실체화된 공간뷰의 일관성 유지를 위한 점진적 변경 알고리즘의 성능 평가

문 상 호[†]·반 재 훈[††]·홍 봉 희[†††]

요 약

본 논문에서는 실체화된 공간뷰의 일관성 유지를 위하여 뷰 유도관련성을 이용한 두가지 방법인 값 복사 실체화 방법과 식별자 유지 실체화 방법에 대하여 실험 평가한다. 이 실험 결과, 뷰 유도관련성을 이용한 점진적 변경 방법을 값 복사 실체화 방법과 식별자 유지 실체화 방법과 비교할 때, 공간뷰 객체에 대한 변경 시간이 거의 차이가 나지 않는다. 그리고 공간뷰에 대한 질의 수행 실험 결과, 값 복사 실체화가 식별자 유지 실체화보다 질의 수행 시간이 훨씬 빠르다. 결론적으로 실체화와 점진적 변경에 따른 성능 평가를 고려해 볼 때, 전체적으로 식별자 유지 실체화보다 값 복사 실체화를 기반으로 뷰 유도관련성을 이용한 점진적 변경 방법이 가장 바람직하다.

Performance Evaluation of Incremental Update Algorithms for Consistency Maintenance of Materialized Spatial Views

Sang-Ho Moon[†]·Chae-Hoon Ban[††]·Bong-Hee Hong[†††]

ABSTRACT

In order to evaluate the performance of incremental update algorithms, we perform experimental tests on the time of updating view objects. In this paper, the incremental update algorithms are evaluated on two kinds of materialized methods : materialization by value-copy and materialization by preserving object identifiers (OIDs). The result of performance evaluation shows that there is little difference in the updating time of view objects between two materialization methods. The evaluation of query processing on spatial views shows that materialization by value-copy is much better than materialization by preserving OIDs. As the results of overall performance evaluation, it is more desirable to use the incremental update method based on materialization by value-copy than the incremental update method based on materialization by preserving OIDs.

키워드 : 실체뷰 관리(materialized view maintenance), 점진적 변경(incremental update), 공간뷰(spatial view)

1. 서 론

지리정보시스템(GIS : Geographic Information System)에서 지리객체(geographic object)를 모델링할 때 다양한 사용자 요구 조건들이 있으며, 특히 사용자의 관점(user's perspective view)에 따른 지리객체의 서로 다른 공간 표현(spatial representation)을 지원하는 것이 매우 중요하다[3, 4, 16, 17]. 본 논문에서는 실세계의 지리객체들에 대한 다양한 사용자 관점을 지원하기 위하여, 객체지향 데이타베이스의 뷰 개념을 확장한 공간뷰(spatial view)를 이용한다. 공간뷰는 공간 데이터베이스(spatial database)에 대한 사용자 관점을 나타내며, 하나 이상의 소스클래스(source class)로부터 지리객체를 동일한 또는 상이한 공간 표현을 가지는 가상클래스(virtual class)로 정의된다.

일반적으로 공간 질의는 복잡한 공간 연산을 통하여 수행되며, 공간 질의의 대상인 공간 데이터베이스는 많은 양의 복잡한 지리객체들로 구성된다. 따라서 질의 수정(query modification/substitution)은 공간뷰에 대한 사용자 질의 수행 방법으로 부적합하다. 왜냐하면 공간뷰에 대한 질의를 수행할 때마다 뷰-정의 질의(view-defining query)인 공간 질의를 반복적으로 수행해야 하므로 관계 뷰나 객체지향 뷰와 같은 전통적인 뷰(classical view)보다 더 많은 시간이 소요되기 때문이다. 따라서 공간뷰에 대한 질의 수행 시간을 향상시키기 위해서는 뷰 실체화(view materialization) 방법이 적합하다.

뷰 실체화에서는 기본클래스의 객체의 변경에 따라 실체

※ 이 논문은 2000년도 한국학술진흥재단의 지원에 의해 연구되었음 (KRF-2000-003-E00243).
† 정 회 원 : 부산외국어대학교 컴퓨터전자공학부 교수
†† 정 회 원 : 경남정보대학 인터넷상거래과 교수
††† 정 회 원 : 부산대학교 컴퓨터공학과 교수
논문접수 : 2002년 1월 18일, 심사완료 : 2002년 4월 19일

화된 뷰 객체의 일관성을 유지해야 하는 문제(materialized view maintenance)가 있다[2, 7, 11]. 실체화된 뷰 관리를 위한 기존의 방법으로는 재수행(re-execution/re-materialization)과 점진적 변경(incremental update)이 있다[1, 2, 7, 12, 13]. 재수행 방법은 소스객체가 변경되었을 때, 이전에 실체화된 뷰 객체를 무시하고 뷰의 모든 객체를 재생성하는 방법이다. 반면에 점진적 변경 방법은 변경된 소스객체와 관련된 뷰 객체만을 변경하는 방법이다[1, 2, 12]. 따라서 점진적 변경 방법은 재수행 방법에 비하여 변경 속도가 빠르고 효율적이지만 변경 알고리즘이 다소 복잡해진다.

실체화된 공간뷰의 일관성 유지를 위한 점진적 변경에서는 소스객체의 변경에 따라 영향받는 뷰 객체를 신속히 검색하는 것이 중요한 문제이다. 이를 위해 본 저자들이 이전 연구에서 공간뷰와 대응되는 소스클래스 및 객체간에 유도관련성(derivation relationship)을 제시하고, 소스객체의 변경에 따른 공간뷰 객체를 효율적으로 변경하기 위하여 뷰 유도관련성을 이용한 점진적 변경 알고리즘을 제시하였다[16, 18]. 이 알고리즘에서는 공간뷰 객체와 소스객체간의 카디널리티 비율(cardinality ratio)인 일대일, 다대일, 일대다, 다대다 관계를 기본으로 한다.

본 논문에서는 뷰 유도관련성을 이용한 점진적 변경 알고리즘의 효율성을 검증하기 위한 성능 평가가 목적이다. 즉, 성능 평가를 통하여 뷰 유도관련성을 이용하여 실체화된 공간뷰 객체를 변경하였을 경우에 얼마나 빠르게 처리하는가를 보여준다. 이를 위하여 먼저 뷰 유도관련성을 이용한 점진적 변경 방법과 이용하지 않는 방법과의 실험 비교를 수행한다.

뷰 유도관련성을 이용한 점진적 변경 방법은 공간뷰 객체의 실체화 방법에 따라 구현 방법에 차이가 있을 수 있다. 즉, 값 복사에 의한 실체화 방법과 식별자 유지에 의한 실체화 방법에 따라 달라질 수 있다. 따라서 본 논문에서는 뷰 유도관련성을 이용한 점진적 변경 방법을 식별자 유지에 의한 실체화 방법과 값 복사에 의한 실체화 방법에 각각 적용한 경우에 대하여 성능 평가를 수행한다. 이 실험에서는 소스객체의 변경에 따른 공간뷰 객체의 변경 시간뿐만 아니라 공간뷰에 대한 질의 수행 시간을 같이 평가하여 종합적인 검토를 할 수 있도록 한다.

2. 관련 연구

공간뷰에 대한 실체화와 실체화된 공간뷰의 점진적 변경에 대한 관련 연구는 거의 없다. 다만 본 논문에서 다루는 공간뷰는 객체지향 뷰를 기반으로 제시하기 때문에, 관련 연구로 객체지향 뷰에 대한 실체화와 점진적 변경을 다룬다.

먼저 실체화된 객체지향 뷰에 대한 관련 연구로는 O_2 뷰와 MultiView가 있다. O_2 뷰는 객체지향 DBMS인 O_2에서 객체지향 뷰를 정의하기 위하여 프로토타입으로 구현되었다[5]. O_2 뷰는 데이터베이스에 있는 실제스키마(real schema)나 다른 가상스키마들로부터 가상스키마(virtual schema)를 정의할 수 있다. 그리고 가상스키마를 정의하기 위하여 가상클래스(virtual class)와 상상클래스(imaginary class)를 제공한다.

MultiView는 객체지향 뷰를 지원하기 위하여 상용 객체지향 DBMS인 Geomstone 상에서 구현되었다[6, 10, 11]. MultiView는 클래스 뷰(class view)와 스키마 뷰(schema view)를 모두 지원하고 있으며, 클래스 뷰는 기본클래스와 함께 하나의 상속계층 구조(inheritance hierarchy)를 형성한다. 즉, 객체지향 뷰를 전역스키마(global schema)중에서 사용자가 선택한 가상적인 서브스키마로 정의하고, 가상클래스를 소스클래스에 대한 질의에 의해 유도되는 뷰 클래스로 정의한다. 실체화된 뷰 객체는 소스객체에 대한 식별자를 유지하며, 뷰 객체에 새롭게 정의된 애트리뷰트들에 대한 값은 내부에 저장된다.

객체지향 뷰의 실체화 방법을 정리하면 크게 2가지가 있다[8]. 첫 번째는 뷰 객체를 실체화 시킬때 실제 데이타를 저장하는 방식이다. 이 방법은 뷰 객체에 실제 데이타가 저장되어 있으므로 뷰에 대한 질의 수행시 속도가 빠르다. 그러나 뷰 객체와 소스객체간에 데이타가 중복되므로, 중복된 데이터에 대한 일관성 유지 문제가 발생한다. 두 번째는 실체화된 뷰 객체에 데이타가 아닌 소스객체의 식별자를 저장하는 방식이다. 이 방법은 객체식별자를 이용하여 뷰 객체와 소스객체간의 데이타를 공유하며, 데이타의 중복이 없으므로 데이타의 일관성 유지가 쉽다. 그러나 뷰 객체를 검색하기 위하여 소스객체를 추가로 접근해야 하는 오버헤드가 있다.

실체화된 객체지향 뷰의 변경에 관한 연구로는, MultiView에서 실체화된 뷰 객체를 변경하기 위하여 점진적 변경 알고리즘을 제시하였다[11]. 이 알고리즘에서는 등록 서비스(registration service)를 이용하여 변경된 소스객체와 관련된 뷰 클래스를 선택한다. 즉 유도 계층 구조(derivation hierarchy)를 별도로 구성하고, 이 정보를 이용하여 뷰 클래스 정의에 있는 Where 조건문을 검사하는 횟수를 줄였다. 등록 서비스는 유도 계층 구조를 구성하기 위해 가상클래스가 정의될 때, 가상 클래스 정의의 Where문에서 사용된 각 소스클래스의 애트리뷰트에 해당 가상 클래스를 등록하는 것이다. 즉, 모든 소스클래스는 자신으로부터 유도된 가상 클래스를 애트리뷰트 별로 분류하여 관리한다.

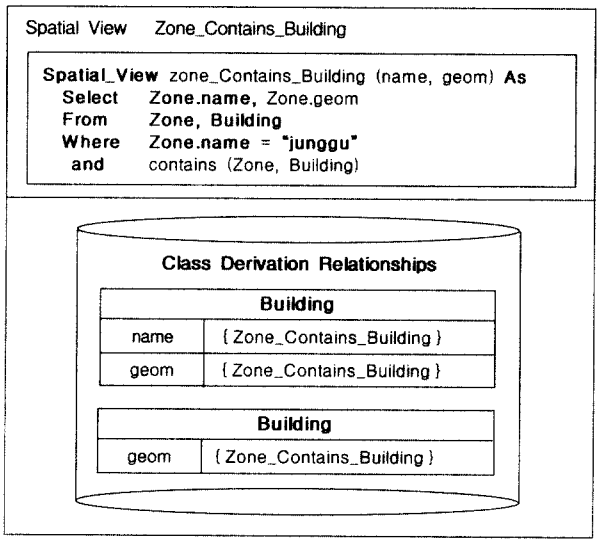
3. 유도관련성

이 장에서는 점진적 변경 알고리즘의 이해를 돕기 위하여 핵심 정보인 유도관련성에 대하여 간략하게 설명하고, 세부적인 내용은 [16, 18]에서 자세하게 기술되어 있다.

3.1 클래스 유도관련성(CDR : Class Derivation Relationship)

소스객체가 변경되었을 때, 이 소스객체의 클래스로부터 유도된 공간뷰들을 찾기 위하여 모든 공간뷰의 뷰-정의 질의에서 From절을 분석하는 것은 매우 비효율적이다. 따라서 공간뷰 정의를 분석하지 않고 변경된 소스객체의 클래스로부터 유도된 공간뷰를 빨리 검색하는 방법이 필요하다.

클래스 유도관련성을 구성하기 위하여, 뷰-정의 질의의 Where절에서 사용한 소스클래스의 애트리뷰트와 SELECT 절에서 사용한 소스클래스의 애트리뷰트별로 유도된 공간뷰의 식별자를 저장한다. 예를들어 (그림 1)과 같이 공간뷰 Zone_Contains_Building을 정의될 때, SELECT절에 기술된 애트리뷰트 리스트와 WHERE절에서 기술된 프레디키트를 분석하여 클래스 유도관련성을 생성한다. 먼저 WHERE절에서 사용된 Zone 클래스의 name 애트리뷰트에 Zone_Contains_Building 공간뷰의 식별자를 저장한다. 그리고 WHERE절에서 프레디키트로 사용된 공간관련성연산자 contains 문을 분석하여 Zone 클래스와 Building 클래스의 geom 애트리뷰트 각각에 Zone_Contains_Building 공간뷰의 식별자를 저장한다. 그리고 SELECT절의 애트리뷰트 리스트를 분석하여 동일한 방법으로 공간뷰의 식별자를 저장한다.



(그림 1) 클래스 유도관련성의 예

3.2 뷰 유도관련성(VDR : View Derivation Relationship)

실체화된 공간뷰의 일관성 유지에서 소스객체가 변경된 경우에 직접적으로 영향받는 뷰 객체를 어떻게 찾느냐가 중요한 문제이다. 이 문제를 해결하기 위하여 본 논문에서는 뷰 객체와 소스객체간에 뷰 유도관련성을 제시한다. 뷰 유도관련성은 뷰 객체와 소스객체의 식별자들로 구성되며, 뷰 객체와 대응되는 소스객체들간에 유도 관계를 나타낸다.

뷰 유도관련성 타입 VDR은 실체화된 뷰 객체 SVO_m과 대응되는 소스객체 간의 결합(association)들의 집합으로 정의되며, 여기서 j는 소스클래스의 인덱스 번호를 나타낸다. 그리고 VDR은 뷰 유도관련성 인스턴스인 VDR_i의 집합이며, VDR_i는 m개의 SVO_m과 n개의 결합으로 표현된다. 여기서 SVO_m은 공간뷰 클래스 SVC_k의 객체 집합에서의 임의의 원소 ($SVO_m \in [SVC_k]$)를 나타내며, SO^{j}_{n}는 공간뷰 클래스 SVC_k와 대응하는 소스클래스 SC_j의 객체 집합에서의 임의의 원소($SO^{j}_{n} \in [SC_j]$)를 나타낸다. 따라서 VDR은 $[SVC_k] \times [SC^1] \times \cdots \times [SC^j] \times \cdots \times [SC^n]$와 같은 카티션 프로덕트(cartesian product)의 부분 집합으로 정의할 수 있다. 앞에서 정의된 이러한 성질을 이용하여 VDR의 i번째 인스턴스를 다음과 같이 정형화하여 정의한다.

$$VDR_i = \langle SVO_m, SO^{j}_{n} \rangle$$

여기서 SO^{j}_{n}는 뷰 객체을 유도하는데 사용된 소스클래스의 객체를 의미한다. 만약 공간뷰 객체 SVO_m이 오직 하나의 소스클래스로부터 유도되었다면, j의 값은 1이 된다($j = 1$). 그리고 공간뷰 객체 SVO_m이 두 개의 소스클래스로부터 유도된다면, VDR_i는 다음과 같이 표현된다.

$$VDR_i = \langle SVO_m, SO^{1}_{n1}, SO^{2}_{n2} \rangle$$

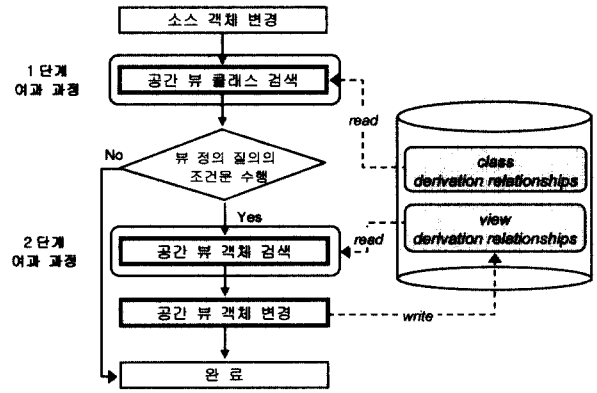
공간뷰 객체와 소스객체들간의 관련성 인스턴스의 수를 카디널리티 비율로 정의한다. 일반적으로 이진관련성(binary relationship)에 대한 카디널리티 비율은 일대일(1 : 1), 다대일(N : 1), 일대다(1 : N), 다대다(N : M)이다. 본 논문에서 뷰 유도관련성을 카디널리티 비율로 분류하는 이유는 공간뷰에 대한 변경 시멘틱이 카디널리티 비율에 따라 서로 달라지기 때문이다.

4. 점진적 변경 알고리즘

소스객체가 변경이 되었을 때, 먼저 변경된 소스객체의 클래스와 관련된 공간뷰 클래스들을 검색한 후에, 이 뷰 클래스에서 소스객체의 변경에 직접적으로 영향받는 뷰 객체들을 변경해야 한다. 이를 위하여 본 논문에서 제시하는 변경 알고리즘은 공간뷰에 관련된 클래스 유도관련성(이하 CDR로 표기)과 뷰 유도관련성(이하 VDR로 표기)을 이용한다. 즉, CDR을 이용하여 변경된 소스객체에 대하여 영향받는 공간뷰 클래스들을 검색하고, VDR을 이용하여 소스객체의 변경에 영향받는 공간뷰 객체들을 검색한다. (그림 2)는 실체화된 공간뷰의 일관성 유지를 위한 점진적 변경 알고리즘의 전체적인 수행 과정이다.

변경 알고리즘에서 1단계 여과 과정은 변경된 소스객체와 관련된 공간뷰 클래스를 검색하는 것으로, CDR에서 변경된 객체의 클래스 이름과 애트리뷰트를 이용한다. 1단계와 2단계 과정의 중간 과정은 변경된 소스객체가 공간뷰 클래스에 영향을 미치는지를 검사하는 과정이다. 즉, 변경

된 소스객체가 1단계 과정에서 검색된 공간뷰의 정의 질의에 있는 조건문(Where절)을 만족하는지를 검사한다. 만약 조건문을 만족한다면 변경된 소스객체가 공간뷰에 영향을 미치므로 2단계 과정으로 넘어가지만, 그렇지 않다면 다음 뷰 클래스에 대하여 계속 검사를 수행한다.



(그림 2) 점진적 변경 알고리즘 수행 과정

2단계 여과 과정은 변경된 소스객체에 영향받는 뷰 객체를 검색하는 과정으로 VDR을 이용한다. 이 과정에서는 변경된 소스객체를 이용하여 VDR에서 관련된 인스턴스를 검색한다. 실제 이 과정에서는 공간뷰 객체의 변경에 필요한 모든 정보를 VDR에서 검색한다. 공간뷰 객체의 변경 과정은 소스객체로부터 유도된 뷰 객체를 변경하는 과정으로, 뷰 객체와 소스객체간의 카디널리티 비율에 따라 뷰 객체의 변경 시맨틱이 달라진다. 이 과정에서는 뷰 객체 변경을 수행한 후에 반드시 VDR을 변경해야 한다. 그러나 뷰 객체와 소스객체간의 카디널리티 비율에 따라 공간뷰 객체를 변경시키지 않고 단순히 VDR만 변경할 수 있다.

소스객체의 변경에 따른 실체화된 공간뷰 객체의 일관성 유지를 위한 변경 시맨틱에서는 먼저 소스객체의 변경(update) 유형을 고려해야 한다. 일반적으로 실체화된 뷰 관리에서 소스객체의 변경 유형은 크게 삽입(insertion), 삭제(deletion), 수정(modification)이 있다. 본 알고리즘에서는 변경 유형들 중에서 삽입과 삭제는 독립적인 연산으로 처리하고, 수정은 삭제 후 삽입 과정으로 처리한다.

소스객체의 변경 유형과 카디널리티 비율에 따라 실체화된 뷰 객체의 변경 시맨틱은 [16-18]에서 자세하게 기술되어 있다. 여기서 소스객체의 삽입/삭제에 따른 공간뷰 객체에 대한 변경 유형은 삽입/삭제와 변경 없음(삽입/삭제 없음) 2가지가 있다. 일대다와 다대다인 경우에는 공간뷰 객체를 유도한 소스객체가 여러 개 존재하기 때문에, 소스객체가 삽입/삭제되더라도 공간뷰 객체에 대한 삽입/삭제가 필요하지 않고 관련된 VDR 인스턴스만 변경하면 된다. 다만 뷰 객체와 대응되는 소스객체가 한 개만 존재하는 경우(대응되는 소스객체가 유일한 경우)에는 소스객체의 삽입/삭제에 따라 뷰 객체의 삽입/삭제가 발생한다. 반면에 일대일과 다대일인 경우에는 공간뷰 객체와 관련된 소스객체가 한 개만 존재하기 때문에, 삽입/삭제된 소스객체에 대하여 이 객체로부터 유도된 공간뷰 객체를 삽입/삭제해야 한다. 그리고 관련된 VDR 인스턴스에 대한 삽입/삭제가 추가로 필요하다.

소스객체의 변경에 따라 영향받는 실체화된 공간뷰의 일관성 유지를 위한 점진적 변경 알고리즘은 이러한 공간뷰 객체의 변경 시맨틱을 기반으로 한다. 그리고 이 알고리즘에서는 앞에서 기술한 CDR과 VDR을 이용하여 점진적 변경을 지원한다. 이 알고리즘들에 대한 세부적인 사항은 [16, 18]에 자세하게 기술되어 있다.

5. 성능 평가

5.1 실험 평가 대상

본 논문에서 제시한 점진적 변경 방법은 기본적으로 VDR을 기반으로 처리한다. 따라서 점진적 변경 방법의 성능 평가의 목적은 VDR을 이용하였을 때의 장점을 보이는 것이다. 즉, VDR을 이용하여 실체화된 공간뷰 객체를 변경하였을 경우에 얼마나 빠르게 처리하는가를 검증하는 것이다. 이를 위하여 VDR을 이용한 점진적 변경 방법과 이용하지 않는 방법과 비교를 한다.

VDR을 이용하지 않는 방법은 기본적으로 CDR을 이용한다. 이 방법은 소스객체의 변경에 영향받는 뷰 클래스들은 검색이 가능하지만, VDR을 이용하지 않기 때문에 변경된 소스객체에 직접적으로 영향받는 뷰 객체는 검색할 수가 없다. 따라서 이 방법에서는 소스객체가 변경된 경우에 영향받는 공간뷰 클래스의 객체들을 재수행하여 생성하는 것이 필요하다.

VDR을 이용하는 방법은 공간뷰 객체의 실체화 방법에 따라 구현 방법에 차이가 있을 수 있다. 즉, 값 복사에 의한 실체화 방법과 식별자 유지에 의한 실체화 방법에 따라 달라질 수 있으며, 실제로 이 두 방법에 따라 VDR을 저장하는 방법이 달라진다. 먼저 값 복사에 의한 실체화 방법에서는 VDR을 뷰 객체와 별도로 저장해야 한다. 만약에 뷰 객체내에 VDR을 표현하면, VDR을 검색하기 위하여 모든 공간뷰 객체들을 검색해야 한다. 이 경우에는 뷰 객체내에 애트리뷰트 값이 복사되어 있기 때문에 VDR을 검색하기 위한 디스크 입출력(I/O)시간이 많이 걸린다. 따라서 값 복사에 의한 실체화 방법에서는 뷰 객체와 VDR을 별도로 저장하는 것이 바람직하며, 본 논문에서는 실험 평가를 위하여 이 방법으로 구현하였다.

식별자 유지 방법에서는 기본적으로 뷰 객체내에 애트리뷰트와 관련된 소스객체의 식별자만을 저장한다. 이 방법에서는 뷰 객체내에 있는 소스객체의 식별자 리스트를 확장하여 VDR을 표현할 수 있다. 이 방법에 대한 실험 평가를 위하여 식별자 유지 실체화 방법을 이용하여 실제 시스템에서 구현되어진 MultiView[6, 10, 11]를 비교 대상으로 하였다.

원래 MultiView는 하나의 소스클래스로부터 유도되는 클

래스 뷰를 대상으로 하지만, 본 논문에서는 여러개의 소스 클래스로부터 유도되는 경우에도 적용하도록 확장하였다. 그리고 MultiView 방법은 실제 뷰 객체에 유도된 애트리뷰트들에 관련된 소스객체의 식별자만을 유지한다. 즉, 이 방법을 여러개의 소스객체들로부터 유도된 공간뷰 객체에 적용하면 기하-소스객체의 식별자만을 유지한다. 따라서 이 경우에는 비기하-소스객체의 변경에 따른 뷰 객체의 일관성을 유지하기가 매우 어렵다. 이 문제를 해결하기 위하여 실체화된 뷰 객체내에 기하-소스객체 뿐만아니라 비기하-소스객체의 식별자를 유지하도록 하였다. 결론적으로 본 논문에서 성능 비교를 목적으로 구현된 MultiView 방법은 실체화된 공간뷰 객체내에 뷰 객체와 직접 관련된 소스객체의 식별자와 VDR을 위한 비기하-소스객체의 식별자들을 같이 저장하도록 확장 구현하였다.

본 논문에서 제시한 VDR을 이용한 점진적 변경 방법의 성능 평가를 위한 실험 평가 대상은 앞에서 언급한 것과 같이 3가지이다. <표 1>은 실험 평가 대상을 비교한 것이다. 방법 3은 본 논문에서 제시하여 구현한 방법이고 방법 2는 MultiView와 같이 식별자 유지에 의한 실체화 방법을 이용한 것이다. 그리고 방법 1은 VDR을 이용하지 않는 방법이다.

〈표 1〉 실험 대상 방법의 비교

비교 항목 / 방 법	클래스 유도관련성	뷰 유도관련성	뷰 유도관련성 저장
방법 1 (CDR만 이용)	이용함	이용하지 않음	해당사항 없음
방법 2 (식별자 유지 + VDR 이용)	이용함	이용함	뷰 객체와 독립적으로 저장
방법 3 (값 복사 + VDR 이용)	이용함	이용함	뷰 객체의 객체 식별자 리스트 이용

방법 2와 방법 3은 공통적으로 VDR을 이용하지만 저장 방법에 차이가 있다. 따라서 이 방법들에 대한 성능 평가는 VDR의 검색 시간에서 차이가 날 수 있다. 세부적으로 살펴보면 방법 3은 변경 알고리즘에서는 뷰 객체와 독립적으로 저장된 VDR을 검색한다. 반면에 방법 2는 뷰 객체내에 저장된 객체식별자 리스트를 이용한다. 즉, 변경된 소스객체와 관련된 뷰 객체를 알아내기 위하여, 변경 알고리즘은 실체화된 뷰 객체들의 식별자 리스트를 조사해야 하므로 공간뷰 객체를 직접 검색한다.

또한 이 방법들은 공간뷰의 실체화 방법과 밀접한 관련이 있다. 즉, VDR의 저장 방법이 뷰의 실체화 방법에 따라 결정이 된다. 따라서 이 두 가지 방법에 대해서는 실체화된 공간뷰의 점진적 변경뿐만 아니라 공간뷰에 대한 질의 수행에 대한 성능 평가를 수행해야 한다. 결론적으로 뷰에 대한 질의 수행과 점진적 변경에 따른 실험 평가를 모두 고려하여 어떤 방법이 더 우수한 가를 검증해야 한다. 이를 위하여 본 논문에서는 이 두 가지 실체화 방법에 대하여 추가로 뷰에 대한 질의 수행 시간을 비교하는 성능 평가를 수행한다.

5.2 실험데이타

성능 평가를 위하여 사용한 실험데이터는 실제 데이터로서 4가지 데이터 집합(data set)을 이용한다. 이 데이터 집합은 건물, 구역, 등고선, 버스 정류소 등 다양한 지리객체들로 구성되어 있다. (그림 3)은 실험데이터를 출력한 것이고, <표 2>는 각 실험데이터 집합을 비교한 것이다. 여기서 각 데이터 집합은 동일한 실제 데이터에 대하여 실험 평가를 목적으로 지리객체의 수를 증가시켜 생성하였다. 특히, Building과 Zone 객체 수를 별도로 표기한 이유는 이 객체들이 나중에 실험에 이용될 공간뷰에 대한 소스객체이기 때문이다.

(그림 3) 실험데이터의 출력

〈표 2〉 실험데이터 집합에 대한 비교

(단위 : 개)

객체 종류 / 데이터 집합	Zone 객체	Building 객체	전체 객체
데이터 집합 1	6	616	4988
데이터 집합 2	12	1232	9976
데이터 집합 3	18	1848	14964
데이터 집합 4	24	2464	19952

5.3 점진적 변경 방법에 대한 성능 평가

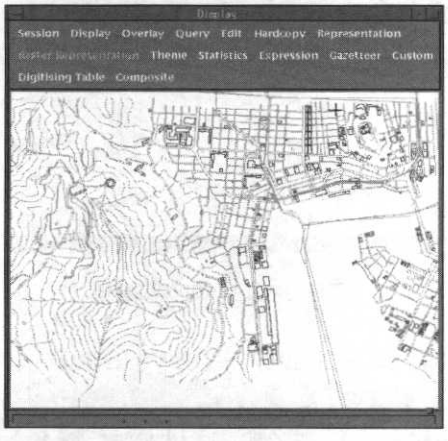
본 논문에서 알고리즘 구현 및 성능 평가를 위하여 공간뷰 시스템의 프로토타입을 객체지향 GIS S/W인 고딕(Gothic)[14] 환경에서 구현하였다. 이 프로토타입에 대해서는 [19, 20]에 자세하게 기술되어 있다. 본 논문에서는 점진적 변경 알고리즘의 성능 평가를 위하여 앞에서 언급한 3가지 방법에 대하여 실험 평가를 수행한다. 특히 이 실험에서는 각 실험 대상 방법을 동일한 환경에서 구현하여 성능 평가를 수행하였다. 실험 환경으로 CPU 속도는 333MHZ이고 메모리는 128M인 디지털사(DEC)의 ALPHA 스테이션을 이용하였다. 그리고 이 실험에서는 일대일과 다대일 관계인 경우에 대하여 수행하고, 일대다와 다대다에 대한 실험은 생략한다. 이유는 일대다와 다대다인 경우에는 소스객체의 삽입/삭제에 따라 뷰 객체의 삽입/삭제없이 VDR 인스턴스의 삽입/삭제만 발생하므로, 일대일과 다대일에 대한 실험

결과에 포함되기 때문이다. 소스객체의 변경 유형은 삽입과 삭제인 경우에 대해서만 수행하였고, 수정은 삭제 후 삽입 과정으로 처리하기 때문에 별도의 실험을 수행하지 않는다.

5.3.1 일대일인 경우

뷰 객체와 소스객체가 일대일인 경우의 점진적 변경 알고리즘의 성능 평가를 위하여 먼저 (그림 4)와 같은 Building_In_Zone 공간뷰를 정의한다. 이 공간뷰는 Building 객체와는 일대일 관계를 가지고, Zone 객체와는 다대일 관계를 가진다.

```
Spatial_View Building_In_Zone (owner, geom) As
Select Building.owner, Building.geom
From Building, Zone
Where contains(Zone, Building)
```



(그림 4) Building_In_Zone 공간뷰의 정의와 출력 결과

일대일인 경우에 점진적 변경 알고리즘의 성능을 평가하기 위하여 Building 객체에 대하여 삽입과 삭제를 하였다. (그림 5)는 성능 평가를 위하여 새로운 Building 객체를 추가한 경우로, 그림에서 "+" 부분의 객체가 새로 삽입되었고 점진적 변경 알고리즘에 의하여 새로운 공간뷰 객체가 생성된다.

(그림 5) 새로운 Building 객체를 삽입한 경우

일대일 관계에 대하여 Building 객체를 변경한 경우에 각 변경 방법에 대한 실험 결과는 (그림 6)과 같으며, 이 실험은 앞에서 기술한 데이터 집합들에 대하여 수행하였다. 각 방법에 대한 성능 평가는 실체화된 공간뷰 객체 수에 따라 변경 시간을 측정하였다. 이 뷰 객체는 실험 대상인 Building_In_Zone 공간뷰를 각 데이터 집합을 대상으로 하여 실체화된 것이고, 실체화된 뷰 객체의 수는 478, 956, 1434, 1912개이다.

실험 결과를 분석하면 전체적으로 VDR을 이용하지 않고 CDR만 이용한 변경 방법이 VDR을 이용한 점진적 변경 방법에 비하여 뷰 객체의 변경 시간이 많이 걸린다. 이유는 CDR만 이용하는 방법은 변경된 소스객체에 직접적으로 영향받는 공간뷰 객체를 찾을 수 없기 때문에 전체적으로 해당 뷰 객체들을 재실체화하기 때문이다.

"식별자 유지 + VDR 이용"과 "값 복사 + VDR 이용" 방법은 기본적으로 VDR을 이용하므로 "CDR만 이용"하는 방법에 비하여 상대적으로 공간뷰 객체의 변경 시간이 매우 빠르다. 그리고 "값 복사 + VDR 이용" 방법이 "식별자 유지 + VDR 이용"에 비하여 뷰 객체의 변경 시간의 차이는 있지만, 거의 유사하다.

"값 복사 + VDR 이용" 방법이 "식별자 유지 + VDR 이용" 방법보다 변경 시간이 더 걸리는 이유는 크게 두 가지이다. 먼저 "값 복사 + VDR 이용" 방법은 VDR이 뷰 객체와 별도로 저장되고 유지되므로 뷰 객체내의 객체식별자 리스트에 VDR을 저장하는 "식별자 유지 + VDR 이용" 방법에 비하여 VDR의 검색 시간이 많이 걸린다. 또 다른 이유는 "값 복사 + VDR 이용" 방법은 변경 과정에서 새로 뷰 객체를 생성할 때 애트리뷰트 값을 복사하는 시간이 추가되기 때문이다. 실제적으로 이 두 방법간의 변경 시간은 값 복사에 의한 시간 차이로 인하여 발생한다. 여기에 관련된 실험은 뒤에서 자세히 설명한다.

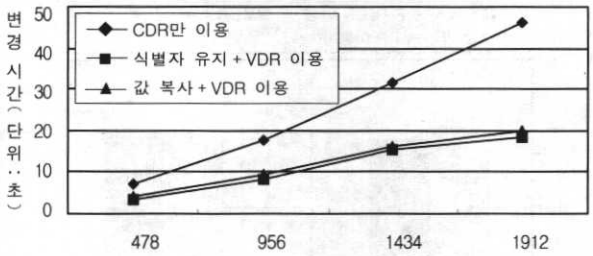
(그림 6)의 실험 결과에서 소스객체의 변경 유형이 삽입인 경우와 삭제인 경우를 비교하면 VDR을 이용하는 두가지 방법 모두가 상대적으로 시간 차이가 난다. 이 이유는 삭제인 경우에는 VDR을 이용하여 소스객체의 삭제로 인하여 영향받는 뷰 객체를 삭제하면 되지만, 삽입인 경우는 삽입된 소스객체를 대상으로 질의 처리기에서 뷰-정의 질의를 수행한 후에 뷰 객체와 VDR 인스턴스를 생성하기 때문이다. 따라서 삭제인 경우보다 삽입인 경우에 뷰 객체의 변경 시간이 많이 걸린다.

5.3.2 다대일인 경우

다대일인 경우에 점진적 변경 알고리즘의 성능을 평가하기 위하여 Zone 객체에 대하여 삽입과 삭제를 하였다. (그림 7)은 성능 평가를 위하여 "+" 부분에 있었던 기존의 Zone 객체를 삭제한 경우이다. (그림 5)와 (그림 7)을 비교하면 삭제된 Zone내에 포함되어 있는 공간뷰 객체들이 삭제된 것을 알 수 있다.

(단위 : 초)

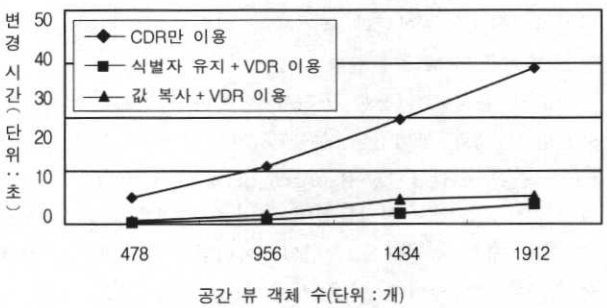
변경 방법 \ 뷰 객체 수	478	956	1434	1912
CDR만 이용	6.98536	17.672064	31.484096	46.272304
식별자 유지 + VDR 이용	3.549488	8.142496	15.549088	18.310944
값 복사 + VDR 이용	4.271328	9.354288	15.979504	20.0244



(가) Building 객체를 삽입한 경우

(단위 : 초)

변경 방법 \ 뷰 객체 수	478	956	1434	1912
CDR만 이용	9.685152	21.681248	39.293376	58.250756
식별자 유지 + VDR 이용	0.8048	1.890688	3.981456	7.713456
값 복사 + VDR 이용	1.149328	3.310944	9.20984	10.513376



(나) Building 객체를 삭제한 경우

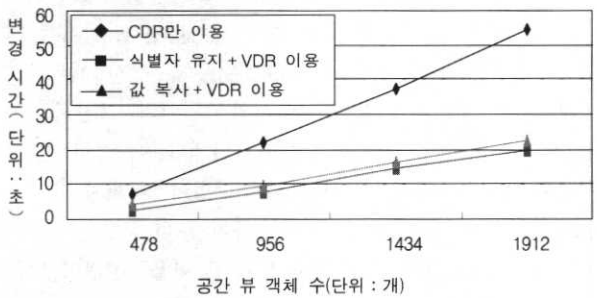
(그림 6) 일대일 관계에서의 점진적 변경 알고리즘의 성능 평가

(그림 7) 기존의 Zone 객체를 삭제한 경우

다대일 관계에 대하여 Zone 객체를 삽입, 삭제한 경우에 각 변경 방법에 대한 실험 결과는 (그림 8)과 같다. 이 실험은 일대일 경우와 마찬가지로 각 데이터 집합들에 대하여 수행하였으며, 실체화된 공간뷰 객체의 수에 따라 변경 시간을 측정하였다. 실험 결과를 분석하면 (그림 6)과 같은 일대일인 경우와 유사한 실험 평가를 얻을 수 있다. 즉, 다대일인 경우에도 CDR만 이용한 변경 방법이 VDR을 이용한 점진적 변경 방법에 비하여 뷰 객체의 변경 시간이 많이 걸린다. 또한 "식별자 유지 + VDR 이용"과 "값 복사 + VDR 이용" 방법은 "CDR만 이용" 방법에 비하여 상대적으로 공간뷰 객체의 변경 시간이 빠르다. 그리고 "값 복사 + VDR 이용" 방법이 "식별자 유지 + VDR 이용"에 비하여 뷰 객체의 변경 시간의 차이는 있지만, 거의 유사하다.

(단위 : 초)

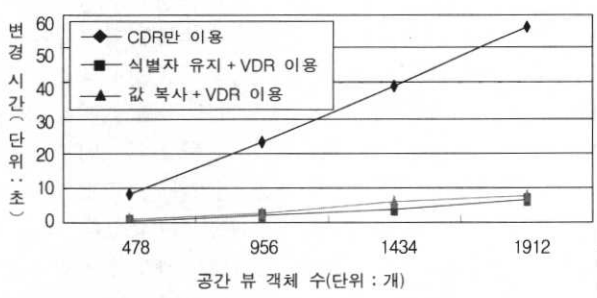
변경 방법 \ 뷰 객체 수	478	956	1434	1912
CDR만 이용	7.552872	21.842532	37.076128	54.608624
식별자 유지 + VDR 이용	2.623264	7.979544	14.749168	19.715984
값 복사 + VDR 이용	4.259616	9.420256	16.318752	22.884832



(가) Zone 객체를 삽입한 경우

(단위 : 초)

변경 방법 \ 뷰 객체 수	478	956	1434	1912
CDR만 이용	8.295728	22.919968	38.843264	56.101504
식별자 유지 + VDR 이용	0.787232	2.389024	3.62424	6.43392
값 복사 + VDR 이용	1.159088	3.008784	6.264496	7.876048



(나) Zone 객체를 삭제한 경우

(그림 8) 다대일 관계에서의 점진적 변경 알고리즘의 성능 평가

다대일인 경우는 일대일 경우와 비교하면 "값 복사 + VDR 이용" 방법이 "식별자 유지 + VDR 이용"에 비하여 변경 시간의 차이가 커질 수 있다. 이것은 다대일인 경우에 하나의

소스객체에 대하여 대응되는 뷰 객체의 수가 여러 개 이므로, 변경 과정에서 새로 생성되는 뷰 객체의 수가 많아진다. 따라서 생성되는 뷰 객체의 수가 많아지게 되면 값 복사에 의한 시간이 많이 걸리므로 두 방법간의 뷰 객체의 변경 시간이 많이 차이가 날 수 있다.

5.3.3 값 복사 시간을 고려한 점진적 변경 방법에 대한 실험

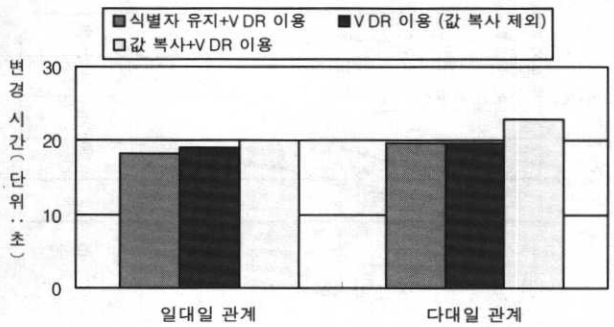
"값 복사 + VDR 이용" 방법과 "식별자 유지 + VDR 이용"에서 뷰 객체의 생성시 값 복사에 의한 변경 시간 차이를 알아보기 위하여, 앞의 실험 결과에서 시간 차이가 많이 나는 데이타 집합 4(뷰 객체수 1912개)에 대하여 다시 실험을 하였다. 이 실험에서는 "값 복사 + VDR 이용" 방법에서 값 복사 부분을 제외한 "VDR 이용(값 복사 제외)"를 추가로 수행하였다. 그리고 이 실험은 일대일과 일대다 관계를 대상으로 소스객체가 삽입된 경우에 실험을 하였다. 소스객체의 삽입에 대해서 실험을 수행하는 이유는 새로운 뷰 객체의 생성이 소스객체가 삽입된 경우에 가능하기 때문이다.

(그림 9)는 이 실험에 대한 결과를 나타낸다. 여기서 "값 복사 + VDR 이용" 방법과 "VDR 이용(값 복사 제외)" 방법의 시간 차이는 변경 과정에서의 뷰 객체 생성시 속성 값 복사에 의하여 걸리는 시간을 나타낸다. 이 실험 결과를 통하여 "VDR 이용(값 복사 제외)" 방법이 "식별자 유지 + VDR 이용"과 거의 변경 시간의 차이가 나지 않는다는 것이다. 따라서 실제로 "값 복사 + VDR 이용" 방법이 "식별자 유지 + VDR 이용"에 비하여 값 복사에 의한 시간을 제외한다면 변경 시간이 거의 동일함을 알 수 있다.

그러나 여전히 "VDR 이용(값 복사 제외)" 방법과 "식별자 유지 + VDR 이용" 방법간의 약간의 변경 시간이 차이가 난다. 이것은 변경 과정시 VDR 검색 시간에 의해 발생하는 시간이다. "식별자 유지 + VDR 이용" 방법에서는 객체식별자를 유지하는 뷰 객체를 VDR과 동일하게 이용한다. 즉, 변경된 소스객체로부터 유도된 공간뷰 객체를 찾기 위하여 공간뷰 객체내의 객체식별자 리스트를 조사한다. 그리고 실체화된 뷰 객체내에 소스객체의 식별자 리스트 정보만을 가지므로 VDR과 비교하더라도 실제 데이타 양은 큰 차이가 나지 않는다. 반면에 본 논문에서는 VDR을 저장하기 위하여 3개의 클래스들을 정의하고, VDR 검색을 위하여 이 클래스들간의 참조 관계를 이용하였다. 따라서 VDR을 검색하기 많은 객체를 검색하기 때문에 시간이 많이 걸린다. 이 부분은 VDR의 저장을 위한 데이터 구조를 효율적으로 설계하면 뷰 객체의 변경 시간을 줄일 수 있다. 그러나 기본적으로 본 논문에서와 같이 값 복사에 의한 실체화 방법을 이용하는 경우에는 VDR을 별도로 저장해야 하므로, 뷰 객체에 저장된 객체식별자를 이용하는 방법에 비하여 검색 시간이 더 걸리게 된다.

(단위 : 초)

변경 방법 \ 대응 관계	일대일	다대일
식별자 유지 + VDR 이용	18.310944	19.715984
VDR 이용(값 복사 제외)	19.081008	19.7174528
값 복사 + VDR 이용	20.0244	22.88483



(그림 9) 값 복사 시간을 고려한 점진적 변경 방법의 성능 평가

5.4 뷰 실체화 방법에 대한 질의 수행 시간 비교

앞의 실험을 통하여 "VDR 이용(값 복사 제외)" 방법과 "식별자 유지 + VDR 이용" 방법간의 성능 평가를 알 수 있다. 그러나 이 두 가지 방법에서는 VDR을 저장하는 방법이 공간뷰의 실체화 방법에 따라 결정되므로, 공간뷰의 점진적 변경뿐만 아니라 공간뷰에 대한 질의 수행에 대한 성능 평가를 수행해야 한다.

식별자 유지를 이용한 공간뷰 실체화 방법과 값 복사를 이용한 실체화 방법간의 성능 평가를 위하여 먼저 (그림 10)과 같은 공간뷰 OverlapArea_Between_Building을 정의한다. 이 공간뷰는 빌딩들간에 겹치는 영역을 정의한 것으로 기하-생성 공간뷰에 속한다. 이러한 기하-생성 공간뷰를 뷰 실체화 방법에 대한 질의 수행의 대상으로 정한 이유는 새로운 기하데이타를 생성하는 경우에 두 가지 실체화 방법에 대한 성능을 분명하게 보일 수 있기 때문이다.

```
Spatial_View OverlapArea_Between_Building (geom) AS
Select overlap(Building1, Building2)
From Building1, Building2
Where overlaps(Building1, Building2)
```

(그림 10) 공간뷰 OverlapArea_Between_Building의 정의

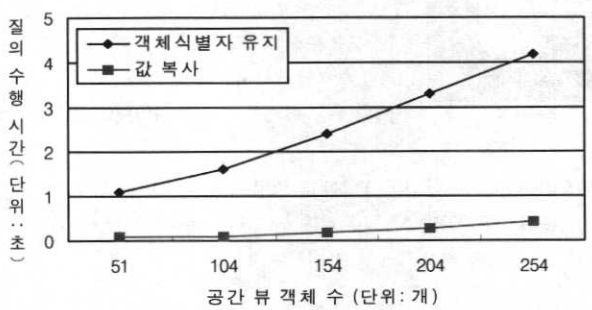
두 가지 실체화 방법에 대한 질의 수행 시간을 비교하기 위하여 앞에서 정의한 OverlapArea_Between_Building 공간뷰에 대한 단순한 검색 질의를 이용한다. (그림 11)은 OverlapArea_Between_Building 공간뷰에 대한 검색 질의 결과를 보여준다. 이 그림에서는 빌딩들간의 겹치는 영역을 분명하게 보이기 위하여 빌딩 객체들을 같이 출력하였다.

그리고 실험은 공간뷰 객체의 수를 증가시켜 검색 질의의 수행 시간을 비교하였다. 여기서 공간뷰 객체의 수는 51, 104, 154, 204, 254개로 하였다. 이 실험에 대한 결과는 (그림 12)와 같다.

(그림 11) 공간뷰 OverlapArea_Between_Building의 검색

(단위 : 초)

실체화 방법 \ 뷰 객체 수	51	104	154	204	254
식별자 유지	1.10736	1.625216	2.408544	3.290848	4.191296
값 복사	0.079056	0.101504	0.193824	0.280112	0.43432



(그림 12) 공간뷰에 대한 질의 수행 시간 비교

(그림 12)의 실험 결과를 통하여 기본적으로 값 복사 실체화 방법이 식별자 유지 실체화 방법보다 질의 수행 시간이 빠름을 알 수 있다. 또한 뷰 객체수의 증가에 따라 두 방법 간의 질의 수행 시간 차이가 많이 난다. 이 이유는 값 복사 방법은 뷰 객체의 기하데이타를 가지고 있기 때문에 바로 검색이 가능하다. 그러나 식별자 유지 방법은 뷰 객체내에 기하데이타를 가지고 있지 않기 때문에 질의 수행 시간이 상대적으로 느리다. 즉, 식별자 유지 방법에서는 먼저 뷰 객체에 저장된 소스객체의 식별자를 이용하여 소스객체의 기하데이타를 검색한 후에 이 기하데이타들에 대하여 공간 연산을 수행한다. 따라서 식별자 유지 방법은 값 복사 방법에 비하여 뷰 객체의 기하데이타를 공간 연산을 통하여 구하는 시간이 더 들기 때문에 질의 수행 시간이 느리다. 또한 실험 결과에서 뷰 객체수의 증가에 따라 시간의 차이가 많이 나는 이유는 뷰 객체들에 대한 기하데이타를 구하는 시간이 많이 걸리기 때문이다. 두 가지 실체화 방법에 따른 공간뷰의 질의 수행에 대한 성능 평가는 [19]에서 자세하게 기술되어 있다.

5.5 종합 검토

앞의 실험 평가를 고려해 볼 때, 소스객체의 변경에 따른 실체화된 공간뷰 객체의 변경에서 VDR을 이용한 점진적 변경이 VDR을 이용하지 않는 방법에 비하여 매우 효과적인 것을 알 수 있다. 그리고 "식별자 유지 + VDR 이용"이 "값 복사 + VDR 이용" 방법에 비하여 변경 시간이 다소 빠르다는 것을 알 수 있다. 또한 저장 공간 측면을 볼 때에도 객체식별자 리스트만을 유지하는 "식별자 유지 + VDR 이용" 방법이 좋지만, "값 복사 + VDR 이용" 방법은 속성 값 복사에 따른 저장 공간과 VDR에 따른 저장 공간이 필요하므로 오버헤드가 있다.

그러나 뷰에 대한 질의 수행시에는 값 복사에 의한 실체화 방법이 식별자 유지에 의한 실체화 방법보다 질의 수행 시간이 많이 차이가 난다. 특히, 기하데이타가 변형되는 기하-생성 공간뷰인 경우에는 식별자 유지에 의한 실체화 방법에서는 공간뷰 객체의 기하데이타를 공간연산에 의해 구해야 하므로 많은 시간이 걸린다. 이 실험 결과를 고려할 때, 복잡한 공간 연산에 의해 유도되는 공간뷰에 대하여 빠른 질의 수행 시간을 제공하기 위해서는 값 복사에 의한 실체화 방법이 적합하다.

실체화 방법과 점진적 변경 방법에 따른 성능 평가를 고려해 볼 때, 전체적으로 식별자 유지 실체화 방법보다 값 복사 실체화 방법을 기반으로 VDR을 이용한 점진적 변경 방법이 효율적이다. 특히, "값 복사 + VDR 이용" 방법에서 VDR을 위한 효율적인 새로운 데이터 구조를 이용한다면 뷰 객체의 변경 시간을 많이 줄일 수 있다.

6. 결 론

본 논문에서는 뷰 유도관련성을 이용한 점진적 변경 알고리즘에 대하여 성능 평가를 수행하였다. 실험 평가를 통하여 먼저 뷰 유도관련성을 이용한 점진적 변경 방법이 이 관련성을 이용하지 않는 점진적 변경 방법에 비하여 매우 효과적인 것을 알 수 있었다. 또한 뷰 유도관련성을 이용한 점진적 변경을 위한 두 가지 방법인 "식별자 유지 + VDR 이용" 방법과 "값 복사 + VDR 이용" 방법을 실험을 통하여 비교할 때, 공간뷰 객체에 대한 변경 시간은 거의 차이가 없었다. 그리고 공간뷰에 대한 질의 수행에 있어서는 값 복사 실체화가 식별자 유지 실체화보다 질의 수행 시간이 훨씬 빠름을 알 수 있었다. 결론적으로 뷰 객체의 변경 시간과 뷰에 대한 질의 수행 시간을 종합적으로 비교한 결과, 식별자 유지 실체화 방법보다 값 복사 실체화 방법을 기반으로 뷰 유도관련성을 이용한 점진적 변경 방법이 효율적이며 바람직함을 알 수 있었다. 앞으로 향후 연구로서 뷰 유도관련성의 검색 속도를 향상시킬 수 있는 새로운 데이터 구조를 제시하는 것이 필요하다.

참 고 문 헌

[1] Ashish Gupta, Inderpal Singh Mumick, and V. S. Sub-

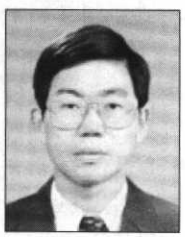
rahmanian, "Maintaining Views Incrementally," Proc. of Int'l Conf. on Management of Data, pp.157-166, May, 1993.
[2] Ashish Gupta and Inderpal Singh Mumick, "Maintenance of Materialized Views : Problems, Techniques, and Applications," Proc. of Int'l Conf. on Data Engineering, pp.3-18, 1995.
[3] Claramunt C. and Mainguenaud M., "Identification of Definition Formalism for a Spatial View," Advanced Geographical Modeling, 1994.
[4] Claramunt C. and Mainguenaud M., "Dynamic and Flexible Vision of a Spatial Database," Proc. of Int'l Conf. and Workshop on Database and Expert Systems Applications, pp. 483-493, 1995.
[5] C. Souza dos Santos, "Design and Implementation of Object-Oriented Views," Proc. of Int'l Conf. and Workshop on Database and Expert Systems Applications, pp.91-102, 1995.
[6] Elke A. Rundensteiner, "MultiView : A Methodology for Supporting Multiple Views in Object-Oriented Databases," Proc. of the Int'l Conf. on VLDB, pp.187-198, 1996.
[7] Eric N. Hanson, "A Performance Analysis of View Materialization Strategies," Proc. of Int'l Conf. on Management of Data, pp.440-453, 1987.
[8] Giovanna Guerrini, Elisa Bertino, Barbara Catania, and Jesus Garcia-Molina, "A Formal Model of Views for Object-Oriented Database Systems," Technical Report, DISI-96-2, 1996.
[9] G. Wiederhold, "Views, Objects, and Databases," IEEE Computer, pp.37-44, 1986.
[10] Harumi. A. Kuno and Elke A. Rundensteiner, "Materialized Object-Oriented Views in MultiView," Int'l Workshop on Research Issues on Data Engineering : Distributed Object Management, 1995.
[11] 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.
[12] Jose A. Blakeley, Per-Ake Larson and Fank Wm Tompa, "Efficiently Updating Materialized Views," Proc. of Int'l Conf. on Management of Data, pp.61-71, 1986.
[13] 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.
[14] Laser-Scan, "Writing and Developing Applications using GOTHIC ADE," Issue 2.0, 1995.
[15] Marc H. Scholl and H. J. Schek, "Supporting Views in Object-Oriented Databases," IEEE Database Engineering Bulletin, Vol.14, No.2, pp.43-47, 1991.
[16] Sang-Ho Moon and Bong-Hee Hong, "Incremental Update Algorithms for Materialized Spatial Views by Using View Derivation Relationships," Proc. of Int'l Conf. and Workshop on Database and Expert Systems Applications, pp.539-550, 1997.
[17] Sang-Ho Moon and Bong-Hee Hong, "Design and Implementation of Object-Oriented Spatial Views," Proc. of Int'l Conf. on Object Oriented Information Systems, pp.386-396, 1997.
[18] 문상호, 김동현, 홍봉희, "실체화된 공간뷰의 점진적 변경", 한국정보과학회논문지, Vol.25, No.1, pp.37-50, 1998.
[19] 문상호, 김동우, 반재훈, 홍봉희, "객체지향 공간뷰의 설계 및 구현", 한국정보과학회논문지, Vol.26, No.2, pp.306-320, 1999.

문 상 호

e-mail : shmoon@mail.uiduk.ac.kr
1991년 부산대학교 컴퓨터공학과 졸업 (공학사)
1994년 부산대학교 대학원 컴퓨터공학과 (공학석사)
1998년 위덕대학교 컴퓨터멀티미디어공학부 교수
2002년~현재 부산외국어대학교 컴퓨터전자공학부 교수
관심분야 : 공간DB, Mobile GIS, 공간뷰, 데이터마이닝, GIS 표준, 정보시스템 감리 등

반 재 훈

e-mail : chban@kit.ac.kr
1997년 부산대학교 컴퓨터공학과 졸업 (공학사)
1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)
2001년 부산대학교 대학원 컴퓨터공학과 박사과정 수료
2002년~현재 경남정보대학 인터넷상거래과 전임강사
관심분야 : 이동객체, 지리정보시스템(GIS), 공간뷰 등



홍 봉 희

e-mail : bhhong@hyowon.cc.pusan.ac.kr
1982년 서울대학교 컴퓨터공학과 졸업 (학사)
1984년 서울대학교 대학원 컴퓨터공학과 졸업(석사)
1988년 서울대학교 대학원 컴퓨터공학과 졸업(박사)
1987년~현재 부산대학교 컴퓨터공학과 교수
관심분야 : 공학DB, 객체지향 DB, GIS, 분산공간 DB, 개방형 GIS