공간 객체의 효율적 전송을 위한 교차영역의 제거

Removal of Intersected Region for Efficient Transmission of Spatial Objects

이경모(李卿牟)*, 박동선(朴東善)*, 김재홍(金在泓)**, 배해영(裵海英)***
Kyung-Mo Lee, Dong-Seon Park, Jae-Hong Kim, Hae-Young Bae

요 약 클라이언트-서버 환경의 공간 데이터베이스 시스템은 방대한 양의 공간 데이터 전송에 따르는 네트워크 부하가 크다. 이러한 환경에서 사용자는 빠른 초기 응답 시간을 위해 일부 영역의 공간 데이터를 요구하는 창(window) 질의를 사용한다. 화면 이동, 확대 및 축소 등으로 인한 일련의 창 질의는 유사한 영역의 데이터를 요구하며, 이는 이미 전송된 영역과 교차되는 영역의 데이터를 재전송하여 네트워크 부하를 가중시킨다. 이러한 문제는 생성된 질의 결과 중 클라이언트에 이미 전송된 데이터들을 제거함으로써 해결이 가능하다. 본 논문에서는 일련의 창 질의에 의해 발생하는 교차 영역을 제거하기 위한 공간 객체 관리자를 설계하고 구현한다. 공간 객체 관리자는 클라이언트로 전송된 객체의 식별자들을 관리하며, 객체 식별자의 비교를 통해 전송 여부를 판단하는 교차 영역 제거 기법을 사용하여 질의 결과로 생성된 공간 객체들 중 전송된 객체들을 제거한다. 본 논문의 공간 객체 관리자는 개방형 클라이언트-서버 공간 데이터베이스 시스템인 GEOMania Millennium server를 위해 구현하였다. 성능 평가를 통해 교차 영역의 제거가 동일 데이터의 중복 전송을 제거하여 네트워크 부하를 감소시키고 시스템의 전체적인 성능을 향상시킴을 보인다.

ABSTRACT Spatial database systems in client-server environment have network overload due to the large amount of spatial data transmission. Users use the window query that loads partial region of a whole map for quick response time in the environment. A series of window query such as screen movement, enlargement or shrinkage requires data in similar region and this increases network overload by re-transmitting the same data in intersected region with the earlier transmitted region. Removing the transmitted data from query results can solve this problem. In this paper, we design and implement a spatial object manager in order to remove the intersected region occurred by a series of window query. The spatial object manager manages the object identifiers of transmitted objects and removes transmitted objects from spatial objects of the query result by using the removal technique of the intersected region for the transmission and comparison.We utilize GEOMania Millennium server, an open client-server spatial database system, as spatial object manager in this paper. The result of the performance evaluation shows that the spatial object manager removes the transmission of the data redundancy, reduces network overload and improves the overall system performance.

키워드 : 클라이언트/서버 공간데이터베이스 시스템, 창(window) 질의, 교차영역제거, 공간객체전송

1. 서 론

공간 데이터를 수집하고 사용하는 사람과 조직의 수가 급격히 증대함에 따라 공간 데이터베이스†를 공유하거나 데이터 구축을 위한 공동 작업의 필요성이 증가하여 클라이언트-서버 환경을 위한 공간 데이터베이스 시스템의 연구가 활발히 진행되고 있다[5, 6, 14]. 방대한 공간 데이터를 관리하는 공간 데이터베

† 본 연구는 정보통신부의 대학 S/W연구센터 지원사업에 의하여 수행되었음.
* 인하대학교 전자계산학과 leekmo@hanmail.net/dseon@netian.com
** 영동대학교 컴퓨터공학과 교수 jhong@youngdong.ac.kr
*** 인하대학교 전산학과 교수 hybae@dragon.inha.ac.kr

이스 시스템은 클라이언트-서버 환경에서 데이터 전송에 따르는 네트워크 부하가 매우 큰 문제이다[5, 18].

공간 데이터의 효율적 관리와 네트워크 부하를 줄이기 위해 공간 데이터베이스 시스템은 공간 데이터를 주제별로 분류하여 레이어 단위로 관리하며, 클라이언트-서버 환경에서는 레이어를 단위로 하여 공간 데이터를 전송한다[16]. 그러나, 레이어가 구성하는 데이터 양이 방대할 경우 레이어 전체의 전송은 불필요한 데이터를 전송하여 네트워크 부하를 유발시키므로, 전체 영역 전송 방법[11, 18]보다 클라이언트가 필요로 하는 영역만을 전송하는 일부 영역 전송 방법[3, 11]을 사용하여 네트워크 부하를 줄인다.

클라이언트 시스템은 일부 영역 전송을 위해 화면 이동, 축소 및 확대 등의 브라우징 기능을 제공하며, 창(window) 질의[13]를 발생시킨다. 창 질의는 공간 데이터의 단순 출력 및 궁극적인 질의를 위한 준비 과정에서 자주 발생하며, 질의에 의해 선택된 영역은 기존의 영역과 교차하는 영역으로 인해 동일한 데이터가 재전송되는 문제가 있다. 이는 클라이언트의 중복 데이터의 제거 비용 뿐 아니라, 동일 데이터의 재전송에 의한 네트워크의 부하를 가중시킨다. 따라서, 창 질의의 처리 시, 교차된 영역의 데이터가 클라이언트에 재전송되지 않도록 하여 네트워크 부하를 최소화하여야 한다.

본 논문에서는 클라이언트-서버 환경의 공간 데이터베이스 시스템에서 일련의 창 질의에서 발생하는 교차 영역을 제거하여 네트워크 부하를 줄이는 공간 객체 관리자를 설계하고 구현한다. 교차 영역 제거 단계는 결과 집합과 이미 전송된 데이터 집합과의 교집합을 제거하는 단계이며, 공간 객체 관리자는 교차 영역 제거 단계의 빠른 수행과 효율적 관리를 위해 이미 전송된 데이터의 객체 식별자들을 비트맵 디렉토리 구조로 변환하여 관리한다. 공간 객체 관리자는 식별자 비교를 통해 창 질의의 결과 중에서 전송되지 않은 공간 객체들만을 전송함으로써 교차 영역의 재전송을 방지하여 네트워크의 부하를 줄이고 클라이언트에서의 중복 데이터를 검출하고 제거하는 과정을 생략하여 시스템의 전체적인 성능을 향상시킨다

본 논문의 구성은 다음과 같다. 2 장은 관련 연구로서 기존 클라이언트-서버 환경에서 공간 데이터베이스 시스템의 데이터 전송 방법의 문제점들을 기술하고, 3장에서는 객체 식별자를 이용한 교차 영역 제거 기법을 공간 객체 관리자를 설계하고 구현한다. 4장에서는 구현한 교차 영역 제거 기법의 성능을 평가하며, 마지막으로 5장에서 결론을 맺는다.

2. 기존 클라이언트-서버 환경의 공간 데이터베이스 시스템

2.1 질의 처리 방식

클라이언트-서버 시스템은 업무가 지역적으로 분산된 곳에서 이루어지고 동시에 여러 응용 업무에 공통된 데이터를 사용할 필요성이 있는 환경에서, 복잡한 분산 환경 시스템을 구축하지 않고 개발비용을 최소로 유지하고 성능을 극대화하기 위한 해결책으로 제시되었다[3, 5, 15]. 클라이언트-서버 시스템은 질의 처리 방식에 따라 질의 전송 방식(Query Shipping)과 데이터 전송(Data Shipping)방식이 있으며, 서버 중심의 시스템은 질의 전송 방식을 사용하며 클라이언트 중심의 시스템은 데이터 전송 방식을 많이 사용한다[5, 11, 15].

2.1.1 질의 전송 방식

질의 전송 방식은 표준 클라이언트-서버 시스템에서 사용하는 방법으로 클라이언트 시스템에서 발생하는 질의를 서버에 전송하여 서버에서 전송 받은 질의를 처리하고 그 결과를 클라이언트 시스템에 전송하는 방식이다[11, 15]. 질의 전송 방식을 이용한 클라이언트-서버 시스템은 질의와 질의 처리 결과만을 서버와 클라이언트 사이에서 전송하기 때문에 클라이언트와 서버의 통신 부하가 작다는 장점이 있다. 그러나, 클라이언트 시스템에서 발생하는 질의가 모두 서버에 전송되어 처리되므로 서버에 부하가 집중된다는 단점이 있다. 또한 클라이언트 시스템은 단지 사용자 인터페이스 기능과 통신 기능만을 수행하기 때문에 최근 그 성능이 개선되고 있는 클라이언트 시스템의 자원을 최대한 활용을 못한다는 단점을 갖고 있다.

클라이언트-서버 환경의 공간 데이터베이스 시스템은 클라이언트에서 공간 데이터의 출력 등의 GUI와 함께 이미 전송된 데이터의 관리와 간단한 질의를 처리하는 등 서버에서의 부하를 줄이려는 방향으로 발전하고 있다[11, 14].

2.1.2 데이터 전송 방식

데이터 전송 방식은 클라이언트 시스템에서 발생한 질의를 처리할 때, 질의 처리에 필요한 데이터를 서버로부터 전송 받아 클라이언트 시스템에서 질의를 처리하는 방식이다[11, 14]. 데이터 전송 방식을 이용한 클라이언트-서버 시스템은 질의 전송 방식을 이용하는 시스템의 서버 기능 중에서 많은 부분을 클라이언트 시스템으로 옮겨와 클라이언트 시스템의 자체 처리 기능을 크게 한 구조이다. 이와 같은 구조에서 서버는

단지 데이터를 관리하는 기능만을 수행하며 클라이언트 시스템에서 질의 처리를 수행한다. 따라서 서버에 부하가 집중되는 것을 방지하는 장점을 갖는다. 그리고, 질의 처리를 위해 서버로부터 데이터가 전송된 후 클라이언트 시스템에 전송된 데이터가 저장 관리된다. 따라서, 동일한 데이터를 이용하는 질의가 발생하였을 경우 서버로부터 데이터 전송을 요구하지 않고 클라이언트 시스템 자체에서 질의 처리를 할 수 있기 때문에 시스템 효율을 높일 수 있다.

그러나, 참조 데이터양이 크거나 지역성(Locality)이 낮아 데이터 요구가 빈번하게 발생하는 질의의 경우는 데이터 전송에 요구되는 비용이 커지는 문제점이 있다[11, 14]. 방대한 양의 공간 데이터를 관리하는 공간 데이터베이스 시스템은 참조 데이터양이 크며, 이에 따르는 데이터 전송에 의한 네트워크 부하가 매우 크다.

클라이언트-서버 환경의 공간 데이터베이스 시스템은 질의 전송 방식과 데이터 전송 방식을 혼용하여 클라이언트 시스템과 서버 시스템이 모두 질의 처리 기능을 가지는 방향으로 발전하고 있다.

2.2 데이터 전송 방식의 문제점

방대한 공간 데이터를 관리하는 공간 데이터베이스 시스템은 클라이언트-서버 환경에서 데이터 전송에 따르는 네트워크 부하가 매우 큰 문제이다[3, 18]. 이러한 시스템에서 공간 데이터를 전송하는 방식은 크게 전체 영역을 전송하는 방법과 일부 영역을 전송하는 방법으로 구분할 수 있다.

2.2.1 전체 영역 전송 방법

기존의 클라이언트-서버 공간 데이터베이스 시스템은 서버에서 클라이언트 시스템으로 데이터를 전송할 때, 전체 공간 데이터를 전송하는 방법이나 레이어 단위의 전송 방법을 사용한다[3, 16].

전체 공간 데이터를 전송하는 방식은 공간 데이터 전송이 완료된 후 클라이언트 시스템에서 모든 공간 데이터를 갖고 있게 된다. 따라서 이와 같은 방식은 데이터 전송 후에 발생하는 질의는 서버와의 통신 연결 없이 클라이언트 시스템 자체에서 처리할 수 있기 때문에 질의 처리가 매우 빠르다는 장점을 갖는다. 그러나, 모든 공간 데이터를 전송 받을 때까지 사용자가 대기해야 하는 단점이 있다.

레이어 전송 방법[3, 18]은 전체 공간 데이터를 각각의 주제에 따라 여러 개의 레이어로 분할 관리하며 서버에서 클라이언트 시스템으로 레이어 단위로 지리 데이터를 전송한다. 사용자가 요구한 레이어의 데이터만 서버에서 전송받기 때문에 전체 데이터를 모두 전송 받는 방식에 비해 초기 사용자 대기 시간이 줄어든다는 장점을 갖는다. 그러나, 클라이언트 사용자가 해당 레이어의 일부 영역의 데이터를 요구하여도 항상 레이어 전체가 서버로부터 전송되어야 하기 때문에 불필요한 데이터 전송에 의한 네트워크 부하가 문제가 된다.

2.2.2 일부 영역 전송 방법

사용자가 요구하는 영역의 데이터를 전송하는 일부 영역 전송 방법은 타일 분할에 의해 원하는 타일을 전송하는 방식과 질의를 통해 원하는 영역을 요구하는 방식이 있다.

타일 분할 방식은 레이어 전체를 일정한 크기의 작은 영역인 타일(tile) 단위로 물리적으로 구분하여 관리하여 클라이언트에서 필요로 하는 타일들을 전송하는 방식이다[3]. 전송되지 않은 타일만을 전송하여 동일한 영역이 재전송되는 것을 방지할 수 있으나, 단위 타일의 영역이 클 경우 실제 영역보다 더 큰 영역의 데이터를 전송하게 되며, 단위 타일의 크기가 작을 경우 여러 타일에 중첩되는 공간 객체가 증가하여 동일 데이터가 중복 전송되는 문제점이 있다.

질의를 통해 일부 영역을 요구하는 방식은 창 질의에 의해 꼭 필요한 영역만을 전송하게 된다. 그러나, 새로이 요구하는 영역이 기존의 영역과 교차하는 경우 동일한 데이터가 재전송되는 문제점이 있다.

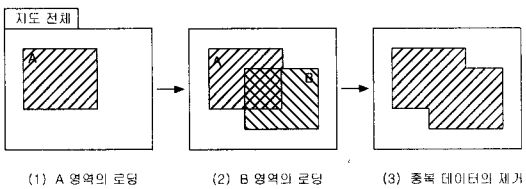


그림 1. 데이터의 중복 로딩

그림 1에서 클라이언트가 A영역의 데이터를 로딩한 상태에서 B영역의 데이터를 요구하면 A와 B가 교차되는 영역의 데이터는 중복 전송되며, 이는 네트워크 부하를 증가시킬 뿐 아니라 클라이언트에서 교차된 부분의 데이터를 제거해야 하는 추가 부담이 생긴다. 창 질의의 효율적인 처리를 위해 공간 인덱스[3, 12]에 의한 영역 질의 처리 방법들이 연구되었으나, 교차 영역 발생에 대한 충분한 고려가 이루어지지 않았다. 따라서, 창 질의의 효율적인 처리를 위해서는 교차 영역에 의해 발생하는 네트워크 부하를 최소화해야 한다.

본 논문에서는 이러한 문제점을 해결하기 위하여 창 질의에 대해 교차 영역 제거 기법을 지원하는 공간 객체 관리자를 설계하고 구현한다.

3. 공간 객체의 효율적 전송을 위한 공간 객체 관리자

본 장에서는 일부 영역을 전송하는 데이터 전송 방식의 문제점을 해결하기 위해, 창 질의의 처리 과정에서 동일한 데이터를 중복 전송하는 것을 방지하여 네트워크의 부하를 줄이는 객체 식별자를 이용한 교차 영역 제거 기법을 제안한다.

3.1 클라이언트-서버 환경에서 공간 데이터 전송 시 고려 사항

클라이언트에서 일부 영역의 데이터 요구는 특정 영역의 선택, 화면 이동, 축소 및 확대 등의 브라우징 기능으로 표현되며, 이는 창 질의를 발생시킨다. 창 질의는 단순 지도 출력이나 궁극적인 질의를 위한 준비 과정에서 발생하며 빠른 질의 처리를 요구한다. 그러나, 브라우징 기능의 잦은 사용은 이미 전송된 영역과 요구하는 영역의 중복으로 인해 동일한 데이터들이 재전송되어 클라이언트의 중복 데이터의 제거를 필요로 할 뿐 아니라, 네트워크 부하를 가중시킨다.

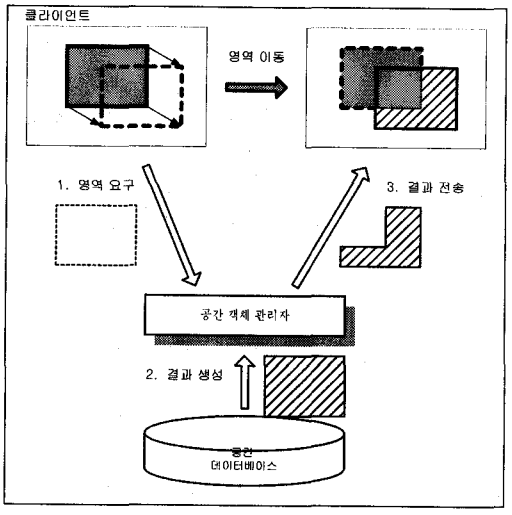


그림 2. 화면 이동에 따른 질의 결과 전송

이러한 문제를 해결하기 위해서는 창 질의를 처리시, 클라이언트로 동일한 데이터가 재전송되지 않도록 해야 한다. 그림 2은 화면 이동에 대한 창 질의가 발생하였을 때 중복된 영역을 서버에서 미리 제거하여 전송하는 과정을 보이고 있다.

그림 2에서 클라이언트 시스템의 사용자가 화면 이동 기능을 사용하면 클라이언트는 이동된 화면 좌표를 지도내의 좌표로 변환하여 공간 연산을 포함하는 창 질의를 서버로 전송한다. 서버의 질의 처리기는 공간 데이터베이스 내의 공간 객체를 읽어 주어진 질의 조건에 부합하는 결과 테이블을 생성한다. 결과 테이블을 구성하는 과정에서 공간 객체 관리자는 질의 조건을 만족하는 객체 집합과 이미 전송된 공간 객체의 집합의 교집합을 제거한다. 서버는 클라이언트에 전송되지 않은 객체들로 구성된 결과 테이블을 전송하며, 클라이언트는 별도의 중복 데이터 제거 과정 없이 화면에 출력할 수 있다. 그리고, 화면 축소에 따른 창 질의의 처리 과정도 화면 이동에서의 처리 과정과 동일하게 진행된다. 화면 확대의 경우는 화면 이동 및 축소와 달리 요구되는 영역의 데이터들이 이미 클라이언트에 존재하므로 클라이언트 시스템에서의 자체적인 처리가 가능하다.

클라이언트의 창 질의는 서버에서 window 연산 수행 단계와 교차 영역 제거 단계의 2단계를 반복적으로 수행하여 처리한다. window 연산 수행 단계는 레이어의 공간 객체가 요구하는 영역 내에 포함되는 지를 판별하는 공간 연산 수행 단계이다. 공간 객체와 선택 영역이 교차하는 지를 비교하여 교차할 경우 교차 영역 제거 단계로 전달한다. 교차 영역 제거 단계에서는 window 연산 처리 단계에서 얻어진 공간 객체가 이미 전송된 객체인지를 판별하여 제거하는 단계이다. 클라이언트에 이미 전송된 객체는 결과 집합에서 제거시키며, 전송되지 않은 공간 객체는 결과 집합에 포함시킨다.

위의 과정을 반복 수행하여 결과 집합을 구성하면 클라이언트에 전송되지 않은 데이터로 구성된 결과 테이블이 생성된다. 따라서, 클라이언트에 이미 전송된 데이터의 재전송을 방지할 수 있어 네트워크 부하를 줄일 수 있다.

3.2 공간 객체 식별자를 이용한 교차 영역 제거 기법

창 질의의 처리 단계에서 교차 영역 제거를 위해 클라이언트에 전송된 데이터 자체를 모두 관리하고 이들을 비교하는 것은 서버의 관리 부하가 크고 처리 속도도 현격히 떨어지게 된다. 따라서, 교차 영역 제거 단계의 효율을 위해서는 전송된 객체와 전송할 객체의

빠른 비교 방법과 효율적인 관리가 필요하다. 이러한 문제의 해결을 위해 공간 객체 관리자는 공간 객체를 대표할 수 있는 객체 식별자를 관리하여 교차 영역 제거 단계를 수행한다.

그림 3은 공간 객체 관리자에서 교차 영역을 제거하는 과정을 보이고 있다. 공간 객체 관리자는 창 질의의 처리 과정에서 선택된 공간 객체가 이미 클라이언트에 전송된 데이터인지를 검사한다. 즉, 공간 객체의 식별자와 클라이언트에 전송된 공간 객체의 식별자들을 비교하여 이미 존재하면 버리고, 존재하지 않으면 식별자를OID 리스트에 추가하고 공간 객체를 전송할 데이터로 결정한다.

공간 객체 관리자는 공간 객체 식별자로 물리적 객체 식별자를 사용하며, 교차 영역 제거 단계의 빠른 처리를 위해 공간 객체 식별자를 레이어 단위로 관리하고, 각 레이어를 페이지 단위의 비트맵을 구성하여 디렉토리 구조로 관리한다.

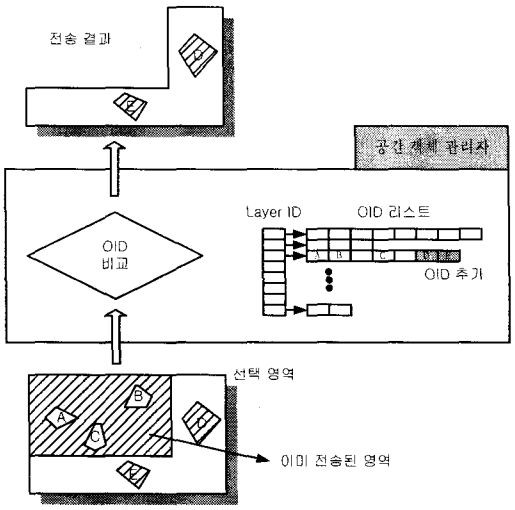


그림 3. 교차 영역 제거 과정

3.2.1 공간 객체 식별자

공간 객체 관리자는 공간 객체의 식별자를 위해 물리적 객체 식별자를 사용한다. 논리적 객체 식별자는 물리적 위치 정보를 포함하지 않지만, 물리적 객체 식별자는 데이터 파일에 레코드 형태로 저장되는 객체가 실제 디스크에 저장될 때의 물리적 위치 정보로 구성되며, 이를 통해 직접 객체에 접근할 수 있다. 물리적 객체 식별자의 사용은 삭제와 변경이 빈번히 발생하는 환경일 경우에는 객체 식별자 유지 관리로 인한 디스크 공간의 낭비와 성능 저하가 발생할 수 있으나, 대부분의 경우에 객체 식별자에 의한 직접 접근이 가능하므로 빠르게 객체에 접근할 수 있는 장점이 있다. 공간 데이터베이스 시스템은 일반 데이터베이스 시스템과 달리 데이터베이스를 구축한 후에는 갱신 연산이 비교적 적어 데이터의 물리적 위치 변동이 적으므로 공간 객체의 식별자로 물리적 객체 식별자가 적합하다.

공간 객체의 식별자는 다음과 같이 구성된다.

```
struct      SpatialOID {
long        PID;    // 페이지 식별자
short       Slot;   // 슬롯 번호
short       RSC;    // 재구성 슬롯 번호
};
```

공간 객체 식별자의 RSC는 동일 슬롯의 레코드가 삭제되고 다른 레코드가 삽입될 때 증가되는 값이다. 하나의 레이어는 여러 개의 페이지로 구성되며, 한 페이지 내의 최대 슬롯 번호는 그 페이지 내의 레코드의 수보다 크거나 같다.

3.2.2 공간 객체 관리자의 객체 식별자 관리 구조

교차 영역 제거 단계의 효율적 처리를 위해서는 공간 객체 식별자를 관리하기 위한 저장 공간의 효율과 객체 식별자간의 빠른 비교를 보장해야 한다. 이를 위해 공간 객체 관리자는 이미 전송된 객체의 식별자들을 레이어 단위로 관리하며, 각 레이어 내에서는 공간 객체 식별자들을 페이지 단위의 식별자 비트맵을 유지하여 관리한다. 그림 4은 공간 객체 관리자에서 전송된 데이터들의 객체 식별자를 관리하는 구조이다.

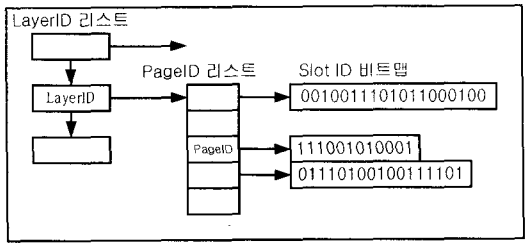


그림 4. 공간 객체 관리자의 객체 식별자 디렉토리

공간 객체 관리자는 이미 전송된 공간 객체들의 식별자들을 관리하기 위해 레이어 ID와 객체 식별자의 구성 요소인 페이지 ID를 이용해 디렉토리 구조로 관리한다. 단위 레이어를 위한 식별자 디렉토리는 레이어를 구성하는 물리적 페이지의 수 만큼 페이지 ID 리스트를 가지며, 각 페이지 ID는 페이지내의 최대 슬롯 값에 비례하는 슬롯 ID비트맵(bitmap)으로 구성

된다. 한 페이지 내의 SlotID 비트맵에서 i번째 항목이 1로 셋팅되면 이미 클라이언트에 전송되었음을 의미하며, 0으로 셋팅되어 있으면 아직 전송되지 않은 공간 객체임을 의미한다. 이와 같이 이미 전송된 공간 객체의 객체 식별자를 한 비트로 표현하여 저장 공간을 최소화한다.

공간 객체 관리자는 객체 식별자 디렉토리를 관리하여 교차 영역 제거 과정에서 선택된 공간 객체의 객체 식별자를 통해 손쉽게 전송 여부를 판단한다. 객체 식별자의 페이지 ID가 p이고 슬롯 ID가 s라고 할 때, 영역 연산이 수행되는 레이어의 ID를 통해 객체 식별자 디렉토리에 접근한다. 해쉬 함수에 의해 p에 해당하는 페이지ID 영역에 접근하고, 슬롯 ID 비트맵에서 s번째 비트를 체크한다. 교차 영역 제거 과정에서 s번째 비트가 1이면 이미 전송된 데이터이므로 버리고, 0이면 전송할 데이터이므로 1로 셋팅하고 결과 테이블에 저장한다. 이와 같이 관리하는 객체 식별자의 수와 관계 없이 동일한 접근 횟수에 의해 교차 영역 내의 객체인지를 손쉽게 판별할 수 있다.

3.3 공간 객체 관리자의 설계 및 구현

본 논문의 효율적 공간 객체 전송을 위한 공간 객체 관리자는 개방형 클라이언트-서버 공간 데이터베이스 시스템인 GEOMania Millennium server를 위해 구현하였다. GEOMania Millennium server 시스템 내에서 창 질의를 처리할 때 이미 전송된 데이터를 제거하여 질의 결과를 구성하는 공간 객체 관리자의 설계 및 구현에 대해 설명한다.

3.3.1 공간 데이터베이스 시스템 GEOMania Millennium server의 구조

GEOMania Millennium server는 공간 데이터와 공간 정보 처리 서비스의 표준 인터페이스를 제공하여 이기종 시스템과의 상호 운영성(Interoperability)을 지원하는 개방형 클라이언트-서버 공간 데이터베이스 시스템이다. 그림 5은 GEOMania Millennium server의 시스템 구조도이다. GEOMania Millennium server는 공간 데이터와 비공간 데이터를 하나의 저장 관리자로 관리하는 통합 구조이다. 질의 처리기는 공간 데이터의 관리를 위해 MiDAS [8] 저장 관리자에서 제공하는 데이터 타입을 확장하여 공간 데이터 타입을 지원하며 이를 위한 공간 연산 처리 모듈을 포함한다. 공간 연산 처리 모듈과 연동하는 공간 객체 관리자는 창 질의 처리 과정에서 이미 전송된 데이터가 재전송되는 것을 방지한다.

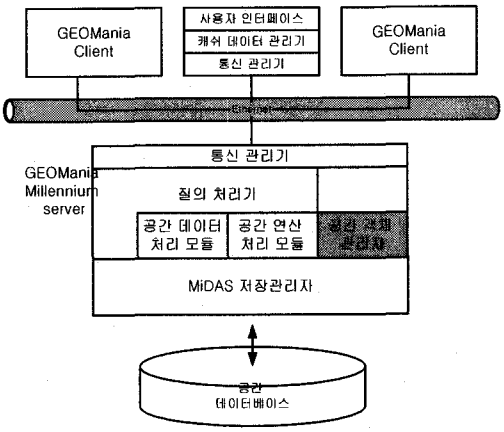


그림 5. GEOMania Millennium server의 전체 시스템 구조

GEOMania Millennium server는 클라이언트의 개별적인 관리의 효율성을 위해 클라이언트와 서버의 프로세스 구조를 일대일로 하였으며, 그 개념도는 그림 6과 같다. 서버 프로세스들의 동시성 제어 및 버퍼 관리 등은 공유 메모리를 통해 제어하며, 공간 객체 관리자는 각 서버 프로세스 내에 존재하여 대응하는 클라이언트에 전송된 데이터들을 관리한다. 클라이언트가 서버에 접속하면 서버는 자식(child) 프로세스를 생성하고, 자식 프로세스가 해당 클라이언트를 관리한다. 서버 프로세스 내의 공간 객체 관리자는 해당 클라이언트에 전송된 공간 객체들의 객체 식별자 디렉토리를 관리하여 공간 객체의 효율적 전송을 관리한다.

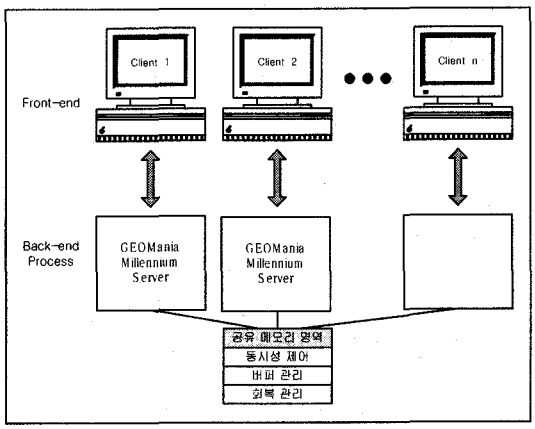


그림 6. GEOMania Millennium server의 클라이언트-서버 프로세스 구조

3.3.2 창 질의의 처리 과정

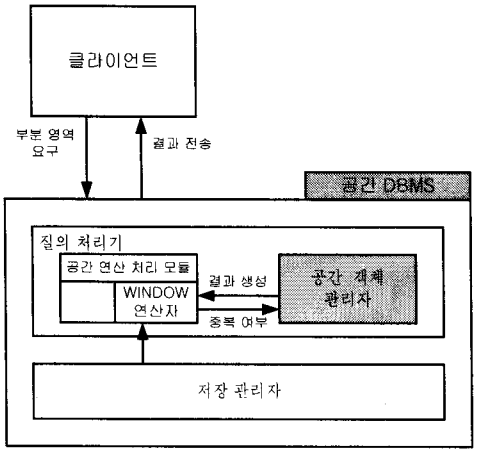


그림 7. 창 질의 처리 과정

창 질의를 처리하기 위해 질의 처리기는 window 연산과 공간 객체 관리자를 사용해 전송할 데이터와 전송하지 않을 데이터를 결정한다. 질의가 처리되는 과정은 그림 7과 같다.

클라이언트에서 일부 영역을 요구하는 창 질의가 오면 질의 처리기는 해당 레이어 내의 데이터를 저장 관리자로부터 로드한다. 공간 연산 처리 모듈은 window 연산을 이용하여 영역 연산 수행 단계를 행한다. 영역 내에 해당하는 데이터일 경우 공간 객체 관리자를 통해 교차 영역 검출 단계를 수행하여 이미 전송된 데이터인지를 판별한다. 만약, 이미 전송된 데이터이면 해당 데이터를 버리고, 전송되지 않은 경우에는 질의 결과 테이블에 저장한다. 질의 처리기는 질의가 끝난 후 질의 결과 테이블을 클라이언트에 전송한다.

그림 8은 창 질의의 수행을 위한 흐름도이다. 질의 처리기는 저장 관리자로부터 해당 레이어에 존재하는 공간 객체를 로드한다.

영역 연산 수행 단계에서 window 연산을 수행하여 공간 객체의 MBR이 원하는 영역의 MBR과 교차하는지를 검사하고, 교차하지 않으면 다음 공간 객체를 로드한다. 공간 객체가 조건에 부합하면 교차 영역 제거 단계를 수행한다. 공간 객체 관리자는 공간 객체의 식별자가 이미 전송한 객체의 식별자인지를 검사하고, 참이면 다음 공간 객체를 로드한다. 이미 전송한 객체가 아니면 공간 객체를 결과 테이블에 저장하고, 그 식별자를 임시 영역에 저장한다. 위의 과정을 더 이상

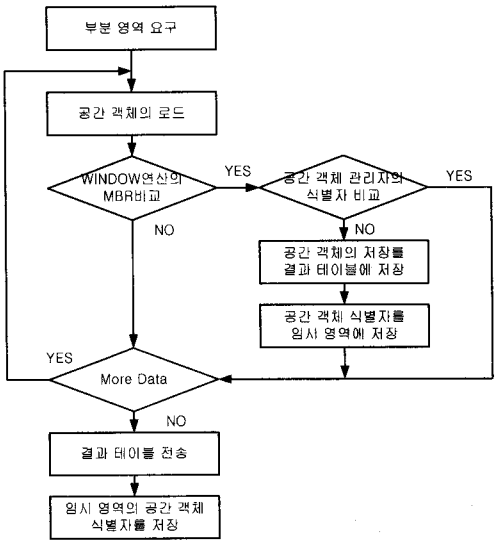


그림 8. 창 질의의 수행 과정

로드 할 공간 객체가 없을 때까지 반복한다. 레이어 내의 모든 데이터를 검사하면, 질의 결과 테이블을 클라이언트로 전송한다. 클라이언트로 전송한 후, 임시 영역에 저장되어 있는 공간 객체 식별자를 공간 객체 관리자를 통해 저장한다.

3.3.3 공간 객체 관리자의 식별자 관리

가. 객체 식별자 관리를 위한 자료 구조

공간 객체 관리자는 객체 식별자의 관리를 위해 물리적 객체 식별자의 특징을 활용하여 레이어 영역, 각 레이어 영역을 구성하는 페이지 영역, 각 페이지 영역을 위한 슬롯 비트맵을 관리한다. 이들의 자료 구조는 다음 그림 9과 같다.

동일한 페이지 내의 공간 객체는 동일한 레이어 내에 존재하게 된다. 따라서, 레이어 영역은 레이어를 구성하는 페이지 수만큼 페이지 영역 리스트를 가진다. 레이어 영역을 위한 할당은 레이어에 대한 질의 요구가 있을 때 초기에 이루어지며, 레이어 영역을 할당할 때 레이어를 구성하는 페이지 수 만큼 페이지 영역 리스트를 할당한다. 페이지 ID를 통한 페이지 영역으로의 접근은 H(PageID) = PageID % MaxPage의 해쉬 함수를 통해 이루어지며 충돌에 의한 버킷(bucket) 할당은 개방 주소(Open Addressing) 기법에 의해 해결한다. 공간 객체의 저장은 한 번에 많은 양이 일괄적으로 이루어지므로 레코드 저장 시 할당되는 페이지들은 연속적인 형태로 할당되어 페이지 ID는 순차적인 값을 갖게 된다. 따라서, 개방 주소 기

법에 의한 충돌 해결 방법으로 해결이 가능하다.

```
struct LayerRegion {
        int             LayerID;        // 레이어 ID
        int             NumPage;        // 페이지 수
        PageRegion      PageList[NumPage];
                                        // 페이지 영역 리스트
        LayerRegion* spNext;            // 다음 레이어 영역
};

struct PageRegion {
        int             PageID;         // 페이지 ID
        int             MaxSlot;        // 페이지 내 슬롯 수
        int             SlotBitMap[MaxSlot/32 + 1];
                                        // 슬롯 비트맵
};
```

그림 9. 객체 식별자 관리를 위한 자료 구조

페이지 영역은 페이지 내의 레코드 수보다 크거나 같은 최대 슬롯 ID 만큼의 슬롯 비트맵 영역을 할당한다. 단일 객체 식별자를 한 비트로 관리할 수 있어 저장 공간의 낭비를 방지하며, 일반 메모리 연산보다 빠른 비트 연산을 사용하여 처리 효율을 향상시킨다. 공간 객체 관리자는 비트 연산을 사용하기 위해 정수형 데이터 타입을 사용해 비트맵 영역을 할당한다. 각 페이지 영역은 n개의 슬롯을 위한 비트 영역으로 n/sizeof(int) + 1개의 정수형 데이터 타입의 영역을 할당한다. 한 페이지를 위한 비트맵 영역은 최초 해당 페이지에 접근이 이루어질 때 할당하며, 1개의 정수형 데이터 타입은 32비트 컴퓨터의 경우 32개의 슬롯 번호를 관리한다. 하나의 공간 객체가 전송되면 식별자의 슬롯 번호에 해당하는 슬롯 비트를 셋팅한다. 객체 식별자의 슬롯 번호가 해당 페이지의 MaxSlot보다 크면 페이지 내에 새로운 객체가 저장된 것이므로 페이지 정보를 새로 읽어 슬롯 비트맵을 확장한다.

나. 공간 객체의 전송 여부 판별

교차 영역 제거 단계에서 공간 객체 관리자는 전송할 공간 객체의 식별자를 이미 전송한 객체 식별자와 비교하여 전송 여부를 판단하게 된다. 전송 여부의 판단 과정은 시스템의 전체적인 성능에 영향을 미치므로 빠른 처리가 가능해야 한다. 이미 전송된 지의 여부를 판단하는 과정은 다음 〔알고리즘 1〕과 같다.

```
〔알고리즘 1〕 전송 여부의 판단
입력 : 레이어 ID와 공간 객체의 식별자
출력 : TRUE or FALSE
IsLoadedSOID( LayerID, SpatialOID )
{
// 레이어 영역 검색
// 처음으로 전송되는 레이어일 경우
if  not Loaded LayerID
return FALSE;
// 페이지 영역 검색
Get PageID from SpatialOID;
// 페이지 영역이 셋팅되어 있지 않으면
if not exist PageRegion for PageID {
    return FALSE;
}
else { // 할당되어 있으면
// 슬롯 비트맵 검색
    Get Slot ID from Spatial OID;
// 새로 저장된 객체이면
    if Slot ID 〉 MaxSlot
    return FALSE;
// 슬롯 비트맵이 셋팅되어 있으면
if  Slot bit for Slot ID is set
return TRUE;
// 슬롯 비트맵이 셋팅되어 있으면
    else
    return FALSE;
    }  // end of if
return FALSE;
}
```

전송 여부의 판별 과정은 레이어 검사, 페이지 검사, 슬롯 검사의 순으로 진행된다. 먼저 레이어 영역의 검색을 통해 해당 레이어가 없을 경우는 클라이언트에서 처음으로 접근하는 경우이므로 이후의 검사 대상이 되는 모든 공간 객체는 전송되지 않았음을 알 수 있다. 레이어 영역이 존재하면 페이지 ID에 해당하는 페이지 영역을 검색하여 페이지 영역이 존재하지 않으면 전송되지 않았음을 알 수 있다. 페이지 ID로부터 해당 페이지로의 접근은 해쉬 함수를 통해 이루어진다. 페이지 영역이 존재하면 슬롯 영역의 해당 슬롯 비트를 검사하여 셋팅되어 있으면 전송된 객체이며 셋팅되어 있지 않으면 전송되지 않은 객체임을 알 수 있다. 해당 슬롯 비트는 Slot ID/32의 몫에 해당하는 비트맵에 나머지에 해당하는 위치의 비트가 된다.

다음 그림 10은 페이지 ID p, 슬롯 ID s를 가지는 객체 식별자의 전송 여부를 판단하는 과정이다. 먼저 레이어 검사 단계에서 얻은 현재 레이어의 위치로부터 p % NumPage에 해당하는 페이지 영역이 p를 위한 영역인지를 검사하는 페이지 검사를 수행하며, 슬롯 비트 검사 단계에서 s/32위치의 비트맵에서 s % 32의 위치가 1로 셋팅되어 있는 지를 검사한다. 그림

10에서와 같이 슬롯 비트가 셋팅되어 있으면 이 객체는 이미 클라이언트에 전송된 객체이다.

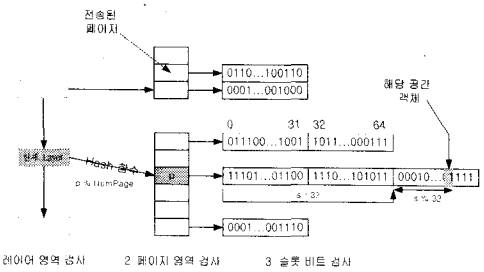


그림 10. 공간 객체 식별자의 전송 여부 판단

다. 전송된 객체의 식별자 저장

창 질의를 처리하고 난 후, 전송한 공간 객체들의 식별자를 식별자 디렉토리 내에 셋팅해야 한다. 객체의 식별자를 셋팅하기 위한 알고리즘은 〔알고리즘 2〕와 같다.

```
〔알고리즘 2〕 공간 객체 식별자의 저장
입력 : 레이어 ID와 공간 객체 식별자
출력 : 없음
AttachSpatialOID( LayerID, SpatialOID )
{
// 레이어 영역 셋팅
// 처음 전송하는 레이어일 경우
if  not Loaded LayerID {
Add Layer Region;
Get the Number of Pages for LayerID;
Add Page Region list;
}
Get PageID from SpatialOID;
// 페이지 영역이 셋팅되어 있지 않으면
if not set PageRegion for PageID
{
// 객체 추가로 페이지 수가 증가하였으면
if  no Space for setting PageRegion {
        Add Page Region;
    }
    Set Page Region;
    Get Max Slot Number for PageID;
    Add Slot BitMap;
}
// 슬롯 비트맵 셋팅
Get Slot ID from Spatial OID;
// 페이지 내에 새로운 객체가 추가되었으면
if  MaxSlot < SlotID  {
Get Max Slot Number for PageID;
Append Slot BitMap;
}
Set Slot bit for Slot ID;
}
```

객체 식별자의 저장 과정은 레이어 영역 할당, 페이지 영역 할당, 슬롯 비트 셋팅의 순으로 진행된다. 전송한 레이어를 위한 영역이 없으면 레이어의 정보를 읽어 레이어 영역과 함께 페이지 수 만큼 페이지 영역을 할당한다. 이 과정은 최초의 객체 식별자를 셋팅할 때만 이루어진다. 객체 식별자의 페이지 ID에 해당하는 페이지 영역을 조사하여 셋팅되어 있지 않으면 페이지 정보를 읽어 최대 슬롯 ID만큼의 슬롯 비트맵 영역을 할당한다. 레이어 영역 및 페이지 영역을 모두 할당하면 객체 식별자의 슬롯 ID에 해당하는 슬롯 비트맵내의 비트를 셋팅한다.

4. 성능 평가

본 장에서는 창 질의에 대해 본 논문에서 제안한 공간 객체 관리자가 교차 영역의 중복 데이터를 제거하고 전송하는 경우와 서버에서 교차 영역을 제거하지 않고 전송하는 경우를 비교 평가한다.

4.1 평가 환경

제안된 공간 객체 관리자는 클라이언트-서버 환경을 위한 공간 데이터베이스 시스템 GEOMania Millennium server를 위해 구현되었으며, 성능 평가를 위해 GEOMania Millennium server 서버와 GEOMania Millennium server 클라이언트를 기반으로 수행한다.

4.1.1. 시스템 환경

실험 평가를 위한 GEOMania Millennium server의 서버 시스템 환경은 표 1과 같으며, GEO/Millennium 클라이언트 시스템 환경은 표 2와 같다. 서버 프로세스들의 동시성 제어 및 버퍼 관리 등을 위한 공유 메모리 영역을 2M로 하였으며, 테이블을 구성하는 페이지의 크기를 16K로 하여 버퍼에서 한번의 I/O를 통해 읽을 수 있는 크기와 동일하게 하였다.

표 1. 서버 시스템 환경

Computer	SUN Ultra-10	DB Disk Size	2G
CPU	UltraSPARC-II i 333MHz	Shared Memory Size	2M
Main Memory Size	256 M	Buffer Size	1M
OS	Solaris 2.6	Page Size	16K
Programming Language	C++	Pages per Extent	16

표 2. 클라이언트 시스템 환경

IBM PC	OS	Windows98
Pentium 333	Disk Size	8G
128M	Programming Language	C++

4.1.2 예제 데이터베이스

실험 평가를 위해 실질적인 공간 데이터베이스인 서울시 데이터베이스의 빌딩 레이어를 사용한다. 예제 데이터베이스의 물리적인 명세는 표 3이다. 빌딩 레이어의 공간 객체는 지역적 분포가 불규칙하며 평균 4-6개의 좌표로 구성된 POLYGON으로 구성되며 평균 140bytes 정도의 크기를 갖는다.

표 3. 서울시의 빌딩 레이어

필드 수	8	테이블 크기	약 6.5M
공간 데이터 타입	POLYGON	평균 레코드 크기	140 bytes
레코드 수	45188개	페이지 수	396 pages

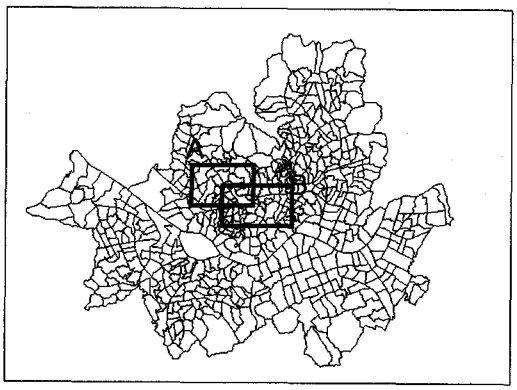


그림 11. 실험 평가 방법

4.2 실험 평가

4.2.1 평가 방법

실험 평가에 사용하는 창 질의의 예는 그림 11과 같다. 우선 클라이언트에서 A 영역의 빌딩 레이어를 로드한 후에, 화면 이동을 통해 B영역을 로드한다. 즉, 새로이 로드하는 B영역 중에서 기존에 로드된 영역 A와의 교차 정도를 변화시켜 이에 따른 성능 변화를 살펴 본다.

A 영역의 전송 후, 화면 이동에 의해 수행되는 B영역의 요구에 해당하는 질의는 다음과 같다.

질의 : B 영역의 요구

```
SELECT *
FROM BUILDING
WHERE WINDOW( X1B, Y1B, X2B, Y2B );
```

4.2.2 평가 결과

제안한 공간 객체 관리자가 교차 영역의 데이터를 제거하고 전송하는 경우와, 서버에서 중복을 제거하지 않고 전송하여 클라이언트에서 교차영역을 제거하는 경우를 비교 평가한다. 성능 비교를 위해 A영역을 전송한 후에 B 창 질의를 처리했을 때 선택된 객체의 수, 서버에서의 처리 시간, 전송 시간, 최종 응답 시간 등을 비교한다.

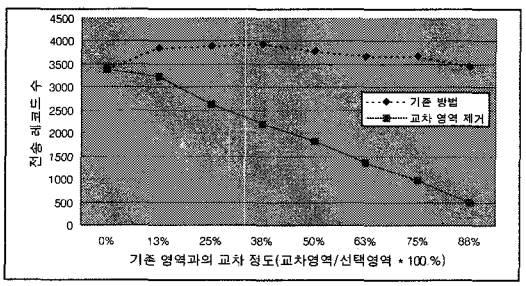


그림 12. 전송된 레코드 수의 비교

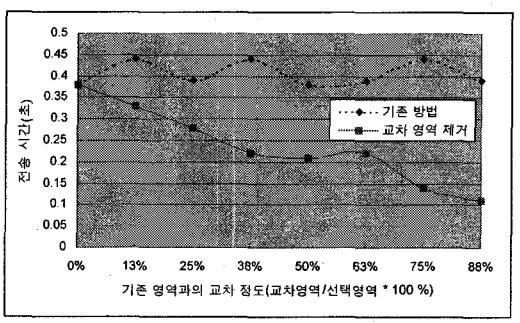


그림 13. 전송 시간의 비교

그림 12와 그림 13은 교차 영역의 데이터를 제거한 경우와 제거하지 않은 경우의 생성된 결과 테이블의 레코드 수와 이에 따르는 전송 속도를 나타낸다. 교차 영역을 제거한 방법이 제거하지 않은 경우보다 교차 영역이 클수록 전송하는 레코드 수가 현격히 감소하며, 이에 따르는 전송 시간도 줄어듦을 볼 수 있다.

그림 14와 그림 15은 교차 영역을 제거한 경우와 제거하지 않은 경우의 서버에서의 처리 시간과 최종 응답 시간의 비교를 보이고 있다.

그림 14에서 교차 영역을 제거하지 않은 경우가 중복을 제거한 경우보다 서버 처리 비용은 적지만, 창 질의에서 중복된 영역이 커질수록 제안한 기법의 서버

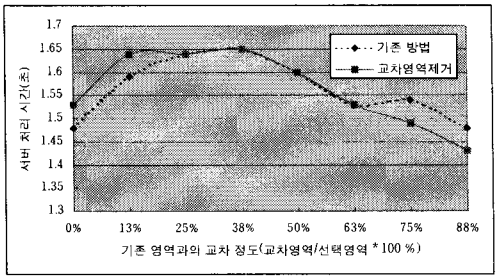


그림 14. 서버 처리 시간의 비교

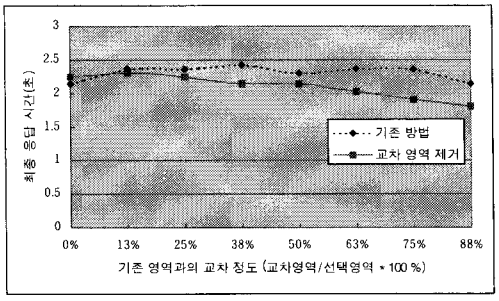


그림 15. 최종 응답 시간의 비교

처리 비용이 오히려 줄어듦을 볼 수 있다. 이는 결과 레코드 수가 감소함으로서 서버에서 관리해야 할 결과 테이블의 관리 부하가 줄어들기 때문이다. 그리고, 최종 응답 시간의 경우 약 7% 이상의 교차 영역이 존재하는 경우 제안한 기법이 보다 나은 성능을 보임을 알 수 있다.

5. 결 론

클라이언트-서버 환경의 공간 데이터베이스 시스템은 관리하는 공간 데이터의 양이 방대하여 데이터 전송에 따르는 네트워크 부하가 문제가 된다. 따라서, 공간 데이터를 전송함에 있어 전체 레이어를 전송하기보다는 사용자가 필요로 하는 일부 영역의 데이터만 전송하는 방법이 필요하다. 그러나, 화면 이동, 축소 및 확대 등의 브라우징 기능에 의한 창 질의는 이미 전송된 영역과 유사한 영역의 데이터를 요구하게 되어 클라이언트에서의 중복 데이터의 제거 비용뿐 아니라, 중복 전송에 의한 네트워크 부하를 가중시키게 된다.

본 논문에서는 일부 영역의 데이터를 전송하는 창 질의의 처리 과정에서 공간 객체의 식별자의 관리를 통해 교차 영역을 제거하는 공간 객체 관리자를 설계하고 구현하였다. 제안한 공간 객체 관리자는 클라이언트에 전송된 공간 객체의 식별자를 비트맵 디렉토리 구조로 관리하여 창 질의의 처리 과정에서 교차 영역을 제거하여 네트워크 부하를 줄이도록 하였다.

본 논문에서 창 질의는 영역 연산 수행 단계와 교차 영역 제거 단계를 거쳐 처리한다. 교차 영역 제거 단계를 수행하는 공간 객체 관리자는 물리적 객체 식별자의 특징을 이용하여 각 식별자를 하나의 비트로서 관리함으로써 저장 공간의 낭비를 최소화하였으며, 레이어 검사, 페이지 검사, 슬롯 검사의 단계를 거쳐 공간 객체의 전송 여부를 빠르게 비교할 수 있도록 하였다. 제안한 공간 객체 관리자는 개방형 클라이언트-서버 공간 데이터베이스 시스템인 GEOMania Millennium server에 구현하였고, 성능 평가를 통해 교차된 영역이 존재하는 창 질의에 대해 질의 수행의 성능이 향상됨을 보였다.

향후 연구 과제로는 클라이언트에 전송된 데이터와 서버내의 데이터와의 일관성 유지를 위한 연구가 필요하며, 창 질의에서 선택된 영역이 기존 영역과 중복된 영역이 없는 경우의 효율적 처리에 대한 연구가 필요하다.

참 고 문 헌

[1] Abiteboul, S. and Kanellakis, P., "Object Identity as a Query Language Primitive," Proceedings of the ACM SIGMOD International Conference on the Management of Data, pp.159-173, 1989.

[2] Brinkhoff T., Horn H., Kriegel H.-P.,

Schneider R. A., Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems, LNCS 692, Springer-Verlag, pp.357-376, 1993.

[3] Cho, Y.-S., Kim, H.-Y., Kim, J.-H., Bae, H.-Y., The Design and Implementation of Componentized Web-enabled GIS, ICIMU, 1998.

[4] Egenhofer, M.J. and Frank, A.U., Towards a Spatial Query Language: User Interface Considerations, Proc. 14th International Conference on VLDB, pp.124-133, 1988.

[5] Franklin, M. J., Caching and Memory Management in Client-Server Query Processing, Proc. of ACM SIGMOD, Vol. 25, pp.149-160, 1996.

[6] Franklin, M. et al., Global Memory Management in Client-Server DBMS Architecture, Proc. of VLDB, pp.596-609, 1992.

[7] Gting, R., "An Introduction to Spatial Database Systems," VLDB Journal, Vol.3, pp.357-399, 1994.

[8] Kim, P.C., Choi, H.I., Lee, Y.J., Lee S.H., and Kim, M.J., "MIDAS: Design philosophy and Internals," IPCCC, pp.132-139, 1992.

[9] Khoshafian, S. and Copeland, G.P., Object Identity, Readings in Object-Oriented Database Systems, Morgan Kaufmann Publishers, pp.37-46, 1990.

[10] Lu, H. and Ooi, B.-C., Spatial Indexing: Past and Future, IEEE Data Engineering Bulletin, Vol.16, No.3, pp.16-21, Sept. 1993.

[11] Manfred von Seggern, The Enterprise GIS: A Client/Server Approach Using Distributed Relational Databases To Create A Multi Location/Multi Application GIS, EGIS, Vol. 1, pp.652-659, 1994.

[12] Papadias, D. Theodoridis, Y., Sellis, T., Egenhofer, M. J., Topological Relations in the World of Minimum Bounding Rectangles : A Study with R-trees, Proc. of ACM SIGMOD, Vol. 24, pp. 92-103, 1995.

[13] Samet, H. and Aref, W. G., Spatial Data Models and Query Processing, Modern Database Systems, ACM Press, pp. 338-360, 1995.

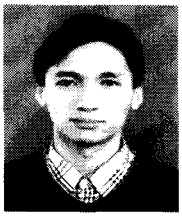
[14] 강규원, 김홍연, 김종훈, 배해영, "클라이언트-서버 GIS에서의 트랜잭션 처리 시간을 최소화하는 역할 분담 스케줄러," 한국정보과학회 봄 학술발표논문집, Vol. 24, No. 1, pp.175-178, 1997.

[15] 강현철, "클라이언트-서버 DBMS 연구동향," 한국정보과학회지, Vol. 12, No. 3, pp.42-52, 1994.

[16] 박상일, 김종훈, 배해영, "공간 데이터베이스 시스템에서의 계층적인 데이터 표현," 한국정보처리학회 봄 학술발표논문집, Vol. 3, No. 1, pp.644-649, 1996.

[17] 이경모, 이충호, 김성희, 배해영, "효율적 공간 검색을 위한 새로운 색인 키 중복 기법," 한국정보과학회 가을 학술발표논문집, Vol. 26, No. 2, pp.261-263, 1999.

[18] 최현호, 조영섭, 김홍연, 배해영, "Web기반 GIS에서의 전송속도 개선을 위한 레이어 테이블 수직 분할 기법," 한국정보과학회 가을 학술발표논문집, Vol. 25, No. 1, pp.71-73, 1998.



이경모

1998년 인하대학교 전자계산공학과 졸업(공학사).

2000년 인하대학교 대학원 전자계산공학과 졸업(공학석사).

2000년~현재 인하대학교 대학원 전자계산공학과 박사과정.

관심분야: 데이터베이스, 공간 데이터베이스, 클라이언트-서버 데이터베이스, 주기억 장치 상주 데이터베이스

박동선

1992년 인하대학교 전자계산공학과 졸업(공학사).

1994년 인하대학교 대학원 전자계산공학과 졸업(공학석사).

1996년~현재 인하대학교 대학원 전자계산공학과 박사과정.

관심분야: 데이터베이스, 지리정보시스템, 시간지원 지리정보시스템

김재홍

1988년 인하대학교 전자계산학과 졸업(이학사).

1990년 인하대학교 대학원 전자계산학과 졸업(이학석사).

1994년 인하대학교 대학원 수학과(전산전공, 이학박사).

1995~현재 영동대학교 컴퓨터공학과 조교수.

관심분야: 데이터베이스, 지리정보시스템, 실시간 데이터베이스.

배해영

1974년 인하대학교 응용물리학과 졸업(공학사).

1978년 연세대학교 대학원 전자계산학과 졸업(공학석사).

1989년 숭실대학교 대학원 전자계산학과 졸업 (공학박사).

1995년 Univ. of Houston 객원교수.

1992년~1994년 인하대학교 전자계산소 소장.

1982년~현재 인하대학교 전자계산공학과 교수.

관심분야: 데이터베이스, 멀티미디어 데이터베이스, 클라이언트-서버 데이터베이스, 지리정보시스템, 실시간 데이터베이스