클라이언트/서버 환경에서 효율적인 공간 질의 처리+

문상호*, 임덕성**, 반재훈***
*위덕대학교 컴퓨터공학과
**부산대학교 컴퓨터공학과, ***경남정보대학 인터넷상거래과
e-mail : shmoon@mail.uiduk.ac.kr

Efficient Spatial Query Processing in Client/Server Environments

Sang-Ho Moon *, Duk-Sung Lim**, Chae-Hoon Ban***
*Dept. of Computer Engineering, Uiduk University,
**Dept. of Computer Engineering, Pusan Nat'l University,
***Dept. of Internet Commerce, Kyungnam College of Information & Technology

요 약

클라이언트/서버 환경에서 뷰 실체화 기법을 이용한 공간 질의 처리 방법은 동일한 질의에 대하여 실체화된 뷰를 이용하므로 처리 비용과 시간을 감소시킬 수 있다. 그러나 뷰 실체화는 서버 데이터베이스의 변경에 따른 뷰 일관성 유지 비용이 추가로 발생하므로, 공간 데이터베이스 변경 등 경우에 따라서는 뷰 실체화를 이용한 공간 질의 처리 방법이 기존 방법에 비하여 더 많은 비용과 시간이 발생할 수도 있다. 따라서 본 논문에서는 이러한 문제점을 고려하여 클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위하여 기존의 질의 처리 방법과 실체화된 뷰를 이용한 질의 처리 방법을 혼용한 방법(hybrid approach)을 제시한다. 이 방법에서는 클라이언트에서 공간 질의 처리를 요청한 경우에 서버에서 공간 질의 재수행 비용과 실체화된 뷰의 일관성 유지 비용을 비교 평가하여 질의 처리 결과를 실체화된 뷰로 유지 여부를 결정한다.

1. 서론

일반적으로 공간 데이터베이스에서는 대용량의 복잡한 공간데이터를 가지고 있기 때문에 질의 처리에 많은 비용과 시간이 든다. 클라이언트/서버 환경에서 공간 질의 처리는 이러한 문제가 더욱 심각하다. 즉, 클라이언트/서버 환경에서 동시에 여러 클라이언트들이 서버에 공간 질의 요청을 하는 경우에는 I/O 연산, 질의 처리, 데이터 전송 등 서버에 집중적으로 많은 부하가 걸리므로 공간 질의 처리에 많은 어려움이 발생한다[7,9].

기존에 클라이언트/서버 환경에서 공간 질의 처리를 효율적으로 수행하기 위한 연구들이 수행되어 왔다[7,9]. 이 연구들은 대부분이 뷰 실체화(view materialization) 기법을 적용하여 공간 질의 처리 방법을 제시하였다. 이 방법에서는 클라이언트에서 요청한 공간 질의를 클라이언트에서 뷰로 정의하고 질의 처리 결과를 뷰 실체화 기법을 이용하여 저장해 두었다가, 차후에 동일한 공간 질의 요청이 있을 때 실체화된 뷰를 이용하여 바로 질의를 처리한다. 이 방법은 기존의 클라이언트/서버 환경에서 공간 질의 처리를 매번 서버에서 수행하는 것에 비하여 질의 처리에 대한 비용과 시간을 감소시킬 수 있으므로 효율적이다. 그러나 이 방법에서는 서버의 공간 데이터 변경에 따른 실체화된 뷰의 일관성을 유지하는 문제를 해결해야 한다.

실체화된 뷰의 일관성 유지(materialized view maintenance)를 위한 방법으로는 재수행(re-

+ 이 논문은 2000년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2000-003-E00243)

execution)과 점진적 변경(incremental update)이 있다[3,4,5,6]. 점진적 변경 방법이 재수행에 비하여 다소 효율적이지만, 점진적 변경을 위한 부가적인 정보와 변경 시간을 고려해야 한다. 따라서 클라이언트/서버 환경에서 실체화된 뷰를 이용한 공간 질의 처리는 실체화된 뷰의 일관성 문제를 신중하게 고려해야 한다. 세부적으로 공간 질의 처리와 실체화된 뷰의 일관성 유지 비용을 상대적으로 비교할 필요가 있다. 또한 서버의 공간 데이터베이스 변경이 자주 발생하는 경우에는 오히려 뷰 실체화를 이용한 공간 질의 처리 방법이 기존 방법에 비하여 더 많은 비용과 시간이 발생할 수도 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위하여 질의 재수행 기법과 뷰 실체화 기법을 혼용한 방법(hybrid approach)을 제시하고자 한다. 이 방법에서는 클라이언트에서 공간 질의 처리를 요청한 경우에, 서버에서 판단하여 공간 질의 처리를 수행한 후에 결과를 실체화된 뷰로 유지 여부를 결정한다. 서버에서 뷰 실체화 여부 판단은 공간 질의 처리 비용과 실체화된 뷰 유지 비용을 기준으로 한다. 이를 위한 구체적인 근거를 위하여 본 논문에서는 비용식을 산정한다. 이 비용식에는 공간연산자 수행 비용, 질의 대상이 되는 공간 객체 수 등이 주요 요소(factor)로 포함된다.

2. 뷰 실체화 및 일관성 유지 방법

2.1 공간 질의 분석

일반적인 공간 질의 예는 그림 1 과 같으며[4,7,8], 이 질의는 금정구에 속한 빌딩들의 이름과 버퍼링한 영역을 보여준다. 기존의 질의와 달리 공간 질의는 공간 객체에 대한 공간연산자를 필요로 한다. 공간연산자는 크게 기하연산자(geometry operator), 공간관련성연산자(spatial relationship operator), 기하생성연산자(geometry generation operator)로 구분된다[4,8,9]. 표 1은 공간연산자 종류를 보여준다.

```
Select  Building.name, buffer(Building, 50)
From    Building, Zone
Where Zone.name="금정구" and encloses(Zone, Building)
```

그림 1. 공간 질의 예

표 1. 공간연산자 종류

유 형	종류
기하연산자	area, distance, length, perimeter 등
공간관련성 연산자	adjacent_to, contains, encloses, meets, crosses, inside, nearest, within 등
기하생성 연산자	buffer, centroid, intersect, overlap, difference, split, merge 등

공간 질의에서 공간연산자의 사용 유형은 먼저 기하생성연산자는 Select 절에서, 기하연산자와 공간관련성연산자는 Where 절의 프레디킷으로 사용된다. 이러한 공간연산자들은 질의 처리 결과에 대한 뷰 실체화 판단을 위한 비용식에서 주요 요소로 사용된다.

2.2 뷰 실체화 및 일관성 유지

본 논문에서는 클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위하여 기본적으로 뷰 실체화 기법을 기반으로 한다. 이를 위한 뷰 실체화 및 일관성 유지 방법은 [9]에서 제시한 방법을 적용한다. 먼저 공간 질의에 대한 뷰 실체화를 위하여 기하생성연산자를 포함한 뷰 객체의 생성과 저장은 클라이언트가, 공간관련성연산자를 포함한 셀렉션(selection)은 서버가 담당하여 수행한다. 그리고 실체화된 뷰의 일관성 유지를 위하여 클래스 유도관련성과 뷰 유도관련성을 부가정보로 유지하여 지연 변경(deferred update) 방법을 이용하여 수행한다. 클라이언트/서버 환경에서 뷰 실체화 방법과 일관성 유지 방법에 대한 세부적인 내용은 [9]에 기술되어 있다.

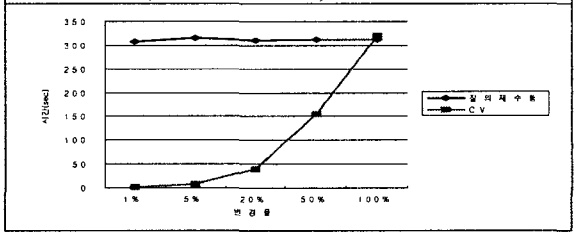
3. 질의 처리와 뷰 유지 비교를 위한 비용식 산정

3.1 주요 요소

비용식 산정을 위한 비교 대상은 질의 재수행 비용과 실체화된 뷰의 유지 비용이다. 이 비용들을 계산하기 위해서는 먼저 비용 계산에 직접적으로 영향을 미치는 주요 요소(factor)들을 추출해야 한다. 기본적인 요소로는 공간 질의의 대상이 되는 클래스들의 크기(객체들의 수)이다. 또한 대상 클래스들의 변화되는 객체들의 수를 나타내는 변경율도 고려해야 한다. 그리고 공간 질의에 포함된 연산자들도 고려해야 한다. 연산자들 중에서는 셀렉션, 프로젝션을 위한 속성연산자 및 공간연산자들이 대상에 포함된다.

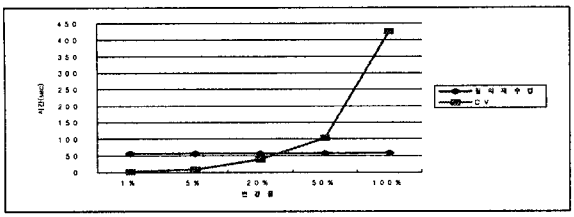
실체화된 뷰의 유지 비용에서 가장 중요한 요소는 서버 데이터베이스의 변경율이다. 세부적으로 서버 데이터베이스에서 실체화된 뷰와 직접적으로 관련된 소스 객체의 변경이 영향을 미친다. 그림 2 는 [공간 질의 1]과 [공간 질의 2]를 실체화된 뷰로 관리한 경우에 일관성 유지 시간과 질의 재수행 시간을 비교하여 보여준다[9].

```
Select  가로등.고유번호, Buffer(가로등,2.0)
From    가로등,행정동경계
Where   행정동경계.부코드="3000140000"
And     within(가로등,행정동경계)
```



(a) 공간질의 1

```
Select  가로등전선관.도로번호,가로등전선관.종류,
        Intersect_To_Buffer(가로등전선관,상수관로,2.0)
From    가로등전선관,상수관로
Where   CROSSES(가로등전선관,상수관로)
```



(b) 공간질의 2

그림 2. 뷰 유지 비용과 질의 재수행 비용과의 비교

그림 2 에서와 같이 서버 데이터베이스의 소스 객체(source object)들의 변경이 증가할수록 뷰 유지 비용이 비례적으로 증가한다. 따라서 소스 객체의 변경율이 증가할수록 뷰 유지 비용이 질의 재수행 비용보다 커지므로 비효율적이다.

그림 2(a), (b)를 비교하면 다소 상이한 결과를 알 수 있다. [공간 질의 2]는 소스 객체의 변경율이 약 25%이상이면 뷰 유지 비용이 질의 수행 비용보다 높아지지만, [공간 질의 1]은 약 90% 이상에서 뷰 유지 비용이 높아진다. 이러한 이유는 각 공간 질의에서 사용된 공간연산자가 다르기 때문이다. 즉 [공간 질의 1]에서 사용된 Within 연산자가 [공간 질의 2]에서 사용된 Crosses 연산자보다 비용이 많이 들기 때문이다.

클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위한 비용식에서는 소스 객체의 변경율 뿐만 아니라 공간연산자의 비용도 고려해야 한다. 공간연산자 중에서도 공간관련성연산자는 Where 절의 프레디킷으로 이용되면서 공간 조인(spatial join)을 수행하므로 다른 연산자들에 비하여 중점적으로 고려해야 한다.

3.2 뷰 유지 비용 분석

소스 객체의 변경으로 인한 클라이언트의 실체화된 뷰의 일관성을 유지하는 비용은 크게 서버와 클라이언트 비용으로 구분된다[9]. 서버에서는 뷰의 변경 여부에 대한 검사 비용과 일관성 유지가 필요한 경우에 공간관련성연산자의 수행 비용, 후보 객체들의 전송 비용이 필요하다. 반면에 클라이언트에서는 실체화된 뷰에 대한 점진적 변경 시간이 필요하다. 그림 3 은 [공간 질의 2]에 대한 뷰 일관성 유지 비용을 뷰 객체들의 변경율에 따라 분석한 것이다.

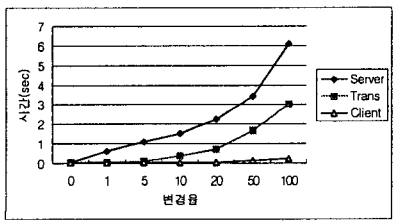


그림 3. 뷰 유지 비용의 분석

그림 3 에서 서버 비용은 대부분이 셀렉션 비용이 차지하고 있으며, 이 중에서도 공간관련성 연산자의 수행 시간이 많이 걸린다. 서버에서 검색된 후보객체들(쌍)에 대한 전송 비용은 서버 처리 비용의 10~50%를 차지하였다. 이것은 뷰 객체의 변경 비율이 증가할수록 후보 객체들의 수가 증가하기 때문이다. 소켓을 이용한 전송 패킷을 생성하기 위한 비용으로 후보 객체(쌍)의 수에 비례하고 후보 객체(쌍)의 수는 뷰 객체의 변경 비율에 비례한다. 그리고 클라이언트에서 수행되는 실체화된 뷰의 점진적 변경 비용은 서버 처리 비용의 4% 이하임을 알 수 있다.

3.3 비용식 산정

본 논문에서 제시하는 방법에서 핵심은 서버에서 클라이언트에서 요청한 공간 질의 처리 결과를 실체화된 뷰로 관리할 것인가를 판단하는 것이다. 이 판단을 위해서는 향후에 클라이언트에서 동일한 질의가 요청이 되었을 때, 질의의 재수행 비용($Cost_{RE}$)과 실체화된 뷰의 유지 비용($Cost_{CV}$)을 비교해야 한다. 결과적으로 실체화된 뷰 유지 비용이 질의의 재수행 비용보다 작으면($Cost_{CV} < Cost_{RE}$), 서버에서 질의 처리 결과를 실체화된 뷰로 관리하는 것이 효율적이다.

공간 질의를 세부적으로 관계 대수를 이용하면 다음과 같이 표현된다[9]. 여기서 SC_i 는 클래스, $geom$ 은 기하속성, $nongeom$ 은 비기하속성, GGO 는 기하생성연산자, SRO는 공간관련성연산자를 나타낸다.

$$\pi_{nongeom,GGO}(\sigma_{nongeom \wedge SRO}(SC_1 \times SC_2 \times \cdots \times SC_n))$$

따라서 공간 질의 수행에 대한 비용식은 다음과 같이 표현된다.

$$Cost_{RR} = Cost(\sigma_{nongeom}) + Cost(SRO) + Cost(\pi_{nongeom}) + Cost(GGO)$$

여기서 고려해야 할 것은 질의의 수행에 있어서 어떤 검색 조건을 먼저 수행하는냐에 따라 수행 시간에 큰 영향을 미친다[10]. 본 논문에서는 질의 수행 시간을 최소화하기 위하여 비기하속성에 대한 프레디킷을 먼저 수행하는 것을 전제로 한다. 이를 위하여 본 논문에서는 각 연산자의 연산자의 연산 시간과 조건문의 선택력을 이용하여 조건문의 검사 순서에 따른 연산 비용을 계산한 후에 최소 비용의 검사 순서를 이용한다. 그리고 질의 재수행 비용에서는 클래스들에 대하여 전체적으로 수행하므로, 객체 변경율은 비용식에 요소로서 반영하지 않는다.

서버 데이터베이스의 변경에 따른 뷰 유지 비용은 3.2 절에서 기술한 바와 같이 다음과 같이 표현된다.

$$Cost_{CV} = Cost_{Server} + Cost_{Trans} + Cost_{Client}$$
$$Cost_{Server} = Cost(\sigma_{nongeom}) + Cost(SRO)$$
$$Cost_{Client} = Cost(\pi_{nongeom}) + Cost(GGO)$$

여기서 고려해야 할 것은 서버에서 셀렉션과 공간관련성연산자의 연산 대상은 전체 객체들이 아니라 변경된 객체들에 제한된다는 것이다. 따라서 뷰 유지 비용에서는 객체 변경율이 중요한 요소가 된다. 그리고 서버에서 검색되는 후보 객체들은 매번 정확하게 계산하는 것이 어렵기 때문에 변경율에 대한 일정 검색 비율을 평균적으로 산출하여 반영한다.

공간 질의 수행 및 뷰 유지 비용식 산정에서 가장 중요한 것은 공간연산자의 비용이다. 3.1절에서 기술한 바와 같이 공간 연산자의 유형에 따라 비용이 다르고, 또한 같은 유형의 공간연산자라도 각 연산자별로 수행 시간이 차이가 난다. 더욱이 공간연산자에 적용되는 공간데이타의 유형(점, 선, 다각형)에 따라 연산 비용이 차이가 날 수 있다. 따라서 각 연산자의 평균적인 연산 비용을 측정할 필요가 있다. 이를 위하여 본 논문에서는 각 연산자들을 공간 객체들에 대하여 직접 연산을 수행한 후에 총 연산 비용과 수행 횟수를 이용하여 평균 연산 비용을 구한다. 이 방법에서는 한 연산을 여러 유형의 객체들에 대해 수행하므로 평균치를 이용한 계산을 보다 정확해진다.

4. 공간 질의 처리 방법

클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위하여 본 논문에서 제시하는 방법은 그림 4 와 같다. 클라이언트에서 공간 질의 요청이 있으면, 서버에서는 먼저 뷰 유지 비용($Cost_{CV}$)과 질의 수행 비용($Cost_{RE}$)을 비교한다. 뷰 유지 비용이 크면 서버에서는 공간 질의를 수행한 후에 결과를 클라이언트에 넘겨준다. 반면에 질의 수행 비용이 크면 질의 수행 결과를 실체화된 뷰로 저장한다. 이를 위하여 서버에서는 공간 질의의 셀렉션을 수행한 후에 후보 객체들을 클라이언트에 전송하고, 클라이언트에서는 후보 객체들에 대하여 프로젝션을 수행한 후에 결과를 실체화된 뷰로 저장한다.

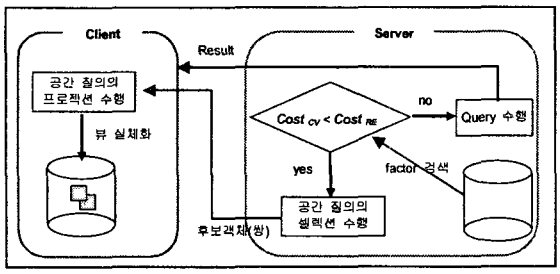


그림 4. 공간 질의 처리 과정

서버측에서 비용 평가를 위해서는 필요한 요소의 검색을 필요로 한다. 이 요소들을 공간연산자 비용, 클래스의 변경률, 클래스의 크기 등으로 시스템 카탈로그 또는 메타데이타 형태로 데이터베이스에 저장한다. 이 정보들은 데이터베이스 변경, 공간연산자 수행 등에 따라 계속 변경시킨다.

일반적인 공간 질의 처리 과정은 클라이언트의 요청 질의에 대하여 대응하는 실체화된 뷰가 존재하는지를 먼저 검색한다. 만약 존재한다면 바로 뷰를 이용하여 검색을 수행하고, 그렇지 않으면 그림 4 의 과정을 적용한다.

5. 결론 및 향후 연구

본 논문에서는 클라이언트/서버 환경에서 효율적인 공간 질의 처리를 위하여 뷰 실체화 기법과 질의 재수행 기법을 혼용한 공간 질의 처리 방법을 제시하였다. 이 방법에서는 질의 수행 비용과 뷰 유지 비용을 비교하여 비용을 최소화할 수 있는 장점이 있다. 앞으로는 실제 구현 및 실험을 통하여 각 공간연산자의 비용, 변경율, 선택율 등의 주요 요소들을 추출하여, 이 요소들을 반영한 성능 평가를 통하여 이 방법의 성능을 입증하는 것이 필요하다.

참고문헌

[1] Nick Roussopoulos , Hyunchul Kang "Principles and Techniques in the Design of ADMS± ", IEEE Computer, pp. 19-25, 1986.

[2] Nicholas Roussopoulos, "An incremental access method for ViewCache: concept, algorithms, and cost analysis", ACM Trans. Database System. Vol. 16, No. 3, pp. 535-563, 1991.

[3] Alex. Delis and N. Roussopoulos, "Techniques for Update Handling in the Enhanced Client-Server DBMS", IEEE TOKD, Vol . 10, No. 3, pp. 458-476, 1998.

[4] Sang-Ho Moon and Bong-Hee Hong, "Incremental Update Algorithms for Materialized Spatial Views by Using View Derivation Relationships", Proc. of DEXA, pp. 539-550, 1997.

[5] Ashish Gupta and Inderpal Singh Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications", Proc. of Int'l Conf. on Data Engineering, pp. 3-18, 1995.

[6] C. Souza dos Santos, "Design and Implementation of Object-Oriented Views", Proc. of DEXA, pp. 91-102, 1995.

[7] 김태연, 정보홍, 이재동, 배해영, " 서버 처리비용 분산을 위한 공간 뷰 클라이언트 실체화 기법" , 한국정보과학회 2001 봄 학술발표논문집, 28 권, 1 호, 2001.

[8] 문상호, 김동우, 반재훈, 홍봉희, "객체지향 공간 뷰의 설계 및 구현", 한국정보과학회 논문지(B), 26 권, 2 호, pp. 306-320, 1999.

[9] 임덕성, 반재훈, 문상호, 홍봉희, "공간 데이터베이스에서 클라이언트 뷰의 일관성 제어 기법", 한국정보과학회 논문지: 데이터베이스, 28 권, 2 호, pp.140-152, 2001.

[10] 선종복, 김진덕, 홍봉희, "객체지향 규칙 기반 지형 질의어 처리를 위한 최적화", 한국정보과학회 봄 학술발표논문집, 22 권 1 호, pp. 89-92, 1995.