데이터베이스 색인선택 문제에 대한 Davis-Putnam 기반 최적화 알고리즘의 성능 분석

서 상 구*

Analyzing the Performance of a Davis-Putnam based Optimization Algorithm for the Index Selection Problem of Database Systems

Sang-Koo Seo*

Abstract

In this paper, we analyze the applicability of a general optimization algorithm to a database optimization problem. The index selection problem is the problem to choose a set of indexes for a database in a way that the cost to process queries in the given workload is minimized subject to a given storage space restriction for storing indexes. The problem is well known in database research fields, and many optimization and/or heuristic algorithms have been proposed. Our work differs from previous research in that we formalize the problem in the form of non-linear Integer Programming model, and investigate the feasibility and applicability of a general purpose optimization algorithm, called OPBDP, through experiments. We implemented algorithms to generate workload data sets and problem instances for the experiment. The OPBDP algorithm, which is a non-linear 0-1 Integer Programming problem solver based on Davis-Putnam method, worked generally well for our problem formulation. The experiment result showed various performance characteristics depending on the types of decision variables, variable navigation methods and other algorithm parameters, and indicates the need of further study on the exploitation of the general purpose optimization techniques for the optimization problems in database area.

※ 이 논문은 1999년도 광운대학교 교내학술연구비에 의하여 연구되었음.
* 광운대학교 경영정보학과

1. 서 론

데이터베이스 시스템이 다양한 응용 분야에서 정보시스템의 핵심 요소로 활용되고 데이터베이스 용량 또한 갈수록 방대해짐에 따라 데이터베이스 시스템의 효과적인 운영 및 관리 업무가 그 중요성을 더해가고 있다 [Chaudhuri & Narasayya 1998]. 특히 최적의 시스템 성능을 제공하기 위한 물리적 데이터베이스 설계의 일환으로서 색인의 적절한 생성과 관리는 데이터베이스 관리자의 중요한 작업중 하나일 것이다. 색인은 질의의 처리 속도를 단축시키는 자료구조로서 최근에는 비트맵 색인, 조인 색인, 등 다양한 형태로 데이터베이스 시스템에서 이용되고 있다. 많은 상용 DBMS에서 색인은 Primary Key 또는 Unique로 선언된 애트리뷰트들에 대해서는 기본적으로 생성된다. 그 밖에 대부분의 색인은 자주 이용되거나 중요도가 높은 질의의 검색 조건에 주어진 애트리뷰트들에 대하여 데이터베이스 관리자의 경험과 지식에 의존하여 생성된다. 그러나 색인은 추가적인 저장 공간을 필요로 할 뿐만 아니라 색인된 테이블의 투플들에 대한 추가/삭제/변경 연산시에 색인 파일 자체에 대한 변경을 요구하기 때문에 색인 선택은 신중히 결정되어야 한다.

데이터베이스 색인의 효과적인 선택에 관한 많은 연구가 진행되어 왔으며, 최근에는 상용 DBMS에 하나의 관리도구로서 제공되기도 한다 [Chaudhuri & Narasayya 1997]. 색인 선택 (또는 구성) 문제는 주어진 저장 공간 한도 내에서 작업부하의 질의와 갱신 연산을 처리하는 비용이 가장 최소가 되도록 색인을 구성하는 문제로 정의될 수 있는데, 이 문제는 NP-Complete 문제로 알려져있다 [Comer 1978, Horowitz 1996]. 따라서 테이블 수가 수 십여 개 이상되는 대규모 응용 환경에서 단순한 완전 탐색 방법으로는 적정 시간내에 최적해를 구하는 것은 현실적으로 어렵기 때문에 많은 연구에서는 휴리스틱 알고리즘을 고안하여 근접해를 구하는 기법을 제안하였다 [Chaudhuri & Narasayya 1997, Seo & Lee 1996, Frank *et al* 1992, Finkelstein *et al* 1988].

본 논문에서는 최근 많은 발전이 이루어지고 있는 범용 최적화 기술을 최적 색인 구성 문제에 적용하는 연구 결과를 소개하고자 한다. OR (Operations Research) 분야에서 개발된 많은 최적화 알고리즘이 상용 또는 실험적 패키지 형태로 제공되거나 또는 최적화 문제를 인터넷을 통하여 접수받고 그 해를 제공하는 네트웍 서비스가 제공될 정도로 폭넓게 발전하고 있다 [OTC 2000, Fourer 2000]. 이러한 최적화 기술을 데이터베이스 분야에서 적용 가능성 여부를 살펴보는 시도는 의미가 있는 일이라고 할 것이다.

이를 위하여 본 논문에서는 최적 색인 구성문제를 비선형 정수 계획법 (Nonlinear Integer Programming) 문제로 정형화하고 시험대상 최적화 프로그램에 대한 문제 모형을 생성하는 알고리즘을 고안하였다. 또, 최적화 프로그램의 실행 능력을 다양한 작업부하 (workload)에서 살펴보기 위하여 TPC 벤치마크 [TPC 1999]의 스키마를 확장하여 다양한 작업부하를 생성할 수 있는 데이터 생성기를 구현하였다. 비선형 계획법으로 공식화된 문제에 대하여 문제 크기, 저장공간 제약, 변수 탐색 휴리스틱, 등의 여러 요소들을 변화시켜 최적 색인 구성 문제에 대한 최적화 프로그램의 실행 능력과 타당성 등을 실험을 통하여 분석하였다. 본 연구에서 실험한 최적화 프로그램는 결정 변수의 형태, 변수 탐색 방법, 등의 실험 매개변수에 따라 성능 차이가 있었으나 전체적으로 적정한 최적화 성능을 보였다.

이하 본 논문은 다음과 같이 구성되어 있다.

2장에서는 색인 선택에 관한 관련 연구를 간략히 소개한다. 3장에서는 질의 처리 모형과 색인 구성 문제를 정의하고 4장에서는 본 연구에서 이용된 최적화 알고리즘을 설명한다. 5장에서는 실험 방법와 결과를 설명하고, 6장에서 결론을 맺는다. 부록에는 실험에 이용된 예제 데이터와 실행 스크립트 예를 첨부하였다.

2. 관련 연구

초기 최적 색인 선택 연구에서는 하나의 테이블을 대상으로 Greedy접근 [Horowitz 1996]에 근거한 ADD and/or DROP 방식의 휴리스틱 알고리즘이 제안되었다 [Comer 1978, Ip 1983]. ADD 방식은 최초의 공백 또는 주어진 개수 만큼의 색인 집합으로부터 출발하여 비용절감 효과가 큰 색인들을 점진적으로 추가하는 방식이고, Drop 방식은 그 반대로 모든 색인 집합으로부터 효과가 작은 색인들을 차례로 제거해 나가는 방식이다. 둘 이상의 테이블에 대한 색인 구성은 두 테이블 간의 조인 연산 방법에 따라 한 테이블에서 선택된 색인이 다른 테이블의 색인 선택에 영향을 줄 수 있기 때문에 보다 복잡하다. Whang *et al* [1984]에서는 일부 조인 연산간의 "분리성" 특성을 이용하여 색인 선택방법을 제시한 바 있다. Finkelstein *et al* [1988]에서는 최소단위 색인 집합 개념을 이용하여 처리비용 함수의 재계산 횟수를 줄이는 기법을 제안하였다.

대부분의 연구에서는 색인 선택을 결정할 때에 필요한 질의 처리 비용을 색인 선택 알고리즘에 의하여 계산하였으나, 이는 실제로 질의 처리기가 생성하는 비용과는 불일치할 수 있는 문제점이 있다. Chaudhuri & Narasayya [1997]에서는 실제 DBMS의 비용기반 질의 최적기가 생성한 비용값을 이용하는 색인 선택 방법을 연구하였다. 이 연구에서는 작업부하로 주어진 각 질의들에 대하여, 각 질의의 처리비용을 가장 최소화하는 색인들을 우선 구하여 이들을 모두 합하여 후보 색인 집합으로 한다. 이 후보 색인 집합으로부터 비용 절감이 큰 색인 순으로 전체 색인 집합을 만들어 나간다. 그러나, 이 방법에서는 최적 색인 집합에 속하는 색인이 후보 색인 집합에서 제외될 수 있는 문제점이 있다. 또한, 저장공간에 따른 제약이 아닌 색인의 개수를 제약조건으로 하고 있다. 이것은 색인크기가 모두 동일하다고 가정하는 의미로서, 실제 DBMS 운영 환경과는 일치하지 않으며, 구해진 해가 최적해와 큰 차이를 보일 수도 있는 문제점이 있다.

본 연구에서는 색인 구성을 위한 근사해가 아닌 최적해를 구하고자 하는 일환으로 비선형 계획법 (NLP: Non-Linear Programming)으로 색인 구성 문제를 정형화하고 적절한 최적화 프로그램을 활용하는 접근 방법이 기존 연구와의 차이점이라고 하겠다.

3. 색인 구성 문제

3.1 질의 처리 모델

DBMS의 질의 최적기는 주어진 질의를 가장 적은 비용으로 처리할 수 있는 실행 계획을 생성한다. 최근 대부분 DBMS들은 기존의 휴리스틱 방법 뿐 아니라 예상 처리비용을 비교하여 최적 실행계획을 생성하는 비용 기반 최적화 (Cost-based Optimization)에 기반을 두고 있다. 이 때 관련 테이블의 정렬 여부, WHERE절에 주어진 검색조건들의 선별도 (selectivity), 관련 애트리뷰트에 대한 색인의 존재여부, 애트리뷰트 값의 분포 등의 정보를 고려하게 된다. 이 중 가장 중요한 결정요소는 색인의 존재여부

이다. 색인은 클러스터 또는 비 클러스터 색인이거나, 단일 컬럼 또는 복수 컬럼 색인일 수도 있고, 일반적인 B+-tree 색인, 비트맵 색인, 조인 색인 등일 수도 있다. WHERE조건절의 여러 애트리뷰트들에 대하여 색인들이 존재할 경우 질의 최적기는 각 각의 색인 접근 결과를 결합 (논리 곱, 합, 등) 처리하는 기능도 제공되고 있다. 본 논문에서는 [Chaudhuri & Narassaya 1997]에서와 같이 예상 질의처리 비용은 색인선택 알고리즘이 계산하지 않고 질의 처리기의 계산 결과를 활용하는 것으로 가정한다.

3.2 작업 부하 모델

작업부하는 최적 색인 선택 작업의 입력으로 주어지는 데이터를 뜻한다. 본 연구에서의 작업부하는 색인의 이용성 또는 변경에 영향을 주는 SELECT, INSERT, UPDATE, DELETE 등의 SQL 데이터 조작어로서, 다음과 같이 검색질의 집합과 변경연산 집합으로 구성된다고 가정한다. 각 검색질의는 그 질의의 WHERE절에 이용되는 애트리뷰트들의 집합을 갖는다. 변경 연산에는 검색 질의가 포함될 수도 있다. 편의상 변경 연산에 포함된 검색 질의 부분을 별도의 검색 질의로 간주하면, 각 변경 연산도 변경되는 애트리뷰트들의 집합을 갖는 것으로 표현될 수 있다. 이들은 2차원 배열로 표현 가능하다. 행 i는 애트리뷰트 A_i이고 열 j는 검색 질의 Q_j (또는 변경 연산 U_j)라고 하면, i행j열의 값은 애트리뷰트 A_i가 질의 Q_j에 이용되면 (또는, 변경연산 U_j에 포함되면) 1, 그렇지 않으면 0의 값을 갖는 것으로 생각할 수 있다 (아래 (그림 1) 참조).

성능평가 실험에서 입력 데이터는 이 같은 방식으로 생성된다. 행열의 한 행에 있는 1의 수는 하나의 애트리뷰트가 작업 부하의 질의들에 이용되는 **빈도**를 나타낸다. 또한 한 열에 있는 1의 수는 하나의 질의에 이용되는 애트리뷰트의 수로서 그 질의의 **복잡도**라고 할 수 있다. 예를 들어, 성능평가 기관인TPC의 데이터 웨어하우징 분야 TPC-H 벤치마크 [TPC 1999]의 작업부하는 모두 7개의 테이블, 총 61개의 애트리뷰트 중 39개의 색인 대상 애트리뷰트(WHERE절 기준)들로 구성되어 있으며, 이 데이터베이스에 대하여 22개의 검색질의와 2개의 삽입/삭제 연산으로 구성되어 있다. 본 연구에서는 TPC-H의 작업부하를 확장하여 다양한 크기의 입력 데이터를 생성하는 프로그램을 구현하였다.

본 연구의 실험에서 이용되는 데이터 생성기로 출력한 질의와 애트리뷰트 작업부하의 예제이다. 모두 10개의 검색 질의와 20개의 색인 대상 애트리뷰트들에 대하여 비균등 분포값을 적용하여 생성하였다. 예를 들면 네번째 열의 질의는 5, 6, 9, 11, 14, 17, 20번째 애트리뷰트를 WHERE 조건절에 포함하며, 네번째 행의 애트리뷰트는 2, 6, 8번째 질의에 이용된다고 하는 의미의 작업부하를 보여주고 있다.

```
0000010000
0010000010
0100000010
0100010100
0001000010
0011001000
0110010000
0010000000
0001000011
0010000010
0101010000
0010000000
0010000001
0111000011
0110001000
1100010010
0111100100
0010000000
0010000000
0101000000
```

(그림 1) 질의와 애트리뷰트 작업부하 모델

질의 또는 변경 연산에는 실행빈도(frequency) 값이 연관되어 있는 것으로 일반적으로 가정된다. 실행 빈도가 높은 질의일수록 전체 시스템의 처리 성능에 큰 영향을 주기 때문에 이들 질의의 처리에 이용되는 색인은 그 이용 가치가 상대적으로 높다고 볼 수 있다. 또 검색질의, 즉, SELECT 문은 AND , OR 등의 논리연산자로 연결된 검색조건들로 가정된다. 본 논문에서는 앞서의 가정과 마찬가지로, 작업부하 각 질의의 실행빈도와 질의 내부구성은 직접 모형하지 않고, 질의 최적기에 의하여 값이 주어지는 것으로 가정한다.

3.3 색인구성 문제 모형

최적 색인구성 문제는 가능한 모든 색인구성 가운데 작업부하의 모든 검색질의의 처리비용과 모든 변경연산에 대한 색인갱신 비용의 합이 최소가 되는 색인 집합 (또는 구성) Ψ를 구하는 문제로서 다음과 같이 식으로 나타낼 수 있다. (아래 <표 1>에 주어진 기호와 표기법 참조.)

Find an index set Ψ such that

$$\sum_{Q\in W} QueryCost(\Psi, Q) + \sum_{U\in W} IndexUpdateCost(\Psi, U)$$

is minimized (1)

subject to $\sum_{I\in\Psi} Size(I) \le M$ (2)

작업 부하의 애트리뷰트 수가 모두 n개라고 할 때 색인 집합의 단순 나열식 방법으로는 모두 2^n 만큼의 애트리뷰트 집합이 가능하다. n이 크고 질의들의 복잡도가 높을 경우 전체 연산 수는 매우 높게 된다.

〈표 1〉 기호 및 용어

W	작업부하 (workload)
I_i	애트리뷰트 A_i에 대한 색인
Q_i	작업부하 W의 검색질의 (Select)
U_i	작업부하 W의 변경 연산 (Insert, Update, Delete)
Y	색인 구성 (a set of indexes)
$Size\ (I_i)$	색인 I_i의 크기
$QueryCost(\Psi, Q)$	색인 집합 Ψ를 이용하여 질의 Q를 처리하는 비용
$IndexUpdateCost(I)$	색인 I의 변경 비용
M	저장공간 한도

작업부하 W의 질의 및 변경연산에 포함되는 애트리뷰트들을 모두 A_1, A_2,…, A_n이라 할 때, 변수 x_i, $1 \le i \le n$, 는 애트리뷰트 A_i에 대한 색인의 생성 여부를 결정 짓는 이진 결정 변수(binary decision variable) 라고 하자. 또, I_i는 애트리뷰트 A_i에 대한 색인이라고 하자. 질의 Q_i를 처리하는 비용은 질의 Q_i에 속한 애트리뷰트들에 대한 색인의 존재 여부에 따라 달라지게 된다. 질의 Q_i가 WHERE 절에 두 개의 애트리뷰트 A_1 과 A_2 를 갖는다고 하자. 이 경우 처리 방법은 두 애트리뷰트에 대한 색인을 하나도 이용하지 않거나, 둘 중 하나만 이용하거나, 또는 두 색인 모두 이용하는 처리 유형이 있을 수 있다. 각 경우에 따라 질의 처리기는 상이한 처리 비용값을 계산하게 된다. 네가지 조합을 결정변수로 함께 표현하면 다음과 같다.

$$\begin{aligned} & x_1 * x_2 * QueryCost(\{I_1, I_2\}, Q) \\ & + x_1 * (1-x_2) * QueryCost(\{I_1\}, Q) \\ & + (1-x_1) * x2 * QueryCost(\{I_2\}, Q) \\ & + (1-x_1) * (1-x_2) * QueryCost(\{\ \}, Q) \end{aligned} \quad (3)$$

$QueryCost(\{\ \}, Q)$와 같이 색인이 없을 경우에는 질의 최적기는 테이블의 정렬 여부 등의 물리적 구성을 고려하여 순차, 이항 등의 접근 방법을 선택할 것이다. 이 때의 접근 비용은 테이블의 튜플수, 튜플 크기, 검색 유형 (일치/범위 검색), 등을 이용하여 계산될 수 있다.

편의상 작업부하 W의 검색질의 Q가 A_1, A_2, …, A_k의 k개의 애트리뷰트를 검색조건 식에서 이용한다고 하고, 변경연산 U는 애트리뷰트 A_1, A_2, …, A_k들을 변경한다고 하자. 위의 최적화 목표 함수식 (1)은 다음과 같이 일반화시킬 수 있다

$$\text{Minimize } \sum_{Q\in W} \sum_{i=0}^{2^k-1} F(i,k) \times QueryCost(S_{i,k}, Q) + \sum_{U\in W} \sum_{i=1}^{k} x_i \times IndexUpdateCost(I_i) \quad (4)$$

여기서 $F(i, k)$는 결정변수의 함수로서 다음과 같다.

$$F(i,k) = \prod_{j=1}^{k} f(i,j,k), \text{ where } f(i,j,k) = x_j$$

if j-th bit of i equals 1, otherwise $(1-x_i)$(5)

$S_{i,k}$는 색인집합으로써 다음과 같이 정의된다.

$S_{i,k}$ contains an index I_j if j-th bit of i equals 1, for $1 \le j \le k$ (6)

식 (4)를 간단히 설명하면 다음과 같다. 첫번째 식은 검색 질의의 처리비용 식이고 두번째 식은 변경 연산의 처리비용 식이다. 검색 질의 처리비용은 각각의 검색질의 처리비용의 합으로서, 각 검색질의 비용은 그 질의에 포함된 애트리뷰트들의 색인 존재여부에 대한 이진 조합 중 하나로 정의된다. 각 조합의 함수 $F(i,k)$에서는 애트리뷰트의 색인이 존재할 경우는 그 애트리뷰트에 해당되는 결정 변수를, 색인이 존재하지 않는 경우는 그 애트리뷰트에 해당되는 결정변수의 부정 (negation)을 서로 곱하게 된다. 변경 연산 비용은 각 변경 연산별 처리비용의 합으로서, 각 변경 연산은 관련 애트리뷰트에 색인이 있을 경우에 한하여 비용이 합산된다.

질의 Q_i의 복잡도가 m_i이고 변경연산 U_j의 복잡도가 m_j라고 하면, 목표함수는 총 $\sum_{Q_i \in W} 2^{m_i} + \sum_{U_j \in W} m_j$개의 항목을 합산하게 된다. 전체적으로는 가장 복잡도가 높은 질의에 의하여 목표함수의 합산 항목 개수가 주로 결정된다고 볼 수 있다.

3.4 고려 사항

- 부정 이진 변수(Negated Binary Variable) 모형

이진 결정변수의 부정형, 즉, $(1-x_i)$, 에 대한 부정 이항 변수를 도입함으로써 목표함수를 변형시킬 수 있다. 즉, 모든 $(1-x_i)$에 대하여 y_i로 대체하고, 제약조건으로서, $x_i + y_i = 1$을 추가하는 것이다. 목표함수의 논리적 의미는 동일하지만, 제약조건이 추가됨으로써 탐색 노드수를 감소시키는 효과가 있을 수 있다. 이는 실행시간이 원래의 목표함수 보다 단축될 수 있음을 의미한다. 단축의 정도에 대한 실험 결과를 5.4.1절에서 보여 준다. 위의 식 (5)에서 $F(i,k)$는 다음과 같이 고쳐 쓸 수 있다. 추가 제약조건은 식 (7)으로 표현된다.

$$F(i,k) = \prod_{j=1}^{k} f(i,j,k), \text{ where } f(i,j,k) = x_j$$

if j-th bit of i equals 1, otherwise y_j (5)'

$$x_i + y_i = 1, \quad 1 \le i \le n \quad (7)$$

where n is the total number of indexable attributes in the workload

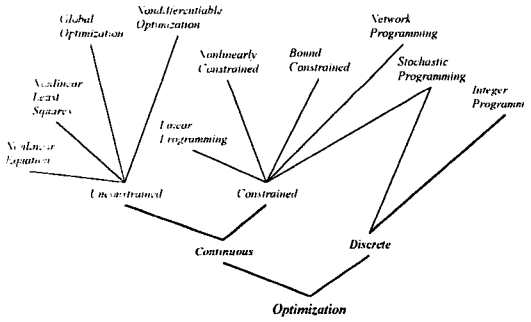
- 변경 연산에 대한 비용 계수

색인에 대한 변경 비용 계수 *IndexUpdate Cost*(I_i)는 애트리뷰트 A_i에 대한 색인이 만들어진 테이블에 대한 새로운 투플의 추가, 삭제, 색인된 애트리뷰트값에 대한 변경 연산에 대하여 발생한다. 색인의 변경 비용은 B+-tree의 높이에 해당되는 높이의 접근 수가 주요 요소이며, 트리의 높이는 색인된 애트리뷰트의 서로 다른 값의 개수, 테이블의 투플 수, 색인 키의 크기, 트리의 노드의 fill-factor, 비단말 노드의 fan-out 비율, 등에 의하여 결정된다. 질의 처리비용과 마찬가지로 비용단위는 보통 디스크 블록 접근 횟수로 표기한다. 변경 연산에 대한 데이터 파일의 변경 비용은 색인 유무에 별개로 필요한 비용이므로, 색인 선택 문제에서는 고려하지 않는다.

4. 최적화 프로그램

최적화 알고리즘은 목표 문제의 특성과 문제 해결 정도에 따라 다양하게 분류될 수 있다. 아

래 <그림 2>은 최적화 문제 해결 알고리즘의 분류도를 참고로 보여준다. 최적 색인 구성 문제는 결정변수의 도메인이 0 또는 1의 값이고 목표함수의 각 항목들은 처리 비용값으로 정수값의 계수를 갖으며, 색인 저장공간의 제한을 갖는 문제이다. 본 연구에서는 최적 색인구성 문제를 해결하는 프로그램으로서 OPBDP[Barth 1995]를 이용하였다. OPBDP는 SAT 문제(proposition al calculus satisfiability) 해결 방법으로 잘 알려진 Davis-Putnam[Davis & Putnam 1960]을 일반화한 선형/비선형 0-1 최적화 알고리즘으로서, 색인구성 최적화 문제의 모형과 부합되는 형태이다.



(그림 2) 최적화 기법 트리[OTC 2000]

OPBDP 프로그램은 몇 가지 실행 파라미터를 갖는데, 주요 내용은 다음과 같다.

- 비선형 계획법 문제는 Fortet 방식과 Barth방식을 써서 선형 문제로 변환.
- 탐색시 5가지 방식의 변수 선택 휴리스틱을 제공
  - H0 : 휴리스틱 이용 않음; 다음 자유 변수를 선택
  - H1 : two-sided Jeroslow-Wang 휴리스틱 (기본)
  - H2 : floating point precision 한도내에서 H1 적용
  - H3 : 최대 fixing을 제공하는 변수를 선택
  - H4 : 무작위 선택 (random)
- 고정 변수 검색, 계수 감소, 등의 전처리 (pre-processing) 기능

5. 실 험

다음과 같은 내용을 분석하는 목표하에 실험을 실시하였다.

- 문제 모형의 수행한계 : 변수 수, 질의 수, 색인공간 제한 등의 영향
- 부정 이진 변수를 도입한 문제 모형의 수행 개선 정도
- 변수 선택 휴리스틱의 영향
- 선형화 방법의 영향

5.1 실험환경

실험은 SunOS v5.6을 탑재한 Sun Microsystems Enterprise 450 서버(2×300MHz CPUs, 512MB Main Memory) 상에서 수행되었다. 실험에 필요한 프로그램은 GNU C/C++ Version 2.8.1로 구현되었다. 프로그램의 실행시간은 유닉스 time 명령을 이용하여 측정하였다[별첨 스크립트 참조]. 실험에 이용되는 작업부하는 TPC-H 벤치마크 데이터베이스의 스키마와 유사한 형태를 생성하였다. 테이블 크기와 그에 따른 데이터 생성 및 색인 파일 크기 등은 실제 계산을 할 수도 있지만 본 실험에서는 실험 편의성과 최적화 공식의 타당성 검증 목적에 초점을 맞추기 위하여 [Chaudhuri & Narasayya 1997]의 환경을 가정하였다. 즉, 각 질의별 관련된 색인들의 구성에 따른 예상 질의 처리 비용을 질의 처리기가 제공한 것으로 가정하였다. *Query Cost*(Ψ, Q), *IndexSize*(I), *IndexUpdateCost* (Ψ, U) 등 수식의 계수 값은 다음의 주요 실험

변수 값의 범위 내에서 임의 생성하였다.

- 질의 수 : 10~40
- 변경 연산 수 : 질의 수의 0~10%
- 색인 대상 애트리뷰트의 총 수 : 질의 수의 100~300 %
- 실행 계획의 처리 비용 : 100~500
- 색인 변경 비용 : 50~100
- 색인의 크기 : 20~40
- 색인의 크기 제한 값 : 색인 대상 전체 애트리뷰트의 30~80 % 에 대한 평균

5.2 데이터 생성

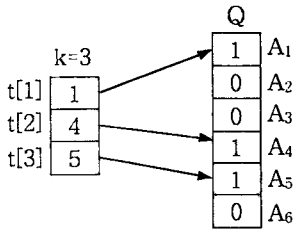
작업부하의 각 질의들의 WHERE절에 포함된 애트리뷰트의 수, 즉, 질의 복잡도는 균등하지 않다. 소수의 매우 복잡한 질의가 전체 작업부하의 대부분을 차지하는 경우가 보다 일반적이다. TPC-H의질의와 애트리뷰트 수의 사이에서도 이와 같은 분포를 확인할 수 있다 [TPC 1999]. 이와 같은 분포를 데이터 생성시에 반영하기 위하여 질의의 복잡도를 데이터 생성시의 파라미터로 하였다. 또한 작업부하의 각 애트리뷰트들도 질의들 사이에 균등 분포하지 않는 것을 알 수 있다. 즉, key, 또는 UNIQUE 애트리뷰트는 검색 절에 여타 애트리뷰트들 보다 집중적으로 많이 쓰이며, 대부분의 질의는 단순히 SELECT 절에 더 자주 이용되는 것이 일반적인 현상이다. 따라서, 애트리뷰트의 이용 빈도를 데이터 생성시 매개변수로 이용하였다. 본 실험에서는TPC-H 스키마를 참고하여 다음과 같은 기본 설정값을 이용하여 질의-애트리뷰트 작업부하를 생성하였다.

- 질의/변경연산의 복잡도 기본 설정 분포 : HOT(20%), MIDIUM(30%), LOW(50%)
- 애트리뷰트의 이용빈도 분포 : HOT(5%), MIDIUM(15%), LOW(30%), SELDOM(50%)
- 애트리뷰트 이용빈도 가감 확률 : HOT (+.25), MIDIUM(+.15), LOW(0), SELDOM (-.1)

위의 세번째 항목은 특정 애트리뷰트의 이용빈도가 결정되었을 때, 그 애트리뷰트가 각각의 질의에 이용되는 확률은 가감하기 위하여 설정하였다. 즉, 각 질의의 복잡도 확률에 애트리뷰트의 이용빈도 확률을 가감하여 다양한 불균등 분포를 생성할 수 있도록 하였다 ((그림 1)을 참고).

5.3 문제 생성

성능평가 문제 데이터의 생성은 목표함수, 제약조건 부분으로 나눌 수 있다. 목표함수는 각 질의에 대하여 포함된 애트리뷰트에 대한 결정변수의 조합 합산식을 만들고 이들을 모두 합하는 수식을 생성하는 것으로구성된다. 중심이 되는 부분은 질의 Q_i의 A_{i1}, A_{i2}, ···, A_{im}개의 애트리뷰트에 대한 2^m개의 결정변수식을 생성하는 일이다. 변경 연산에 대해서는 식 (4)에서와 같이 간단히 구할 수 있다. 아래 (그림 3)은 결정변수식 생성 방법과 예를 간략히 보여주기 위한 그림이다. (a)는 질의 Q가 색인 A_1, A_4, A_5를 WHERE절에 포함하고 있다는 것을 자료구조 형태'로 보여주고 있다. 배열 t[ ]는 그림 (b)과 같은 이진 조합을 구하기 위한 보조 자료로서 질의가 포함하는 애트리뷰트들을 연속된 배열로 표현한다. 비용식에서 C_i는 비용 계수를 뜻한다. (c)의 대략적 프로그램에서 배열 t[ ]가 이진 조합 생성에 이용되는 것을 볼 수있다. 프로그램의 설명은 생략한다.



(a) 질의와 속성 자료표현

```
(1-x1)*(1-x4)*(1-x5)*C1 +
x1*(1-x4)*(1-x5)*C2 +
(1-x1)*x4*(1-x5)*C3 +
x1*x4*(1-x5)*C4 +
(1-x1)*(1-x4)*x5*C5 +
x1*(1-x4)*x5*C6 +
(1-x1)*x4*x5*C7 +
x1*x4*x5*C8 +
```

(b) 비용식 예

```
for each query Q do
 for (i=0; i < 2^k; i++) {
   for (j=1; j<=k; j++)
     if (j-th bit of i is set)
        print "x"+t[j]+"*";
     else
        print "(1x"+t[j]+")*";
   print random( )%RANGE+"+";
 };
```

(c) C-like Pseudo Code

(그림 3) 질의처리 비용식 생성 방법과 예

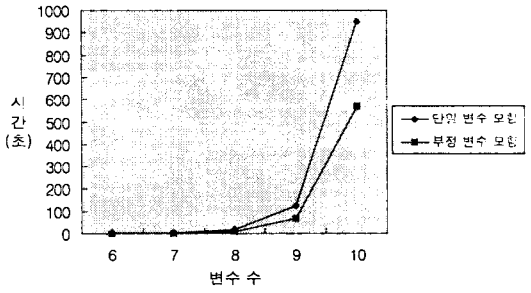
5.4 실험 결과

5.4.1 단일 변수 모형과 부정 이진 변수 모형의 분석

아래 <표 2>는 색인 대상 애트리뷰트의 개수를 변화시키면서 OPBDP 프로그램의 실행결과를 보여주고 있다. 제일 단순한 형태에서 변화를 알아보기 위하여 하나의 질의를 기준으로 하였다. 변수(즉, 애트리뷰트) 수 6~10에 대한 실행시간은 3회 반복한 평균값을 구하였으나 그 이상의 경우에는 실험 편의상 1회 실행 측정한 결과이다. (그림 4)는 이 때의 실행 시간 만을 그래프 형태로 보여준다. 부정(negation)을 (1 - x_i)로 표현하는 단일 변수모형에 비하여 부정 변수 y_i로 표현하고 $x_i + y_i = 1$ 제약조건을 추가한 부정 변수 모형의 실행 시간이 절반 정도 단축됨을 보이고 있다. 이는 제약조건의 추가가 OPBDP 프로그램의 성능에 큰 영향을 주고 있음을 보여준다. Fixed Literal의 개수는 변수 수가 7미만일 때에는 단일변수 모형과 비슷하나 그 이후에는 약 3배의 감소가 있음을 보여준다.

<표 2> 변수 모형에 대한 성능 비교

	단일 변수 모형			부정 변수 모형		
변수 수	실행 시간	Explored nodes	Fixed Literals	실행 시간	Explored nodes	Fixed Literals
6	2.7	36	351	1.2	56	396
7	3.0	46	798	1.2	84	812
8	17.7	252	4044	9.1	236	1754
9	127.5	532	12007	67.5	542	4309
10	950.0	966	27599	570.0	902	8578
11	9087.0	2698	117192	5085.0	1716	16150
12	80018.0	3754	196495	49353.0	2326	29841



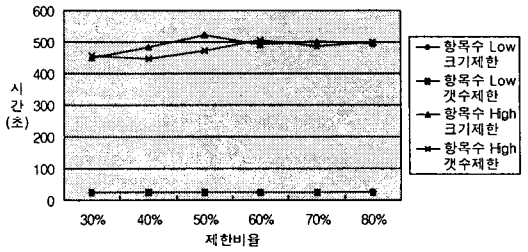
(그림 4) 변수 모형에 대한 실행시간 비교

참고로, 작업부하의 질의가 하나인 경우 최적 색인 구성을 구하는 일은 실질적으로 정렬을 이용하여 훨씬 신속하게 구할 수 있다. 즉, 애트리뷰트 수가 n개 일 때 2^n개의 선택 조합을 정렬하고 저장공간 제약조건을 만족하는 최소의 조합을 선택하면 될 것이다. 본 최적화 알고리즘은 부정 변수 모형의 문제에 대한 실행 속도가 우수한 것으로 나타났으며, 목표함수의 합산 항목 수가 2000개 이상의 문제에 대하여서는 긴

실행시간이 소요됨이 관찰되었다.

5.4.2 색인 제한값의 영향

본 실험에서는 질의수, 색인대상 애트리뷰트 수를 각각 10, 20으로 고정하고, 부정 변수모형(결정변수40개)의 문제를 생성하였다. 색인에 대한 제약조건은 전체 유효한 색인에 대한 색인 크기에 대한 제약과 색인 갯수에 대한 두가지 제약으로 실험하였다. 두 제약 모두 제한 값을 전체의 30~80%로 변화시켰다. 문제 크기의 영향을 함께 살펴보기 위하여 목표 함수의 합산 항목 개수가 436개(Low)와 956개(High)의 두 경우를 생성하였다. 아래 (그림 5)는 그 측정 결과를 보여주고 있다. 항목 수가 Low의 경우에는 실행시간이 20~30초 범위 내여서 큰 차이를 알 수 없을 정도였다. 항목 수가 High인 경우에는 실행시간이 길어지면서 실행시간의 차이도 구별 가능하였다. 일반적으로 저장공간 제약조건은 최적화 프로그램이 부적절 해를 제거하는 정도에 영향을 미친다. 즉, 허용된 저장공간의 크기가 클수록 feasible 해의 개수와 local optimum이 많이 발생할 것으로 예상되었다. 하지만 본 최적화 알고리즘은 제약조건의 값의 변화에는 큰 영향을 받지 않는 특성을 보인 것으로 관찰되었다.



(그림 5) 색인 제약 값의 영향 비교

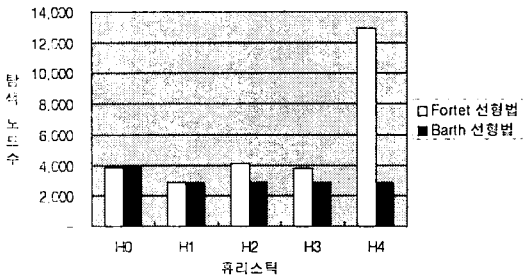
5.4.3 변수 선택 휴리스틱과 선형화 방법의 관찰

OPBDP최적화 알고리즘은 변수들에 대한 탐색 공간을 펼쳐 나갈 때 5가지의 변수 선택 휴리스틱을 구현하고 있다. 이번 실험은 질의수 10, 애트리뷰트 수 20, 부정 변수모형, 항목 수 436인 입력 문제에서 측정하였다. 또한 동일 환경에서 NLP를 LP로 변환하는 두가지 방법을 비교하였다. 측정에 결과가 아래 <표 3>에 정리되어 있다. Fortet 선형화 방법의 경우에서는 임의 선택 휴리스틱의 성능이 노드 탐색수, 탐색 깊이, 변수 Fixing수 등이 가장 큰 값을 보이고, Jeroslow-Wang 휴리스틱이 상대적으로 적은 값을 보였으나, 실행 시간 측면에서는 미세한 차이를 보였다. 문제 크기가 다소 작은 규모인 원인도 있을 것으로 보인다. Barth 선형화 방법에서는 휴리스틱을 이용하지 않는 경우(H0)를 제외하고는 나머지 휴리스틱의 성능이

〈표 3〉 변수 선택 휴리스틱과 선형화 방법의 비교

		H0	H1	H2	H3	H4
		no heuristics	two-sided Jeroslow-Wang	H1 within double range	maximal fixings	random
Fortet Linearization	Explored Nodes	3,826	2,884	4,092	3,786	17,928
	Max Depth	20	18	20	21	57
	Fixed Literals	59,975	37,138	45,672	40,108	311,563
	Run Time(sec.)	23.0	23.2	23.3	23.5	26.6
Barth Linearization	Explored Nodes	3,826	2,884	2,884	2,884	2,884
	Max Depth	20	18	18	18	18
	Fixed Literals	59,975	37,138	37,138	37,138	37,138
	Run Time(sec.)	22.9	23.0	23.0	22.9	23.6

유사한 결과를 보였다. 두 선형화 방법을 비교해보면, Barth 방법이 각 부문에서 적은 결과치를 보이는 것을 관찰할 수 있었다. (그림 6)은 두 선형화 방법을 탐색 노드 수에 대한 비교를 그래프로 보여주고 있다.



(그림 6) 선형화 방법의 노드 탐색 수 비교

6. 결 론

최적 색인 구성 문제는 데이터베이스 시스템의 성능과 자원 활용에 큰 영향을 주는 물리적 데이터베이스 설계의 중요한 요소 중 하나이다. 본 논문에서는 Operation Rerearch 분야의 최적화 기술을 데이터베이스 분야에 활용하고자 하는 실험적 연구사례로서, Davis-Putnam 알고리즘을 일반화한 OPBDP 프로그램을 이용하여 최적 색인 선택 문제를 접근하였다. 이를 위하여 비선형 계획법으로 색인 구성문제를 정형화하고, 문제 생성 알고리즘 및 다양한 불균등 작업 부하를 생성할 수 있는 알고리즘을 구현하였다. 실험 결과 최적화 프로그램은 부정 변수 모형에서 2배 이상의 실행시간 개선을 보여 주었으며 Jeroslow Wang 변수 선택 휴리스틱과 Barth 선형법에서 효율적인 성능을 나타내었다. 한편 저장공간 제약조건의 변화에 대하여는 큰 변화를 보이지 않았으며 목표 함수의 합산 항목 수가 일정 규모(e.g., 2000)이상 일 때에 긴 탐색 시간 소요되는 한계를 보이기도 하였다. 향후 보다 면밀한 분석과 성능 개선을 위한 연구 노력이 필요할 것이다.

데이터베이스 시스템에는 최적 색인 구성 뿐 아니라, 질의 최적화, 구체화 뷰의 최적화 [Labrinidis & Roussopoulos 2000, Gupta 1997], 등 다양한 최적화 문제가 존재한다. 이들 문제들에 대하여 상용 또는 실험적 최적화 패키지를 DBMS시스템과 적절히 연동하게 된다면 시스템을 효율적으로 구성하고 운영할 수 있을 것이다. 수 년 전부터 연구되고 있는 확장형 DBMS [Lakshmi *et al* 1998, Lanzelotte & Valduries 1991]의 연구에서는 사용자 데이터타입의 자유로운 확장 뿐 아니라 이와 같이 새로운 검색기술의 접목과 요소기술의 결합을 시도하고 있다는 측면에서 본 논문에서와 같은 실험적 노력은 앞으로도 많이 필요할 것으로 생각된다.

참 고 문 헌

[1] Barth, P., "A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization," MPI-I-95-2-003, MPI Informatic, Germany, 1995 (http://www.mpi-sb.mpg.de/units/ag2/software/opbdp/).

[2] Chaudhuri, S., and V. Narasayya, "An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server," Proceedings of Int'l Conference on Very Large DataBases, 1997, pp.146-155.

[3] Chaudhuri, S., and V. Narasayya, "AutoAdmin: "What-if" Index Analysis Utility," Proceedings of ACM SIGMOD Conference, 1998.

[4] Comer, D, "The Difficulty of Optimum Index Selection," ACM Transactions on Database Systems, Vol.3, No.4, December 1978, pp.440-445.

[5] Davis, M. and H. Putnam, "A Computing

Procedure for Quantification Theory," Journal of the ACM, Vol.7, 1960, pp.201-205.

[6] Finkelstein, S, M. Schkolnick, and P. Tinerio, "PhysicalDatabase Design for Relational Databses," ACM Transactions on Database Systems, Vol.13, No.1, March 1988, pp.91-128.

[7] Fourer, R., "Nonlinear Programming Frequently Asked Questions," Optimization Technology Center of Northwestern University and Argonne National Laboratory, http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html, 2000.

[8] Frank, M., E. Omiecinski, S. Navathe, "Adaptive and Automative Index Selection in RDBMS," Proceedings of EDBT, 1992.

[9] Gupta, H., "Selection of Views to Materialize in a Data Warehouse," Proceedings of Int'l Conference on Data Technology (ICDT), 1997.

[10] Horowitz, E., *Computer Algorithms*, Freeman W.H. & Company, 1996.

[11] Ip, M., L.V. Saxton, and V.V. Raghaven, "On the Selection of an Optimal Set of Indexes," IEEE Transactions on Software Engineering, Vol.9, No.2, March 1983, pp. 135-143.

[12] Labrinidis, A. and N. Roussopoulos, "Web View Materialization," Proceedings of ACM SIGMOD Conference, 2000.

[13] Lakshmi, S.M., S. Zhou, "Selectivity Estimation in Extensible Databases - A Neural Network Approach," Proceedings of the Very Large Data Bases Conference, 1998, pp.623-627.

[14] Lanzelotte, R.S.G. and P. Valduries, "Extending the Search Strategy in a Query Optimizer," Proceedings of the Very Large Data Bases Conference, 1991, pp.363-373.

[15] OTC, "NEOS Guide Optimization Tree," Optimization Technology Center, http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/index.html

[16] Seo, S.K. and Y.J. Lee, "Methodology for Index Configurations in Object-Oriented Databases," Information Sciences : An International Journal , Vol.93, No.3, 1996, pp. 187-219.

[17] Shek, E.C. and R.R. Muntz, "Exploiting Data Lineage for Parallel Optimization in Extensible DBMSs," Proceedings of Int'l Conference on Data Engineering, 1999,

[18] TPC, "TPC-H Benchmark Specification," Transaction Processing Performance Council, http://www.tpc.org/hspec.html, 1999.

[19] Whang, K.Y., G. Wiederhold, and D. Sagalowicz, "Separability An Approach to Physical Database Design," IEEE Transactions on Software Engineering, Vol.16, No.1, March 1984, pp.209-378

저자소개

서 상 구

서울대학교 컴퓨터공학과를 졸업하고, KAIST 전산학과에서 석사, 박사학위를 취득하였으며 현대전자에서 데이터웨어하우징 업무를 담당하였다. 현재 광운대학교 경영정보학과 교수로 재직하고 있으며 주요관심분야는 데이터베이스 최적화, 웹 데이터베이스, 데이터 웨어하우징, 등이다.

부 록

1. 문제 예제 및 실행 결과 예

● 작업부하

	Q_1	Q_2	Q_3	Q_4	빈도
A_1	1	0	1	1	3
A_2	1	0	1	0	2
A_3	0	0	0	1	1
A_4	1	1	0	1	3
A_5	0	0	1	0	1
A_6	1	0	1	0	2
A_7	1	0	1	0	2
A_8	1	0	0	0	1
복잡도	6	1	5	3	15

● 입력문제

```
minimize:
y8*y7*y6*y4*y2*y1*258 + x8*y7*y6*y4*y2*y1*226 +
y8*x7*y6*y4*y2*y1*303 + x8*x7*y6*y4*y2*y1*144 +
                ··· (중략) ···
x4*y3*x1*212 + y4*x3*x1*208 + x4*x3*x1*301
subject to:
x1 + y1 = 1
x2 + y2 = 1
x3 + y3 = 1
x4 + y4 = 1
x5 + y5 = 1
x6 + y6 = 1
x7 + y7 = 1
x8 + y8 = 1
x1*59 + x2*57 + x3*26 + x4*29 + x5*34 + x6*56 +
        x7*29 + x8*45 <= 150
```

● 실행 결과

y1 = y2 = y3 = y4 = x5 = x6 = x7 = y8 = 1

Global Minimum : ****** 774 ******

Explored Nodes : 66

Max. Depth : 10

Fixed Literals: 652

Time(secs) : init(0.4) prepro(0.05)

solve(0.0166667)

real 0.8

user 0.5

sys 0.0

2. 실행 스크립트 예

```
#! /bin/sh -x
if [ $# = 0 ] ; then
        echo "Usage: run [alb] file"
        exit
fi
/bin/rm -f /tmp/_sk_$2
../sim/a $1 > /tmp/_sk_$2   # for OP_A and OP_B
if [ -s /tmp/_sk_$2 ] ; then
   time /bin/sh -c "opbdp -v4 -nf < /tmp/_sk_$2 >
   res_$2" 2> time_$2
   /bin/rm -f /tmp/_sk_$2
   fold res_$2 | tail -3
   cat time_$2
fi
```