다중 공간 조인에서 간접 술어의 활용
(Using Indirect Predicates in Multi-way Spatial Joins)

박 호 현 † 정 진 완 ††
(Ho-Hyun Park) (Chin-Wan Chung)

요 약 공간 조인은 일반적으로 많은 처리 시간을 필요로 하므로 공간 조인에 대한 처리 시간을 단축하기 위해 많은 연구가 있었다. M-다중 공간 조인에서 M 개의 R-트리를 동시에 탐색하는 M-다중 R-트리 조인(MRJ)도 그 중의 하나이다. 본 논문에서는 M-다중 공간 조인에서 간접 술어(indirect predicate) 개념을 소개한다. 다중 공간 조인에서 간접 술어란 질의 조건문에 직접 나타나는 술어가 아니라 이들 직접 술어들로부터 간접적으로 유도되는 술어를 말한다. 간접 술어 개념을 M-다중 R-트리 조인에 적용함으로서 다중 공간 조인의 성능을 향상시킬 수 있다. 우리는 이러한 간접 술어를 이용한 다중 공간 조인 처리 방법을 간접 술어 여과(IPF)라 부른다. 가공 데이타와 실제 지도 데이타를 이용한 실험을 통하여 우리는 IPF 기법이 M-다중 R-트리 조인의 성능을 상당히 개선한다는 것을 보였다.

키워드 : 공간 데이타베이스, 공간 조인, M-다중 R-트리 조인, 간접 술어

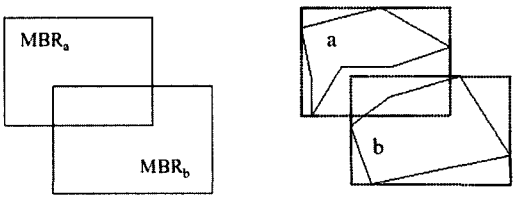
Abstract Since spatial join processing consumes much time, several algorithms have been proposed to improve spatial join performance. The M-way R-tree join (MRJ) is a join algorithm which synchronously traverses M R-trees in the M-way spatial join. In this paper, we introduce indirect predicates which do not directly come from the multi-way join conditions but are indirectly derived from them. By applying the concept of indirect predicates to MRJ, we improve the performance of MRJ. We call such a multi-way R-tree join algorithm using indirect predicates indirect predicate filtering (IPF). Through experiments using synthetic data and real data, we show that IPF significantly

1. 서 론

최근에 지리정보시스템(GIS), CAD, 원격 탐사 등과 같은 공간 정보를 이용하는 응용 분야가 늘어나면서 공간 데이타베이스 시스템에 대한 연구가 활발히 진행되고 있다. 공간 조인이란 여러 공간 객체 집합을 대상으로 조인 조건(예, 교차 질의)을 만족하는 공간 객체들의 조합을 찾아내는 질의이다. 공간 조인은 공간 데이타베이스 시스템에서 흔히 발생할 수 있는 질의 형태이나 공간 데이타가 일반적으로 복잡하고 양이 방대하다는 사실을 미루어 볼 때 많은 처리 시간을 필요로 한다. 따라서 전체 질의 처리 시간을 줄이기 위해 공간 조인은 주로 *여과 단계(filter step)*와 *정제 단계(refinement step)*로 불리는 2 단계로 처리되어 왔다[1,2].

여과 단계에서는 복잡한 공간 객체를 최소 경계 사각형(MBR: Minimum Bounding Rectangle)과 같은 단순한 형태로 근사시킨 후 그 MBR들에 대해 공간 조인을 실시한다. 여과 단계의 결과는 정확한 결과치가 아니라 결과치의 잉여 집합(super set)을 구성하는 *후보 객체(candidate objects)*들이다. 정제 단계에서는 이들 후보 객체들에 대해 정확한 계산 기하 알고리즘 등을 적용하여 실제로 조인 조건을 만족하는 객체들만을 골라낸다.

그림 1은 조인 조건이 겹침(intersection)인 공간 질의 처리 과정의 예를 보여준다. 그림 1(a)에서 보는 바와 같이 여과 단계에서는 공간 객체들의 근사치인 최소 경계 사각형(MBR: Minimum Bounding Rectangle)에 대



(a) 여과 단계 (b) 정제 단계

그림 1 공간 조인의 2단계 처리

· 이 논문은 2003년도 중앙대학교 학술연구비 지원에 의한 것임.
† 정 회 원 : 중앙대학교 전자전기공학부 교수
hohyun@cau.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 2003년 2월 4일
심사완료 : 2003년 8월 4일

해 조인 조건를 체크하여 질의 조건을 만족하는 후보 객체들만 우선 찾아낸다. 그림 1(a)는 여과 단계를 통과한 두 공간 후보 객체 쌍을 보여 준다. 정제 단계에서는 이들 후보 객체들에 대해 정확한 계산 기하 알고리즘 등을 적용하여 이들이 실제로 조인 조건을 만족하는지 검사한다. 그림 1-(b)는 후보 객체 a와 b 쌍에 대해 정확한 겹침 알고리즘을 적용하면 a와 b 쌍은 거짓 겹침(false intersection)이 되어 최종 조인 결과가 될 수 없다는 것을 보여준다.

M-다중 공간 조인은 M 개의 공간 객체 집합에 대해 M-1 개 이상의 공간 술어를 사용해서 그 술어들을 만족하는 M-다중 공간 객체 조합을 생성한다. 즉, M 개의 공간 객체 집합 R_1, R_2, ..., R_M이 있고 Q_{ij}는 R_i와 R_j 사이의 공간 술어라고 가정할 때, M-다중 공간 조인은 아래의 식으로 표현된다.

$\{<r_1, r_2, ..., r_M> | \forall i,j: r_i \in R_i, r_j \in R_j \wedge r_i\ Q_{ij}\ r_j\}$ (1)

3중 공간 조인의 예로는 "도로에 인접해 있으면서 행정 구역 경계선과 교차하는 모든 빌딩을 찾아라"와 같은 것이 있다. M-다중 공간 조인은 질의 그래프 형태로 모델링 될 수 있다. 그 때 노드(node)는 공간 객체 집합을 선(edge)은 공간 술어를 나타낸다.

최근에 다중 공간 조인에 대한 연구가 많이 있었다[3-6]. M-다중 공간 조인을 처리하기 위한 한가지 방법은 2중 조인의 연속으로 처리하는 것이고[3], 또 다른 방법은 조인에 참가하는 모든 공간 객체 집합들에 대해 R-트리가 있을 경우, 그 M 개의 R-트리를 동시에 탐색하는 *M-다중 R-트리 조인(MRJ: M-way R-tree Join)*이라고 하는 것이다[4-6]. MRJ는 질의 그래프가 밀접하고 공간 객체의 밀도가 높고 M 값이 작을 경우에 2중 조인의 연속 처리 보다 특별히 효율적인 것으로 알려져 있다[5]. 게다가 MRJ는 2중 조인의 연속 처리에 비해 아래와 같은 장점을 가지고 있다. 이들 중 세 번째 장점은 대한 자세한 설명은 2.2 절에서 언급될 것이다.

1. 2중 조인의 연속에서 필연적으로 나타나는 중간 결과를 생성하지 않는다.
2. 프로그램이 시작하자마자 바로 결과가 나오기 시작하므로 대화형 질의 형태나 인터넷 정보 검색을 위한 브라우징 환경에 적합하다[4].
3. 일부 공간 객체 쌍에 대한 불필요한 정제 단계 연산을 피할 수 있다.

최근에 MRJ를 위한 몇 가지 알고리즘들이 개발되었다[4-6]. 이들은 모두 다중 공간 조인의 여과 단계 알고리즘들이다. 본 논문에서도 다중 공간 조인의 여과 단계 알고리즘에 대해 기술한다. 본 논문에서는 MRJ에 *간접 술어(indirect predicate)* 개념을 도입한다. 다중 공간 조인에서 간접 술어란 질의 조건문에 직접 나타나는 술어가 아니라 이들 직접 술어들로부터 간접적으로 유도되는 술어를 말한다. 간접 술어를 MRJ에 적용함으로서 여과 단계에서 MBR 간의 거짓 겹침(false intersection)을 빨리 제거할 수 있고 MRJ 처리 성능을 향상시킬 수 있다. 우리는 이러한 간접 술어를 이용한 다중 공간 조인 처리 방법을 *간접 술어 여과(IPF: Indirect Predicate Filtering)*라 부른다. 본 논문에서는 IPF 기법을 확장하여 R-트리의 *도메인 최대값(DMAX: Domain Maximum), 노드 최대값(NMAX: Node Maximum), 엔트리 최대값(EMAX: Entry Maximum)*을 이용하는 IPF 기법을 제안한다. 우리는 IPF의 성능을 측정하기 위해 가공 데이타와 TIGER[7]라는 미국의 실제 지도 데이타를 이용하여 실험을 하였다. 실험 결과 우리는 IPF 기법을 이용한 MRJ 처리 방법이 간접 술어를 쓰지 않는 기존의 방법 보다 조인 처리 시간을 훨씬 단축시킬 수 있다는 사실을 밝혔다. 그리고 IPF 기법들 간에는 EMAX가 DMAX와 NMAX 보다 우수한 성능을 보였다.

본 논문의 나머지 부분은 아래와 같이 구성되어 있다. 제2절은 본 논문의 기반이 되는 2중 R-트리 조인과 M-다중 R-트리 조인(MRJ)에 대해 간단히 소개한다. 제3절에서 우리는 간접 술어의 기본 개념과 MRJ에서 IPF의 기본적인 활용 방법을 소개하고, 제4절에서는 DMAX/NMAX/EMAX와 같은 IPF 기법의 여러 가지 변형과 그에 대한 R-트리 구조를 설명한다. 제5절에서는 실험을 통하여 IPF의 성능을 분석하였고, 마지막으로 제6절에서 우리는 본 논문에 대한 결론을 맺는다.

2. 관련 연구

2.1 R-트리 조인

R-트리는 인접한 공간 객체들의 MBR을 묶어서 하나의 노드를 만들고, 인접한 노드들의 MBR을 다시 묶어서 부모 노드를 만드는 식으로 해서 구성된다. R-트리는 마치 B+-트리가 다차원으로 확장된 형태로 Guttman에 의해 제안되었다[8]. 그림 2는 공간 객체들의 MBR과 그 위에 구축된 R-트리의 예를 보여준다.

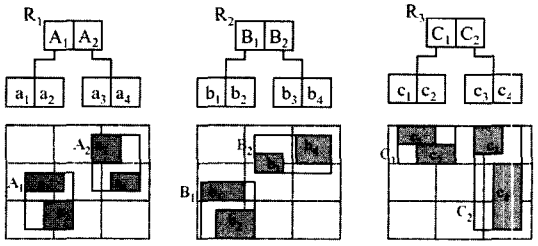


그림 2 공간 객체의 MBR과 R-트리

Guttman의 제안 이후 몇몇 연구자들에 의해 R+-트리[9], R*-트리[10]와 같은 R-트리의 변형들이 제안되었다. 우리는 이들을 모두 R-트리 가족이라 부른다. R-트리 가족은 여러 공간 색인들 중 가장 많이 사용되고 있고 공간 데이타베이스에 대한 많은 연구가 R-트리를 기반으로 하고 있다[11].

공간 조인 대상이 되는 두 객체 집합이 모두 R-트리를 가지고 있을 경우, 두개의 R-트리를 동시에 탐색하는 공간 조인 방법이 제안되었다[11]. 기본적인 동작은 아래와 같다. 우선 두 R-트리의 루트 노드를 읽어 들인 후 각각의 루트 노드 내에 있는 엔트리들 간의 MBR이 서로 겹치는지 체크한다. 루트 노드 검사에서 겹침이 있는 엔트리 쌍들에 대해 그 자식 노드들을 읽어 들인 후 그 자식 노드 내에 있는 엔트리들 간의 MBR에 대해서도 계속해서 겹침 여부를 검사한다. 이런 식으로 해서 동시 탐색이 리프 노드에 도달하면 리프 노드 내에 있는 엔트리들 간에 MBR이 서로 겹치는 쌍은 조인 결과로 출력되고 탐색은 부모 노드 단계로 복귀한다.

그림 2에 있는 두개의 R-트리 R1과 R2 간의 조인의 예를 생각해 보자. 우선 두 루트 노드에 있는 엔트리 집합 {A1, A2}와 {B1, B2} 간의 MBR 겹침이 체크되어야 한다. 루트 단계에서 <A1, B1>과 <A2, B2> 쌍이 서로 겹치므로 이들 쌍에 대해 자식 노드들을 동시 방문한다. 두 노드 A1과 B1 내에 있는 엔트리 집합 {a1, a2}와 {b1, b2} 간의 체크에서 <a1, b1>과 <a2, b2>가 서로 겹치므로 질의 결과로 출력된다. 동시 탐색은 다시 부모 노드 단계로 복귀하여 이번에는 노드 A2와 B2가 방문된다. 노드 A2와 B2 내에 있는 엔트리 집합 {a3, a4}와 {b3, b4} 간의 체크에서는 <a3, b4>가 서로 겹치므로 질의 결과로 출력된다.

본 논문에서는 이러한 조인 방법을 *2중 R-트리 조인(2-way R-tree Join)* 또는 단순히 *R-트리 조인(RJ: R-tree Join)*이라 부르며 그 알고리즘은 아래와 같다.

```
PROCEDURE RJ (Rtree_Node R, S)
BEGIN
  FOR all ES∈S DO
    FOR all ER∈R with ER.rect∩ES.rect ≠∅ DO
      IF R is a leaf page THEN // S is also a
        leaf page
      output (ER, ES)
        ELSE
        ReadPage(ER.ref)
       ReadPage(ES.ref)
       RJ(ER.ref, ES.ref)
      ENDIF
    ENDFOR
  ENDFOR
END
```

RJ에서 처리 시간을 줄이기 위해 *검색 공간 제한(search space restriction)*과 *평면 쓸기(plane sweep)*라는 방법이 사용되었다[11]. 검색 공간 제한은 양쪽 노드 내에 있는 엔트리들 간의 MBR 겹침을 체크하기 전에 각각의 노드 내에 있는 엔트리들에 대해 상대방 노드의 모든 엔트리를 둘러싸는 노드 MBR과 겹침을 체크하여 겹치지 않는 엔트리들은 미리 제거한다. 위의 예에서 노드 A2와 B2 간의 조인 시 노드 A2에 있는 엔트리 a3, a4 중에서 a4는 상대방 노드 B2의 MBR과 겹침이 없다. 따라서 a4는 B2 내에 있는 어느 엔트리와도 겹칠 수 없기 때문에 검사에서 제외될 수 있다.

평면 쓸기는 각각의 노드 내의 엔트리 MBR들을 x-축(또는 y-축)으로 정렬한 후 청소 라인(sweep line)을 따라 가면서 다른 축에 대해 엔트리들 간 겹침을 체크한다. 위의 예에서 노드 A1과 B1 간의 조인 시 각각의 노드 내에 있는 엔트리들을 x-축으로 정렬하면 (a1, a2)와 (b1, b2)가 된다. 그런 후 두 정렬 순서를 하나의 순서로 합치면 (b1, a1, b2, a2) 순서가 청소 라인이 된다. 청소 라인을 따라가면서 현재 청소 라인 뒤에 있으면서 x-축과 겹치는 상대방 엔트리에 대해 y-축 겹침을 체크한다. 위의 예에서는 b1에서는 상대방 엔트리 {a1, a2}와, a1에서는 {b2}와, b2에서는 {a2}와 y-축에 대해 겹침을 체크하고, a2에서는 더 이상 청소 라인에서 뒤에 있는 엔트리가 없으므로 평면 쓸기는 종료된다.

두 공간 객체 집합에 R-트리가 있을 경우 R-트리 조인은 가장 효율적인 조인 방법으로 알려져 있다[3,12,13].

2.2 M-다중 R-트리 조인

R-트리 조인(RJ)을 다중 공간 조인으로 확장한 M-다중 R-트리 조인(MRJ)이 최근에 개발되었다[5,6,14]. RJ와 마찬가지로 MRJ의 기본 동작도 M 개의 R-트리의 동시 탐색이다. 이 알고리즘도 M 개의 R-트리의 루트 노드로부터 시작한다. 각각의 노드 내에 있는 엔트리들로부터 나올 수 있는 모든 M-조합에 대해(이 조합을 *엔트리 튜플(entry tuple)*이라 한다) 질의에 정의된 모든 술어들을 적용한다. 술어들을 모두 만족하는 엔트리 조합에 대해서는 다음 2가지 경우 중 하나로 처리된다.

1. R-트리 노드의 조합(이 조합을 노드 *튜플(node tuple)*이라 부른다)이 중간 노드 단계에 있으면 엔트리 조합에 의해 포인팅 되는 자식 노드 조합에 대해 재귀적으로 본 알고리즘을 호출한다.
2. R-트리 노드의 조합이 리프 노드 단계에 있으면 그 엔트리 조합은 다중 조인의 결과로 출력되고 다음 엔트리 조합으로 넘어 간다.

만약 엔트리 조합이 하나의 술어라도 만족하지 않으면 그 엔트리 조합은 더 이상 처리되지 않고 제거

(prune)된다. 이런 식으로 하나의 노드 조합 내에 있는 모든 엔트리 조합이 모두 처리되면 알고리즘은 부모 노드 조합으로 복귀한다.

MRJ의 예를 다시 그림 2의 R-트리를 가지고 설명한다. 조인 술어가 "R1 intersect R2 and R2 intersect R3"인 3중 공간 조인을 생각해 보자. 루트 노드 단계에서 나올 수 있는 8 가지 엔트리 조합 중 조인 술어를 만족하는 조합은 <A2, B2, C1>과 <A2, B2, C2>이다. MRJ는 엔트리 조합 <A2, B2, C1>이 가리키는 자식 노드들을 읽어 와서 그 노드들 내의 엔트리들 간에 조인 술어를 체크한다. 위의 예제에서는 노드 A2, B2, C1 내에 있는 엔트리들 간에는 조인 술어를 만족하는 조합이 없다. 알고리즘은 부모 노드로 복귀하여 이번에는 <A2, B2, C2>가 가리키는 자식 노드들을 읽어 와서 엔트리들 간에 조인 술어를 체크한다. 엔트리 조합 <a3, b4, c3>가 조인 술어를 만족하므로 조인 결과로 출력된다.

위의 예제에서 <A2, B2, C1>는 루트 단계에서는 조인 술어를 만족하나 그 아래의 자식 노드들은 조인 술어를 만족하는 엔트리 조합을 하나도 생성하지 못한다. 우리는 이와 같이 리프 노드 단계에서 질의 결과를 생성하지 못하면서 중간 노드에서는 겹침으로 나타나는 현상을 *거짓 겹침(false intersection)*이라 한다. 거짓 겹침은 조인 술어의 수가 적을수록 많이 나타난다. 본 논문에서는 간접 술어 개념을 이용하여 MRJ에서 거짓 겹침을 미리 찾아 제거하는 간접 술어 여과(IPF) 기법을 제안한다.

MRJ에 대한 알고리즘은 아래에 나와 있다. 아래의 알고리즘에서 SpaceRestriction과 FindTuples은 각각 RJ의 검색 공간 제한과 평면 쓸기 휴리스틱을 M-다중 공간 조인으로 일반화시킨 것에 해당된다.

```
PROCEDURE MRJ (Query_graph Q[ ][ ], Rtree_Nodes N[ ])
BEGIN
  IF NOT SpaceRestriction(Q[ ][ ], N[ ]) THEN
    RETURN
  FOR each τ∈FindTuples(Q[ ][ ], N[ ]) DO
    IF N[ ] are leaf nodes THEN
      output ( τ )
    ELSE // all tree heights are equal
      MRJ (Q[ ][ ], τ.ref[ ])
    ENDIF
  ENDFOR
END
```

SpaceRestriction의 성능을 향상하기 위한 방법으로는 *정적 변수 순서화(SVO: Static Variable Ordering)* [4], *공간 제한 순서화(SRO: Space Restriction Ordering)* [6]와 같은 알고리즘들이 제안되었다. 한 노드 조합에 대한 MRJ 처리 시 그 노드 조합에 속하는 R-트리 노드들을 어떤 순서로 읽어 들이는 것이 효율적인지에 대한 고려가 필요하다.

SVO는 질의 그래프에서 각 노드들의 차수(degree)가 감소하는 순서에 따라 노드들을 처리한다. 차수가 같으면 질의 변수(query variable) 순서 상 먼저 나오는 노드를 읽어 들인다. 그 이유는 노드의 차수가 높으면 그 노드에 적용되는 술어의 개수가 많다는 것을 나타내므로, 높은 차수의 노드가 검색 공간을 빨리 줄여주기 때문이다. SRO는 자신과 인접 노드들 간의 공통 겹침 영역(common intersection area)이 작은 노드를 우선적으로 읽어 들인다. 만약 공통 겹침 영역이 0인 경우는 인접 노드들 간의 거리가 가장 긴 노드를 먼저 읽어 들인다. SRO의 배경에는 공통 겹침 영역이 작거나 0인 경우에 검색 공간이 작아지거나 공집합이 될 가능성이 높기 때문이다. 특히 검색 공간이 공집합인 경우는 그 노드 조합은 거짓 겹침이 되어 다른 노드들은 읽어 올 필요가 없게 된다.

그림 3(b)는 4중 공간 조인 "A intersect B and B intersect C and C intersect D"에서 R-트리 노드들 간의 MBR 겹침의 한 예를 보여 준다. 여기서 각 노드 A, B, C, D에 대한 공통 겹침 영역은 각각 A.rect ∩ B.rect, A.rect ∩ B.rect ∩ C.rect, B.rect ∩ C.rect ∩ D.rect, C.rect ∩ D.rect 이다. 노드 B와 C는 모두 공통 겹침 영역이 0이다. 그러나 노드 C에서의 인접 노드 간 거리, 즉 B와 D 사이의 거리, 가 노드 B에서의 인접 노드 간 거리, 즉 A와 C 사이의 거리, 보다 크므로 SRO는 노드 C를 선택한다. 노드 C에서 앞 절에서 설명한 검색 공간 제한(search space restriction)을 적용하면 B와 D의 MBR과 동시에 겹치는 엔트리는 노드 C 내에는 없으므로 검색 공간이 공집합이 된다. 즉 그림 3(b)는 거짓 겹침이 되므로 다른 노드들은 읽어 올 필요가 없다. 그러나 그림 3(b)에 SVO를 적용하면 노드 B와 C는 차수가 2이고 노드 A와 D는 차수가 1이다. SVO는 차수가 2인 두 노드 중 질의 변수 순서 상 먼저 나오는 노드 B를 읽어 들인다. 그러나 노드 B에는 A와 C의 MBR과 동시에 겹치는 엔트리가 존재하므로 검색 공간 제한을 적용한 후의 검색 공간은 공집합이 아니다. 다음으로 SVO는 노드 C를 읽어 들이게 되고 그때서야 이 노드 조합이 거짓 겹침이라는 사실을 알게 된다. 따라서 그림 3(b)에서 SVO는 SRO 보다 한 노드를 더 읽게 된다. [15]에 의하면 일반적으로 SRO가 SVO 보다 더 좋은 성능을 보이는 것으로 나타나 있다.

FindTuples의 성능 향상을 위해서 *다단계 전방향 검사(MFC: Multi-level Forward Checking)*[14], *평면 쓸기 및 전방향 검사(PSFC: Plane Sweep and Forward Checking)*[4]와 같은 알고리즘들이 제안되었다.

MFC는 M 개의 R-트리를 동시 탐색하는 최초의 MRJ 알고리즘이다. 각 R-트리 단계마다 MFC는 *전방향 검사(FC: forward checking)*를 실시한다.

FC는 M 개의 질의 변수(query variable) v_j, $0 \le j \le M-1$에 대해 M^2 개의 도메인으로 구성된 테이블, D[M][M]을 메인 메모리에 가지고 있다. FC는 질의 변수와 공간 술어들 간의 관계를 이용해 각 질의 변수에 대해 전진하면서 단계별로 질의 변수의 도메인을 줄여 나가는 방법으로 다음과 같이 동작한다. 처음에는 각 질의 변수의 도메인이 R-트리의 노드로 초기화 된다. 즉, 각각의 j, $0 \le j \le M-1$에 대해 D[0][j]는 R-트리 노드 N[j]로 초기화 된다. 그리고 D[0][0]로부터 하나의 값, t[0]를 꺼내 변수 v_0에 대입한다. 그런 후 FC는 v_0와 공간 술어 관계에 있는 엔트리들로 v_0를 제외한 나머지 변수들 v_j, $1 \le j \le M-1$ 의 도메인을 새로이 구성한다. 즉, 아래의 기준에 의해서 D[0][1], ..., D[0][M-1]로부터 새로운 도메인 D[1][1], ..., D[1][M-1]을 계산한다:

1. 각각의 j, $1 \le j \le M-1$에 대해 v_j가 v_0와 술어가 있으면 D[0][j]에 있는 엔트리들 중 t[0]와 겹치는 것들만 D[1][j]에 복사된다.
2. v_j가 v_0와 술어가 없으면 D[0][j]에 있는 모든 엔트리들은 D[1][j]에 복사된다.

새로 계산된 도메인 D[1][1], ..., D[1][M-1]에 공집합이 없으면 FC는 전진하여 다음 변수 v1에 D[1][1]로부터 하나의 값, t[1]이 할당된다. 그리고 FC는 새로운 도메인 D[1][1], ..., D[1][M-1]에 대해 계속해서 적용된다. 일반적으로 변수 v_i에 하나의 값 t[i]가 할당될 때, 남아 있는 v_j, $i+1 \le j \le M-1$에 대해서 D[i+1][j]는 다음과 같이 계산 된다: Q_{ij}=TRUE 이면, $D[i+1][j] = \{E_k | E_k \in D[i][j] \wedge Q_{ij}(t[i], E_k)=\text{TRUE}\}$. 그렇지 않으면, D[i+1][j] = D[i][j]. 일반적으로 t[i]가 v_i의 현재 값이면, D[i+1][j]는 Q_{ij}와 t[i]에 대해 유효한 D[i][j]의 부분집합이 된다.

이런 식으로 질의 변수에 값이 할당되는 것을 실체화(instantiation)라 한다. 각 질의 변수가 실체화(instantiation)될 때, 아직 실체화 되지 않은 미래 변수(future variable)들의 도메인은 계속해서 줄어든다. 미래 변수들 중 하나 이상의 도메인이 공집합이 되면 현재 변수(current variable)의 값은 결과 값이 되지 못하므로 다음 값이 현재 변수에 할당된다. 최종적으로 FC는 마지막 변수에 값이 할당될 때 결과 값을 출력한다. 현재 변수의 도메인을 모두 체크했으면 FC는 이전 변수로 역행(backtrack) 한다.

MFC에서는 평면 쓸기(plane sweep) 알고리즘이 사용되지 않았다. 그러나 2중 R-트리 조인에서 보았듯이 평면 쓸기는 사각형 간의 겹침 문제(rectangle intersection problem)를 해결하는데 아주 효율적인 알고리즘으로 알려져 있다[11]. PSFC는 평면 쓸기와 전방향 검사(forward checking)를 결합한 알고리즘으로, 각 노드 엔트리를 x-축으로 정렬한 후 첫 질의 변수의 실체화는 평면 쓸기로 하고 나머지 변수들에 대해서는 전방향 검사(forward checking)를 실시한다.

각 엔트리들의 MBR이 (xl, xh, yl, yh)라고 하면 PSFC는 먼저 각각의 노드 내에서 엔트리들을 xl-값에 따라 정렬한다. 그런 다음 여러 노드에 걸쳐 정렬된 모든 엔트리들에 대해 청소 라인(sweep line)이 앞으로 지나간다. 즉 청소 라인은 노드 조합에 있는 노드들 간에 지그재그로 옮겨 다닐 수 있다. 그러기 위해서 각 노드 별로 다음 청소 라인 엔트리를 기억하기 위해 포인터 배열인 head[ ]가 유지된다. 현재 청소 라인이 속해 있는 노드를 청소 노드(sweeping node)라 한다. 청소 라인을 지나면서 PSFC는 청소 노드 N[i]를 첫 번째 변수 vi에 할당한 후, 청소 노드가 아닌 다른 노드에 있는 엔트리들에 대해 전방향 검사를 실시한다. 나머지 변수들($\forall v_j$, $j \ne i$)에 대해서 그들의 도메인은 다음과 같이 할당된다: 각각의 j, $j \ne i$에 대해 Q_{ij}=TRUE이면, 노드 N[j]에서 head[j] 이후의 모든 엔트리에 대해 청소 라인 엔트리와 겹치면 도메인 D[1][j], 즉 $\forall E_k \in N[j]$, $k > head[j]$, s.t. $E_{head[j]}.xl \le E_k.xh \wedge$ y-intersect(E_k,$E_{head[j]}$)에 포함된다. 나머지 변수들의 실체화는 전방향 검사(forward checking)와 동일하다. 단 평면 쓸기의 특성상 청소 라인의 뒤에 있는 엔트리들에 대해서만 y-intersect를 체크하면 된다. [4, 15]에 따르면 평면 쓸기와 전방향 검사를 모두 이용하는 PSFC가 MFC보다 일반적으로 좋은 성능을 보이는 것으로 나타나 있다.

MRJ는 질의 그래프가 밀접하고 공간 객체의 밀도가 높고 M 값이 작을 경우에 2중 조인의 연속 처리 보다 특별히 효율적인 것으로 알려져 있다[5]. 본 논문에서는 여러 MRJ 알고리즘들 중 가장 효율적인 것으로 알려진 SRO-PSFC 조합을 기본 알고리즘으로 채택하여, 이것을 기반으로 IPF 기법을 구현 및 실험하였다. 이들 알고리즘에 대한 보다 자세한 설명 및 성능 비교는 [15]을 참조하기 바란다.

제1절 서론에서 우리는 MRJ는 2중 조인의 연속에 비해 불필요한 정제 연산을 줄일 수 있다고 하였다. 다시 위의 3중 공간 조인의 예를 생각해 보자. 위의 3중 공간 조인이 2중 조인의 연속으로 처리되고, 질의 최적화기에 의해 처리 순서가 ((R1, R2), R3)로 결정되었다고 하자. 2.1 절의 예에 의하면 <a1,b1>과 <a2,b2>가 R1과 R2 간의 조인의 여과 단계 결과이므로 이들 간에 정제 단계 연산이 이루어 져야 한다. 그러나 그림 2를 보면 b1과 b2는 R3에 있는 어느 객체와도 겹칠 수 없으므로 중간 결과 <a1,b1>과 <a2,b2>는 결코 3중 공간 조인

의 결과에 포함될 수 없고 그들 간의 정제 단계 연산은 불필요하게 수행된 것이다. MRJ는 여과 단계에서 이들을 모두 제거하므로 이러한 불필요한 정제 단계 연산을 막을 수 있다.

3. 간접 술어를 이용한 M-다중 R-트리 조인

M-다중 공간 조인에서 가능한 최대 술어의 수는 M*(M-1)/2 이다. 즉, 모든 공간 객체 집합들 간에 조인 술어가 있는 경우이다. 우리는 그러한 조인을 *완전 조인(complete join)*이라 한다. 조인이 완전 조인이 아닌 경우, 즉 조인 술어의 수가 M*(M-1)/2 보다 적은 경우를 *불완전 조인(incomplete join)*이라 한다.

2.2 절에서 지적한 바와 같이 MRJ는 중간 노드 단계에서 많은 거짓 겹침(false intersection)을 유발한다. 특히 불완전 조인에서 거짓 겹침이 더욱 많이 나타난다. MRJ 처리 도중 특정 노드 조합을 방문하기 전에 그 노드 조합이 거짓 겹침이라는 사실을 미리 알 수 있으면, 그 노드 조합은 방문하지 않아도 되므로 많은 처리 시간을 절약할 수 있다. 본 절에서는 MRJ의 중간 노드 단계에서 거짓 겸침을 미리 찾아 낼 수 있는 한 가지 방법을 제안한다.

3.1 간접 술어

4중 공간 조인의 한 예로 "A intersect B and B intersect C and C intersect D"라는 질의를 생각해 보자. 그림 3(a)는 조인 결과에 대한 MBR 겹침의 한 예를 보여준다.

위의 질의 예에서 A와 C 간, B와 D 간, A와 D 간에는 조인 술어가 없으므로 이들 간에는 아무런 관계가 없는 것처럼 보인다. 그러나 b_x와 c_x가 각각 공간 객체 b와 c에 대한 MBR의 x-길이라고 하면, 질의 결과 <a,b,c,d>에 대해서 x-축으로 아래와 같은 관계가 성립한다.

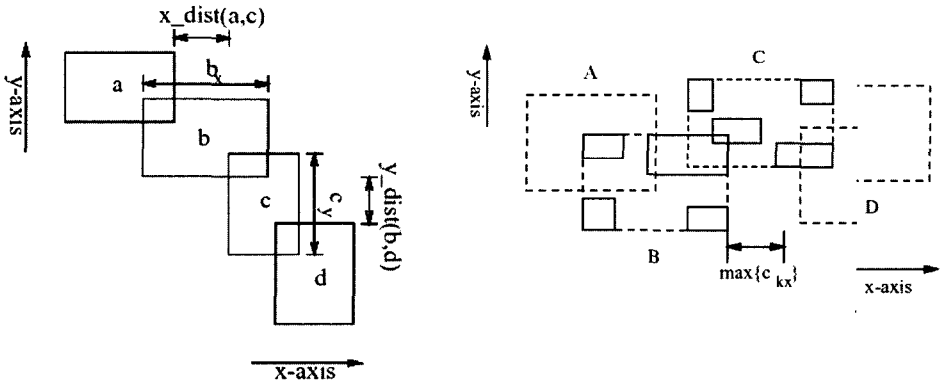
$$x\_dist(a,c) \le b_x,\ x\_dist(b,d) \le c_x,\ x\_dist(a,d) \le b_x + c_x \quad (2)$$

y-축에 대해서도 같은 조건이 성립한다. b_x, c_x 등은 공간 객체 별로 가지고 있는 값이므로 MRJ 처리 시 중간 노드 단계에서는 그 값을 알 수가 없다. 그러나 공간 객체 집합 별로 x-길이의 최대값은 질의 최적화기의 카탈로그 정보로 미리 그 값을 가질 수 있다. 그리고 수식 (2)를 만족하는 질의 결과 <a,b,c,d>에 대해서는 아래의 관계도 성립한다.

$$x\_dist(a,c) \le \max\{b_{jx} | b_j \in dom(B)\},$$
$$x\_dist(b,d) \le \max\{c_{kx} | c_k \in dom(C)\},$$
$$x\_dist(a,d) \le \max\{b_{jx} | b_j \in dom(B)\} + \max\{c_{kx} | c_k \in dom(C)\} \quad (3)$$

위의 수식에서 dom(B)란 질의 변수 B의 도메인, 즉 공간 객체 집합, 을 나타낸다. y-축에 대해서도 같은 조건이 성립한다. R-트리의 중간 노드 단계에서는 MBR의 x-길이 또는 y-길이가 도메인 내에 있는 공간 객체들의 x-길이 또는 y-길이의 최대값 보다 훨씬 클 수가 있다. 따라서 위의 수식 (3)은 MRJ 처리 시 중간 노드 단계에서 거짓 겹침을 미리 찾아 내는데 활용될 수 있다. 그림 3(b)는 위의 질의 예에 대한 R-트리 노드들 간의 MBR 겹침의 한 예를 보여준다. 만약 $x\_dist(A,C) > \max\{b_{jx}\}$, $x\_dist(B,D) > \max\{c_{kx}\}$ 또는 $x\_dist(A,D) > \max\{b_{jx}\} + \max\{c_{kx}\}$ 라면, 이 노드 조합과 그 후손 노드 조합은 더 이상 내려가 봐도 질의 결과를 낼 수가 없다. 따라서 그림 3(b)와 같은 노드 조합은 그 부모 노드 단계에서 엔트리 조합 만 체크해 보고 탐색에서 제거(prune)할 수가 있다.

위의 예에서 "A intersect B", "B intersect C" 그리고 "C intersect D" 처럼 질의문에 직접 나타나는 술어



(a) 조인 결과에서 MBR 겹침 (b) R-트리 노드들 간의 MBR 겹침

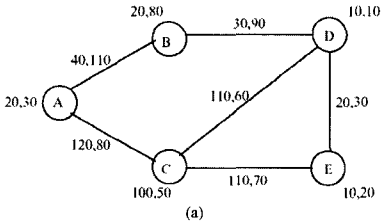
그림 3 4중 공간 조인에서 MBR 겹침

를 *직접 술어(direct predicates)*라 하고, 수식 (3)과 같이 직접 술어로 부터 유도되어 MRJ 처리에 부가적으로 활용될 수 있는 술어를 *간접 술어(indirect predicates)*라 부른다. 또한 직접 술어 뿐만 아니라 간접 술어까지 활용하여 중간 노드 단계에서 거짓 겹침을 미리 체크하여 이를 탐색에서 제거하는 기법을 *간접 술어 여과(IPF: Indirect Predicate Filtering)*이라 한다.

3.2 간접 술어 경로 및 길이

그림 3(b)에서 간접 술어를 이루는 쌍 AC, BD, AD에 대한 경로 ABC, BCD, ABCD를 *간접 술어 경로(ipp: indirect predicate paths)*라 하고, 그 경로의 x-길이인 $\max\{b_{jx}\}$, $\max\{c_{kx}\}$, $\max\{b_{jx}\}+\max\{c_{kx}\}$를 *간접 술어 x-경로 길이(x_ippl: indirect predicate x_path lengths)*라 한다. *간접 술어 y-경로 길이(y_ippl: indirect predicate y_path lengths)*도 유사하게 정의될 수 있다. 그림 3에서는 각각의 간접 술어 쌍에 대해 간접 술어 경로가 하나만 존재하므로 간접 술어 경로와 그 길이를 구하는 것은 어려운 일이 아니다. 그러나 일반적인 M-다중 공간 조인에서는 하나의 간접 술어 쌍에 대해 여러 개의 간접 술어 경로가 존재할 수 있으므로, 간접 술어 경로와 그 길이를 구할 수 있는 보다 체계적인 방법이 필요하다.

먼저 노드(node)가 공간 객체 집합을 나타내고 선(edge)이 직접 술어를 나타내는 질의 그래프(query graph)를 그린다. 그런 다음 각 노드에 가중치(weights)를 할당한다. 여기서 노드의 가중치는 그 노드가 나타내는 공간 객체 집합에서 x-길이 최대값(x_max)과 y-길이 최대값(y_max)이다. 간접 술어를 이루는 노드 쌍 간에는 여래 개의 경로가 있을 수 있으므로, 우리는 최단 경로 알고리즘(shortest path algorithm)[16]을 이용하여 ipp와 ippl을 구한다. 노드 쌍 간의 최단 경로를 구하기 위해서는 노드 가중치(node weights)가 아닌 선



(a)

pairs	x_ipp	x_ippl	y_ipp	y_ippl
AD	ABD	20	ACD	50
BC	BDC	10	BDC	10
BE	BDE	10	BDE	10
AE	ABDE	30	ACE	50

(b)

그림 4 최대값 가중치 질의 그래프

가중치(edge weights)가 필요하다. 그러나 우리는 현재 노드 가중치만 가지고 있으므로 노드 가중치로부터 선 가중치로의 변환이 필요하다. 선의 가중치는 인접한 노드의 가중치를 합침으로서 구할 수 있다. 5중 조인에서 노드 가중치와 선 가중치를 모두 표시한 질의 그래프의 예가 그림 4(a)에 나와 있다. 우리는 이러한 질의 그래프를 *최대값 가중치 질의 그래프(maximum weighted query graph)*라 한다.

최대값 가중치 질의 그래프에서 두 노드 S와 D 간에 직접 술어가 없을 때, S와 D 간의 ipp와 ippl은 아래와 같이 구해진다.

1. 먼저 x-축 또는 y-축 별로 최단 경로와 최단 경로 길이를 구한다.
2. 최단 경로 길이로부터 노드 S와 D의 가중치를 뺀 후 2로 나눈다.

위에서 두 번째 단계와 같이 하는 이유는 우리가 구하고자 하는 것은 최단 경로에서 중간 노드에 있는 가중치의 합인데 비하여, 첫 번째 단계에서 구한 최단 경로 길이에는 시작 노드 S와 끝 노드 D의 가중치가 포함되어 있고 중간 노드의 가중치는 두 번 포함되어 있기 때문이다. 따라서 노드 S와 D 간의 x_ippl은 아래 수식과 같이 계산되어진다.

$$x\_ippl(S,D) = (x\_shortest\_path\_length(S,D) - x\_max(S) - x\_max(D))/2 \quad (4)$$

y_ippl도 같은 방법으로 정의된다. 그림 4-(a)에서 모든 간접 술어 쌍에 대한 ipp와 ippl은 그림 4-(b)에 나와 있다. 그림 4(b)에서 간접 술어 쌍 AD와 AE의 경우를 보면 같은 노드 쌍이라도 x-축 또는 y-축 별로 최단 경로가 다를 수 있으므로 x_ipp와 y_ipp는 서로 다르게 나타난다.

이렇게 해서 얻어진 노드 간 간접 술어 경로 및 길이는 MRJ의 중간 노드 처리 시 직접 술어만으로는 찾아내지 못하는 거짓 겹침을 찾는데 활용된다. 지금까지 예로 보였던 그림 4(a)의 질의 그래프와 그림 4(b)의 공간 객체 집합들 간의 간접 술어 경로 및 길이를 가지는 5중 공간 조인 처리에서 R-트리 중간 노드들 간의 MBR 조합이 그림 5와 같다고 가정하자. 그림 5에서 점선의 큰 사각형은 R-트리 중간 노드들 간의 MBR을 나타내고, 실선의 작은 사각형은 R-트리의 리프 노드 엔트리에 해당하는 공간 객체들의 MBR을 나타낸다.

그림 5의 중간 노드들 간의 MBR 조합은 그림 4(a)의 질의 그래프 상의 직접 술어들, 즉 (A intersect B) and (A intersect C) and (B intersect D) and (C intersect D) and (C intersect E) and (D intersect E), 은 모두 만족한다. 그림 4(a)의 노드 D를 보면 공간 객체 집합 D에 있는 모든 공간 객체들에 대해 MBR의

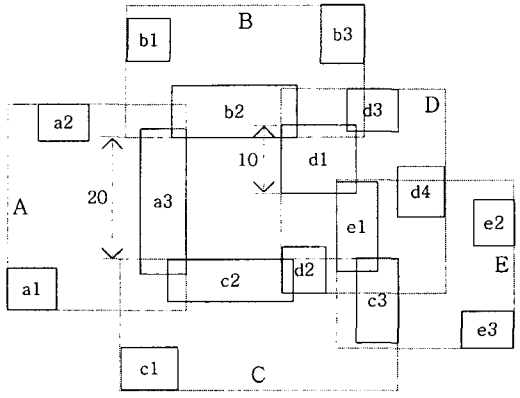


그림 5 5중 공간 조인에서 간접 술어의 적용 예

x-길이, y-길이의 최대값이 각각 10이라는 사실을 알 수 있다. 그림 5의 중간 노드 조합이 질의 결과를 내기 위해서는 그 노드들 아래에 있는 리프 노드들의 엔트리 조합들 간에도 위의 술어가 모두 만족되어야 한다. 따라서 그 술어들 중 일부분인 (B intersect D) and (C intersect D)도 당연히 만족되어야 한다. 그리기 위해서는 공간 객체 집합 B의 엔트리는 D의 엔트리와 겹쳐야 하고, D의 엔트리는 C의 엔트리와 겹쳐야 한다.

그러나 그림 5에 의하면 중간 노드 간의 겹침에서 이미 B와 C 간의 y-길이가 20이므로, 최대 y-길이가 10인 D의 엔트리들은 리프 노드 단계에서 B의 엔트리와도 겹치면서 동시에 C의 엔트리와도 겹치는 엔트리는 존재할 수 없다. 그림 5의 예에서 최대 y-길이를 가진 리프 노드 엔트리 d1은 b3와는 겹칠 수 있으나 C에 있는 어느 엔트리와는 겹칠 수 없다. 따라서 그림 5의 중간 노드들 간의 MBR 조합은 더 이상 진행되지 못하고 중간 노드 단계에서 제거된다. 즉, 노드 D가 B와 C 간의 간접 술어 y-경로(y_ipp)로 활용되고, 공간 객체 집합 D에서 공간 객체들의 최대 y-길이 값(10)이 B와 C 간의 간접 술어 y-경로 길이(y_ippl)로 활용된 셈이다.

그림 4(a)를 보면 노드 B와 C 간의 간접 술어 y-경로로 BAC, BDC 등이 가능하나 노드 D의 y-길이 최대값(10)이 A의 y-길이 최대값(30) 보다 작으므로 위에서 설명한 최단 경로 알고리즘에 의해 BDC가 간접 술어 y-경로로 결정되었다.

4. 간접 술어의 다양화

4.1 노드 최대값

지금까지 우리는 공간 객체 집합 별로 하나의 x-길이 최대값과 y-길이 최대값 만을 가정하였다. 우리는 그것을 *도메인 최대값(DMAX: domain maximum)*이라 부른다. 그러나 이런 경우에 공간 객체 집합에 있는 대부분의 객체는 그리 크지 않더라도 한두 개의 아주 큰 객체가 있게 되면 간접 술어 효과는 현저히 떨어질 수 있다. 이에 대한 한가지 해결책은 x-길이 최대값과 y-길이 최대값을 R-트리 노드별로 가지는 것이다. 각 노드의 최대값 정보는 그 노드 이하에 있는 모든 부 트리 영역 내에 있는 공간 객체들의 최대값이다. 리프 노드는 그 노드 내에 있는 모든 공간 객체 엔트리들의 MBR에 대한 x-길이 최대값과 y-길이 최대값을 가진다. 중간 노드는 모든 자식 노드들의 x-길이 최대값과 y-길이 최대값 중 가장 최대값을 자신의 최대값으로 가진다. 이렇게 하면 결국 루트 노드는 전체 공간 객체 집합에 대한 x-길이 최대값과 y-길이 최대값을 가지게 된다. 따라서 R-트리 노드의 x-길이 최대값은 아래의 수식과 같이 재귀적으로 정의된다.

$$x\_max(N) = \begin{cases} \max N_1.rect_x \ldots N_n.rect_x & \text{for } \leq af\ node \\ \max x\_\max(N_1.ref) \ldots x\_\max(N_n.ref) & \text{for } nonleaf\ node \end{cases} \quad (5)$$

y-길이 최대값도 같은 방법으로 정의된다. 우리는 이와 같이 노드별로 가지고 있는 x-길이 최대값과 y-길이 최대값을 노드 *최대값(N_MAX: node maximum)*이라 부른다.

도메인 최대값 대신에 노드 최대값을 사용함으로써 우리는 MRJ 처리 시 IPF에 의한 탐색 제거 효과를 한 층 더 높일 수 있다. 그림 6에서 보는 바와 같이 2차원 R-트리에서 노드 최대값을 위해서 노드별로 단지 2개의 정보, 즉 x-길이 최대값과 y-길이 최대값만 더 가지면 되므로 기억장소 오버헤드는 별로 없다. 우리는 노드 최대값 정보를 가지는 이러한 형태의 R-트리를 *최대값 태그 R-트리(maximum tagged R-tree)*라 부른다.

MRJ 처리 시 모든 노드 조합에 대해 최단 경로를 계산하는 것은 상당한 처리 시간 오버헤드를 가져올 수 있다[1]. 따라서 본 논문에서는 루트 노드에 있는 최대값 정보를 이용하여 x-축과 y-축에 대해 ipp를 각각 한번씩만 계산한 후 그 경로를 모든 노드 조합에서 그대로 이용한다. 이는 루트 노드들을 이용해 계산한 최단 경로가 다른 단계에서의 노드들 간의 최단 경로와 크게 차이가 나지 않을 것이라는 가정 때문이다. 그러나 ipp가 같아도 ippl은 노드 조합 마다 일반적으로 다르므로 ippl은 모든 노드 조합에 대해 계산을 해야 한다. ipp가 이

NMAX	엔트리$_1$		엔트리$_n$
<x_max, y_max>	<rect$_1$, ptr$_1$>	. . .	<rect$_n$, ptr$_n$>

그림 6 최대값 태그 R-트리의 노드 구조

1) 모든 노드 쌍 간의 최단 경로를 계산하기 위한 알고리즘 복잡도는 $O(M^3)$으로 알려져 있다[16].

미 정해져 있으면 ippl을 계산하는 것은 선형 시간에 가능하다.

4.2 엔트리 최대값

IPF의 목적은 MRJ 처리 시 직접 술어를 만족하는 특정 엔트리 조합이 간접 술어를 만족하지 않으면 그 엔트리 조합이 가리키는 자식 노드 이하의 모든 객체 조합은 질의 결과가 될 수 없기 때문에 자식 노드들을 더 이상 방문할 필요 없이 탐색에서 제거할 수 있다는 것이다. 최대값 태그 R-트리에서는 자식 노드들을 가리키는 엔트리들 간에 간접 술어를 적용하기 위해서는 현재 노드의 최대값을 이용한다. 그러나 현재 단계에서 특정 노드의 최대값은 그 노드에 속한 모든 자식 노드들의 노드 최대값의 최대값이므로(수식 5), 이는 우리가 필요로 하는 최대값 보다 더 큰 값이다. 그리고 R-트리의 높이가 2인 경우는 유일한 중간 노드인 루트 노드의 최대값이 도메인 최대값과 같으므로 노드 최대값을 이용하는 이점이 없다. 엔트리 조합에서 간접 술어를 적용하기 위해서는 그 엔트리가 가르키는 자식 노드들의 노드 최대값이 가장 최적의 값이다. 그러나 최대값 태그 R-트리 구조에서는 그 자식 노드들을 방문하지 않고는 그들의 최대값 정보를 알 수가 없다. 그래서 우리는 IPF의 마지막 전략으로 중간 노드의 엔트리 정보 내에 그 엔트리가 가르키는 자식 노드의 최대값을 갖게 한다. 우리는 그러한 엔트리별 최대값 정보를 *엔트리 최대값 (EMAX: entry maximum)*이라 하고, 엔트리 최대값을 가지는 R-트리를 *최대값 엔트리 R-트리(maximum entried R-tree)*라 부른다.

그림 7에서 보는 바와 같이 최대값 엔트리 R-트리의 중간 노드에서 엔트리는 (rect, ptr, max)로 구성되어 있다. 여기서 rect는 자식 노드의 MBR을, ptr은 자식 노드에 대한 포인터를, max는 자식 노드의 노드 최대값 정보를 각각 나타낸다. 리프 노드의 R-트리 구조는 일반적인 R-트리 노드 구조와 같다. MBR에서 x1, x2, y1, y2 값, 자식 노드에 대한 포인터, 엔트리 최대값에서 x_max, y_max 값 모두 4 바이트 씩이라고 가정하면, 최대값 엔트리 R-트리에서 중간 노드에 있는 하나의 엔트리가 차지하는 저장 공간은 28 바이트이다. 이는 일반 R-트리의 20 바이트 보다 많은 저장 공간을 필요로 한다. 그러나 R-트리에서 대부분의 저장 공간을 리프 노드가 차지하므로 이는 그리 큰 오버헤드는 아니다. C를 R-트리 노드에서 평균 엔트리 개수라고 할 때 같은 R-트리 단계에서 엔트리 최대값이 커버하는 공간 객체 수는 노드 최대값이 커버하는 객체 수에 비해 1/C 밖에 안되므로, 엔트리 최대값을 이용한 IPF는 적은 저장 공간 오버헤드로 많은 탐색 제거 효과를 볼 것으로 기대된다.

엔트리$_1$		엔트리$_n$
<rect$_1$, ptr$_1$, max$_1$>	. . .	<rect$_n$, ptr$_n$, max$_n$>

그림 7 최대값 엔트리 R-트리의 노드 구조

5. 실 험

MRJ에서 IPF의 효과를 측정하기 위하여 우리는 가공 데이타와 실제 데이타를 활용하여 몇가지 실험을 하였다. 본 실험에서는 지금까지 제안된 몇 가지 MRJ 알고리즘 중 가장 효율적이라고 알려져 있는 SRO-PSFC 조합을 이용하였다.

우리는 실험을 위하여 4가지 형태의 다중 공간 질의를 선택하였다: 절반(half), 링(ring), 체인(chain), 스타(star). 그림 8은 5중 공간 조인에서 각각의 질의 형태에 대한 질의 그래프를 보여 준다. 실험에 사용된 공간 술어는 모두 교차(intersect: not disjoint)이다.

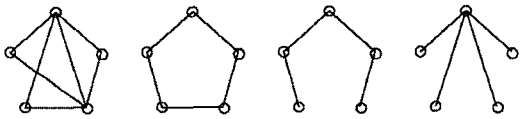


그림 8 5중 조인에 대한 질의 그래프의 예

5.1 가공 데이타

가공 데이타 집합을 이용해 IPF의 성능을 측정하기 위해 우리는 먼저 크기가 (100000, 100000)인 도메인 내에서 각각 0.25와 1.0의 데이타 밀도와 균등 분포를 가지는 10000 개의 사각형으로 구성된 몇 개의 데이타 집합을 생성하였다. 각각의 데이타 집합에 대해 데이타 밀도를 일정하게 유지하기 위해 우리는 같은 크기의 사각형을 생성하였다. 그런 후 각각의 데이타 집합에 대해 노드 크기가 1K와 4K인 R*-트리[10]를 구축하였다. 각각의 노드 크기에 대해 R*-트리의 높이는 3과 2이고, LRU 버퍼의 수는 각각 512와 256 페이지이다[2].

그림 9는 체인 형태의 질의에서 IPF를 사용하지 않은 경우(NO_IPF)의 응답 시간과 IPF를 사용한 경우에는 DMAX 값의 변화에 따른 MRJ 알고리즘의 응답 시간의 차이를 보여준다. 실제로는 같은 크기의 사각형을 생성하였으므로 DMAX, NMAX, EMAX의 값이 사각형의 크기와 같다. 그러나 실험의 편의를 위해 가공 데이타 집합에서는 DMAX 값만 변화가 가능하다고 가정하였다. 이 같은 경우는 데이타 집합 내에서 나머지 사각형은 그대로 두고 한 개의 사각형만 DMAX에 따라 크기를 변화 시키면 가능해진다. 그림 9에서 보듯이 M이

2) 우리는 R*-트리 노드 하나가 한 개의 페이지를 차지한다고 가정한다.

증가하거나 DMAX가 작을수록 IPF의 효과는 커진다. 또한 데이타 밀도가 작을수록 IPF의 효과는 크게 나타났다. 특히 노드 크기가 4K인 경우에는 데이타 밀도의 차이에 의한 IPF 효과도 더욱 분명하게 나타났다. 본 실험에서는 밀도가 낮고, M은 가장 크고, DMAX도 가장 작을 경우 (밀도=0.25, M=7, DMAX=500)에 IPF는 NO_IPF 보다 약 5배 가량 빠른 성능을 보였다. 그러나 DMAX의 값이 큰 경우에는 IPF의 효과가 별로 나타나지 않았다.

우리는 다른 질의 형태에 대해서도 IPF의 성능을 측정하였다(그림 10). 절반 질의 형태에서는 별로 효과가 없으나 스타 질의 형태에서는 큰 효과가 나타났다. 이는 직접 술어의 개수가 적으면 결과적으로 간접 술어의 개수는 많아지므로 IPF의 효과는 크게 나타난다라는 사실을 보여준다.

5.2 실제 데이타 집합

실험에 사용된 실제 데이타 집합은 미국의 디지털 지도인 TIGER/Line[7] 데이타로부터 추출되었다. 우리는 TIGER 데이타 중 California 주의 7개 카운티 지역의 도로 데이타를 사용하였다. California 지역의 TIGER 데이타에 대한 통계 정보는 표 1에 정리되어 있다. 표 1에서 Li_max는 데이타 집합 위에 구축된 R*-트리의 i 번째 단계에서 평균 노드 최대값을 의미한다. L0_max는 리프 노드 단계에 대한 값이며 L2_max는 루트 노드 단계에 대한 값(즉 평균 도메인 최대값)이다. 지역적으로 다른 데이타 집합들 간에 조인을 하기 위해서 모든 카운티 지역의 데이타의 중앙점을 일치시켰다. 즉, 원래 TIGER 데이타의 x와 y 좌표에서 각 카운티 지역의 중앙점 좌표값을 뺐다.

우리는 각각의 데이타 집합에 대해 노드 크기가 4K인 R*-트리를 구축하였다. R*-트리의 높이는 모두 3이다. 본 실험을 위해 우리는 표 1에 보이는 TIGER 데이

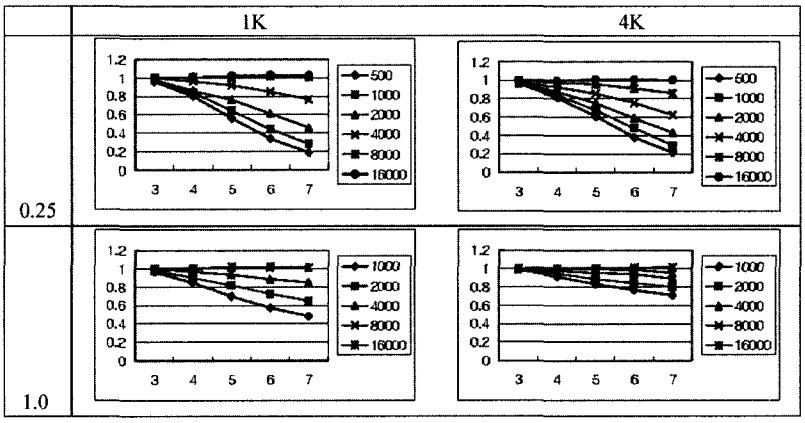
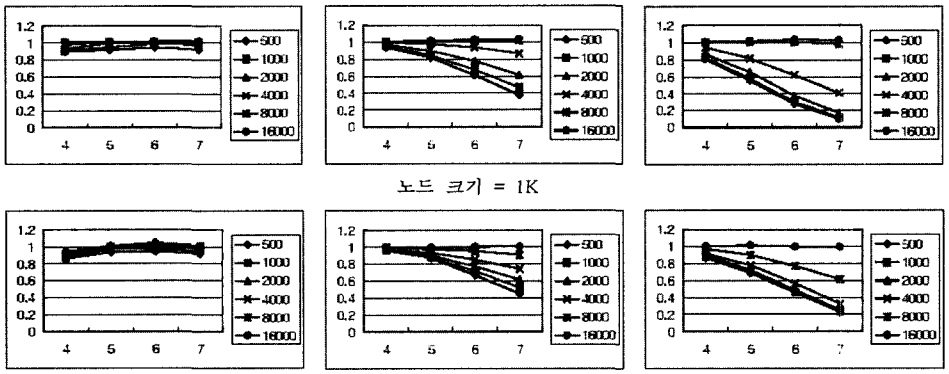


그림 9 체인 형태의 질의에서 IPF의 상대적 성능비 (IPF/NO_IPF)



(a) 절반 (b) 링 (c) 스타

그림 10 다른 질의 형태에서 IPF의 상대적 성능비 (밀도=0.25)

표 1 California TIGER 데이타의 통계 정보

(a) 기본 통계 정보

카운티	객체수	도메인 크기	평균 길이	밀도
Ala.	49070	86222,44995	102,80	0.23
Ker.	113407	257781,100758	212,169	0.26
Mon.	35417	175744,112068	234,192	0.20
Ora.	91970	69999,55588	80,66	0.21
Sac.	46516	75771,71218	111,86	0.24
S.D.	103420	151241,96476	122,104	0.22
S.B.	64037	99301,58696	100,81	0.22

(b) R*-트리 단계별 평균 노드 최대값

카운티	L2_max	L1_max	L0_max
Ala.	4662,3940	3691,2640	676,550
Ker.	8204,6497	5848,4599	1590,1273
Mon.	9085,6194	6608,4504	1675,1351
Ora.	3658,6735	2423,2638	557,488
Sac.	6442,4103	4642,3223	788,604
S.D.	8054,6828	4695,3819	973,807
S.B.	4541,6460	2931,2845	660,540

타 집합으로부터 무작위로 아래의 3개 데이타 조합을 추출하였다. 각각의 조합에 대해서 M-다중 조인은 처음 M 개의 카운티에 대해서 수행되었다(M = 3, ..., 7).

데이타 조합 1: Ala. S.D. Sac. Ker. Mon. S.B. Ora.
데이타 조합 2: Mon. S.B. S.D. Sac. Ker. Ala. Ora.
데이타 조합 3: Ora. Ala. S.B. S.D. Ker. Sac. Mon.

위의 데이타 조합에 대해 우리는 DMAX, NMAX, EMAX와 같은 최대값 정보를 이용해 IPF의 성능을 측정하였다. 그림 11은 실제 데이타 집합의 다양한 조합에서 IPF를 사용하지 않은 경우와 IPF를 사용한 경우의 질의 응답 시간 차이를 보여준다.

가공 데이타 집합에 대한 실험의 경우와 마찬가지로 실제 데이타 집합에서도 M 값이 크거나 직접 술어의 수가 적은 체인 또는 스타 질의에서 IPF의 효과가 크게 나타났다. 또한 DMAX, NMAX, EMAX 순으로 최대값 정보의 크기가 작아지므로 IPF의 성능 향상은 EMAX, NMAX, DMAX 순으로 나타났다. 특히 EMAX의 성능이 DMAX와 NMAX의 성능 보다 월등히 뛰어났다.

스타 질의의 경우에 IPF의 성능은 중앙 노드의 특징에 많은 영향을 받는다. 이는 질의 그래프에서 모든 ipp는 중앙 노드를 통과하기 때문이다. 본 실험에서 스타 질의를 위한 중앙 노드는 각각의 데이타 조합에서 첫 번째 카운티를 채택했다. 만약 스타 질의의 중앙 노드에 해당되는 데이타 집합이 작은 도메인 영역에 많은 객체들로 밀집되어 있으면, R*-트리의 중간 노드들의 MBR 크기도 상대적으로 작아져서 IPF의 효과가 줄어들 것이다. 이런 이유로 해서 데이타 조합 3에 대한 스타 질의의 경우에 IPF 효과가 별로 나타나지 않은 것으로 풀이된다. 위로부터 우리는 IPF는 도메인 영역이 클수록 효과가 크다는 사실을 알 수 있다.

본 실험에 대한 최종 결론은 IPF는 NO_IPF에 비해 많은 경우에 성능 향상을 꾀할 수 있다. 공간 질의 특성으로 보면 M이 크고 직접 술어의 개수가 적으면 효과가 크게 나타나고, 데이타 특성으로 보면 밀도가 낮고 R-트리의 노드 크기가 크고 도메인 영역이 크고 x-길이/y-길이 최대값이 작을수록 효과가 크게 나타난다. 특히 최대값 정보와 관련하여서는 EMAX를 사용한 IPF가 DMAX와 NMAX를 사용한 경우 보다 성능이 훨씬 뛰어났다.

6. 결 론

본 논문에서 우리는 M-다중 R-트리 조인(MRJ)에 간접 술어 개념을 도입하였다. 그리고 그 간접 술어를 이용하여 MRJ의 성능을 향상시킬 수 있는 간접 술어 여과(IPF) 기법을 제안하였다. IPF를 위해 우리는 도메인 최대값(DMAX), 노드 최대값(NMAX), 엔트리 최대값(EMAX)이라는 세가지 최대값 정보를 이용하였다. DMAX를 위해서 우리는 질의 최적화기의 카탈로그 정보를 이용하였고, NMAX와 EMAX를 위해서는 최대값 태그 R-트리와 최대값 엔트리 R-트리라는 새로운 R-트리 구조를 제안하였다.

가공 데이타와 실제 지도 데이타를 이용한 실험을 통해서 우리는 IPF가 기존의 MRJ의 성능을 상당히 높일

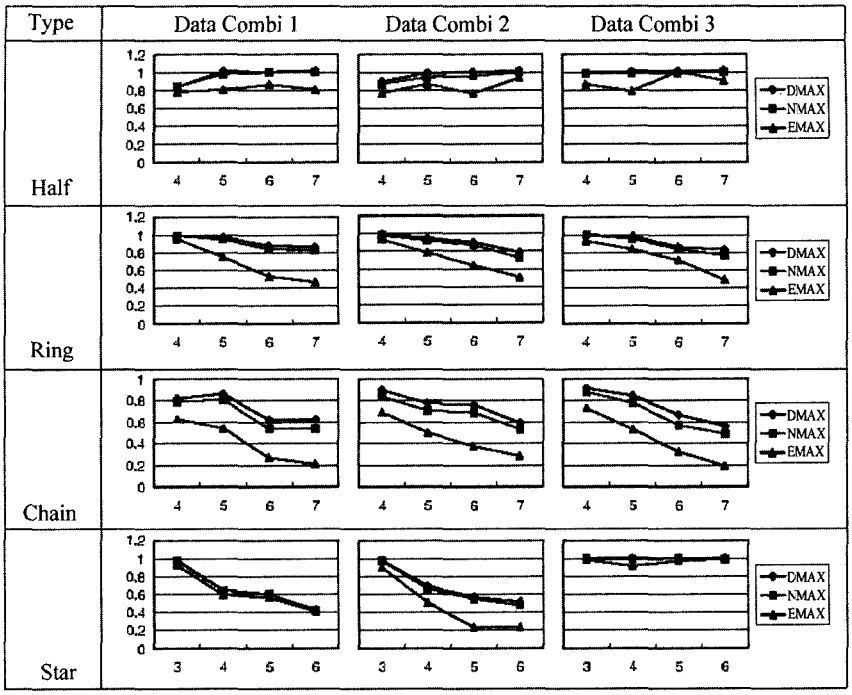


그림 11 실제 데이타 조합에서 상대 성능비 (IPF/NO_IPF)

수 있다는 사실을 보였다. 공간 질의 특성으로 보면 M이 크고 직접 술어의 개수가 적으면 효과가 크게 나타났고, 데이타 특성으로 보면 밀도가 낮고 R-트리의 노드 크기가 크고 도메인 영역이 크고 x-길이/y-길이 최대값이 작을수록 효과가 크게 나타났다. 또한 EMAX를 이용한 IPF가 DMAX와 NMAX를 이용하는 경우 보다 성능이 우월하였다.

참 고 문 헌

[1] R. H. Gueting, "An Introduction to Spatial Database Systems," VLDB Journal, Vol. 3, No. 4, 357-399, 1994.

[2] J. A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System," Proc. of ACM SIGMOD, 326-336, 1986.

[3] N. Mamoulis and D. Papadias, "Integration of Spatial Join Algorithms for Processing Multiple Inputs," Proc. of ACM SIGMOD, 1-12, 1999.

[4] N. Mamoulis and D. Papadias, "Multiway Spatial Joins," ACM Transactions on Database Systems (TODS), Vol. 26, No. 4, 424-475, 2001.

[5] D. Papadias, N. Mamoulis and Y. Theodoridis, "Constraint-based Processing of Multiway Spatial Joins," Algorithmica, Vol. 30, No. 2, 188-215, 2001.

[6] H.-H. Park, G.-H. Cha and C.-W. Chung, "Multi-way Spatial Joins Using R-trees: Methodology and Performance Evaluation," Proc. of SSD, 229-250, 1999.

[7] U.S. Bureau of the Census, Washington, DC., "TIGER/Line Files, 1995, Technical Documentation."

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

[9] T. Sellis, N. Roussopoulos and C. Faloutsos, "The R+-tree:a dynamic index for multidimensional objects," In Proc. of VLDB, 1987.

[10] N. Beckmann, H.-P. Kriegel, R. Schneider and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. of ACM SIGMOD, 322-331, 1990.

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

[12] M. L. Lo and C. V. Ravishankar, "Spatial Joins Using Seeded Trees," Proc. of ACM SIGMOD, 209-220, 1994.

[13] J. M. Patel and D. J. DeWitt, "Partition Based Spatial-Merge Join," Proc. of ACM SIGMOD, 259-270, 1996.

[14] D. Papadias, N. Mamoulis and V. Delis, "Algorithms for Querying by Spatial Structure," Proc. of VLDB, 546-557, 1998.

[15] H.-H. Park, Early Separated Filter/Refinement Strategies and Multi-way Spatial Joins for Spatial

Query Optimization, Ph.D Thesis, KAIST, 2001.
[16] E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1978.

박 호 현

1983년 3월~1987년 2월 서울대학교 계산통계학과(학사). 1993년 9월~1995년 8월 한국과학기술원 정보및통신공학과(석사). 1995년 9월~2001년 2월 한국과학기술원 전자전산학과(박사). 1987년 1월~2002년 2월 삼성전자 통신연구소 수석연구원. 2003년 3월~현재 중앙대학교 전자전기공학부 조교수. 관심분야는 시공간 데이타베이스, 멀티미디어 데이타베이스, XML, 웹 데이타베이스, 홈네트웍

정 진 완

정보과학회논문지 : 데이타베이스
제 30 권 제 3 호 참조