□ 기술해설 □

지리 데이타베이스 시스템에서 2단계 질의 처리 기법

가톨릭대학교 황 병 연*

목 차

1. 서 론

기존의 전통적인 데이타베이스 관리 시스템은 지리 정보 시스템(geographic information system), 사무 정보 시스템, 컴퓨터 지원 설계(computer-aided design), 혹은 전문가 데이타베이스 시스템과 같은 복잡한 응용에 적합하지 않다. 전통적인 시스템의 접근 기법, 질의 언어, 데이타 모델 등은 정수나 실수, 혹은 스트링과 같은 단순한 데이타 타입만을 다루는 반면에 복잡한 응용에서는 다차원 공간상의 점, 선분, 사각형, 다각형 등의 공간 데이타를 다루는 것이 요구된다. 본 논문에서는 어떤 주어진 위치와 조건에 해당하는 데이타를 빨리 검색하기 위해서 요구되는 공간 질의 언어와 접근 기법을 기술하고, 공간 색인을 이용한 2단계 질의 처리 기법을 제안하였다.

지리 데이타베이스 시스템은 지도 형태로 데이타를 표현할 수 있어야 함은 물론이고 공간 데이타의 저장, 검색, 분석, 관리를 할 수 있어야 한다. 이런 시스템에서 데이타베이스는 2차원 지도 상에 공간 객체의 집합으로 구성된다. 주어진 지도에서 공간 객체들은 다른 객체들과 *교차(intersect)*하거나, 다른 객체를 *포함(cover)*하거나, 다른 객체와 *인접(adjacent)*할 수도 있다. 이와같은 객체간의 관계를 *공간 관계(spatial relationships)*라고 하며, 공간 관계는 속성에 기반을 두기보다는 객체 지향적이라고 할 수 있다. 교차와 같은 공간관계는 "종로와 교차하는 거리를 검색하라"와 같은 질의에 답을 주기 위해 사용된다.

지리 데이타베이스 시스템에서 적당한 공간 색인을 사용하지 않을 경우, 저장된 모든 객체가 질의 조건에 대해 검색되어지며 이것은 저장된 객체의 수가 증가함에 따라 상당한 성능의 저하를 초래한다. 따라서 효율적인 질의 처리를 하기 위해서는 공간 접근 기법이 무엇보다 요구된다. 공간 객체의 가장 단순한 종류인 다차원 점 데이타를 처리하기 위한 공간 접근 법으로는 [3,9, 16] 등이 있고, 선분이나 다각형 등의 공간 데이타를 처리하기 위한 공간 접근 기법으로는 [1, 4,11,17,18] 등이 있다.

최근에 공간 접근 기법에 기반을 둔 여러 질의 처리 기법이 제안되었다. 그러나, 그들 중 [10,14]에서는 *최근접(nearest)*, *최원격(furthest)*, *거리(distance)*와 같은 다양한 공간 연산자를 고려하지 않았고, [12]에서는 선분이나 다각형 등의 복잡한 객체를 다루지 않았다. 본 논문에서는 점,

*종신회원

선분, 다각형 등의 복잡한 객체를 다루고, 교차, 최근접, 최원격, 인접, 포함, 조인, 거리 등의 공간 연산자를 고려하였다.

지리 데이타베이스 시스템에서 도시, 길, 혹은 호수와 같은 객체들은 고정된 형태를 갖지 않기 때문에 비슷한 모양으로 변형시켜야 하며, 잘 알려진 방법으로는 영역 분할(region decomposition)과 최소 경계 사각형(minimum bounding rectangle: MBR)이 있다. 영역 분할은 quad-tree [14]와 같이 객체를 원하는 해상도의 래스터 사각형으로 분할한다. 한편, MBR은 어떤 객체를 둘러싸는 가장 작은 사각형이다. 영역 분할이 영상 처리에 적당한 반면 MBR은 공간 질의 처리에 널리 사용된다. MBR은 공간 조건에 만족하지 않는 객체들을 사전에 검사해서 제거함으로써 효율적인 질의 처리를 할 수 있도록 한다. 즉, 두 객체의 MBR이 서로 교차하지 않으면 실 객체도 교차하지 않는다. K차원의 MBR은 다음과 같이 정의된다.

$(I_0, I_1, I_2, \cdots, I_{k-1})$

여기서 I_i는 i번째 차원에 대한 객체의 시작과 끝을 나타내는 간격을 의미한다. 복잡한 공간 객체의 질의 처리는 2단계로 처리된다. 첫번째 단계인 *여과 단계(filter phase)*에서는 전체 객체들을 그들의 주어진 위치를 이용해서 후보 객체들의 집합으로 감소시키며, MBR 객체를 관리하는 공간 접근 기법에 기반을 둔다. 그러나, MBR 객체는 정확한 객체를 표현할 수 없고, 후보 객체들은 두번째 단계인 *정제 단계(refinement phase)*에서 재 검사되어진다. 정제 단계에서는 본래의 실 객체에 대해 계산 기하학에서 이용되는 복잡한 알고리즘을 적용해서 주어진 질의 조건에 만족하는 최종 객체를 검출한다.

본 논문의 구성은 다음과 같다. 2장에서는 공간 질의 언어와 공간 연산자에 대해서 기술하였고, 3장에서는 공간 접근 기법을 설명하였다. 4장에서는 2단계 공간 질의 처리 기법을 제시하였고, 끝으로 5장에서 결론을 제시하였다.

2. 공간 질의 언어

데이타베이스 질의어는 질의 형식을 간단히 하기 위해서 사용하기 쉽고 배우기 쉬워야 하며, 다양한 기능을 지원해야 한다. SQL[2], QUEL [6], QBE[19]와 같은 기존의 데이타베이스 언어는 공간 응용에서 요구되는 다양한 연산자들을 효율적으로 지원하지 못하므로 확장되어야 한다. GSQL[8]은 SQL의 기본 형태에 AT, ON 절과 WITHIN WINDOW, WITH 구를 첨가하여 공간 데이타를 다룰 수 있도록 확장한 질의어이다. GSQL의 형태는 다음과 같다.

```
SELECT  list_of_attributes
FROM    list_of_relations
AT      area_specification WITHIN WIN-
        DOW(query_window)
ON      list_of_picture WITH LEGEND
        list_of_legends
WHERE   qualification.
```

AT 절은 공간 질의의 대상이 되는 영역을 제한하는 역할을 하고, WITHIN WINDOW 구를 사용함으로써 질의 영역을 더욱 축소시킨다. ON 절은 AT 절에서 주어진 영역에 대해 공간 질의의 대상이 되는 공간 데이타의 릴레이션을 선언하는 곳이며, WITH 구는 레전드 및 중첩(overlay)의 연산을 할 수 있는 정보를 제공함으로써 처리 결과에 대한 데이타 표현성을 증가시킨다. WHERE 절은 비공간 질의에 대한 조건 뿐만 아니라 공간 질의 조건을 지원한다. 예를 들어 "서울에 있는 인터컨티넨탈 호텔의 반경 1Km 이내에 있는 모든 극장을 찾아라."와 같은 질의는 다음과 같이 표현된다.

```
SELECT  t.name
FROM    THEATRE t, HOTEL h,
        STREET s
AT      Seoul
ON      THEATRE_MAP WITH LE-
        GEND CHAR (t.name, s.name)
WHERE   h.name="Intercontinental" AND
        distance(h, t)≤1 km.
```

위 질의에서 비공간 연산자는 h.name="Inte-

표 1 GSQL에 의해 지원되는 공간 연산자

연 산 자	정 의
교차 (intersect)	객체 A와 객체 B가 임의의 점을 공유한다면 두 객체는 교차한다.
인접 (adjacent_to)	객체 A와 객체 B가 교차하고, 교차하는 모든 점이 두 객체의 경계점이라면 객체 A는 객체 B에 인접한다.
포함 (cover)	객체 B의 모든 점이 객체 A의 점이라면 객체 A는 객체 B를 포함한다.
조인 (join)	객체 A와 객체 B의 교차하는 모든 점이 극점이라면 객체 A는 객체 B를 조인한다.
거리 (distance)	객체 B의 모든 점이 객체 A의 경계로부터 X 단위만큼 확장함으로써 얻어진 가상의 객체라면 객체 B는 객체 A의 X 단위의 거리내에 존재한다.
최근접 (nearest)	모든 객체에 대해서 객체 B보다 객체 A에 더 가까운 다른 객체가 없을 때, 객체 B는 객체 A에 최근접한 객체이다.
최원격 (furthest)	모든 객체에 대해서 객체 B보다 객체 A로부터 더 먼 다른 객체가 없을 때, 객체 B는 객체 A로부터 최원격인 객체이다.

rcontinental"이고, 공간 연산자는 distance(h, t) ≤1 km이다. 위 질의에서 ON 절의 WITH LEGEND 구에 의해 공간 질의의 결과는 극장과 거리 이름을 표시한다. GSQL에 의해 지원되는 연산자는 표 1과 같다.

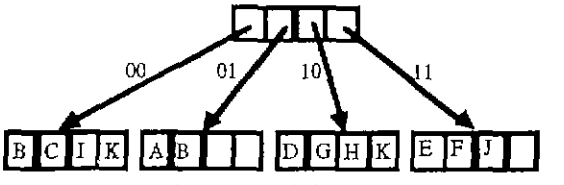

그림 1 BR 트리의 색인 구조

3. 공간 색인 구조

지리 데이타베이스 시스템에서 데이타베이스는 특정한 다차원 공간상의 객체들의 집합으로써 효율적인 질의 처리는 이와같은 객체들의 공간 색인을 어떻게 잘 지원하는가에 달려 있다. 본 장에서는 우리가 제안했던 공간 접근 기법인 BR 트리 [7]에 대해 간략히 기술한다.

BR 트리는 Z-변형(Z-transform) [13]에 기반을 둔 동적 공간 접근 기법이다. 리프 노드는 R 트리나 R^+ 트리와 같이 (I, t_i) 형태의 항목을 갖는다. 여기서 I는 데이타베이스에 저장된 객체를 둘러싸는 사각형 영역이고, t_i(tuple identifier)는 데이타베이스에 저장된 객체의 주소를 가리키는 지시자이다. 중간 노드나 루트 노드는 (cp) 형태의 항목을 갖는다. 여기서 cp는 자식 노드의 주소를 가리키는 지시자로서 BR 트리는 R 트리나 R^+ 트리에서 요구하는 자식 노드의 사각형 영역을 포함하는 최소 경계 사각형을 나타낼 필요가 없다. 이것은 BR 트리가 영역을 같은 크기의 두 영역으로 각 좌표축을 번갈아 가면서 분할함으로써 각각의 분할된 영역의 위치를 사전에 알 수 있기 때문이다. 또한, BR 트리는 하나의 객체가 여러 노드에 중복 저장되게 함으로써 R 트리에서 발생하는 겹치는 영역이 존재하지 않는다. 그림 1과 그림 2는 각각 BR 트리와 저장된 공간 객체들을 나타내고 있다.

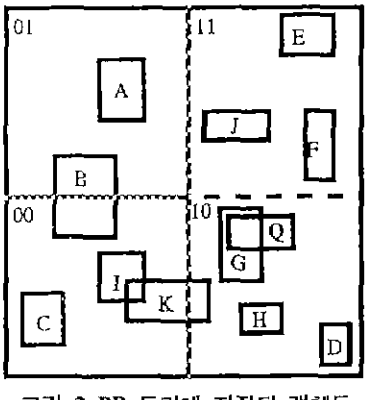

그림 2 BR 트리에 저장된 객체들

BR 트리의 검색 알고리즘은 질의 공간을 각 차원에 대해서 영역 비트를 구한 다음, 각 영역 비트의 맨 좌측 비트부터 우측으로 한 비트씩

옮겨가면서 트리의 루트부터 각각의 자식노드로 분할된 영역들을 찾아 밑으로 내려간다.

BR 트리의 삽입 알고리즘은 먼저 검색 알고리즘을 사용하여 객체가 삽입될 객체 블럭을 찾는다. 찾은 객체 블럭에 객체를 삽입할 때 범람이 발생하면 두개의 객체 블럭으로 나누고 나누어진 객체 블럭에 해당 객체를 각각 삽입한 후, 트리의 위로 올라가면서 루트 노드까지 계속 분할해 나간다.

BR 트리의 삭제 알고리즘은 먼저 검색 알고리즘을 사용하여 객체가 삭제될 객체 블럭을 찾는다. 찾은 객체 블럭에서 객체를 삭제한 후, 그 블럭이 어떤 수준 이하의 객체들을 포함할 경우 이웃한 다른 객체 블럭과 병합하며, 트리의 위로 올라가면서 루트 노드까지 계속 병합해 나간다.

4. 2단계 공간 질의 처리

본 장에서는 GSQL에 의해 지원되는 각 공간 연산자들을 구현하기 위한 알고리즘들을 기술한다. 각 알고리즘과 사용자 인터페이스는 X 윈도우 시스템의 Motif와 X library를 사용하여 구현하였다.

GSQL에 의해 지원되는 공간 연산자를 구현하기 위해 우리는 다음과 같은 방법을 사용하였다. 아래 질의 처리 알고리즘에서 *query_object*는 주어진 객체의 각 끝점의 집합이고 *list_of_picture*는 처리되어질 공간 객체의 지리적 엔티티 클래스이다. 여과 단계의 입력 매개 변수인 *rect_query_object*는 *query_object*의 최소 경계 사각형이며, 최소 경계 사각형은 *query_object*를 둘러싸는 가장 작은 사각형이다.

Query Processing(*query_object,list_of_picture*)

QP1. Collected_object_list=Collecting(*list_of_picture*);

QP2. Filtered_object_list=Filtering(rect_query_object, Collected_object_list);

QP3. Refined_object_list=Refining(query_object, Filtered_object_list);

QP4. Return(Refined_object_list)

첫번째 단계에서는 GSQL에서 표현되는 ON 절의 *list_of_picture*에 상응하는 공간 객체를 모아서 그 객체들에 대한 공간 색인 구조인 BR 트리를 생성한다. 두번째 단계에서는 검색 시간을 줄이기 위해 BR 트리 구조와 여과 알고리즘을 이용해서 검색 공간을 감소시킨다. 세번째 단계에서는 두번째 단계에서 여과된 객체들에 대한 공간 질의 연산을 수행하고, 끝으로 처리된 객체를 출력한다.

4.1 여과 단계

본 절에서는 공간 연산자들인 최근접, 최원격, 교차, 포함, 거리에 대한 여과 기법을 기술한다. 불규칙한 형태를 갖는 공간 객체를 처리하기 위해 최소 경계 사각형을 사용한다.

4.1.1 최근접 여과와 최원격 여과

그림 3은 최근접 여과와 최원격 여과에 대한 예제를 나타낸다.

Nearest Filter(P(X,Y))

Input: A coordinate value of a given point P

Output: A rectangular object existing in the nearest distance from the point P

NF1. Select a candidate object that has the nearest distance between X value of the point P and two X value(X_{mm} and X_{max}) of each object. Let *XN_object* be the candidate object. In 그림 3, *XN_object* is C.

NF2. Let *XN_length*(*XN_object*) be the furthest di-

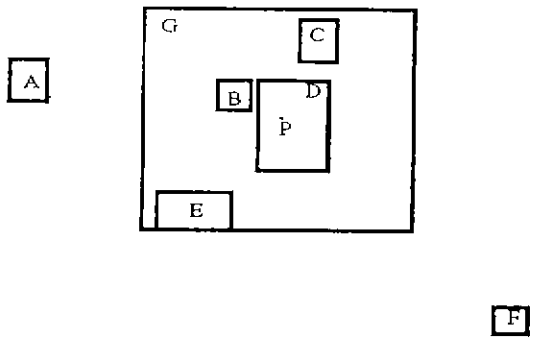

그림 3 최근접 및 최원격 여과

stance between the point P and *XN_object* and *XN_length*(O_i) be the nearest distance between the point P and an object *Oi*

NF3. In case that *XN_length*(O_i) is longer than *XN_length*(*XN_object*), if the object O_i does not cover the point P then deletes it, e.g. A and F in 그림 3, otherwise does not delete it, e.g. G in 그림 3.

NF4. Select a candidate object that has the nearest distance between Y value of the point P and two Y value(Y_{min} and Y_{max}) of each object. Let *YN_object* be the candidate object. In 그림 3, *YN_object* is B.

NF5. Let *YN_length*(*YN_object*) be the furthest distance between the point P and *YN_object* and *YN_length*(O_i) be the nearest distance between the point P and an object O_i.

NF6. In case that *YN_length*(O_i) is longer than *YN_length*(*YN_object*), if the object O_i does not cover the point P then deletes it, e.g. E in 그림 3, otherwise does not delete it, e.g. G in 그림 3.

NF7. Return the remaining objects, e.g. B, C, D, and G in 그림 3.

Furthest Filter(P(X,Y))

Input: A coordinate value of a given point P

Output: A rectangular object existing in the furthest distance from the point P

FF1. Select a candidate object that has the furthest distance between X value of the point P and two X value(X_{min} and X_{max}) of each object. Let *XF_object* be the candidate object. In 그림 3, *XF_object* is A.

FF2. Let *XF_length*(*XF_object*) be the nearest distance between the point P and *XF_object* and *XF_length*(O_i) be the furthest distance between the point P and an object O_i.

FF3. If *XF_length*(O_i) is shorter than *XF_length* (*XF_object*) then deletes it, e.g. B, C, D, E, and G in 그림 3, otherwise does not delete it, e.g. F in 그림 3.

FF4. Select a candidate object that has the furthest distance between Y value of the point P and two Y value(Y_{min} and Y_{max}) of each object. Let *YF_object* be the candidate object. In 그림 3, *YF_object* is F.

FF5. Let *YF_length*(*YF_object*) be the nearest distance between the point P and *YF_object* and *YF_length*(O_i) be the furthest distance between the point P and object O_i.

FF6. If *YF_length*(O_i) is shorter than *YF_length* (*YF_object*), then deletes it, e.g. A in 그림 3, otherwise does not delete it.

FF7. Return the remaining objects, e.g. F in 그림 3.

4.1.2 교차 여과와 포함 여과

교차 여과 알고리즘에서 *M*은 트리내의 최대 항목수를 나타내고, *ROOT*는 트리의 루트 노드를 의미한다. Z-영역의 범위 비트는 각각 주어진 차원 영역을 포함하는 Z-영역에 대한 비트를 나타낸다. 예를 들면, Z-영역 (00101011-00110101)의 영역 비트는 001이다. 또한 최대 영역 비트 길이는 각 차원의 영역 비트 길이 중에서 가장 큰 값을 갖는 길이를 의미한다. 교차 알고리즘은 임의의 질의 윈도우의 경계 영역과 교차하거나 포함되는 객체 블럭을 검색하기 위한 것이고, 검색 알고리즘은 교차 알고리즘을 통해 찾은 객체 블럭에서 질의 윈도우의 경계 영역과 교차하거나 포함되는 객체들을 검색하기 위한 알고리즘이다.

Intersect Filter(*ROOT,I*)

Input: A BR tree rooted at node *ROOT* and a query window $I=(I_0, I_1, \cdots, I_{n-1})$, where I_i denotes the range of i^{th} dimension.

Output: All objects overlapping with *I*

IF1. If *ROOT* is an object block, return all objects overlapping with *I*.

IF2. Let *RBL* be the range bit length and *MB* be the maximum range bit length. Return all the objects that overlap with I in **Search**(*ROOT, O, MB, O, n, RB, RBL, 1, M*).

검색 알고리즘에서 탐색 공간은 각 차원의 영역 비트들과 비교되며, 탐색 공간을 1/2로 감소시킨다. 어떤 차원의 영역 비트인 D_1의 길이가 다른 차원의 영역 비트 D_2의 길이보다 짧을 경우 D_1의 길이와 D_2의 길이를 같도록 하기 위해서 D_1의 영역 비트에 0과 1을 교대로 첨가한다. 영역 비트들은 시작 비트 위치부터 최대 영역 비트 길이까지 차례로 검색되며 리프 노드가 발견될 때까지 순환적으로 검색 알고리즘이 불리워진다. 어떤 노드의 아래와 위 경계는 각 중간 노드의 질의 윈도우와 교차하거나 포함하는 객체 블럭에 대한 최소와 최대 항목을 나타낸다.

Search(*NODE, BP, MB, D, ND, RB, RBL, Lb, Ub*)

Input: *NODE* is the node of a tree, *BP* is the starting bit position of a range bit, *D* is a dimension, *ND* is the number of dimension, *Lb* is the lower bound of *NODE* and *Ub* is the upper bound of *NODE*.

Output: All object blocks overlapping with the region from BP^{th} to MB^{th} range bit, beginning with D^{th} dimension in the search space from *NODE*[*Lb*] *to NODE*[*Ub*].

S1. Initialize *i* to *BP* and *j* to *D*. Do steps S2 through S5 for each i(bit position) and *j*(dimension).

S2. If the i^{th} bit value of j^{th} dimension is O, then the search space lies from *Lb* to [(*Lb*+*Ub*)/2], otherwise it lies from [(*Lb*+*Ub*)/2] to *Ub*.

S3. If current bit position *i* of j^{th} dimension is greater than the range bit length of *i* then call **Search** algorithm for the search space (*Lb,Mid*) and do for the space (*Mid,Ub*), where *Mid* is (*Lb*+*Ub*)/2.

S4. In case that the partitioned region of *NODE* is bigger than that of a query window, if *NODE* is object block then return the block, otherwise change current node to the child node pointed by *NODE*[*Lb*].

S5. In case that the partitioned region of *NODE* is smaller than that of a query window, if the current bit position exceeds *MB*, return all object blocks pointed from *Lb* to *UB* of *NODE*.

질의 윈도우 Q에 대해서 객체 O와 교차하기 위해서는 다음과 같은 조건이 만족되어야 한다.

$QX_{min} \le OX_{max}$ and $QX_{max} \ge OX_{min}$ and
$QY_{min} \le OY_{max}$ and $QY_{max} \ge OY_{min}$.

포함 여과 알고리즘은 질의 윈도우 Q에 대해서 객체 O와 포함하기 위해 다음과 같은 조건이 만족된다는 것을 제외하고는 교차 알고리즘과 같다.

$QX_{min} \le OX_{min}$ and $QX_{max} \ge OX_{max}$ and
$QY_{min} \le OY_{min}$ and $QY_{max} \ge OY_{max}$.

4.1.3 거리 여과

거리 여과는 다음과 같은 질의 "인터콘티넨탈 호텔의 반경 1km 이내에 있는 모든 극장을 검색하라."를 처리하기 위해 요구된다. 질의에서 *P*는 인터컨티넨탈 호텔의 좌표값이라고 가정했을때, 질의 윈도우는 *P*의 확장된 가상 객체로서, *P*를 사방 1km씩 확장함으로써 얻어진다. 거리 여과 알고리즘은 4.1.2절에서 기술한 교차 여과 알고리즘과 같으며, 그림 4는 거리 여과 알고리즘에 대한 객체들의 예를 나타낸다. 그림 4에서 점 *P*의 질의 윈도우는 *Q*이고 거리 여과 단계를 거친 후 출력되는 객체들은 B와 D이다.

4.2 정제 단계

본 장에서는 공간 연산자들인 최근접, 최원격, 교차, 포함의 정제 기법을 기술한다.

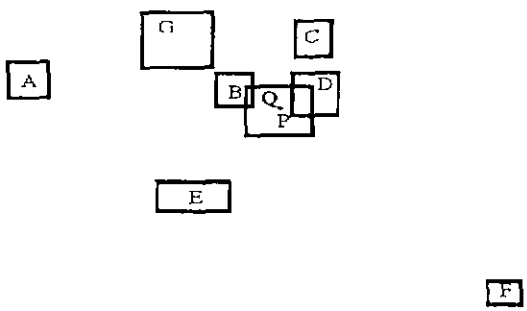

그림 4 거리 여과 단계의 예

4.2.1 최근접 정제와 최원격 정제

사각형이나 다각형과 같은 복잡한 공간 객체들은 여러개 선분 세그먼트들의 집합으로 구성된다. 주어진 점과 임의의 객체 사이의 거리는 그 점과 그 객체를 구성하는 각 선분 세그먼트 사이의 가장 가까운 거리 중에서 최근접한 거리를 의미한다. 주어진 점 P와 두 점 P_1, P_2를 연결하는 선분 세그먼트 사이의 가장 가까운 거리 D는 다음과 같다. 그림 5와 그림 6에서 d_1, d_2, d_{12}는 각각 p와 p_1, p와 p_2, p_1과 p_2 사이의 거리라고 가정한다. 또한, d는 두 점 p_1과 p_2를 연결하는 선분과 직각으로 만나는 교차점 사이의 거리라고 가정한다. 교차점이 존재할때, $d_1^2 \geq d_2^2 + d_{12}^2$이라면 D는 d_2이고, $d_2^2 \geq d_1^2 + d_{12}^2$이라면 D는 d_1이며, $d_{12}^2 \geq d_1^2 + d_2^2$이라면 D는 d를 의미하고, 여기서 d는 $d_2 = d_1^2 - ((d_1^2 - d_2^2 + d_{12}^2)/2d_{12})^2$이다. 교차점이 존재하지 않을 때 $d_1^2 \geq d_2^2 + d_{12}^2$이면 D는 d_2이고, $d_2^2 \geq d_1^2 + d_{12}^2$이면 D는 d_2이다. 즉, 그림 5와 그림 6에서 D는 각각 d와 d_2이다.

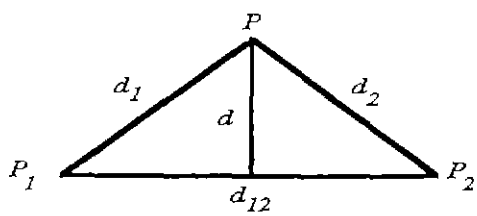

그림 5 교차점이 존재할 때 최근접 정제 예

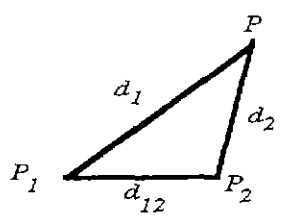

그림 6 교차점이 존재하지 않을 때 최근접 정제 예

4.2.2 교차 정제

그림 7은 교차 정제의 예를 나타낸다. l_{12}는 두 점 $P_1(x_1, y_1)$과 $P_2(x_2, y_2)$를 연결하는 선분이고, l_{34}는 두 점 $P_3(x_3, y_3)$와 $P_4(x_4, y_4)$를 연결하는 선분이라고 할 때, l_{12}와 l_{34}의 교차점의 값은 다음과 같다.

$$X = (m_{12}x_1 - m_{34}x_3 + y_3 - y_1)/(m_{12} - m_{34}) \qquad (1)$$

식 (1)에서, X의 범위는 선분 l_{12}와 l_{34} 사이의 교차점을 갖기 위해서 다음과 같은 조건을 만족해야 한다.

$$(m_{12} \neq m_{34}) \text{ and } (X_1 \leq X \leq X_2) \text{ and } (X_3 \leq X \leq X_4)$$

4.2.3 포함 정제

주어진 점이 어떤 다각형 내에 존재하는가를 알아보기 위해서는 다음과 같은 과정을 거친다. 첫째, 주어진 점과 다각형 외부에 있는 임의의 점 사이의 선분을 그린다. 다각형 외부에 있는 점은 다각형의 가장 작은 X 좌표값보다 작은 X 좌표값에 있는 어떤 점을 선택한다. 다음은 다각형의 경계선과 앞에서 얻은 선분과 교차하는 점의 수를 구한다. 교차점의 수가 홀수이면 주어진 점은 다각형 내부에 존재하고, 짝수이면 외부에 존재한다 [6]. 그림 8은 내부 검사의 예를 나타낸다.

교차점의 수를 구할 때, 교차점이 두 선분이 만나는 점인 경우는 주의해야 한다. 교차점을 구성하는 두 선분 세그먼트의 반대쪽 꼭지점이 첫번째 단계에서 구한 선분과 같은 쪽에 있으면 그 점은 짝수로 계산하고, 다른 쪽에 있으면 홀

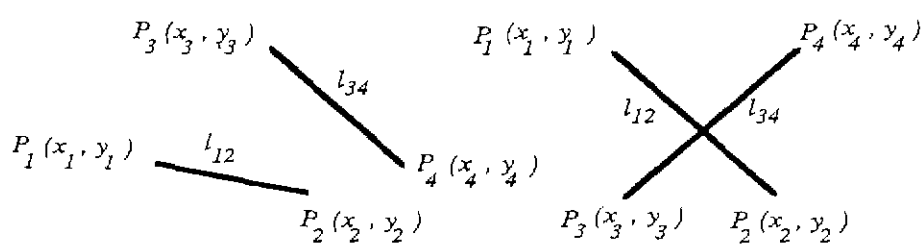

그림 7 교차 정제의 예

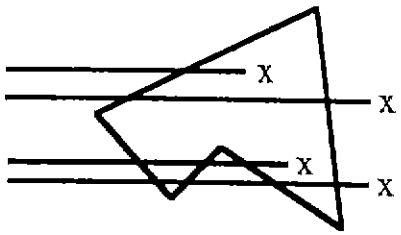

그림 8 내부 검사 예

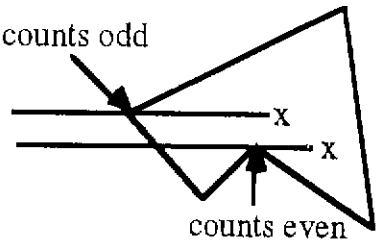

그림 9 교차점의 수 예

수로 계산한다. 그림 9는 교차점의 수 예를 나타낸다.

5. 결 론

본 논문에서는 우리가 제안했던 공간 접근 기법에 기반을 둔 공간 질의 처리 기법을 제안하였다. 또한, 점, 선분, 사각형, 다각형 등의 복잡한 공간 객체를 처리하기 위해 GSQL이라는 공간 질의 언어를 기술하였다. 제안된 질의 처리 기법은 최근접, 최원격, 포함, 교차, 거리 등의 공간 연산자를 처리할 수 있다. 본 논문에서 제안된 질의 처리 기법은 2단계로 구성된다. 첫번째 단계인 여과 단계는 최소 경계 사각형을 이용한 공간 접근 기법을 사용한다. 최소 경계 사각형은 정확한 객체를 표현할 수 없기 때문에 여과 단계는 질의에 대해 정확히 만족하는 객체를 추출할 수 없고, 단지 후보 객체를 구할 뿐이다. 따라서 후보 객체들은 두번째 단계인 정제 단계에서 검사되어야 한다. 정제 단계에서 질의 조건에 만족하는 실 객체를 검출한다.

본 논문에서 제안된 2단계 질의 처리 기법은 복잡한 공간 객체를 간단히 하기 위해서 최소 경계 사각형을 사용하였다. 앞으로 더 연구되어야 할 것은 호수를 포함하는 섬 영역과 같이 빈 공간을 포함하는 객체들도 고려되어야 한다.

참고문헌

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

[2] Chamberlin, D. D., "SEQUEL: A Structured English Query Language," *Proc. of ACM SIGMOD*, pp. 249-264, 1974.

[3] Freeston, M., "The BANG file: A New Kind of Grid File," *Proc. of ACM SIGMOD*, pp. 260-269, 1987.

[4] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching," *Proc. of ACM SIGMOD*, pp. 47-57, 1984.

[5] Harrington, S., *Computer Graphics, McGraw-Hill Book Company*, 1983.

[6] Held, G. D., Stonebraker, M. R., and Wong, E., "INGRES: A Relational Database Management System," *Proc. of AFIPS National Comp.* 44, pp. 409-416, 1975.

[7] Hwang, B. Y., Kim, B. W., and Moon, S. C., "Efficient Access Method for Multi-Dimensional Complex Objects in Spatial Databases: BR Tree," *Journal of Microprocessing and Microprogramming*, Vol. 32, No. 1-5, pp. 765-772, 1991.

[8] Hwang, B. Y., Byun, B. K. and Moon, S. C., "Spatial Query Processing in Geographic Database Systems," *Proc. of 20th Conf. on EUROMICRO*, pp. 53-60, 1994.

[9] Kriegel, H. P. and Seeger, B., "PLOP-Hashing: A Grid File without Directory," *Proc. of 4th Int. Conf. on Data Engineering*, pp. 369-376, 1988.

[10] Kriegel, H. P., Horn, H., and Schiwietz, M., "The Performance of Object Decomposition Techniques for Spatial Query Processing," Proc. of 2nd Symp. on Large Spatial Databases, pp. 257-276, 1991.

[11] Lu, W. and Han, J., "Distance-Associated Join Indices for Spatial Range Search," *Proc. of 8th Int. Conf. on Data Engineering*, pp. 284-292,

1992.

[12] Ooi, B. C., *Efficient Query Processing in Geographic Information Systems,* P. 208, Springer Verlag, 1990.

[13] Orenstein, J. A., " Spatial Query Processing in An Object-oriented Database System," *Proc. of ACM SIGMOD,* pp. 326-336, 1986.

[14] Orenstein, J. A., "Redundancy in Spatial Databases," *Proc. of ACM SIGMOD,* pp. 294-305, 1989.

[15] Samet, H., "The Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys,* Vol. 16, pp. 187-260, 1984.

[16] Seeger, B. and Kriegel, H. P., "The Buddy Tree: An Efficient and Robust Access Method for Spatial Databases," *Proc. of 16th Int. Conf. on Very Large Databases,* pp. 590-601, 1990.

[17] Sellis, T., Roussopoulos, N., and Faloutsos, C., "The R^+-tree: A Dynamic Index for Multi-dimensional Objects," *Proc. of 13th Int. Conf. on Very Large Databases,* pp. 507-518, 1987.

[18] Shaffer, C. A. and Brown, P. R., "A Paging Scheme for Pointer-based Quadtrees," Proc. of 3rd Symp. on Large Spatial Databases, pp. 89-104, 1993.

[19] Zloof, M. M., "Query-by-Example: Operations on Hierarchical Databases," *Proc. of National Comp. Conf* 45, pp. 845-853, 1976.

황 병 연

1986 서울대학교 공과대학 컴퓨터공학과(학사)
1989 한국과학기술원 전산학과(석사)
1994 한국과학기술원 전산학과(박사)
1994 ~현재 가톨릭대학교(구 성심여자대학교) 전산학과 조교수
관심 분야 : 공간 데이타베이스, 멀티미디어 데이타베이스, 객체지향 데이타베이스