2002년 한국정보처리학회 춘계학술발표논문집 제 9 권 제 1 호

분산 공간 데이터베이스 시스템에서의 효율적인 질의 처리를 위한 병렬 연쇄 공간 죠인 기법

고주일*, 이환재*, 김명근*, 이순조**, 배해영*
*인하대학교 전자계산공학과
**서원대학교 컴퓨터정보통신공학부
e-mail : ilovjc@hanmail.net

Parallel Pipelined Spatial Join Method for Efficient Query Processing In Distributed Spatial Database Systems

Ju-Il Ko*, Hwan-Jae Lee*, Myoung-Keun Kim*, Soon-Jo Lee**, Hae-Young Bae*
*Dept. of Computer Science, In-Ha University
**School of Computer, Information & Communication, Seowon University

요 약

분산 공간 데이터베이스 시스템에서 자주 수행되는 공간 죠인 질의는 공간 데이터의 특징인 대용량성과 복잡성으로 인하여 공간 연산 수행시 연산을 수행하는 서버의 CPU 및 디스크 I/O 상의 과부하를 일으킨다. 본 논문은 이러한 분산 공간 데이터베이스 시스템에서 수행 비용이 많이 드는 원격 사이트간의 공간 죠인 질의를 병렬적이며 연쇄적으로 수행하는 기법을 제안한다. 본 기법은 공간 죠인 연산의 대상이 되는 릴레이션들을 공간 연산의 특성에 따라 순서화하고, 그 중 최하위의 죠인에 참여하는 릴레이션들 중 하나를 이등분 하는 방법으로 공간 죠인 연산을 분리한 후, 질의 수행에 참여하는 두 서버에게 죠인 연산을 분배한다. 각 서버는 분할된 공간 죠인 연산을 동시에 연쇄적으로 처리하고 결과를 병합하여 최종 죠인 결과를 생성한다. 본 기법은 릴레이션을 분할하여 죠인을 수행함으로써 공간 연산에 참여하는 객체의 수를 절반으로 줄이며 R-Tree 등의 공간 인덱스 탐색 횟수와 그 범위를 감소시킨다. 또한 연쇄적인 질의 처리로 죠인의 결과인 임시 릴레이션을 생성하지 않으므로 대용량의 데이터에 대한 복잡한 질의에 대해서도 제한 없이 수행한다.

1. 서론

분산 공간 데이터베이스 시스템에서 서로 다른 사이트에 존재하는 두 릴레이션 간에 질의를 수행하면 죠인 연산이 수반되는데, 그 중 죠인 조건에 공간 연산이 포함된 분산 공간 죠인 질의는 공간 데이터의 대용량성과 그 복잡성으로 인하여 죠인의 조건이 되는 공간 연산을 수행하는 서버에서 CPU 및 I/O 상의 지역적 부하와 네트워크상의 전송 부하를 발생시켜 질의 처리시간 증가로 인하여 결과가 전달되는 초기 응답시간이 지연되어 사용자의 만족도를 떨어뜨린다.

이런 문제들을 해결하기 위해 문자 및 숫자 기반의 일반적인 분산 데이터베이스 시스템에 대한 질의 처리에 사용되는 세미죠인[1,2]을 기반으로 한 여러 가지 공간 죠인 기법들과 질의 최적화 기법들이 제안되었다. 이 기법들은 데이터 전송량을 감소시켜 네트워크의 부하를 줄이며 공간 연산을 여과 단계와 정제 단계로 나누어 서로 다른 서버에서 수행함으로써 공간 연산 부하를 다소 분산한다는 장점이 있다. 하지만, 공간 연산을 수행하는 부하 가운데 대부분을 차지하는 정제 연산이 특정 서버에 집중되어 병목 현상을 발생시키며, 각 죠인 연산의 결과인 임시 릴레이션을 생성하므로 병렬 처리로 인하여 전체 처리시간 단축은 가능하지만 초기 응답시간 단축을 위한 연쇄적 질의 처리는 어려운 단점이 있다.

이러한 문제를 해결하기 위해 본 논문은 이러한 분산된 환경의 대용량 공간 데이터베이스 시스템에서 수행 비용이 많이 드는 원격 사이트간 공간 죠인 질의를 병렬적이며 연쇄적으로 수행하는 기법을 제안한다. 본 기법은 여러 개의 공간 죠인 연산으로 이루어진 질의를 수행 비용이 적게 들도록 죠인 순서를 재조정한 후, 최하위 죠인 연산의 경우 본 논문에서 제안하는 병렬 연쇄 죠인 기법을 사용하여 수행하며, 나머지 죠인 연산은 색인 기반 죠인 기법을 사용하여 연쇄적으로 수행한다. 최하위의 병렬 연쇄 죠인 연산은 공간 죠인의 대상이 되는 영역을 균등한 수의 공간 객체가 포함 되도록 나누어 대상 릴레이션을 분할한 후 질의 수행에 참여하는 두 서버에 분배하여 병렬로 수행시키므로 질의 처리를 향상시키는 기법이다.

본 기법은 비용이 많이 드는 두 기본 릴레이션(Base Relation)들 간의 공간 죠인 연산 수행 시 입력되는 객체의 수를 반으로 줄여 공간 연산에 의한 CPU 부하와 R-Tree 등의 공간 인덱스 탐색 시 소요되는 디스크 I/O 비용을 줄여 전체적인 질의 수행시간을 단축시킨다[7]. 또한 죠인의 결과 튜플이 다른 죠인의 입력으로 바로 사용되는 연쇄적 질의처리를 위한 죠인 질의 최적화 기법과 반복자(iterator)[3]를 이용한 대수 연산자의 사용으로 사용자에게 결과가 전달되기 시작하는 시간인 초기 응답시간을 단축시키며, 임시 릴레이션을 만들지 않음에 따라 다수의 대용량 데이터를 죠인하는 복잡한 질의에 대해서도 저장공간의 제한 없이 수행할 수 있도록 한다.

2. 관련 연구

본 장에서는 분산된 환경의 공간 데이터베이스 시스템에서 죠인 질의 처리를 위한 기존의 기법에 대해 설명하고, 질의의 연쇄적 처리를 위한 대수 연산자 들에 대하여 설명한다.

2.1 분산 공간 죠인

기존의 일반적인 관계형 분산 데이터베이스를 위한 대표적인 분산 질의 최적화 알고리즘들은 연산비용 보다는 전송 비용 최소화에 중점을 두고 설계가 된 알고리즘들이다. 하지만 네트워크 기술이 발달됨에 따라 전송부하가 점점 감소되고 있으며, 또한 분산된 다중 사이트간의 공간 죠인 질의는 수행 비용의 대부분이 네트워크상의 전송 비용이 아닌 공간 연산 수행 시에 들게 되어 병목현상이 발생된다[4,5]. 따라서 분산 데이터베이스에 대한 공간 죠인 질의의 최적화에는 공간 연산의 수행 비용의 최소화에 중점을 두고 각 사이트간의 연쇄적 질의처리(Pipelining)를 하기 위한 새로운 최적화 기법이 필요하다.

여러 개의 공간 죠인 연산이 포함된 질의를 최적화하기 위해서는 각 죠인의 조건이 되는 공간 연산의 비용을 분석해야 하며, 공간 연산의 총 비용을

*본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임

구하려면 공간 연산의 각 단계에서 필요한 비용을 메타 정보들을 이용하여 추측해 내야 한다.

공간 연산의 총 비용은 여과 과정에 필요한 CPU 및 I/O 비용과 정제과정에 필요한 CPU 및 I/O 비용의 합이라고 할 수 있다. 이 중 여과 과정의 CPU 비용은 객체의 MBR 정보를 객체 헤더에 저장할 경우 객체가 가진 점의 개수와 상관없이 일정하므로 전체 비용을 계산할 때 고려하지 않는다. R-Tree를 이용한 공간 객체 여과 I/O 비용 C_{FIO}은 다음의 (수식 2.1)과 같다[5, 8].

- $$C_{FIO} = \left[\left(\frac{1}{m-1}\right)\left(1-\frac{1}{m^{h-1}}\right) * S_{MBR} * T + 1\right] * C_{randio}$$ .....(수식 2.1)

여기서 S_{MBR}은 특정 MBR 에 의한 Selectivity. T는 릴레이션 내의 튜플의 개수, m 은 R-Tree 노드당 평균 엔트리 개수, h 는 R-Tree 의 높이, 그리고 C_{randio}는 무작위로 한 페이지를 읽는 데 걸리는 비용을 의미한다.

다음으로 두 공간 객체에 대한 공간연산의 정제 비용 중 I/O 비용 C_{RIO}은 다음과 같은 식으로 구해진다.

- $$C_{RIO} = S_{MBR} * T * \left(C_{polyio} + V_{cand} * C_{vertio}\right)$$ .....(수식 2.2)

위의 (수식 2.2)에서 C_{polyio}는 공간 객체 헤더를 읽는 비용을 가리키는 상수이며, C_{vertio}는 점들을 읽는 비용, 그리고 V_{cand}는 후보 객체들의 평균적인 점의 수 이다.

마지막으로, 정제 비용 중 CPU 비용 C_{RCPU}은 다음과 같이 나타내어 진다.

- $$C_{RCPU} = S_{MBR} * T * \left(V_q + V_{cand}\right)\log\left(V_q + V_{cand}\right) * C_{polytest}$$ .....(수식 2.3)

위의 (수식 2.3)에서 V_q와 V_{cand}는 두 객체의 점의 수를 나타내며, $C_{polytest}$는 상수이다.

공간 죠인 질의의 최적화시에 위의 수식들을 이용하기 위해서는 각 릴레이션내에 포함된 객체들의 분포 특성인 S_{MBR} 과 튜플들의 개수인 T, 그리고 객체들의 평균 점 개수 V 에 관한 정확한 정보가 필요하다.

일반적인 관계형 분산 데이터베이스 시스템에서는 서로 떨어진 두 원격 사이트간의 점의 처리 시에 죠인 연산은 세미죠인(Semijoin)을 이용하여 수행한다. 세미죠인을 이용한 죠인 기법을 사용할 경우 릴레이션의 전체 튜플을 전송하지 않고 죠인에 관련된 튜플들만 프로젝션하여 전송하므로 데이터 전송량의 감소로 인한 네트워크 부하 감소의 효과가 있다.[1]

공간 데이터베이스 시스템에서도 기존의 관계형 데이터베이스 시스템에서의 분산 죠인 연산과 마찬가지로 세미죠인을 이용한 죠인 연산의 수행이 가능하지만, 네트워크를 통한 전송 부하가 최대의 걸림돌이던 기존의 관계형 데이터베이스와는 달리, 죠인의 조건이 되는 공간 연산의 처리 부하가 전송 부하보다 더 많은 비중을 차지한다. 그리고, 릴레이션의 크기가 대용량화 되고 객체가 복잡해 짐에 따라 공간 연산의 처리 부하는 죠인 비용의 거의 대부분을 차지하게 된다. 따라서 세미죠인 기법을 이용한 공간 죠인의 수행 알고리즘은 데이터베이스의 크기에 따라 위의 두 가지 부하의 균형을 맞추어 설계되어야 한다[4].

세미죠인 기반의 공간 죠인 연산을 처리하는 알고리즘은 다음과 같다.

1. 릴레이션 R 에서 튜플 한 개를 읽어 공간 객체의 근사 정보를 S 의 사이트로 전송한다.
2. S 의 사이트에서 S 와 전송 받은 근사 정보를 이용하여 여과 연산을 수행하여 후보 객체 리스트인 S' 를 생성한다.
3. S' 를 R 의 사이트로 전송한다.
4. R 의 사이트에서 읽은 튜플과 S' 와의 정제(Refinement) 연산을 수행하여 적합한 객체를 선별하여 죠인한다.
5. 1 ~ 4 의 과정을 R 의 모든 객체에 대하여 수행한다.

위의 알고리즘에 의하면 죠인 연산은 죠인의 조건인 공간 연산의 수행을 두개의 사이트에서 여과연산과 정제연산으로 나누어 수행한다. 하지만, 공간 연산에서 가장 큰 부하로 작용하는 정제연산을 특정 서버에서 모두 처리하므로, CPU 에 걸리는 부하를 균등하게 분산시키지 못하는 단점이 있다.

2.2 연쇄적 질의 처리를 위한 대수 연산자

대수 연산자의 구현 기법은 일반적으로 임시 릴레이션(Temporary Relation)을 사용하는 실체화(Materialization) 기법과 반복자를 이용하는 연쇄적 처리(Pipelined Processing)기법의 두 가지로 나뉜다[3]. 후자의 반복자를 이용하는 기법은 그 수행단위가 각 대수 연산에 입력되는 튜플이며(Tuple at a time), Group By 나 Order By 등의 집합단위(Set at a time)연산이 필요한 경우를 제외하고는 임시 릴레이션을 만들지 않고 Open(), GetNext(), Close()의 세가지 인터페이스를 사용하여 질의를 연쇄적으로 수행한다. 각각의 대수 노드는 일정 크기의 입력 버퍼와 출력 버퍼를 가지며, 하위 노드의 출력 버퍼는 그대로 상위 노드의 입력 버퍼가 되며, 각 대수 연산자가 하나의 프로세스 혹은 쓰레드 등을 사용하여 구현된다. 따라서 모든 대수 연산자가 동시에 병렬로 수행되므로 각 노드의 수행 시간이 중첩되어 질의 처리에 걸리는 전체 시간이 감소되며, 멀티 프로세스와 많은 디스크를 가진 서버기종에서 최대한의 성능을 발휘할 수 있다.[6]

3. 분산 공간 데이터베이스의 병렬 연쇄 공간 죠인 기법

본 장에서는 둘 이상의 원격 사이트에 분산되어 있는 공간 데이터베이스에 대한 질의 수행시 포함되는 분산 공간 죠인 질의를 효율적으로 수행할 수 있는 병렬 연쇄 공간 죠인(PPSJ : Parallel Pipelined Spatial Join)기법에 대하여 설명한다. 본 기법은 질의 최적화를 위해 죠인 순서를 재조정하고, 최하위의 죠인을 PPSJ 기법으로 죠인을 수행하며, 나머지 노드들은 색인 기반 죠인을 이용하여 연쇄적으로 질의를 처리한다.

본 장에서는 우선 PPSJ 기법을 적용하기 위한 분산 공간 데이터베이스 시스템의 전체적인 구조와 필요한 메타 정보들을 설명하고, PPSJ 기법의 효율적인 수행을 위한 사이트간 죠인 순서 최적화 기법과 본 기법을 수행하기 위한 릴레이션 분할 알고리즘에 대해 설명한다.

3.1 메타 정보

각 사이트에서는 자신이 가진 공간 데이터 중 읽기 목적으로 공유할 수 있는 데이터의 메타 정보들을 전역 스키마 관리서버에 등록해 놓아야 한다. 각 서버가 등록해야 할 기본적인 메타 정보로는 서버 관련 정보로 IP 정보, 서버에서 주로 관리하는 데이터의 분류 정보, 사용자 권한 정보 등이 있으며, 공유하고자 하는 릴레이션들에 대한 정보로는 릴레이션의 이름, 어트리뷰트 정보를 등록해야 한다. 그리고 본 논문에서 제안하는 병렬 연쇄 공간 죠인 기법을 사용하기 위해서는 다음과 같은 정보를 추가적으로 필요로 한다.

[표 3-1] 공간 죠인 연산 비용 추정을 위해 필요한 메타 정보

메타 정보	설명
S_{midx}	공간 객체 MBR 의 중앙점 x 좌표의 총 합
S_{midy}	공간 객체 MBR 의 중앙점 y 좌표의 총 합
S_x	공간 객체 MBR 의 가로 너비의 총 합
S_y	공간 객체 MBR 의 세로 너비의 총 합
V	객체당 평균 점의 수

3.2 다수의 죠인을 포함하는 분산 질의 최적화

분산 데이터베이스 시스템에 대한 질의에는 다수의 사이트간의 질의가 포함되게 되며, 이러한 질의를 수행할 때는 사이트간 죠인의 수행 순서에 따라 질의 처리 시간에 상당한 차이가 나게 된다. 따라서 분산 질의에 대한 최적화 시에는 각 사이트간의 죠인 순서를 정하는 것이 중요한 주제가 되는데, 기존의 질의 최적화 기법들은 공간 데이터의 특징을 고려하지 않고 단순히 임시 결과 릴레이션의 전송 부하만을 고려하여 설계되어 분산 공간 데이터베이스에 대한 최적화 기법에는 적당하지 않다. 왜냐하면 네트워크의 전송 속도 향상등으로 전송부하가 줄어들고 있고 대용량의 복잡성을 가진 공간 데이터의 경우 전송 부하보다 죠인 조건인 공간 연산의 수행 비용이 부하의 더 많은 부분을 차지하며 병목 현상의 원인이 되기 때문이다.

본 장에서는 다수의 공간 죠인을 포함한 분산질의를 최적화하기 위해 릴레이션간의 공간 죠인 연산 비용을 추측하는 방법과 이를 이용해서 각 릴레이션들간의 죠인 순서를 최적화하는 알고리즘에 대해서 설명한다.

3.2.1 공간 죠인 연산 비용의 비교

본 기법에서는 두 릴레이션간의 죠인 연산 비용을 추정하기 위해서 관련 연구에서 제시한 수식들을 사용한다. 이 수식을 적용하기 위해서는 각 릴레이션의 S_{MBR} , T, V 를 계산해야 할 필요가 있다. 튜플의 개수 T 와 V 는 전역 스키마 관리 서버에서 얻을 수 있다. S_{MBR}는 단위 MBR 이 객체와 연산될 확률이므로, 릴레이션의 전체영역과 객체의 MBR 이 차지하고 있는 영역의 비율로써 계산해 낼 수 있다.

메타 정보중 공간 객체 MBR 의 가로 넓이의 총 합을 S_x, 공간 객체 MBR 의 세로 넓이의 총 합을 S_y 라 할 경우, 릴레이션 R 의 객체의 평균 면적 Size(R)는 다음과 같은 수식으로 계산될 수 있다.

$$Size\ (R) = \frac{S_x}{T} * \frac{S_y}{T} = \frac{S_x * S_y}{T^2}$$ (수식 3-1)

릴레이션의 전체영역이 (R_{mix}, R_{miny})와 (R_{max}, R_{maxy})로 나타내진다고 가정하면 S_{MBR} 는 다음과 같은 식으로 구해진다.

$$S_{MBR} = \frac{Size(R) * T}{\left(R_{max\,x} - R_{min\,x}\right) * \left(R_{max\,y} - R_{min\,y}\right)}$$ (수식 3-2)

위의 두 수식들과 관련 연구에서 소개한 [수식 2-1] ~ [수식 2-3]들에 의해, 두 릴레이션의 공간연산 비용은 다음과 같이 추정된다.

C_{TOTAL} = Size(R1) * T1 * (C_{FIO} (R2) + C_{RIO}(R2) + C_{RCPU}(R2)) (수식 3-3)

PPSJ 기법에서는 (수식 3-3)을 이용하여 각 릴레이션들 간의 죠인 비용을 계산하여 죠인 순서를 정하게 된다. 예외적으로, 두 릴레이션간의 죠인 조건이 없는 Cartesian Product 연산에 대해서는 그 죠인 순서를 가장 나중으로 늦추기 위하여 비용을 ∞ 로 한다.

3.2.2 공간 연산 비용을 이용한 죠인 순서의 최적화

본 기법에서는 이러한 분산 공간 죠인 연산의 특징을 이용한 최적화 알고리즘을 사용한다. 본 알고리즘은 여러 개의 죠인 조건들이 질의에 포함된 multivariable 질의(MVQ) 안의 릴레이션들을 [알고리즘 3-1]의 DISTRIBUTE 알고리즘을 통해 비공간 죠인이 되는 릴레이션들과 공간 죠인이 되는 릴레이션들로 나눈다. 질의의 실행은 일반적으로 비용이 적게 드는 술어부터 하는 것이므로, 비공간 죠인 집합에 대한 죠인을 먼저 수행 한 후, 이 결과를 다시 공간 죠인 집합의 최초 입력 릴레이션으로 사용한다.

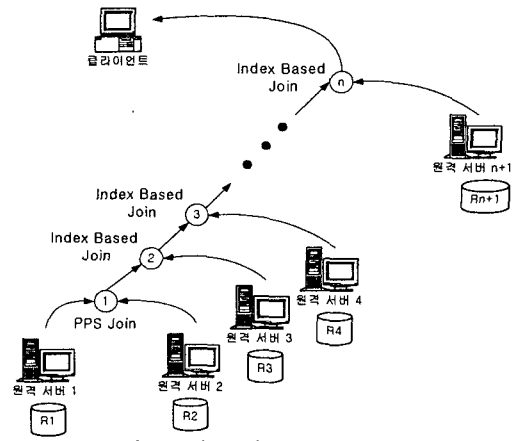
[알고리즘 3-1] ***DISTRIBUTE***
input: MVQ: multivariable query
output: RMVQ: reordered multivariable query
begin
  for each relation R_i in MVQ **do**
    if R_i has a aspatial join predcate with another relation **then**
    begin
      RMVQ ← R_i
      Remove R_i from MVQ
    end-if
  end-for
  for each relation R_i in MVQ **do**
    RMVQ ← R_i
  end-for
end.

공간 조건에 의해 죠인이 되는 사이트들은 공간 연산 수행시의 비용이 적게 드는 순서대로 질의의 수행순서를 정한다. 공간 죠인 집합들의 순서를 정하는 REORDER-SPATIAL 알고리즘은 다음과 같다.

[알고리즘 3-2] ***REORDER-SPATIAL***
input: SMVQ: spatial multivariable query
output: RSMVQ: *reordered spatial multivariable query*
begin
  RelationPair = (S_1, S_2)
  C_{min} ← ∞
  for *i* ← 1 **to** count of relations in SMVQ **do**
  begin
    for *j* ← *i*+1 **to** count of relations in SMVQ **do**
    begin
      C_{ij} ← $C_{TOTAL}(S_i, S_j)$
      if $C_{min} > C_{ij}$ **then**
      begin
        C_{min} ← C_{ij}
        RelationPair = (S_i, S_j)
      end-if
    end-for
  end-for
  RSMVQ ← *RelationPair.S1* **and** Remove *RelationPair.S1* from SMVQ
  RSMVQ ← *RelationPair.S2* **and** Remove *RelationPair.S2* from SMVQ

  while there is any relation in SMVQ **do**
  begin
    RelationPair = (R_1, S_1)
    C_{min} ← ∞
    for *i* ← 1 **to** count of relations in RSMVQ **do**
    begin
      for *j* ← 1 **to** count of relations in SMVQ **do**
      begin
        C_{ij} ← $C_{TOTAL}(R_i, S_j)$
        if $C_{min} > C_{ij}$ **then**
        begin
          C_{min} ← C_{ij}
          RelationPair = (R_i, S_j)
        end-if
      end-for
    end-for
    RSMVQ ← *RelationPair.S*
    Remove *RelationPair.S* from SMVQ
  end-while
end.

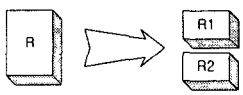
위의 REORDER-SPATIAL 알고리즘은 우선 SMVQ 의 릴레이션들 중에서 가장 죠인 비용이 적게 드는 릴레이션 쌍을 찾아 RSMVQ 로 이동시킨다. 다음으로 RSMVQ 의 각 릴레이션과 SMVQ 의 각 릴레이션들 간의 죠인 중에서 가장 적은 비용으로 죠인이 가능한 SMVQ 의 릴레이션을 찾아 하나씩 RSMVQ 로 이동시킨다. 위의 알고리즘을 통해 서로 죠인이 되는 여러 개의 릴레이션들은 비용순서로 정렬되며, 각 사이트간의 죠인도 이와 같은 순서로 수행된다. 정렬된 릴레이션들 간의 죠인은 최초로 수행되는 죠인의 경우 본 논문에서 제안하는 병렬 연쇄죠인 기법(PPSJ)으로 수행되며, 나머지 릴레이션들은 모두 색인 기반 죠인을 이용하여 튜플단위로 죠인연산이 이루어 짐으로써 [그림 3-1]와 같이 연쇄적인 질의처리가 가능하게 된다.



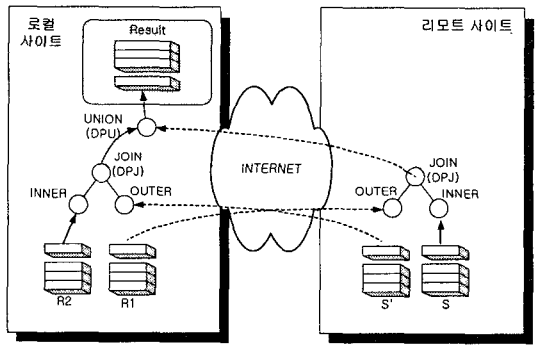
[그림 3-1] n 개의 죠인 연산의 연쇄적 처리

3.2.3 최하위 사이트간의 병렬 연쇄 죠인

본 논문에서는 여러 개의 죠인이 포함된 복잡한 질의에 대해 우선 죠인의 순서를 정한 후,그 중 최하위 노드에 대해 PPSJ 기법을 사용한다. PPSJ 기법은 크게 병렬 연쇄 죠인을 위한 분석 단계, 그리고 질의 실행 단계의 2 단계로 나뉘어 수행되며, 전체적인 수행 방법은 [그림 3-2]과 같다. PPSJ 기법의 분석 단계에서는 죠인에 참여하는 두 개의 릴레이션들 중 하나를 선택하여 릴레이션의 전체 영역을 2 개로 분할한다. 질의 실행 단계에서는 각 서버에 한 개의 영역을 할당하여 각 영역에 속하는 튜플들과 영역이 분리되지 않은 릴레이션과의 죠인을 동시에 수행한 후, 두 사이트에서 생성되는 죠인의 결과 튜플들을 병합하여 최종 결과를 생성하여 클라이언트에게 전송한다.



(a) 릴레이션의 분할에 의한 런의 생성



(b) 죠인의 수행

[그림 3-2]연쇄 병렬 분산 공간 죠인

질의 실행 단계에서는 새로운 대수 연산자를 사용하게 되는데, 이중 DPJ(Distributed Pipelined Join)는 분산 환경에서 원격 사이트로부터 릴레이션을 입력 받아 자신의 사이트에 있는 릴레이션과 죠인을 수행하는 노드이며, DPU(Distributed Pipelined Union)는 DPJ 와 마찬가지로 분산 환경에서 자신의 사이트에서 입력되는 결과 릴레이션과 원격 사이트에서 입력되는 결과 릴레이션을 UNION 연산하여 병합하는 대수 노드이다.

예를 들어 [그림 3-2 (b)]와 같이 지역(Local) 서버에 R 라는 릴레이션이 있고, 또 다른 원격(Remote) 서버에 S 이라는 릴레이션이 있으며, 지역 서버에 접속한 클라이언트가 R 과 S 와의 공간 죠인 질의를 요청하였다고 하자. 분

석 단계에서 지역 서버는 질의를 분석하여 질의의 참여하는 릴레이션들의 이름으로부터 리모트 사이트에 있는 S 라는 릴레이션과의 죠인이 필요하다는 것을 알아내어 전역 스키마 관리 서버에게서 S 라는 릴레이션을 가지고 있는 사이트의 정보와 S 의 메타 정보들을 얻어온다. 지역 서버는 이러한 정보들을 이용하여 R 과 S 중 하나를 선택한 후 영역을 분할하는데, 릴레이션 선택 및 영역 분할 기준점의 선택 등의 기준은 죠인의 수행 시 계산 비용 및 전송 비용이 최소가 되도록 하는 것이다.

질의 실행 단계에서는 분석 단계에 의해 [그림 3-2 (a)]에서와 같이 지역 서버에 있는 R 릴레이션이 R1 과 R2 로 분할되기로 결정되었을 경우, 지역 사이트에서는 R2 와 죠인 가능한 영역의 객체들의 집합인 릴레이션 S'를 원격 사이트에 요청하여 R2 와 S'의 공간 죠인 연산을 수행한다. 이와 동시에 원격 사이트에서는 S 와 지역 서버로부터 전송 받은 R1 과의 공간 죠인 질의를 수행하며, 각 사이트에서 수행한 두 개의 죠인 연산의 결과를 지역 서버에서 조합하여 클라이언트로 전송된다. 죠인 단계와 병합 단계는 순차적으로 수행되지 않고 연쇄적으로 수행되어 실행시간 중첩으로 인해, 사용자는 빠른 응답시간을 보장 받는다.

[그림 3-2]에서 설명하는 PPSJ 기법은 세미죠인 기반의 분산 공간 죠인 기법과 비교하여 죠인의 조건인 공간 연산 중 비용이 많이 드는 정제 연산의 수행을 두 서버에서 나누어 수행함으로써 각 서버에 CPU 비용을 분담시키며, 영역 분할을 통해 릴레이션을 분리한 후 죠인을 수행함으로써 공간 연산 중 여과 단계에서 인덱스 탐색 범위를 줄여 디스크 및 버퍼에 소요되는 비용을 절감할 수 있다.

3.3 병렬 죠인을 위한 릴레이션의 분할

PPSJ 기법의 분석 단계에서는 릴레이션을 분할하여 죠인 연산을 병렬로 처리할 수 있도록 하는데, 분할될 릴레이션을 잘 못 선택하거나 분할될 영역의 기준점을 잘 못 선택할 경우 어느 한쪽 죠인에 더 많은 부하가 편중되어 균등하게 병렬 죠인을 할 수 없다. 본 장에서는 PPSJ 기법에서 릴레이션을 분할할 때 두 개의 릴레이션 중 분할될 릴레이션을 선택하는 기준에 대한 설명을 하고, 선택된 릴레이션의 전체 영역을 두 부분으로 분할하는 기준점을 정하는 방법에 대하여 설명한다.

3.3.1 대상 릴레이션의 결정

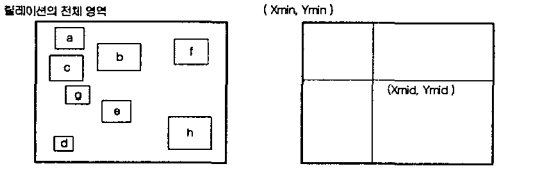
죠인이 되는 두개의 릴레이션중 분할할 릴레이션을 선택할 경우, 두 릴레이션의 크기가 많은 차이가 있으면 분할된 릴레이션 쪽이 더 작은 영역을 전송하게 되므로 크기가 더 큰 릴레이션을 분할하여야 하며, 두 릴레이션의 크기가 비슷한 경우는 어느 쪽을 분할해도 전송하는 데이터의 양은 같지만 공간 인덱스의 탐색 연산의 횟수를 줄이기 위하여 튜플의 수가 많은 릴레이션을 분할하여야 한다.

3.3.2 분할 영역의 결정

영역 단위로 공간 릴레이션을 분할하여 병렬 죠인을 수행하는 것은 죠인의 대상이 지역적으로 군집화 되기 때문에 R-Tree 와 같은 인덱스 사용 시 버퍼 사용의 효율을 높일 수 있다. 하지만 PBSM 죠인 등의 기존의 기법들에서는 특정 인덱스의 존재를 가정하고, 죠인에 필요한 부가적인 정보를 구축하기 위해 죠인 연산 전에 전처리 단계를 필요로 하며, 중복된 결과 튜플을 제거하는 후처리 단계도 필요하다.

본 논문의 PPSJ 기법에서 제안하는 영역 분할 기법은 빠른 응답속도와 연쇄적인 질의 처리를 위하여 계산비용이 적게 들고 중복 튜플이 생성되지 않도록 하는 것을 우선 목표로 한다. 본 기법에서는 다음과 같이 릴레이션이 차지하는 전체 영역을 둘로 분할하여 그 중 하나의 영역에 MBR 이 INTERSECT 되는 객체와 그렇지 않은 객체로 나눈다[9].

분할의 대상이 되는 릴레이션에 객체가 분포되어 있고 객체들의 MBR 들이 [그림 3-3 (a)]과 같이 분포되어 있다고 가정하자. 만일 영역을 X 축 혹은 Y 축을 따라서 정확하게 2 등분을 하여 객체들을 나눌 경우, 두개의 런 중 하나의 런에 객체들이 편중되는 현상이 발생하게 된다. 이러한 편중현상을 방지하기 위해서는 영역을 분할할 때 객체들의 분포 상황을 감안해야 한다. PPSJ 기법에서는 영역 분할의 기준점으로 모든 객체들의 MBR 의 중점(CPM : Central Point of MBRs)의 평균좌표(APC : Average Point Of CPMs)를 사용한다.



(a) 릴레이션 내의 객체들의 MBR의 분포 (b) MBR의 중점들의 평균 좌표

[그림 3-3] 릴레이션 내의 분포된 객체 MBR 들의 평균좌표

객체의 MBR 의 중점을 구하는 함수인 MidX( )는 다음과 같은 식으로 간단히 구현된다.

$$MidX(a) = \frac{X\max + X\min}{2} \quad \text{.....(수식 3-4)}$$

MidY(a) 역시 MidX 와 같은 식으로 구할 수 있으며, 이 두 가지 함수를 이용하여 릴레이션의 APC 좌표를 구할 수 있다. [그림 3-3 (b)]에서 APC 좌표 중 X 축 좌표값인 Xmid 는 다음과 같은 식으로 구할 수 있다.

$$Xmid = \frac{MidX(Envelope(a)) + MidX(Envelope(b)) + ... + MidX(Envelope(h))}{Cardinality} \quad \text{.....(수식 3-5)}$$

APC 의 Y 축 좌표값인 Ymid 역시 Xmid 와 같은 방법으로 구할 수 있으며, APC 좌표의 X 축 혹은 Y 축을 기준으로 하여 영역을 분할하여 객체의 MidX() 혹은 MidY() 값이 영역에 포함되는지를 체크하여 분할하면 객체들의 수를 균등하게 분할할 수 있다.

3.4 연쇄적인 질의 수행을 위한 대수 연산자

2 장에서 설명한 바와 같이, 질의 처리기에서 작성하는 대수 연산자에서 사용하는 알고리즘 중 실체화를 이용한 기법을 사용하게 되면 질의처리 시에 디스크 I/O 로 인해 발생하는 성능의 저하 및 초기 응답시간의 지연 현상이 발생하게 된다. 본 논문에서 제시하는 죠인 기법에서는 Group By 등의 특별한 경우를 제외하고는 각 대수 노드를 모두 반복자를 이용한 알고리즘을 이용한다. 본 장에서는 그 중 제안하는 죠인 기법에 사용되는 두 가지의 새로운 대수 노드를 설명하는데, 그 중 하나는 각 사이트에서 죠인을 수행하는 죠인 노드(DPJ)이고, 다른 하나는 각 사이트에서 수행된 결과들을 모아 상위 노드로 보내는 집계 노드(DPU)이다.

DPJ 노드는 단일 지역 서버에서 수행되는 일반적인 연쇄 죠인 알고리즘과 유사하며 단지 두 개의 입력 버퍼 중 한 개는 네트워크를 통해 입력 받는다는 것만이 다르다. DPU 노드 역시 단일 지역 서버에서 수행되는 일반적인 UNION 연산자와 같다. 단지 두 개의 입력 버퍼 중에서 원격 서버로부터 전송된 튜플을 가진 버퍼부터 우선적으로 반환한다는 점을 고려하여 설계한다.

4. 결론

본 논문은 분산 공간 데이터베이스 시스템에서의 원격 사이트간 공간 죠인 질의를 연쇄적이며 병렬적으로 수행하는 PPSJ 기법을 제안하였다. 본 기법은 질의에 포함된 여러 개의 공간 죠인을 공간연산의 비용에 따라 재정렬한 후, 최하위 죠인 연산의 대상이되는 두개의 릴레이션들 중 하나를 이등분하는 방법으로 공간 죠인 연산을 둘로 분할한 후 질의 수행에 참여하는 두 서버에게 분배한다. 각 서버는 분할된 공간 죠인 연산을 연쇄적인 방법으로 동시에 처리하며, 최종적을 지역서버에서 생성된 각각의 죠인 결과를 병합하여 최종 결과를 생성하여 클라이언트에게 전송한다.

향후 연구 과제로는 본 기법을 수행하기 위하여 릴레이션을 분할할 때 더욱 효율적인 분할을 수행하기 위하여, 대상 릴레이션을 결정하는 것과 분할 영역을 결정하는데 작용하는 각 서버들의 성능 차이, 릴레이션의 복사본의 존재 유무 등 다양한 요소들을 감안한 최적의 죠인 계획 생성에 관한 연구가 필요하다.

5. 참고 문헌

[1] M. Tamer Ozsu, Patrick Valduriez, " Principles of Distributed Database Systems," Prentice Hall, 1991.

[2] Zhe Li, Kenneth A. Ross, " PERF Join: An Alternative To Two-way Semijoin And Bloomjoin," Proc. of Int' l Conf. on Information And Knowledge Management, pp. 137 - 144 , 1995.

[3] Hector Garcia-Molina, Jeffry D Ulman, Jennifer Widom, "Database System Implementation," Prectice Hall, 2000.

[4] Kian Lee Tan, Beng Chin Ooi, David J. Abel, "Exploiting Spatial Indexes for Semijoin-Based Join Processing in Distributed Spatial Databases," IEEE Transaction on Knowledge And Data Engeering, Vol. 12, No. 6, pp. 920-937, 2000.

[5] Ashraf Aboulnaga, Jeffrey F. Naughton, "Accurate Estimation of the Cost of Spatial Selections," Proc. of the 16th International Conference on Data Engineering, Feb. – Mar., 2000.

[6] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts," Mc. Graw Hill, 1997.

[7] Guttman, A., "R-Trees: An Dynamic Index Structure for Spatial Searching," Proc. Of ACM SIGMOD Int' l Conf. on Management of Data, pp.47-57, 1984.

[8] 김홍연, " 공간 데이터베이스 시스템을 위한 고비용 술어 질의 최적화," 공학 박사 학위 논문, 인하대학교, 1999.

[9] Donovan A. Schneider, David J. Dewitt, "A Performace Evaluation of Four Parallel Join Algorithms in a Shared Nothing Multiprocessor Environment," Proc. of ACM SIGMOD Int' l Conf. on Management of data, Vol. 18, Issue 2, pp. 110-121, June 1989.