클라이언트/서버 환경에서 효율적인 공간질의 처리를 위한 데이터 캐싱과 변경에 관한 연구

문상호*

A Study on Data Caching and Updates for Efficient Spatial Query Processing in Client/Server Environments

Sang-Ho Moon*

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

요 약

본 논문에서는 클라이어언트/서버 환경에서 클라이언트의 공간질의를 효율적으로 처리하기 위한 데이터 캐싱과 캐시된 데이터의 일관성 유지를 다룬다. 이를 위하여 먼저 클라이언트에서 데이터 캐싱을 위하여 실체화된 공간뷰를 이용한다. 그리고 서버의 데이터 변경에 따른 실체화된 클라이언트 뷰의 변경을 위하여 유도관련성을 이용한 점진적 변경 기법을 적용한다. 실체화된 뷰는 클라이언트의 공간질의를 효율적으로 처리할 수 있지만, 일관성 유지에 대한 오버헤드가 발생한다. 본 논문에서는 효율적인 공간질의 처리를 위하여 서버에서 공간연산자의 비용을 기반으로 질의 수행과 뷰 일관성 유지에 따른 비용식을 제시한다. 이를 기반으로 클라이언트에서 질의 요청시 비용을 평가하여 뷰의 실체화 여부를 서버에서 결정한다. 이 방법은 클라이언트/서버 환경에서 뷰의 실체화와 재수행 방법을 병행함으로써, 클라이언트의 공간질의를 효율적으로 처리할 수 있는 장점을 제공한다.

ABSTRACT

This paper addresses several issues on data caching and consistency of cached data in order to process client's queries efficiently in client/server environments. For the purpose, first of all, materialized spatial views are adapted in a client side for data caching, which is called client views. Also, an incremental update scheme using derivation relationships is applied to keep cached data of clients consistent with the rest of server databases. Materialized views support efficient query processing in a client side, however, it is difficult to keep consistent their contents by the update of a server database. In this paper, we devise cost functions on query execution and view maintenance based the cost of spatial operators so as to process client's queries efficiently. When request the client's query, in our query processing scheme, the server determines whether or not materialize it as a view due to evaluation using the related cost functions. Since the scheme supports a hybrid approach based on both view materialization and re-execution, hence, it should improve query execution times in client/server environments.

키워드

공간질의 처리, 뷰 실체화, 뷰 일관성 제어, 클라이언트/서버 환경

* 부산외국어대학교 컴퓨터공학부

접수일자 : 2003. 4. 12

Ⅰ. 서 론

일반적으로 공간 DB는 대용량의 복잡한 공간 데이터를 가지고 있기 때문에 질의 처리에 많은 비용과 시간이 든다. 특히, 클라이언트/서버 환경에서는 동시에 여러 클라이언트들이 서버에 질의 요청을 하는 경우에는 I/O 연산, 질의 처리, 데이터 전송 등 서버에 집중적으로 부하가 걸리므로 공간질의 처리에 많은 비용이 발생한다[1,2,8].

클라이언트/서버 환경에서 질의 처리를 위하여 스냅샷, 뷰 실체화 등을 이용한 데이터 캐싱에 관한 연구들이 수행되어 왔다[1,4,5]. 이 연구에서는 질의 처리 결과를 클라이언트에 저장하고, 차후에 질의 처리에서 캐싱된 데이터를 이용한다. 이것은 질의 처리를 매번 서버에서 수행하는 것에 비하여 비용을 줄일 수 있지만, 서버의 데이터 변경에 따른 캐시된 데이터의 일관성을 유지해야 한다.

본 논문에서는 클라이언트의 데이터 캐싱을 위하여 뷰 실체화 기법을 이용한다. 따라서 클라이언트/서버 환경에서 실체화된 뷰를 이용한 질의 처리는 뷰의 일관성 제어를 고려해야 하며, 이 방법으로는 재수행과 점진적 변경이 있다[3,4,9]. 뷰 실체화를 이용한 질의 처리에서는 질의 처리 비용과 실체화된 뷰의 일관성 유지 비용을 상대적으로 비교할 필요가 있다. 특히, 서버의 공간 DB 변경이 자주 발생하는 경우에는 오히려 뷰 실체화를 이용한 질의 처리가 기존 방법에 비하여 더 많은 비용과 시간이 발생할 수도 있다.

본 논문에서는 먼저 클라이언트/서버 환경에서 데이터 캐싱을 위하여 공간뷰 개념을 클라이언트 측에 적용한 클라이언트 뷰를 이용한다. 또한, 서버 데이타의 변경에 따른 클라이언트 뷰의 일관성 유지를 위하여 유도관련성을 이용한 점진적 변경 알고리즘[6]을 확장하여 적용한다. 그리고 효율적인 공간질의 처리를 위하여 질의 재수행과 뷰 실체화를 혼용한 방법을 제시한다. 이 방법에서는 클라이언트에서 질의 처리를 요청한 경우에, 서버에서 판단하여 질의 처리 결과를 실체화된 뷰로 유지 여부를 결정한다. 서버에서 뷰 실체화 여부는 질의 처리 비용과 실체화된 뷰 유지 비용을 기준으로 판단한다.

Ⅱ. 클라이언트 뷰의 실체화 및 일관성 제어

클라이언트 뷰는 클라이언트/서버 환경에서 자주 사용되는 공간질의를 클라이언트측의 뷰로 정의한 것이다[8]. 이 뷰 정의에서는 공간연산자를 필요로 하며, 이 연산자는 기하연산자, 공간관련성연산자, 기하생성연산자로 구분된다. 공간뷰 정의에서 기하생성연산자는 Select절에서, 기하연산자와 공간관련성연산자는 Where절의 프레디킷으로 사용된다. 표 1은 공간연산자 종류를 보여준다.

표 1. 공간연산자 종류
Table 1 Classification of Spatial Operators

유형	종류
기하연산자	area, distance, length, perimeter 등
공간관련성연산자	encloses, overlaps, meets, crosses 등
기하생성연산자	buffer, overlap, split, merge 등

클라이언트 뷰는 실체화 과정에서 셀렉션(σ)과 프로젝션(π)을 누가 담당하냐에 따라 분류된다[8]. 본 논문에서 적용하는 클라이언트 뷰는 뷰의 실체화 및 일관성 제어를 클라이언트와 서버가 분담하는 형태로, 뷰의 실체화 과정은 그림 1과 같다. 이 과정에서 생성되는 유도관련성은 실체화된 클라이언트 뷰의 일관성 제어에 이용된다.

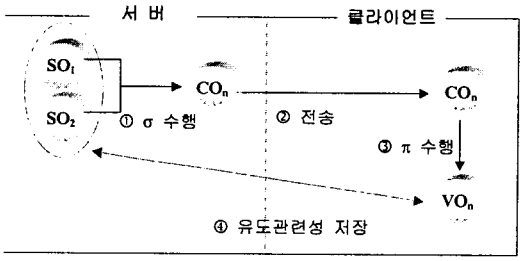

그림 1. 클라이언트 뷰의 실체화
Fig. 1 Client View Materialization

서버에서 소스객체가 변경될 경우에 관련된 클라이언트 뷰 객체의 일관성 제어가 필요하다. 뷰의 일관성 제어 방법으로는 재수행과 점진적 변경이 있다. 점진적 변경이 재수행에 비하여 효율적이지만, 변경을 위한 부가적인 정보가 필요하며

알고리즘이 복잡하다. 본 논문에서는 데이터베이스의 변경에 따른 클라이언트 뷰의 일관성 유지를 위하여 점진적 변경 기법을 적용한다.

뷰의 일관성 제어를 위하여 부가 정보로서 유도관련성을 이용하며, 이 유도관련성으로는 클래스 유도관련성과 뷰 유도관련성이 있다. 또한, 뷰의 일관성 검사 및 변경을 위하여 클라이언트와 서버 간의 일련의 프로토콜을 이용한다. 이러한 부가 정보와 프로토콜을 기반으로 실체화된 클라이언트 뷰의 일관성 제어를 수행하며, 관련된 세부 내용은 [8]에서 자세히 기술되어 있다.

III. 공간질의 처리를 위한 비용 분석

질의 처리를 위한 비용 분석에서 고려할 대상은 질의 수행과 실체화된 뷰의 유지 비용이다. 이 비용들을 산정하기 위해서는 직접적으로 영향을 미치는 주요 요소들을 결정해야 한다. 먼저 질의에서 셀렉션, 프로젝션을 위해 포함되는 기하생성연산자, 공간관련성연산자 등의 공간연산자들을 고려해야 한다. 그리고 질의 대상이 되는 객체들의 수와 해당 객체들의 변경율을 기본적인 요소로 고려해야 한다. 이 중에서 질의 수행 및 뷰 유지 비용식 산정에서 가장 기본적인 것은 공간연산자의 비용이다. 이것은 공간연산자의 유형에 따라 비용이 다르고, 같은 유형이라도 각 연산자별로 수행 시간이 차이가 나기 때문이다. 또한, 공간연산자에 적용되는 공간데이타의 유형(점, 선, 면)에 따라 비용이 차이가 날 수 있다.

공간질의를 관계대수를 이용하면 다음과 같이 표현된다[8]. 여기서 SC_i는 소스클래스, *geom*은 기하속성, *ngeom*은 비기하속성, *GGO*는 기하생성연산자, *SRO*는 공간관련성연산자를 나타낸다.

$$\pi_{ngeom,GGO}(\sigma_{ngeom \wedge SRO}(SC_1 \times \ldots \times SC_n))$$

그리고 이 관계대수를 기반으로 공간질의 수행에 대한 비용식은 다음과 같다.

$$Cost_{RE} = Cost(\sigma_{ngeom}) + Cost(SRO) + Cost(\pi_{ngeom}) + Cost(GGO)$$

여기서 어떤 검색 조건을 먼저 수행하는냐에 따라 수행 시간에 큰 영향을 미친다. 본 논문에서는 질의 수행 시간을 최소화하기 위하여 비기하속성에 대한 프레디킷을 먼저 수행한다. 이를 위하여 각 연산자의 수행 시간과 조건문의 선택력을 이용하여 조건문의 검사 순서에 따른 연산 비용을 계산한 후에 최소 비용의 검사 순서를 이용한다.

소스객체의 변경으로 인한 클라이언트 뷰의 일관성 유지는 서버와 클라이언트 비용으로 구분된다. 서버에서는 뷰 객체의 변경 여부에 대한 검사와 일관성 유지가 필요한 경우에 공간관련성연산자의 수행, 후보 객체들의 전송 비용이 필요하다. 반면에 클라이언트에서는 실제 뷰 객체에 대한 변경 시간이 필요하다. 이 비용식을 표현하면 다음과 같다.

$$Cost_{CV} = Cost_{서버} + Cost_{전송} + Cost_{클라이언트}$$
$$Cost_{서버} = Cost(\sigma_{ngeom}) + Cost(SRO)$$
$$Cost_{클라이언트} = Cost(\pi_{ngeom}) + Cost(CGO)$$

여기서 고려해야 할 것은 서버에서 셀렉션과 공간관련성연산자의 연산 대상은 전체 객체들이 아니라 변경된 객체들에 제한된다는 것이다.

IV. 공간질의 처리 기법

공간질의 처리를 위하여 제시하는 기법의 핵심은 클라이언트에서 요청한 질의 처리 결과를 실체화된 뷰로 생성할 것인가를 판단하는 것이다. 클라이언트에서 동일한 질의가 요청이 되었을 때, 질의의 재수행 비용($Cost_{RE}$)과 뷰의 유지 비용($Cost_{CV}$)을 비교해야 한다. 결과적으로 $Cost_{CV}$가 $Cost_{RE}$보다 작으면, 클라이언트에서 질의 처리 결과를 실체화된 뷰로 관리하는 것이 효율적이다.

공간질의 처리 기법의 전체적인 수행 과정은 그림 2와 같다. 클라이언트에서 공간질의 요청이 있으면, 서버에서는 먼저 뷰 유지 비용과 질의 수행 비용을 비교한다. 뷰 유지 비용이 크면 서버에서는 질의를 수행한 후에 결과를 클라이언트에 넘겨준다. 반면에 질의 수행 비용이 크면 실체화된 뷰로 저장한다. 이를 위하여 서버에서는 질의의

셀렉션을 수행한 후에 후보 객체들을 클라이언트에 전송하고, 클라이언트에서는 후보 객체들에 대하여 프로젝션을 수행한 후에 결과를 실체화된 뷰로 저장한다.

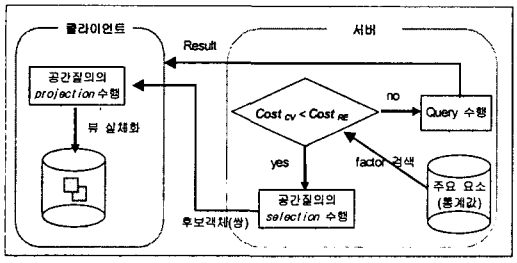

그림 2. 공간질의 처리 기법
Fig. 2 Spatial Query Processing Scheme

질의 처리를 위한 알고리즘은 그림 3과 같다. 먼저 질의를 분석하여 소스클래스, 연산자 등의 정보를 추출한 후에, 이 정보를 이용하여 질의 수행과 뷰 유지 비용을 산정한다. 이때, 비용식의 주요 요소들은 서버 데이터베이스에서 검색한다. 이 요소들은 공간연산자 비용, 해당 객체 수, 객체 변경률 등으로 시스템 카탈로그 형태로 데이터베이스에서 관리되며, 데이터베이스 변경, 공간연산자 수행 등에 따라 계속 변경시킨다. 그리고 결과를 비교하여 뷰 유지 비용이 작으면 질의 결과를 뷰로 실체화하고, 그렇지 않으면 질의 결과를 바로 클라이언트로 전송한다.

```
SQj := a spatial query
Ck := a client requesting to execute SQj
Procedure Query_Processing(SQj)
begin
  Infosqj := Analyze_Query(SQj)
  CostRE := Evaluate_Cost_Query_Exe(Infosqj)
  CostCV := Evaluate_Cost_View_Maintain(Infosqj)
  if (CostCV < CostRE)
    begin
      tmp_result := Execute_Query_Selection(SQj)
      Send_Query_Tmp_Result(Ck, tmp_result)
    end
  else
    begin
      result := Execute_Query(SQj)
      Send_Query_Result(Ck, result)
    end
end
```

그림 3. 질의 처리 알고리즘
Fig. 3 Query Processing Algorithm

일반적인 질의 처리 과정은 먼저 클라이언트에서 질의에 대응하는 뷰가 존재하는 지를 먼저 검색한다. 만약 존재한다면 바로 실체화된 뷰를 이용하여 질의를 수행하고, 그렇지 않으면 질의를 서버로 전송하여 그림 2와 같은 과정을 적용한다.

V. 성능 평가

성능 평가에서는 직접 질의 처리와 실체화된 뷰를 이용한 질의 처리를 중점적으로 비교한다. 또한, 소스객체의 변경율에 따른 뷰 일관성 유지와 직접 질의 처리에 대한 수행 시간도 비교 평가한다. 본 논문에서 제시한 공간질의 처리 기법은 상용 객체지향 GIS S/W인 고딕(Gothic)에서 개발하였다[7]. 본 논문에서의 실험 환경은 디지탈사의 ALPHA Station 500/333으로써 CPU속도는 333MHz이고 메모리는 128M이다.

5.1 실험 평가 대상

먼저 실험 데이타는 전체 16324개의 객체이며, 세부적으로 점 객체가 174개, 선 객체가 13821개, 면 객체가 2329개로 구성되어 있다. 그림 3은 성능 평가를 위해 사용한 실험 데이터이다.

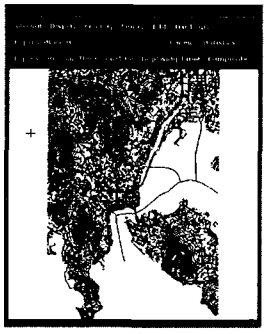

그림 3. 실험 데이타
Fig. 3 Experimental Data

실험에서 사용된 공간질의들의 정의는 그림 4와 같다. 세부적으로 [공간질의 1, 2]는 2196개의 Building 객체와 4개의 Zone 객체를 대상으로 하며, encloses, buffer 공간연산자를 수행한다. 그리고 [공간질의 3]은 417개와 127개의 Building 객체

를 대상으로 하며, overlaps와 overlap 공간연산자를 수행한다.

[공간질의 1]	Select Building.name, Building.geom From Building, Zone Where Zone.name = "Nam-Gu" and encloses(Zone, Building)
[공간질의 2]	Select Building.name, buffer(Building, 50) From Building, Zone Where Zone.name = "Nam-Gu" and encloses(Zone, Building)
[공간질의 3]	Select overlap(Building1, Building2) From Building1, Building2 Where overlaps(Building1, Building2)

그림 4. 실험을 위한 예제 공간질의
Fig. 4 Sample Spatial Query for Experimental Tests

5.2 질의 처리에 대한 성능 평가

예제 질의에 대하여 직접 질의 처리와 뷰를 이용한 질의 처리의 수행 시간을 비교 측정하였다. 세부적으로 실체화된 뷰를 이용하는 경우에는 클라이언트에서 뷰 객체들을 대상으로 검색하면 되지만, 그렇지 않은 경우에는 서버 데이터베이스에 대하여 질의를 직접 수행해야 한다.

그림 5는 질의들에 대하여 각 방법에 따른 질의 수행 결과를 보여준다. 이 결과에서 뷰를 이용한 질의 처리가 직접 질의 처리에 비하여 매우 빠르게 수행됨을 알 수 있다. 이것은 뷰를 이용한 질의에서는 실체화된 뷰 객체의 검색으로 질의를 수행할 수 있지만, 직접 질의 처리에서는 질의내의 공간관련성 연산과 기하생성 연산을 전체나 일부 소스 객체들에 대하여 수행하기 때문이다.

[공간질의 1, 2]는 비슷한 유형의 질의이지만 시간 차이가 많이 나는 것은 [공간질의 2]가 buffer 연산자를 추가로 수행하기 때문에 직접 질의 처리의 시간이 증가하게 된다. 반면에 뷰를 이용한 질의 처리에서는 실체화된 뷰 객체내에 기하 데이타를 가지고 있기 때문에 검색 시간이 거의 동일하다. [공간질의 3]이 [공간질의 2]에 비하여 수행시간 차이가 많이 나는 이유는 질의내의 overlaps와 overlap 연산자가 encloses와 buffer 연산자에 비하여 비용이 많이 들기 때문이다. 즉, 질의 처리시 소스객체에 적용되는 공간연산자의 유형에 따라 시간 차이가 많음을 알 수 있다.

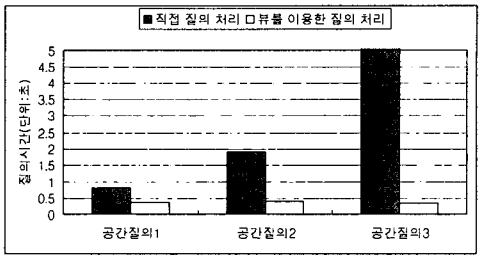

그림 5. 공간질의 수행 시간 비교
Fig. 5 Comparison of Query Execution Time

그림 6은 [공간질의 2]에 대하여 DB 크기를 증가시키면서 질의 수행 결과를 보여준다. 여기서 DB 크기의 증가를 위하여 질의의 대상이 되는 소스객체들의 수를 변화시켰다. 실험 결과에서 객체 수가 증가하면서 시간 차이가 비례적으로 커지는 이유는 직접 질의 처리에서 encloses 연산자를 이용한 공간조인의 대상이 되는 소스객체들의 수가 증가하고, 또한 buffer 연산자를 수행해야 할 후보 객체들의 수가 많아지기 때문이다. 반면에 뷰를 이용한 질의 처리에서는 뷰 객체만을 대상으로 하므로 수행 시간의 차이가 크지 않음을 알 수 있다.

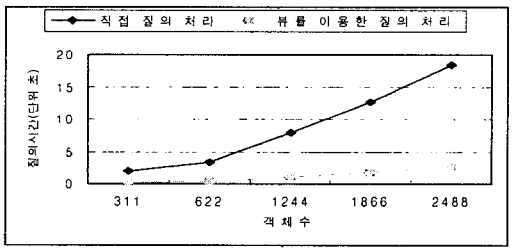

그림 6. DB 크기에 따른 [공간질의 2]의 수행 시간
Fig. 6 Execution Time of Query 2 for varying DB Size

그림 7은 DB 크기 증가에 따른 [공간질의 3]에 대하여 수행 시간을 비교한 결과이다. 이 결과에서도 앞에서와 마찬가지로 뷰를 이용한 질의 처리가 직접 질의 처리에 비하여 매우 빠름을 알 수 있다. 그림 6의 실험 결과와 비교해 볼 때, 두 실험 간에 시간 차이가 많이 나는 이유는 적용되는 공간연산자의 유형이 다르기 때문이다. 즉, [공간질의 2]에 이용된 encloses와 buffer 연산자에 비하여 [공간질의 3]에 이용된 overlaps와 overlap 연산자의 비용이 크기 때문이다. 따라서 이 연산

자의 비용에 따라 소스 객체의 수가 많아질수록 수행 시간 차이는 더욱 커지게 된다.

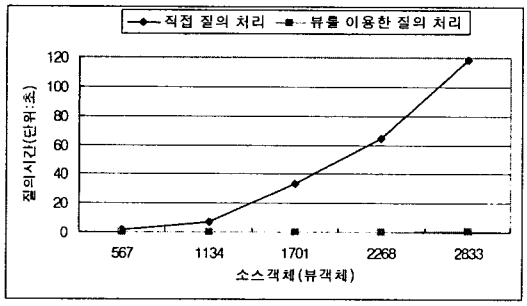

그림 7. DB 크기에 따른 [공간질의 3]의 수행 시간
Fig. 7 Execution Time of Query 3 for varying DB Size

5.3 뷰 일관성 유지에 대한 성능 평가

실체화된 뷰의 일관성 유지 비용에서 중요한 요소는 서버 DB의 변경율이다. 세부적으로 서버 데이터베이스의 소스 객체들에 대하여 변경을 한 후에, 뷰 일관성 유지 비용에 대하여 성능 평가를 수행하였다. 그림 8과 9는 [공간질의 2, 3]을 각각 클라이언트 뷰로 실체화한 후에, 서버의 소스객체들에 대한 변경율에 따른 뷰 일관성 유지 비용에 대한 실험 결과를 보여준다. 이 실험에서 [공간질의 2, 3]의 전체 소스객체는 각각 2488개와 2833개이고, 이 객체들에 대하여 1%, 5%, 10%, 50%, 100% 변경율을 대상으로 하였다.

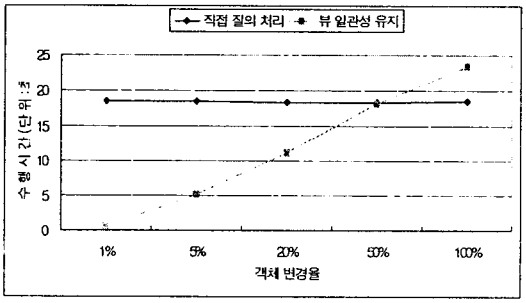

그림 8. 객체 변경율에 따른 [공간질의 2]의 수행 시간
Fig. 8 Execution Time of Query 2 for varying Object Modification Ratio

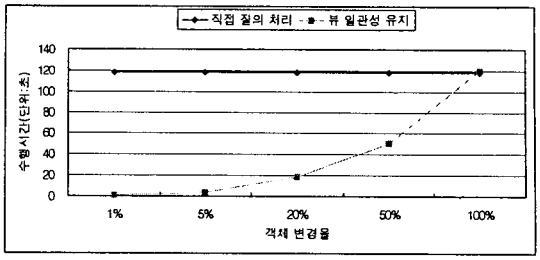

그림 9. 객체 변경율에 따른 [공간질의 3]의 수행 시간
Fig. 9 Execution Time of Query 3 for varying Object Modification Ratio

이 결과에서 서버 DB의 객체 변경이 많아질수록 뷰 유지 비용이 비례적으로 증가하지만, 직접 질의 처리는 거의 일정함을 알 수 있다. 이것은 실험에서 기존 소스객체의 기하데이타는 변경되었지만 전체 객체 수는 일정하기 때문이다. 그리고 실험 결과를 비교하면 다소 상이한 결과를 알 수 있다. [공간질의 2]는 객체의 변경율이 약 50%이상이면 뷰 유지 비용이 질의 수행 비용보다 높아지지만, [공간질의 3]은 약 95% 이상에서 뷰 유지 비용이 높아진다. 이러한 이유는 [공간질의 3]에서 사용된 공간연산자들이 [공간질의 2]에서 사용된 공간연산자들보다 비용이 많이 들기 때문이다. 또한, 객체 변경율이 증가하더라도 뷰 유지 비용이 질의 수행 비용보다 상대적으로 적은 이유는 유도관련성을 이용하여 변경된 소스객체와 관련된 뷰 객체만을 점진적으로 변경하기 때문이다.

5.4 종합 검토

앞의 실험 평가를 고려해 볼 때, 서버 DB의 변경이 비교적 적은 경우에는 뷰를 이용한 질의 처리가 직접 질의 처리에 비해 빠르게 수행됨을 알 수 있었다. 반면에 서버 DB의 변경이 많은 경우에는 공간연산자의 유형과 변경율에 따라 직접 질의 처리가 뷰를 이용한 질의 처리에 비해 성능이 우수했다. 이것은 소스객체의 변경이 많아질수록 뷰의 일관성 유지 비용이 증가했기 때문이다. 세부적으로 분석하면, 질의에 포함된 연산자가 저비용이고 변경율이 상대적으로 낮은 경우에는 뷰 일관성 유지 비용이 직접 질의 처리에 비하여 높았다. 반면에 고비용 연산자이고 변경율이 높은 경우에는 오히려 뷰 일관성 유지 비용이 질의 처리에 비하여 낮았다. 이것은 연산자

의 수행 비용이 높기 때문에 객체에 대한 질의 처리 시간이 상대적으로 많이 들기 때문이다.

따라서 본 논문에서 제시한 공간질의 처리 기법은 공간연산자 비용, 객체 변경율 등을 기반으로 질의 수행 비용과 뷰 유지 비용을 비교 분석하여 적합한 방법으로 질의를 수행하기 때문에 효율적일 수 있다. 즉, 직접 질의 처리와 뷰를 이용한 질의 처리를 주어진 여건에 맞게 적절하게 병행함으로써 질의를 빠르게 처리할 수 있다.

VI. 결 론

본 논문에서는 클라이언트/서버 환경에서 공간질의 처리를 위하여 먼저 데이터 캐싱을 위하여 공간 뷰 개념을 확장한 클라이언트 뷰를 적용하였다. 그리고 서버 데이터의 변경에 따른 뷰의 일관성 유지를 위하여 유도관련성을 이용한 점진적 변경 기법을 확장하여 적용하였다. 이를 기반으로 클라이언트/서버 환경에서 질의 처리를 위하여 뷰 실체화와 질의 재수행을 혼용한 질의 처리 기법을 제시하였다. 세부적으로 질의 수행과 뷰 유지에 대한 비용식을 공간연산자 등을 기반으로 산출하였고, 서버에서는 이 비용식을 이용하여 클라이언트 질의의 실체화 여부를 결정한다. 이 기법에서는 질의 처리를 위하여 뷰 실체화와 재수행을 병행함으로서 클라이언트의 공간질의를 효율적으로 처리할 수 있다.

감사의 글

본 연구는 2000년도 한국학술진흥재단의 신진교수 연구과제 지원에 의하여 이루어진 연구로서, 관계부처에 감사 드립니다.

참고 문헌

[1] N. Roussopoulos, An Incremental Access Method for ViewCache: Concept, Algorithms, and Cost Analysis, ACM Trans. Database Syst. vol 16, no 3, pp. 535-563, 1991.

[2] A. Delis and N. Roussopoulos, Techniques for Update Handling in the Enhanced Client-Server DBMS, IEEE TOKD, pp. 458-476, 1998.

[3] Latha S. Colby, Timothy Griffin, Leonid, and Libkin, Algorithms for Deferred View Maintenance, Proc. of Int'l Conf. on Management of Data, pp. 469-480, 1996.

[4] 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.

[5] Harumi A. Kuno and Elke A. Rundensteiner, Using Object-Oriented Principles to Optimize Update Propagation to Materialized Views, Proc. of Int'l Conf. on Data Engineering, 1996.

[6] Sang-Ho Moon and Bong-Hee Hong, Incremental Update Algorithms for Materialized Spatial Views by using View Derivation Relationships, Proc. of Int'l Conf. on DEXA, pp. 539-550, September, 1997.

[7] Laser Scan Ltd, Gothic Concepts, Training Course, 1995.

[8] 임덕성, 반재훈, 문상호, 홍봉희, 공간 데이터베이스에서 클라이언트 뷰의 일관성 제어 기법, 정보과학회논문지(B), 28권 2호, pp. 246-258, 2001.

[9] 문상호, 반재훈, 홍봉희, 실체화된 공간뷰의 일관성 유지를 위한 점진적 변경 알고리즘의 성능평가, 정보처리학회논문지, 9-D권 4호, pp. 561- 570, 2002.

저자 소개

문상호(Sang-Ho Moon)

한국기계연구원 정보지원실 연구원
부산대학교 컴퓨터공학과 공학석사부산대학교 컴퓨터공학과 공학박사위덕대학교 컴퓨터공학부 조교수
부산외국어대학교 컴퓨터공학부 조교수
※ 관심분야 : GIS, 공간DB, 데이터마이닝, GIS표준, 정보시스템 감리