DOT 공간조인 알고리즘의 효율적인 여과단계 처리

유용혁* 백현* 윤지희* 이건배**
* 한림대학교 컴퓨터공학과
** 경기대학교 전자기계공학부
{yhyu, hback, jhyoon}@iris.ce.hallym.ac.kr kblee@kuic.kyonggi.ac.kr

Efficient Filter Step of DOT Spatial Join Algorithm

Yong-Hyuk Yu Hyun Back Jee-Hee Yoon Keon-bae Lee
Dept. of Computer Engineering, Hallym University
Dept. of Electronic and Mechanic Engineering, Kyonggi University

요 약

공간조인 연산은 지리정보시스템의 연산 중 매우 높은 처리비용을 요구하는 연산이다. DOT 공간색인 기법은 전통적인 데이터베이스 시스템의 주색인 기법을 적용할 수 있으며, 공간객체의 상호인접성이 유지되도록 Hilbert 값으로 정렬되어 클러스터링 된다. 이러한 특징을 이용한 DOT공간조인 알고리즘은 적정한 버퍼크기를 유지하는 경우 잘 알려진 R*-tree를 이용한 공간조인 알고리즘에 비해 디스크 액세스면에서 유리한 장점이 있으나, 조인가능영역 산출시 많은 양의 공간변환 연산을 필요로 하므로 전체적인 성능이 만족스럽지 못하다. 본 논문은 DOT 공간조인 알고리즘의 성능을 향상시키기 위하여 이러한 공간변환 연산의 횟수를 최소화시킨 효율적인 여과단계처리 방법을 제시하며, 이를 적용한 DOT 공간조인 알고리즘과 R*-tree 공간조인 알고리즘의 실행시간을 비교 분석하여 DOT 공간조인 알고리즘이 최대 약 2배까지 우수한 성능을 가지고 있음을 보인다.

1. 서론

지리정보시스템(GIS : Geographic Information System)은 다차원 공간 객체를 액세스 할 수 있는 효율적인 공간색인 기법과 공간 연산자를 필수적으로 제공하여야 한다. 특히, 점질의, 영역질의, 공간조인과 같은 공간 연산자 중 공간조인(Spatial Join)은 두 객체집합에 대하여 교차, 포함과 같은 공간 관련성을 만족시키는 객체쌍의 집합을 생성하는 연산으로 가장 많은 계산량과 디스크 액세스를 필요로 하게 된다.

본 논문에서는 DOT(DOuble Transformation) 공간색인 기법[2,3]을 이용한 공간조인 알고리즘[1]의 성능향상을 위하여 효과적인 여과단계(filter step) 처리 방법을 제안하고 이를 적용한 DOT 공간조인 알고리즘과 R*-tree를 이용한 공간조인 알고리즘[4]의 실행시간을 비교 분석한다.

2. 관련연구

DOT 공간색인 기법은 원공간에 존재하는 공간객체의 MBR정보를 DOT(DOuble Transformation)기법에 의해 1차원적인 하나의 값으로 변환하고, 이 값을 검색키(search key)로 하는 B^+-tree 색인 구조를 구성하는 방법을 말하며, DOT기법의 단계적인 정의는 다음과 같다.

(1) k차원의 원공간(Initial space)상의 객체들을 표현하는 MBR정보를 사용하여 2k차원의 중간공간(Intermediate space)상에 단편화 없이 하나의 점으로 사상(mapping)한다. 이것을 1차 변환이라 한다.

(2) 공간 순서화 곡선(Space Filling Curve)을 이용하여 중간공간에 사상된 공간 객체들을 1차원상의 한 점을 나타내는 값(x-value)으로 변환한다. 이것을 2차 변환이라고 하며, 공간 순서화 곡선은 원래의 거리간격을 최대한 보존하여 1차원으로 표현할 수 있는 곡선으로 선택되는데 여기에는 Hilbert 곡선을 이용하는 것이 가장 유리한 것으로 알려져 있다[3].

공간조인 연산은 여과(Filter)와 정제단계(refinement)로 나뉘어 진다[5]. 여과 단계는 각 공간객체의 근사적 표현법인 최소경계사각형(Minimum Bounding Rectangle:MBR)을 사용한 공간조인(MBR 공간조인) 단계로서, 최종조인 결과에 남을 가능성이 있는 후보 쌍을 여과해 내는 작업을 수행하며, 정제단계는 여과단계에서 얻어진 각 후보 객체 쌍에 대하여 정밀한 기하연산을 적용하여, 최종적인 조인결과를 산출하는 작업을 수행한다. 일반적으로 공간조인 연산의 정제단계는 대단히 많은 계산량을 필요로 하므로, 효율적인 여과 알고리즘을 선택하는 것이 공간조인연산 성능에 커다란 영향을 미치게 된다.

관련연구 [1]에서는 DOT색인을 이용한 공간조인 알고리즘을 제안하였으며, 적정한 크기의 버퍼를 유지하는 경우 잘 알려진 R*-tree를 이용한 공간조인 알고리즘보다 디스크 액세스 면에서 유리함을 밝혔다. 이 연구에서는 한번에 액세스되는 객체들의 MBR정보를 이용하여 가장 작은 시작점과 가장 큰 끝점을 좌표로 하는 대표객체로 삼고, 이를 통해 반복적인 영역질의(Range query)를 수행함으로써 여과단계를 처리한다. 이 때 공간조인 알고리즘에서는 DOT 공간색인이 가지는 공간객체의 상호인접성 유지라는 특성을 활용하여 전체 디스크 액세스 수의 절감효과를 얻을 수 있다.

Step1. 파일 R로부터 데이터페이지를 읽어옴
Step2. 페이지 내에서 대표객체로 적용할 값을 계산
Step3. 대표객체의 MBR값으로 파일 S와 영역질의
Step4. Step3에서 구해진 영역질의 결과와 파일 R과의 정제(refinement)단계 처리
Step5. Step4에서의 결과를 공간조인 결과 집합에 합하고, 파일 R의 모든 데이터페이지를 검색할 때까지 Step1부터 반복

알고리즘 1. DOT색인을 이용한 공간조인 알고리즘

알고리즘 1은 관련연구 [1]에서 제안한 DOT공간조인 알고리즘의 처리과정을 단계별로 설명한 것이다. 여기서 R과 S는 각각 DOT색인을 가지는 1차원객체의 파일을 의미한다. step3은 공간조인처리의 여과단계에 해당하며, step4는 정제단계에 해당한다. 이 알고리즘의 여과단계

에서는 질의영역을 DOT색인값의 연속되는 범위로 표현하기 위하여(라인 세그먼트의 집합) 질의영역에 속하는 모든 점(cell)들에 대해 2차변환 연산을 수행한다. 그 이유는 공간순서화곡선으로 적용된 Hilbert곡선이 질의영역내에서 반드시 연속적이라 단정할 수 없기 때문인데, 이때 수행되는 변환연산의 횟수가 공간조인 알고리즘의 성능을 저하시키는 주된 원인으로 작용할 수 있다.

3. 효율적인 여과단계 처리를 위한 접근방법

중간공간상에 대표객체에 의해 정해지는 질의영역에 속하는 점들의 갯수는 무시할 수 없을 정도로 많다. 질의영역을 중간공간의 대각선 위쪽 삼각형 전체라고 가정하면 중간공간의 크기 n=1024라고 하였을 때 점들의 갯수는 $(1024^2+1024)/2 = 524,800$개가 된다. 이러한 점들에 대하여 공간 순서화 곡선을 적용하여 연속적인 라인세그먼트의 집합으로 변환하기 위해서는 다음과 같은 접근방법이 있다.

1) 순차 변환

중간공간의 질의영역을 순차적으로 주사(scan)하면서, 영역에 속한 모든 점(cell)들을 DOT색인값(x-value)으로 변환하고, 그 값들의 연속성을 조사하여 라인-세그먼트를 구성하는 방법이다. 이 방법은 처리가 단순한 반면, 질의 영역내의 모든 점들에 대하여 변환연산을 적용하기 때문에 변환 연산의 횟수가 많음은 물론 유지해야 할 중간결과의 집합이 커서 메모리 유지비용이 매우 높은 단점을 가진다. 대표객체가 중간공간에 사상되는 점을 q라고 할 때, q의 위치에 따른 변환연산 횟수는 다음과 같다.

(1) 점 q가 중간공간의 좌상점에 위치할 때 질의영역의 면적이 가장 크며, 질의영역에 속하는 점들의 개수는 $(n^2+n)/2$개이므로 $(n^2+n)/2$번의 변환연산을 수행한다.

(2) 우상점이나 좌하점에 위치하는 경우 질의영역의 크기가 가장 작으며, 그 면적은 중간공간의 한 변의 길이와 같다. 따라서, 이 경우 질의영역에 속하는 점(cell)들의 개수는 n개이며 n번의 변환연산을 수행한다.

따라서 공간변환연산의 횟수를 St라고 하면 $[n \le St \le (n^2+n)/2]$이며, 변환결과를 유지하기 위한 메모리공간 역시 St*length로 표현되고 이로 인해 원공간의 크기가 큰 경우에는 적용하기 어려운 접근방법이 된다.

2) 역방향 변환

중간공간의 모든 점을 변환하면서 메모리비용을 절감하는 방법이다. 먼저, 질의영역의 테두리에 해당하는 경계부분만을 DOT색인값으로 변환 한 뒤, 변환된 DOT색인 값들을 이용하여 그들 사이의 숫자범위가 질의영역에 포함되는지를 역으로 조사하여(Inverse transformation) 라인-세그먼트를 구성하는 방법이다. 이 방법에서 필요한 메모리 유지비용은 질의영역의 둘레에 해당하는 점의 개수와 비례하므로 최소(좌하점이나 우상점에 점q가 사상되는 경우) n 개로부터 최대 3n개(q가 좌상점에 사상되는 경우)의 변환결과를 유지하는 비용이 소요된다. 그러나 변환연산의 횟수는 여전히 순차변환의 경우와 같다. (질의영역의 둘레 + 둘레를 제외한 면적의 역방향 변환)

3) 쿼터 분할 변환

역방향 변환방법으로 메모리 유지비용을 개선 할 수 있으나 여전히 변환연산의 횟수는 만족스럽지 못하다. 쿼터 분할 변환 방법은 중간공간을 주사하는 공간순서화곡선(Hilbert curve)의 특징을 이용한 것으로 메모리유지비용과 변환연산비용을 동시에 절감하는 방법이다.

DOT색인 값을 만들기 위하여 공간순서화 곡선으로 적용된 Hilbert 곡선은 중간공간을 재귀적인 규칙을 가지고 주사(scan)하여 얻어지는 1차원 값(x-value)들의 집합이다. Hilbert 곡선은 그림 1-(a)와 같이 중간공간의 한 변의 길이를 재귀적으로 n/2인 정방사각형(quarter)으로 분할하였을 때 연속적인 값을 갖도록 운행(traverse)한다. 이러한 성질을 이용하여 최종공간에서 연속적인 x-value 범위를 분할되는 쿼터를 통해 예측할 수 있다.

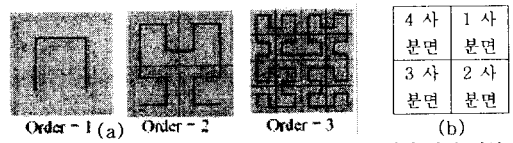


그림1.Order에 따른 Hilbert 곡선의 운행과 쿼터의 분할 사분면

또한, 분할되는 쿼터 내에서 x-value의 시작점을 알게 되면 쿼터의 크기를 이용하여 라인-세그먼트의 범위를 쉽게 계산할 수 있다. 쿼터로 분할되는 각 사분면을 그림 1의(b)와 같이 정할 때 Hilbert 곡선의 운행 규칙은 다음과 같다.

(1) Hilbert 곡선은 처음에 중간공간을 3-4-1-2사분면의 순서로 운행하며,이를 'pattern3412'라고 표현한다.

(2) Hilbert 곡선은 분할된 쿼터를 운행했던 순서에 따라 다음 Order에서의 운행 규칙이 정해져 있으며 표1과 같다. 표1의 Depth는 분할될 쿼터의 운행규칙을 의미하며, 그 쿼터가 분할되어 각 사분면의 새로운 쿼터들이 가지는 운행규칙이 Depth+1에 나타나 있다. 또, Hilbert 곡선의 시작점은 'pattern3412'와 'pattern 3214'인 쿼터의 경우 좌하점(lower left)이 되며, 'pattern1432'와 'pattern1234'인 쿼터는 우상점(upper right)이 된다.

표1. Order 변화에 따른 Hilbert 곡선의 운행규칙

Depth	Depth + 1			
	1st quarter	2nd quarter	3rd quarter	4th quarter
Pattern3412	Pattern3214	Pattern3412	Pattern3412	Pattern1432
Pattern3214	Pattern3412	Pattern3214	Pattern3214	Pattern1234
Pattern1432	Pattern1234	Pattern1432	Pattern3214	Pattern3412
Pattern1234	Pattern1432	Pattern1234	Pattern1234	Pattern3214

이러한 Hilbert 곡선의 특징을 이용하면 분할된 쿼터당 1회의 공간변환 연산을 수행하여 연산비용을 최소화하고, 쿼터의 수와 동일하게 x-value의 범위를 유지함으로서 메모리 비용 역시 최소화 할 수 있다.

3.1 분할되는 쿼터의 개수

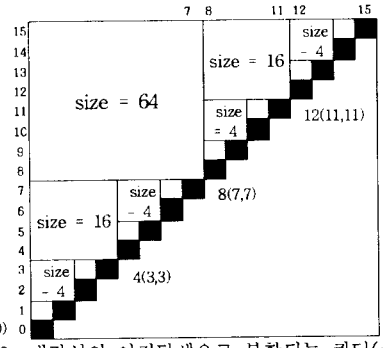


그림 2. 대각선의 이진탐색으로 분할되는 쿼터(n=16)

중간공간의 한 변의 크기를 n이라고 하였을 때, 질의영역이 가장 큰 중간공간의 위쪽 삼각형 전체를 고려해 보자. 쿼터의 분할은 중간공간의 대각선상의 점들을 이진탐색(Binary Search)의 순서로 검색하면서 각 검색된 점에서 X축(삼각형의 윗변)과 Y축(삼각형의 좌측변)에

수선을 내려 얻어진 면적으로 분할하는 것과 일치한다. 처음 얻어진 면적은 중간공간의 좌상점에서 각 수선과 교차된 점까지이며, 한번 계산된 면적은 제외된다.

이렇게 구해진 쿼터의 갯수는 대각선에 걸치지 않고 완전히 포함되는 쿼터의 수를 의미하므로 n-1개가 된다. 그리고, 대각선상의 크기가 1인 점(cell)들의 수 n개를 합한 2n-1이 분할되는 쿼터의 수 가 된다. 따라서 여과 알고리즘의 공간변환 연산 횟수는 질의영역의 모든 점들을 변환하는 경우에 비해 훨씬 줄어들게 되므로 향상된 성능을 제공해 줄 수 있다. 표2는 효율적인 여과단계 처리를 위한 접근방법들을 비교한 것이다.

표2. 접근방법 비교(T:변환횟수,M:메모리사용량)

q의 위치		변환 방법		
		순차 변환	역방향 변환	쿼터 분할
좌상점	T	$(n^2+n)/2$	$(n^2+n)/2$	2n-1
	M	$(n^2+n)/2$	3n	2n-1
좌하(우상)점	T	n	n	n
	M	n	n	n

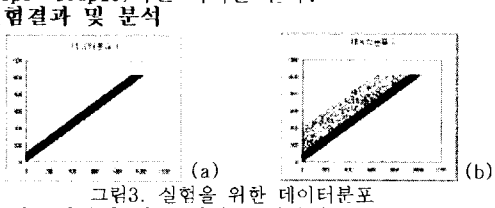
3.2 쿼터 분할 방법을 적용한 여과 알고리즘

```
Procedure Recursive_Quarter(Q:quarter, var LS:line_segmens)
          Quarter SQ;  /* sub quarter */
(step1)   IF is_in_Region(Q) THEN
(step2)     Q.segment.start = Start_x_value(Q, pattern);
(step3)     Q.segment.end = Q.start + Q.size^2-1;
(step4)     LS =LS ∪ Q.segment;
          ELSE
(step5)   IF Q.size ≥ 2 THEN
(step6)    FOR each sub quarter SQ of Q do
(step7)     SQ.size = Q.size / 2;
(step8)     SQ.pattern = next_pattern(Q.pattern, quarter_seq);
(step9)     SQ.position =
             Upper_left_of_sub_quarter(SQ, quarter_seq);
(step10)    IF Does_Touch_Region(SQ) THEN
(step11)     Recursive_Quarter(SQ, line_segments);
            END IF
            End FOR /* SQ */
          End IF
        End IF
END procedure
```

알고리즘 2. 여과 알고리즘

알고리즘2는 쿼터 분할 방법을 적용한 여과 알고리즘이다. 이 알고리즘의 처리는 분할된 쿼터가 질의영역에 속하는지를 판별하여(step1), 영역에 속하는 쿼터인 경우 쿼터 내에서 곡선의 시작점에 대한 x-value를 구하여(step2), 질의영역을 최종공간에서 연속적인 라인-세그먼트의 집합으로 변환하고(step3,4), 속하지 않는 경우 쿼터의 면적이 4이상이면 다음 order의 서브-쿼터로 분할(step5~step10)하는 처리를 한다.

4. 실험결과 및 분석



(a) (b)

그림3. 실험을 위한 데이터분포

여과 알고리즘의 성능 평가를 위하여 역방향 변환 방법과 쿼터 분할 방법을 적용한 DOT 공간조인 알고리즘과 R*-tree 공간조인 알고리즘의 실행시간을 비교하였다(표3). 순차변환 방법은 실험에서 제외하였다. 실험은 동일한 환경에서 내용이 같은 1차원 객체의 파일 R과 S로 이루어 졌으며, Sun Ultra 10/Solaris 7의 플랫폼을 사용하였다. 그림 3은 실험에 사용된 입력객체 집합을 중간공간 상에 표시한 것이다. (a)는 공간객체가 일반적으로 실세계의 전체공간에 비해 크기가 매우 작으므로, 대각선 위쪽에 좁은 띠 형태를 보이는 경우이며, (b)는 (a)보다 조금 더 일반적인 형태를 가지는 경우이다.

표3. 각 데이터분포에 대한 실행시간(단위:초)

		데이터분포 1					데이터분포 2				
		Buffer Size					Buffer Size				
입력객체수	알고리즘	8	16	32	64	128	8	16	32	64	128
10000	R*-tree	7	7	7	7	7	11	10	11	10	10
	역방향	540	540	540	539	540	579	579	580	579	579
	쿼터분할	8	8	8	8	8	12	12	11	11	12
20000	R*-tree	27	27	26	26	26	38	37	37	37	37
	역방향	1061	1061	1061	1062	1061	1128	1128	1127	1127	1126
	쿼터분할	19	19	19	18	19	27	27	27	27	26
40000	R* tree	100	100	100	98	98	137	137	137	135	134
	역방향	2100	2099	2098	2098	2097	2237	2235	2234	2235	2234
	쿼터분할	54	55	54	55	54	77	77	77	77	76
80000	R* tree	378	377	377	371	370	523	523	520	514	512
	역방향	4245	4254	4254	4255	4255	4542	4540	4540	4540	4539
	쿼터분할	188	188	188	188	187	258	259	260	260	261
100000	R*-tree	582	582	580	573	569	802	802	800	789	785
	역방향	5341	5341	5341	5342	5344	5767	5737	5740	5743	5746
	쿼터분할	282	283	284	284	285	397	397	398	399	401

표3에서 알 수 있듯이 쿼터 분할 방법을 이용한 여과 알고리즘의 성능이 역방향 변환 방법에 비해 훨씬 뛰어난 성능을 보여주고 있음은 물론, 입력 객체수가 많아질수록 R*-tree 공간조인 알고리즘의 실행시간 보다 최대 약 2배까지 우수한 성능을 보여 주고 있음을 알 수 있다.

5. 결론

DOT 공간색인 구조는 기존 데이터베이스 시스템의 B+-tree를 이용한 주색인 기법을 그대로 적용할 수 있는 장점이 있다. 본 논문에서는 DOT 공간색인 기법을 이용한 공간조인 알고리즘의 성능향상을 위하여 여과단계를 효율적으로 처리할 수 있는 방법을 제시하였다. 또한, 각각 다른 데이터분포와 공간객체 수로 이루어진 실험을 통해 잘 알려진 R*-tree를 이용한 공간조인 알고리즘보다 좋은 성능을 가지고 있음을 보였다. 현재 원공간을 2차원으로 확장하여 적용할 수 있는 DOT 공간조인 알고리즘에 대한 연구가 진행 중에 있다.

5. 참고 문헌

[1] 최익수, 윤지희, 이건배, "DOT 공간색인 기법을 이용한 공간조인 알고리즘," 한국정보과학회 데이터베이스 연구회, 동계 데이터베이스 학술대회논문집, 제14권 1호,pp.21-26,1998

[2] C. Faloutsos, and Y. Rong, "Spatial Access Methods Using Fractals : Algorithms and Performance Evaluation," UMIACS-TR-89-31, CS-TR-2214, Univ. of Maryland, pp. 1-17, 1989.

[3] C. Faloutsos, and Y. Rong, "DOT : A Spatial Access Method Using Fractals," Proc. 7th Intl. Conf. on Data Engineering, pp. 152-159, 1991.

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

[5] J. Orenstein, F. Manola, "PROBE Spatial Data Modeling and Query Processing in an Image Database Applications," IEEE Trans. on Software Engineering, Vol 14, No. 5, 1988.

[6] 윤지희, "상용 ORDB를 하부구조로 갖는 객체관계형 지리정보 시스템의 설계 및 구현," 한국 GIS학회지, 제5권 1호, pp.77-88,1997.