Journal of Korea Multimedia Society Vol. 7, No. 7, July 2004(pp. 859-876)

객체지향 질의의 효율적 처리를 위한 다차원 경로 색인구조의 최적 구성방법

이 종 학†

요 약

본 논문에서는 객체 데이타베이스에서 객체지향 질의의 효율적 처리를 위한 다차원 경로 색인구조(Multidimensional Path Indexes: MPIs)의 최적 구성방법을 제시한다. MPI는 중포속성과 여러 클래스 계층이 포함된 중포술어를 효율적으로 지원하기 위하여 다차원 색인구조를 이용한다. B^+-tree와 같은 일차원 색인구조를 이용한 중포속성 색인구조로서는 이와 같은 술어를 잘 지원할 수 없다. 본 논문에서는 타겟 클래스 또는 도메인 클래스의 대치가 있는 경로식으로 표현된 여러 중포술어들의 접속으로 구성된 복합질의의 관점에서 MPI 색인들의 구성에 관하여 분석한다. 먼저, 데이타베이스의 변경에 따른 MPI 색인구조의 운용과 하나의 중포술어를 가지는 질의의 경우에 대한 MPI 색인의 사용에 대하여 색인구성을 분석한다. 그리고 겹침 경로와 비겹침 경로 상에 주어지는 여러 개의 중포술어들로 구성된 보다 일반적인 질의의 관점에서 MPI 색인의 구성에 관하여 분석한다. 겹침 경로는 경로들 사이에 공통의 부경로가 있는 것이고, 비겹침 경로는 공통의 부경로가 없는 것이다.

Optimal Configurations of Multidimensional Path Indexes for the Efficient Execution of Object-Oriented Queries

Jong-Hak Lee†

ABSTRACT

This paper presents optimal configurations of multidimensional path indexes (MPIs) for the efficient execution of object-oriented queries in object databases. MPI uses a multidimensional index structure for efficiently supporting nested predicates that involve both nested attribute and class hierarchies, which are not supported by the nested attribute index using one-dimensional index structure such as B^+-tree. In this paper, we have analyzed the MPIs in the framework of complex queries, containing conjunctions of nested predicates, each one involving a path expression having target classes and domain classes substitution. First of all, we have considered MPI operations caused by updating of object databases, and the use of the MPI in the case of a query containing a single nested predicate. And then, we have considered the use of the MPIs in the framework of more general queries containing nested predicates over both overlapping and non-overlapping paths. The former are paths having common subpaths, while the latter have no common subpaths.

Key words: Object Database(객체 데이타베이스), Query Processing(질의 처리), Nested Attribute(중포속성), Index Configuration(색인 구성)

※ 교신저자(Corresponding Author) : 이종학, 주소 : 경북 경산시 하양읍 금락1리 330(712-702), 전화 : 053)850-2746, FAX : 053)850-2740, E-mail : jhlee11@cu.ac.kr

접수일 : 2003년 11월 4일, 완료일 : 2003년 12월 12일

† 정회원, 대구가톨릭대학교 공과대학 컴퓨터정보통신공학부 교수

※ 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음.(KRF-2002-003-D00276)

1. 서 론

객체 데이타베이스 관리 시스템(Object Database Management System: ODBMS)의 질의처리 성능 최적화는 중요한 연구과제이다. 최근에 제안된 객체 데이타베이스 색인기법들은 객체지향 질의처리의

성능 향상에 크게 기여하고 있다[3]. 그러나, 이들 색인구조들은 기존의 관계형 데이타베이스의 단순속성에 대한 색인구조에 비해 저장공간 및 갱신유지 비용에 큰 부담이 있다. 또한, 색인구조의 종류에 따라 검색 성능이 다른 특성도 있다[3]. 그러므로, 객체 데이타베이스 색인기법을 통한 질의처리의 이점이 색인구조의 저장공간 및 갱신유지를 위한 부담으로 인해 상쇄되지 않기 위해서는 색인들을 매우 신중히 구성하여야 하며, 효과적인 색인구성 방법에 관한 연구가 필수적이라 할 수 있다.

객체 데이타베이스는 클래스 집단화(aggregation) 개념에 의하여 한 클래스가 가지는 속성의 도메인이 또 다른 클래스가 될 수 있도록 함으로써(이러한 속성을 *복합 속성*이라 함) 클래스들 사이에 클래스 집단화 계층을 이룬다. 따라서, 클래스 집단화 계층을 이루는 클래스들에서 정의된 어떠한 속성도 논리적으로는 루트 클래스의 속성이라고 볼 수 있다. 우리는 클래스 집단화 계층에서 루트 클래스가 아닌 클래스에서 정의된 속성을 루트 클래스의 *중포속성*(*nested attribute*)[1,8]이라 한다. 이로 인하여 객체지향 질의어에서는 기존의 관계형 질의어와는 달리 중포속성에 조건이 주어지는 *중포술어*(*nested predicate*)[1,8]를 가진다는 특징이 있다. 객체지향 질의어에서는 중포속성을 표현하기 위해서 *경로식*(*path expression*)[4]을 사용한다. 경로식은 루트 클래스로부터 클래스 집단화 계층을 따라 나타나는 속성들의 나열로 표현한다.

객체 데이타베이스는 또 하나의 중요한 개념인 클래스 상속(inheritance) 개념에 의하여 클래스들 사이에 클래스 상속 계층(앞으로 클래스 계층이라 약칭함)이라는 하나의 또 다른 계층구조를 이룬다. 즉, 하나의 클래스는 여러 개의 서브클래스를 가지며, 각 서브클래스는 또 다른 여러 서브클래스들을 가진다. 따라서, 객체 데이타베이스에서는 하나의 질의에 대한 대상 범위를 두 가지 경우로 해석할 수 있다. 한 경우는 질의의 대상 범위를 질의에 나타나는 클래스만으로 한정하는 것이고, 또 다른 경우는 질의의 대상 범위를 질의에 나타나는 클래스와 그의 모든 서브클래스들을 포함하는 것이다[8]. 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.

중포속성을 표현하는 경로식에는 속성값(객체 참조자: OID)들의 참조관계에 의한 객체와 객체 사이에 *암시적 조인*(*implicit join*)[8]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, 질의처리 시 이를 이용하여 성능 향상을 꾀할 수 있으며 이를 중포속성에 대한 색인기법[1,2,4,15]이라 한다. 그러나, 이러한 중포속성에 대한 기존의 색인기법들은 B^+-tree와 같은 일차원 색인구조를 이용함으로써, 클래스 상속의 특징으로 인한 질의의 대상 범위가 클래스 계층상의 임의의 클래스들로 제한되거나, 경로식에 나타나는 속성의 도메인이 클래스 계층상의 임의의 클래스들로 제한이 되는 질의들을 지원하기 어려운 문제점을 가지고 있다.

이와 같은 일차원 색인구조를 이용하는 기존의 중포속성에 대한 색인기법들이 가지는 문제점을 해결하기 위하여, 다차원 색인구조를 중포속성에 대한 색인구조로 이용할 수 있으며 이를 *다차원 중포속성 색인구조*[11]라 한다. 다차원 중포속성 색인구조에서는 중포속성의 값들을 키로 하는 *키 속성 도메인*과 함께, 경로식에 나타나는 루트 클래스의 클래스 계층과 각 복합 속성의 도메인 클래스 계층을 이루는 클래스들의 *클래스 식별자 도메인*을 할당하여 다차원 색인구조를 구성한다. 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.

다차원 중포속성 색인구조는 다차원 도메인 공간상의 색인 엔트리들의 클러스터링 정도를 질의 패턴에 따라 조정함으로써 주어진 질의에 의해서 액세스되는 색인 페이지의 개수를 최소로 할 수 있는 색인구조의 최적 설계기법[11]을 적용하여 구성할 수 있다. 본 논문에서는 먼저, 이러한 다차원 중포속성 색인구조의 두 가지 형태인 다차원 중포 색인구조와 다차원 경로 색인구조에 대해서 데이타베이스 변경

에 따른 유지비용 면에서 각각의 한계점을 제시한다. 그리고, 이를 바탕으로 비교적 유지보수가 쉬운 다차원 경로 색인구조에 대해서 두 개 이상의 경로상에 각각 주어진 술어들로 구성된 복합질의의 경우, 주어진 경로 스키마상에서 다차원 경로 색인구조를 구성할 수 있는 다양한 방법을 제시하고, 각 구성방법에 따른 질의처리의 성능을 비교분석 한다.

본 논문의 구성은 다음과 같다. 먼저, 제 2절에서는 관련 연구로서 객체 데이타베이스의 색인 구축에 필요한 기본 개념으로 객체지향 질의어와 질의처리 전략 및 기존의 색인 기법들을 살펴본다. 그리고, 제 3절에서는 객체 데이타베이스의 중포속성에 대한 색인기법으로 다차원 색인구조를 이용하는 다차원 중포속성 색인구조들을 소개하고, 각 색인구조의 운용기법을 제시한다. 그리고, 제 4절에서는 다차원 중포속성 색인구조의 대표적인 다차원 경로 색인구조의 최적 구성을 위한 다양한 색인 구성과 그에 따른 질의처리 전략을 제시하고 각각에 대한 비교분석 결과를 제시한다. 마지막으로, 제 5절에서는 결론을 내린다.

2. 관련 연구

객체 데이타베이스에서 사용자가 정의하는 클래스들은 루트를 가진 이차원 방향성 그래프(rooted two-dimensional directed graph)를 형성하며, 이 그래프를 그 클래스에 대한 *스키마 그래프*(*Schema Graph: SG*)라 정의한다[8]. 그림 1은 클래스 Person에 대한 SG이다. 그림 1에서 클래스 Person은 서브클래스 Instructor와 Student, 그리고 Instructor와 Student의 서브 클래스들을 포함하는 클래스 상속

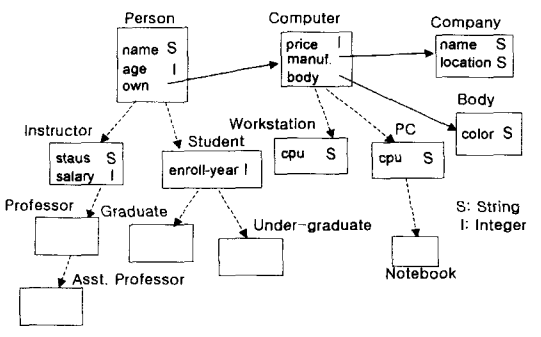


그림 1. 클래스 Person에 대한 스키마 그래프

계층구조와 Computer과 Company를 포함하는 클래스 집단화 계층구조의 루트이다. 속성 own의 도메인은 Computer과 그의 서브 클래스 Workstation, PC 및 Notebook들이고, 속성 manuf.의 도메인은 Company이다.

객체 데이타베이스에서 한 클래스가 가지는 속성들은 정수 타입이나 문자 타입과 같은 기본 타입의 값을 가지는 *단순속성*과 다른 클래스에 속하는 객체의 객체 식별자(Object IDentifier: OID)를 값으로 가지는 *복합 속성*으로 구분된다[8]. 클래스 C의 *중포속성*[1]이란 클래스 C로부터 실선 링크로 연결된 클래스에서 정의된 속성들을 의미하며, 클래스 C와 속성들의 연속으로 표시한다. 예를 들어 그림 1에서 클래스 Company에 정의된 속성 name은 클래스 Person과 Computer의 중포속성이며, 각각 Person.own.manuf.name과 Computer.manuf.name으로 표시한다. 클래스 집합 C^*는 클래스 C와 그의 모든 서브 클래스들을 원소로 하는 집합으로 정의한다. 예를 들어 그림 1에서 클래스 Student*는 집합 {Student, Graduate, Under-graduate}이다.

2.1 객체지향 질의어

객체지향 질의어도 관계형 데이타베이스에서 널리 쓰이는 SQL(Structured Query Laguage)에서처럼 Select, From, Where 절로 구성할 수 있으며, 각 절에서 객체지향 개념을 지원하도록 확장하여 사용한다[5]. From 절에는 질의의 대상이 되는 클래스를 기술하며, Where 절에는 단순속성에 대한 조건인 *단순술어*와 더불어 중포속성에 대한 조건인 *중포술어*[1]를 사용할 수 있다.

객체지향 질의어에서 중포속성을 표현하는 방법으로 *경로식*[4,5]을 사용한다. 경로식은 클래스 집단화 계층상의 클래스 이름과 속성의 교차적인 나열로서 다음과 같은 형태를 가진다. 여기서, A_i 뒤의 중괄호({ })는 생략 가능함을 나타낸다.

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

경로식 P에서 클래스 C_1은 질의의 대상이 되는 *타겟 클래스*를 나타내고, $C_{i+1}(1 \le i \le n)$은 속성 A_i의 *도메인 클래스*를 나타낸다. 타겟 클래스와 도메인 클래스는 클래스 대치(substituability) 개념에 의해 클래스 계층상의 특정 클래스들로 한정될 수 있다[5,8].

식(1)의 경로 P에서 속성 A_i의 도메인은 뒤에 대괄호([])가 생략되면 자동으로 클래스 계층상의 클래스 집합 C^*_{i-1}로 되지만, C^*_{i+1}에 속하는 특정 클래스가 대괄호 안에 지정되면 대괄호 안의 클래스로 도메인이 한정된다.

경로식에서 타겟 클래스가 특정 클래스로 한정되는 것을 *타겟 대치(target substitution)*라 하고, 속성의 도메인 클래스가 특정 클래스로 한정되는 것을 *도메인 대치(domain substitution)*라 한다. 이러한 클래스 대치는 질의의 범위를 특정한 클래스로 한정할 수 있도록 하여 클래스 상속의 개념을 객체지향 질의어에 표현하도록 한 것이다[5].

경로식 P에서 *경로 인스턴스(path instance)*는 다음 조건을 만족하는 객체들의 리스트 (O_1, O_2,..., O_{n-1})로 정의한다[1,5]. (1) 객체 O_1은 클래스 C_1의 객체이다. (2) 객체 O_i ($1<i<= n+1$)는 클래스 C_i의 객체로서 객체 O_{i-1}의 속성 A_{i-1}의 값이다. 예를 들어, 경로식Person.own[Workstation].manuf.name에서 객체 p(∈Person)가 속성 own의 값으로 객체 w(∈Workstation)의 OID를 가지고, 객체 w에서 속성 manuf.의 값으로 객체 c(∈Company)의 OID를 가지며, 객체 c에서 속성 name의 값으로 LG를 가진다면, 객체들의 나열인 (p, w, c, LG)는 경로식 Person.own[Workstation].manuf.name에 대한 경로 인스턴스가 된다.

2.2 질의처리 전략 및 기존 색인기법

객체 데이타베이스의 중포술어를 가지는 질의를 처리하기 위한 스키마 그래프의 운행 전략으로 순방향 운행법(Forward Traversal: FT) 전략과 역방향 운행법(Backward Traversal: BT) 전략이 있다[6,9]. FT 전략은 스키마 그래프의 집단화 계층구조에서 상위 클래스를 먼저 탐색하고 하위 클래스를 나중에 탐색하는 방법으로 깊이-우선 순서로 집단화 계층구조를 운행한다. 그리고, BT 전략은 하위 클래스를 먼저 탐색하고 상위 클래스를 나중에 탐색하는 방법으로 집단화 계층구조를 아래에서 위로 운행한다. 역방향 운행시에는 하위 클래스의 객체를 참조하는 상위 클래스의 객체를 탐색하기 위해서는 많은 비용을 필요로 한다.

그리고, 방문한 클래스의 객체들을 검색하는 검색 전략으로 네스티드 루프(Nested Loop: NP) 전략과 소트 도메인(Sort Domain: SD) 전략이 있다[6,9]. NP 전략은 클래스를 구성하는 각 객체를 각각 따로 탐색하는 방법이다. 즉, 한 클래스에 단순술어가 주어지면, 그 클래스의 각 객체별로 술어를 만족하는지 검사하고, 만족하게 되면 그 객체는 하위 클래스(순방향 운행시) 또는 상위 클래스(역방향 운행시)로 보내진다. 그리고, SD 전략은 클래스를 구성하는 모든 객체들을 한꺼번에 탐색하는 방법이다. 즉, 한 클래스에 단순술어가 주어지면, 먼저 그 클래스의 모든 객체들에 대해서 한꺼번에 술어를 만족하는지 검사하고, 만족하는 모든 객체들을 한꺼번에 하위 클래스(순방향 운행시) 또는 상위 클래스(역방향 운행시)로 보낸다.

기본적인 검색 전략과 함께 그래프 운행 전략들을 결합함으로써, 네 가지 기본적인 질의처리 전략을 얻을 수 있다. 길이 n인 경로의 루트로부터 시작하는 부경로 $p = C_1.A_1....A_m (1 \le m \le n)$에 대해서 네 가지 질의처리 전략들을 설명하면 다음과 같다.

- **FTNL**(Forward Traversal with Nested Loop): 이 전략은 다음과 같은 절차를 따른다. 먼저, 클래스 C_1의 객체를 검색하여 속성 A_1의 값을 결정한다. 이 값은 클래스 C_2의 객체 식별자이다. 그리고, 클래스 C_2의 객체를 검색하기 위해 두 번의 액세스가 필요하다. 첫 번째 액세스는 A_1의 값으로 객체의 주소를 얻기 위한 시스템 테이블의 액세스이고, 두 번째 액세스는 객체의 주소에 의한 객체의 액세스로서 속성 A_2의 값을 검색한다. 이를 속성 A_m에 도달할 때까지 반복하여 술어를 평가한다. 이와 같은 과정을 클래스 C_1의 각 객체에 대해서 반복한다.
- **FTSD**(Forward Traversal with Sort Domain): 이 전략은 경로를 따라서 각 클래스의 모든 객체들이 동시에 방문된다는 점에서 이전 전략과 다른 점이다. 필요한 속성들의 모든 값들을 한꺼번에 검색하여, 이들 속성 값에 의한 객체들의 주소를 정렬시킨다. 이는 경로상의 다음 클래스의 객체들을 두 번 이상 방문하지 않기 위해서다.
- **RTNL**(Reverse Traversal with Nested Loop): 이 전략은 FTNL과 비슷하다. 단지 경로를 따라서 방문되는 첫 번째 클래스가 C_m일 뿐이다. 먼저, C_m의 객체 O를 검색하여 술어를 평가한

다. 그리고, 만약 그 객체가 술어를 만족하면, 속성 A_{m-1}의 값으로 O를 가지는 객체를 찾기 위해서 클래스 C_{m-1}을 검색한다. 이 과정을 C_1에 도달할 때까지 반복한다. 만약 모든 객체에 역 참조자(reverse reference)가 저장되어 있으면, 각 클래스에 대한 검색이 필요 없다.

- **RTSD**(Reverse Traversal with Sort Domain): 이 전략은 검색이 경로의 마지막 클래스의 객체들로부터 시작한다는 점에서 RTNL과 유사하다. 그리고, 모든 객체들의 검색을 FTSD의 경우와 같이 모두 한꺼번에 처리한다.

객체 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 복합 속성이면 역방향 운행시 상위 클래스의 객체를 탐색하기 위한 비용을 줄일 수 있다. 따라서, 객체 데이타베이스에서 색인의 키 값은 단순속성의 도메인이 되는 기본 타입의 값들뿐만 아니라 복합 속성의 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[3,8].

지금까지 중포속성에 대한 색인기법을 위해 제안된 색인구조들로, Bertino 와 Kim이 제안한 중포 색인(nested index)[1], 경로 색인(path index)[1], Kemper와 Moerkotte가 제안한 ASR(Access Support Relation)[4], 그리고 Xie와 Han이 제안한 JIH(Join Index Hierarchy)[15] 등이 있다. 임의의 경로 $p = C_1.A_1....A_n$에 대하여, *중포 색인*은 A_n을 키로 하는 킷값과 C_1에 속하는 객체들의 OID를 색인 엔트리로 구성하는 색인구조이고, *경로 색인*은 A_n을 키로 하는 킷값과 $C_1,...,C_n$에 속하는 객체들의 OID 리스트(즉, 경로 인스턴스)를 색인 엔트리로 구성하는 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 색인구조이다. 경로상에서 서로 인접한 두 개의 클래스 사이에 존재하는 직접 참조관계가 있는 OID들의 쌍들을 기본 조인 색인구조라 하고, 경로 상에서 서로 인접하지 않는 두 개의 클래스 사이의 간접 참조관계가 있는 OID들의 쌍들은 기본 조인 색인구조로부터 유도하여 구축함으로써 유도된 조인 색인구조라 한다.

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.

Bertino는 질의의 대상 범위가 임의의 클래스들로 제한되는 타겟 클래스 대치가 있는 경로식으로 표현된 중포속성에 대한 질의를 지원할 수 있는 NIX (Nested-Inherited Index)[2]라고 하는 색인구조를 제안하였다. NIX는 클래스 계층에 대한 색인구조의 하나인 CH-index[7]와 중포 색인을 통합한 형태의 색인구조이다. 즉, NIX는 B^+-tree의 단말 노드를 색인된 중포속성의 키 값과 스키마 그래프상에서 직접 또는 간접적으로 이 중포속성을 가지는 모든 클래스들의 OID들을 대상으로 클래스별 구분을 위한 클래스 디렉토리와 함께 구성한 색인구조이다. 그러나, NIX 역시 도메인 클래스의 대치가 있는 경로식으로 표현된 질의를 지원하지 못한다. 이는 색인된 중포속성으로부터 각 타겟 클래스까지의 경로 정보를 유지하지 않기 때문이다.

다음 제 3절에서 소개하는 다차원 중포속성 색인구조[11]는 다차원 색인구조를 중포속성에 대한 색인구조로 이용하는 색인기법으로 타겟 클래스 대치 또는 도메인 클래스 대치가 있는 질의를 모두 지원할 수 있다.

3. 다차원 중포속성 색인구조

객체 데이타베이스 질의어의 중포술어에는 클래스 대치가 있을 수 있으며, 이러한 중포술어의 처리를 지원하기 위한 색인구조로 다차원 색인구조를 이

용할 수 있다. 즉, 색인할 중포속성의 키 속성 도메인과 함께 중포속성을 표현하는 경로상의 각 클래스 계층마다 클래스 식별자들로 구성된 한 차원씩의 클래스 식별자 도메인[10,13]을 할당함으로써, (*경로길이 + 1*)차원의 도메인 공간을 구성하여 이를 다차원 색인구조에 적용한다. 예를 들어, 그림 1과 같은 스키마 그래프에서 Person 클래스의 중포속성 price (Computer 클래스의 단순속성)에 대한 색인구조로서 중포속성 price의 키 속성 도메인, Person 클래스 계층의 클래스 식별자 도메인, 그리고 Computer 클래스 계층의 클래스 식별자 도메인으로 삼차원 도메인 공간을 구성할 수 있다.

본 논문에서는 객체 데이타베이스의 중포속성에 대한 색인구조를 다차원 색인구조의 하나인 계층 그리드 화일(multilevel grid file: MLGF)[13,14]을 이용하여 구성하고, 이를 *다차원 중포속성 색인구조 (Multidimensional Nested Attribute Index: MD-NAI)*라 한다. MD-NAI는 디렉토리와 색인 페이지로 구성된다[1]. 디렉토리는 다단계의 균형된 트리 구조를 가지며, 디렉토리를 구성하는 디렉토리 페이지의 구조는 MLGF에서와 마찬가지이다. 색인 페이지는 색인 레코드들로 구성되며, 각 색인 레코드에는 경로상의 각 클래스 식별자 값(class-id value) 필드, 키 값(key value) 필드, 객체 또는 경로 인스턴스의 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.

MD-NAI는 색인 페이지의 색인 레코드에 있는 색인 엔트리의 구성방법에 따라 다차원 중포 색인구조과 다차원 경로 색인구조의 두 가지 색인구조로 분류할 수 있다. *다차원 중포 색인구조(Multidimensional Nested Index: MNI)*는 색인 엔트리를 색인된 중포속성의 타겟 클래스 계층에 속하는 객체에 대한 객체 식별자(즉, OID)들로 구성하며, *다차원 경로 색인구조(Multidimensional Path Index: MPI)*는 색인 엔트리를 색인된 중포속성에 대한 경로 인스턴스(즉, OID 리스트)들로 구성한다. MPI와 같이 색인 엔트리를 경로 인스턴스들로 구성하는 것은 색인 엔트리를 타겟 클래스 계층의 객체 식별자만으로 구성하는 경우에 발생하게 되는 데이타베이스의 변경에 따른 색인구조의 막대한 유지비용을 줄이기 위함이다. 그림 2는 MPI의 색인 페이지 구조를 나타낸다.

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[1,3]. 다음은 데이타베이스의 변경에 따른 두 색인구조의 운용법과 그에 따른 유지비용에 대한 비교 분석이다.

3.1 다차원 중포 색인구조(MNI)의 운용

MNI는 경로 인스턴스를 유지하는 MPI에 비해 저장 공간의 오버헤드가 적은 반면에, 데이타베이스의 변경에 따른 색인구조의 유지비용에 대한 오버헤드가 많다. 경로 P에서 i번째 클래스 계층에 있는 임의의 객체 O_i에서 속성 A_i의 값으로 O_{i-1}에서 새로운 O'_{i+1}로 변경될 경우, MNI의 갱신을 위해서는 다음과 같은 과정의 작업이 필요하다.

- **과정 1:** 객체 O_{i+1}로부터 중포속성 A_n까지 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_i가 다중값을 갖지 않으면 K(A)는 하나의 원소만을 가진다.
- **과정 2:** 객체 O'_{i+1}로부터 중포속성 A_n까지 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 없으므로 작업을 종료하고, 그렇지 않으면 과정 3을 시행한다.
- **과정 3:** O_i를 직접 또는 간접적으로 참조하는 타겟 클래스 계층 C_1의 객체 OID들의 집합 R을 구한다. 이를 위해서는 객체 O_i로부터 경로를 따라 역방향 운행을 하여야 한다. 여기서, i = 1이면 R은 $\{O_i\}$가 된다.
- **과정 4:** 색인의 갱신을 다음과 같이 시행한다. — K(A) ⊃ K(B) 이면, K(A) - K(B)를 δ로

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

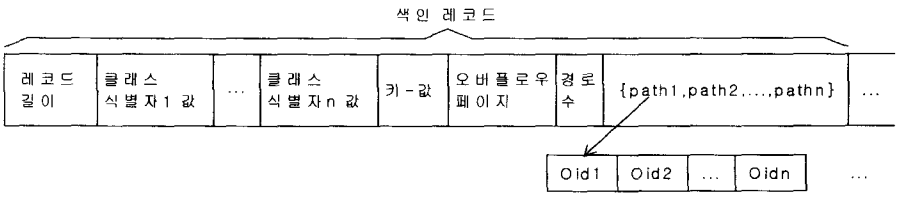


그림 2. MPI의 색인 페이지 구조

하고 δ에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 OID들을 제거한다.

— K(A) ⊂ K(B) 이면, K(B) - K(A)를 δ로 하고 δ에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 OID들을 첨가한다.

— 그렇지 않으면, K(A) - K(B)를 δ1, K(B) - K(A)를 δ2라 하고, δ1에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 OID들을 제거하고, δ2에 속하는 각 색인키 리스트에 해당하는 색인 레코드에서 집합 R에 있는 OID들을 첨가한다.

이와 같이 MNI에서는 데이타베이스의 변경에 따른 색인구조의 갱신을 위하여, 경로상의 역방향 운행이 필요하다. 따라서, 데이타베이스의 각 객체 내에 자신을 참조하는 객체에 대한 역 참조자가 존재하지 않으면, MNI의 사용은 불가능하다. 그리고, 객체의 삽입과 제거에 따른 색인구조의 갱신을 위해서는 한 번의 순방향 운행만이 필요하며, 이는 변경의 경우와 같다.

3.2 다차원 경로 색인구조(MPI)의 운용

MPI는 색인 레코드에 경로 인스턴스를 저장하기 때문에 MNI에 비해 많은 양의 저장 공간을 필요로 하는 반면에, 데이타베이스의 변경에 따른 색인구조의 유지비용에 대한 오버헤드가 MNI에 비해 적게 된다. 경로 P에서 i번째 클래스 계층에 있는 임의의 객체 O_i에서 속성 A_i의 값으로 O_{i+1}에서 새로운 O'_{i+1}로 변경될 경우, MPI의 갱신을 위해서는 다음과 같은 과정의 작업이 필요하다.

- **과정 1:** 객체 O_{i+1}로부터 중포속성 A_n까지 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.
- **과정 2:** 객체 O'_{i+1}로부터 경로에 따른 순방향 운행을 통하여 중포속성 A_n까지 서브경로 인스턴스들의 집합 PI.suf와 경로상의 클래스 식별자 값들과 키 속성 값으로 구성된 색인키 리스트들의 집합 K(B)를 구한다.
- **과정 3:** K(A)에 있는 각 색인키 리스트에 해당하는 색인 레코드를 액세스하여 i번째 항목이 O_i이고 i+1번째 항목이 O_{i+1}인 경로 인스턴스를 삭제함과 동시에, 각 경로 인스턴스의 첫번째 항목에서 i번째 항목까지의 부분을 PI.pre에 보관한다.
- **과정 4:** i번째 항목이 O_i이고 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.
- **과정 5:** K(B)에 있는 각 색인키 리스트에 해당하는 색인 레코드에 집합 PI에 있는 경로 인스턴스를 첨가한다.

이와같이 MPI에서는 MNI에서와는 달리 데이타베이스의 변경에 따른 색인구조의 갱신을 위한 역방향 운행이 필요 없다. 이는 경로 인스턴스가 색인 레코드에 저장되어 있기 때문이다. 따라서, MPI는 데이타베이스의 객체 내에 역 참조자가 존재하지 않아도 사용할 수 있다.

3.3 운용에 따른 MNI와 MPI의 비교 분석

검색 질의에 대한 두 색인구조의 성능에 대해서는 MNI가 MPI에 비해 좋은 성능을 가진다. 이는 MPI에서는 색인 엔트리를 색인된 중포속성의 경로 인스턴스로 구성하기 때문에 색인 엔트리를 타겟 클래스 계층에 속하는 객체 식별자만으로 구성하는 MNI에 비하여 많은 저장 공간의 오버헤드 때문이다. 그러나, 두 색인구조의 운용에 따른 유지비용에 대한 오

버헤드는 색인된 중포속성의 경로 길이에 따라 많은 차이를 보이게 된다. 따라서, 본 절에서는 각 색인구조의 운용에 따른 유지비용을 비교 평가한다.

첫째로, 색인구조를 구축할 경로의 길이가 2인 경우에 대한 분석이다. 경로의 길이가 2이고 역 참조자가 데이타베이스의 객체 내에서 제공될 경우에는 ENI와 MTI의 유지비용은 같게 된다. 그 이유로서, 먼저 경로상의 두 번째 클래스 계층에 있는 객체 O_2에서 색인된 속성 A_2가 변경되면 객체 O_2내에 역 참조자가 존재하기 때문에 객체 O_2를 참조하는 첫 번째 계층의 객체를 결정하기 위한 역방향 운행이 필요 없기 때문이다.

그리고, 데이타베이스의 변경이 첫 번째 클래스 계층 C_1에서 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_1에서 속성 A_1의 값이 두 번째 클래스 계층의 다른 객체를 참조하게 변경될 경우, ENI와 EPI에서는 색인키 값을 얻기 위하여 A_1의 변경전의 값과 변경후의 값에 대한 객체 O'와 O''를 액세스하는 두 번의 순방향 운행이 필요하다. 그러나, 이 두 객체의 액세스는 O_1에 대한 역 참조자의 값을 변경하기 위하여 색인구조와 무관하게 반드시 액세스해야 한다.

둘째로, 색인구조를 구축할 경로의 길이가 3이상인 경우에 대한 분석이다. 경로 길이가 2보다 크게되면, MNI에서는 역 방향 운행이 필요하게 되므로 MPI에 비해 유지비용이 증가하게 된다. MNI의 유지비용을 지배하는 것은 역방향 운행에 의한 것으로, 역방향 운행에 필요한 객체의 액세스 개수는 객체 참조 공유도[2]에 의해 결정된다. MPI에서는 역방향 운행이 필요 없기 때문에 MNI에서보다 유지비용이 매우 적게 된다. 한편, 데이타베이스의 변경이 첫 번째 클래스 계층 C_1과 두 번째 클래스 계층 C_2에서 발생할 경우에는, MNI에서도 역방향 운행이 필요 없게되므로 색인구조의 유지비용이 MPI에서와 같게 된다.

따라서, 위와 같은 분석의 결과로서 다음과 같은 결론을 얻을 수 있다. 첫째로 색인을 구축할 경로의 길이가 2인 경우에는 MNI가 적합하다. 이것은 경로의 길이가 2인 경우의 유지비용에 대한 분석에서 살펴본 바와 같이, 유지비용은 두 가지 색인구조 모두 비슷한 반면에 MNI에서 검색 성능이 가장 좋기 때문이다. 둘째로 색인을 구축할 경로의 길이가 3이상인 경우에는 일반적으로 MPI가 적합하다. 이것은 MPI의 검색 성능은 MNI에 필적하는 반면에 데이타베이스 변경에 의한 유지비용이 적게 되기 때문이다. 그리고, MPI는 각 객체 내에 역 참조자가 존재하지 않을 경우에도 사용이 가능하다.

2) 클래스 계층 C_i에서 속성 A_i의 값으로 동일한 값을 가지는 객체의 평균 개수.

4. 다차원 경로 색인구조의 구성기법

본 절에서는 제 3절의 다차원 중포속성 색인구조의 운용에 따른 유지비용의 결과를 바탕으로, 단일 술어가 아닌 여러 개의 중포술어들을 가지는 보다 일반적인 복합질의에 대해서, 데이타베이스의 각 객체가 역 참조자를 가지고 있지 않거나 경로의 길이가 3이상인 경우에 유지비용 면에서 유리한 다차원 경로 색인구조(MPI)를 구성하여 질의를 처리하는 방법을 분석한다. 먼저, 두 개의 중포술어가 주어지는 질의의 경우를 고려하고, 이를 여러 개의 중포술어로 구성된 일반적인 경우로 확장한다.

두 개의 술어로 구성된 각 질의의 형태에 따른 MPI 색인의 구성 방법은 그림 3과 같이 분류할 수 있다. 그림 3에서는 먼저, 술어가 주어진 두 개의 경로 P_1과 P_2에 대해서 경로가 서로 겹침이 없는 경우와 있는 경우로 분류하고, 그 다음에 각각에 대해서 MPI를 구성할 수 있는 방법들을 나타낸다. 그리고, 각 색인구성 방법에 대하여 기호를 매기고, 앞으로 이 기호로서 각 구성 방법을 대신해서 사용한다. 그림 3에서는 하나의 경로를 두 개의 부경로로 분리해서 각 부경로에 대해서 별도의 MPI를 구성하는 경우도 포함하고 있다.

경로 $P = C_1.A_1.A_2. \ldots A_n$이 주어지면, P상에서 대응되는 클래스 계층들은 $\{C_1, C_2, \ldots, C_n\}$이다. 다음 표 1은 본 절의 색인구성에 따른 질의처리 분석을 위한 비용 모델에서 필요한 파라미터들이다.

클래스 계층 C_i에서 속성 A_i의 값으로 동일한 값을 가지는 객체의 평균 개수를 나타내는 파라미터 $k_i(1 \leq i \leq n)$는 참조의 공유정도를 나타내며, 이것은 질의처리 비용에 가장 크게 영향을 미친다. 앞으로 연속된 참조 공유도의 곱으로 다음과 같은 표기법을 사용한다.

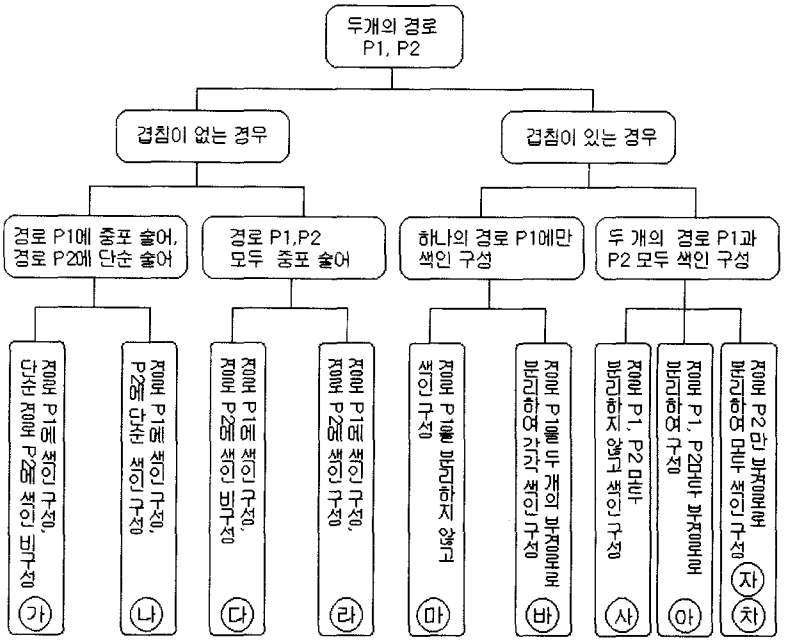


그림 3. 두 개의 술어를 가진 경로상에 가능한 색인구성의 분류

$$k(i, j) = k_i \times \cdots \times k_j \quad (2)$$

그리고, 비용 모델을 간단히 하기 위해서 다음과 같은 몇 가지 가정을 한다. 이러한 가정들은 실제 시스템과 완전히 부합하지는 못하지만 객체 데이타베이스의 액세스 구조를 위한 여러 분석 모델에 의한 연구[2,7]에서 일반적으로 사용하는 가정들이며, 이러한 연구의 결과는 유용하게 사용되고 있다.

- 클래스 C_i의 각 객체는 클래스 C_{i-1}의 객체들에 의해 반드시 참조된다. 이것은 파라미터의 관점에서는 $D_i = N_{i-1}$를 의미한다.
- 모든 키 값들은 동일한 길이를 가진다. 이것은 모든 비단말 노드 색인 레코드들이 동일한 길이를 가짐을 의미한다.
- 속성의 값들은 속성을 정의하는 클래스의 객체들 중에서 균일하게 분포된다.
- 모든 속성은 단일 값을 가진다.
- 모든 색인은 클러스터 되지 않았다. 이것은 객체들이 단말노드 색인 레코드들에 저장되어 있는 OID들의 순서에 따라서 저장되지 않음을 의미한다.

본 논문에서의 비용 함수는 참조 공유도를 나타내는 파라미터 k_i와 클래스의 카디널리티 N_i에 중점을 둔다. 이들은 MPI의 액세스 비용에 가장 큰 영향을 미친다.

표 1. 비용 모델에서 사용한 파라미터

파라미터	의 미
h	다차원 색인구조를 구성하는 디렉토리 트리의 높이
P	색인 페이지의 크기(4096 바이트)
D_i	클래스 계층 C_i에서 속성 A_i가 가지는 서로 다른 값의 개수($1 \le i \le n$)
N_i	클래스 계층 C_i의 카디널리티($1 \le i \le n$)
K_i	클래스 계층 C_i에서 속성 A_i의 값으로 동일한 값을 가지는 객체의 평균 개수($K_i = \lceil N_i/D_i \rceil$)
XP	MPI 색인구조의 평균 색인 레코드 길이
$OIDL$	객체 식별자의 길이(8 바이트)
$PC(C_i)$	클래스 C_i의 객체들을 가지는 디스크 페이지의 개수
MPA	MPI 색인구조의 액세스 비용($MPA = h_{MPI} + \lceil XP/P \rceil$)

4.1 순방향 운행기법에 의한 질의처리의 비용식

앞 절에서 논의한 것처럼, 색인이 제공되지 않을 때에는 술어가 객체들 사이의 참조에 의해서 명시적으로 평가되어야 한다. 본 절에서는 제 2.2절에서 소개한 순방향 운행법(FT)과 네스티드 루프(NP) 검색법에 의한 질의처리 전략인 FTNL 전략에 대한 질의처리 비용식을 나타낸다. 이는 앞으로 여러 색인구성에 대한 비교 평가를 위해 필요한 비용식이다. 비용식에서 다음과 같은 표기법을 사용한다.

- $A_c(C, N, A, S)$: 클래스 C의 객체 수 N의 집합에서 중포속성 A를 위한 값으로 집합 S에 속하는 OID들을 가지는 객체들을 결정하는 비용을 나타낸다. 예를 들어, 그림 1의 스키마에서 집합 $S = \{Company[i], Company[m]\}$이 주어질 때, $A_c(Computer, N_{Computer}, manuf., S)$는 *manufacturer*로서 $Company[i]$ 또는 $Company[m]$을 가지는 클래스 *Computer*의 객체를 결정하는 비용을 나타낸다.
- $A_p(C, N, A, pred)$: 클래스 C의 객체 수 N의 집합에서 술어 *pred*를 만족하는 중포속성 A를 가지는 객체들을 결정하는 비용을 나타낸다. 예를 들어, 술어 "*location = 'Daegu'*"를 고려할 때, $A_p(Person, N_{Person}, location, location = 'Daegu')$는 *Daegu에 있는* Company에서 만든 Computer를 가지고 있는 클래스 Person의 객체를 결정하는 비용을 나타낸다.

FTNL 전략은 객체들을 하나하나 방문하는 전략이다. 객체 탐색에는 두 가지의 형태가 있다. 하나의 형태 (a)는 타겟 클래스 전체로부터 시작하는 탐색으로, 클래스 C_i의 객체 수가 N_i일 때 비용식은 다음과 같다.

$$cost\_FTNL = PC(C_i) + 2 \times N_i \times (n-i) \quad (3)$$

이 식은 다음과 같이 유도된다. C_i의 객체들을 검색하기 위해서, 모든 페이지들이 액세스되어야 하고, 이 페이지들의 수는 $PC(C_i)$이다. 그리고, 각 객체들에 대해서 속성 A_n의 값에 도달해야 한다. 각 속성 $A_j(i \le j \le n)$에 대해서, 각 객체의 위치가 저장되어 있는 시스템 테이블의 액세스와 객체 자체의 액세스가 필요하다.

다른 형태 (b)는 중간 질의처리의 결과에 따른 탐색으로, 이전의 질의에 의해서 반환된 인스턴스의 처리과정이다. 중간 질의처리 결과의 객체 수를 N이라 할 때, 타겟 클래스 객체들의 OID는 알려져 있으므로 비용식은 다음과 같다.

$$cost\_FTNL = 2 \times N \times (n-i+1) \quad (4)$$

이식은 형태 (a)와 동일하게 유도된다. 단지, 첫 번째 클래스의 액세스가 필요 없기 때문에 $PC(C_i)$가 생략된 것이다. 이식들을 종합하면 다음과 같다:

$$A_c(C_i, N, A_n, S) = A_p(C_i, N, A_n, pred) = PC(C_i) + 2 \times N_i \times (n-i) \text{ if } N = N_i,$$
$$2 \times N \times (n-i+1) \text{ if } N < N_i \quad (5)$$

여기서 A_c와 A_p는 동일하다. 왜냐하면 이 운행법에서는 탐색의 종류에 관계없이 모든 인스턴스들이 액세스되어야 하기 때문이다.

4.2 겹침이 없는 두 경로 사이의 색인구성

본 절은 그림 3에서의 색인구성 ㉮, ㉯, ㉰, ㉱에 대한 분석이다. 경로의 일부가 겹치지 않을 때에는 질의처리 전략에서 MPI 색인을 이용하거나 MNI 색인을 이용하거나 차이가 없다. 예를 들어, 동일한 클래스로부터 시작하는 두 개의 경로 $P_1 = C_1.A_1.A_2. \ldots .A_n$과 $P_2 = C_1.A'_1.A'_2. \ldots .A'_m$에 대해서, 각 경로에 주어진 두 가지 술어 $pred_n$과 $pred_m$을 가지는 질의를 고려해 보자. MPI 색인이 각 경로에 구성된다면, 어떠한 질의처리 전략이라도 시작 클래스와 경로의 마지막 속성만을 사용하므로, 경로 인스턴스를 색인 값으로 가지는 MPI의 특성에 의한 이득이 없다. 또한, 이와 같은 사항은 단순술어의 경우에도 마찬가지로 적용된다. 두 경로 사이에 겹침이 없는 경우의 각 색인구성에 따른 질의처리 전략의 결과를 정리하면 다음과 같다.

(1) 중포술어와 단순술어에 대한 색인구성 ㉮와 ㉯

중포술어에 대해서 MPI를 사용하는 전략의 검색 비용이 MNI을 사용하는 전략보다 항상 조금 많게 된다[3]. 경로가 길어지면 MPI의 성능은 더욱더 나빠

3) 그러나, MPI는 MNI보다 일반적으로 데이타베이스의 변경에 따른 색인구조의 유지비용에 대해서는 더 좋은 성능을 가진다(제 3.3절 참조). 본 절에서는 주로 복합 질의에 대한 검색 성능만을 고려하기 때문이며, 앞으로도 유지비용에 대해서는 고려하지 않는다.

진다. 이것은 MPI가 가지고 있는 본래의 공간 사용의 복잡성으로 쉽게 알 수 있다. 중포술어와 단순술어로 구성된 질의의 경우, MPI 색인을 사용하는 전략의 질의처리 성능은 참조 공유도 $k(1, n)$의 크기에 따라 많은 영향을 미친다. 단순 경로에는 색인이 구성되지 않은 색인구성 ㉮의 경우, 단순술어를 만족하는 객체의 개수가 N개 이면 중포술어가 있는 경로상에서도 $(k(1, n) \leq 1000 \times N)$일 경우에만 색인의 사용이 유용하다. 그렇지 않을 경우에는 색인 레코드의 길이가 너무 크게되어 색인을 이용하지 않는 전략보다 질의처리 성능이 떨어지게 된다. 단순 경로에도 색인이 구성되는 색인구성 ㉰의 경우에도 비슷한 상황을 나타내지만, 단순 경로상의 색인이 질의처리 성능에 매우 큰 영향을 미치게 된다.

(2) 두 개의 중포술어에 대한 색인구성 ㉱와 ㉲

두 개의 중포술어에 대해서 P_2상에는 색인을 구성하지 않고 P_1상에만 구성하는 색인구성 ㉱에서는 경로의 길이가 짧은 경우(n=m=2)에는 MPI의 사용이 MNI의 사용과 메모리 액세스 면에서 차이가 없다. 그러나 경로의 길이가 길 경우(n=m=5)에는 그림 4의 (a)에서처럼 MPI 색인을 이용한 질의처리의 비용이 증가한다. 이것은 MPI 색인 레코드의 크기가 MNI 색인 레코드에 비해 매우 많이 커지기 때문이다. 이 비용은 참조 공유도에 따라 일정하게 증가함을 알 수 있다. 그리고, 두 개의 중포술어에 대해서 두 개 모두에 색인이 구성된 색인구성 ㉲에서는 그림 4의 (b)에서처럼 MNI 색인의 사용에 대한 MPI 색인을 사용하는 질의처리 비용의 증가가 색인구성 ㉱에서 보다 훨씬 많이 증가하게 된다. 이것은 단순히 MPI가 가지고 있는 본래의 공간 사용의 크기에 따른 색인 액세스 비용이 MNI 색인 보다 훨씬 많아지기 때문이다. 그러나, 이것은 데이타베이스의 변경이 거의 없을 경우에만 해당된다. 데이타베이스의 변경이 많아지면 MNI 색인은 많은 유지비용 때문에 사용할 수 없게 된다.

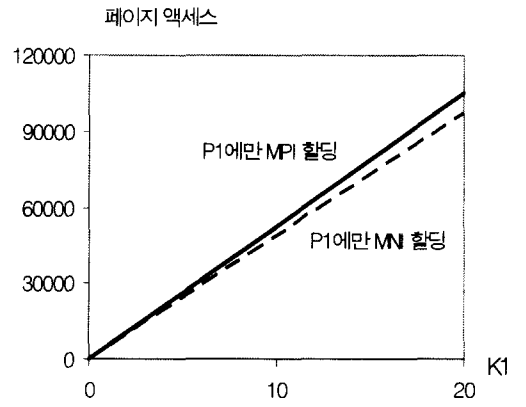
4.3 겹침이 있는 두 경로 사이의 색인구성

본 절에서는 동일한 클래스에서 시작하여 경로의 일부가 겹치는 두 개의 경로상에서, 각 경로의 마지막 속성에 술어를 가지는 질의에 대해서 각 색인구성에 대한 질의처리 전략의 분석이다.

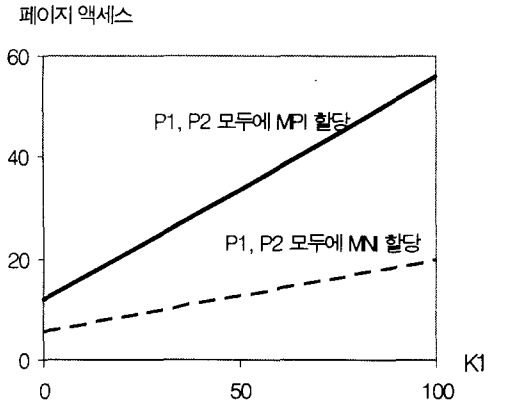
두 개의 경로 $P_1 = C_1.A_1.A_2. \ldots .A_n$과 $P_2 = C_1.A'_1.A'_2. \ldots .A'_m$에서, $1 \leq i \leq j$에 대해서는 $A_i = A'_i$이고, $i>j$에 대해서는 $A_i \neq A'_i$인 $j(< \min(n, m))$가 존재한다면, 이 두 경로는 겹쳐진 경우이다. 여기서, 먼저 분리된 경로구성에 대해서 소개한다. 분리된 경로구성은 하나의 경로를 두 개의 부경로로 분리하여서, 각 부경로에 색인구조를 구성한 경우이다. 그림 5는 집단화 계층에 따른 경로구성을 본 논문에서 앞으로 사용할 경로들의 이름과 함께 나타낸 것이다. 그림 5의 분리된 경로에서 분리가 시작되는 클래스 계층은 C_1이다.

(1) P_1에만 MPI 색인이 있는 경우: 색인구성 ㉳

그림 3의 색인구성 ㉯에서처럼 겹침 경로에 대해



(a) 색인구성 ㉱의 경우, 객체의 크기=100, n=m=5, $k_i=k'_i=50$ (모든 i에 대해서)



(b) 색인구성 ㉲의 경우, 객체의 크기=100, n=m=3, $k_i=k'_i=10$ (모든 i에 대해서)

그림 4. 겹침이 없는 경로에서 색인구성 ㉱와 ㉲의 질의처리 성능.

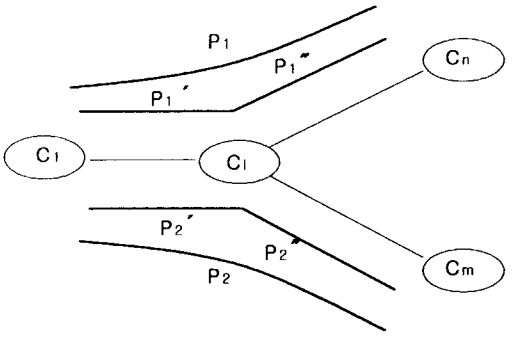


그림 5. 분리된 경로구성 스키마

서 경로구성을 분리하지 않고 경로 P_1에만 MPI 색인을 구성할 경우 다음과 같은 네 가지 질의처리 전략이 있을 수 있다. 이 전략들 사이의 차이점은 단지 C_{i+1}에서 C_m까지 방문하는 방법에 있다. 이러한 전략들에서는 MPI의 색인 엔트리인 경로 인스턴스에서 객체 식별자들의 튜플을 생성하는 프로젝션 연산이 사용된다. 이 네 가지 전략의 세부사항은 다음과 같다.

(A) 술어 $pred_n$은 P_1상에 정의된 MPI에 한번 액세스하는 것으로 해결된다. 먼저, 탐색된 경로 인스턴스들에 대하여 프로젝션 연산을 수행하여 (O_1, O_i)쌍들을 생성한다. 그리고, 각 쌍들의 두 번째 원소들로서 정렬하여 술어 $pred_m$을 해결하기 위하여 C_m까지 FTNL 전략을 사용한다.

(B) (A)의 전략에서 C_i에서 C_m까지의 객체들의 탐색을 위하여 FTSD 전략을 사용한다.

(C) (A)의 전략에서 C_i에서 C_m까지의 객체들의 탐색을 위하여 RTNL 전략을 사용한다.

(D) (A)의 전략에서 C_i에서 C_m까지의 객체들의 탐색을 위하여 RTSD 전략을 사용한다.

본 논문에서는 단지 순방향 운행(FT)을 사용한 전략을 위한 비용식만 제시한다. 역방향 운행(RT)은 대부분의 경우에서 매우 많은 비용을 필요로 하기 때문이다. 또한 각 색인구성의 비교를 쉽게 하기 위하여 네스티드 루프(NL) 전략만을 고려한다. 따라서, 전략 (A)의 비용식을 비용식 (5)를 이용하여 나타내면 다음과 같다.

$$Cost(A) = MPA(P_1) + A_i(C_i, k(i,n), A'_m, A'_{m-1} = value_m) = MPA(P_1) + 2 \times k(i,n) \times (m-i+1) \quad (6)$$

이 식은 다음과 같이 설명할 수 있다. $MPA(P_1)$은 경로 P_1에 구성된 MPI의 액세스에 대한 비용이다. 이것은 경로 P_1의 인스턴스들을 반환한다. 이 인스턴스들에 대한 프로젝션 연산은 $k(i, n)$개의 쌍들을 반환한다. 이 쌍들의 두 번째 원소들에 대해서 그 속성의 값을 얻기 위하여 두 번의 페이지 액세스가 필요하고, 이것은 경로 P''_2의 길이인 $(m-i+1)$만큼 반복되어야 한다.

여기서, MPI 색인의 사용과 MNI 색인의 사용을 비교하면, MPI 색인을 사용하는 경우가 MNI 색인을 사용하는 경우에 비해서 대부분의 경우에 유리하다. 이는 MNI의 경우에는 술어 $pred_m$을 평가하기 위하여 O_i가 아닌 O_1으로부터 C_m까지 운행을 해야하기 때문이다. 그림 6은 색인구성 ㉮의 경로구성에서 MPI와 MNI의 성능을 나타내며, 참조 공유도 k가 너무 크지 않을 때는 경로 P_1에 MPI를 구성하는 것이 효율적임을 나타낸다.

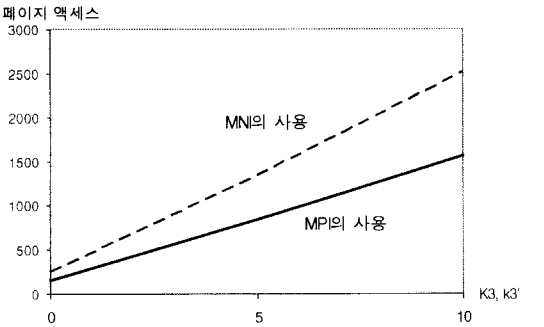


그림 6. 색인구성 ㉮에서 MPI과 MNI의 성능 비교

(2) P'_1과 P''_1에 MPI 색인이 있는 경우: 색인구성 ㉯

그림 5의 겹침 경로에 대해서, 색인구성 ㉯는 P_1 경로를 분리하여 부경로 P'_1과 P''_1상에 MPI을 구성하는 경우이다. 이러한 경로구성에서 있을 수 있는 질의처리 전략들은 다음과 같다.

(A) 경로 P''_1에서 색인을 액세스하고, $pred_m$을 평가하기 위해서 C_i에서부터 C_m까지 FTNL 전략을 사용한다. 그리고 검색된 각 객체에 대해서 경로 P'_1에서 색인을 액세스한다.

(B) 경로 P''_1에서 색인을 사용하지 않은 전략으로, 두 술어를 평가하기 위하여 C_i에서 C_n과 C_m까지 각각 FTNL 전략을 사용한다. 그리고 검색

된 각 객체에 대해서 경로 P'_1에서 색인을 액세스한다.

(C) 경로 P'_1에서 색인을 사용하지 않은 전략으로, 경로 P''_1에서 색인을 액세스하고, $pred_m$을 평가하기 위해서 C_i에서부터 C_m까지 FTNL 전략을 사용한다. 그리고 C_1에서부터 C_i까지 FTNL 전략을 사용한다.

이들 각 전략의 비용식은 다음과 같다.

$$\begin{aligned} Cost(A) &= MPA(P''_1)+A_p(C_i,\ k(i,\ n),\ A'_m,\ A'_m \\ &\quad = value_m)+no\times MPA(P'_1) \\ &= MPA(P''_1)+2\times k(i,\ n)\times(m-i+1)+ \\ &\quad \lceil k(i,\ n)/D'_m \rceil \times MPA(P'_1) \end{aligned} \quad (7)$$

여기서, $no = \lceil k(i,\ n)/D'_m \rceil$로서 두 술어를 만족하는 클래스 C_i 객체들의 추정치를 나타낸다.

$$\begin{aligned} Cost(B) &= A_p(C_i,\ N_i,\ A_n,\ A_n = value_n)+A_p(C_i, \\ &\quad k(i,\ n),\ A'_m,\ A'_m = value_m)+no\times \\ &\quad MPA(P'_1) \\ &= PC(C_i)+2\times N_i\times(n-i)+2\times k(i,\ n)\times \\ &\quad (m-i+1)+\ \lceil k(i,\ n)/D'_m \rceil \times MPA(P'_1) \end{aligned} \quad (8)$$

$$\begin{aligned} Cost(C) &= MPA(P''_1)+A_p(C_i,\ k(i,\ n),\ A_m,\ A_m = \\ &\quad value_m)+A_c(C_1,\ N_i,\ A_i,\ no) \\ &= MPA(P''_1)+2\times k(i,\ n)\times(n-i+1)+ \\ &\quad PC(C_1)+2\times N_1\times(i-1) \end{aligned} \quad (9)$$

다음은 색인구성 ㉯에 대한 각 질의처리 전략들의 비교 분석이다. $MPA(P''_1) > PC(C_i)+2\times N_i\times(n-i)$, 즉 $h_{P'_1} + \lceil \frac{k(i,n)\times(n-i)}{500} \rceil > PC(C_i)+2\times N_i\times(n-i)$ 이면, Cost(A) > Cost(B) 이다. 그리고, $\lceil \frac{k(i,n)}{D'_m} \rceil \times (h_{P'_1} + \lceil \frac{k(i,n)\times(n-i)}{500} \rceil) > PC(C_i)+2\times N_1\times(i-1)$이면, Cost(A)> Cost(C)이다. 따라서, 이와 같은 비교에서 다음 정리 1과 같은 결과를 얻을 수 있다.

정리 1: 겹침이 있는 두 경로에서 한 경로를 두 개의 부경로로 분리 구성할 때, 각 부경로의 참조 공유도가 높을 때는 그 부경로에 색인을 사용하는 것보다 FTNL 전략을 사용하는 전략이 더 효율적이다.

다음은 겹침이 있는 두 경로에서 한 경로에만 색인을 구성할 때, 그 경로를 분리하는 경우(색인구성 ㉰)와 분리하지 않는 경우(색인구성 ㉮)에 대한 비교이다. 먼저, 그 결과를 그림으로 나타내면 그림 7과 같다. 이는 비용식 (7)을 색인구성 ㉮의 비용식 (6)과 비교함으로써 알 수 있다. 그림 7에서는 속성 A'_m에 대한 서로 다른 값들의 수인 D'_m이 매우 작을 때는 분리하지 않는 것이 좋음을 나타낸다.

(3) P_1과 P_2에 MPI 색인이 있는 경우: 색인구성 ㉱

그림 3의 겹침 경로에서 두 경로 P_1과 P_2 모두 분리하지 않고 색인을 구성하는 경우이다. 이 경우의 질의처리 전략은 다음과 같다.

(A) 두 개의 색인을 모두 이용하는 전략으로 두 색인을 각각 액세스하여 각 결과의 공통된 부분을 도출한다.

(B) 두 개의 색인 중에 하나만 사용하는 전략이다. 전략 (A)의 비용식은 다음과 같이 나타낼 수

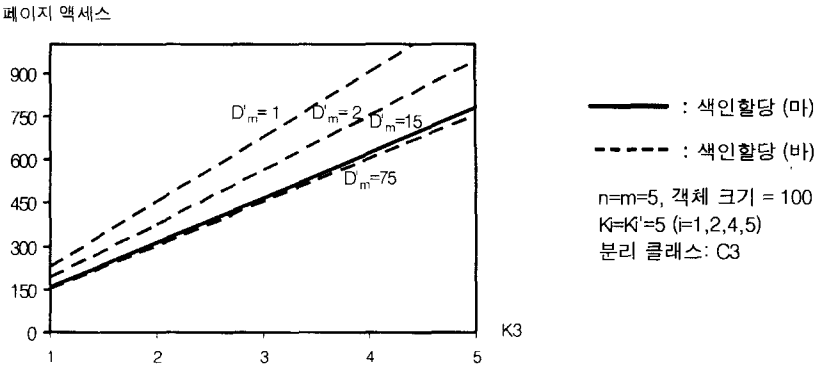


그림 7. 색인구성 ㉯과 ㉰의 성능 비교

있다.

$$Cost(A) = MPA(P_1)+MPA(P_2) \quad (10)$$

두 개의 경로 모두에 색인이 있는 경우와 하나의 경로에만 색인이 있는 경우에 대한 질의처리 전략의 비교는 다음과 같다. 이는 비용식 (10)과 색인구성 ㉮의 비용식 (6)을 비교함으로써 알 수 있다. 두 개의 색인을 사용하는 것이 하나의 색인을 사용하는 것보다 유용하게 되는 경우는 다음과 같은 조건이 성립할 때이다. 비용식 (10)의 값이 비용식 (6)의 값보다 작을 때이므로, $MPA(P_2) < 2\times k(i, n)\times(m-i+1)$일 때, 즉 $h_{P_2}+\lceil \frac{k'(1, m)\times m}{500} \rceil < 2\times k(i, n)\times(m-i+1)$인 경우이다. 이 경우에서는 경로 인스턴스를 색인 값으로 가지는 MPI의 이점을 이용하지 않기 때문에, 두 경로 상에서 많은 갱신이 없으면 MPI의 사용이 MNI의 사용보다 시간과 공간을 낭비하게 된다.

(4) P'_1, P''_1과 P''_2에 MPI 색인이 있는 경우: 색인구성 ㉯

색인구성 ㉯는 겹침이 있는 두 경로에서 모두 분리된 경로구성을 선택하고, 두 경로 모두에 색인을 구성하는 경우이다. 여기에서 두 경로의 첫 번째 부경로의 색인들은 동일하다. 이 경우의 질의처리 전략은 다음과 같다.

(A) 모든 색인을 이용하는 전략으로, 술어 $pred_n$과 $pred_m$을 만족하는 클래스 C_i의 객체들을 구하기 위하여 먼저 P''_1과 P''_2상의 색인을 각각 액세스하여 검색하고, 그들의 공통된 결과를 도출한다. 그리고, 도출된 각 객체에 대하여 경로 P'_1상의 색인을 액세스하여 최종 결과를 얻는다.

(B) P'_1상의 색인을 이용하지 않는 전략으로, P'_1상의 색인을 이용하지 않고 대신에 C_1에서 C_i까지 FTNL 전략을 사용한다.

(C) 경로 P''_1이나 P''_2상의 색인을 모두 또는 하나를 이용하지 않는 전략으로, 이 전략은 색인구성 ㉰에서 제시한 전략들과 같아진다.

각 전략들의 비용식은 다음과 같다.

$$Cost(A) = MPA(P''_1)+MPA(P''_2)+no\times MPA(P'_1) \quad (11)$$

여기서, $no = \lceil k(i, n)/D'_m \rceil (= \lceil k'(i, m)/D_n \rceil)$로서 두개의 술어 $pred_n$과 $pred_m$을 동시에 만족하는 객체의 수를 나타낸다.

$$Cost(B) = MPA(P''_1)+MPA(P''_2)+A_c(C_1, N_1, A_i, no) = MPA(P''_1)+MPA(P''_2)+PC(C_1)+2\times N_1\times(i-1) \quad (12)$$

이 전략들 사이의 비교는 다음과 같다. 먼저, 전략 (A)와 전략 (B)에 대해서 비용식을 비교하면, $k(1, n)\times(i-1)>500\times D'_m\times(PC(C_1)+2\times N_1\times(i-1))$이면, $Cost(A) > Cost(B)$이다. 이것은 경로 P_1상의 참조 공유도가 높을 때는 경로 P'_1상의 색인을 사용하지 않고 FTNL 전략을 사용하는 것이 좋음을 나타낸다.

일반적으로 색인구성 ㉯가 색인구성 ㉰에 비해 효율적이지만, 항상 그렇지는 않다. 색인구성 ㉯와 색인구성 ㉰의 비용식을 비교하면, $h_{P'_2}+\lceil \frac{k'(i, m)\times(m-i)}{500} \rceil > 2\times k(i, n)\times(m-i)$, 즉 $h_{P'_2}+k'(i, m) > 1000\times k(i, n)$이면, 비용식 (11) > 비용식 (7)이다. 따라서, 다음의 정리 2와 같은 결과를 유도할 수 있다.

정리 2: 그림 5와 같은 겹침 경로에서, 부경로 P''_2의 참조 공유도가 부경로 P''_1의 참조 공유도보다 훨씬 클 경우에는 부경로 P''_2에는 색인을 구성하지 않는 것이 좋다. 그리고 색인구성 ㉯와 같이 경로구성을 완전히 분리할 경우에는 MPI 색인보다 MNI 색인을 구성하는 것이 더 효율적이다. 이는 MPI에서와 같은 경로 인스턴스의 사용이 필요없기 때문이다.

(5) P_1, P'_2과 P''_2에 MPI 색인이 있는 경우: 색인구성 ㉱

그림 5의 겹침 경로에서 가능한 또 다른 색인구성은 경로 P_1, P'_2와 P''_2에 색인을 구성하는 것이다. 이 경우의 질의처리 전략은 다음과 같다.

(A) 모든 색인을 이용하는 전략으로, 먼저 술어 $pred_m$을 평가하기 위하여 경로 P''_2상의 색인과 경로 P'_2상의 색인을 차례로 액세스한다. 그리고 술어 $pred_n$을 평가하기 위하여 경로 P_1상의 색인을 액세스하여 $pred_m$의 평가 결과와 교집합을 취한다.

(B) 이 색인구성에 대해서 더욱 적합한 질의처리 전략으로, 먼저 $pred_n$을 평가하기 위하여 경로 P_1상의 MPI 색인을 액세스하여 (O_1, O_i)쌍들의 집합을 프로젝션하여 구한다. 그리고 $pred_m$을 평가하기 위하여 경로 P''_2상의 색인을 액세스하고, 그 결과를 $pred_n$의 평가 결과인 (O_1, O_i)쌍들의 집합에서 두 번째 요소와 교집합을 취하여 최종 결과를 구한다.

각 전략의 비용식은 다음과 같다.

$$Cost(A) = MPA(P''_2)+no\times MPA(P'_2)+MPA(P_1) \quad (13)$$

여기서, $no = k'(i, m)$이다.

$$Cost(B) = MPA(P_1)+MPA(P''_2) \quad (14)$$

비용식 (13)과 (14)를 비교함으로써 다음 정리 3과 같은 결과를 얻을 수 있다.

정리 3: 주어진 두 개의 겹침 경로 P_1과 P_2에서 경로 P_1은 분리하지 않고 색인을 구성하고, P_2는 분리하여 P'_2와 P''_2에 각각 색인을 구성하는 경우에는 모든 색인을 이용하는 질의처리 전략보다 P'_2상의 색인은 이용하지 않는 질의처리 전략이 항상 효율적이다.

두 경로 모두에 색인을 사용하는 경우에, 두 경로 모두 분리하지 않는 색인구성이 한 경로를 분리하는 색인구성보다 항상 질의처리 비용이 많다는 중요한 결과를 얻을 수 있다. 이는 색인구성 ㉮의 비용식 (10)과 비용식 (14)를 비교함으로써 얻을 수 있다. 그리고 이것은 P''_2가 P_2의 부경로이고, 부경로에 정의된 색인의 액세스 비용이 항상 P_2 전체 경로에 정의된 색인의 액세스 비용보다 항상 적다는 사실로서 쉽게 증명할 수 있다. 그러나, 이와 같은 결과는 MNI 색인을 사용하는 경우와는 반대의 결과이다. 이는 MPI 색인의 경우는 프로젝션 연산을 사용함으로써, MNI에서 필요하게 되는 부경로 P'_2상의 색인을 액세스하기 위한 비용이 절약되기 때문이다.

(6) P_1과 P''_2에 MPI 색인이 있는 경우: 색인구성 ㉱

색인구성 ㉮에서의 분석으로부터, 단지 MPI 색인만을 사용할 경우에는 지금까지 제안하지 않은 다른 색인구성을 자연스럽게 생각할 수 있다. 이 색인구성은 경로 P_1상에 색인을 구성하고, P_2는 P'_2와 P''_2로 분리하여 P''_2에만 색인을 구성하는 것이다. 이와 같은 색인구성에서 질의처리 전략에 대한 비용식은 색인구성 ㉮에서의 비용식 (14)와 같다. 즉, 부경로 P'_2에는 색인을 사용하지 않는 것이 사용하는 것보다 항상 유리하므로, 가장 좋은 색인구성 전략은 P'_2상에는 색인을 구성하지 않는 것이다.

4.4 두 개 이상의 경로들에 대한 색인구성의 일반화

본 절에서는 제 4.2절과 제 4.3절의 결과를 확장하여 두 개 이상의 중포술어가 결합된 복합 질의의 경우에 대해서 색인구성과 그에 따른 질의처리 전략을 제시한다.

첫째로, 겹침이 없는 여러 경로들에 각각 정의된 술어들을 가지는 경우이다. 이런 경우는 이미 제 4.2절에서 언급했듯이 MPI 색인의 구성이 MNI 색인의 구성에 비해서 특별히 유용하지는 않다. 특정 경로에 하나의 색인만을 구성하는 경우에는 참조 공유도 $(k(1, n))$가 가장 낮은 경로에 색인을 구성한다. 이런 경우의 질의처리 전략은 먼저 색인된 경로에 주어진 술어들에 대해서 색인을 액세스하여 평가한 후, 그 결과의 객체들에 대해서만 FTNL 전략으로 나머지 술어들을 평가하는 것이다.

둘째로, 겹침이 있는 여러 경로들에 각각 정의된 술어들을 가지는 경우이다. 만약 색인구조를 그림 8의 (a)와 같이 분리되지 않은 경로 하나에만 구성한다면, 색인구성 ㉰의 경우를 일반화하여 참조 공유도 $(k(1, n_i))$가 가장 낮은 경로에 MPI 색인을 구성한다. 이런 경우의 질의처리 전략은, 먼저 MPI 색인을 액세스하여 색인이 구성된 경로의 술어를 만족하는 경로 인스턴스들을 얻는다. 그리고 이들에 대하여 프로젝션 연산을 수행하여 (O_1, O_i)쌍들을 생성하고, 각 쌍들의 두 번째 원소들로서 나머지 경로들의 술어들을 해결하기 위하여 FTNL 전략을 사용한다. 그리고, 만약 색인구조를 경로 하나에만 구성하되 그 색인구조를 그림 8의 (b)와 같이 분리할 수 있다면, 색인구성 ㉯의 결과로부터 MPI 색인보다 MNI 색인을 구성하는 것이 좋다. 이런 경우에는 MPI에서와 같은 경로 인스턴스의 사용이 필요 없기 때문이다.

두 개의 경로 사이에 겹침이 있는 경우에 대한 색인구성 ㉱로부터 얻은 결과를 일반화함으로써 다음과 같은 결과를 얻을 수 있다. 즉, 겹침이 있는 여러 경로들에서 모든 경로에 색인을 구성하고자 할 경우에는 먼저, 하나의 경로는 분리하지 않고 MPI 색인

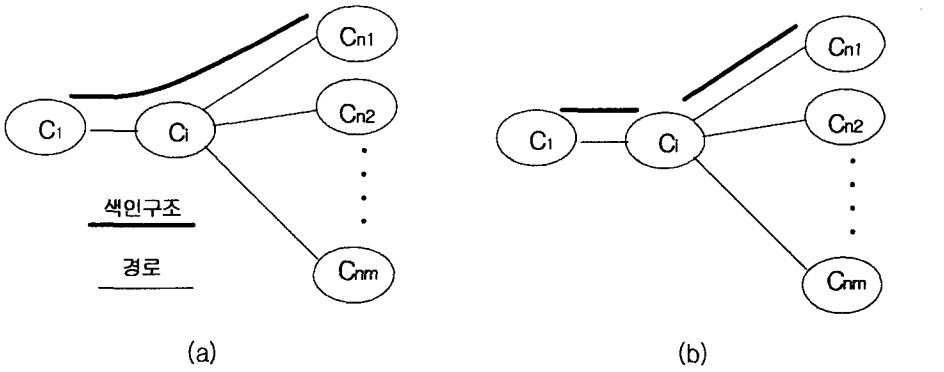


그림 8. 두 개이상의 겹침 경로에서 하나의 경로에 대한 색인구성

을 구성하고, 나머지 경로들은 모두 분리하여 겹쳐지지 않는 부경로에 대해서만 각각 색인을 구성하는 것이 좋다. 이런 경우의 질의처리 전략은 분리하지 않은 MPI 색인에 대해 프로젝션 연산을 이용하는 전략을 사용한다. 결과적으로, 색인구성 ㉮에서처럼 겹쳐진 부경로에 별도로 구성된 색인은 이용하지 않는 것이 유리하므로 색인구성 ㉯의 색인 구성을 일반화하여 그림 9와 같이 구성한다. 이런 경우에 가장 적합한 질의 수행 전략은 색인구성 ㉮의 질의처리 전략 (B)를 일반화하는 것이다. 즉, 가장 긴 MPI 색인을 액세스하여 그 경로에 주어진 중포술어를 만족하는 경로 인스턴스들을 탐색하고, 이들에 대해 프로젝션 연산을 수행하여 (O_1, O_i, ..., O_j) 형태의 튜플들을 생성한다. 그리고, 이들 중 다른 술어를 만족하지 않는 것들을 제거하기 위하여 각 부경로에 주어진 색인들을 액세스하면 된다.

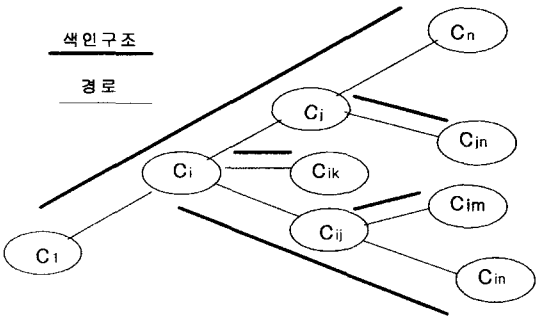


그림 9. 색인구성 ㉯의 일반화

5. 결 론

본 논문에서는 객체 데이타베이스에서 두 개 이상의 중포술어를 가지는 복합질의의 처리를 효율적으로 지원하기 위하여 다차원 경로 색인구조를 최적으로 구성할 수 있는 색인구성 방법을 분석하였다. 다차원 경로 색인구조는 색인된 중포속성의 키 속성 도메인과 함께 중포속성의 경로상에 나타나는 각 클래스 계층마다 한 축의 클래스 식별자 도메인을 할당하여 구성된 다차원 도메인 공간상의 색인 엔트리들을 다루는 다차원 중포속성 색인구조로서 색인 엔트리를 경로 인스턴스로 구성한다. 다차원 중포속성 색인구조는 질의의 대상 범위가 타겟 클래스 계층상의 임의의 클래스들로 대치되거나 경로상의 복합속성의 도메인이 클래스 계층상의 임의의 클래스들로 대치되는 중포술어를 만족하는 질의의 처리를 효율적으로 지원할 수 있는 색인구조이다.

본 논문에서는 먼저, 하나의 중포술어가 주어진 경우에 다차원 중포속성 색인구조의 운용에 대한 분석에서, 중포술어에 나타나는 경로의 길이가 3이상일 때는 색인 엔트리를 타겟 클래스의 OID들만으로 구성하는 다차원 중포 색인구조보다 색인 엔트리를 경로에 대한 경로 인스턴스들로 구성하는 다차원 경로 색인구조가 적합함을 보였다. 이는 데이타베이스의 변경에 따른 다차원 중포 색인구조의 유지비용의 오버헤드가 너무 크기 때문이다. 따라서, 두 개이상의 중포술어가 주어지는 경우에 대해서는 색인구성을 다차원 경로 색인구조로 한정하였다.

두 개의 중포술어가 주어지는 경우에 대해서는 먼저 두 경로 사이에 가능한 모든 색인구성 방법들을 분류하고, 각 색인구성 방법에 대해서 가능한 질의처리 전략들을 제시하고, 이들에 대한 질의처리 비용식을 비교함으로써 최적이 되는 색인구성 방법과 그에 따른 질의처리 전략을 도출하였다. 첫 째로, 겹침이

있는 두 경로에서 한 경로에만 색인을 구성할 경우에는 경로 전체에 대해 하나의 다차원 중포 색인구조를 구성하는 것이 그 경로를 겹침 부경로와 비겹침 부경로로 분리하여 각각 색인구조를 구성하는 경우보다 대부분 더 효율적이다. 이는 질의처리시 비겹침 부경로의 색인 액세스의 결과에 따른 겹침 부경로의 색인 액세스의 횟수가 전체 경로에 구성된 색인구조의 크기에 따른 오버헤드를 상회하기 때문이다. 둘 째로, 두 경로 모두 색인을 구성할 경우에는 경로상의 객체 공유도가 낮은 한 경로에 대해서는 경로 전체에 대해 다차원 경로 색인구조를 구성하고, 나머지 경로에 대해서는 그 경로를 겹침 부경로와 비겹침 부경로로 분리하여 비겹침 부경로에 대해서만 색인을 구성하는 것이 최적이다. 이는 질의처리시 전체 경로 색인구조의 색인 엔트리에 대한 프로젝션 연산을 이용함으로써 질의처리 비용을 최소로 할 수 있기 때문이다.

세 개이상의 중포술어가 주어지는 여러 경로들에서 모든 경로에 색인을 구성하고자 할 경우에는 두 개의 중포술어가 주어지는 경우를 확장하여 먼저, 하나의 경로는 분리하지 않고 다차원 경로 색인구조를 구성하고, 나머지 경로들은 모두 분리하여 겹쳐지지 않는 부경로에 대해서만 각각 색인을 구성하여, 분리하지 않은 다차원 경로 색인구조에 대해 프로젝션 연산을 이용하는 질의처리 전략으로 질의처리 비용을 최소화할 수 있다.

참 고 문 헌

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

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

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

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

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

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

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

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

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

[10] J. H. Lee et al, "A Tunable Class Hierarchy Index for Object-Oriented Databases Using a Multidimensional Index Structure," *Information and Software Technology*, Vol. 43, No. 5, pp. 3095-323, 2001.

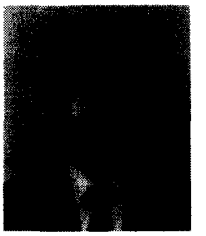
[11] 이 종학, 윤동하, "객체지향 질의처리를 위한 다차원 중포 속성 색인구조의 최적 설계기법," *한국정보처리학회 2002년 추계학술발표논문집*, 제 9권, 제 2호, pp. 1863-1866, 2002년 12월.

[12] T. A. Mueck and M. L. Polaschek, "A Configurable Type Hierarchy Index for OODB," *The VLDB Journal*, Vol. 6, No. 4, pp. 312-332, 1997.

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

[14] K. Y. Whang and R. Krishnamurthy, "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, 1991.

[15] 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

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 대구가톨릭대학교 컴퓨터정보통신공학부 교수
관심분야 : 데이타베이스 시스템, 객체 데이타베이스, 트랜잭션 프로세싱, 생물정보학 등