing of Spatial Joins," SIGMOD Conference, ACM, pp.197-208, 1994.

[13] T. Brinkhoff, H. Kriegel and R. Schneider, "comparison of Apporximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems," Proceedings Ninth International Conference on DATA ENGINEERING, pp.40-49, 1993.

[14] E. Welzl, "smallest Enclosing Disk (Balls and Ellipsoids)," Technical report B91-09, Freie University of Berlin, 1991.

[15] Y. Lee, et. al., "Controlled Decomposition Strategy for Complex Spatial Objects," 7th International Conference on Database and Expert System Applications, Lecture Notes in Computer Science, vol.1134, Springer-Verlag, pp.321-329, 1994.