분산 공간 데이터베이스 시스템에서의 적응적 공간 조인 기법

이재훈*°, 김호석*, 이재동**, 배해영*
*인하대학교 전자계산공학과
**단국대학교 전산과
neohacz@dblab.inha.ac.kr

Adaptive Spatial Join Method in Distributed Spatial Database System

Jaehun Lee*°, Ho-Seok Kim*, Jae Dong Lee**, Hae-Young Bae*
*School of Computer Science & Engineering, Inha University
**Dept. of Computer Science, Dankook University

요 약

네트워크의 빠른 발전으로 인해 분산된 공간 데이터의 질의 처리 연구가 활발히 진행되었다. 하지만 이런 질의 처리 환경에서는 질의 처리의 최적화를 위한 정확한 정보를 수집하기 어렵고 네트워크 상태의 불확실성으로 인해 데이터의 전송 상태를 예측하기가 힘들다. 이런 동적인 환경에 적응하기 위해서는 기존의 공간 조인 기법을 수정할 필요가 생겼다. 특히 기존의 공간 조인 기법은 처리 방식이 비대칭적(asymmetric)이기 때문에 데이터 전송의 지연으로 인해 처리가 잠시 중단되거나 빠른 응답 시간을 보장할 수 없다.

본 논문에서는 분산 공간 데이터베이스에서의 공간 조인의 문제점을 해결하기 위해서 대칭적인 해시 공간 조인을 사용하는 적응적 공간 조인 기법을 제안한다. 제안된 기법은 초기의 전송된 데이터들을 조인하여 조인 결과를 빠르게 보여주며 데이터 전송의 지연 시에는 이미 전송된 데이터 중 조인되지 않은 객체들을 조인함으로써 지속적으로 조인을 수행한다.

1. 서 론

인터넷과 WAN의 빠른 발전으로 인해 분산된 공간 데이터들에 대한 공간 질의 처리 연구가 활발히 진행되었다. 하지만 이런 질의 처리 환경은 공간 질의 처리에 필요한 정보가 실시간으로 변하고 네트워크의 상태가 수시로 변하여 공간 질의 처리에 필요한 데이터들이 지연되어 전송되는 등의 동적인 특징을 가진다[1,10].

기존의 공간 질의 처리는 컴파일 시간에 수집된 정보를 가지고 질의 실행 플랜을 작성하여 처리하기 때문에 동적인 환경을 고려하지 못한다. 특히 공간 질의 중 공간 조인은 그 연산 자체가 고비용이기 때문에 분산 공간 데이터베이스에서의 효율적인 질의 처리를 하기 위해서 기존의 공간 조인 기법을 수정할 필요가 있다.

비공간 조인에서는 동적 환경을 고려한 조인 기법에 대해서 많은 연구가 진행되었다. 연구되어진 비공간 조인 기법에는 Ripple 조인, XJoin 등의 기법이 있다[6,7,8]. 이 조인 기법들은 동적 환경의 불확실성을 고려하여 대칭적인 Nested-Loop 조인, 해시 조인을 사용한다. 더불어 XJoin은 데이터 전송의 지연으로 인한 일시적인 처리 중단 상태를 해결하고자 했다.

이와 달리 공간 조인에서는 동적 환경을 고려한 조인 기법에 대해 많은 연구가 진행되지 않았다. 지금까지 연구되어진 공간 조인으로는 Non-Blocking 병렬 공간 조인 기법이 있다[2]. 이는 Ripple Join을 공간 조인에 적용하여 동적 환경에서 효율적인 공간 조인을 수행하도록 하였다. 하지만 이 기법은 데이터의 전송이 지연될 때 디스크에 저장되어 있는 데이터를 활용하여 지속적인 공간 조인을 수행하는 것에 대해서는 고려하지 않았다.

본 논문에서는 분산 공간 데이터베이스에서의 적응적 공간 조인 기법을 제안한다. 제안하는 적응적 공간 조인 기법은 R-트리 기반의 대칭적인(Symmetric) 공간 해시 조인을 이용하여 동적인 분산 공간 데이터베이스에서 효율적인 적응적 공간 조인을 수행한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구들을 살펴본다. 3장에서는 제안하는 적응적 공간 조인 기법을 기술한다. 끝으로 4장에서 결론 및 향후 연구 방향을 논한다.

2. 관련 연구

2.1 공간 해시 조인 기법

기존의 공간 해시 조인 기법에는 Seeded 트리를 사용하는 공간 해시 조인과 파티션 기반의 공간 합병 조인이 있다[3,4].

공간 해시 조인은 Seeded 트리 기반에 버켓 Extent와 할당 함수를 해시 함수로 사용하여 공간 조인을 수행한다. 이 기법은 세 단계로 수행된다. 먼저 한 릴레이션을 샘플링하고 분할하여 버켓 Extent를 정한 다음 그 릴레이션을 다시 해당 버켓에 일대일로 할당한다. 다음으로 다른 릴레이션을 해당하는 버켓에 할당한다. 마지막으로 각각의 할당을 통해 버켓에 있는 공간 조인을 위한 후보 쌍들을 조인한다. 이 조인 기법은 우선적으로 한 릴레이션의 모든 데이터가 요구되는 비대칭적인 공간 해시 조인이기 때문에 빠른 응답 시간을 보장하지 못한다. 따라서 본 논문에서는 빠른 응답 시간을 보장하기 위하여 대칭적인 공간 해시 조인을 제안하여 사용한다.

파티션 기반의 공간 합병 조인은 입력되는 릴레이션의 객체들을 해당 파티션으로 할당하는 분할 단계와 각각의 파티션에 속해 있는 객체들에 대해 Forward-sweep 알고리즘을 사용하여 Plane-sweep하는 단계로 공간 조인을 수행한다. 이 기법은 각각의 파티션에 해당하는 객체들의 중복이 많다는 문제를 가지고 있다.

2.2 동적 환경을 고려한 조인 기법

동적 환경을 고려한 조인 기법에는 Ripple 조인, XJoin, Non-Blocking 병렬 공간 조인이 있다[2,7,8]. Ripple 조인에서는 대칭적 Nested-Loop 조인을 이용하여 적응적 조인을 수행하며, 이 기법은 Non-Blocking 병렬 공간 조인에서 공간 조인 기법으로 사용되어진다. Non-Blocking 병렬 공간 조인에서 사용되는 적응적 공간 조인은 세 단계로 나누어져서 수행한다. 첫 번째 단계는 각각의 릴레이션의 객체들이 전송되어지면 각

각의 릴레이션에 해당하는 R-트리에 삽입되고 다른 릴레이션의 R-트리를 통해 공간 조인을 수행한다. 두 번째 단계는 첫 번째 단계에서 전송된 객체들이 메모리 용량을 벗어나게 되면 이후로 전송된 객체에 대해서 상대 릴레이션의 R-트리를 사용하여 현재 메모리에 있는 상대 릴레이션의 객체들과 공간 조인을 하고 전송된 객체는 디스크에 저장한다. 전송된 객체는 파티션 기반의 공간 합병 조인에서 사용된 공간 분할 함수로 생성된 해당 버켓에 저장된다. 세 번째 단계는 두 릴레이션의 객체들이 모두 전송되었을 때 두 번째 단계에서 디스크에 저장된 객체들에 대해서 공간 조인을 수행한다. 공간 조인은 대응되는 버켓의 객체들을 상대로 수행된다. 이 기법은 데이터 전송의 지연을 고려하여 지속적인 공간 조인을 수행하지 않는 단점을 가진다.

XJoin은 대칭적 비공간 해시 조인을 사용한 적응적 비공간 조인 기법이다[8]. 이 기법은 먼저 두 릴레이션에 같은 해시 함수를 사용하여 전송되는 튜플들을 해당 버켓에 삽입하고 대응되는 버켓의 튜플들과 조인을 수행하는 단계를 행한다. 다음으로 메모리상의 튜플들 크기가 메모리 용량을 벗어났을 때 한 버켓을 선정하여 디스크에 저장하게 되는데 튜플의 전송이 지연되면 이 저장된 버켓의 튜플을 가지고 지속적으로 조인을 수행하는 단계를 행한다. 마지막으로 모든 튜플이 전송되고 난 후에는 조인이 수행되지 않은 저장된 튜플들에 대해 조인을 수행하는 단계를 수행한다. 이 기법에서는 저장되는 버켓에 대한 조인을 수행할 때 중복이 생기게 되어 중복 제거를 위한 추가적인 작업을 수행하는 단점을 가진다.

3. 적응적 공간 조인 기법

본 논문에서 제안하는 적응적 공간 조인 기법은 다음을 목적으로 한다. 먼저 공간 조인의 결과에 대해 빠른 응답 시간을 제공한다. 그리고 데이터의 전송이 지연될 때 지속적인 공간 조인을 수행한다. 또한 메모리의 제약을 고려한 공간 조인을 수행한다.

3.1 적응적 공간 조인 기법을 위해 사용되는 대칭적 공간 해시 조인 기법

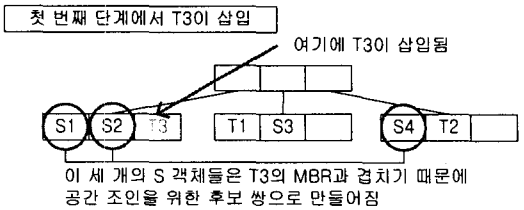
기존의 공간 해시 조인은 비대칭적인 공간 해시 조인을 수행하는데 XJoin은 빠른 응답 시간을 제공하기 위해서 대칭적인 비공간 해시 조인을 사용한다. 따라서 본 논문에서는 기존의 비대칭적인 공간 해시 조인 기법을 사용하지 않고 두 릴레이션에 대해 하나의 R-트리를 사용하는 대칭적인 공간 해시 조인 기법을 제안하여 사용한다.

두 릴레이션 S와 T에 대한 대칭적인 공간 해시 조인은 다음의 두 단계로 수행된다.

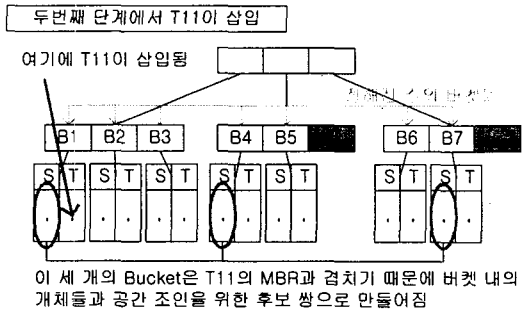
첫 번째 단계는 먼저 공간 해시 조인을 위해 메모리의 크기나 S와 T의 객체 개수 및 크기를 고려하여 버켓 개수를 정한다. 그리고 S와 T의 객체들이 전송되어 오면 그 객체들을 하나의 R-트리에 삽입한다. 이 때 R-트리를 구성하면서 R-트리의 하위 노드들이 정해진 버켓 개수만큼 생성되어 있는가를 확인하고 생성되었다면 생성된 하위 노드들을 버켓으로 고정시키고 다음 단계를 수행한다.

두 번째 단계는 들어오는 객체들에 대해서 R-트리 삽입을 행하는 것이 아닌 해당 버켓을 찾아서 그 버켓에 객체를 삽입한다. 객체가 해당하는 버켓을 찾을 때 기존의 버켓들의 범위(MBR)에 들어가지 않는 객체라면 객체의 중심과 가까운 버켓을 찾아서 삽입한다. 객체가 버켓에 삽입되면 버켓의 범위는 삽입된 객체의 범위를 포함하도록 재조정된다.

예를 들어 [그림1]과 같이 첫 번째 단계에서 T3이 전송되어 왔다면 R-트리에 삽입된 후 R-트리를 이용하여 T3의 MBR과



[그림 1] 두 릴레이션 S,T에 대한 대칭적 공간 해시 조인의 첫 번째 단계에서 T3이 삽입되었을 경우



[그림 2] 두 릴레이션 S,T에 대한 대칭적 공간 해시 조인의 두 번째 단계에서 T11이 삽입되었을 경우

겹쳐지는 S의 모든 객체를 검색하여 S1, S2, S4를 찾아낸다. 이 세 개의 S 객체들과 T3은 공간 조인을 위한 후보 쌍을 이루게 되고 공간 조인을 수행하여 바로 결과를 내보내게 된다. S 객체의 경우에도 이와 같은 순서로 공간 조인을 수행한다. 객체 삽입과 공간 조인을 수행하면서 R-트리의 하위 노드의 개수가 지정된 버켓의 수인지 확인한다. 만약 지정된 버켓 수만큼의 하위 노드가 생성되었다면 하위 노드 각각을 버켓으로 지정하고 두 번째 단계를 행한다.

두 번째 단계에서 T11이 전송되어 왔다면 R-트리를 통해 해당 버켓을 찾는다. T11의 범위를 포함하는 버켓이 없다면 기존의 버켓들 중 T11의 중심과 가까운 버켓을 찾는다. T11을 해당 버켓에 삽입한 다음 해당 버켓의 범위가 T11의 범위를 포함하도록 재조정하고 버켓내의 T의 대표 MBR도 재조정한다. 그 다음 해당 버켓의 S의 대표 MBR과 비교하여 T11이 겹쳐진다면 버켓내의 모든 S 객체와 겹쳐지는지 검사한다. 검사 후 겹쳐지는 객체들에 대해서 후보 쌍을 생성한다. 또한 다른 버켓의 범위와 겹쳐지는지 검사하여 T11이 버켓 4,7과 겹쳐지는 것을 확인한다. 이 버켓도 S 대표 MBR과 검사 후 겹쳐지면 버켓내의 모든 S 객체와 겹쳐지는지 검사하여 후보 쌍을 생성한다. 생성된 후보 쌍들에 대해 공간 조인을 수행한 후 결과를 내보낸다.

버켓의 구성은 다음과 같다.

S	T
해당 버켓 내의 S의 모든 객체를 포함하는 대표 MBR	해당 버켓 내의 T의 모든 객체를 포함하는 대표 MBR
{S_n의 ID,S_n의 MBR}	{T_m의 ID,T_m의 MBR}
⋮	⋮

(n, m은 임의의 숫자로서 Sn과 Tm은 S와 T의 임의의 객체를 나타냄)

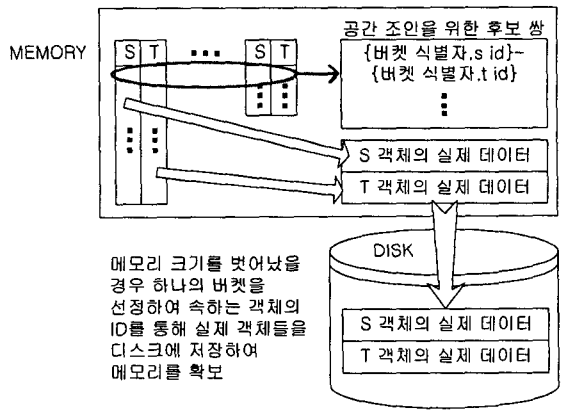
[그림 3] 한 버켓의 구조

S와 T의 객체를 따로 넣을 수 있는 큐에 S와 T의 객체의 ID 및 MBR를 해당 큐에 삽입한다. ID는 해당 객체에 대한 메모리에서의 위치나 디스크의 위치를 찾을 수 있는 식별자이다. 그리고 버켓에 해당하는 모든 S(T)의 객체들을 포함하는 대표 MBR을 유지한다. T(S)의 객체가 삽입되어 영역의 겹침을 검사할 때 해당 버켓이 겹쳐진다면 버켓에 있는 S(T)의 객체들과 겹침을 검사하기 이전에 S(T)의 대표 MBR과 먼저 검사하여 다음 검사를 수행할지를 결정한다. 이는 개별적인 객체들에 대한 검사 전에 겹치지 않는 객체들을 선별하는 역할을 한다.

3.2 제안된 적응적 공간 조인 기법

제안된 대칭적 공간 해시 조인 기법을 사용하여 다음 세 단계로 적응적 공간 조인을 수행한다.

첫 번째 단계는 제안된 대칭적 공간 해시 조인을 메모리 상에서 수행한다. 대칭적 공간 해시 조인의 첫 번째 단계를 행하여 공간 조인을 위한 후보 쌍을 생성하고 공간 조인을 수행한다. 정해진 버켓 수의 하위 노드들이 생성되어 버켓으로 고정되면 이후로 삽입되는 객체들은 해당 버켓에 삽입되며 제안된 대칭적 공간 해시 조인의 두 번째 단계를 수행한다. 이 때 생성된 후보 쌍들은 메모리에서 {버켓 식별자, 객체 ID}-{버켓 식별자, 객체 ID}의 쌍으로 유지되며 공간 조인을 수행한 후에 삭제된다.



[그림 4] 메모리 크기를 벗어났을 경우 수행되는 두 번째 단계

이 첫 번째 단계를 수행하면서 메모리 크기를 벗어나게 되면 하나의 버켓을 선택하여 버켓 내에 있는 두 릴레이션의 객체들을 디스크에 저장하게 되고 이후에 이 버켓과 겹쳐지는 객체들에 대한 후보 쌍을 메모리 상에 저장한 후 직접적인 공간 조인을 수행하지 않는다. 디스크에 저장되는 데이터는 버켓 내의 객체 ID들이 가르키는 객체의 실제적인 데이터로 메모리에서 삭제된 후 다시 첫 번째 단계를 수행하게 된다. 직접적인 공간 조인을 수행하지 않은 후보 쌍들은 두 번째 단계나 세 번째 단계를 통해 공간 조인을 수행한 후 삭제된다.

두 번째 단계는 첫 번째 단계가 데이터 전송의 지연으로 인해 일시적으로 중지 될 경우 수행된다. 첫 번째 단계에서 메모리 크기를 벗어남으로 디스크에 저장된 객체에 대한 후보 쌍이 있을 경우 저장된 객체를 메모리에 로딩하여 그 후보 쌍에 대한 공간 조인을 수행한다. 공간 조인을 수행한 다음에는 해당하는 후보 쌍은 삭제한다. 이 단계는 두 릴레이션 객체들의 전송이 다시 시작되었을 때 중지되고 첫 번째 단계의 대칭적 공간 해쉬 조인을 계속하여 수행한다.

세 번째 단계는 모든 객체들이 전송되었을 경우 수행된다. 메모리에 있는 객체들에 대한 후보 쌍의 공간 조인이 다 수행되고 난 후 디스크에 있는 객체들을 메모리로 로딩하여 아직 수행되지 않은 후보 쌍에 대해서 공간 조인을 수행한다.

이 기법은 공간 조인을 위한 후보 쌍을 메모리에 유지하여 XJoin에서 저장된 튜플들의 조인에서 중복이 생기는 것을 방지하였고 아직 수행되지 않은 공간 조인을 이후에 수행할 수 있도록 한다. 객체의 ID와 그 객체의 데이터가 존재하는 실제 주소, 즉 메모리에 있다면 메모리의 주소, 디스크에 존재할 경우 디스크의 주소로 이루어져 있는 테이블이 메모리 상에 유지되며 공간 조인을 위한 후보 쌍에 있는 ID는 이 테이블을 이용하여 각각의 실제 객체 데이터에 접근할 수 있다.

4. 결론 및 향후 연구 방향

본 논문의 적응적 공간 조인 기법은 기존의 비대칭적인 공간 해시 조인 대신에 R-트리를 이용한 대칭적인 공간 해시 조인을 제안하여 적응적 공간 조인 기법에 이용한다. 적응적 공간 조인 기법은 전송되는 데이터들을 실시간으로 공간 조인하여 빠른 응답 시간으로 결과를 나타낸다. 그리고 데이터 전송의 지연으로 인한 공간 조인의 일시적인 중지를 해결하여 지속적인 공간 조인을 수행한다. 또한 메모리 용량을 벗어나게 된 경우 버켓에 속하는 객체의 실제 데이터를 디스크에 저장하여 공간 조인을 계속적으로 수행한다. 이와 같은 적응적 공간 조인 기법은 데이터가 분산되어 있고 공간 조인을 위해 필요한 정보가 불확실한 분산 데이터베이스에서 효율적인 공간 조인을 수행할 수 있다.

향후 연구 과제로는 본 기법을 그리드, 데이터 웨어하우징 등의 동적 환경에 적용되도록 확장하는 것이다.

참고 문헌

[1] A. Gounaris, N.W. Paton, A.A.A. Fernades, and R. Sakellariou, "Adaptive Query Processing: A Survey", Lecture Note in Computer Science 2405, 2002

[2] G. Luo, J.F. Naughton and C.J. Ellmann, "A Non-blocking Parallel Spatial Join Algorithm", Proc. 2002 Int. Conf. on Data Engineering(ICDE'02), pp. 697~705, 2002

[3] J.M. Patel, D.J. DeWitt, "Partition Based Spatial-Merge Join", Proc. ACM-SIGMOD, pp. 259~270, 1996

[4] M.L. Lo and C.V. Ravishankar, "Spatial Hash-Joins", Proc. ACM-SIGMOD Int. Conf. Management of Data, pp. 209~220, 1996

[5] M.L. Lo and C.V. Ravishankar, "Spatial Joins using seeded trees", Proc. ACM-SIGMOD Int. Conf. Management of Data, pp. 209~220, 1994

[6] P.J. Haas and J.M. Hellerstein, "Ripple Joins for Online Aggregation", Proc. ACM-SIGMOD Int. Conf. Management of Data, pp 287~298, 1999

[7] R. Avnur and J.M. Hellerstein, "Eddies: Continuously Adaptive Query Processing", SIGMOD, 2000

[8] T. Urhan, M.J. .Franklin, "XJoin: Getting Fast Answers from Slow and Bursty Networks", University of Maryland Technical Report, CS-TR-3994, 1999

[9] W. Hong, M. Stonebraker, "Optimization of Parallel Query Execution Plans in XPRS", Distributed and Parallel Databases, 1(1):9~32, 1993

[10] Z. Ives, D. Florescu, M. Friedman, A. Levy, D.S. Weld, "An Adaptive Query Execution System for Data Intergration", Proc. ACM-SIGMOD Conf., 1999