공간 데이터베이스에서 방향과 거리 관계가 혼합된 질의어로부터 점진적으로 가까운 객체 추출에 대한 연구

권준희, 윤종필
숙명여자대학교 전산학과
e-mail : kwonjh@sicc.co.kr, jyoon@sookmyung.ac.kr

Combining direction and distance operations for querying incrementally close objects

Joonhee Kwon, Jongpil Yoon
Computer Science Dept. Sookmyung Women's University

요약

공간 데이터베이스에서 공간 데이터간의 근접성을 알아보아야 할 필요가 많이 발생한다. 이를 위해, 본 연구에서는 방향 관계와 거리 관계가 혼합된 형태의 질의에서 공간적으로 가까운 객체를 순서적으로 추출해내는 방법을 제안한다. 점진적 거리 조인 알고리즘을 근간으로 섹터 기반 모델을 적용하여 객체들을 순서적으로 추출할 수 있도록 한다. 섹터의 분류와 섹터들간 정렬 순서에 따라 추출된 값은 필요에 따라 제한조건의 조절이 가능하다는 장점이 있다. 또한, 점진적 거리 조인 알고리즘에 있어서의 데이터 크기에 따른 성능 저하 문제도 어느 정도 해결할 수 있음을 기술하고 있다.

1. 서론

공간 데이터베이스에서 공간 데이터간의 근접성을 알아보아야 할 필요가 종종 발생한다. 공간 데이터간의 근접성은 데이터간의 관계에 따라 달라지게 된다.

공간 데이터에서 거리(distance), 방향(direction), 위상 (topology) 관계가 존재한다. 공간 데이터간에 공간적으로 근접하다는 것은 거리상 서로 가깝고, 서로 같은 방향에 있고, 위상적으로 서로 관련이 있다면 근접한 관계에 있다고 볼 수 있을 것이다.

그 동안 거리 관계에 대해서는 주로 근접성에 대한 연구가 이루어져 왔다[7]. 이에 반해 방향과 위상 관계에 대해서는 관계의 종류를 정형화하고 그러한 관계에 대한 질의를 보다 효율적으로 처리하고자 하는 연구가 많이 이루어져 왔다[5,6,8]. 또한 공간 관계가 혼합된 형태의 연구에 대해서는, 혼합된 공간 관계에 있어서 보다 효율적인 질의어 처리에 대한 연구[4]와, 방향 관계와 거리 관계가 혼합된 추론(reasoning)에 대한 연구[2,3]가 있었다. 하지만, 공간 데이터간 근접성을 여러 공간 관계가 혼합된 형태에서 살펴보고자 하는 연구는 없었다.

본 연구에서는 이를 위해, 여러 공간 관계가 혼합된 형태에서 공간 데이터간 근접성을 결정하는 방법에 대해 연구한다. 특히, 공간 관계 중 거리와 방향 관계가 혼합된 질의에 한정해서 데이터간 공간적으로 근접한 정도를 결정하는 방법에 대해 제안하고자 한다. 위상 관계에 대한 연구는 향후 연구 과제로 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 연구와 관련해서 이루어진 연구 내용들을 소개한다. 3 장에서는 본 연구에서 제안한 거리와 방향 관계가 혼합된 질의어에서 점진적으로 가까운 객체 추출에 대한 알고리즘과 그 특징들을 설명한다. 끝으로 4 장에서는 결론 및 향후 연구 과제를 논한다.

2. 관련 연구

2.1 점진적 거리 조인 알고리즘

공간 데이터베이스에서 자주 사용되는 질의어 중 하나가 "XX 에서 가까운 순(혹은 가장 가까운)으로 위치해 있는 객체를 찾아라"이다. 이에 대한 연구가 근접성(nearest neighbor)질의어이다. 기존의 근접성 질의어에 관한 연구는 상대적으로 관심이 없는 먼 거리에 있는 데이터까지 모두 찾은 후 정렬하는 불필요한 과정이 있었다. 이러한 문제점을 보완한 알고리즘이 점진적 거리 조인 알고리즘(incremental distance join algorithm)이다[1]. 점진적 거리 조인 알고리즘의 모조 코드(pseudo code)는 [1]을 참조하기 바란다.

점진적 거리 조인 알고리즘의 특징은 우선순위 큐를 사용하여, 가까운 거리에 있는 객체가 우선적으로 결과로 나올 수 있도록 한 점이다. 이로써 관심 있는 객체부터 결과가 곧바로 추출되어, 상대적으로 관심이 없는 객체는 생략 가능하여 효율을 높일 수 있었다.

이 알고리즘을 약간 변형하면, 거리가 먼 순으로 객체를 얻거나, 가장 가까운 거리에 위치한 객체만을 얻을 수도 있다. 장점은 알고리즘에 대한 약간의 변형만으로도 다양한 결과를 얻을 수 있다는 점과, 사용자들이 주로 관심이 되는 결과에 보다 집중해 우선적으로 그 결과를 볼 수 있게 했다는 점이다.

반면 단점은 우선순위 큐에 드는 메모리 크기에 있다. 즉, 우선순위 큐에 모든 데이터를 다 저장하지 못하는 경우 디스크에 저장하는 방식을 취한다. 따라서 데이터가 메인 메모리에 저장할 수 있는 수준보다 커지게 되면, 입출력 시간 때문에 성능이 급격히 떨어지게 된다는 문제점을 가진다.

2.2 거리와 방향이 혼합된 형태에서 정량적이고 정성적인 추론

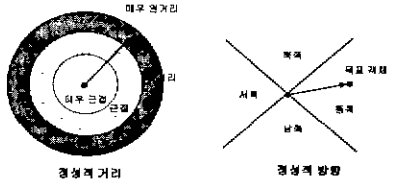
공간 관계가 혼합된 형태의 연구에 대해서는, 방향 관계와 거리 관계가 혼합된 추론에 대한 연구[2,3]가 있었다. 이 중, 추론에 대한 연구는 본 연구와 직접적인 관련성이 없으므로 생략하기로 하고, [2]에서 사용한 거리와 방향에 대한 정량적(quantitative)이고 정성적(qualitative)인 표현 방법에 대해 논의하고자 한다.

정량적인 거리와 방향에 대한 모델을 설정하기 위해 이 연구에서는 방향과 거리에 대해 심볼값을 부여하였다. 즉, <그림 2.1>과 같은 4 가지 심볼 거리값과 8 가지 심볼 방향값을 부여하였다. 이렇게 부여된 정성적인 값을 정량적인 값으로 매핑하기 위해서는, 정성적인 값을 정

량적인 값의 범위로 매핑하는 방법을 사용하였다. 이를 그림으로 표현하면 <그림 2.2>와 같은 섹터 기반 모델(sector-based model)이 된다. 각 섹터는 방향과 거리에 대한 심볼값 중 어느 한 부분에 속하게 되어, 같은 섹터 내의 객체들은 같은 정량적인 위치 관계를 가지게 된다.

거리 : {매우 근접, 근접, 원거리, 매우 원거리}
방향 : {북쪽,북동쪽,동쪽,남동쪽,남쪽,남서쪽,서쪽,북서쪽}

<그림 2.1 심볼 거리값과 심볼 방향값>



<그림 2.2 섹터 기반 모델>

3. 본론

본 연구에서는 정성적 거리와 방향을 정량적 거리와 방향 관계로 매핑한 섹터 기반 모델을, 점진적 거리 조인 알고리즘에 도입하여 방향과 거리 관계에 있어 공간적으로 가까운 데이터를 찾는 방법을 제안하였다.

3.1 방향 관계와 거리 관계를 혼합한 질의

공간 관계 질의어에는 다양한 형태가 존재한다. 이러한 질의어는 한가지 형태의 공간 관계로 이루어진 경우와, 두가지 이상의 공간 관계가 혼합된 형태로 이루어진 경우가 존재한다. 이러한 공간 관계의 형태 이외에 더 요구되는 사항 중 하나는 공간적으로 서로 근접한 정도를 알고자 하는 것이다. 공간적으로 서로 근접하다면, 두개의 객체들은 그렇지 않은 객체들보다 관련성이 높다고 볼 수 있을 것이다.

이러한 질의어에서 문제가 되는 경우는 두가지 형태 이상의 공간 관계가 혼합된 경우 근접성을 어떻게 결정하는가이다. 이 중 본 논문에서는 방향 관계와 거리 관계가 혼합된 형태를 대상으로 제한하였다.

예를 들어, 다음과 같은 경우를 살펴보도록 한다. A하천으로부터 북쪽 방향으로 거리가 가까운 공장이 A하천 수질 오염의 주된 원인이라고 가정하자. 오염의 주 원인이 되는 공장을 찾고자 한다. 이 때, 북쪽 방향의 모든 공장을 대상으로 그 원인 분석을 한다면 매우 소모적인 작업이 될 것이다. 이러한 경우, A하천 수질 오염에 영향을 많이 끼친 공장을 A하천으로부터 가까운 순서로 찾아내어 제시할 수 있다면, 순서적인 원인 규명이 가능하므로 보다 효율적으로 원인을 찾아낼 수 있을 것이다. 이를 질의어로 표현하면 다음과 같다.

```
SELECT factory
FROM factory, river
WHERE river.name='A' and
      distance_close_to(river.geo, factory.geo) and
      direction(river.geo, factory.geo)='North'
ORDER BY close_to(river.geo, factory.geo);
```

이 중, where 절에 나타난 "distance_close_to"는 2개의 객체가 거리적으로 서로 근접한가를 알려주는 함수 (UDF: User Defined Function)이고, "direction"은 2개의 객체간 방향을 알려주는 UDF이다. order by 절에 나타난 "close_to"는 방향 관계와 거리 관계가 혼합되어 있을 때 서로 가까운 정도를 알려주는 UDF이다.

이러한 질의어에 대한 결과를 내보내기 위해서 가장 핵심이 되는 문제는, "close_to"라는 UDF를 어떻게 결정해야 하는가이다.

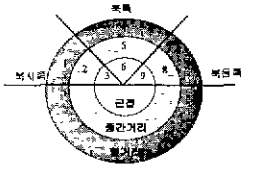
3.2 방향과 거리 관계가 혼합된 형태에서 공간적으로 가까운 객체를 추출하는 방법

"close_to"함수에 있어 고려할 점은 방향과 거리를 혼합하여 어떻게 가까운 정도를 결정할 수 있느냐는 것이다. 사실 거리적으로 가깝다거나, 방향적으로 가깝다는 말은 정성적인 표현이다. 정성적인 거리와 방향에 있어서의 속성은 부정확한 지오메트리(geometry)의 의미에 있다. 정량적인 관계가 숫자값으로 나타나는데 비해, 정성적인 관계는 기호값으로 나타날 수 있다.

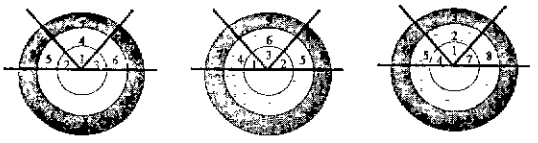
본 연구에서는, 이를 위해 다음과 같은 방향과 거리에 관한 기호값을 사용하기로 한다. 하지만, 이러한 기호값은 필요에 따라 보다 자세하게 세분될 수도 있겠다.

거리 : {근접, 중간거리, 원거리}
방향
- 북쪽 : {북서쪽, 북쪽, 북동쪽}
- 동쪽 : {북동쪽, 동쪽, 남동쪽}
- 서쪽 : {북서쪽, 서쪽, 남서쪽}
- 남쪽 : {남서쪽, 남쪽, 남동쪽}

이와 같이 정의된 심볼값에 따라 위치 관계 시스템을 섹터 기반 모델로 만든다[2]. 예를 들면, 북쪽 방향에 대해 만들어진 섹터의 모양은 <그림 3.1>과 같다.



<그림 3.1 북쪽 방향에 대해 분할된 섹터>



<그림 3.2 섹터별로 부여된 순번의 예>

분할된 섹터는 숫자로 표시를 했으며, 정의된 기호값에 따라 그림에서 보는 바와 같이 총 9개의 섹터로 나뉘어지고 있음을 알 수 있다. 기호값의 종류가 보다 더 세분화되면 분할된 섹터수는 보다 더 많아지게 된다.

이와 같이 정의된 섹터 기반 모델을 바탕으로, 공간적으로 가까운 객체를 어떻게 얻어내는지 간략하게 알고리즘을 소개하면 아래와 같다.

우선, 섹터별로 가까운 순으로 순번을 부여한다. 이는 방향과 거리 중 어느 부분을 더 고려해서 가까운 순번을 부여하느냐에 따라 달라진다. 이를 위해 방향과 거리 함수를 정의한다.

이렇게 부여된 순번에 따라 우선 순위가 높은 섹터부터 점진적 거리 조인 알고리즘을 적용한다. 한 섹터 내에서의 가까운 정도값은 이미 정의된 방향과 거리 함수를 사용하여 결정한다. 예를 들어, <그림 3.1>과 같이 분할된 섹터에 있어 나타날 수 있는 섹터별 순서 중 일부를 그림으로 나타내면 <그림 3.2>와 같다. 이 때, 섹터에 표시한 순번은 섹터간 정렬된 순번으로 1이 가장 가까운 섹터이고, 9가 가장 먼 섹터임을 의미한다.

즉, 이 알고리즘의 기본 아이디어는 거리와 방향을 혼합하기 위해 함수를 정의하고, 이 함수를 이용하여 섹터라는 부분 영역을 순차 정렬시킨 후, 다시 각 섹터 내에서 순차 정렬을 시킴으로써 각 객체와의 가까운 정도값이 순차적으로 정렬되게 한다.

이를 모조 알고리듬(pseudo algorithm)으로 나타내면 <그림 3.3>과 같다. 이 중, IncDirDistJoin 알고리즘이, 기존의 점진적 거리 조인 알고리즘과 다른 부분은 이텔릭체로 표시하여 나타내었다. 서브 프로그램인 ProcessNode1과 ProcessNode2는 기존 알고리즘과 유사하므로 기술하지 않았다.

```
Boolean Close(distance1,direction1,distance2,direction2)
/* distance1 direction1 의 근접값이 distance2, direction2 의 근접값에 비해 더 가까운지
   결정한다  이러한 결정값은 섹터별로 순차 정렬시킨 규칙에 따른다 */
1 If (distance1,direction1 의 근접값 < distance2,direction2 의 근접값) then
2          return true;
3 Else return false.

IncDirDistJoin(R1, R2)
1 비교대상이 되는 객체를 섹터로 분류
2 분류된 섹터에 순차 정렬시킨 규칙을 적용하여 섹터당 순번 결정
(가정   결정된 순번값은 SortedSector 배열에 순번값 순으로 섹터번호 부여
        전체 섹터의 수는 SortedMax 로 두어)
3 Q <- NewPriorityQueue(),
4 For (int i=0, i<SortedMax, i++)
5   ENQUEUE(Q, 0, 0, <RootNode(R1), RootNode(SortedSector[i])>),
6   While ( not ISEMPTY(Q) ) do
7     Elem <- DEQUELE(Q),
8     If (both items in Elem are data objects) then
9       Report Elem;
10    Else if (both items are object bounding rectangles) then
11      let O1 and O2 be the corresponding objects references,
12      Distance= O1과 O2의 distance;
13      Direction=O1과 O2의 direction,
14      If ( IsEmpty(Q) or
            (Close(Distance,Direction,  Front(Q) Distance,Front(Q).Direction)}
15          ) then   Report(O1,O2):
16      Else
17          ENQUEUE(Q,Distance,Direction,< O1,O2>),
18      Endif
19    Else if (item 1 in Elem is a node) then
20        PROCESSNODE1(Q,Elem),
21    Else  PROCESSNODE2(Q,Elem),
22    Endif
23  EndDo
24 EndFor
```

<그림 3.3 방향과 거리 관계가 혼합된 형태에서 공간적으로 가까운 객체를 추출하는 모조 알고리즘>

3.3 고찰

본 절에서는 3.2절에서 살펴본 알고리즘의 의미를 보다 살펴보고, 기존의 점진적 조인 알고리즘에 비해 향상된 점을 살펴보도록 한다.

3.3.1 Close 함수의 의미

Close 함수는 방향과 거리에서 비교 대상이 되는 객체에 비해 가까운지를 결정하기 위해, 섹터를 정렬시킨 순서와 같은 방법으로 섹터 내의 순서를 결정하는 불리언(boolean) 함수이다.

Close 함수의 1번째 라인이 구체적으로 어떻게 행해지는지 살펴보면, 다음과 같다.

첫째, 객체의 가까운 정도값을 결정하기 위해서, 객체당 (거리,방향)이라는 순서쌍을 만든다. 따라서, 기준 객체와, 비교 객체에 대해서 각각 (distance1,direction1)과 (distance2, direction2)라는 2개의 순서쌍이 만들어지게 된다.

둘째, 섹터별로 순차 정렬시킨 근접값 결정 순서에 따라 거리 우선, 혹은 방향 우선으로 가까운 정도를 비교한다. 예를 들면, 정렬 순서가 <그림 3.2>의 (다)와 같다고 하자.

이 때의 근접값을 결정하기 위한 순서는 다음과 같다.

```
If (direction1 < direction2) then
   If (distance1 < distance2)  then 객체 1이 객체 2에 비해 가깝다.
Else 객체 1이 객체 2에 비해 가깝지 않다.
```

다른 정렬 규칙에 대해서도 이와 유사한 방법을 적용하면 되겠다.

3.3.2 섹터 분할에 따른 의미

본 연구에서는 사용자가 선택할 수 있는 인자값으로 섹터를 사용하고 있다. 즉, 섹터의 크기와 기호값은 상황에 따라 변화가 가능하다.

섹터의 방향과 거리는 세분화 정도에 따라 다음과 같은 특징을 가지게 된다.

섹터의 방향과 거리에 관한 기호값은 세분화될수록, 섹터의 크기는 더 작아지고 보다 정밀하게 거리와 방향을 혼합하여 가까운 정도값을 결정할 수 있게 된다. 또한, 섹터의 크기가 더 작아질수록, 메인 메모리에 저장할 데이터 크기가 작아지므로 입출력 시간 때문에 생기는 성능 저하 문제가 줄어들 수 있다.

하지만 섹터가 지나치게 세분화되면, 섹터의 분류와 기호값을 부여하는데 시간이 많이 소요된다 또한, 디스크에서 메인 메모리로 한 섹터를 읽어들이는 횟수가 세분화된 섹터의 갯수만큼 된다 따라서, 메인 메모리로 한 섹터를 읽어들이는데 드는 입출력 시간이 많이 걸리게 되어 성능 저하의 문제가 야기될 수 있다

따라서, 섹터당 객체의 수가 메모리에서 작업하기에 충분한 정도의 수준에서 그리고 값의 정확성 정도에 비추어 지나치게 세분화하지 않는 수준에서 섹터를 세분화하면 되겠다

3.3.3 점진적 조인 알고리즘에 비해 향상된 면

점진적 조인 알고리즘에 있어 문제점으로 지적되는 점은 우선순위 큐에 드는 메모리 크기이다 이러한 문제점은 우선순위 큐에 의해 점진적인 결과 추출이 가능한 반면 이로 인해 생기는 역기능이라고 볼 수도 있겠다

점진적 조인 알고리즘은 우선순위 큐에 모든 데이터를 다 저장하지 못할 경우 디스크에 데이터를 저장하게 된다. 따라서, 데이터가 메인 메모리에 저장할 수준을 넘어서게 되면, 계속적인 입출력 시간 때문에 성능이 급격히 떨어지게 되는 문제점을 갖는다[4].

본 연구에서 제안된 방법에서는 이러한 문제점을 어느 정도 해결하며 섹터 크기가 우선순위 큐의 크기가 된다. 따라서, 데이터의 크기에 비해 매우 작은 섹터의 크기는 메인 메모리에 모든 데이터가 저장될 가능성이 상대적으로 커지게 되어 성능 지하의 문제가 줄어든다. 또한 섹터의 크기는 섹터의 방향과 거리에 따른 세분화 정도에 따라 조절이 가능하므로, 문제점이 발생할 때 튜닝이 가능하다.

4. 결론과 향후 연구방향

본 연구에서는 방향 관계와 거리 관계가 혼합된 형태의 질의에서 공간적으로 가까운 객체를 순서적으로 추출해는 방법을 제안하였다. 이를 위해, 각 객체들을 우선적으로 섹터로 부분 정렬한 후, 섹터 각각에 대해서는 점진적 조인 알고리즘을 근간으로 하여 최종적으로는 각 객체들이 완전 정렬되어 순서적으로 추출될 수 있도록 하였다

이의의 부가적인 연구 공헌으로는, 점진적 조인 알고리즘을 사용하는데 있어서의 문제점인 데이터 크기에 따른 입출력 시간의 문제점을 어느 정도 해결하였다는 점이다. 즉, 섹터 크기의 데이터로 데이터 크기를 제한함으로써, 데이터 크기가 커짐에 따라 입출력 시간에 의해 성능이 크게 저하되는 문제점을 해결할 수 있었다.

향후 연구방향은 다음과 같다.

첫째, 위상 관계를 더 추가하여 모든 공간 관계에 대한 근접성을 결정할 수 있도록 알고리즘을 확장시킬 수 있는 연구가 필요하다.

둘째, 공간 관계의 근접성을 이용하여 이를 공간 데이터 마이닝과 같은 응용분야에 적용하여 그 성능을 확인해볼 수 있어야 한다.

참고문헌

[1] G.R. Hjaltason and H Samet, "Incremental Distance Join Algorithms for Spatial Database", Proc. of ACM SIGMOD, pp237-248, Seattle, June 1998

[2] J.H Hong, M Egenhofer, A. Frank, "On the Robustness of Qualitative Distance and Direction Reasoning", Proc. Of Autocarto 12, D.Peuquet(Ed), Charlotte, Nc, February 1995

[3] J H. Hong, "Qualitative Distance and Direction Reasoning in Geographic Space", Ph.D Thesis, University of Maine, 1994

[4] D.Papadias, Y.Theodorisis, "Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures", IJGIS 11(2), pp111-138, 1997

[5] D.Papadias, Y.Theodoridis, T.Sellis, "The Retrieval of Direction Relations Using R-Trees", Proc. Of the 5th DEXA Conference, Springer Verlag, September 1994

[6] D.Papadias, Y.Theodoridis, T.Sellis, M.Egenhofer, "Topological Relations in the World of Minimum Bounding Rectangles A Study with R-trees", Proc. of ACM SIGMOD, May 1995

[7] N.Roussopoulos, S Kelley, and F.Vincent, "Nearest neighbor queries", Proc. of ACM SIGMOD, San Jose, CA, May 1995

[8] Y. Theodoridis, D Papadias, Em stefanakis, "Supporting Direction Relations in Spatial Database Systems", Proc. Of the 7th SDH Symposium, August 1996