한국지역지리학회지 제3권 제2호(1997) 105~120

객체지향 지리정보 데이터베이스를 위한 색인기법*

夫 起 東**

객체지향 지리정보 데이타베이스 시스템의 설계시 중요한 고려 사항은 저장된 데이타에 대한 좋은 접근 전략을 갖도록 하는 것이다. 객체지향 시스템에서는 이러한 목적으로 여러가지 색인 기법이 개발되었으나, 이러한 기법들은 객체지향 데이타 모델의 집단화 계층이나 상속 계층 중 어느 한 가지만을 고려하는 경우가 대부분이었다. 본 연구에서는 포인터 체인 디렉토리를 이용하여 객체지향 지리 데이터베이스의 집단화 및 상속 계층을 접근하는 데 효율적인 색인 기법을 제안하였다. 제안된 기법의 효용성을 기존의 색인 기법들과 다양하게 비교하였으며, 저장비용과 검색비용 측면에서 그 성능을 시뮬레이션한 결과를 제시하였다.

主要語 : 지리정보시스템(GIS), 데이터베이스, 색인기법, 포인터 체인

1. 서론

지리정보시스템(GIS)은 지리 형태에 관한 자료를 수집, 저장하고 이를 분석 및 가공하여 각종 지리 관련 응용분야에 활용할 수 있는 정보를 제공하는 컴퓨터 소프트웨어 시스템이라 할 수 있다. GIS에서는 대용량의 다양한 지리적인 자료들을 효율적으로 저장 및 검색 할 수 있는 데이터베이스 관리 시스템이 필요하다.

이러한 지리정보 데이터베이스 관리 시스템에서는 공간 데이터들은 여러 개의 점들이 모여서 선을 형성하고, 여러 개의 선이 모여서 영역을 형성하는 등 복합 데이터의 성격을 가지는 경우가 많기 때문에 데이터간의 의미적 연관성(semantic relationship)을 유지하면서 복합 데이터를 효율적으로 처리할 수 있는 객체 관리 기법이 필요하다. 또한, 벡터와 래스터 이미지 같은 공간 데이터와 비공간 데이터로서 속성자료같은 서로 다른 다양한 데이터 타입을 지원할 수 있어야 한다. 특히 공간 데이터는 지도 공간상의 가변 길이 도형 데이터로서 이차기억장치에서 데이터의 군집(bulk)으로 나타나게 되는데, 길이가 긴 가변길이의 데이터 군집을 효율적으로 처리할 수 있는 저장 관리 기법이 필요하다.

최근의 연구에서는 이러한 기능을 보다 원활하게 수행하기 위해 객체지향 데이터베이스 시스템을 GIS의 기반 구조로 사용하고자 하는 시도가 활발히 이루어지고 있다(최웅세 외, 1995; 황규영 외, 1995; Oosterom, 1989; Worboy, 1994). 일반적으로 GIS에서 사용되는 공간 객체들은 객체지향 개념의 중요한 요소인 상속성(inheritance)과 복합객체를 표현할 수 있는 집단화(aggregation)에 의해 쉽게 모델링될 수 있으며, 길이가 긴 가변길이의 데이터 저장을 허용함으로써 공간 및 속성 데이터를 하나의 데이터베이스에 저장 및 관리할 수 있다는 장점이 있다.

데이터베이스 시스템에서는 데이터에 대한 접근 시에 최적의 접근 경로를 찾아 효율적인 입출력을 수행함으로써 성능을 향상시키기 위한 색인 기법이 시스템의 성능 향상을 위해 중요하게 다루어지

* 이 연구는 1997년도 경일대학교 교내학술 연구비 지원에 의하여 연구되었음.
** 경일대학교 컴퓨터공학과 부교수

고 있으며, GIS는 지리정보의 특성을 갖는 대량의 데이터를 취급하기 때문에 검색 및 갱신 속도를 향상시킬 수 있는 색인 기법의 비중이 그 만큼 더 크다고 할 수 있다. 특히 GIS의 기반으로서 객체지향 시스템을 적용하였을 경우에는 객체지향 개념의 중요 특성인 상속관계와 집단화 관계를 통해 데이터에 접근하여야 하기 때문에 기존의 B^+-트리나 공간 색인 기법만으로는 적용이 어렵다. 따라서 본 연구에서는 객체지향 데이터베이스에서 지금까지 연구된 색인구조를 토대로 하여 지리 데이터베이스에 효율적으로 적용할 수 있는 색인구조를 제시하고 그 성능을 평가하는 데 주안점을 두었다.

그리고 지금까지 개발된 객체지향 데이터베이스의 색인 기법들은 객체지향 데이터 모델의 집단화 계층(aggregation hierarchy)이나 상속 계층(inheritance hierarchy) 중 어느 한 가지만을 접근하는 질의에 대해서 고려하는 경우가 대부분이었다. 최근의 연구에서는 이 두 계층을 동시에 접근하는 질의를 효율적으로 평가하기 위한 색인 구조를 설계하는 데 초점을 두고 있으며, 이 분야의 대표적 기법으로는 Bertino(1995) 등이 제안한 중첩 상속 색인(nested-inherited index)이 있다.

본 연구에서는 이 중첩 상속 색인을 객체지향 지리정보 데이터베이스에 적용하는 문제를 검토한 후 문제점을 개선한 새로운 색인 기법을 제안하였다. 제안된 방법의 성능은 기존의 색인 기법 중 집단화 계층 및 상속 계층을 동시에 접근하는 질의에 적용이 가능한 중첩 상속 색인을 비교 대상으로 하여 시뮬레이션을 통해 평가하였다.

2. 상속 계층의 모델링

객체지향 데이터베이스의 클래스/서브 클래스간의 계층 구조는 상속 관계를 나타내며 이러한 상속 관계는 사이클이 없는 방향 그래프(directed acyclic graph) 형태의 상속 계층을 구성하게 된다. 이때, 임의의 클래스에서 기술된 속성들과 메소드는 그 클래스의 서브 클래스로 상속되며, 서브 클래스에서는 상속받은 특징들 외에 자신만의 고유한 특성을 추가적으로 유지할 수 있다. 객체지향 지리정보 데이터베이스에서는 관점에 따라 다양한 상속 계층의 모델링이 가능한데, 여기에서는 여러 가지 모델 중에서 대표적인 세가지 계층 구조의 스키마에 대해 서술하고자 한다.

1) 공간 객체 기반 계층 모델

GIS에서 공간 데이터의 가장 기본이 되는 객체는 점, 선, 영역이며 이 세가지 기본 객체의 위상관계 및 기하 관계를 〈그림 1〉과 같은 계층 스키마로서 표현할 수 있다. 그림에서와 같이 상속 계층의 루트로서 모든 클래스의 공통적인 지리 정보

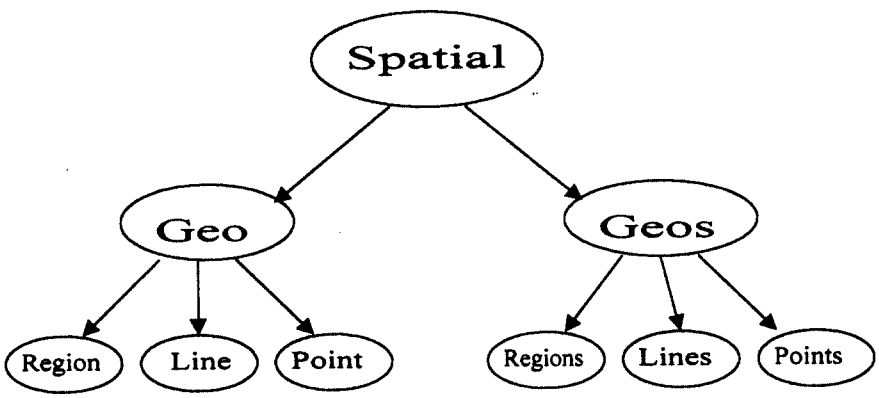

그림 1. 공간 객체 계층의 스키마 예

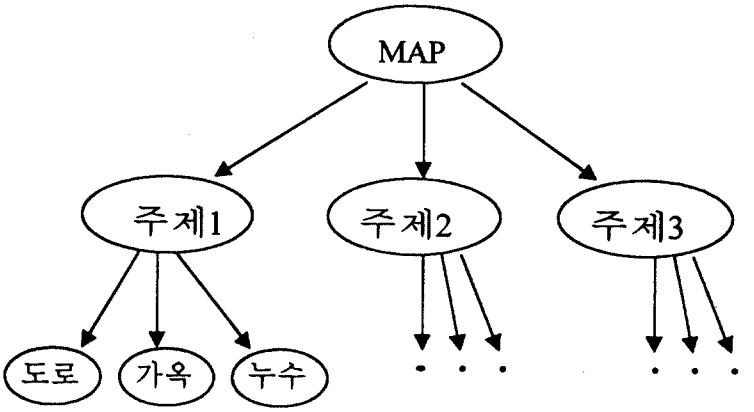

그림 2. 레이어 기반 물리적 계층의 스키마 예

를 표현하는 클래스가 "Spatial" 클래스이다. "Geo" 클래스는 점, 선, 영역 각각의 공통된 성질을 대표하는 상위 클래스로서 "Geo" 내의 메소드가 그 서브클래스(점, 선, 영역)에서 재정의된다. "Geos"는 "Geo" 클래스의 점, 선, 면의 집합이 갖는 성질을 표현하는 클래스로서, 여러 개의 선이 연결되어 하나의 큰 선을 구성할 때 여러 개의 점, 선들이 공통적인 성질을 갖게 되어 그룹화 되는 경우를 그 예로 들 수 있다.

2) 레이어 기반 물리적 계층 모델

기존의 상용 GIS인 ARC/INFO나 MAP/INFO 등에서는 객체지향 개념을 적용하고 있지 않지만 이들 시스템에서 사용하는 레이어 개념을 기반으로 객체지향 시스템의 계층 구조를 모델링한 예를 들면 〈그림 2〉와 같다. 〈그림 2〉의 스키마에서 보는 것처럼 MAP 클래스는 여러 개의 주제도 레이어를 서브 클래스로 갖고 각각의 주제도 레이어는 다시 도로, 가옥, 누수 등의 서브 클래스를 갖는다. 이렇게 기존의 상용 GIS에서 사용하는 물리적 레이어 구조에 객체지향 개념을 적용하면 효율적인 상속 계층을 구현할 수 있다.

3) 응용 기반 논리적 계층 모델

GIS에서는 사용자의 응용 분야에 따라 다양한 논리적 계층 구조가 있다. GIS의 대부분의 응용이 객체지향 개념의 계층 구조로 모델링이 가능하다고 할 수 있다. 〈그림 3〉은 주요도로를 중심으로 지번이 부여된 주거 영역에 대한 속성 자료의 예를 계층 구조로 나타낸 것이다. 물론 이 속성들 중에는 지도상의 위치를 나타내는 공간 데이터도 포함될 수 있다. 〈그림 3〉에서 보는 것처럼 "주택"이나 "상가"같은 하위 클래스들은 슈퍼 클래스인 "주거지역"으로부터 공통적인 속성을 상속받고 자기만의 추가적인 특징들을 기술할 수 있다.

3. 집단화 계층의 모델링

객체지향 개념에서는 전술한 상속 계층 외에도 객체의 한 속성에 대한 도메인이 복합객체일 경우 이 도메인의 값을 별도의 클래스에 관리하여 집단화 계층을 구성할 수 있다. 〈그림 3〉에서는 "주거지역" 클래스의 "소유인" 속성이 복합 도메인으로서 집합 값을 갖기 때문에 이 "소유인" 속성과 집단화 관계로 연결된 "소유주명세" 클래스를 별도로 분리

하면 〈그림 4〉와 같다. 이때, 집단화 관계는 클래스 "주거지역"을 루트로 하는 방향 사이클 그래프(directed cyclic graph) 형태의 집단화 계층을 구성하게 된다.

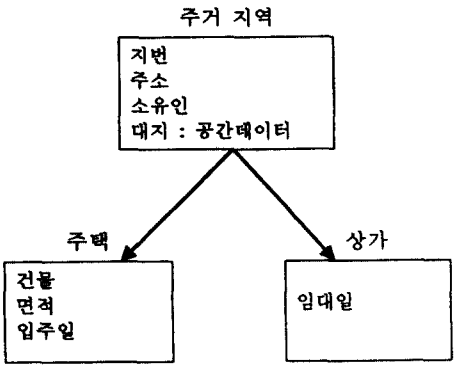

그림 3. 응용기반 논리적 계층의 스키마 예

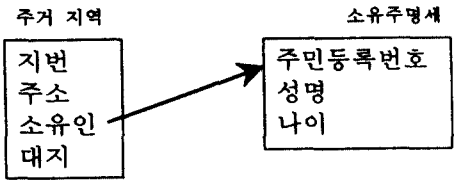

그림 4. 집단화 계층의 스키마 예

객체지향 데이터 모델에서는 복합객체를 효율적으로 구현 할 수 있는 집단화 계층과 상속 관계성을 갖는 상속 계층을 합성하여 데이터베이스 스키마를 구성할 수 있다. 〈그림 5〉는 앞에서 예를 든 상속 계층과 집단화 계층을 합성한 스키마로서 앞으로 본 논문에서 적용할 색인 기법의 예제 스키마로 사용하고자 한다. 본 논문에서 제안하는 색인 기법은 이와 같이 응용 기반 논리적 계층 모델의 상속 계층과 집단화 계층을 합성한 스키마에 대해 적용했을 경우에 그 효용성이 높다.

〈그림 5〉와 같은 합성 계층의 스키마에 사용되는 질의 형식은 크게 세 가지 형식의 범주로 구분할 수 있으며, 그림에서 제시한 스키마를 대상으로 세 가지 형태의 검색 질의 예를 들면 다음과 같다.

① 상속 계층만을 접근하는 질의 :

"대명동에 위치한 주거지역 클래스와 그 서브클래스의 모든 인스턴스를 검색하라."

② 집단화 계층만을 접근하는 질의 :

"소유주의 직업이 공무원인 주거지역 클래스의 모든 인스턴스를 검색하라"

③ 집단화 계층과 상속 계층을 모두 접근하는 질의 :

"소유주의 직업이 공무원인 주거지역과 그 서브클래스의 모든 인스턴스를 검색하라."

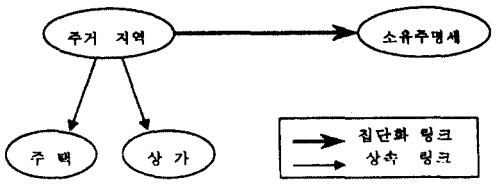

그림 5. 집단화 계층과 상속 계층의 합성 스키마의 예

4. 저장 모델링

본 연구의 색인기법을 적용할 저장구조는 Lu (1991) 등이 제안한 지리정보 데이터베이스의 저장구조를 확장한 모델이다. Lu 등은 객체지향 멀티미디어 데이터베이스인 EXODUS(Carey, 1986)의 저장 구조를 확장하여 공간 객체를 저장할 수 있는 새로운 저장구조를 제안하였으며 데이터베이스 관리 시스템의 저장관리자(Stored Manager)가 객체지향의 중심 개념인 상속 계층을 지원할 수 있도록 설계하였다. 본 논문에서는 Lu 등이 제안한 지리 데이터베이스의 저장구조를 상속 계층과 집단화 계층을 합성한 스키마에 대해서도 적용할 수 있도록 확장하였다. 〈그림 5〉에서 제시한 스키마에 대해 객체 식별자인 OID(Object Identifier)를 부가한 인스턴스들의 예를 나타내면 〈그림 6〉과 같다.

여기에서 "주거지역" 클래스의 "대지" 속성의 값은 길이가 긴 가변길이 포맷을 갖는 공간 데이터에 대한 포인터로서 이 포인터가 가르키는 가변길이 튜플 중 어느 하나를 나타내면 〈그림 7〉과 같다. 〈그림 7〉에서 보는 것처럼 "대지" 속성의 가변

주거지역				
OID	지번	주소	소유인	대지
주거지역〈o〉	100	대구시 22-1	소유주〈I〉	↑p1
주거지역〈p〉	110	대구시 33-10	소유주〈j〉	↑p2
주거지역〈q〉	115	대구시 42-15	소유주〈l〉	↑p3

주 택						
OID	지번	주소	소유인	대지	건물면적	입주일
주택〈i〉	108	대구시 28-8	소유주〈l〉	↑p4	65	97-05-11
주택〈j〉	109	대구시 29-9	소유주〈k〉	↑p5	50	92-03-04

상 가					
OID	지번	주소	소유인	대지	임대일
상가〈i〉	114	대구시 38-1	소유주〈l〉	↑p6	97-08-09

소유주명세			
OID	주민등록번호	성명	직업
소유주〈i〉	571112-1690713	손인태	회사원
소유주〈j〉	290223-2745556	정승희	교사
소유주〈k〉	640405-1776253	김윤식	교사
소유주〈 l〉	551012-1677322	곽동호	공무원

그림 6. 〈그림 5〉의 스키마에 대한 객체 인스턴스들의 예

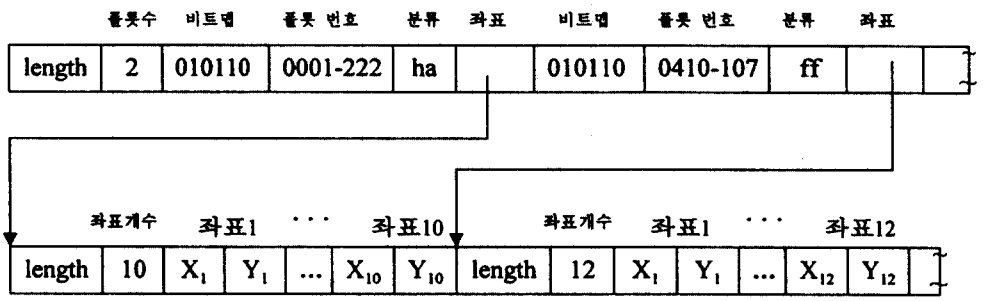

그림 7. 길이가 긴 가변길이 튜플의 저장 포맷

길리 튜플은 길이, 플롯 수, 비트맵 등의 헤드 필드와 데이터 필드로 구성되며, 데이터 영역의 공간 좌표 필드는 다시 x, y좌표들의 집합을 수록하고 있는 가변 길이 레코드에 대한 포인터를 수록하고 있다. 이러한 데이터 필드들의 속성이 길이가 긴 가변 튜플인지 고정 길이로서 원자 값인지를 구분하기 위하여 비트맵을 사용한다. 즉, 비트 맵의 값이 00이면 널 값을 나타내고, 01은 길이가 짧은 고정 길이 포맷을, 10은 길이가 짧은 가변길이 포맷을, 11은 길이가 긴 가변길이 포맷을 나타낸다.

5. 중첩 상속 색인 기법

지금까지 연구된 상속 계층에 대한 색인 기법으

로는 Kim(1989) 등에 의하여 제안된 클래스 계층 색인과 Low(1991) 등에 의해 제안된 H-트리 등이 있다. Low 등이 제안한 객체지향 시스템에서는 H-트리를 상속 계층에 대한 색인 기법으로 사용하고 있지만, 이러한 색인 기법은 상속 계층을 접근하는 질의를 위해 사용되는 색인이며, 집단화 계층과 상속 계층이 합성된 계층을 접근하는 질의에 대해 적용 가능한 색인구조로서는 Bertino 등에 의해 제안된 중첩 상속 색인이 있다.

중첩 상속 색인에 대한 서술에 앞서, 색인 기법과 관련된 몇가지 용어를 정의하면 다음과 같다. 〈그림 5〉의 예에서 클래스 "주거지역"에서 클래스 "소유주명세"의 임의의 속성인 "직업"까지의 집단화 관계를 "주거지역 · 소유주 · 직업"으로 연결하여 나타낼 수 있는데, 이를 경로(path)라 정의한다. 경로의 길이는 경로 상에 나타나는 속성의 수를 나타내며, 경로 상에서 마지막에 위치한 "소유주 명세" 클래스의 "직업" 속성의 원시 값을 키 값으로 하는 색인을 편성하였을 경우 "직업"은 색인 속성이 되고, "소유주 명세"는 색인 클래스가 된다. 또한, 경로 상에 놓인 모든 클래스와 그 클래스들의 상속 계층에 포함되는 모든 서브 클래스의 집합을 해당 경로에 대한 스코프(scope)라 정의하는데, 위에서 주어진 경로의 예에 대한 스코프는 클래스 주거지역, 주택, 상가, 소유주 명세가 된다.

중첩 상속 색인은 집단화 계층과 상속 계층을 모두 접근할 수 있으며 〈그림 8〉과 같이 두개의 B^+-트리 구조를 기반으로 한다. 이 그림에서 볼 때, 색인 속성에 대한 키 값으로 구성된 기본 색인에 대한 B^+-트리가 기본 색인이 되고, 키 값과 연관된 인스턴트들의 부모 리스트를 보관하는 또 다른 B^+-트리는 보조 색인이 된다. 또한 기본색인의

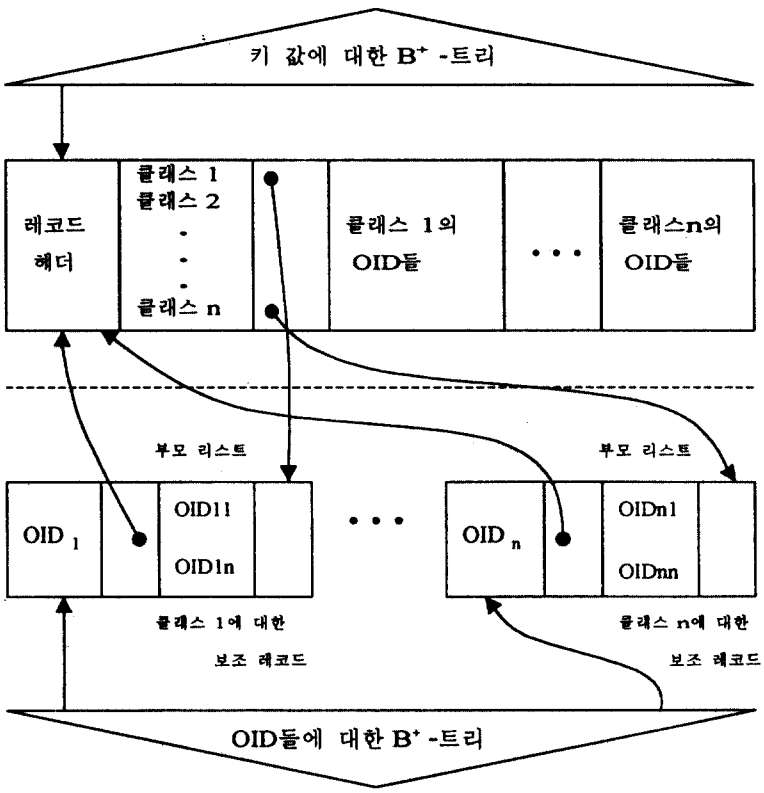

그림 8. 중첩 상속 색인의 구조

단말노드는 기본 레코드, 보조 색인의 단말노드는 보조 레코드가 되며, 이들은 포인터로 서로 연결되어있다.

예를 들어 "직업이 공무원인 소유주가 소유한 주택을 검색하라"라는 질의가 발생했을 경우 색인키로서 "직업" 속성의 값으로 "공무원"을 갖는 기본 레코드를 접근하여 클래스 목록부에서 클래스 "주택"을 식별한 후 변위에 의해 해당 OID 리스트를 탐색함으로써 원하는 결과를 빠른 속도로 검색할 수 있다.

그리고 중첩 상속 색인에서는 객체에 대한 갱신 연산 시에 객체 탐색 없이 변화된 내용을 색인의 기본 레코드부에 반영할 수 있도록 다수의 보조 레코드들과 보조 레코드에 대한 비단말 노드들로 구성된 보조 색인을 사용하는데, 이러한 보조 색인만으로도 키 값과 연관되어 함께 갱신해야 할 OID 리스트들을 식별해 낼 수 있다. 중첩 상속 색인은 전술한 바와 같이 집단화 및 상속계층을 동시에 접근하는 질의를 효율적으로 평가하고 보조색인을 사용하여 추가의 객체탐색 없이 갱신 결

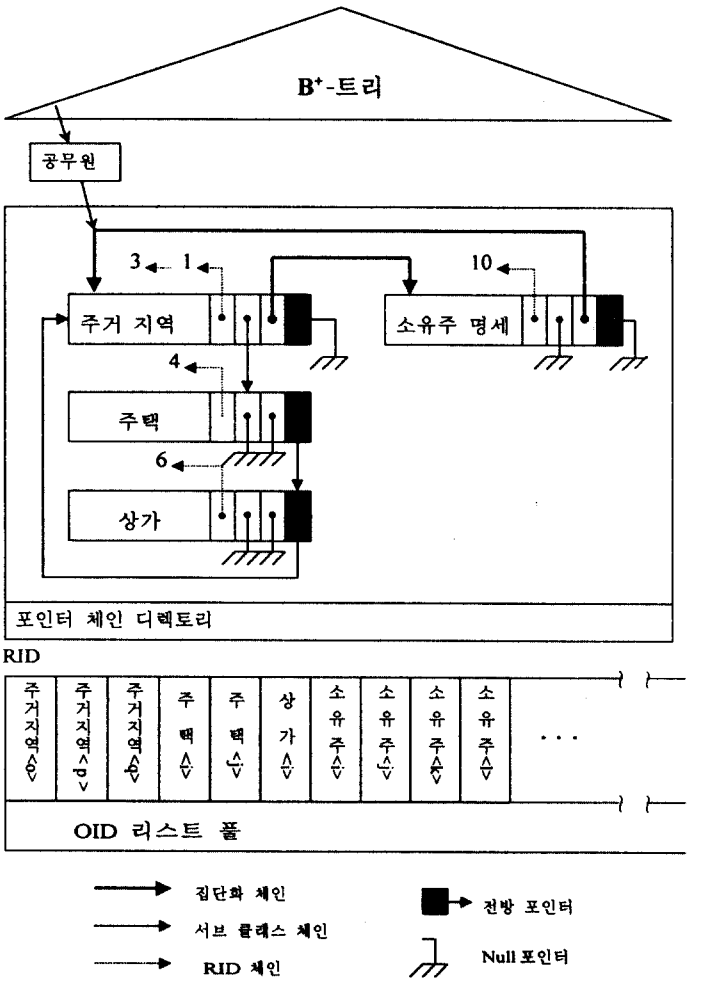

그림 9. 포인터 체인 디렉토리를 이용한 색인 구조

과를 색인에 반영할 수 있으나, 다음과 같은 몇가지 관점에서 그 기능이 취약하다.

첫째, 중첩 상속 색인의 기본 레코드의 목록부에서는 집단화 계층의 경로에 포함되는 여러 상속 계층의 클래스들이 나타나기 때문에 목록만으로는 특정 상속 계층인 클래스 "주거지역"의 서브 클래스들, 즉 "주택" 클래스와 "상가" 클래스를 식별할 수 있는 방법이 없다. 따라서, "공무원이 소유한 주거지역과 그 서브 클래스의 모든 인스턴스를 검색하라"라는 "SELECT-ALL" 형의 질의를 처리하기 위해서는 색인 정보 외에도 상속 계층에 대한 스키마 의미정보를 추가로 채취해야만 처리가 가능하다.

둘째, 기본 레코드의 요소로서 OID 리스트를 포함하고 있는데 〈그림 8〉에서 나타나는 것처럼 특정 클래스에 속하는 OID들이 논리적으로는 집합으로 표현되게 된다. 이는 기본 레코드가 가변길이를 갖는 자료구조임을 의미하고 기본 레코드의 물리적 구현에 상당한 장애 요인이 될 수 있다. 즉 각 클래스에 속하는 OID의 집합 요소가 증가하거나 감소할 때마다 기본 레코드의 전체 길이가 변화하고 기본 레코드의 길이부와 디렉토리부의 각 변위 역시 그때마다 수정되어져야만 하는데 이러한 집합 요소를 물리적 레코드로 구현하기 위한 방안이 구체적으로 제시되지 않고 있다.

셋째, 객체들의 삽입, 삭제 연산시 객체 인스턴스의 직접적인 탐색을 회피하기 위해 고안된 보조 색인부가 객체들의 물리적 저장장소를 탐색하지 않고서도 색인부 만으로 키 값과 연관된 각 OID들의 부모 리스트를 추적할 수 있기 때문에 효율적이나, 기본 레코드에 한번 나타났던 OID들이 보조 색인부에 또 한번 중복 저장되어야 하기 때문에 기억장소를 많이 소모하게 된다.

따라서 본 논문에서는 위와 같은 제반 문제점들을 해결하기 위한 포인터 체인을 이용한 색인 구조를 제안하고자 한다.

1) 포인터 체인 디렉토리를 이용한 색인 구조의 제안

객체지향 시스템에서 사용하는 OID는 논리적 OID와 물리적 OID로 구분할 수 있다. 논리적 OID를 사용할 경우에는 사상(mapping) 기구를 통해 인스턴스의 물리적 주소로 사상되고 물리적 OID는 인스턴스의 물리적 OID를 주소를 OID에서 직접 채취하게 되므로 색인 기법에서는 어떤 방식의 OID를 사용하는가에 상관없이 키 값과 연관된 OID를 효율적으로 식별해 낼 수만 있으면 된다.

본 연구에서 제안하는 색인구조는 〈그림 9〉에서 나타나는 바와 같이 OID 리스트를 저장하는 풀과 키 값과 연관된 클래스와 OID 리스트에 대한 주소를 관리하는 포인터 체인 디렉토리로 구성된다.

포인터 체인 디렉토리에는 집단화 계층의 경로상에서 루트가 되는 클래스들의 집단화 관계를 노드의 후위 포인터를 통하여 체인으로 연결하며 체인의 끝 노드는 각 계층의 선행 노드를 가리키도록 하여 사이클을 갖는 체인 구조를 이룬다. 이때 각 노드는 자신의 서브 클래스들을 가질 수 있기 때문에 노드에 수록된 서브 클래스 멤버 필드를 통하여 상속 관계에 속하는 서브 클래스들을 다시 후위 포인터로 연결하게 된다. 이러한 서브 클래스 멤버 체인 역시 사이클을 이루며, 상속 계층의 깊이에 따라 이러한 사이클을 몇 단계로 중첩하여 구성할 수 있다.

따라서 포인터 체인 디렉토리는 상속계층의 깊이에 따라 중첩된 사이클로 구성된 그래프 구조를 갖는다고 할 수 있다. 그리고 디렉토리의 각 노드는 OID 리스트에서 키 값과 연관된 객체를 식별하기 위한 RID(Record Identifier) 리스트 체인을 가진다. RID는 OID 리스트에서 OID들의 위치를 나타내는 주소로서 디렉토리에서 키 값과 연관된 RID 체인을 탐색하여 OID 리스트에 대한 I/O를 통해 필요한 OID를 채취할 수 있게 된다. OID 리스트 풀의 각 노드에는 OID가 수록되고 각 노드는 RID에 의해 식별된다. 디렉토리의 노드에는 집단화 체인 포인터 필드와 서브 클래스 체인 포인터 필드, 그리고 전방(forward) 포인터 필드 및 RID 체인 포인터 필드가 포함되며, 이를 그림으로 나타내면 〈그림 10〉과 같다.

색인의 운영 예로서 "공무원이 소유한 모든 주거지역과 그 서브 클래스의 인스턴스를 검색하라"라

클래스 식별자	RID 리스트 체인 포인터 필드	서브 클래스 체인 포인터 필드	집단화 체인 포인터 필드	전방포인터 필드

그림 10. 포인터 체인 디렉토리의 노드 구조

는 질의가 발생했을 때 키 값 "공무원"에 의해 포인터 체인 디렉토리로 진입한다. 진입점으로부터 "주거지역" 클래스의 집단화 체인에서 "주거지역" 클래스를 식별한 후, "주거지역" 클래스의 서브 클래스 멤버 체인을 따라 순회하면서 각 노드에 수록된 RID 체인을 통해 소속 RID를 추출한 다음 OID 리스트로부터 해당 OID를 채취하면 된다. 중첩 상속 색인의 경우에는 클래스 목록에 포함되는 정보만으로는 "주거지역" 클래스의 서브 클래스인 "주택" 클래스와 "상가" 클래스를 식별할 수 없어 스키마에 관련된 추가의 의미 정보 없이는 위 질의의 처리가 불가능하다.

또한 포인터 체인 디렉토리 구조의 레코드는 순서적으로 나열된 고정길이의 물리적 레코드들을 포인터로 연결한 형태이기 때문에 물리적 레코드 내에 OID 요소들을 집합으로 표현한 가변 길이의 기본 레코드보다는 구현이 용이하다.

또 다른 질의 예로서 "소유주 명세 클래스와 그 서브 클래스에서 모든 교사들의 주민등록 번호와 성명을 검색하라"라는 질의를 고려해보자. Bertino 등의 중첩 상속 색인은 앞서의 예와 마찬가지로 "소유주 명세" 클래스의 서브 클래스가 있는지 없는지를 식별할 수 없다. 시스템 카탈로그 등에 있는 스키마 정보를 참조하여 이러한 클래스를 식별하였다 할지라도 기본 레코드의 클래스 목록부를 순차적으로 탐색하여 소유주 명세 클래스의 OID 리스트가 위치한 변위를 추출하여야 한다. 본 논문에서 제안한 포인터 체인 디렉토리 구조에서도 소유주 명세 클래스 레코드까지의 경로를 찾기 위해서 이러한 탐색을 시행하여야 하는데 탐색 경로는 진입점으로부터, "주거지역 → 주택 → 상가 → 주거지역 → 소유주 명세"의 순으로 이루어진다. 이때, "주거지역" 클래스의 서브 클래스 노드를 순회하기 위해서는 서브 클래스 체인과 전방 포인터 체인만을 사용하지만 "주거지역"으로부터 "소유주 명세"까지의 경로는 집단화 체인을 경유하게 된다. 즉 "주거지역" 클래스 노드에서 서브 클래스 체인으로 진입하여 "주택" 클래스 노드를 방문하여 서브 클래스 포인터 필드가 널인지를 확인하고 널인 경우에는 하위 클래스가 존재하지 않으므로 "주택" 노드의 전방 포인터에 의해 "상가" 클래스 노드를 방문하고, 마찬가지로 "상가" 클래스 노드의 서브 클래스 포인터 필드의 값이 널이므로 "상가" 노드의 전방 포인터에 의해 "주거지역"으로 다시 귀환하게된다. "주거지역" 노드에서는 다시 집단화 체인 포인터에 의해 "소유주 명세" 노드에 도달할 수 있다. "소유주 명세" 노드의 멤버가 되는 OID들은 RID 체인에서 얻어지는 RID에 의해 OID 리스트 풀로부터 추출할 수 있다.

제안한 색인 기법에서의 검색 비용은 질의에 내포된 해당 클래스까지의 경로를 찾아 RID를 추출하기 위한 포인터 체인 디렉토리의 탐색 비용과 추출된 RID를 가지고 OID 리스트에서 해당 OID를 채취하기 위한 입출력 비용으로 대별될 수 있다. 전자인 포인터 체인 디렉토리의 체인 탐색 비용은 체인의 길이가 길어질 경우에는 상대적으로 증가하기 마련이나, 디렉토리에서 사용되는 노드들은 클래스 명과 키 값과 연관된 OID 들에 대한 RID, 그리고 몇가지 포인터로 구성되기 때문에 그 크기가 아주 작다. 따라서 포인터 체인 디렉토리는 한번 채취 후 주기억 장치에서 계속적인 운영이 가능하다. 또한 후자의 경우인 OID 리스트에 대한 입출력 비용 역시 OID 리스트 상에서 키 값에 의한 집중화(clustering)에 의해 입출력 회수를 최소화 할 수 있다. Bertino 등의 중첩 상속 색인에서는 OID 리스트에 대응되는 기본 레코드부가 역시 키 값에 의해 집중화되어 있기 때문에 오버플로우가 발생하지 않는 이상 이를 한번만 채취하여 주기억 장치에서만 운영하는 것으로 가정하였고 이러한 가정은 본 논문에서 제안한 색인에서도 동일하게 적용할 수 있다.

표 1. 성능 평가에 사용된 인수

D_i	클래스 C_i의 한 속성에 대해 서로 다른 키 값의 수
N_i	클래스 C_i의 인스턴스 수
K_i	클래스 C_i의 한 속성의 키 당 평균 OID의 수 ($K_i = \lceil N_i / D_i \rceil$)
$C_{i,j}$	클래스 C_i를 루트로 하는 상속 계층에서 j번째 클래스
$D_{i,j}$	클래스 $C_{i,j}$의 한 속성에 대해 서로 다른 키 값의 수
$N_{i,j}$	클래스 $C_{i,j}$의 인스턴스의 수
$K_{i,j}$	참조 공유 차수 ($K_{i,j} = \lceil N_{i,j} / D_{i,j} \rceil$)
$OIDL$	객체 식별자의 길이
f	비단말 노드의 평균 분기 수
p	페이지의 크기
pf	포인터 필드의 길이
kl	색인된 속성의 키 값의 평균 길이
kll	키 길이 필드의 크기
rl	레코드 길이 필드의 크기
no	OID리스트의 개수 필드의 길이
of	중첩 상속 색인에서 변위 필드의 길이
cl	포인터 체인 색인에서 클래스 식별자 필드의 길이
Ll	단말노드의 크기(페이지 수)
NL	비단말 노드의 크기(페이지 수)
$nch(i)$	클래스 C_i를 루트로 하는 상속 계층의 모든 클래스 수
$len(p)$	경로의 길이
XNI	중첩 상속 색인의 기본 레코드의 평균길이
XR	포인터 체인 색인의 단말노드의 평균 길이
np	레코드가 페이지 크기 보다 클 때 그 레코드가 점유하는 페이지 수

5. 성능 평가

1) 시뮬레이션 환경

본 연구에서 제안한 포인터 체인 색인의 성능은 집단화 및 상속 계층을 동시에 접근하는 질의에 적용이 가능한 중첩 상속 색인을 비교대상으로 하여 저장 비용과 검색 비용 측면에서 평가하였으며, 시뮬레이션은 SONY-NEWS 워크스테이션의 UNIX 운영체제 하에서 FORTRAN언어와 IMSL 함수 라이브러리를 사용하여 수행하였다. 이때 적용된 비용식과 인수는(Bertino, 1989; Meier, 1986)에서 제시된 내용을 참고하였으며, 사용된 인수는 〈표 1〉과 같다.

그리고 정의 2에서 기술한 바와 같이 상속 계층 전체를 지칭하는 첨자로서 "*" 표기를 사용하기로 한다. 즉, $C_{i,*}$는 클래스 C_i를 루트로 하는 상속 계층의 모든 클래스를 지칭하고, 클래스 C_i를 루트로 하는 상속 계층의 모든 인스턴스의 수를 $N_{i,*}$, 클래스 C_i를 루트로 하는 상속 계층의 모

든 서로 다른 키 값의 수를 $D_{i,*}$ 등과 같이 표기하기로 한다. 〈표 1〉의 인수 중에서 참조 공유 차수(degree of reference sharing)를 나타내는 $K_{i,j}$는 클래스 $C_{i,j}$의 한 속성에 대해 같은 값을 갖는 평균 인스턴스의 수를 나타내며, 이는 다른 클래스의 동일한 객체에 대해 $K_{i,j}$개만큼 공유 참조가 있음을 의미한다.

본 논문에서는 다음과 같은 가정을 기반으로 시뮬레이션을 수행하였다.

첫째, 다중 상속 색인이나 중첩 상속 색인에서 클래스 계층에 관한 정보를 얻기 위해 시스템 카탈로그에 접근하는데 드는 비용은 한번의 입출력 비용으로 산정한다.

둘째, 임의의 상속 계층에서 각 클래스의 인스턴스 수와 한 속성에 대해 서로 다른 키 값의 수는 같다. 즉,

$N_{i,1} = N_{i,2} = \ldots = N_{i,n}$, $D_{i,1} = D_{i,2} = \ldots = D_{i,n}$이다.

2) 저장공간 평가를 위한 비용모델

저장공간 평가를 위한 비용 모델은 비교대상 색인들이 B^+-트리를 기반으로 하고 있기 때문에 편성된 색인에 대해 단말 노드의 크기와 비단말 노드의 크기를 계산하여 이차기억장치를 점유하는 페이지 수로 환산하는 수식들로 구성된다. 단말 노드의 크기에 중요한 영향을 미치는 인수는 키 값과 연관된 객체의 수와 단말 노드에 포함되는 각 필드의 크기인데, 특히 키 값과 연관된 객체의 수는 참조 공유 차수에 의해 결정된다. 이러한 단말 노드들이 점유하는 페이지의 크기가 결정되면 B^+-트리의 비단말 노드의 평균 분기수에 의해 비단말 노드의 페이지 수가 결정되게 된다.

(1) 중첩 상속 색인

중첩 상속 색인의 크기는 기본 색인의 크기와 보조 색인의 크기를 합한 것이며, 기본 레코드의 크기는 경로의 마지막에 나타나는 색인 클래스의 속성에 대한 도메인의 크기와 참조 공유 차수의 크기에 의존한다. 또한 보조 색인의 크기는 집단화 계층의 루트 클래스와 그 서브 클래스를 제외한 나머지 클래스들의 참조 공유 차수의 크기에 비례한다.

먼저 중첩 상속 색인에서 기본 레코드의 평균 길이는 경로의 마지막에 나타나는 색인 클래스의 속성에 대한 도메인의 크기와 참조 공유 차수의 크기를 반영하여 다음과 같은 수식으로 표현할 수 있다.

$$XNI = OIDL \times \left(\sum_{i=1}^{n} \sum_{j=1}^{nch(i)K_{i,j}}\right) + kl + kll + rl + (OIDL + of + pf) \times \left(\sum_{i=1}^{n} nch(i)\right) \quad (1)$$

기본 색인에서 기본 레코드 즉, 단말 노드는 경로 상에서 마지막 속성인 A_n에 대하여 서로 다른 키 값의 수인 $D_{n,*}$ 개수만큼 만들어지며, 이러한 모든 기본 레코드의 크기를 페이지 수로 환산하면 다음과 같다.

$$\begin{cases} Ll = \lceil D_{n,*} / \lfloor P / XNI \rfloor \rceil & \text{if } XNI \le P \\ Ll = D_{n,*} \times \lceil XNI / P \rceil & \text{if } XNI > P \end{cases} \quad (2)$$

중첩 상속 색인에서 비단말 노드가 점유하는 페이지 수는 B+-트리의 단말 노드인 기본 레코드의 크기로부터 비단말 노드의 평균 분기수를 이용하여 다음과 같이 구한다.

$$NL = \lceil Ll / f \rceil + \lceil \lceil Ll / f \rceil / f \rceil + \ldots + X \quad (3)$$

이때, 기본 색인의 크기는 식 (2)와 식 (3)에서 구한 기본 레코드의 크기와 비단말 노드의 크기를 합하면 된다.

$$IS(pri) = Ll + NL \quad (4)$$

중첩 상속 색인에서 보조 색인의 크기는 $C_{1,*}$를 제외한 나머지 클래스들의 키값과 연관된 OID 수에 의해 결정됨으로 이러한 OID 수를 계산하는 수식은 다음과 같다.

$$n(K) = \sum_{j=2}^{n-1}\left(\prod_{i=j}^{n} K_{i,*}\right) \quad (5)$$

식(5)에서 구한 키 값과 연관된 OID 수에 의해 보조 색인의 단말 노드의 페이지 수를 구하는 수식은 다음과 같다.

$$Ll_{aux} = \lceil (n(K) \times OIDL + (OIDL + pf + no) \times (\sum_{i=2}^{n}\sum_{j=1}^{nch(i)} K_{i,j})) / P \rceil \quad (6)$$

중첩 상속 색인의 보조 색인에서 비단말 노드가 점유하는 페이지 수는 식(6)에서 구한 보조 색인의 단말노드의 크기로부터 비단말 노드의 평균 분기수를 이용하여 다음과 같이 계산한다.

$$NL_{aux} = \lceil Ll_{aux} / f \rceil + \lceil \lceil Ll_{aux} / f \rceil / f \rceil + \ldots + X \quad (7)$$

따라서 중첩 상속 색인에서 보조 색인의 크기는 식(6)과 식(7)에서 구한 단말노드의 크기와 비단말 노드의 크기를 합하면 된다.

$$IS(aux) = Ll_{aux} + NL_{aux} \quad (8)$$

중첩 상속 색인의 전체 크기는 식(4)에서 구한 기본 색인의 크기와 식(8)에서 구한 보조 색인의 크기를 합하여 얻을 수 있다.

$$IS = IS(pri) + IS(aux) \quad (9)$$

(2) 포인터 체인 디렉토리

포인터 체인 디렉토리 구조를 갖는 색인의 B+-트리에서 비단말 노드의 크기와 단말 노드에 해당하는 포인터 체인 디렉토리의 구조의 크기를 합한 것이다. 포인터 체인 디렉토리의 평균 길이는 경로의 마지막에 나타나는 색인 클래스의 속성에 대한 도메인의 크기와 참조 공유 차수의 크기를 반영하여 다음과 같은 수식으로 표현할 수 있다.

$$XR = (\sum_{i=1}^{n}\sum_{j=1}^{nch(i)} K_{i,j}) \times (OIDL + pf \times 3) + (cl + pf \times 3) \times \sum_{i=1}^{n} nch(i) \quad (10)$$

포인터 체인 색인에서 단말 노드의 개수는 경로상에서 마지막 속성인 A_n에 대하여 $D_{n,*}$ 개수만큼 만들어지며, 이러한 모든 단말노드 크기를 페이지 수로 환산하면 다음과 같다.

$$\begin{cases} Ll = \lceil D_{n,*} / \lfloor P / XR \rfloor \rceil & \text{if } XR \le P \\ Ll = D_{n,*} \times \lceil XR / P \rceil & \text{if } XR > P \end{cases} \quad (11)$$

포인터 체인 색인에서 비단말 노드가 점유하는 페이지 수는 식(11)에서 구한 단말노드의 크기로부터 비단말 노드의 평균 분기수를 이용하여 다음과 같이 구할 수 있다.

$$NL = \lceil Ll / f \rceil + \lceil \lceil Ll/f \rceil / f \rceil + \ldots + X \quad (12)$$

끝으로 포인터 체인 색인의 전체 크기는 디렉토리 구조를 갖는 단말 노드의 크기와 비단말 노드의 크기를 모두 합하면 된다.

$$MS = Ll + NL \quad (13)$$

3) 검색을 위한 비용 모델

검색을 위한 비용 모델은 B+-트리의 비단말 노드를 경유하여 단말 노드를 채취하기까지의 비용을 입출력 횟수로 환산하는 수식으로 구성된다.

(1) 중첩 상속 색인

중첩 상속 색인에서는 집단화 및 상속 계층을 동시에 접근하는 질의를 처리하기 위해서 한번의 색인 탐색이 필요하며, 이러한 검색 비용을 입출력 수로 계산하기 위한 비용식은 다음과 같다.

$$\begin{cases} N(I/O) = \log_f NL + 1 & \text{if } XNI \le P \\ N(I/O) = \log_f NL + \lceil XNI / P \rceil & \text{if } XNI > P \end{cases} \quad (14)$$

(2) 포인터 체인 디렉토리

포인터 체인 디렉토리에서는 B+-트리를 제외한 디렉토리의 영역을 단말 노드 영역과 동일하게 취급하기 때문에 집단화 및 상속 계층을 동시에 접근하는 질의에 대해서 한번의 색인 탐색이 필요하며, 검색 시에는 다음과 같이 중첩 상속 색인과 동일한 입출력 비용식을 적용할 수 있다.

$$\begin{cases} N(I/O) = \log_f NL + 1 & \text{if } XR \le P \\ N(I/O) = \log_f NL + \lceil XR/P \rceil & \text{if } XR > P \end{cases} \quad (15)$$

4) 시뮬레이션 결과 및 평가

표 2. 시뮬레리션을 위한 인수의 가정

P	= 4096 바이트	OIDL	= 8 바이트
kl	= 8 바이트	kll	= 2 바이트
f	= 218	rl	= 2 바이트
len(p)	= 5	of	= 2 바이트
pf	= 2 바이트	cl	= 2 바이트
no	= 2 바이트		

본 연구에서는 시뮬레이션에 사용된 인수들 가운데 고정 값을 갖는 인수를 〈표 2〉와 같이 가정하였으며, Ki와 nch(i)의 크기의 변화에 따른 다양한 실험을 통하여 성능을 평가하였다. 먼저 한 클래스의 인스턴스 수가 Ni=10,000이고 클래스들 간에는 서로 같은 크기의 Ni와 Di를 가진다고 가정했을 때, nch(i)=5일 경우에 Ki의 크기에 따르는 저장비용의 실험 결과는 〈표 3〉과 같다.

〈표 3〉의 실험 결과에서 보는 것처럼 중첩 상속 색인의 보조 색인은 Ki 값이 증가할수록 키 값과 연관된 부모 리스트의 크기가 커지기 때문에 Ki 값이 증가할수록 색인의 크기가 커지게 되고 반면에 포인터 체인 디렉토리는 Ki 값이 증가할수록 Di 값이 작아지기 때문에 색인의 크기가 작아지게 된다. 그러나 〈표 3〉의 결과에서 포인터 체인 디렉토리의 경우 색인의 크기가 감소하다가 Ki가 12인 시점에서 다시 색인의 크기가 커지게 되는데, 그 이유는 Ki 값이 커짐에 따라 단말노드의 평균 길이가 페이지 크기보다 커져서 한개 이상의 페이지를 점유하게 되었기 때문이다. 실험 결과에서 보는 것처럼 포인터 체인 디렉토리는 저장비용 측면에서 Ki 값이 작을 때는 중첩 상속 색인에 비해서 불리하지만, Ki 값이 8보다 커지는 시점부터는 포인터 체인 색인이 중첩 상속 색인 보다 저장비용에서 큰 이득을 볼 수 있다. 즉, Ki 값이 클 수록 중첩 상속 색인에서의 키 값과 연관된 부모 리스트의 크기가 커져서 보조 색인의 크기가 증가하기 때문에 중첩 상속 색인이 불리한 반면, 상대적으로 Di의 값이 작아짐에 따라 포인터 체인 디렉토리의 크기는 더욱 작아지게 된다. 따라서 포인터 체인 디렉토리의 저장비용은 Ki 값이 클 수록 유리함을 알 수 있다.

다음의 〈표 4〉는 Ni=10,000으로 설정했을 때, 한 상속 계층의 클래스의 수를 나타내는 nch(i)의 변화에 따른 저장비용을 Ki=5, Ki=10, Ki=20에 대해 각각 실험한 결과이다. 실험의 결과에서 Ki=20인 경우에는 포인터 체인 디렉토리가 중첩 상속 색인보다 모든 경우에서 공간 효율이 우수한

표 3. N_i=10,000일 때 K_i의 변화에 따른 저장비용의 실험 결과

저장 비용(페이지수) Ni = 10,000, nch(i) = 5								
Ki / 색인종류	2	4	6	8	10	12	14	16
중첩 상속 색인	4,210	4,519	5,832	8,293	17,538	30,055	51,373	84,526
포인터 체인	8,436	6,306	6,330	6,296	5,037	8,385	7,185	6,288

표 4. N_i=1,000일 때 nch(i)의 변화에 따른 저장비용의 실험 결과

저장비용 (페이지수) N_i = 10,000							
K_i	색인종류	nch(i)=1	nch(i)=2	nch(i)=3	nch(i)=4	nch(i)=5	nch(i)=6
5	중첩 상속 색인	137	598	1,314	3,112	4,150	7,673
	포인터 체인	231	1,016	2,026	4,040	6,306	12,087
10	중첩 상속 색인	149	837	3,152	7,163	17,538	31,890
	포인터 체인	205	1,011	3,023	4,030	5,037	12,072
20	중첩 상속 색인	458	5,656	27,369	83,396	203,220	416,316
	포인터 체인	256	1,009	3,019	4,025	5,031	9,051

표 5. N_i=10,000일 때, nch(i)=2, K_2=1일 때 K_1의 변화에 따른 검색비용

저장 비용(I/O수) Ni = 10,000, nch(i) = 2, len(p) = 2, K_2 = 1,								
K_1 / 색인종류	K_1=2	K_1=4	K_1=6	K_1=8	K_1=10	K_1=12	K_1=14	K_1=16
중첩 상속 색인	3	3	3	3	3	3	3	3
포인터 체인	2	2	2	2	2	2	2	2

것으로 나타났고, Ki 값이 커질수록 이러한 저장 비용의 격차가 더욱 커지게 됨을 알 수 있다. Ki=10이고 nch(i)=3 이상인 경우에서도 포인터 체인 디렉토리의 성능이 우수한 것으로 나타났으며, Ki=5인 경우에는 중첩 상속 색인의 보조 색인의 크기가 포인터 체인 디렉토리의 크기보다 상대적으로 작아지기 때문에 중첩 상속 색인의 성능이 포인터 체인 디렉토리의 성능보다 우수한 것으로 나타났다.

다음으로 검색비용에 관해 고찰하기로 한다. 중첩 상속 색인의 검색 비용은 기본 레코드를 채취하기까지의 B+-트리의 탐색비용으로 볼 수 있고, 기본 색인의 단말노드의 크기가 페이지 크기보다 클 때는 추가의 입출력 비용을 부담해야 한다. 포인터 체인 디렉토리의 검색비용 역시 디렉토리로의 진입을 위한 B+-트리의 탐색비용으로 볼 수 있으며, 단말노드가 여러 페이지를 점유하게 되면 마찬가지로 추가의 입출력 비용을 부담해야 한다.

다음의 표 5는 N_i=10,000, nch(i)=2, len(p)=2, K_2=1로 가정했을 때 K_1의 변화에 따른 검색비용을 실험한 결과이다. 표에서 보는 것처럼 중첩 상속 색인의 경우에는 질의가 목표 클래스와 그의 서브 클래스에 소속된 인스턴스까지를 요구할 때는 클래스 계층에 관한 스키마 정보를 얻기 위해 시스템 카탈로그를 추가로 접근해야하기 때문에 평균 1회의 입출력 비용을 추가로 지불해야만 한다.

7. 결 론

객체지향 지리정보 데이터베이스 시스템에서는 집단화 계층과 상속 계층에서 최적의 접근 경로를 찾아 효율적인 입출력을 수행할 수 있도록 하는 색인기법이 중요하게 다루어지고 있다. 본 연구에서는 집단화 계층과 상속 계층을 동시에 접근하는 질의를 효율적으로 평가하기 위한 포인터 체인 디렉토리 구조를 갖는 색인 기법을 제안하였다. 이 분야의 대표적 기법인 중첩 상속 색인에 비해 제안된 방법의 장점은 목표 클래스의 상속 계층에 나타나는 클래스간의 계층 관계가 내포될 수 있어

추가의 스키마 의미 정보 없이도 쉽게 관련 OID를 추출할 수 있으며, 집합 요소를 갖는 가변 길이 레코드보다 구현이 용이하고, 보조 색인부에서 발생하는 데이타의 중복 저장을 회피함과 동시에, 갱신 연산 시에 발생하는 색인의 유지 보수 절차를 대폭 간략화 할 수 있다는 것이다.

제안된 방법 효용성을 기존의 색인 기법들과 다양하게 비교하였으며, 저장비용과 검색비용 측면에서 그 성능을 다양하게 시뮬레이션한 결과를 제시하였다. 본 연구의 후속 연구는 포인터 체인 디렉토리의 탐색항해 기능을 응용하여 다중 키(multi key)에 의해 중첩 및 상속 계층을 동시에 접근할 수 있도록 확장하는 것이다.

文 献

최웅세 · 장영권 · 홍의경, 1995, GIS를 위한 객체지향 데이터 모델링, **정보과학회지**, 13(3), 7~17.

황규영 · 송주원, 1995, 대형 지리정보 데이터베이스를 위한 객체지향 GIS 엔진, **정보과학회지**, 13(3), 77~87.

Bertino, E. and Poscoli, P., 1995, Index organizations for object-oriented database systems, ***IEEE Trans. on Knowledge and Data Engineering***, 7(2), 193~209.

Bertino, E. and Kim, W., 1989, Indexing techniques for queries on nested objects, ***IEEE Trans. on Knowledge and Data Engineering***, 1(2), 196~214.

Carey, M., DeWitt, D., Richardson, J. and Shekita, E., 1986, Object and file managemant in the EXODUS extensible database system, ***Proc. Int'l. Conf. on Very Large Data Bases***.

Low, C.C., Ooi, B.C. and Lu, H., 1992, H-trees: A dynamic associative search index for CODB, ***Proc. of 1992 ACM SIGMOD Int'l Conf. Management of Data***, 134~143.

Kim, W., Kim, K.C. and Dale, A., 1991, Indexing techniques for object-oriented databases, ***Object-Oriented Concepts, Databases, and Applications***, 371~394.

Lu, H., Ooi, B.C., D'Souza, A. and Low, C.C., 1991, Storage management in geographic information systems, ***Advances in Spatial Databases***, 451~470.

Maier, D. and Stein, J., 1986, Indexing in an object-oriented DBMS, ***Proc. IEEE Workshop on Object-Oriented DBMS***, 171~182.

Oosterom, P. and Bos, J., 1989, An object-oriented approach to the design of geographic information systems, ***Design and Implementation of Large Spatial Databases***, 255~269.

Worboy, M.F., et al., 1990, Object-oriented data modeling for spatial databases, ***Int'l J. of GIS***, 4(4), 369~383.

객체지향 지리정보 데이터베이스를 위한 색인기법

An Indexing Technique for Object-Oriented Geographical Databases

Bu, Ki-Dong*

Summary

One of the most important issues of object-oriented geographical database system is to develop an indexing technique which enables more efficient I/O processing within aggregation hierarchy or inheritance hierarchy. Up to present, several indexing schemes have been developed for this purpose. However, they have separately focused on aggregation hierarchy or inheritance hierarchy of object-oriented data model.

A recent research is proposing a nested-inherited index which combines these two hierarchies simultaneously. However, this new index has some weak points. It has high storage costs related to its use of auxiliary index. Also, it cannot clearly represent the inheritance relationship among classes within its index structure.

To solve these problems, this thesis proposes a pointer-chain index. Using pointer chain directory, this index composes a hierarchy-typed chain to show the hierarchical relationship among classes within inheritance hierarchy. By doing these, it could fetch the OID list of objects to be retrieved more easily than before. In addition, the pointer chain directory structure could accurately recognize target cases and subclasses and deal with "select-all" typed query without collection of schema semantic information. Also, it could avoid the redundant data storing, which usually happens in the process of using auxiliary index.

This study evaluates the performance of pointer chain indexing technique by way of simulation method to compare nested-inherited index. According to this simulation, the pointer chain index is proved to be more efficient with regard to storage cost than nested-inherited index. Especially in terms of retrieval operation, it shows efficient performance to that of nested-inherited index.

Key words : Geographical Information System(CIS), databases, indexing technique, pointer chains

* Associate Professor, Department of Computer Engineering, Kyungil University.