# 다차원 화일구조를 이용한 객체지향 데이타베이스의 중포속성 색인기법

이 종 학†


## 요 약

본 논문에서는 객체지향 데이타베이스의 중포속성에 대한 색인기법으로 다차원 색인구조를 이용하는 다차원 중포속성 색인기법인 MD-NAI를 제안한다. 중포속성에 대한 기존의 색인기법들은 중포된 객체에 대한 기존의 색인기법들은 B+-tree와 같은 일차원 색인구조를 이용함으로써, 클래스 계층과 중포속성이 포함된 복합 형태의 질의들에 대한 처리를 잘 지원하지 못한다. MD-NAI에서는 객체지향 데이타베이스의 클래스 계층에 대한 색인기법인 이차원 클래스 계층 색인기법(2D-CHI)을 다차원으로 확장한다. 2D-CHI는 키 속성 도메인과 클래스 식별자 도메인으로 구성된 이차원 도메인 공간상에서 객체들의 클러스터링을 다루는 색인기법이다. 본 논문의 MD-NAI에서는 색인된 중포속성을 표현하는 경로상의 각 클래스 계층마다 하나의 클래스 식별자 도메인을 할당하여 구성된 다차원 도메인 공간상에서 색인 엔트리들의 클러스터링을 다룬다. 따라서, MD-NAI에서는 기존의 색인기법에서 지원하기 어려운 질의의 대상 범위가 클래스 계층상의 임의의 클래스들로 제한되거나, 질의에 포함된 복합속성들의 도메인이 클래스 계층상의 임의의 클래스들로 제한되는 경우에도 잘 지원할 수 있다.


# Indexing Techniques for Nested Attributes of OODB Using a Multidimensional Index Structure

Jong-Hak Lee†


## ABSTRACT

This paper proposes the *multidimensional nested attribute indexing techniques (MD-NAI)* in object-oriented databases using a multidimensional index structure. Since most conventional indexing techniques for object-oriented databases use a one-dimensional index structure such as the B+-tree, they do not often handle complex queries involving both nested attributes and class hierarchies. We extend a tunable two-dimensional class hierachy indexing technique(2D-CHI) for nested attributes. The 2D-CHI is an indexing scheme that deals with the problem of clustering objects in a two dimensional domain space that consists of a key attribute domain and a class identifier domain for a simple attribute in a class hierachy. In our extended scheme, we construct indexes using multidimensional file organizations that include one class identifier domain per class hierarchy on a path expression that defines the indexed nested attribute. This scheme efficiently supports queries that involve search conditions on the nested attribute represented by an extended path expression. An extended path expression is a one in which a class hierarchy can be substituted by an individual class or a subclass hierarchy in the class hierarchy.


## 1. 서 론

객체지향 데이타베이스는 클래스 상속(inheritance) 개념에 의하여 클래스들 사이에 클래스 상속 계층(일반적으로 클래스 계층으로 약칭함)이라는 하나의 계층구조를 이룬다. 즉, 하나의 클래스는 여러개의 서브클래스를 가지며, 각 서브클래스는 또 다른 여러 서브클래스들을 가진다. 따라서, 객체지향 데이타베이스에서는 질의의 대상 범위를 두가지 경우로 나타낼 수 있

※ 본 논문은 정보통신부에서 지원하는 대학기초연구지원사업으로 수행되었음.

† 종신회원 : 대구효성가톨릭대학교 컴퓨터정보통신공학부 교수

논문접수 : 2000년 4월 17일, 심사완료 : 2000년 7월 25일

다. 한 경우는 질의의 대상 범위를 질의에 나타나는 클래스만으로 한정하는 것이고, 또 다른 경우는 질의의 대상 범위를 질의에 나타나는 클래스와 그의 모든 서브클래스들을 포함하는 것이다[10]. 이러한 클래스 계층에 대한 질의를 효율적으로 처리할 수 있는 색인구조는 특정 클래스에 속하는 객체들의 탐색뿐만 아니라 특정 클래스를 루트로 하는 클래스 계층의 모든 클래스들에 속하는 객체들의 탐색도 효율적으로 처리할 수 있어야 한다.

객체지향 데이타베이스는 또 하나의 중요한 개념인 클래스 집단화(aggregation) 개념에 의하여 한 클래스가 가지는 속성의 도메인이 또 다른 클래스가 되게 함으로써(이러한 속성을 *복합속성*이라 함) 클래스 집단화 계층을 이룬다. 따라서, 클래스 집단화 계층상의 모든 클래스에서 정의된 어떠한 속성도 논리적으로는 루트 클래스의 속성이라고 볼 수 있다. 우리는 이런 속성을 루트 클래스의 *중포속성(nested attribute)*[2, 11]이라 한다. 이로 인하여 객체지향 질의어에서는 기존의 관계형 데이타베이스에서의 질의어와는 달리 중포속성에 조건이 주어지는 *중포술어*(nested predicate)[2, 10]를 가진다는 특징이 있다. 중포속성을 표현하기 위해서는 *경로식(path expression)*[3, 6]이 사용되며, 경로식은 루트 클래스로부터 클래스 집단화 계층을 따라 나타나는 속성들의 나열로 표현된다.

중포속성을 표현하는 경로식에는 속성값(객체 참조자 : Oid)들의 참조관계에 의한 객체와 객체 사이에 *암시적 조인(implicit join)*[11]의 의미를 가지고 있다. 이러한 암시적 조인은 데이타베이스 스키마에 의해 미리 예상이 가능하다. 따라서, 질의에 자주 나타나는 중포속성에 대한 암시적 조인을 미리 계산하여 그 결과를 색인으로 구축하여 놓음으로써, 질의처리시 이를 이용하여 성능 향상을 꾀할 수 있으며 이를 중포속성에 대한 색인기법[2, 4, 6, 16]이라 한다. 그러나, 이러한 중포속성에 대한 기존의 색인기법들은 B$^+$-tree와 같은 일차원 색인구조를 이용함으로써, 클래스 상속에 의한 특징으로 질의의 대상 범위가 클래스 계층상의 임의의 클래스들로 제한되거나, 경로식에 나타나는 속성의 도메인이 클래스 계층상의 임의의 클래스들로 제한이 되는 질의들을 지원하기 어려운 문제점을 가지고 있다.

본 논문에서는 이와 같은 일차원 색인구조를 이용하는 기존의 중포속성에 대한 색인기법들이 가지는 문제점을 해결하기 위하여, 다차원 동적 화일구조를 중포속성에 대한 색인구조로 이용하는 방안을 제안한다. 다차원 동적 화일구조는 다차원 클러스터링을 지원하는 화일구조로서 클러스터링의 특성이 화일을 구성하는 여러 속성들에 의해서 공유된다[12]. 본 논문에서 제안하는 중포속성에 대한 색인기법에서는 단순속성의 클래스 계층에 대한 색인기법인 이차원 클래스 계층 색인기법[13](2D-CHI : 2-Dimensional Class Hierarchy Index)에서 이용한 이차원 화일구조를 다차원 화일구조로 확장하여 이용한다. 이차원 클래스 계층 색인기법에서는 키 속성의 값들로 구성된 *키 속성 도메인*과 클래스 계층을 이루는 클래스들의 *클래스 식별자 도메인*으로 구성된 이차원 도메인상에 주어진 색인 엔트리(객체 식별자인 Oid)들의 클러스터링 문제를 다룬다.

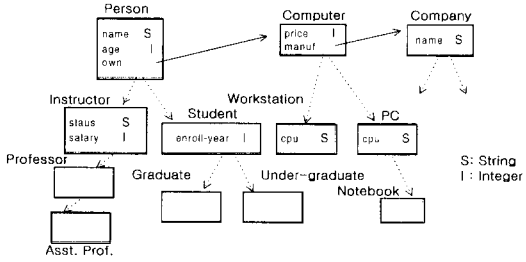
이차원 클래스 계층 색인기법에서 이용한 이차원 화일구조를 다차원 화일구조로 확장하여 중포속성에 대한 색인기법으로 확장할 수 있다. 즉, 다차원 화일구조를 이용한 중포속성에 대한 색인기법에서는 중포속성의 키 속성 도메인과 함께, 경로식에 나타나는 각 속성마다 한 축의 클래스 식별자 도메인을 할당하여 구성된 다차원 도메인상에 주어진 색인 엔트리들의 클러스터링 문제를 다룬다. 이와 같은 색인기법에서는 기존의 B$^+$-tree와 같은 일차원 색인구조를 이용한 색인기법들에서 문제가 되는 질의의 대상 범위가 클래스 계층상의 임의의 클래스들로 제한되거나, 경로식에 나타나는 속성의 도메인이 클래스 계층상의 임의의 클래스들로 제한이 되는 질의들의 처리를 한번의 색인 탐색으로 가능하게 할 수 있다.

본 논문의 구성은 다음과 같다. 제2절에서는 관련 연구로서 객체지향 데이타베이스의 색인 구축에 필요한 기본 개념들을 살펴본다. 제3절에서는 객체지향 데이타베이스의 중포속성에 대한 색인기법으로 이차원 클래스 계층 색인구조를 확장하여 다차원 화일구조를 이용하는 방법을 제안하고, 제4절에서는 제안된 중포속성 색인기법들에 대한 성능 비교의 결과를 제시한다. 마지막으로, 제5절에서는 결론과 향후 연구방향을 기술한다.

## 2. 관련 연구

객체지향 데이타베이스에서 사용자가 정의하는 클래스들은 루트를 가진 이차원 방향성 그래프(rooted two dimensional directed graph)를 형성하며, 이 그래

프를 그 클래스에 대한 *스키마 그래프(schema graph)*라 정의한다[10, 11]. (그림 1)은 클래스 Person에 대한 스키마 그래프이다. (그림 1)에서 클래스 Person은 서브 클래스 Instructor와 Student, 그리고 Instructor와 Student의 서브 클래스들을 포함하는 클래스 상속 계층구조와 Computer와 Company를 포함하는 클래스 집단화 계층구조의 루트이다. 속성 own의 도메인은 Computer와 그의 서브 클래스 Workstation, PC 및 Notebook들이고, 속성 manuf.의 도메인은 Company이다.




(그림 1) 클래스 Person에 대한 스키마 그래프

객체지향 데이타베이스에서 클래스는 속성들(attributes)을 가진다. 속성은 정수 타입이나 문자 타입과 같은 기본(primitive) 타입의 값(value)을 가지는 *단순속성(simple attributes)*과 다른 클래스에 속하는 객체의 객체 식별자(Object identifier : Oid)를 값으로 가지는 *복합속성(complex attributes)*으로 구분된다[10]. 클래스 $C$의 *중포속성(nested attributes)*[2]이란 클래스 $C$로부터 실선 링크로 연결된 클래스에서 정의된 속성들을 의미하며, 클래스 $C$와 속성들의 연속(sequence)으로 표시한다. 예를들어 (그림 1)에서 클래스 Company에 정의된 속성 name은 클래스 Person과 Computer의 중포속성이며, 각각 Person.own.manuf.name과 Computer.manuf.name으로 표시한다.

클래스 집합 $C^*$는 클래스 $C$와 그의 모든 서브 클래스들을 원소로 하는 집합으로 정의한다. 예를들어 (그림 1)에서 클래스 Person*는 집합 {Person, Instructor, Student, Professor, Asst. Prof., Graduate, Under-graduate}이고, Student*는 {Student, Graduate, Under-graduate}이다.

### 2.1 객체지향 질의어

본 논문에서는 Kifer[7]가 제안한 질의어 모델을 가정하여 객체지향 질의어의 특징을 설명한다. 객체지향 질의어는 Select, From, Where 절로 구성되며 각 절에서 객체지향 개념을 지원하도록 확장되었다. From 절에는 질의의 대상이 되는 클래스를 기술하며, Where 절에는 단순속성에 대한 조건인 *단순술어(simple predicate)*와 더불어 중포속성에 대한 조건인 *중포술어(nested predicate)*[2]를 사용할 수 있다.

*경로식(path expression)*[3, 6, 7]은 객체지향 질의에서 중포속성을 표현하는 방법이다. 경로식은 클래스 집단화 계층구조상에서 클래스 이름과 속성의 교차적인 나열(sequence)로서 다음과 같은 형태를 가진다. 단, $A_i$ 뒤의 중괄호({ })는 선택적임을 나타내는 표시이다.

$$P = C_1.A_1\{[C_2]\}.A_2\{[C_3]\}...A_n\{[C_{n+1}]\} \quad (1)$$

경로식 $P$에서 클래스 $C_1$을 *타겟 클래스*, 속성 $A_i$의 도메인이 되는 $C_{i+1}$을 $A_i$의 *도메인 클래스*라 정의한다. 타겟 클래스와 도메인 클래스는 경로에서 클래스 계층구조에 속하는 특정 클래스로 한정(limit)될 수 있다[1] [7, 10, 11]. 수식 (1)의 경로 $P$에서 속성 $A_i$의 도메인은 뒤에 대괄호 ([])가 생략되면 자동으로(default) 클래스 집합 $C^*_{i+1}$로 되지만, $C^*_{i+1}$에 속하는 특정 클래스가 대괄호 안에 지정되면 대괄호 안의 클래스로 한정된다.

경로식에서 속성의 도메인 클래스가 특정 클래스로 한정되는 것을 *도메인 대치(domain substitution)*라 하고, 타겟 클래스가 특정 클래스로 한정되는 것을 *타겟 대치(target substitution)*라 한다. 그리고 타겟 클래스나 도메인 클래스의 대치를 통칭하여 *클래스 대치(class substitution)*라 한다. 이와 같은 클래스 대치가 있는 경로식을 특히 *확장된 경로식(extended path expression)*이라 한다[7]. 이러한 클래스 대치는 질의의 범위를 특정한 클래스로 한정할 수 있도록 하여 클래스 상속의 개념을 객체지향 질의에 표현하도록 한 것이다[7, 11]. 다음 중포술어들은 (그림 1)의 클래스 스키마 그래프에 대해 확장된 경로식으로 표현된 클래스 대치에 대한 예를 보여주고 있다.

*pd*1 : Student*.own.price >= "300만"
*pd*2 : Student.own.price >= "300만"
*pd*3 : Student*.own[PC*].price >= "300만"
*pd*4 : Student.own[PC].price >= "300만"

1) 참고 문헌 [1, 10] 등에서 수퍼 클래스 대신에 서브 클래스를 사용할 수 있다는 대치(substituability) 개념을 의미함.

중포술어 pd1은 클래스 집합 Student*, 즉, Student, Graduate, Under-graduate에 속하는 객체를 대상으로 하는 조건이며, pd2는 클래스 Student에만 속하는 객체를 대상으로 하는 조건이다. 그리고, pd3은 클래스 집합 Student*를 대상으로 own의 도메인을 클래스 집합 PC*, 즉, PC, Notebook 등으로 하며, pd4는 own의 도메인을 PC만으로 한정하여 부과한 조건이다.

경로식 $P$에서 *경로 인스턴스*(*path instance*)는 다음 조건을 만족하는 객체들의 리스트 $(O_1, O_2, ..., O_{n+1})$로 정의한다[2, 7]. (1) 객체 $O_1$은 클래스 $C_1$의 객체이다. (2) 객체 $O_i$ $(1 <= i <= n+1)$는 클래스 $C_i$의 객체로서 객체 $O_{i-1}$의 속성 $A_{i-1}$의 값이다. 예를 들어, 경로식 Person.own[Workstation].manuf. name에서 객체 s ∈ Person가 속성 own의 값으로 객체 t ∈ Workstation의 Oid를 가지고, 객체 t에서 속성 manuf.의 값으로 객체 c ∈ Company의 Oid를 가지며, 객체 c에서 속성 name의 값으로 hyundai를 가진다면, 객체들의 나열인 (*s, t, c, hyundai*)는 경로식 Person.own[Workstation].manuf. name에 대한 경로 인스턴스가 된다.

### 2.2 객체지향 질의처리를 위한 기존 색인기법

객체지향 데이타베이스의 중포술어를 가지는 질의를 수행하기 위한 기법으로 순방향 운행법(forward traversal), 역방향 운행법(backward traversal), 그리고 혼합 운행법(mixed traversal)이 제시되었다[8]. 순방향 운행법은 스키마 그래프의 집단화 계층구조에서 상위 클래스를 먼저 탐색하고 하위 클래스를 나중에 탐색하는 방법이고, 역방향 운행법은 하위 클래스를 먼저 탐색하고 상위 클래스를 나중에 탐색하는 방법이다. 그리고, 혼합 운행법은 순방향과 역방향 운행법을 혼합한 탐색방법이다. 역방향 운행시에는 하위 클래스의 객체를 참조하는 상위 클래스의 객체를 탐색하기 위해서는 많은 비용을 필요로 한다.

객체지향 데이타베이스에서 색인을 유지하는 속성이 단순속성이면 관계형 데이타베이스 시스템에서와 같이 색인된 속성에 대한 제한 술어를 만족하는 객체의 신속한 탐색을 위해서 사용될 수 있고, 색인을 유지하는 속성이 복합속성이면 역방향 운행시 상위 클래스의 객체를 탐색하기 위한 비용을 줄일 수 있다. 따라서, 객체지향 데이타베이스에서 색인의 키값은 단순속성의 도메인이 되는 기본 타입의 값들뿐만 아니라 복합속성의 도메인이 되는 사용자 정의 클래스의 객체 식별자인 Oid들도 될 수 있다. 그리고, 중포속성에 대한 색인기법으로 중포속성을 표현하는 경로식에 의한 암시적 조인을 미리 계산하여 색인구조로 유지함으로써, 질의처리시 클래스 집단화 계층에 대한 운행을 생략할 수도 있다[5, 10].

지금까지 중포속성에 대한 색인기법을 위해 제안된 색인구조들로, Bertino 와 Kim[2]이 제안한 중포 색인(nested index), 경로 색인(path index), Kemper와 Moerkotte[6]가 제안한 ASR(Access Support Relation), 그리고 Xie와 Han[16]이 제안한 JIH(Join Index Hierarchy)등이 있다. 임의의 경로 $p = C_1.A_1...A_n$에 대하여, *중포 색인*은 $A_n$을 키로하여 $C_1$에 속하는 객체들의 Oid를 찾는 색인구조이고, *경로 색인*은 $A_n$을 키로하여 $C_1$, ..., $C_n$에 속하는 객체들의 Oid 리스트(즉, 경로 인스턴스)를 찾는 색인구조이다. ASR은 경로식에 대한 경로 인스턴스들에서 Oid만 추출하여 릴레이션 형태로 구성한 색인구조로서 경로 색인과 유사한 색인구조이다. 그리고, JIH는 하나의 경로에서 두 개의 클래스 사이에 직접 또는 간접 참조관계가 있는 객체들의 Oid쌍들로 구성된 색인구조로서, 직접 참조관계가 있는 Oid들의 쌍들을 기본 조인 색인구조라 하였고, 간접 참조관계가 있는 Oid들의 쌍들은 기본 조인 색인구조로부터 유도하여 구축함으로써 유도된 조인 색인구조라 하였다.

그러나, 이러한 색인구조들은 클래스 상속에 의한 객체지향 데이타 모델의 특징을 반영하지 못하는 것들로서 타겟 클래스의 대치 또는 도메인 클래스의 대치가 있는 경로식으로 표현된 질의는 지원하지 못한다. 즉, 질의의 대상 범위가 클래스 상속 계층상의 임의의 클래스들로 제한되거나, 경로식에 나타나는 어떠한 속성의 도메인이 클래스 상속 계층상의 임의의 클래스들로 제한이 되는 질의들을 지원하지 못한다.

Bertino[4]는 질의의 대상 범위가 임의의 클래스들로 제한되는 타겟 클래스 대치가 있는 경로식으로 표현된 중포속성에 대한 질의를 지원할 수 있는 NIX(Nested-Inherited Index)라고 하는 색인구조를 제안하였다. NIX는 클래스 계층에 대한 색인구조의 하나인 CH-index[9]와 중포 색인을 통합한 형태의 색인구조이다. 즉, NIX는 B+-tree의 단말 노드를 색인된 중포속성의 값을 키로하여 스키마 그래프상에서 직접 또는 간접적으로 이 중포속성을 내포하는 모든 클래스들의 Oid들을 대상으로 클래스별 구분을 위한 클래스 디렉토리와 함께 구성한 색인구조이다. 그러나, NIX 역시 도메인 클

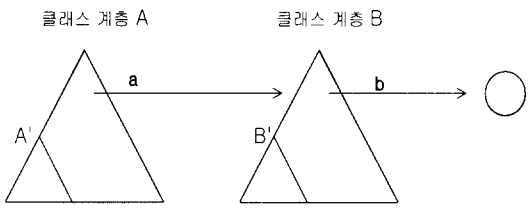
래스의 대치가 있는 경로식으로 표현된 질의를 지원하지 못한다. 이는 색인된 중포속성으로부터 각 타겟 클래스까지의 경로 정보를 유지하지 않기 때문이다.

## 3. 중포속성에 대한 다차원 색인기법

본 절에서는 객체지향 데이타베이스의 중포속성에 대한 색인기법으로 참고문헌 13에서 제안한 이차원 클래스 색인기법을 확장하는 다차원 중포속성 색인기법에 관하여 논의한다. 먼저, 제3.1절에서는 객체지향 데이타베이스의 질의에 나타나는 중포술어들의 특징을 살펴보고, 제3.2절에서는 이러한 중포술어를 만족하는 객체들의 신속한 탐색을 위한 색인구조들을 제안한다. 그리고, 제3.3절에서는 이들에 대한 운용 알고리즘을 제시한다.

### 3.1 중포술어의 특징

(그림 2)는 객체지향 데이타베이스의 스키마 그래프에서 세개의 클래스 계층으로 이루어진 하나의 경로를 나타내는 경로 스키마이다. (그림 2)의 경로 스키마에서, 클래스 계층 A에서 정의된 복합속성 a의 도메인은 클래스 계층 B이며, 클래스 계층 B에서 정의된 단순속성 b는 클래스 계층 A의 중포속성이다. 그리고, A'와 B'는 각각 클래스 계층 A와 B의 서브클래스 계층을 나타낸다.




(그림 2) 객체지향 데이타베이스의 경로 스키마

(그림 2)와 같은 경로 스키마에 대하여 중포속성 b에 조건이 주어진 중포술어에는 중포속성을 표현하는 경로식의 특성에 따라 다음과 같은 세가지 형태로 분류할 수 있다.

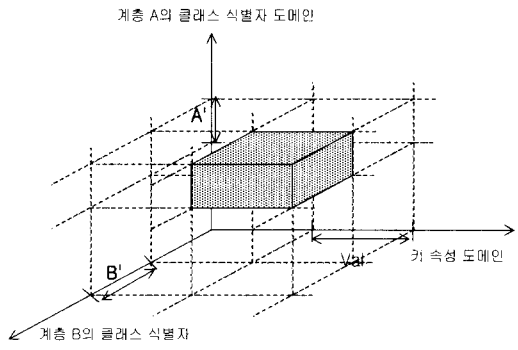
- 중포술어 1 : $A$.a.b $\theta$ Val
- 중포술어 2 : $A'$.a.b $\theta$ Val
- 중포술어 3 : $A'$.a[B'].b $\theta$ Val

이들은 모두 중포속성 b에 조건이 주어진 술어들이지만, 각 중포술어를 만족하는 결과는 서로 다르게 된다. 즉, 중포술어 1은 클래스 계층 A의 모든 클래스를 질의의 대상으로 중포속성 b에 조건이 주어진 술어이고, 중포술어 2는 중포술어 1에서 질의의 대상을 클래스 계층 A에서 A'에 속하는 클래스들만으로 한정한 것이다. 그리고, 중포술어 3은 중포술어 2에서 복합속성 a의 도메인으로 클래스 계층 B에서 B'에 속하는 클래스들로 한정한 것이다. 중포술어 2에서 사용된 경로식에는 타겟 클래스 대치가 있는 경우이며, 중포술어 3에서 사용된 경로식에는 타겟 클래스 대치와 도메인 클래스 대치가 있는 경우이다.

### 3.2 중포속성에 대한 다차원 색인구조

이차원 클래스 색인기법의 이차원 색인구조를 (중포속성을 표현하는 경로식의 *경로길이+1*)차원의 색인구조로 확장하여 중포속성에 대한 다차원 색인기법으로 이용할 수 있다. 즉, 중포속성에 대한 색인구조로서 키 속성 도메인과 함께 중포속성을 표현하는 경로상의 각 클래스 계층마다 클래스 식별자들로 구성된 한 차원씩의 클래스 식별자 도메인을 할당하여 다차원의 도메인 공간을 구성한다. 예를들어, (그림 2)와 같은 경로 스키마에서 중포속성 b에 대한 색인구조로서 삼차원 색인구조를 이용하여 다음과 같은 삼차원 도메인 공간을 구성할 수 있다. (그림 3)은 이와 같이 구성된 삼차원 도메인 공간상에서 중포술어 3에 대한 질의 영역을 표현한 것이다.

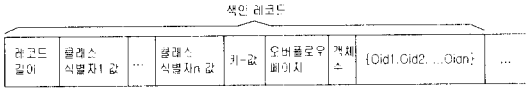
- 축1 : 중포속성 b의 키 속성 도메인
- 축2 : 클래스 계층 A의 클래스 식별자 도메인
- 축3 : 클래스 계층 B의 클래스 식별자 도메인



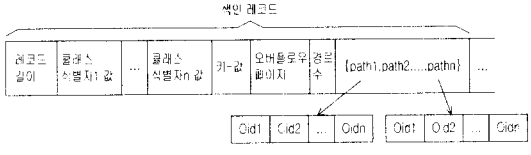

(그림 3) 중포술어 3에 대한 질의 영역

본 논문에서는 객체지향 데이타베이스의 중포속성에 대한 색인구조를 다차원 화일구조의 하나인 계층 그리드 화일(multilevel grid file : MLGF)[14, 15]을 이용하여 구성하고, 이를 *MD-NAI(Multidimensional Nested Attribute Index)*라 한다. MD NAI는 디렉토리와 색인 페이지로 구성된다.[2] 디렉토리는 다단계의 균형된 트리 구조를 가지며, 디렉토리를 구성하는 디렉토리 페이지의 구조는 MLGF[14, 15]에서와 마찬가지이다. 색인 페이지는 (그림 4)와 (그림 5)에서와 같이 색인 레코드들로 구성되며, 각 색인 레코드에는 경로상의 각 클래스 식별자 값(class-id value) 필드, 키 값(key value) 필드, 객체 또는 경로의 개수 필드, 및 이들에 대한 색인 엔트리들의 리스트 필드가 있다. 그리고, 레코드의 크기가 페이지의 크기보다 크게될 때 오버플로우 페이지를 할당하고 이를 포인트하기 위한 오버플로우 페이지(overflow page) 필드가 있다.

본 논문에서는 MD-NAI를 색인 레코드에 있는 색인 엔트리의 구성방법에 따라 확장된 중포 색인구조과 확장된 경로 색인구조의 두 가지 색인구조로 분류한다. *확장된 중포 색인구조(ENI : Extended Nested Index)*는 (그림 4)와 같이 색인 엔트리를 색인된 중포속성의 타겟 클래스 계층에 속하는 객체에 대한 객체 식별자(즉, Oid)들로 구성하며, *확장된 경로 색인구조(EPI : Extended Path Index)*는 (그림 5)와 같이 색인 엔트리를 색인된 중포속성에 대한 경로 인스턴스(즉, Oid 리스트)들로 구성한다.




(그림 4) ENI의 색인 페이지 구조




(그림 5) EPI의 색인 페이지 구조

중포속성에 대한 색인기법은 경로상의 참조관계에 의한 암시적 조인의 연산을 미리 계산하여 두는 기법으로 중포술어를 만족하는 객체들의 탐색에는 매우 유용하지만, 상대적으로 색인구조의 유지비용을 많이 필요로 한다. 이러한 색인구조의 유지비용은 색인을 유지하는 경로의 길이에 비례하기 때문에, 경로의 길이가 매우 길 때는 경로를 여러 개의 서브경로로 나누어서, 서브경로별로 여러 개의 색인구조를 유지함으로써 유지비용을 줄일 수 있다. 특히, 서브경로의 길이가 1인 경우의 MD-NAI 색인구조는 참고문헌 13에서 제안한 이차원 클래스 색인구조와 같게 된다. 즉, 경로상의 클래스 계층마다 이차원 클래스 색인구조를 단계적으로 구축함으로써 중포술어를 만족하는 타겟 클래스 계층의 객체들을 여러번의 단계적인 색인 검색으로 구할 수 있다. 우리는 이와 같이 경로상의 클래스 계층마다 구축하는 색인구조를 *다중 이차원 클래스 색인구조(MTI : Multiple Two dimensional class Index)*라 한다.

### 3.3 각 색인구조의 운용 알고리즘

#### 3.3.1 ENI : 확장된 중포 색인구조

ENI는 경로 인스턴스를 유지하는 EPI에 비해 저장공간의 오버헤드가 적은 반면에, 데이타베이스의 변경에 따른 색인구조의 유지비용에 대한 오버헤드가 많다. 경로 $P$에서 i번째 클래스 계층에 있는 임의의 객체 $O_i$에서 속성 $A_i$의 값으로 $O_{i+1}$에서 새로운 $O'_{i+1}$로 변경된 경우, ENI의 갱신을 위해서는 다음과 같은 단계의 작업이 필요하다.

- **단계 1** : 객체 $O_{i+1}$로부터 중포속성 $A_n$까지 경로상의 클래스 식별자 값들과 함께 키 속성 값으로 구성된 색인키 리스트들의 집합 K(A)를 구한다. 이를 위해서는 객체 $O_{i+1}$로부터 경로를 따라 순방향 운행을 하여야 한다. 여기서, 경로상의 임의의 속성 $A_i$가 다중값을 갖지 않으면 K(A)는 하나의 원소만을 가진다.
- **단계 2** : 객체 $O'_{i+1}$로부터 중포속성 $A_n$까지 경로상의 클래스 식별자 값들과 함께 키 속성 값으로 구성된 색인키 리스트들의 집합 K(B)를 구한다. 여기서, K(A) = K(B)이면 색인의 갱신이 필요 없으며, 그렇지 않으면 단계 3을 시행한다.
- **단계 3** : $O_i$를 직접 또는 간접적으로 참조하는 타겟

2) MD-NAI의 디렉토리 페이지와 색인 페이지는 B-tree의 비단말(non-leaf) 페이지와 단말(leaf) 페이지에 해당한다.

클래스 계층 $C_1$의 객체 Oid들의 집합 R을 구한다. 이를 위해서는 객체 $O_i$로부터 경로를 따라 역방향 운행을 하여야 한다. 여기서, i = 1이면 R은 $\{O_i\}$가 된다.

- **단계 4** : 색인의 갱신을 다음과 같이 시행한다.
  - K(A) ⊃ K(B)이면, K(A) - K(B)를 $\delta$로 하고 $\delta$에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 Oid들을 제거한다.
  - K(A) ⊂ K(B)이면, K(B) - K(A)를 $\delta$로 하고 $\delta$에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 Oid들을 첨가한다.
  - 그렇지 않으면, K(A) - K(B)를 $\delta$1, K(B) - K(A)를 $\delta$2라 하고, $\delta$1에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 Oid들을 제거하고, $\delta$2에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 Oid들을 첨가한다.

이와같이 ENI에서는 데이타베이스의 변경에 따른 색인구조의 갱신을 위하여, 경로상의 역방향 운행이 필요하다. 따라서, 데이타베이스의 각 객체내에 자신을 참조하는 객체에 대한 역 참조자(reverse reference)가 존재하지 않으면, ENI는 사용할 수 없다. 그리고, 객체의 삽입과 제거에 따른 색인구조의 갱신을 위해서는 한번의 순방향 운행만이 필요하며, 이는 변경의 경우와 같다.

3.3.2 EPI : 확장된 경로 색인구조

EPI는 색인 레코드에 경로 인스턴스를 저장하기 때문에 ENI에 비해 많은 양의 저장 공간을 필요로 하는 반면에, 데이타베이스의 변경에 따른 색인구조의 유지 비용에 대한 오버헤드가 ENI에 비해 적게 된다. 경로 $P$에서 i번째 클래스 계층에 있는 임의의 객체 $O_i$에서 속성 $A_i$의 값으로 $O_{i+1}$에서 새로운 $O'_{i+1}$로 변경될 경우, EPI의 갱신을 위해서는 다음과 같은 단계의 작업이 필요하다.

- **단계 1** : 객체 $O_{i+1}$로부터 중포속성 $A_n$까지 경로상의 클래스 식별자 값들과 함께 키 속성 값으로 구성된 색인키 리스트들의 집합 K(A)를 구한다. 이를 위해서는 객체 $O_{i+1}$로부터 경로를 따라 순방향 운행을 하여야 한다.
- **단계 2** : 객체 $O'_{i+1}$로부터 경로에 따른 순방향 운행을 통하여 중포속성 $A_n$까지 서브경로 인스턴스들의 집합 PI.suf와 경로상의 클래스 식별자 값들과 키 속성 값으로 구성된 색인키 리스트들의 집합 K(B)를 구한다.
- **단계 3** : K(A)에 있는 각 색인키 리스트에 해당하는 색인 레코드를 액세스하여 i번째 항목이 $O_i$이고 i+1번째 항목이 $O_{i+1}$인 경로 인스턴스를 삭제함과 동시에, 각 경로 인스턴스의 첫번째 항목에서 i번째 항목까지의 부분을 PI.pre에 보관한다.
- **단계 4** : i번째 항목이 $O_i$이고 i+1번째 항목이 $O'_{i+1}$인 새로운 경로 인스턴스 집합 PI를 생성한다. 이는 PI.pre에 있는 요소들과 PI.suf에 있는 요소들을 각각 연결함으로써 얻을 수 있다.
- **단계 5** : K(B)에 있는 각 색인키 리스트에 해당하는 색인 레코드에 집합 PI에 있는 경로 인스턴스를 첨가한다.

이와같이 EPI에서는 ENI에서와는 달리 데이타베이스의 변경에 따른 색인구조의 갱신을 위한 역방향 운행이 필요 없다. 이는 경로 인스턴스가 색인 레코드에 저장되어 있기 때문이다. 따라서, EPI는 데이타베이스의 객체내에 역 참조자가 존재하지 않아도 사용할 수 있다.

3.3.3 MTI : 다중 이차원 클래스 색인구조

주어진 경로 $P = C_1.A_1.A_2...A_n$과 이 경로에 대해 구성된 n개의 MTI에서는 경로상의 임의의 클래스 계층 $C_i$에 대해 중포속성 $A_n$에 조건이 주어진 중포술어를 만족하는 객체를 탐색하기 위해서는 (n - i + 1)번의 색인 검색이 필요하다. 특히, 클래스 계층 $C_1$에 대한 중포술어를 만족하는 객체를 검색하기 위해서는 n개의 색인구조가 검색되어야 한다. 그러나, MTI에서는 데이타베이스의 변경에 따른 경로상의 순방향 및 역방향 운행이 필요없으므로 색인구조의 유지를 위한 오버헤드가 매우 적게되는 장점이 있다.

## 4. 성능 평가

먼저, 본 논문에서 제안한 각 색인구조의 검색 성능

에 대하여 살펴보면, ENI가 EPI에 비해 좋은 성능을 가진다. 이는 EPI에서는 색인 엔트리를 색인된 중포속성의 경로 인스턴스로 구성하기 때문에 색인 엔트리를 타겟 클래스 계층에 속하는 객체 식별자만으로 구성하는 ENI에 비하여 많은 저장 공간의 오버헤드 때문이다. MTI에서는 경로상의 클래스 계층마다 이차원 클래스 색인구조를 구축하여 중포속성을 표현하는 경로 길이 만큼의 단계적 색인검색이 필요하므로 가장 나쁜 검색 성능을 나타낸다. 즉, ENI가 가장 좋은 검색 성능을 가지게 되고, EPI가 그 다음으로 좋은 검색 성능을 가지게 된다.

그러나, 제3.3절에서 살펴본 바와 같이 각 색인구조의 운용에 따른 유지비용에 대한 오버헤드는 색인된 중포속성의 경로 길이에 따라 많은 차이를 보이게 된다. 따라서, 본 절에서는 각 색인구조의 운용에 따른 유지비용을 비교 평가함으로써, 경로의 길이에 따른 색인구조의 선택 기준을 제시한다. 경로 $P = C_1.A_1.A_2 ...A_n$상에 있는 객체 $O_i(1 \le i \le n)$의 속성 $A_i$의 값이 $O_{i-1}$에서 $O'_{i-1}$로 변경될 때, 경로 $P$에 대해 구축된 색인구조도 변경되어야 한다. 먼저, 이러한 데이타베이스의 변경에 따른 각 색인구조의 갱신을 위한 유지비용을 모델링하고, 각 색인구조의 유지를 위한 오버헤드를 상호 비교한다.

### 4.1 비용 모델

다음 <표 1>은 각 색인구조의 비용 모델에서 사용한 파라미터들이다.

〈표 1〉 비용 모델에서 사용한 파라미터

| 파라미터 | 의 미 |
|---|---|
| $h$ | 다차원 색인구조를 구성하는 디렉토리 트리의 높이 |
| $P$ | 색인 페이지의 크기 |
| $K_i$ | 클래스 계층 $C_i$에서 속성 $A_i$의 값으로 동일한 값을 가지는 객체의 평균 개수 |
| $D_i$ | 클래스 계층 $C_i$에서 속성 $A_i$가 가지는 유일값의 개수 |
| $LXN$ | ENI(확장된 중포 색인구조)의 색인 레코드 길이 |
| $LXP$ | EPI(확장된 경로 색인구조)의 색인 레코드 길이 |
| $LXM(i)$ | MTI(다중 이차원 클래스 색인구조)에서 i번째 색인구조의 색인 레코드 길이 |
| $CFT$ | 경로의 순방향 운행에 따른 비용 |
| $CBT$ | 경로의 역방향 운행에 따른 비용 |
| $CIU$ | 색인구조의 갱신 비용 |

#### 4.1.1 ENI : 확장된 중포 색인구조

먼저, 데이타베이스의 각 객체에는 자신을 참조하는 객체에 대한 역 참조자가 존재한다고 가정한다. 역 참조자가 제공되지 않으면 ENI는 사용할 수 없다. 제3.3절의 각 색인구조의 운용 알고리즘에서 언급한 것처럼 데이타베이스의 변경에 따라 ENI를 갱신하기 위해서는 데이타베이스내의 경로를 따라 두번의 순방향 운행과 한번의 역방향 운행이 필요하며, 한번의 색인구조 자체의 갱신이 필요하게 된다. 역방향 운행과 색인구조의 갱신은 객체 $O_{i-1}$에 대한 검색키 값이 객체 $O'_{i-1}$의 것과 다를 경우에만 필요하므로, 객체 $O_{i-1}$에 대한 검색키 값이 객체 $O'_{i-1}$의 것과 다를 확률을 $pd$라 할 때, 유지비용 $M$은

$$M = 2 \times CFT + pd \times (CBT + CIU) \tag{2}$$

이다. 순방향 운행을 위해서 액세스 해야 할 객체의 수는 $(n - i)$개이고, 각 객체를 액세스하기 위해서는 먼저 물리적 주소를 구해야 하므로,

$$CFT = 2 \times (n - i) \tag{3}$$

이다. 그리고, 역방향 운행을 위해서 액세스해야 할 객체의 개수 $No$는

$$No = \begin{cases} \sum_{j=2}^{i-1} \left( \prod_{k=j}^{i-1} K_k \right) & \text{if } i > 2 \\ 0 & otherwise \end{cases} \tag{4}$$

이다. 따라서, 각 객체에 대해서 두번의 I/O가 필요하므로,

$$CBT = \begin{cases} 2 \times No & \text{if } i > 2 \\ 0 & otherwise \end{cases} \tag{5}$$

이다. 여기서, 경로의 길이가 2이고(즉, $n = 2$), 변경된 객체가 두번째 클래스 계층이면(즉, $i = 2$), $CFT = 0$이고, $CBT = 0$이다. 그리고, 확률 $pd$는 다음과 같이 주어진다.

$$pd = \begin{cases} 1 & \text{if } i = n \\ 1 - \left( \frac{\left( \prod_{j=i+1}^{n} K_j \right) - 1}{D_i - 1} \right) & \text{if } i < n \end{cases} \tag{6}$$

색인구조의 갱신 비용은 객체 $O_i$를 직접 또는 간접으로 참조하는 타겟 클래스 계층의 객체에 대한 색인 엔트리를 변경전 색인키 리스트에 해당하는 색인 레코드에서 제거하고, 새로운 색인키 리스트에 해당하는 색인 레코드에 삽입하는 것이다. 따라서, 색인 레코드를 포함하는 색인 페이지를 찾아서 그 페이지를 읽고

쓰기 위한 비용을 $CI$라 하면 $CIU$의 평균 값은 다음과 같다.

$$CIU = CI \times (1 + pl) \quad (7)$$

여기서, $pl$은 색인 엔트리를 제거할 색인 레코드와 삽입할 색인 레코드가 서로 다른 색인 페이지에 있을 확률이다. 그리고, $CI$는 다음과 같이 계산된다.

$$CI = \begin{cases} h+2 & \text{if } LXN \le P \\ h+2+(np-1)/np & otherwise \end{cases} \quad (8)$$

여기서, $np$는 하나의 색인 레코드를 저장하기 위하여 필요한 색인 페이지의 개수이다. 그리고, 확률 $pl$은 다음과 같다.

$$pl = \begin{cases} 1 & \text{if } LXN > P \\ 1-[(NR-1)/(D_n-1)] & otherwise \end{cases} \quad (9)$$

여기서 $NR$은 색인 페이지당 색인 레코드의 개수이다.

4.1.2 EPI : 확장된 경로 색인구조

제3.3절의 각 색인구조의 운용 알고리즘에서 언급한 것처럼 데이타베이스의 변경에 따라 EPI를 갱신하기 위해서는 두번의 순방향 운행이 필요하며, 한번의 색인구조 자체의 갱신이 필요하다. ENI에서와는 달리, 객체 $O_{i+1}$과 $O'_{i+1}$에 대한 검색키 값이 서로 동일하여도 색인구조의 변경은 필요하다. 이는 색인 레코드에 저장된 경로 인스턴스의 변경을 반영하여야 하기 때문이다. 그러나, EPI에서는 역방향 운행이 필요없다. 따라서, 유지비용 $M$은

$$M = 2 \times CFT + CIU \quad (10)$$

이다. 여기서, $CFT$는 ENI에서와 마찬가지이지만, $CIU$는 다르게 된다. 즉, EPI에서는 객체 $O_{i+1}$과 $O'_{i+1}$에 대한 검색키 값이 서로 다를뿐만 아니라 이들에 대한 색인 레코드가 서로 다른 색인 페이지에 있을 경우에만 두 개의 색인 페이지가 갱신된다. 따라서,

$$CIU = CI \times (1 + pd \times pl) \quad (11)$$

이다. 여기서, $CI$는 식 (8)에서 $LXN$을 $LXP$로 대치한 것과 같다.

4.1.3 MTI : 다중 이차원 클래스 색인구조

경로상에서 $i$ 번째 클래스 계층에 있는 임의의 객체 $O_i$의 속성 $A_i$의 값이 $O_{i+1}$에서 $O'_{i+1}$로 변경된 경우에 MTI에서는 $i$ 번째 색인구조만 갱신하면 된다. 따라서,

$$M = CIU = CI \times (1 + pl) \quad (12)$$

이다. 여기서, $CI$는 식 (8)에서 $LXN$을 $LXM(i)$로 대치한 것과 같으며, $pl$은 식 (9)에서 $D_n$을 $D_i$로 대치한 것과 같다.

4.2 비교 평가

본 절에서는 색인구조를 구축할 경로의 길이에 따른 각 색인구조의 운용에 필요한 유지비용을 파라미터 $K_i$의 변화에 따라 비교 평가한다. 각 색인구조의 유지비용은 객체의 참조 공유도 $K_i$에 의해서 크게 영향을 받는다. 그 이유는 첫째로 색인 레코드의 크기가 $K_i$값들의 곱에 비례하기 때문에 색인 레코드의 크기가 페이지의 크기보다 크게 되면 각 색인구조 자체의 갱신 비용에 영향을 미치기 때문이며, 둘째로 ENI에서는 $K_i$가 역방향 운행에서 액세스해야 할 객체의 개수를 결정하기 때문이다. 그리고, 객체 식별자 Oid의 크기는 12바이트로 하고, 색인 페이지의 크기 $P$는 4K바이트로 한다. 이는 색인구조의 일반적인 구현에서 널리 사용하는 파라미터 값이다[2, 4, 6, 9].

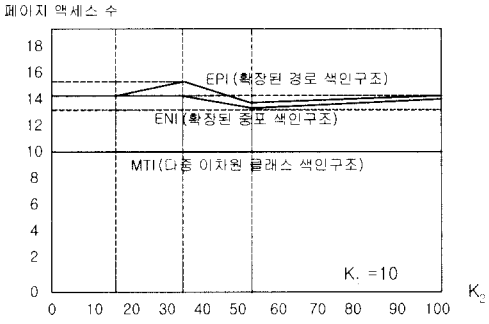
4.2.1 경로의 길이가 2인 경우

다음은 색인구조를 구축할 경로의 길이가 2인 경우에 데이타베이스의 변경에 따른 각 색인구조의 갱신을 위한 유지비용에 대한 분석이다. 첫째로, 데이타베이스의 변경이 첫 번째 클래스 계층 $C_1$에서 발생할 경우이다. (그림 6)은 $K_1 = 10$일 때, $K_2$ 값의 변화에 따른 각 색인구조의 갱신을 위한 유지비용의 변화를 나타낸다.

데이타베이스의 변경이 클래스 계층 $C_1$에서 발생하면, ENI에서도 갱신을 위한 역방향 운행에 대한 오버헤드가 필요없다. 따라서, ENI와 EPI의 두 색인구조에서 색인 레코드의 오버플로우가 없게되는 $K_1 \times K_2 < 167$인 경우에는 유지비용 $M = 2 \times CFT + pd \times (CBT + CIU) = 4 + CIU = 4 + CI \times (1 + pl) = 14$로서 일정하다.

두 객체 참여도의 곱이 $167 < K_1 \times K_2 < 334$인 경우에는 EPI의 색인 레코드에서 오버플로우가 발생하므로 유지비용이 1만큼 증가한다. 그러나, $334 < K_1 \times K_2 < 500$인 경우에는 두 색인구조의 유지비용은 감소한다. 이것은 색인 레코드의 개수가 감소함에 따른 디렉토리의 높이 $h$가 감소하기 때문이다. $K_1 \times K_2 = 500$인 경우에는 하나의 색인 레코드를 저장하기 위하여 ENI에서

는 두개의 색인 페이지가 필요하며, EPI에서는 세 개의 색인 페이지가 필요하다. 따라서, 추가적인 페이지 액세스의 확률은 ENI에서는 0.5가 되며, EPI에서는 0.7이 된다(참고 : $CI$를 위한 식 (8)). 따라서, ENI에서는 $CI = 4.5$가 되므로 $M = 13$이고, EPI에서는 $CI = 4.7$이 되므로 $M = 13.4$이다.




(그림 6) 경로 길이가 2인 경로상의 첫 번째 클래스 계층에서 데이타베이스의 변경이 발생한 경우 각 색인구조의 갱신을 위한 유지비용의 변화

그러나, 레코드의 크기가 세 개의 페이지보다 크게 되면, 추가적인 페이지 액세스의 확률이 증가하므로, 디렉토리 높이의 감소에 따른 디렉토리 페이지 액세스의 감소에 대한 이득을 상쇄하게 된다. 따라서, $500 < K_1 \times K_2$인 경우에는 두 색인구조에서 모두 유지비용이 14에 가깝게 됨을 알 수 있다. 그리고, MTI에서는 첫 번째 클래스 계층에 대한 이차원 클래스 색인구조만 변경하며, 색인 레코드의 크기는 $K_1$에만 관계하므로 유지비용 $M = 10$으로서 일정하게 된다.

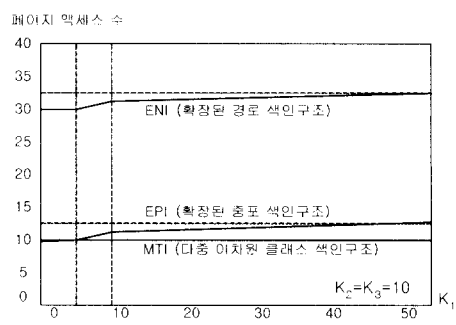
둘째로, 데이타베이스의 변경이 두 번째 클래스 계층 $C_2$에서 발생할 경우에는, 역방향 운행뿐만 아니라 순방향 운행도 필요 없게된다. 따라서, 객체 참조의 공유도가 매우 낮으면(즉, $K_1 = K_2 = 1$에 가까우면), 세 가지 색인구조 모두 유지비용이 같게 된다. 객체 참조의 공유도가 증가하면, ENI와 EPI의 유지비용은 데이타베이스의 변경이 클래스 계층 $C_1$에서 발생할 경우와 같은 경향을 보인다. 즉, 일정한 비용을 어느 정도 유지하다가 조금 감소한 후 점점 증가하게 된다.

경로의 길이가 2이고 역 참조자가 데이타베이스의 객체 내에서 제공될 경우에는 ENI와 MTI의 유지비용은 같게 된다. 그 이유로서, 먼저 경로상의 두번째 클래스 계층에 있는 객체 $O_2$에서 색인된 속성 $A_2$가 변경되면 객체 $O_2$내에 역 참조자가 존재하기 때문에 객체 $O_2$를 참조하는 첫번째 계층의 객체를 결정하기 위한 역방향 운행이 필요 없기 때문이다.

그리고, 데이타베이스의 변경이 첫 번째 클래스 계층 $C_1$에서 발생할 경우, ENI와 EPI의 갱신을 위해 필요한 순방향 운행에 대한 오버헤드는 색인구조와 무관하게 데이타베이스 자체의 변경에서도 필요하다. 즉, 경로상의 첫 번째 클래스 계층에 있는 객체 $O_1$에서 속성 $A_1$의 값이 두 번째 클래스 계층의 다른 객체를 참조하게 변경될 경우, ENI와 EPI에서는 색인키 값을 얻기 위하여 $A_1$의 변경전의 값과 변경후의 값에 대한 객체 $O'$와 $O''$를 액세스하는 두 번의 순방향 운행이 필요하다. 그러나, 이 두 객체의 액세스는 $O_1$에 대한 역 참조자의 값을 변경하기 위하여 색인구조와 무관하게 반드시 액세스해야 한다. 이것은 MTI를 이용하는 경우에도 피할 수 없는 오버헤드이다.

4.2.2 경로의 길이가 3인 경우

경로 길이가 2보다 크게되면, ENI에서는 역 방향 운행이 필요하게 되므로 EPI에 비해 유지비용이 증가하게 된다. (그림 7)은 데이타베이스의 변경이 세 번째 클래스 계층 $C_3$에서 발생할 경우, $K_2 = K_3 = 10$이라 가정하고 $K_1$의 변화에 따른 각 색인구조의 갱신을 위한 유지비용의 변화를 나타낸 것이다. ENI의 유지비용을 지배하는 것은 역방향 운행에 의한 것으로, 역방향 운행에 필요한 객체의 액세스 개수는 $K_2$개이다(참고 : $CBT$를 위한 식 (5)). 예를들어, $K_2 = K_3 = 10$인 경우 열 개의 객체가 반드시 액세스되어야 하며, 각 객체에 대해 두 번의 페이지 액세스가 필요하므로 역방향 운행에 필요한 비용은 20번의 페이지 액세스가 된다.




(그림 7) 경로 길이가 3인 경로상의 세 번째 클래스 계층에서 데이타베이스의 변경이 발생한 경우 각 색인구조의 갱신을 위한 유지비용의 변화

(그림 7)에서는 ENI의 유지비용이 $K_1$의 증가에 따라 증가함을 나타낸다. 이것은 색인 레코드의 크기가 페이지의 크기보다 커지기 때문에 추가적인 페이지 액세스의 가능성이 높아지기 때문이다. 이러한 현상은 EPI에서도 마찬가지이나, EPI에서는 역방향 운행이 필요 없기 때문에 ENI에서보다 유지비용이 매우 적게 된다.

한편, 데이타베이스의 변경이 첫 번째 클래스 계층 $C_1$과 두 번째 클래스 계층 $C_2$에서 발생할 경우에는, ENI에서도 역방향 운행이 필요 없게되므로 색인구조의 유지비용이 EPI에서와 같게 된다.

### 4.3 성능분석 결과

첫째, 색인구조를 구축할 중포속성을 표현하는 경로의 길이가 2인 경우에는, ENI가 가장 적합한 색인구조이다. 이것은 제4.2절의 경로의 길이가 2인 경우의 유지비용에 대한 분석에서 살펴본 바와 같이, 유지비용은 세 가지 색인구조 모두 비슷한 반면에 ENI에서 검색 성능이 가장 좋기 때문이다.

둘째, 색인구조를 구축할 중포속성을 표현하는 경로의 길이가 3인 경우에는, 일반적으로 EPI가 가장 적합한 색인구조이다. 이것은 EPI의 검색 성능은 ENI에 필적하는 반면에 데이타베이스 변경에 의한 유지비용이 적게 되기 때문이다. 그러나, 데이타베이스의 변경이 첫 번째와 두 번째 클래스 계층에서 주로 발생하고, 각 클래스 계층의 객체 참조 공유도의 곱인 $\Pi_{i=1}^{3} K_i$의 값이 크게 되면 ENI가 가장 적합한 색인구조이다. 이것은 제4.2절의 경로의 길이가 3인 경우의 유지비용에 대한 분석에서 살펴본 바와 같이, 데이타베이스의 변경이 첫 번째와 두 번째 클래스 계층에서 발생할 경우에는 ENI에서도 갱신을 위한 역방향 운행이 필요 없게되므로 색인구조의 유지비용이 EPI에서와 같게 되기 때문이다. 그리고, $\Pi_{i=1}^{3} K_i$의 값이 크게 되면, EPI에서는 색인 레코드가 여러 개의 페이지를 점유하게 되어 증가되는 검색비용이 ENI에 대한 유지비용에서의 이득을 상쇄할 수 있기 때문이다.

셋째, 색인구조를 구축할 중포속성을 표현하는 경로의 길이가 4이상일 경우에는, 경로를 길이가 1, 2 또는 3이 되는 서브경로들로 분할한 다음에, 각 서브경로에 따라 ENI 또는 EPI를 할당하는 정책을 사용하는 것이 좋다. 이것은 객체 참조 공유도 $K_i$가 매우 낮은 경우(거의 1일 때)가 아니면 ENI와 EPI 모두 사용하기가 어렵기 때문이다. 경로의 길이가 4이상인 경우, ENI에서는 역방향 운행에 의해 매우 높은 유지비용을 가지며, 이 비용은 참조 공유도의 정도에 비례하여 증가한다. 따라서, ENI는 데이타베이스의 변경이 거의 발생하지 않거나, 객체 참조 공유도가 매우 낮은 경우가 아니면 사용하기가 어렵게 된다. 그리고, EPI에서는 경로 인스턴스를 유지하기 때문에 참조 공유도 $K_i$가 증가함에 따라 색인 레코드가 커져서 여러 개의 페이지를 점유하게 되어 검색비용이 증가하게 된다. 따라서, EPI도 참조 공유도가 매우 낮은 경우가 아니면 사용하기가 어렵게 된다.

## 5. 결 론

본 논문에서는 중포속성의 키 속성 도메인과 함께 중포속성의 경로상에 나타나는 각 속성마다 한 축의 클래스 식별자 도메인을 할당하여 구성된 다차원 도메인 공간상의 색인 엔트리들을 다루는 다차원 중포속성 색인기법을 제안하였다. 제안된 색인기법은 기존의 일차원 색인구조를 이용한 중포속성 색인기법에서 지원할 수 없는 질의의 대상 범위가 클래스 계층상의 임의의 클래스들로 제한되거나, 질의에 포함된 복합속성들의 도메인이 클래스 계층상의 임의의 클래스들로 제한되는 중포술어로 구성된 질의의 경우에도 매우 효율적으로 지원할 수 있다.

또한, 본 논문에서 제안한 다차원 중포속성 색인기법의 성능평가를 위하여, 색인 엔트리를 타겟 클래스 계층의 객체 식별자들로 구성하는 확장된 중포 색인구조(ENI)와 색인 엔트리를 색인된 중포속성에 대한 경로 인스턴스들로 구성하는 확장된 경로 색인구조(EPI)를 제안하고, 각 색인구조의 운용에 따른 유지비용을 비교 평가하였다. 평가 결과로서 경로의 길이가 2인 경우에는 ENI를, 경로의 길이가 3인 경우에는 EPI를 구축하는 것이 일반적인 경우에 합당한 것으로 나타났다. 그리고, 경로의 길이가 4이상일 경우에는 경로의 길이에 따라 증가하는 색인구조의 유지비용으로 인하여, 경로의 길이가 1, 2, 또는 3이 되는 서브경로로 나누어서 각 서브경로별로 적합한 색인구조를 할당하여야 함을 알 수 있었다. 그러나, 이러한 길이가 4이상인 경로에 대한 각 색인구조의 할당 문제는 데이타베이스의 특성, 작업부하(workload), 그리고 각 색인구조의 검색비용과 유지비용 등을 고려하는 매우 복잡한 문제로서 앞으로 더 연구해야 할 연구과제이다.

## 참 고 문 헌


[1] Atkinson, M. et al., "The Object-Oriented Database System Menifesto," In *Proc. Intl. Conf. on Deductive and Object-Oriented Databases*, pp.40-57, Kyoto, Japan, Dec. 1989.

[2] Bertino, E. and Kim, W., "Indexing Techniques for Queries on Nested Objects," *IEEE Trans. on Knowledge and Data Eng.*, Vol.1, No.2, pp.196-214, June 1989.

[3] Bertino, E. et al., "Object-Oriented Query Languages : The Notion and the Issues," *IEEE Trans. on Knowledge and Data Engineering*, Vol.1, No.3, pp.223-237, June 1992.

[4] Bertino, E. and Foscoli, P., "Index Organizations for Object-Oriented Database Systems," *IEEE Trans. on Knowledge and Data Eng.*, Vol.7, No.2, pp.193-209, April 1995.

[5] Bertino, E. and Ooi, B. C., "The Indispensability of Dispensable Indexes," *IEEE Trans. on Knowledge and Data Eng.*, Vol.11, No.1, pp.17-27, Jan. 1999.

[6] Kemper, A. and Moerkotte, G., "Access Support Relations : An Indexing Method for Object Bases," *Information Systems*, Vol.17, No.2, pp.117-145, 1992.

[7] Kifer, M., Kim, W., and Sagiv, Y., "Querying Object-Oriented Databases," In *Proc. Intl. Conf. on Management of Data*, ACM SIGMOD, San Diego, Calif., pp.393-402, May 1992.

[8] Kim, K. C. et al., "Acyclic Query Processing in Object-Oriented Databases," In *Proc. Intl. Conf. on Entity-Relationship Approach*, Rome, Italy, pp. 329-346, Nov. 1989.

[9] Kim, W. et al., "Indexing Techniques for Object-Oriented Databases," *Object-Oriented Concepts, Databases, and Applications*, (Kim, W. and Lochovsky, F.eds.), Addison Wesley, 1989.

[10] Kim, W., "A Model of Queries for Object-Oriented Databases," In *Proc. Intl. Conf. on Very Large Data Bases*, pp.423-432, Amsterdam, Aug. 1989.

[11] Kim,W., *Introduction to Object-Oriented Databases*, The MIT Press, 1990.

[12] Lee, J. H. et al., "A Region Splitting Strategy for Physical Database Design for Multidimensional File Organizations," In *Proc. Intl. Conf. on Very Large Data Bases*, pp.416-425, Athens, Greece, Aug. 1997.

[13] 이종학, 황규영, "다차원 색인구조를 이용한 객체지향 데이타베이스의 조율 가능한 클래스 계층 색인기법", *한국정보과학회 논문지(B)*, 제26권 제3호, pp.365-379, 1999년 3월.

[14] Whang, K. Y. and Krishnamurthy, R., *Multilevel Grid Files*, IBM Research Report RC 11516, IBM Thomas J. Watson Research Center, Nov. 1985.

[15] Whang, K. Y. and Krishnamurthy, R., "The Multilevel Grid File-A Dynamic Hierarchical Multidimensional File Structure," In *Proc. Intl. Conf. on Database Systems for Advanced Applications (DASFAA)*, pp.449-459, Tokyo, Apr. 1991.

[16] Xie, Z. and Han, J., "Join Index Hierarchies for Supporting Efficient Navigations in Object-Oriented Databases," In *Proc. Intl. Conf. on Very Large Data Bases* pp. 522-533, Santiago, Chile, Sept. 1994.



**이 종 학**

e-mail : jhlee11@cuth.cataegu.ac.kr

1982년 경북대학교 전자공학과 (전자계산 전공) 졸업 (학사)

1984년 한국과학기술원 전산학과 졸업(공학석사)

1997년 한국과학기술원 전산학과 졸업(공학박사)

1991년 정보처리기술사

1984년~1987년 금성통신(주) 부설연구소 주임연구원

1987년~1998년 한국통신 연구개발본부 선임연구원

1998년~현재 대구효성가톨릭대학교 컴퓨터정보통신공학부 조교수

관심분야 : 데이타베이스 시스템, 객체지향 데이타베이스, 트랜잭션 프로세싱, 지리정보 시스템 등