R-tree 재구성 방법을 이용한 공간 뷰 실체화 기법

정 보 흥† · 배 해 영††

요 약

공간데이터베이스 시스템에서는 사용자에게 효과적인 공간데이터베이스 접근방법을 제공하기 위하여 공간 뷰를 지원하며 비 실체화 방법과 실체화 방법으로 관리한다. 비 실체화 방법은 동일질의에 대한 반복적인 수행으로 서버 병목 현상과 네트워크 부하가 발생하는 문제점이 있고, 실체화 방법은 공간 기본 테이블 변경에 대한 실체화 뷰 관리 방법이 어렵고, 비용이 많이 든다는 문제점이 있다. 본 논문에서는 R-tree 재구성 방법을 이용한 공간 뷰 실체화 관리 기법(SVMT : Spatial View Materialization Technique)을 제안한다. 제안한 SVMT는 공간 뷰 객체 분포율 오차를 이용하여 공간 뷰를 실체화하는 기법으로, 공간 뷰 객체 분포율 오차가 오차 한계 범위내에 존재하면 공간 뷰를 실체화하고 공간 뷰 높이에 해당하는 노드를 공간 뷰에 대한 R-Tree의 루트로 사용하고, 오차 한계 범위를 벗어나면 공간 뷰를 실체화함과 동시에 R-Tree를 재구성하는 방법이다. 이 기법에서 공간 뷰에 대한 정보는 공간 뷰 정보 테이블(SVIT : Spatial View Information Table)를 통하여 관리되며, 이 테이블의 레코드는 공간 뷰에 대한 정보를 저장하고 있다. SVMT는 실체화된 공간 뷰에 대한 질의수행을 통해 공간 질의 처리 수행 속도를 빠르게하며, 이를 통하여 반복적인 질의 변환을 통해 발생하는 부가적인 질의 수행 비용을 제거한다. 따라서, 제안하는 기법은 다중 사용자, 동시 작업 환경에서 공간 뷰에 대한 빠른 접근 속도와 빠른 질의 응답을 제공하여 서버의 병목현상과 네트워크 부하를 최소화한다는 장점을 가진다.

Spatial View Materialization Technique by using R-Tree Reconstruction

Bo-Heung Chung† · Hae-Young Bae††

ABSTRACT

In spatial database system, spatial view is supported for efficient access method to spatial database and is managed by materialization and non-materialization technique. In non-materialization technique, repeated execution on the same query makes problems such as the bottle-neck effect of server-side and overloads on a network. In materialization technique, view maintenance technique is very difficult and maintenance cost is too high when the base table has been changed. In this paper, the SVMT (Spatial View Materialization Technique) is proposed by using R-tree re-construction. The SVMT is a technique which constructs a spatial index according to the distribution ratio of objects in a spatial view. This ratio is computed by using a SVHR (Spatial View Height in R-tree) and SVOC (Spatial View Object Count). If the ratio is higher than the average, a spatial view is materialized and the R-Tree index is re-used. In this case, the root node of this index is exchanged a node which has a MBR (Minimum Boundary Rectangle) value that can contains the whole region of spatial view at a minimum size. Otherwise, a spatial view is materialized and the R-tree is re-constructed. In this technique, the information of spatial view is managed by using a SVIT (Spatial View Information Table) and is stored on the record of this table. The proposed technique increases the speed of response time through fast query processing on a materialized view and eliminates additional costs occurred from repeatable query modification on the same query. With these advantages, it can greatly minimize the network overloads and the bottle-neck effect on the server.

키워드 : 공간뷰(Spatial View), 데이터베이스(Database), 지리정보시스템(GIS), 공간데이터베이스 시스템(Spatial Database System)

1. 서 론

공간데이터베이스 시스템은 방대한 지도데이터를 공간데이터와 비공간 데이터로 구성된 공간 객체로 관리하며, 국토/환경/도시/지하시설물 관리 등과 같은 다양한 응용 분야에 사용되는 시스템이다[1, 6, 9, 14]. 최근의 네트워크 환경의 발달과 인터넷의 급속한 보급으로 인하여 공간 데이터베이스 시스템은 점차 클라이언트/서버 환경에서 대형화 다중 서버/클라이언트 환경 및 분산환경으로 발전하고 있다. 이러한 환경에서는 다중 사용자 동시 접근/복잡한 질의 처리/빠른 응답시간, 공유 데이터의 효율적인 관리가 필수적으로 필요하며, 사용자의 질의에 의하여 기본 테이블로부터 생성되는 동적 데이터에 대한 처리와 다수의 테이블에 존재하는 데이터에 대한 복잡한 질의를 효과적으로 수행할 수 있는 기능에 대한 지원이 필요하다[2, 3]. 또한, 시스템에서 다수의 사용자에 대한 접근을 서버의 병목현상과 네트워크 부하를 최소화하여 효율적으로 지원 할 수 있는 방법이 필

※ 본 논문은 정보통신부의 대학소프트웨어 연구센터 지원사업의 연구 결과임.
† 준 회 원 : 인하대학교 대학원 전자계산공학과
†† 종신회원 : 인하대학교 전자계산공학과 교수
논문접수 : 2000년 6월 1일, 심사완료 : 2001년 7월 18일

요하다. 이를 위하여 공간 데이터베이스 시스템은 사용자에게 효과적인 공간데이터베이스 접근방법을 제공하기 위하여 공간 인덱스와 공간 뷰[1, 9, 13]를 제공한다.

공간 데이터베이스 시스템에서 사용하는 공간 인덱싱 방법은 R-tree, Quad-Tree, Grid-Tree등의 다양한 SAM(Spatial Access Method)이 있으며, 이 중에서 다양한 공간 객체에 대하여 적은 비용으로 효율적인 질의처리를 수행할 수 있는 R-Tree를 SAM으로 사용한다[4-8]. 공간 뷰는 관계형, 객체지향형 데이터베이스 시스템의 뷰를 확장하여 형태로 구현된다. 뷰란 데이터베이스의 기본 테이블에서 유도된 가상의 테이블이며, 물리적 데이터베이스에 대한 논리적인 독립성을 제공하는 가장 효과적인 수단이다[1, 9, 13]. 뷰에 대한 관리방법은 뷰를 실체화하지 않고 질의 수행시 질의 수정(query modification/ substitution)을 통하여 처리하는 비 실체화 방법과 뷰를 실행하여 결과 객체들을 데이터베이스에 저장하여 관리하는 뷰 실체화(View Materialization) 방법이 있다[1, 9, 13]. 질의 수정 방법은 뷰에 대한 질의를 기본 테이블에 대한 질의로 변환하여 수행하는 방법이다. 반면에 뷰 실체화 방법은 먼저 뷰를 수행하여 그 결과를 물리적 데이터베이스에 저장하고, 뷰에 대한 질의를 이미 저장된 결과에 대하여 직접 수행하는 방법이다. 질의 수정 방법은 반복되는 공간 질의 수행으로 서버 병목 현상과 네트워크 부하가 발생한다는 문제점이 있고, 실체화 방법은 공간 기본 테이블 변경시 실체화 뷰에 대한 갱신이 매우 어렵다는 문제점이 있다. 이러한 뷰 관리 문제점을 해결하기 위하여 뷰 대한 연구가 최근까지 활발히 진행되고 있다[11, 12]. 이러한 연구들은 베이스 테이블 데이터 변경에 대한 주기적 또는 점진적 뷰 관리 방법과 어떤 테이블 또는 뷰를 실체화하여 관리할 것인지에 대한 뷰 셀렉션 방법에 대한 것들이다. 그러나, 이러한 연구들은 관계형, 객체지향형 데이터베이스 시스템의 뷰 관리 방법에 대한 것이어서 공간 데이터베이스 특성을 고려한 공간 뷰에 대한 연구가 미비한 실정이다.

본 논문에서는 R-tree 재구성 방법을 이용한 공간 뷰 실체화 관리 기법(SVMT : Spatial View Materialization Technique)을 제안한다. 제안한 SVMT는 공간 뷰 객체 분포율 오차를 이용하여 공간 뷰를 실체화하는 기법으로, 공간 뷰 객체 분포율 오차가 오차 한계 범위를 이용하여 실체화하고 실체화된 공간 뷰에 대한 관리를 수행한다. 이 기법에서 공간 뷰에 대한 정보는 공간 뷰 정보 테이블(SVIT : Spatial View Information Table)를 통하여 관리되며, 이 테이블의 레코드는 공간 뷰에 대한 정보를 저장하고 있다. 제안한 기법은 공간 뷰 실체화 과정과 공간 뷰 관리 과정으로 구분된다. 공간 뷰에 대한 실체화 과정은 새로 정의된 공간 뷰에 대한 정보가 SVIT의 레코드로 추가되며, 공간 뷰 정의 영역(SVDA : Spatial View Definition Area)의 공간 뷰 객체 비율(SVOR : Spatial View Object Ratio)의 확률적인 값과 실제 값을 계산하여 공간 뷰 객체 분포율 오차가 오차 한계 범위내에 존재하면 공간 뷰를 실체화하고 공간 뷰 높이에 해당하는 노드를 공간 뷰에 대한 R-Tree의 루트로 사용하고, 오차 한계 범위를 벗어나면 공간 뷰를 실체화함과 동시에 R-Tree를 재구성한다. 공간 뷰 관리 단계는 공간 뷰에 대한 공간 질의를 처리하는 과정이며, 공간 뷰에 대한 검색, 추가/삭제등의 변경 연산후 SVOR값을 계산하여 이 값이 오차 범위와 관계를 조사한다. 즉, 이전의 SVIT가 가진 SVOR값과 연산후 재 계산된 SVOR값에 변동이 발생하였고, 이 값의 오차가 SVODR값의 오차 범위를 벗어난다면, 실체화된 공간 뷰에 대하여 R-Tree를 재구성한다. 반대의 경우에는 공간 뷰 높이에 해당하는 R-Tree노드를 공간 뷰에 대한 R-Tree의 루트 노드로 세팅한다.

SVMT는 공간 뷰를 실체화하여 동일 공간 뷰 접근을 통해 발생하는 반복적인 질의 변환 수행시간을 제거하고 공간 뷰 실체화 시에는 SVODR의 오차 한계를 이용하여 R-tree 관리비용을 최소화한다. 이 과정을 통하여 공간 뷰에 대한 질의 처리 속도를 향상시키며 실체화된 공간 뷰 관리 비용을 최소화한다. 제안하는 기법은 실체화된 공간 뷰를 통한 빠른 질의처리를 통해 사용자에게 빠른 질의 응답을 제공하여 클라이언트/서버 환경과 같은 다중 사용자, 동시 작업 환경에서 서버의 병목현상과 네트워크 부하를 최소화할 수 있다는 장점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 시스템에서 공간 뷰를 제공하는 방법과 실체화된 공간 뷰에 대한 효율적인 뷰 관리작업을 수행하는 기존의 방법에 대하여 살펴본다. 3장에서는 본 논문에서 제안하는 SVMT에서 사용되는 모든 용어에 대한 정의와 공간 뷰 관리를 위한 자료구조에 대하여 살펴보고, 4장에서는 SVMT를 구현하기 알고리즘에 대해서 살펴보고, 5장에서는 본 논문에서 제안하는 기법에 대하여 성능평가 결과에 대하여 설명하고, 마지막으로 결론을 제시한다.

2. 관련 연구

본 절에서는 공간 데이터베이스 시스템에서 제공하는 공간 뷰 관리 기법과 공간 인덱스를 이용하는 질의처리 방법에 대하여 알아본다. 먼저 R-tree, Grid tree, Quad tree등의 다양한 공간 인덱스와 이를 이용하여 공간 질의를 처리하는 방법에 대하여 설명하고, 다음으로는 공간 뷰에 대한 개념, 질의 변환과 뷰 실체화를 통한 공간 뷰 관리 기법에 대하여 설명하고, 마지막으로 실체화된 공간 뷰와 R-tree 인덱스와의 관련성에 대하여 설명한다.

2.1 공간 인덱스를 이용한 공간 질의 처리 방법

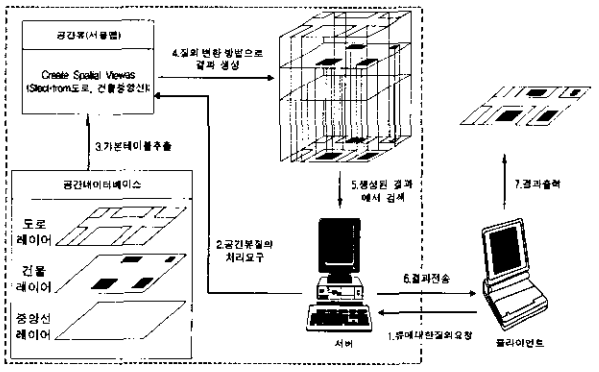
공간 데이터베이스 시스템은 공간 데이터의 효율적인 관

리와 빠른 검색질의처리를 위하여 공간 인덱스를 사용한다[1, 10]. 공간 인덱스는 R-Tree, Grid tree, Quad tree, TV-tree, X-tree등과 같은 다양한 인덱싱 방법이 존재하며, 복잡하며 다양한 공간 객체를 경계사각형 또는 격자 형태의 단순한 객체로 근사하여 빠른 검색과 효율적인 저장관리를 수행하기 위하여 사용된다[5-8]. 일반적으로 공간 데이터베이스 시스템은 다른 인덱싱 방법에 비해 객체 근사의 간편성과 빠른 검색 효율을 제공하는 R-tree를 공간 인덱스로 사용한다.

공간 인덱스를 이용한 공간 질의 처리 과정은 여과단계와 정제단계로 구분되며, 여과 단계는 사용자가 요구한 공간 질의결과를 구하기 위한 후보객체를 선택하는 과정이며, 정제단계는 이 후보객체에서 실제로 조건을 만족하는 공간 객체를 추출하는 과정이다. 따라서, 효율적인 질의처리를 위해서는 위의 실제 공간 객체를 추출하기 위해 후보객체에 대한 정제 과정을 수행하는 비용을 최소화하여야 하며, 가능한 한 후보객체 수를 최소화해야 한다.

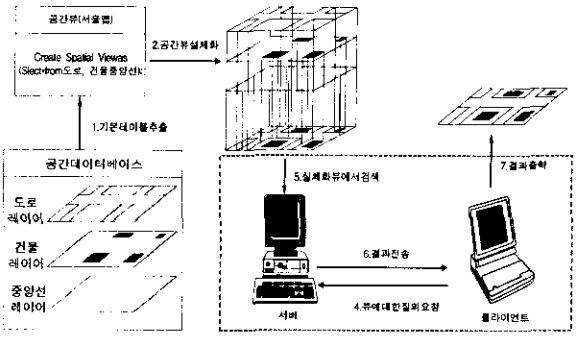
2.2 공간 뷰 관리 방법

공간 뷰는 관계형, 객체지향형 데이터베이스 시스템의 뷰를 확장하는 방식으로 구현된다[13]. 뷰란 데이터베이스의 기본 테이블에서 유도된 가상의 테이블이며, 물리적 데이터베이스에 대한 논리적인 독립성을 제공하는 가장 효과적인 수단이다[9]. 기존의 공간 뷰 관리 방법은 질의 변환 방법과 뷰 실체화 방법이 있다.



(그림 2-1) 질의 변환 방법

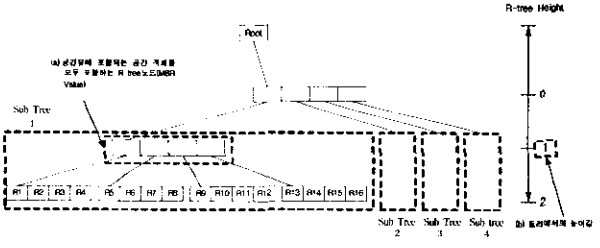
질의 변환 방법은 (그림 2-1)과 같이 공간 뷰를 가상의 테이블로 관리하며, 공간 뷰에 대한 질의를 실제 기본 공간 테이블에 대한 질의로 변환하여 수행하는 방법이다. 이 방법에서 사용자가 정의한 공간 뷰는 공간 SQL의 형태로 서버에 저장 관리되며, 공간 뷰에 대한 질의 수행은 (그림 2-1)의 붉은 점선부분과 같은 기본 테이블에 대한 질의 변환 과정을 거쳐 수행된다. 이 방법은 동일한 공간 뷰에 대한 질의처리를 위해 반복적인 질의 변환과정을 수행해야 하며, 다중사용자, 동시작업 환경에서는 이 과정으로 인하여 서버 부하가 발생한다.



(그림 2-2) 뷰 실체화 방법

뷰 실체화 방법은 (그림 2-2)와 같이 사용자의 공간 정의 질의를 수행하여, 그 결과를 데이터베이스의 실제 테이블로 저장한다. 공간 뷰에 대한 질의 처리는 (그림 2-2)의 붉은 점선 영역에서 보인 것처럼, 공간 뷰에 대한 질의 처리시 실체화된 테이블에 대하여 질의를 수행하는 방법이다. 질의 변환 방법에 비하여 질의 처리 수행 비용은 감소하지만, 실체화된 뷰와 공간 기본 테이블사이의 일관성을 유지하기 위한 비용이 추가적으로 발생한다. 실체화된 뷰의 일관성 유지를 위한 방법으로는 관계형 시스템의 실체화 뷰 관리기법에서는 재 계산 방법과 점진적인 방법을 사용하며, 점진적인 방법은 counting과 DRed라는 알고리즘을 이용하여 관리한다[11]. 공간 데이터베이스 시스템에서의 실체화 뷰 관리 기법은 객체지향형 개념을 이용하여 공간 뷰에 대한 점진적인 변경을 수행하는 방법이 있다[14]. 그러나, 두 가지 분야 모두 공간 데이터베이스에서 사용하는 R-tree를 이용하는 경우에는 적용할 수 없으며, 이에 대한 연구가 미비한 실정이다.

2.3 공간 뷰 실체화와 R-tree 공간 인덱스



(그림 2-3) R-Tree의 구성

R-tree를 이용한 공간 인덱싱 방법은 공간 객체를 최소로 포함하는 사각형의 영역(MBR : Minimum Boundary Rectangle)으로 근사화하여 관리하는 방법이다[6]. 공간 기본 테이블에 대한 R-tree인덱스의 구축은 공간 객체 집합 S = {S1, …, Sm}를 가지는 공간 기본 테이블에 대하여 각각의 공간 객체를 근사화한 MBR값에 대한 R-tree를 구축하는

것이며, (그림 2-3)과 같이 표현된다. 이 경우 사용자가 임의의 공간 뷰(SV)를 정의한다면, 공간 뷰에 포함되는 공간 기본테이블의 공간 객체에 대한 MBR값을 가지는 R-tree노드는 (그림 2-3)(a)과 같고, R-tree에서 이 노드에 대한 트리 높이는 (그림 2-3)(b)와 같다. 여기서, R-Tree노드가 가질 수 있는 최대 키 값이 n이라고 하고, R-Tree의 최대 높이가 k라고 한다면, 이 R-Tree가 가질 수 있는 최대 공간 객체의 수는 $= n^{k+1}$이고, 높이가 $j(0 < = j < k)$인 노드가 가질 수 있는 하위 노드의 갯수는 $n^{(k-j)+1}$이다. 따라서, 위와 같은 경우 공간 기본 테이블의 모든 객체중에서 임의의 공간 뷰 SV가 가질 수 있는 확률적인 공간 객체 비율은 다음의 식 (2-1)과 같다.

$$SVOR(S) = n^{(k-j)} + 1 / n^{k+1} \qquad (2\text{-}1)$$

즉, 노드의 최대 키 개수가 4이고, 트리의 최대 높이가 5이고, 공간 뷰 정의 영역에 대한 트리의 높이가 4라고 한다면, 이 공간 테이블에 대한 공간 객체 비율 값은 $4^{(5-4)+1}/4^{5+1} = 1/4^4 = 1/256 = 0.0039$ 이다. 전체 공간 객체의 약 0.39%가 공간 뷰에 포함된다는 의미이다. 이 경우 공간 뷰의 MBR값이 기본 공간 테이블의 MBR값의 30%라고 한다면, R-Tree에서 이영역에 대한 서브 트리가 매우 복잡하게 구성되어 있으며, 적은 수의 공간 객체를 찾기 위하여 많은 서브 트리에 대한 탐색을 수행해야 한다. 많은 서브 트리에 대한 검색으로 적은 수의 공간 객체를 검색하게 되면 공간 뷰에 대한 질의처리 수행시 후보공간 객체가 많이 발생하여 전체적인 질의처리 시간과 비용이 높아지게 된다.

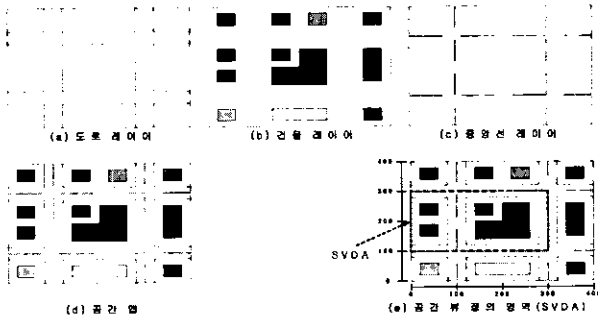
공간 뷰에 대한 질의처리 효율을 높이기 위해서는 공간 인덱스를 이용하여 질의처리를 수행하는 것이 좋다. 공간 뷰는 공간 기본 테이블에 대하여 정의된 가상의 테이블이기 때문에 인덱스를 사용하기 위해서는 공간 뷰를 실체화 하여야 한다.

3. 공간 뷰 객체 분포율 오차와 공간 뷰 관리 자료 구조

본 절에서는 본 논문에서 제안하는 SVMT에서 공간 뷰 실체화 과정에서 R-tree 재구성 여부를 판단하기 위한 기준인 SVODR과 4장의 알고리즘을 수행하기 위하여 필요한 자료구조에 대하여 설명한다. 이를 위하여 먼저, SVODR을 설명하기 위해 필요한 기본적인 용어들을 정의하고, 정의된 용어를 바탕으로 SVODR 값의 오차 값을 이용하여 R-tree 재구성 여부를 결정하는 방법을 설명한다. 마지막으로, SVMT를 위해 필요한 자료구조에 대하여 설명한다.

3.1 Terminology

공간 레이어 또는 기본 공간 테이블(BST : Base Spatial Table)이란 공간 및 비공간 데이터에 대한 정보가 하나의 단위(레코드로)로 관리되며, 이들 레코드들의 집합으로 구성되는 테이블을 말한다. **공간 뷰(Spatial View)**라는 것은 기본 공간 테이블에 대하여 정의된 가상의 테이블을 말한다. 즉, 공간 뷰 정의 질의를 통하여 다수 개의 테이블로부터 유도하여 새롭게 정의된 공간 테이블이다. **공간 맵(SM : Spatial Map)**은 동일한 영역에 존재하는 다수의 공간 기본 테이블을 위치상으로 겹침(Overlay)한 가상 공간 뷰를 말한다. **공간 뷰 정의 영역(SVDA : Spatial View Definition Area)**은 공간 기본 테이블들에 대하여 공간 뷰가 정의할 때, 이 공간 뷰가 정의된 영역의 MBR값을 말한다.

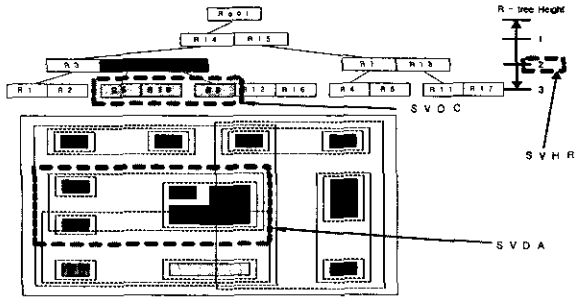


(그림 3-1) 공간 기본 테이블과 공간 맵의 개념

(그림 3-1)은 공간 기본 테이블과 공간 맵의 관계를 보인 것이다. (그림 3-1) (a), (b), (c)는 각각 도로, 건물, 중앙선 레이어(공간 테이블)을 의미한다. (그림 3-1)(d)는 이 들 3개의 테이블에 대하여 동일영역에 존재하는 모든 공간 객체를 overlay하여 표현한 것인 공간 맵이다. 그리고, (그림 3-1)(e)는 (그림 3-1)(d) 공간 맵에 대하여 정의된 공간 뷰 정의영역을 보인 것이다.

최대 공간 객체 수(MSOC : Maximum Spatial Object Count)는 R-tree에서 최대로 포인팅할 수 있는 공간 객체의 수를 말한다. **객체 수(OC : Object Count)**는 공간 기본 테이블 또는 공간 뷰에 포함되는 공간 객체의 수를 말한다. **R-tree에서의 공간 뷰 높이(SVHR : Spatial View Height in R-tree)**는 공간 뷰 정의 영역에 해당하는 공간 객체를 공간 기본 테이블의 R-tree에서 검색하였을 때, 이들 공간 객체를 모두 포함하는 MBR을 가지는 R-tree노드의 높이를 말하는 것이다. **공간 뷰 객체수(SVOC : Spatial View Object Count)**라는 것은 R-tree의 공간 뷰 높이의 노드의 MBR영역에 공간 뷰 정의영역에 포함되는 실제 공간 객체의 개수를 말한다.

(그림 3-2)는 공간 뷰 정의영역, 공간 뷰 높이, 공간 뷰 실체 객체수의 관계를 보인 그림이다. (그림 3-2)(a)는 공간 뷰 높이가 높은 경우 즉, 공간 뷰 정의 영역에 포함되는 공간 객체가 R-tree상에 일부 영역에 포함되는 경우이고, (b)는 공간 뷰 높이가 낮은 경우를 말하며, 공간 객체가 넓게 분포하여 있는 경우를 보인 것이다.



(그림 3-2) SVDA, SVOC 그리고 SVHR의 관계

3.2 공간 뷰 객체 분포율 오차3

제안하는 기법에서는 공간 뷰를 실체화하고, 실체화된 공간 뷰에 대한 질의처리를 빠르고 효율적으로 수행하기 위해서 R-tree 공간 인덱스를 구축하여 사용한다. 이 과정은 공간 뷰 실체화를 위해 발생하는 추가적인 비용이기 때문에 최소한의 비용으로 수행되어야 한다. 최소한의 비용으로 실체화된 공간 뷰의 R-tree 공간 인덱스를 구축하기 위해서는 새로 재 구축한 경우와 공간 기본 테이블의 인덱스를 재 사용한 경우의 비용을 비교하여 판단하여야 한다. 이것을 결정하는 가장 중요한 요인이 공간 뷰 객체 분포율 오차이다.

공간 뷰 객체 분포율 오차는 공간 뷰에 대한 질의 수행 시 발생하는 후보객체들 중에서 실제 공간 뷰에 포함되는 공간 객체가 얼마인지를 나타내는 확률적/실제적 비율 값의 차이이다. 이 값은 SVDA에 대한 확률적/실제적인 공간 객체 분포율을 계산하고, 두 값의 차이에 대한 절대값 연산을 통하여 계산된다. 실체화된 공간 뷰의 공간 인덱스 구축은 계산된 값이 평균 공간 객체 분포율 보다 큰 값이면 공간 뷰에 대한 R-tree 공간 인덱스를 재 구축하고, 적은 값이면 기본 공간 테이블의 공간 인덱스를 재 사용한다. 이렇게 함으로써 실체화된 공간 뷰에 대한 R-tree 공간 인덱스를 최적의 비용으로 구축할 수 있다.

확률적인 공간 뷰 객체 비율(SVOR : Spatial View Object Ratio)은 R-tree가 이론적으로 가질 수 있는 전체 공간 객체 수와 이론적으로 공간 뷰 정의 영역에 포함되는 공간 객체 수의 확률적인 비율을 말하는 것이다. 이 값을 계산하기 위해서는 먼저 공간 기본 테이블에 대하여 구성된 R-tree에 대하여 SVDA 영역에 해당하는 트리 높이를 계산하여야 한다. 공간 뷰 트리 높이는 다음의 (알고리즘 1)을 통하여 계산된다.

```
1 : {
2 : SVRH를 1로 초기화;
3 : intersect, contain값을 0으로 초기화;
4 : for(TreeMaxHeight - 1 ) {
5 : for(현재 노드의 모든 MBR에 대하여 수행 ) {
6 : SVDA와 노드의 MBR값과 교차관계 존재하면
    intersect값을 증가시킴;
7 : SVDA와 노드의 MBR값과 포함관계 존재하면
    contain값을 증가시킴;
8 : if( contain > 0 && intersect > 0 || contain < 0 )
9 : 현재노드를 공간 뷰 트리 높이로 설정하고 종료한다;
10 : }
11 : 현재노드의 하위 노드로 진행;
12 : }
13 : return( FALSE );
14 : }
```

(알고리즘 1) GetHeightOfSVDA(SVDA, R, SI)

(알고리즘 1)의 수행은 R-tree의 루트 노드에서 시작하여 공간 뷰 정의 영역에 대한 트리 높이를 검사한다. 이 검사는 공간 뷰 정의 영역과 노드에 존재하는 모든 MBR값과의 비교를 통하여 수행된다. 먼저, 공간 뷰 정의 영역과 노드의 MBR사이의 교차관계를 조사하고, 그 개수를 카운팅한다. 다음은 공간 뷰 정의 영역과 노드의 MBR 사이의 포함관계를 조사하여 그 개수를 카운팅 한다. 마지막으로 트리 높이를 결정한다. 이 노드가 포함관계가 존재하지 않거나, 포함관계와 교차관계가 존재하면 이 노드를 트리 높이로 결정한다. 만약, 이 노드가 포함관계만 존재하면 하위 노드로 진행한다. 이 과정을 트리의 최대 높이 보다 하나 적은 값까지 진행한다. 즉, 포함관계가 존재하지 않거나 포함관계와 교차관계가 존재하면 이 노드가 공간 뷰 정의 영역을 최소한으로 포함할 수 있는 노드라는 것이고, 그렇지 않다면 하위 노드에 좀 더 작게 포함하는 노드가 존재한다는 의미이다.

실제 공간 뷰 객체 비율이라는 것은 공간 뷰 정의 영역에 해당하는 공간 객체를 R-tree에서 검색한 개수와 공간 기본 테이블의 전체 공간 객체 수와의 비율을 말하는 것이다. 즉, 전체 공간 객체 중에서 공간 뷰 정의 영역에 해당하는 공간 객체 수가 얼마나 되는가를 나타내는 척도이다. 실제 공간 뷰 객체 비율을 구하기 위해서는 공간 뷰 객체 수를 먼저 구해야 한다. 이는 (알고리즘 2)를 통하여 계산할 수 있다.

```
1 : {
2 :   현재 노드 포인터가 NULL값이면 1을 돌려준다;
3 :   for( 현재 노드에 있는 모든 MBR값에 대하여 조사 ) {
4 :     if(SVDA와 교차, 포함 관계가 있는 MBR값이 있는
          지 조사 == TRUE ) {
5 :       GetSpatialViewObjectCount 함수를 재귀호출후
          결과값을 nRet에 저장;
6 :       nRet값이 1(단말노드)인 경우만 카운트 증가시킴;
7 :     }
8 :   }
9 :   return( 0 );
10 : }
```

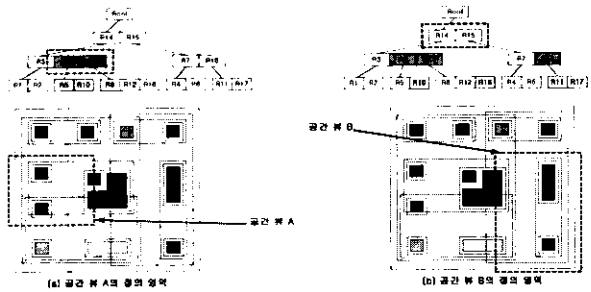
(알고리즘 2) GetSpatialViewObjectCount (CurNode, SVDA, Count)

(알고리즘 2)의 수행은 R-tree의 루트 노드에서 시작하여 공간 뷰 정의 영역에 포함되는 공간 객체 수를 계산한다. 이 검사는 공간 뷰 정의 영역과 노드에 존재하는 모든 MBR값

과의 포함, 교차관계 비교를 통하여 수행된다. 먼저, 현재 전달받은 노드 포인터가 널 값인지를 조사한다. 이 값이 널 값이면 1값을 돌려준다. 이는 이 함수를 호출한 노드가 R-tree의 단말 노드이기 때문이다. 다음은, 현재 노드에 있는 모든 MBR값에 대하여 SVDA와 교차 및 포함 관계가 있는지 조사한다. 교차, 포함관계가 있으면 이 노드가 가르키는 하위 노드에 대하여 이 조사를 다시 수행한다. 다음은, 조사를 수행하고 나온 결과값이 1인 경우에만 Count값을 증가시킨다. 마지막으로 노드에 대한 모든 MBR값에 대한 검사를 마쳤으면 0값을 돌려주며 종료한다.

3.3 실체화된 공간 뷰에 대한 R-tree 인덱스 구축 전략

SVMT는 공간 뷰를 실체화하며, 실체화된 공간 뷰에 대한 빠르고 효율적인 공간 질의처리를 위하여 공간 인덱스를 구축한다. 이 때, 공간 뷰에 인덱스를 새로 재 구축할 것인지, 아니면 공간 기본 테이블의 인덱스를 재 사용할 것인지를 결정하는 가장 중요한 요인이 공간 뷰 객체 분포율 오차이다. 실체화된 공간 뷰의 공간 인덱스 구축은 계산된 공간 뷰 객체 분포율 오차 값이 평균 공간 객체 분포율 보다 큰 값이면 공간 뷰에 대한 R-tree 공간 인덱스를 재 구축하고, 적은 값이면 기본 공간 테이블의 공간 인덱스를 재 사용한다.



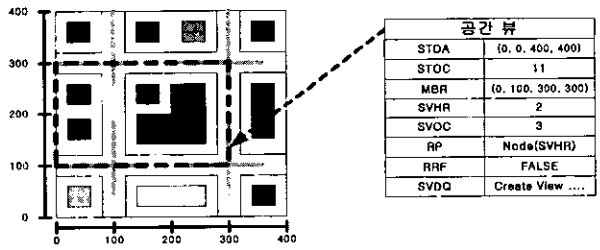
(그림 3-3) 공간 뷰와 SVHR, SVOC값과의 관련성

(그림 3-3)은 최대 키 값이 4, 최대 높이 2인 R-tree와 사용자가 정의한 임의의 공간 뷰 A, B를 도식적으로 표현하고 있다. 실체화된 공간 뷰의 R-tree 인덱스 구축은 공간 뷰 객체 분포율 오차 계산 과정과 이 값을 통한 공간 인덱스 재 구성 여부 판단후 공간 인덱스 구축의 순서로 수행된다. (그림 3-3)을 이용하여 공간 뷰 객체 분포율 오차 계산 과정과 공간 인덱스 재 구성 여부 판단에 대해 설명하면, (a) 경우 높이가 1 이므로 확률적 공간 뷰 객체 비율은 $4^{(2-1)+1}/4^{2+1} = 4^2/4^3 = 1/4 = 0.25$ 이고, 실제 공간 뷰 객체 비율은 총 공간객체 수는 11이고, 여기의 공간객체 수는 3이므로 3/11 = 0.273이다. (b) 경우 높이가 0이므로, 확률적 공간 뷰 객체 비율은 $4^{(2-0)+1}/4^{2+1} = 1.0$ 이고, 총 공간 객체 수는 11이고, 공간 객체수는 4이므로 실제 공간 뷰 객체 비율은 4/11 = 0.367이다. 따라서, SVODR 값은 (a)경우는 0.25 - 0.27 = abs(-0.02) = 0.02이고 (b)경우는 1.0 - 0.376 = abs(0.634) = 0.634 이다. 즉, (a)경우는 예측값과 실제값이 비슷하므로 공간 기본 테이블의 R-tree를 그대로 사용할 수 있으며, (b)경우는 예측값과 실제값에 큰 차이가 있으므로 공간 기본 테이블의 R-tree를 그대로 사용할 수 없다.

(그림 3-3)를 통해 알수 있듯이 (a)경우는 실체화하고 R-tree를 재구성하지 않고 공간 뷰 높이에 해당하는 노드를 공간 뷰 R-tree의 루트로 설정하는 것이 효율적이며, (b)경우는 실체화하고 R-tree를 재구성하는 것이 효율적이다.

3.4 공간 뷰 관리를 위한 자료구조

SVMT에서는 공간 뷰에 대한 효율적인 관리를 위하여 공간 뷰 정보 테이블을 사용한다. 공간 뷰 정보 테이블이라는 것은 공간 뷰에 대한 모든 정보를 가지고 있으며, 공간 데이터베이스의 메타 테이터베이스에 저장되는 테이블을 말한다. 이 자료구조의 구성은 STDA는 공간 뷰가 정의된 공간 맵의 MBR값을 저장한 것이고, STOC는 공간 맵의 전체 객체 수, MBR값은 SVDA는 공간 뷰 정의 영역의 MBR값, SVHR은 SVDA의 R-Tree높이, SVOC는 SVDA의 공간 객체 수, 인덱스 포인터(RP : R-tree Pointer)는 공간 뷰에 대하여 구성된 R-Tree의 루트를 가르키는 포인터 값이다. R-tree 재구성 여부 플래그(RRF : R-tree Reconstruction Flag)공간 뷰 정의 질의어(SVDQ : Spatial View Definition Query)는 공간 뷰를 정의할 때 사용한 공간 SQL을 저장하고 있다.



(그림 3-4) 공간 뷰 정보 테이블

즉, "Create SpatiaView 선택된강남도로 as (Select * from 도로 where Contain(도로.OBJ, MBR(SelectRegion));)"과 같이 공간 뷰가 정의되었다면, 그에 해당하는 자료구조의 내용들은 (그림 3-4)와 같다.

4. 공간 뷰 실체화 관리 기법

본 절에서는 3장에서 설명한 R-tree 재구성 여부의 판단 기준인 공간 뷰 객체 분포율 오차와 공간 뷰 관리 자료구조를 이용하여 공간 뷰 실체화 관리 기법에 대하여 설명한다. 공간 뷰 실체화 관리는 공간 뷰에 대한 R-tree 설정 과정, 공간 뷰 실체화 과정, 공간 뷰에 대한 질의 처리 과정으로 구분된다.

4.1 공간 뷰에 대한 R-tree 설정

공간 뷰에 대한 R-tree 설정이라는 것은 공간 뷰에 대한 R-tree를 재구성하거나, 공간 뷰 높이에 해당하는 노드를 공간 뷰 R-tree의 루트로 설정하는 과정이다. 이 과정은 (알고리즘 3)에 의하여 수행된다.

```
1 : {
2 :   if(SVODR값이 오차한계에 있는지 검사 == TRUE) {
3 :     공간뷰에 높이에 해당하는 노드를 공간뷰 R-tree 의 루트
        로 설정;
4 :   }
5 :   else
6 :     공간 뷰에 대한 R-tree 재구성;
7 : }
```

(알고리즘 3) SetRtreeOnSpatialView

(알고리즘 3)의 수행 과정은 SVODR값이 오차 한계 범위에 존재하면, 공간 뷰 높이에 해당하는 노드를 공간 뷰에 대한 R-tree의 루트로 설정한다. 반대로, SVODR값이 오차 한계 범위에 존재하지 않으면 공간 뷰에 대한 R-tree를 재구성한다.

4.2 공간 뷰 실체화

공간 뷰 실체화 과정은 공간 뷰 객체 분포율 오차를 계산하여 이 값이 공간 뷰 객체 분포율 오차 범위에 포함되는 경우 실체화를 수행하는 과정이다. 먼저, (알고리즘 3)을 이용하여 공간 객체 분포율 오차를 계산하고, 이 값이 공간 뷰 객체 분포율 오차 범위내에 포함되는지를 조사한다. SVDOR값이 오차 범위에 포함되면 실체화 과정을 수행하고 동시에 공간 뷰에 대한 R-tree 루트를 공간 뷰 높이에 해당하는 노드로 설정하고, 이 범위에 포함되지 않으면 실체화를 수행하고 R-tree를 재구성한다. 이 과정은 (알고리즘 4)과 (알고리즘 5)를 통하여 처리된다.

```
1 : {
2 :   h = 트리 높이 계산;
3 :   이론적 공간 객체 분포율(ratio1) = h가 가르키는 노드 개수/
                                        전체 노드 개수;
4 :   oc = 공간 뷰 객체 수 계산; ratio2 = 공간 뷰 객체수/
                                        총 객체수;
5 :   SVODR = 절대값-abs(ratio1-ratio2);
6 :   return( SVODR )
7 : }
```

(알고리즘 4) CalcSVODR

(알고리즘 4)의 수행은 먼저, 공간 뷰 정의 영역의 트리 높이를 계산한다. 이를 이용하여 확률적인 공간 뷰 객체 비율과 실제 공간 뷰 객체 비율을 구한다. 다음으로 확률적 공간 뷰 객체 비율과 실제 공간 뷰 객체 비율의 차이에 대한 절대값을 구한다. 마지막으로, 이 값을 돌려주고 종료한다.

```
1 : {
2 :   SVIT에서 공간 뷰에 대한 레코드 검색;
3 :   공간 뷰 높이에 대한 공간 뷰 객체 오차 한계율(g) 계산;
4 :   새로운 테이블 생성;
5 :   질의 수행하여 결과 레코드를 새로운 테이블에 저장;
6 :   R-Tree 설정 알고리즘 수행;
7 :   SVHR, SVOC, SVDA, RRF등의 공간 뷰 정보 내용 변경;
8 : }
```

(알고리즘 5) SpatialViewMaterialize

(알고리즘 5)의 수행은 먼저, 공간 뷰 정보 테이블에서 공간 뷰에 대한 정보를 검색하고, 공간 뷰에 대한 객체 오차 한계율을 구한다. 다음으로, 새로운 테이블을 생성하고, 질의를 수행하여 결과 레코드를 새로 생성한 테이블에 저장한다. 다음으로, R-tree 설정 알고리즘을 수행한다. 마지막으로, 공간 뷰 정보테이블의 공간 뷰에 대한 정보를 갱신하여 저장한다.

4.3 공간 뷰에 대한 공간 질의 수행

공간 뷰에 대한 관리라는 것은 공간 뷰에 대한 검색 질의와 추가, 삭제, 삽입, 변경등의 변경연산에 대한 처리를 말하는 것이다. 이는 공간 뷰에 대한 검색 질의 수행 단계과 공간 뷰 변경 관리 단계로 구분되어 처리된다.

4.2.1 공간 뷰 검색 질의 수행 단계

공간 뷰에 대한 질의 수행 방법은 공간 뷰에 대하여 공간 질의를 수행하며, 수행후 공간 질의 수행 비용과 질의 수행 예상 비용을 비교하여 그 값이 질의 수행 예상 비용보다 R-tree에 대한 설정의 변경한다. 이 과정은 (알고리즘 6)을 통하여 수행된다.

```
1 : {
2 :   SVIT에서 공간 뷰에 대한 레코드 검색;
3 :   공간 뷰(가상, 실체화)에 대한 검색 질의 수행;
4 :   if( 질의 수행 시간 > 질의 수행 예상시간 ) {
5 :   R-tree 설정 알고리즘 수행(R-tree 재구성);
6 :   }
7 :   SVHR, SVOC, SVDA, RRF등의 공간 뷰 정보 내용 변경;
8 : }
```

(알고리즘 6) SpatialViewSearchQuery

(알고리즘 6)의 수행 과정은 먼저, 공간 뷰에 대한 검색 질의를 수행한다. 검색 질의 수행후, 실제 질의 수행 시간과 공간 뷰 객체 검색 효율을 이용하여 계산한 검색 질의 예상 시간을 비교하여 실제 값이 예상 값보다 크면 R-tree 설정 알고리즘을 이용하여 R-tree를 재구성한다.

4.2.2 공간 뷰 변경 관리 단계

실체화된 공간 뷰의 관리 방법은 공간 기본 테이블에 대한 변경 발생시 실체화된 공간 뷰에 대하여 변경을 수행하며, 변경된 내용에 의하여 실체화된 공간 뷰에 대한 R-tree

를 재구성 여부를 판단하여, 변경의 필요성이 있으면 재구성한다. 이 과정을 수행하는 알고리즘은 (알고리즘 7)과 같다.

```
1 : {
2 :   공간 기본 테이블에 변경된 레코드 변경(추가, 삭제, 추가);
3 :   실체화된 공간 뷰에 변경된 레코드 변경(추가, 삭제, 추가);
4 :   공간 뷰 객체 분포율 오차 계산;
5 :   if(SVODR값이 오차 한계에 포함되는지 검사 == TRUE) {
6 :      R-Tree 설정 알고리즘 수행(비 재구성);
7 :   }
8 :   else
9 :     R-Tree 설정 알고리즘 수행(재 구성)
10 :  SVHR, SVOC, SVDA, RRF등의 공간 뷰 정보 내용 변경;
11 : }
```

(알고리즘 7) SpatialViewMaintenance

(알고리즘 7)의 수행 과정은 먼저, 공간 기본 테이블에 대한 변경 작업을 수행하고, 실체화된 공간 뷰에도 변경 과정을 수행한다. 이 때, 공간 뷰 객체 오차 범위값을 계산하여 이 값이 공간 뷰 객체 오차 한계에 포함되면 R-tree를 재구성하지 않는다. 이때, 기존에 공간 뷰가 R-tree가 재구성된 경우에만 공간 뷰 높이에 해당하는 노드를 공간 뷰의 R-tree로 설정한다. 반대로, 오차 한계를 벗어나면 R-Tree를 재구성한다. 이때, 기존에 공간 뷰가 R-tree가 재구성이 아닌 경우에만 R-tree를 재구성한다. 마지막으로, 공간 뷰 정보 테이블의 레코드 값들을 변경한다.

5. 실험 및 성능평가

본 절에서는 본 논문에서 제안한 R-Tree 재구성을 이용한 공간 뷰 실체화 관리 기법의 성능평가에 대하여 설명한다. 먼저, 제안한 기법의 성능평가 환경에 대하여 설명하고, 제안 기법과 기존의 공간 뷰 관리 기법에 대한 성능 비교를 수행한다.

5.1 성능 평가 환경

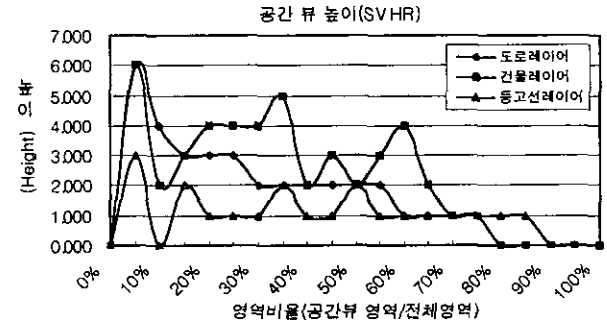
제안된 R-tree 재구성을 이용한 공간 뷰 실체화 관리 기법은 IBM호환 PC상에서 Visual C++와 GIS 응용 프로그램 개발 툴킷인 GEOMania/GDKv2.5[15]를 이용하여 구현되었다. 구현 된 시스템의 사양은 펜티엄III 800MHz, 128MB 메인 메모리이다. 제안기법의 테스트는 공간 기본 테이블에 대한 공간 뷰의 영역 비율을 0%에서 100%로 증가시키며 수행하였고, R-tree의 최대 키 값을 4로 하여 실험하였다. 실험 평가에 사용된 도로, 건물, 등고선등의 공간 기본테이블은 파일의 크기는 비슷하고, 다양한 공간 객체 분포율을 가지도록 선택하였다. <표 5-1>은 이 공간 기본 테이블의 세부적인 속성값을 보인 것이다.

〈표 5-1〉 성능평가 환경

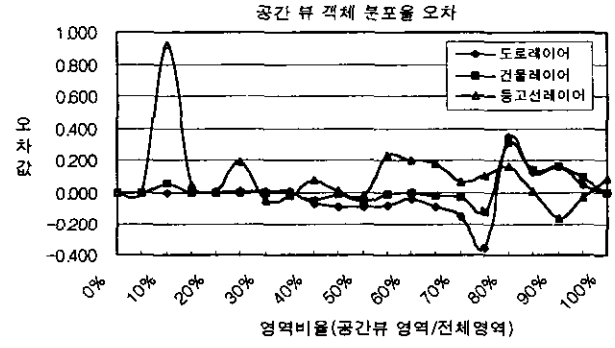
공간 기본 테이블 이름	사이즈	공간 객체 개수	최대 트리 높이
도로	1,394KB	6,482	7
건물	1,348KB	7,739	7
등고선	1,898KB	181	4

5.2 제안기법의 성능 평가

성능평가는 <표 5-1>의 데이터와 파라미터를 이용하여 기존에 실체화된 공간 뷰에 대하여 공간 인덱스를 구축하지 않는 기법과 제안 기법을 비교하였다. 두 가법의 비교를 위하여 확률적/실제적 공간 뷰 객체 분포율을 먼저 계산하고, 그 다음 두 기법의 질의 수행시간을 비교하였다.

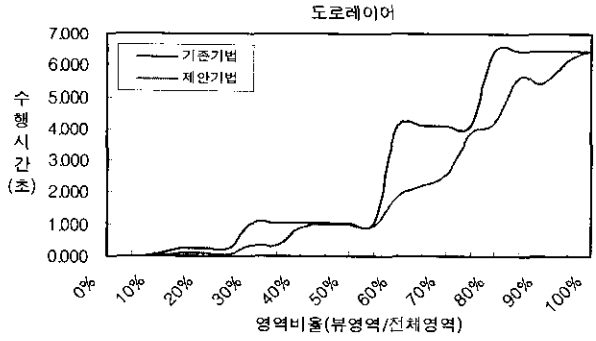


(그림 5-1) 공간 뷰 높이(SVHR)

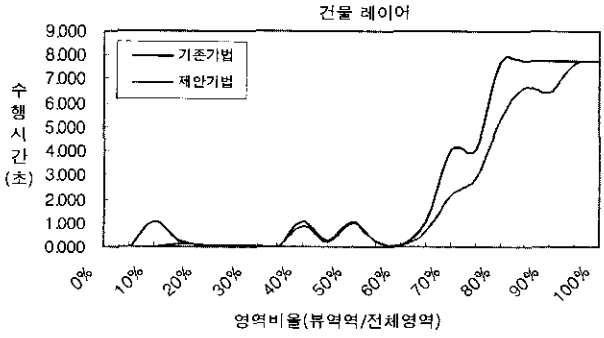


(그림 5-2) 공간 뷰 객체 분포율 오차

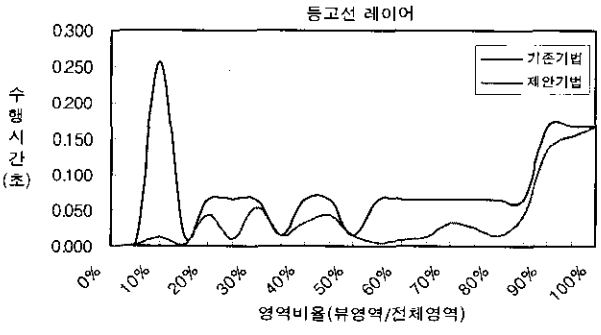
(그림 5-1)은 전체영역에 대한 공간 뷰 영역이 증가함에 따라 도로, 건물, 등고선 레이어의 공간 뷰 높이 변화를 보인 것이고, (그림 5-2)는 SVHR값, 확률적/실제적 객체 분포율과 오차 값을 보인 것이다. 그림에서 보는 바와 같이 공간 기본테이블의 전체 영역에 대한 공간 뷰 영역 비율이 45%에서 100%로 증가함에 따라서 공간 뷰 높이는 낮아지고, SVODR의 오차 값 역시 증가한다. 일반적으로 공간 뷰 높이가 낮아지면 공간 질의 처리시 여과단계에서 제외되는 후보객체수가 증가한다. 후보객체가 증가하게 되면 전체적인 공간 질의처리 시간이 증가하게 되어, 빠른 질의처리시간을 보장하기 힘들다. 따라서, 실체화된 뷰 영역이 45~100%에 해당하는 경우에는 공간 뷰를 실체화하고 공간 인덱스를 재구성하는 것이 효율적이라고 할 수 있다.



(그림 5-3) 도로레이어에 대한 질의 수행시간 비교



(그림 5-4) 건물레이어에 대한 질의 수행시간 비교



(그림 5-5) 등고선레이어에 대한 질의 수행시간 비교

(그림 5-5), (그림 5-6), (그림 5-7)은 위 그림은 전체영역에 대한 공간 뷰 영역이 증가함에 따라 도로, 건물, 등고선 레이어에 대한 기존기법과 재안기법과의 공간 질의처리 시간을 비교한 결과이다. 그림에서 보는 바와 같이 제안기법은 공간 뷰 영역비율의 증가하더라도 일정한 수행시간을 보임을 알 수 있다. 이러한 결과는 제안 기법이 기존 기법에 비하여 공간 뷰 객체 분포율 오차가 증가하면 공간 뷰를 실체화하고 공간 인덱스를 재 구축하여 사용하기 때문이다.

6. 결 론

본 논문에서는 R-tree 재구성 방법을 이용한 공간 뷰 실체화 관리 기법(SVMT : Spatial View Materialization Technique)을 제안하였다. 제안한 SVMT는 공간 뷰를 실체화하고 실체화된 공간 뷰에 대하여 공간 질의를 처리하여 공간 질의 처리 수행 속도를 빠르게 하였으며, 공간 뷰 객체 분포율 오차를 이용하여 공간 뷰에 대한 R-tree 재구성 여부를 판단하여 실체화된 공간 뷰 관리비용을 최소화 하였다. 또한 공간 뷰를 실체화하여 동일 공간 뷰 접근을 통해 발생하는 반복적인 질의 변환 수행시간을 제거하였다. 그리고, SVMT 방법의 성능 평가 결과를 통해서 공간 뷰에 대한 인덱스를 공간객체 분포율 오차값이 큰 경우에는 공간 뷰에 대한 R-tree를 재구성하는 것이 서버의 부하와 성능 향상을 이룩할 수 있다. 또한, 성능 평가를 통하여 공간 뷰 관리를 위한 최적의 공간 뷰 객체 분포율 오차값을 제시하였다.

본 논문에서 제안한 SVMT 방법은 공간 데이터베이스 시스템에 적용을 통하여 클라이언트/서버 환경과 같은 다중 사용자, 동시 작업 환경에서 공간 뷰에 대한 빠른 접근 속도와 빠른 질의 응답을 제공할 수 있으며, 서버의 병목현상과 네트워크 부하를 최소화할 수 있는 효율적인 공간 뷰 관리할 수 있다는 장점을 가진다. 향후 연구 과제로는 클라이언트/서버 환경에서 이런 실체화된 뷰를 클라이언트와 서버가 공동으로 관리하여 서버 병목 현상과 네트워크 부하를 좀 더 최소화할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Ralf Hartmut Güting, "An Introduction to Spatial Database Systems," VLDB Journal, Vol.3, No.4, pp.357-399, October, 1994.

[2] Michael J. Franklin, Michael J. Carey and Miron Livny, "Transactional Client-Server Chche Consistency : Alternatives and Performance," ACM Transactions on Database Systems, Vol.22, No.3, pp.315-363, September, 1997.

[3] Hasan Davulcu, Juliana Freire, Michael Kifer, I. V. Ramakrishnan, "A Layered Architecutre for Querying Dynamic Web Content," ACM Sigmod Record, 1999.

[4] Volker Gaede and Oliver Günther, "Multidimensional Access Methods," ACM Computing Surveys, Vol.30, No.2, pp.170-231, June, 1998.

[5] Bernhard Seeger and Hans-Peter Kriegel, "Techinques for Design and Implementation of Efficient Spatial Access Methods," Proceedings of the 14th VLDB Conference L.A., California, pp.360-371, 1988.

[6] Antonin Guttman, "R-Trees : A Dynamic Index Structure for Spatial Searching," ACM, pp.47-57, 1984.

[7] Timos Sellis, Nick Roussopoulos and Chrisots Faloutsos, "The R+-Tree : A Dynamic Index for Multi-Dimensional Objects," Proceedings of the 13th VLDB Conference, Brigh-

ton, pp.507-518, 1987.

[8] Norbert Beckmann, Hans-Peter Kriegel Ralf Schneider and Bernhard Seeger, "The R*-Tree : An Efficient and Robust Access method for Points and Rectangles," ACM, pp.322-331, 1990.

[9] C. J. Date, "An Introduction to Database Systems," Addison-Wesley publishing Company, 7th Edition, 2000.

[10] Volker Gaede and Oliver Gunther, "Multidimensional Access Methods," ACM Computing Surveys, Vol.30, No.2, June, pp.170-231, 1998.

[11] Ashish Gupta, Inderpal Singh Mumick and V. S. Subrahmanian, "Maintaining Views Incrementally," SIGMOD, Washington, DC, USA, pp.157-166, 1993.

[12] Yue Zhuge, Janet L. Wiener and Hector Garcia-Molina, "Multiple View Consisency for Data Warehousing," Proceedings of the 13th International Conference on Data Engineering, pp.289-300, 1999.

[13] 문상호, "실체화된 객체 지향 공간 뷰의 점진적 변경 알고리즘", 부산대학교 컴퓨터 공학과 공학박사학위논문, 1998.

[14] 정보흥, 오영환, 박동선, 김재홍, 배해영, "Simplification Technique for Spatial View with 1 : N Mapping Cardinality," International Conference of GIS 2000, 2000.

[15] GEOMania/GDK v2.5, GEOMania. Co. Ltd. 2000.

정 보 흥

e-mail : bigseven@netsgo.com

1996년 인하대학교 전자계산공학과(공학사)

1998년 인하대학교 대학원 전자계산공학과 (공학석사)

1998년~현재 인하대학교 대학원 전자계산공학과 박사과정

관심분야 : Database, GIS, 분산 GIS, Clustering, LDAP

배 해 영

e-mail : hybae@dragon.inha.ac.kr

1974년 인하대학교 응용물리학과(공학사)

1978년 연세대학교 대학원 전자계산학과 (공학석사)

1989년 숭실대학교 대학원 전자계산학과 (공학박사)

1992년~1994년 인하대학교 전자계산소소장

1982년~현재 인하대학교 전자계산공학과 교수

1999년~현재 정보통신부 국가GIS기술 개발분과 자문위원

1999년~현재 지능형 GIS연구센터 소장

1999년~현재 연변과학기술 대학겸직 교수

2000년~현재 중경우전대학 대학원 명예교수

관심분야 : 데이터베이스, GIS, 실시간 데이터베이스 시스템, 이동 데이터베이스 시스템