Integer Programming을 이용한 객체지향 데이타베이스에서 중첩 애트리뷰트 색인의 최적구성

(Optimal Configuration of Nested Attribute Indexes Using Integer Programming in Object-Oriented Databases)

서 상 구 † 이 윤 준 ††
(Sang-Koo Seo) (Yoon-Joon Lee)

요 약 성능 최적화는 객체지향 데이타베이스에서의 중요한 연구분야로 다루어지고 있다. 최근 제안된 중첩 애트리뷰트 색인은 질의처리의 성능 최적화에 크게 기여할 수 있는 것으로 보고되었고 이를 활용하기 위한 질의 최적기의 연구도 활발히 진행되고 있다. 그러나 이들 중첩 애트리뷰트 색인은 그 구성 방법에 따라 검색성능이나 저장공간 및 유지부담이 크게 달라질 수 있다. 본 논문에서는 중첩 애트리뷰트 색인의 최적구성 문제, 즉, 주어진 저장공간내에서 검색 및 갱신질의의 처리비용을 가장 최소화할 수 있는 색인의 선택문제를 연구한다. 이를 위해, 각 색인의 비용절약을 비용함수로 나타내고, 최적 색인구성을 위한 방법으로 Integer Programming 모델을 제안한다. 제안된 최적화 모델은 실험을 통해 그 적용 가능성을 보인다

Abstract Performance optimization is one of important issues in object-oriented databases. It has been reported recently that nested attribute indexes can help reduce query processing costs significantly, and much research efforts have been made to exploit those indexes for the optimization of object-oriented queries. Also pointed out, however, is that they have much higher storage and update overheads than indexes on simple attributes In this paper we address the problem of the optimal index configuration of nested attribute indexes, which is to decide a set of nested attribute indexes subject to a storage limit so that the cost to process queries and maintain updates under the configuration is minimal. We present an Integer Programming formulation and show its feasibility by an experiment.

1. 서 론

컴퓨터 기술의 비약적 발달로 인해, 정보를 데이타베이스화함으로써 DBMS의 지원을 얻고자 하는 응용 분야들이 최근 수년간 많이 등장하고 있다. 몇 예를 들자면 CAD/CAM, CASE, 다중매체 시스템, AI 시스템 등이 있다. 이들 응용분야들은 풍부한 데이타 모델링 기능을 필요로 하며, 객체지향 데이타베이스 기술은 이에 적합한 접근방법중의 하나로 인식되고 있다 [1,7,18].

최근 객체지향 데이타베이스 관련 연구로서, 질의 최적화 연구가 활발히 진행되고 있는데, 이는 이들 새로운 응용 분야들이 데이타베이스 시스템의 뛰어난 성능을 또한 요구하기 때문이다[4,8,16,23,22]. 객체지향 데이타베이스 응용시스템에서의 질의는 흔히 경로식을 포함한다. 예를 들어, 경로식 "Vehicle.manufacturer.president.name"는 "Vehicle", "Company", "Employee"의 세 클래스에 대하여 "Vehicle"의 객체와 그 객체를 생산하는 회사의 사장의 이름을 연결시키는 의미이다. 여기서, "manufacturer"는 "Vehicle" 클래스의 애트리뷰트로서 그 도메인은 "Company" 클래스이고, "president"는 "Employee" 클래스를 도메인으로 하는 "Company" 클래스에 정의된 애트리뷰트이다. 마지막으로 "name"은 "Employee" 클래스에 정의되어 있는 단순 애트리뷰트이다.

† 비 회 원 : 현대전자(주) 소프트웨어연구소
†† 종신회원 : 한국과학기술원 전산학과 교수
논문접수 : 1994년 10월 20일
심사완료 : 1995년 9월 12일

경로식은 전방탐색 및 후방탐색의 두가지 방법으로 처리될 수 있다 [17]. 많은 OODBMS에서 질의처리를 위해 객체를 주기억장치에 탑재시킬 때에 객체의 애트리뷰트 값으로 저장된 객체식별자는 메모리 포인터로 변환하는 기법을 이용하고 있다 [18]. 그러므로 경로식에 대한 반복되는 전방탐색은 메모리 포인터 추적으로 대체됨으로서 신속하게 처리될 수 있다 [21]. 그러나 대부분의 시스템에서처럼 애트리뷰트에 역방향 포인터가 지원되지 않을때에는 후방탐색 연산을 요구하는 질의의 처리비용은 일반적으로 매우 높게된다. 예를 들어, 앞서의 경로식에 대하여, "사장의 이름이 John인 회사에서 생산된 자동차를 모두 검색하라"와 같은 후방탐색 질의를 처리하기 위해서는 세 클래스의 객체들을 거의 모두 검색해야 하는 높은 비용이 요구될 것이다.

이와 같은 후방탐색 질의의 처리 비용을 줄이는 방법으로서 색인은 큰 역할을 할 수 있다. 관계형 데이타베이스에서는 색인이 주로 하나의 릴레이션에서 정의되는데 반하여, 객체지향 데이타베이스에서 색인은 여러 클래스에 걸쳐 정의될 수 있다. 앞서의 경로식 "Vehicle.manufacturer.president.name"에서 "name"을 키로 하고 Vehicle 클래스에 속한 객체들의 객체식별자를 색인 레코드의 값으로 하는 색인을 구성할 수 있다. 이렇게 정의된 색인은 위에 설명한 후방검색 질의를 신속히 처리하는데 아주 유용할 것이다. [3,15,20] 등에서 이러한 색인의 구현방법을 제안하였는데, 이들은 통칭 중첩 애트리뷰트 색인 (Nested attribute index) 라 불리운다 [7].

최근의 객체지향 질의 최적화 연구에서도 이러한 중첩 애트리뷰트 색인을 활용하는 방안이 고려되고 있다 [4,8,16,23]. 그러나, 이들 색인은 기존의 단순 애트리뷰트에 대한 색인보다 갱신 부담 비용이 훨씬 크기 때문에, 신중히 구성되지 않으면 색인을 이용함으로써 얻는 잇점이 상쇄되어 버릴 수도 있다 [7]. 또한 저장공간도 무시할 수 없다. 객체간의 참조 공유도가 큰 클래스들에 걸쳐 정의된 중첩 애트리뷰트 색인일수록 색인화일의 크기는 큰 부담이 된다 [3]. 관계형 시스템에서도 색인의 저장공간은 데이타 저장공간의 30% 정도로 제한하고 있는 것이 보통이다 [10]. 그러므로 중첩 애트리뷰트 색인이 질의 최적화에 효과적으로 이용되기 위해서는 이러한 부담을 고려하여 신중히 구성되어야 한다. 중첩 애트리뷰트 색인의 최적구성에 관련된 연구로서, [3]에서는 갱신 부담을 덜기 위하여, 긴 경로에 대해서 짧은 색인들로 구성하는 것을 권고한 바 있고, [15]에서는 임의의 색인 구성에 대한 질의 및 유지비용을 구하기 위한 비용함수를 제시하였다. 최근에 발표된 E. Bertino의 연구는 본 논문과 매우 유사한 주제를 다루고 있다 [5,6]. 먼저 [6]에서는 클래스 상속계층을 고려한 중첩 애트리뷰트 색인들로 최적 색인구성을 구하는 branch-and-bound기법의 알고리즘을 소개하였다. [5]에서는 dynamic programming을 이용한 알고리즘을 제안하였다. 그러나 이들 연구에서는 실제 데이타베이스 시스템에서 반드시 고려되어야하는 색인의 저장공간 제한문제를 제외시키고 있다. 본 논문에서는 저장공간을 고려하여 [5,6]의 연구범위를 포함할 뿐 아니라, 최적 색인구성 문제를 보다 일반화하고 정형화한다. 또한, 색인구성을 위한 질의의 모델도 집합값 애트리뷰트를 고려하여 보다 일반적인 형태를 고려하고 있다.

본 논문에서는 객체지향 데이타베이스에서 하나의 경로에 대하여, 저장공간의 한계를 고려한 중첩 애트리뷰트 색인의 최적 구성문제를 연구한다. 데이타베이스 정보 (경로길이, 클래스의 객체수 등), 검색 및 갱신질의와 그 중요도 (이하 workload라 부른다), 그리고 색인화일에 대한 저장공간의 한계치 등이 문제의 입력으로 주어진다. 최적화 목표는 하나의 경로에 대한 workload로 주어진 질의들의 총 처리비용을 최소화하기 위한 색인을 구성하되, 구성된 색인들의 총 크기가 주어진 저장공간의 크기를 초과하지 않도록 하는 것이다. 본 논문에서는, 색인별 비용절약 함수를 유도하고 비용절약의 합이 최대화되는 색인구성을 찾는 문제로 전환하는 방법을 기술한다. 실제 최적해를 구하는 방법으로서, 본 논문에서는 OR분야에서 많이 이용되는 Integer Programming (IP) 기법으로 문제를 정형화하고, 실제 IP 패키지를 이용한 실험을 통해 그 타당성을 입증하고자 한다.

본론에 앞서, IP 기법을 데이타베이스색인구성문제에 적용하는 배경에 대하여 언급하고자 한다. 뒤에서 밝히겠지만, 최적 색인구성 문제는 NP-hard [11]에 속하는 문제로서, 문제크기가 증가함에 따라 최적해를 구하는 시간이 지수적으로 증가하게된다. 본 논문의 이전 연구인 [24]에서 backtracking에 근거한 최적값 탐색 알고리즘을 제안하고 실험을 행한 바가 있으나, 어느정도 이상의 경로길이부터는, 부적격한 해를 사전에 제거하더라도, 수행시간이 실용적이지 못할 정도로 오래 걸리게 되었다. IP도 근본적으로는 지수함수적인 복잡도의 알고리즘에 의존하지만, 최근의 상당한 기술진전으로 문제해결의 범위가 넓어지고 있다 [12]. 그러므로, 단순한 완전탐색에 의한 방법보다 훨씬 긴 길이의 경로에 대한 최적 색인구성을 얻을 수 있을 것으로 기대된다. 뿐만 아니라 이를 통하여 다양한 문제 크기에 대한 최적 해를 분석할

수 있으므로, 향후, 짧은 시간내에 근접해를 찾고자 하는 heuristic 알고리즘의 고안에도 도움이 될 것이다.

IP의 이용에 대한 또 다른 동기는 최근 데이타베이스의 최적화 분야에서 연구되는 최적화 전략의 다변화에서 찾을 수 있다 [2,4,19]. 여러 응용분야에서 요구되는 다양한 최적화 문제 유형들에 대하여, 데이타베이스 시스템은 문제 특성에 부합되는 최적화전략을 적용하는 것이 바람직하다 [19]. 이러한 접근방식은 확장형 데이타베이스 시스템에 특히 적용가능할 것이다. 즉, IP 패키지 등의 수행모듈이 데이타베이스 설계도구에 연결되어, 물리적 설계 등의 최적화 작업에 이용될 수 있을 것이다.

한편, 본 논문에서 취급되는 최적 색인구성 문제는 기존 데이타베이스 시스템에서 물리적 데이타베이스 설계시의 색인선택 문제와도 관련이 있으나 [10], 단순 애트리뷰트 색인만을 고려하는 기존 문제보다 더 복잡하며, 나아가 클래스 애트리뷰트 계층에서의 색인구성을 위한 첫 단계로서 보다 큰 의미가 있다고 하겠다.

본 논문은 다음과 같이 구성되어 있다. 2절에서는 기본 용어를 기술하고 문제를 정의한다. 3절에서는 질의처리 방식과 비용함수를 소개하고, Integer Programming으로 문제를 재구성하는 과정을 설명한다. 4절에서는 실험방법과 결과를 기술한 후, 5절에서 본 논문을 맺는다.

2. 용어 및 문제 정의

$C_1,C_2,...,C_{n+1}$을 클래스라 하고, $A_1,A_2,...,A_n$을 애트리뷰트라 할때, 경로 (혹은 경로식)은 $C_1.A_1.A_2.\cdots.A_n$으로 표시된다. 이 때, A_i는 C_i의 참조 애트리뷰트라 불리우며 C_{i+1}을 도메인으로 한다,i=1,...,n [3,20]. 임의의 경로 (혹은 그 경로전체에 정의된 색인)의 길이는 참조 애트리뷰트의 수로 정의된다.

본 논문에서는 중첩색인 (Nested index) 및 경로색인 (Path index)의 두가지 종류의 중첩 애트리뷰트 색인을 최적 색인구성의 요소로 고려한다 [3]. 길이가 1인 경우 중첩 색인과 경로 색인은 동일하다. 다른 종류의 중첩 애트리뷰트 색인은 이들을 이용한 형태로 표현될수 있다. 즉, [20]의 색인은 길이가 1인 중첩 (또는 경로)색인의 연속된 형태로 볼 수 있고, [15]의 ASR은 경로색인을 경로의 양 방향으로 정의한 형태이다. 하나의 경로상의 클래스 C_i와 C_j,i<j, 에 대하여, C_j의 객체식별자를 키로 하고 C_i의 객체 식별자 (경로 색인의 경우, C_i부터 C_{j-1}의 객체식별자의 리스트) 를 값으로 하는 색인을 $I(C_i,C_j)$로 표기한다. 예를 들어, 앞서의 경로식 "Vehicle.manufacturer.president.name"에 대한 색인은 I(Vehicle, Name)으로 나타낸다 ("name"의 도메인이 Name이라는 클래스로 가정). 두 경로 (혹은 색인)가 하나 이상의 참조 애트리뷰트를 공유할 때 이들은 서로 겹친다고 말한다. 겹친 색인을 포함하는 색인구성은 갱신질의시 유지비용 측면이나 저장공간측면에서 부담을 가중시키기 때문에 본 논문에서도 이를 고려하지 않는다 [6,15]. 중첩 애트리뷰트 색인의 구성은 다음과 같이 정의한다.

[정의1] 한 경로에 대한 색인구성은 경로상에서 정의된 서로 겹침이 없는 중첩 애트리뷰트 색인들의 집합이다.

길이 n인 경로에 대해 조합 가능한 색인구성의 수를 알아보자. 만약, 길이 1인 색인으로만 구성한다고 하면, 각 참조 애트리뷰트마다 하나의 색인을 둘수 있으므로 가능한 색인의 수는 n이고, 총 가능한 색인구성 수는 2^n이다. 모든 길이의 색인들을 고려한다면, 길이가 2이상인 부경로마다 중첩색인 또는 경로색인이 가능하므로, 정의가능한 색인 수는 $n + 2((n-1) + (n-2) + \cdots + 1) = n^2$이다. 총 가능한 색인구성의 수는 $2*3^{n-1}$으로 계산된다 (자세한 유도과정은 부록에 기술되어 있다.). 이 식을 보면, 경로 길이가 1씩 길어질 때 조합 가능한 색인구성의 수는 세배씩 늘어나는, 탐색공간이 방대한 문제임을 알 수있다.

본 논문에서는 색인 화일의 저장공간에 제한을 두고 있으므로, 적격한 색인구성은 색인 화일 크기의 합이 주어진 M페이지를 초과하지 않는 색인구성으로 이루어진다. 따라서, 최적 색인구성 문제는 한 경로상에서 조합 가능한 모든 적격한 색인구성 가운데, 주어진 검색및 갱신질의 처리비용을 최소화하게 하는 구성을 찾는 문제로 정의된다. 서술적으로 표현된 이 정의는 다음장에서 비용모델 및 수학적인 형태인 IP 모델로 표현된다.

3. 색인 구성

3.1 질의처리및 비용함수

질의 처리와 비용함수의 정의에 있어서, 본 논문에서는 다음과 같은 가정을 전제로 한다.

- 경로상의 클래스들은 그 클래스에 직접 속한 객체와 상속계층에 따른 그 하위 클래스에 속한 객체를 포함한다.
- 각 참조 애트리뷰트는 단 방향이다. 즉, 참조되는 클래스에 역방향 애트리뷰트는 가정하지 않는다. 참고로, Ontos와 같은 일부 객체지향 DBMS에서는 역방향 애트리뷰트를 지원하기도 하는데, 역방향 애트리뷰트도 갱신부담이 있기 때문에 신중히

설정되어야 한다. 색인구성 문제는 역방향 애트리뷰트 설정문제를 포함하고 있다.

- 참조 애트리뷰트는 집합값이다.

색인구성에 참고될 workload, 즉, 검색및 갱신질의는 일정기간 데이타베이스 이용 통계를 수집하거나 DBA에 의해 중요하다고 판단되는 것들로 구성되며, 각 검색및 갱신질의에는 발생빈도 혹은 가중치가 부여된다. 본 논문에서는 existentially quantified 후방질의를 고려한다 [15,23]. 즉, 클래스 C_j가 C_i의 중첩 애트리뷰트 A_{j-1}의 도메인이라 할 때, 클래스 C_j의 주어진 k개의 객체 중 일부를 A_{j-1}의 값으로 참조하는 C_i의 객체를 구하는 검색질의로서, $Q(C_i,C_j,k)$, $i < j$, 로 표현하기로 한다. 후방검색은 특히 역방향 애트리뷰트가 제공되지 않는 경우에 그 처리비용이 클뿐 아니라, 복합객체의 록킹, 버전등의 지원을 위해 자주 이용되는 형태이므로 신속한 처리가 요구되고 색인의 활용을 필요로 하는 질의이다 [18].

갱신질의는 경로상의 한 참조 애트리뷰트의 값을 수정하는 것으로, $U(C_k)$로 나타낸다. $U(C_k)$는 경로상의 모든 색인 $I(C_s,C_t)$,$s \le k < t$,의 수정을 야기한다. 갱신되는 데이타 페이지의 수정비용은 색인구성과 무관하므로 고려하지 않는다.

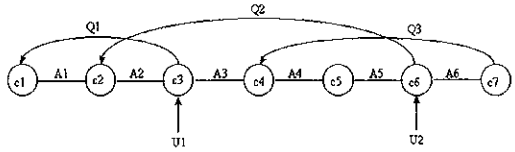


그림 1 경로와 workload의 예

그림 1은 2절에서의 표기와 위의 검색질의 및 갱신질의의 예로서 경로와 workload를 보여주고 있다. 이 그림에서 각 클래스 C_i의 참조 애트리뷰트 A_i는 클래스 C_{i+1}을 도메인으로 하고 있다, $1 \le i \le 6$. 검색질의의 한 예로, Q_1은 클래스 C_3의 주어진 객체 중 일부를 중첩 애트리뷰트 A_2의 값으로 참조하는 C_1의 객체를 구하는 질의이다. 각 질의에 대한 가중치와 색인 화일 저장을 위한 저장한계가 주어졌을 때, 최적 색인 구성은 각 질의의 처리비용을 가장 최소화하는 색인들로 구성하는 것이다. 예들 들면, 클래스 C_1과 클래스 C_3에 걸쳐 경로색인을 두고, 클래스 C_4과 클래스 C_6사이에 중첩색인을 둔 색인 구성이 있을 수 있다. 본 논문에서는, 아래에 소개될 비용함수와 3.2절의 문제 재구성을 통하여, 각 색인별로 비용절약 값과 색인크기를 구하여 비용절약 값을 최대화시키는 목표함수를 유도하고, 이를 만족하는 색인들을 구하도록 색인구성 문제를 수학적 형태, 즉, Integer Programming 모델로 표현하고자 한다.

아래에 기본 비용함수들을 요약한다. 보다 상세한 정의와 수식은 [24]에 제시되어 있다. 각 함수의 비용단위는 페이지 수이다.

- $IndexSize(I(C_s,C_t))$: 색인 $I(C_s,C_t)$의 크기.
- $C_{IdxAcc}(I(C_s,C_t), k)$: k개 키로 색인접근 비용.
- $C_{IdxUpd}(I(C_s,C_t),U(C_k))$: 갱신질의 $U(C_k)$에 대한 색인 $I(C_s,C_t)$의 수정비용.
- $C_{BwdAcc}(C_i,C_j,k)$: 후방검색 비용. 즉, 색인을 이용하지 않고 C_j의 k개 객체를 참조하는 C_i의 객체 식별자를 구하는 비용.

중첩 애트리뷰트 색인은 후방검색질의에 주로 이용되고, 또한 본 논문에서는 집합값 참조 애트리뷰트와 객체의 참조공유를 기본 데이타베이스 모델로 하기 때문에, 질의 처리방식은 후방 탐색에 준한 sort-domain 전략을 가정한다 [15,17]. 일반적으로 중첩색인 $I(C_s,C_t)$는 C_t의 한 객체를 참조하는 C_s의 객체를 구하는데 쓰이고, 경로색인 $I(C_s,C_t)$는 C_t의 한 객체를 참조하는 임의의 C_k, $s \le k < t$, 의 객체를 구하는데 유용하다. 임의의 색인구성하에서 질의를 처리하는 비용은 아래와 같이 정의된다

[정의2] 임의의 색인구성 Ψ상에서 질의 $Q(C_i,C_j,k)$의 처리비용:

$$C_{query}(Q,\Psi) = \sum_{I \in \Psi} C_{IdxAcc}(I(C_s,C_t), Ref(C_t,C_j,k)) + \sum_{[l..m]} C_{BwdAcc}(C_l,C_m,Ref(C_m,C_j,k))$$

$I(C_s,C_t)$는 $i \le s < t \le j$인 중첩 또는 경로색인이거나, 혹은 $s \le i < t \le j$의 경로색인이고, [l..m], $i \le l < l \le j$,는 색인이 정의되지 않은 인터벌들이다. $Ref(C_t,C_j,k)$는 C_j의 k개 객체를 참조하는 C_t의 객체수를 나타낸다 [15].

2절에서 정의한 최적 색인구성 문제는 아래의 식을 최소화하는 색인구성을 구하는 것이다.

[정의3] 임의의 색인구성 Ψ상에서, 주어진 workload W의 검색및 갱신질의 처리비용:

$$C_{total}(W,\Psi) = \sum_{Q \in W} f_Q * C_{query}(Q,\Psi) + \sum_{U \in W} \sum_{I \in \Psi} f_U * C_{IdxUpd}(I,U)$$

U는 Ψ의 색인 I의 수정을 발생시키는 W의 갱신질

의이다. f_Q, f_U는 검색및 갱신 질의의 가중치이다.

3.2 문제의 재구성

전절의 질의모델과 질의처리 방식에 의거하여, 하나의 질의 및 workload의 총 질의에 대한 임의의 중첩 애트리뷰트 색인의 비용절약을 아래와 같이 정의한다.

[정의4] 질의 $Q(C_i,C_j,k)$에 대한 임의의 색인 $I(C_s,C_t)$의 비용절약:

$$C_{saving}(I, Q) = C_{BwdAcc}(C_m, C_t, Ref(C_t, C_j, k)) - C_{IdxAcc}(I, Ref(C_t, C_j, k))$$

여기서 M은, $i \le s < t \le j$이면 $m = s$, 혹은 I가 $s < i < t \le j$의 경로색인이면 $m = i$이다.

[정의5] workload W의 모든 질의에 대한 임의의 색인 $I(C_s,C_t)$의 총 비용절약:

$$C_{saving}(I, W) = \sum_{Q \in T} f_Q * C_{saving}(I, Q) - \sum_{U \in W} f_U * C_{IdxUpd}(I, U)$$

T는 색인 $I(C_s,C_t)$와 겹치는 W의 검색질의 $Q(C_i, C_j,k)$로서, $i \le s < t \le j$ 혹은 $I(C_s,C_t)$가 경로색인이면 $s \le i < t \le j$을 만족한다. f_Q, f_U는 검색및 갱신 질의의 가중치이다.

2장에서 정의된 최적 색인구성 문제는, 색인의 비용절약을 중심으로 색인의 비용절약의 총 합이 최대가 되는 적격한 색인구성을 구하는 문제로 다시 정의될수 있다. 아래의 정리1은 이와같은 문제 재구성이 옳음를 밝혀준다. 주어진 workload W에 대해서, L_1과 L_2는 각각 색인저장 한계 M을 만족하면서, $C_{total}(W, \Psi)$을 최소화하고 $\sum_{I \in \Psi} C_{saving}(I, W)$를 최대화하는 색인구성 Ψ를 구하는 문제라고 하자.

[정리1] L_2에 대한 최적 해결은 L_1에 대한 최적해결과 일치한다.

증명: 비용함수를 이용하여, L_1의 임의의 문제 g_1을 L_2 형태로 g_2로 reduce했을 때, g의 최적 해결이, g_1의 최적 해결과 일치함을 보인다. Ψ^*을 g_2의 최적 해결이라고 하면, 모든 Ψ, $\Psi^* \neq \Psi$,에 대하여 다음이 성립한다.

$$\sum_{I \in \Psi^*} C_{saving}(I, W) > \sum_{I \in \Psi} C_{saving}(I, W)$$

만약, Ψ^*가 g_1의 최적해가 아니라고 하자. 그러면, 아래를 만족하는 Ψ', $\Psi' \neq \Psi^*$가 반드시 존재한다.

$$C_{total}(W, \Psi') < C_{total}(W, \Psi^*)$$

A, B, Z를 색인 집합이라고 하면, 일반적으로, $\Psi' = Z \cup A$, $\Psi^* = Z \cup B$로 볼수 있다. Z는 Ψ^*와 Ψ'에 공통되는 색인 집합이다. S는 $A \cup B$의 적어도 하나의 색인과 겹치는 workload W의 검색질의 $Q(C_i,C_j,k)$의 집합이라 하자. 정의 4에 의해 $C_{total}(W, \Psi^*)$와 $C_{total}(W, \Psi')$은 다음과 같이 전개된다.

$$C_{total}(W, \Psi') = \sum_{q \in W-S} f_q * C_{query}(q, \Psi') + \sum_{q \in S} f_q * C_{query}(q, \Psi') + \sum_{u \in W, I \in \Psi'} f_u * C_{IdxUpd}(I, u)$$

$$C_{total}(W, \Psi^*) = \sum_{q \in W-S} f_q * C_{query}(q, \Psi^*) + \sum_{q \in S} f_q * C_{query}(q, \Psi^*) + \sum_{u \in W, I \in \Psi^*} f_u * C_{IdxUpd}(I, u)$$

위 식에서 첫번째 항들은 $W - S$의 질의를 처리하는 비용으로서, $A \cup B$와 겹침이 없으므로, 서로 동일하다. 각 두번째 항은 S의 각 질의별로 색인접근 비용 (색인이 이용될 때)과 후방탐색 비용으로 구성된다. 두 식에 (2)에 대입하고 양변에 $\sum_{q \in S} f_q * C_{BwdAcc}(C_i,C_j,k)$를 제하면 후방탐색 비용이 제거되어, 정의4와 정의 5에 의하여 색인접근 비용은 비용절약식으로 다음과 같이 정리된다.

$$\sum_{I \in A} C_{saving}(I, W) > \sum_{I \in B} C_{saving}(I, W)$$

여기에 $\sum_{I \in Z} C_{saving}(I, W)$을 더하면, 다음과 같다.

$$\sum_{I \in \Psi^*} C_{saving}(I, W) > \sum_{I \in \Psi'} C_{saving}(I, W)$$

위 식 (3)은 (1)의 전제, 즉, 색인구성 Ψ^*가 g_2의 최적 색인이라는 전제에 모순된다. 그러므로, L_2로 유도된 문제의 최적해는 반드시 원래의 문제 L_1의 최적해와 일치한다.

Q.E.D.

3.3 Integer Programming Model

정리1로부터, 새로 유도된 색인구성 문제에 주어지는 것은 각 중첩 애트리뷰트 색인의 비용절약과 화일 크기, 그리고 색인 저장공간 한계값이다. 이렇게 재구성된 문제는 색인구성에 있어서 색인간의 비겹침 제약성을 제외하고는 0/1 knapsack 문제 [11]와 유사함을 알 수 있다.

[정리2] 하나의 경로에 대한 최적 색인구성을 구하는 문제는 NP-hard이다.

증명: 이미 NP-hard로 알려진 0/1 knapsack 문제 [

11]가 최적 색인구성 문제의 특수한 경우 (즉, 길이 1인 색인들로 조합 가능한 최적 색인구성)로 reduce됨을 간단히 보일 수 있다 [24].

NP 문제를 포함한 많은 응용 시스템의 최적화 문제들의 신속한 해결을 위해 OR 분야에서는 여러 기법들이 개발되어 이용되고 있다. 그 중 많이 이용되는 형태가 Linear Programming인데 이는 문제를 수학적으로 표현하였을 때, 각 수학적 함수가 선형 함수 (linear function)인 경우를 말한다. Integer Programming 문제는 그 중에서도 각 결정변수의 값이 정수인 문제를 말한다. 나아가, 각 결정변수의 값이 0 또는 1로 국한된 문제를 Integer Programming의 수학적으로 표현하였을 때, 이를 Binary Integer Programming (BIP) 모델이라고 한다 [12]. 복잡한 문제를 Integer Programming으로 표현할 수 있다면, 이 분야에서 연구된 많은 최적화 관련 기법을 적용할 수 있다. 예를 들면, simplex 등을 이용한 기법은 최근 여러 상용 패키지에 응용되어, 매우 복잡하고 대규모인 문제들에 대해서도 적절한 시간내에 최적 혹은 최적값에 근사한 해결을 제시할 능력을 갖추고 있다 [12,9]. 따라서, 정형화된 문제를 Integer Programming의 수학적 형태에 맞게 표현하는 것이 어려우면서도 중요한 작업이라 하겠다.

본 논문의 주제인 최적 색인구성 문제는 각 색인을 변수로 볼때, 임의의 색인을 최적 색인구성에 포함시킬지 여부를 결정하는 것이므로 BIP 문제에 해당한다. Integer Programming 모델은 각 결정 변수의 설정, 최대화 목적 함수, 그리고 제약조건을 명시하여야 한다. 최적 색인구성 문제를 BIP 모델로 표현하기 위해 각 색인에 대해 변수를 할당하고, 최대화 목적 함수로서 비용절약의 합, 그리고 제약조건으로서 저장공간과 색인 사이의 겹침정보를 나타내고자 한다. 이 중에서 어려운 점은 겹침정보의 효율적인 표현에 있다.

먼저 다음과 같이 변수를 설정하였다. 길이가 n인 경로에서 길이가 1인 색인 $I(C_i,C_{i+1})$에 대해 변수 $z_{i,i+1}$를 지정한다, $1 \le i \le n$. $x_{i,j}$,$y_{i,j}$는 각기 중첩색인과 경로색인을 나타내는 변수로서 i = 1,2,...,n-1 그리고 j = i+1,...,n+1이다. $P_{i,i,j}$와 $S_{i,i,j}$는 각각 유형 t (z, x, y중 하나)인 색인 $I(C_i,C_j)$의 비용절약과 화일크기를 나타낸다. 최적 색인구성 문제의 BIP 모델은 다음과 같다.

Objective function: Maximize

$$\sum_{i=1}^{n} P_{z,i,i+1} * z_{i,i+1} + \sum_{i=1}^{n-1} \sum_{j=i+2}^{n+1} (P_{x,i,j} * x_{i,j} + P_{y,i,j} * y_{i,j})$$

Subject to:

$$\sum_{i=1}^{n} S_{z,i,i+1} * z_{i,i+1} + \sum_{i=1}^{n-1} \sum_{j=i+2}^{n+1} (S_{x,i,j} * x_{i,j} + S_{y,i,j} * y_{i,j}) \le M$$

$$z_{k,k+1} + \sum_{i=1}^{k-1} \sum_{j=k+1}^{n+1} (x_{i,j} + y_{i,j}) + \sum_{j=k+2}^{n+1} (x_{k,j} + y_{k,j}) \le 1, \quad \text{for } k = 1,2,\ldots,n \text{ and } z_{k,k+1}, x_{i,j}, y_{i,j} \text{ are binary integers,}$$

$$\text{for } k = 1,2,\ldots,n,\ i = 1,2,\ldots,n-1, \text{ and } j = i+2,\ldots,n+1.$$

위의 BIP 모델식을 간단히 설명한다. 목적 함수는 변수값이 1인 해당 색인들의 비용절약 합을 최대화하는 것이다. 첫항은 길이 1인 색인을, 둘째항은 길이 2이상인 중첩및 경로색인의 비용절약을 나타내고 있다. 첫번째 제약식은 색인구성에 포함되는 색인 (변수값이 1)의 화일크기의 합을 M이하로 제한한다. 다음으로, 비겹침 제약 조건식에 이어 각 변수는 0 혹은 1의 이진 정수로 제한된다. 비겹침 제약은 경로길이 n에 대하여 n개의 식으로 구성되어 있다. 각각은 하나의 길이 1인 색인과 이것에 겹치는 모든 길이 2 이상의 중첩 및 경로 색인들에 해당하는 변수들의 합이 1 이하로 국한되어 있다. 즉, 최대한 하나의 변수만 1이 될 수 있다. 비겹침 제약을 표현하는 다른 방법으로, 겹침이 있는 모든 색인들에 해당하는 변수의 쌍을 열거하고 각 쌍의 합을 1 이하로 국한시킬 수도 있을 것이나. 그러나, 이렇게 할 경우, 각 쌍을 일일이 생성하는 것이 용이하지 않을 뿐아니라, 그 수효도 방대해짐으로써, IP 패키지의 성능저하를 초래할 우려가 있다. 위의 식은 다음절에 설명될 실험에서, 실제 IP 패키지의 입력을 생성하는 프로그램으로 구현되었다.

4. 실험 및 분석

NP-hard 문제일지라도 문제가 일정 크기이내로 국한되거나 작은 문제일 경우에는 완전탐색 알고리즘으로 적정 시간내에 최적해결을 구할 수 있을 것이다. 이 때 backtracking이나 branch-and-bound 기법 [13]을 이용하면 탐색공간을 상당히 줄일 수 있다. [6,24]에 이들 기법을 이용한 색인구성 알고리즘들이 소개된 바 있다.

색인구성은 주기적으로 시스템을 정지시킨 상태에서 수행하므로, 시스템 성능에 직접적으로 큰 영향은 없으나 너무 장시간 동안의 수행은 시스템 가용시간을 줄이기 때문에 바람직하지 못하다. 또, 경로 길이가 매우 길 때에는 IP 패키지라 할지라도 근본적으로 지수함수의 알고리즘이기 때문에 긴 수행시간을 요할 수 있다. 본 절에서는 backtracking에 기초한 완전탐색 알고리즘[24]과 잘 알려진 IP 패키지인 CPLEX Version 2.1 [9]을 이용하여, 하나의 경로에 대한 최적 색인구성을 구하는 문제의 복잡도를 파악하고 3.4절에서 제시한 BIP 모델식의 성능을 가늠하기 위한 실험과 그 결과를 기술한다. 실험은 SPARC Sun 4/280 SunOS 4.1.1상에서 수행되

었고, backtracking 알고리즘과 CPLEX packge의 입력을 생성하는 프로그램은 C 언어로 구현되었다.

실험의 주요 변수는 경로의 길이로서, 5에서 50까지 변화시켰다. 하나의 경로 길이에 대하여 아래의 동적 변수를 통해 10개의 문제를 생성시켜 그 평균 수행시간을 측정하였다.

- 한 클래스의 객체 수: 1,000 ~ 10,000.
- 각 클래스에서 객체의 크기: 50 ~ 500 bytes.
- workload의 총 질의 수: 10 ~ 30. 갱신 질의의 갯수는 총 질의수의 30%이내로 임의 결정되고 나머지는 검색질의이다.
- 참조 애트리뷰트의 집합값 갯수: (1 ~ 5) 혹은 (10 ~ 20)의 범위.
- 각 검색질의 $Q(C_i,C_j,k)$는 0 ~ 1 사이의 빈도치 f_q를 갖고, C_i와 C_j, $i < j$, 는 경로내 임의의 두 클래스이다. k는 클래스 C_j의 객체수의 0.1% ~ 1%사이의 값을 갖는다.
- 각 갱신질의 $U(C_k)$도 빈도치 f_u를 갖는데 갱신부담을 살피기 위하여 낮은 영역 (0 ~ 0.2) 혹은 높은 영역 (0.6 ~ 0.8)에서 임의 선택된다. C_k는 경로내에서 임의로 선택된 클래스이다.
- 색인의 저장공간 한계는 경로상의 총 데이타 크기의 30%로 한다 [10].

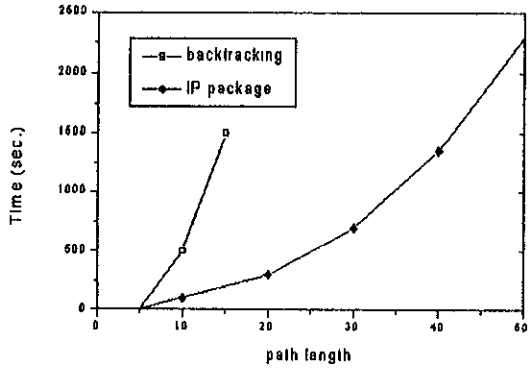


그림 2 경로 길이당 10개의 문제에 대한 평균 수행시간

이상과 같은 실험변수로부터 생성된 workload의 예가 부록2에 간략히 소개되어있다. 이러한 형태의 데이타가 CPLEX package와 backtracking 알고리즘으로 입력된다. 그림 2에 각 경로 길이당 10개씩 생성된 문제들에 대한 backtracking 알고리즘과 CPLEX 패키지의 평균수행시간이 나타나있다. 가로축은 경로 길이를 나타내고 세로축은 수행시간 (단위: 초)을 나타낸다. backtracking 알고리즘은 경로길이 15이상부터는 과다한 수행시간으로 측정이 곤란하였다. CPLEX는 경로길이 50까지도 적절한 시간내에 문제를 해결하였고 (40분 이내), 본 논문에서 제시한 BIP 모델이 색인의 최적구성 문제로 잘 표현되었음을 보여주었다. 경로길이 50 이상의 문제들에 대해서 많은 경우에 1시간 이상의 계산시간이 소요되었다.

실험을 통해 여러 경로 길이에서 최적 색인구성을 관찰할 수 있었다. 표 1에서는 경로길이를 6에서 20까지 변화시키면서 각 경우에 10회의 worklaod를 생성/실행시켜 최적 색인구성에 포함된 색인의 평균갯수, 그때 길이별 색인의 분포, 그리고 길이가 2 이상인 경우에 중첩색인의 비율을 종합적으로 도표화하였다. 참고로, 앞서 그림 2에서의 평균시간 측정에서는 본 논문에서 제안한 BIP 모델의 성능을 실제 package를 통하여 알아보기 위하여 긴 경로길이까지 대상으로 하였다. 그러나 실제 응용시스템에서는 길이 20이상은 비교적 드물것으로 상정하여 경로길이 20까지만 표로써 나타내었다. 그밖에 실험의 결과가 명백한 경우 (예를 들면, 갱신질의의 빈도치 변화에 따른 색인구성의 분포), 도표를 생략하였다. 주요 관찰내용은 다음과 같다.

표 1 경로길이별 최적색인구성의 분포 (길이당 10회 실험의 평균)

경로길이	최적구성의 평균 색인갯수	길이가 1인 색인의 비율 (%)	길이가 2인 색인의 비율 (%)	길이가 3인 색인의 비율 (%)	길이가 4인 색인의 비율 (%)	길이>1일때 중첩색인의 비율 (%)
6	3	60	26	9	4	52
8	4	44	30	19	5	56
10	5	34	31	25	7	55
12	8	31	31	28	8	59
14	6	19	30	32	12	61
16	7	16	27	34	15	57
18	7	11	25	35	15	55
20	9	8	33	34	19	59

- 색인의 길이: 대부분의 경로길이에서 최적 색인구성은 비교적 짧은 색인 (길이 3 혹은 4 이하) 들로 이루어져 있음을 확인하였다. 특히 질의 갯수가 보통이상 (10개 이상)이고 경로길이가 10 이상부터 두드러졌다. 이 사실은 [3]의 권고와도 일치한다고 할 수 있다. 이는 또한 본 실험에서 질의 및 데이타의 속성들이 경로 전체에 결

처 균등하게 제시되었기 때문일 수도 있을 것이다. 따라서, 경로의 일부에 질의가 집중되는 workload에서는 긴 길이의 색인이 선택될 수도 있을 것이다.

• 색인의 종류: 많은 경우에 경로 색인보다는 중첩색인이 약간 많이 포함된 것을 발견하였는데 이는 중첩색인의 크기가 경로색인보다 더 작기 때문에 색인당 비용절약을 높히는 원인으로 작용했을 것으로 추정된다.

• 갱신 질의의 영향: 갱신질의가 발생한 클래스를 포함하는 색인은 종류에 관계없이 최적 색인 구성에 대부분 포함되지 못하였다. 높은 색인 유지비용을 재확인시켜 주는 것이다.

• 비용절약값의 관계: 색인의 비용절약과 색인의 크기 비율 순으로 색인들을 정렬하였을 때, 많은 경우에 짧은 길이의 중첩 색인이 긴 길이의 경로 색인보다 순서의 앞부분에 위치하고 있었으며, 일반적인 workload 하에서 최적 색인구성은 이 비율이 높은 색인들로 주로 이루어지고 있었다.

이상의 관찰을 종합하면, 균등 workload 환경하에서는, 갱신 질의가 발생하는 클래스를 기준으로 경로를 분리하여, 각 부경로별로 색인의 비용절약 대 크기의 비율 순으로 색인을 구성하는 접근방식이 효과적일 수 있음을 시사하고 있다. 이는 일반적인 0/1 knapsack의 데이타와 달리, 경로의 색인 구성에서는 각 색인의 가치와 크기의 관계가 무관하지 않은, 즉, 색인 구성 문제 특유의 성질이 있음을 의미한다. 이러한 성질에 대한 보다 치밀한 분석이 필요할 것이다. 최적 색인구성에 대한 이와 같은 관찰은 짧은 시간내에 근접해를 구하기 위한 heuristic의 개발에도 유용할 것으로 사료된다.

5. 맺음말

성능 최적화는 객체지향 데이타베이스 시스템의 중요한 연구과제이다. 최근에 제안된 중첩 애트리뷰트 색인은 객체지향 질의의 최적화에 크게 기여할 수 있는 것으로 보고되었다. 그러나, 이들 색인은 기존의 단순 애트리뷰트에 대한 색인에 비해 저장공간 및 갱신유지 비용이 큰 부담이 있다. 또한, 색인의 종류에 따라 검색성능도 달라진다. 그러므로, 중첩 애트리뷰트 색인을 통한 질의처리 잇점이 저장공간 및 갱신유지를 위한 부담으로 인해 상쇄되지 않기 위해서는 색인들이 매우 신중히 구성되어야하며, 효과적인 색인구성 방법에 관한 연구가 필수적이다.

본 논문에서는, 객체지향 데이타베이스에서 클래스 애트리뷰트 계층상의 하나의 경로에 대한 중첩 애트리뷰트 색인의 최적 구성 문제를 연구하였다. 최적 색인구성 문제는 하나의 경로에 대한 검색및 갱신질의들이 주어지고 색인화일에 대한 저장공간 한계가 주어졌을 때에, 질의의 총 처리비용을 최소화할 수 있는 색인구성을 결정하는 문제이다. 분석을 통해, 최적 색인구성 문제는 경로길이가 1씩 증가함에 따라 가능한 색인구성의 수가 세배씩 늘어나는 방대한 문제공간을 갖으며 NP-hard에 속한다는 것을 설명하였다. 비용모델로부터 색인의 비용절약을 유도하여, 비용절약의 총 합이 최대화되는 색인구성을 구하는 문제로 변환하는 방법을 제안하고, 이의 옳음을 증명하였다. 최적 구성을 구하기 위하여 Integer Programming (IP)을 이용하는 방안을 제안하였다. IP는 근본적으로 지수함수의 복잡도를 갖는 알고리즘을 바탕으로하지만, 최근의 상당한 기술진전으로 그 문제해결의 범위가 넓어지고 있다. 그러나, 최적 색인구성 문제를 IP 모델로 표현하는 것은 그리 간단하지 않다. 본 논문에서는, 색인에 대해 결정함수를 부여하고, 최적화 목표함수와 저장공간 및 비겹침 제약조건을 표현하는 방법을 상세히 기술하였다. 이렇게 표현된 IP 모델은, backtracking 기법으로 구현한 알고리즘과 비교한 실험을 통하여, 그 문제표현의 옳음 뿐만 아니라 큰 문제범위에 대해서도 그 문제해결 가능성을 보였다.

향후 연구과제로서, 먼저 본 논문에서 제안한 방법을 바탕으로, 클래스 상속 계층을 고려한 보다 복잡한 질의형태와 색인들을 지원하여, 클래스 애트리뷰트 계층을 대상으로하는 중첩 애트리뷰트 색인의 최적구성 문제가 연구되어야 할 것이다. 이를 위해서는, 클래스 애트리뷰트 계층상의 질의 최적화 문제도 고려되어 할 것이다. 또한, 보다 짧은 시간내에 근접해를 구할 필요가 있는 응용 시스템을 위해서는, 본 논문의 실험을 통해 얻어진 최적색인 구성을 분석하여, 효과적인 heuristic 알고리즘의 개발이 필요할 것이다.

참 고 문 헌

[1] 이 윤준, "객체지향 데이타베이스 시스템," 정보과학회지, 12(3), 1994.

[2] K. Bennett, *et al*, "A Genetic Algorithm for Database Query Optimization," Technical Report 1004, Univ of Wisconsin, 1991

[3] E Bertino and W. Kim, "Indexing Techniques for Queries on Nested Attributes," IEEE TKDE, 1(2), 1989.

[4] E. Bertino, "Optimization of Queries using Nested Indexes," Proc. EDBT, 1990.

[5] E. Bertino, "Index Configuration in Object-Oriented Databases," VLDB Journal, 3, 1994.

[6] S. Choenni *et al*, "On the Selection of Optimal Index Configuration in OO Databases," Proc. Data Engineering, 1994.
[7] R.G.G. Cattell, *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, 1991.
[8] S. Cluet and C. Delobel, "A General Framework for the Optimization of Obejct-Oriented Queries," Proc. ACM SIDMOD Conference, 1992
[9] CPLEX, *Using the CPLEX™ Callable Library and CPLEX™ Mixed Integer Library*, CPLEX Optimization Inc., 1993.
[10] S.Finkelstein *et al*, "Physical Database Design for Relational Databases," ACM TODS, 13(1), 1988.
[11] M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.
[12] F.S.Hillier and G.J.Lieberman, Introduction to Operations Research, McGraw-Hill, 1990.
[13] E.Horowitz and S.Sahni, *Fundamentials of Computer Algorithms*, Computer Science Press, 1978.
[14] Y. Ioannidis and Y. Cha Kang, "Randomized Algorithms for Optimizing Large Join Queries," Proc. ACM SIGMOD Conference, 1990.
[15] A.Kemper and G.Moerkotte, "Access Support in Object Bases," Proc. ACM SIGMOD, 1990.
[16] A.Kemper and G.Moerkotte, "Advanced Query Processing in Object Bases," Proc. VLDB Conference, 1990.
[17] K.C. Kim *et al*, "Acyclic Query Processing in Object-Oriented Databases," Proc. 7th Intl' Conference on E-R Approach, 1989.
[18] W.Kim, *Introduction to Object-Oriented Databases*, The MIT Press, 1990.
[19] R.S.G. Lanzelotte and P.Valdruiez, "Extending the Search Strategy in a Query Optimizer," Proc. of VLDB Conference, 1991.
[20] D. Maier and J. Stein, "Indexing in an Object-Oriented DBMS," Proc. Intl. Workshop on Object-Oriented Database Systems, 1986.
[21] J. Eliot B. Moss, "Working with persistent objects: to swizzle or not to swizzle," COINS Object-riented Lab. TR 90-38, Univ. of Massachusetts at Amherst, 1990.
[22] G.M. Mitchell *et al*, "Obejct-Oriented Query Optimization: What's the Problem?," Technical Report CS-91-41, Brown University, 1991.
[23] J.Orenstein *et al*, "Query processing in the ObjectStore Database System," Proc. ACM SIGMOD, 1992.
[24] S.K.Seo and Y.J.Lee, "Methodology for Optimal Index Configuration in Object-Oriented Database," Technical Report CS-TR-93-80, KAIST, Korea, 1993.

부 록 1. 하나의 경로에 대한 색인구성의 수

길이 n인 경로에 대해 경로의 맨 첫번째 클래스를 루트 클래스라고 하고, $f(k)$는 길이 k의 경로에 대해 조합가능한 색인구성의 수라고 하자.

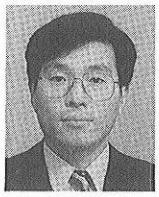
루트 클래스를 포함하는 부경로들은 모두 n개가 있다. 먼저, 그 중에서 경로길이가 1인 부경로에는, 색인을 두는것과 두지 않는 두가지 선택이 있으므로 각각의 선택에 대하여 $f(n-1)$의 색인구성을 조합가능하다. 루트 클래스를 포함하는 길이 2 이상의 임의의 경로에 대해도 중첩색인 혹은 경로색인을 부여할 수 있으므로 마찬가지로 두가지 선택방법이 있다. 즉, 루트 클래스를 포함하는 길이 k, $1 \le k \le n$의 각 부경로에 대해서 $2*f(n-k)$개의 색인구성이 가능하다. $k=n$일 때에는 경로 전체에 중첩색인 혹은 경로 색인을 둘수 있으므로, $f(0)=1$이 된다. 참고로, 색인이 정의되지 않는 부경로를 포함하는 색인구성은 길이가 1인 부경로를 순환적으로 적용함으로서 제대로 고려된다. 이상의 전개를 바탕으로, 길이 n인 경로에 대한 총 색인구성 수 $f(n)$은 다음과 같이 유도된다.

$$
\begin{aligned}
f(n) &= 2*(f(n-1)+f(n-2)+\cdots+f(1)+1) \\
&= 2*(2*f(n-2)+\cdots+f(1)+1)+f(n-2)+\cdots+f(1)+1) \\
&= 2*3*(f(n-2)+\cdots+f(1)+1) \\
&= 2*3^2*(f(n-3)+\cdots+f(1)+1) \\
&\vdots \\
&= 2*3^{n-1}
\end{aligned}
$$

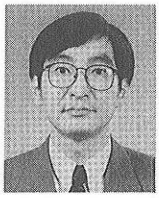
부 록 2. 입력되는 workload의 구성 예

```
#Length of the path
10
#Number of objects of each class
4000
#Object Size
400
#Set Cardinality:[1..5]
1 3 4 3 1 4 1 5 2 3
#Number of retrieval queries and Q(Ci,Cj,k,fq)
15
Q 5 10 8 0.61
Q 8 11 8 0.26
Q 3  6 8 0.77
Q 4  6 9 0.93
```

:
#Number of update queries and $U(C_k, f_u)$
5
U 2 0.17
U 5 0.15
:



서 상 구
1984년 서울대학교 공과대학 전자계산기 공학과 학사. 1986년 한국과학기술원 전산학과 석사. 1986년~1989년 현대전자(주) 소프트웨어 사업본부. 1995년 한국과학기술원 전산학과 박사. 1995~현재 현대전자(주) 소프트웨어연구소 시스템연구실. 관심분야는 OODB, 분산데이타베이스



이 윤 준
1977년 서울대학교 계산통계학과 졸업. 1979년 한국과학기술원 전산학과에서 석사학위 취득. 1983년 France, INPG-ENSIMAG에서 박사학위 취득. 1983년 ~ 1984년 France, IMAG 연구원, 1984년 ~현재 한국과학기술원 전산학과 부교수. 1989년 MCC(미) 초빙연구원. 1990년 CRIN(불) 객원교수. 관심분야는 데이타베이스 시스템, 정보검색, 실시간 데이타베이스 등임.