관계형 데이터 베이스 설계에서 분지한계법을 이용한 수직분할문제*

- Branch-and-bound method for solving vertical partitioning problems in the design of the relational database -

윤병익**
Byung-Ik Yoon
김재련**
Jae-Yern Kim

Abstract

In this paper, a 0-1 integer programming model for solving vertical partitioning problem minimizing the number of disk accesses is formulated and a branch-and-bound method is used to solve the binary vertical partitioning problem. In relational databases, the number of disk accesses depends on the amount of data transferred from disk to main memory for processing the transactions. Vertical partitioning of the relation can often result in a decrease in the number of disk accesses, since not all attributes in a tuple are required by each transactions. The algorithm is illustrated with numerical examples and is shown to be computationally efficient. Numerical experiments reveal that the proposed method is more effective in reducing access costs than the existing algorithms.

1. 서 론

관계형 데이타베이스에서 트랜잭션을 처리하기 위해 디스크와 주기억장치간에 이동해야 하는 자료의 양이 많으면 디스크 액세스 횟수(the number of accesses)가 증가하여 반응시간(response time)이 증가한다. 그러므로 논리적 설계(logical design) 또는 물리적 설계(physical design)를 실시할 때 반응시간을 줄이기 위하여 릴레이션(relation)을 몇 개의 단편(fragments)으로 수직분할(vertical partitioning)하여 디스크에 저장한다[6]. 이렇게 수직분할 앨고리듬에 의해 생성된 각 단편에 키 속성(primary key attribute)을 추가하여 단편의 결합이 필요한 경우 다시 만들 수 있다.

수직분할문제는 릴레이션의 속성들을 몇 개의 단편에 할당하는 조합 최적화 문제(combinatorial optimization problem)이다. 즉, m개의 속성들로 이루어진 릴레이션을 서로 다르게 수직분할하는 방법의 수는 Bell number(B(m))로 알려져 있다[12]. m이 클 경우, B(m)은 m^m과 같아져, 수직분할문제는 $O(m^m)$의 지수형 계산량을 갖는 NP-Complete 문제다.

수직분할문제에 대한 기존 연구로, Hoffer[7]는 각 단편에 주어진 용량 제약을 고려하여 저장, 검색, 갱신 비용을 최소화 하는 비선형, 0-1 정수계획모형을 개발하였다. 해법으로 분지한계법을 사용하였으며, 계산량을 줄이기 위한 clustering 기법을 적용하였다. Eisner and

* 본 논문은 1995년 한양대학교 교내 연구비에 의하여 연구 되었음.

** 한양대학교 산업공학과

Severance[5]는 사용자에 의해 자주 사용되는 속성을 주기억장치(primary memory)에 저장하고, 나머지 속성을 보조기억장치(secondary memory)에 저장하여 보조기억장치의 액세스 비용을 줄이기 위한 수리적 기법을 제시하였다. Hoffer and Severance[8] 는 속성들을 조합하여 높은 친밀도(high affinity)를 갖는 것들을 그룹핑하는 엘고리듬을 제시하였다. Navathe 등[12]은 [8]의 연구를 확장하여 2단계 이진 수직분할 엘고리듬을 제시하였다(1단계 : 직관적 설계, 2단계 : 비용을 고려한 설계). Navathe와 Ra[11]는 [12]의 수직분할방법의 친밀도 행렬에서 두 속성들 사이의 친밀도를 edge값으로 하는 친밀도 그래프 엘고리듬(affinity graphical algorithm)을 개발하였다. Cornell and Yu[4]는 속성들을 물리적인 단편에 할당하여 디스크 액세스 횟수를 최소화하는 이진분할 선형 정수계획법을 개발하였다. Chu[3]는 Cornell and Yu의 문제를 트랜잭션에 근거한 접근 방법(transaction-based approach)으로 이진 수직분할을 연구하였다. 이진 최적 수직분할은 reasonable cut 에서만 존재함을 보이고, 이진 분할에 대한 해법으로 분지한계법과 발견적 기법을 제시하였다. 엘고리듬의 계산량은 $O(2^n)$으로 트랜잭션의 수 n에 영향을 받는다. 그러나 이 연구에서 제시한 분지한계법은 가지치기(fathoming)의 효과가 발생하지 않으므로 완전 열거법(exhaustive search)과 동일하다. Cheng[2]은 속성 사용 행렬의 mutually separable cluster를 찾는 분지한계법과 발견적 기법을 제시하였다. 분지한계법에서 CI 엘고리듬[9]을 나무(tree)의 각 노드(node)마다 적용하여 inter-submatrix attributes을 찾는다.

최근의 수직분할문제에 대한 연구는 디스크 액세스 횟수를 줄이기 위한 노력에 집중되어 있다[3,4]. 그러나 이러한 연구는 트랜잭션의 수가 많을 경우 풀 수 없으며[3], 수리모형에 의한 방법도 기존의 상용 패키지에 의존할 수밖에 없다[4]. 그러므로 효율적인 수직분할 해법 개발이 필요하다.

다음절에서 수직분할문제에 대한 개념을 소개한다. 3.1에서는 본 연구에서 개발한 수직분할에 대한 수리모형을 제시하고, 3.2에서는 이진 수직분할 문제에 대한 해법으로 분지한계법을 제시한다. 4절에서 수치 예제를 제시하고, 분지한계 엘고리듬을 평가하며, 5절에서 결론을 제시한다.

2. 수직 분할 문제

트랜잭션은 대상 릴레이션의 모든 속성을 필요로 하지 않는 경우가 대부분이다. 이러한 경우에 모든 속성들을 참조 한다면 필요하지 않은 속성의 자료들이 디스크와 주기억장치 사이를 이동하여야 하므로 디스크 액세스 횟수는 증가한다. 그러므로 릴레이션을 몇 개의 단편으로 수직분할하여 불필요한 자료의 이동 양을 줄이면 디스크 액세스 횟수를 줄일 수 있다.

디스크 액세스 횟수는 트랜잭션의 액세스 패턴(access pattern), 액세스 빈도수(access frequency), 액세스 방법(access method)에 따라 달라진다[3]. 그러므로 액세스 횟수에 영향을 주는 요인 분석이 필요하다. 본 연구에서는 액세스 횟수에 영향을 주는 요인을 입력변수와 액세스 방법으로 나누고, 입력변수는 릴레이션에 관련된 변수와 트랜잭션에 관련된 변수로 나누어 분석한다.

입력변수 중 릴레이션에 관련된 변수는 릴레이션을 구성하는 속성의 수와 각 속성의 길이(byte) 및 터플의 수가 있다. 트랜잭션에 관련된 변수로는 트랜잭션 수행에 필요한 속성을 나타내는 액세스 패턴(access pattern)과 단위 시간에 발생하는 상대적 발생 횟수(relative arrival frequency), 명제의 선택 조건(selectivity of predicate) 등이 있다. 이러한 입력변수값은 시스템 분석 및 설계단계에서 얻어진다. 특히, 트랜잭션에 관련된 변수는 일반적으로 80/20 규칙을 따른다[1]. 즉, 중요한 20%의 트랜잭션을 효과적으로 처리하면, 전체 업무 중 80%를 효과적으로 처리하는 결과가 된다.

디스크 액세스 횟수는 액세스 방법에 따라 다르기 때문에 액세스 방법을 고려하여야 한다[10]. 데이타베이스 시스템에서 트랜잭션을 처리할 때, 시스템의 질의 최적기(query optimizer)가 디스크 액세스 횟수를 최소화하는 스캔방법을 선택한다. 시스템마다 스캔방법은 다를 수 있

으므로 관계형 데이타베이스에서 많이 사용하는 세그먼트 스캔방법 (segment scan method)과 인덱스 스캔방법(index scan method)을 설명한다[4].

세그먼트 스캔은 한 릴레이션의 모든 레코드를 검색한다. 스캔은 순차적이기 때문에 대상 릴레이션의 터플들을 차례로 모두 주메모리(main memory)로 가져와야 한다. 인덱스 스캔방법에서는 색인되어 있는 속성의 조건을 만족하는 터플의 수에 따라 트랜잭션에 의해 검색되는 페이지 수가 달라진다. 만약, 후보 키(candidate key)가 아닌 속성으로 색인되어 있고, 이 속성에 대하여 클러스터링(clustering)되어 있다면, 이를 clustered index scan이라 한다. 후보 키 또는 주 키(primary key)에 대하여 색인되어 있는 경우를 unclustered index scan이라 하며, 이 방법은 조건을 만족하는 레코드를 검색하기 위하여 하나의 블럭을 이동하여야 하는 것이 일반적이다[6].

이와 같은 관계형 데이타베이스에서 사용하는 스캔방법들의 액세스 횟수는 다음과 같이 추정할 수 있다[3].

· Segment scan method :

$$Nmber\ of\ accesses = \frac{(number\ of\ tuple)(length\ of\ tuple)}{(page\ size)(prefetch\ blocking\ factor)} \quad \cdots\cdots (2.1)$$

· Clustered index scan method :

$$Nmber\ of\ accesses = \frac{(number\ of\ tuple)(selectivity)(length\ of\ tuple)}{(page\ size)} \quad \cdots\cdots (2.2)$$

· Unclustered index scan method :

$$Nmber\ of\ accesses = (number\ of\ tuple)(selectivity) \quad \cdots\cdots (2.3)$$

위의 식에서, 액세스 횟수는 segment scan방법과 clustered index scan 방법에서 터플의 길이에 비례하므로, 터플의 길이를 사용하여 액세스 횟수를 구한다. 다음 절에서 트랜잭션 수행에 필요한 터플의 길이를 고려하여 액세스 횟수를 최소화 하는 수리모형을 제시한다.

3. 수직분할 모형

본 연구에서 개발한 수직분할문제에 대한 수리모형, 앨고리듬 및 선행 처리를 설명한다.

3.1 사용기호 및 모형

본 연구에서 사용하는 기호는 다음과 같다.

- 사용기호
  - m : 릴레이션의 속성 개수
  - n : 릴레이션을 액세스하는 트랜잭션 수
  - p : 단편 개수
  - i : 속성 번호 (i=1,…,m)
  - j : 트랜잭션 번호 (j=1,…,n)
  - k : 단편 번호 (k=1,…,p)

- 입력변수
  - ID : 키 속성의 길이(byte)
  - a_i : 속성 i의 길이(byte)
  - F_k : 단편 k의 터플 길이
  - C_R : 릴레이션의 터플의 수

$b_{ij}=\begin{cases}1 & \text{속성 } i\text{를 트랜잭션 } j\text{가 사용할 경우}\\ 0 & \text{사용하지 않을 경우}\end{cases}$

f_j : 트랜잭션 j의 단위 시간당 상대적 발생 횟수

S_j : 트랜잭션 j의 명제만족률

A_j : 트랜잭션 j의 액세스 비용

$T_{jk}=\begin{cases}1 & \text{트랜잭션 } j\text{가 단편 } k\text{를 필요로 하는 경우}\\ 0 & \text{필요로 하지 않는 경우}\end{cases}$

- 결정변수

$X_{ik}=\begin{cases}1 & \text{속성 } i\text{가 단편 } k\text{에 할당될 경우}\\ 0 & \text{그렇지 않을 경우}\end{cases}$

수직분할문제에 대한 본 연구의 수행 척도는 트랜잭션에 필요한 디스크 액세스 비용(본 연구에서 액세스 횟수 대신 액세스 비용을 수행척도로 사용한다. 만약, 각 단편에 대해 액세스 횟수를 구체적으로 계산하려면 시스템에서 정의하는 page size와 blocking factor를 모형에 포함시켜서 본 앨고리듬을 사용할 수 있다(식 (2.1), (2.2) 참조))이다. m개의 속성으로 구성된 릴레이션을 액세스하는 트랜잭션이 n개가 있다고 하자. 이 릴레이션이 p개의 단편으로 분할되었다고 가정할 때, 트랜잭션 j가 필요한 단편들을 접근하는 데 필요한 액세스 비용은 분할된 단편에 영향을 받으므로 $A_j=\sum_{k=1}^{p} l(C_R\cdot S_j\cdot (F_k+ID)\cdot T_{jk})$ 로 나타낼 수 있다. 이 식에서 l 은 트랜잭션 j가 단편 k를 액세스하는 비용 함수이며, 이 값은 액세스 방법에 따라 달라진다. 예를 들어, segment scan방법을 사용할 경우, 트랜잭션 j에 단편 k만 필요하면, 단편을 차례로 주 메모리로 이동시켜야 하므로, 이에 필요한 액세스 비용은 $A_j=C_R\cdot (F_k+ID)$ 가 된다. 또한, 단위 시간에 발생하는 트랜잭션의 상대적인 발생 횟수가 다를 수 있으므로, 트랜잭션 j의 단위 시간에 발생하는 액세스 비용은 $f_j\cdot A_j$ 가 된다. 그러므로 단위 시간에 발생하는 모든 트랜잭션의 총 액세스 비용은 $\sum_{j=1}^{n} f_j\cdot A_j$가 된다. clustered index scan방법을 사용할 경우는 S_j 가 고려되어야 한다. 액세스 방법을 고려한 트랜잭션의 총 액세스 비용(Z)을 최소화 하는 0-1 정수계획모형은 다음과 같다.

$$Minimize\ Z=\sum_{j=1}^{n} f_j\cdot A_j \tag{3.1}$$

Subject to

$$A_j=\sum_{k=1}^{p} l(C_R\cdot S_j\cdot (F_k+ID)\cdot T_{jk}),\quad \forall j \tag{3.2}$$

$$F_k=\sum_{i=1}^{m} a_i X_{ik},\quad \forall k \tag{3.3}$$

$$\sum_{k=1}^{p} X_{ik}=1,\quad \forall i \tag{3.4}$$

$$(\sum_{j=1}^{m} b_{ij}\cdot X_{ik})/m\le T_{jk}\le \sum_{i=1}^{m} b_{ij}\cdot X_{ik},\quad \forall j,k \tag{3.5}$$

$$X_{ik}=\{0,1\},\quad \forall i,k \tag{3.6}$$

목적식 (3.1)은 모든 트랜잭션의 액세스 비용의 합으로, 각 트랜잭션이 접근해야 하는 데이타의 양이 감소하면 총 액세스 비용은 감소한다. 제약식 (3.2)는 트랜잭션 j의 액세스 비용을

나타낸다. 제약식 (3.3)은 단편 k에 할당된 속성들의 속성 길이의 합으로, 단편 k의 터플 길이를 나타낸다. 식 (3.4)는 모든 속성은 하나의 단편에만 할당되어야 하는 제약이다. 본 연구에서는 중복 할당을 고려하지 않는다. 식 (3.5)는 트랜잭션 j가 단편 k에 할당된 속성들 중에서 적어도 하나의 속성을 필요로 하면, 단편 k를 액세스 해야 하는 조건이다. 식 (3.6)은 변수가 0 또는 1의 값을 갖는 제약이다.

3.2 분지한계법

분지한계법을 이용하여 이진 수직분할을 시도한다. 분지한계법은 다음 두 가지 특성을 이용한다.

1) 액세스 비용은 터플의 길이에 대하여 비감소함수 (non-decreasing function)이다.
2) 단편에 속성을 추가하면, 추가되기 전 보다 액세스 비용은 증가한다.

이와 같은 사실을 이용하여 속성을 하나씩 단편에 추가해서 모든 속성이 할당된 하나의 가능해(feasible solution)를 구한 후 다른 조합으로 생성되는 열등해를 제외시킬 수 있다. 이렇게 할당 순서를 변경시켜 가면서 최적해를 찾는 분지한계법은 다음과 같다.

단계 0. [초기화]
- $b_{ij}, a_i, f_j, s_j, C_R, ID$
- p=2
- 모든 가능해를 표현하는 나무(tree)를 구성한다.
- node_cost=0
- $Z_u = \infty$

단계 1. [분기]
- 깊이 우선순위로 가지치기 하지 않은 새 노드를 선택한다.
- 선택된 새 노드의 node_cost를 구한다.

단계 2. [가지치기]
- 만약 node_cost가 Z_u보다 크면, 그 노드의 자식 노드를 가지친다.

단계 3. [한계]
- 만약 노드가 잎새(leaf)노드이고 node_cost가 Z_u보다 작으면 Z_u를 node_cost로 바꾼다.

단계 4. [종료 조건]
- 나무에서 가지치기 안된 남아 있는 노드가 없으면 종료하고, 현재의 Z_u를 최적해로 한다. 그렇지 않으면 단계 1로 간다.

위의 알고리듬에서 모든 가능해를 표현하는 나무는 다음과 같다(Figure 1 참조). 뿌리 노드는 단편 F_1에 속성 1을 할당 시켰을 때의 액세스 비용(node_cost)이다. 자식 노드의 개수는 단편의 수 p=2와 같고, 깊이 우선순위의 첫 번째 자식 노드는 단편 F_1에 속성 1과 2를 할당 시켰을 때의 액세스 비용을 나타낸다. Figure 1에서 Level 2의 왼쪽 첫 번째 노드는 단편 F_1에 속성 1,2 및 3을 할당 시켰을 때의 액세스 비용을 나타낸다. 이러한 순서로 할당하여 첫 번째 잎새 노드는 모든 속성을 단편 F_1에 할당 시켰을 때의 액세스 비용이며, 하나의 가능해(feasible solution)를 나타낸다. 두 번째 잎새 노드는 마지막 속성만 단편 F_2에 할당 시키고, 나머지 속성은 모두 단편 F_1에 할당 했을 경우의 액세스 비용을 나타낸다. 모든 가능해는 잎새 노드로 표현되며, 자식 노드의 값은 부모 노드의 값 보다 크거나 같다. 알고리듬은 최소화 문제에 대한 것이므로, Z_u는 액세스 비용의 상한(upper bound)을 나타낸다. 뿌리 노드에서 잎

새 노드로 탐색할 때 노드 값은 증가하므로 임의의 노드 값이 Z_u보다 크면 그 노드의 자식 노드 값도 Z_u보다 크게 되어 자식 노드를 가지칠(fathom) 수 있다. 이와 같은 방법으로 가지치기 되지 않은 나무의 모든 노드를 탐색했을 때, 현재의 Z_u가 최적해가 된다.

4. 수치예제 및 앨고리듬 평가

예제를 통하여 앨고리듬을 설명하고, 임의로 만든 문제로 기존 연구의 앨고리듬을 비교한다.

4.1 수치예제

속성과 트랜잭션이 각각 4개인 문제를 풀어 본다. 입력변수의 값이 Table 1 에 주어졌으며, ID=2(byte), S_j=1로 사용하였다. C_R 값은 모든 단편에서 동일하므로 최적해를 구하는 과정에 영향을 미치지 않으므로 C_R=1 로 사용한다.

Table 1. Input parameters

Attribute usage matrix(b_{ij})						Arrival frequency of transactions per unit time(f_j)
		Attribute				
		1	2	3	4	
Transactions	1	1	1	0	0	30
	2	0	0	1	1	5
	3	1	1	0	0	25
	4	0	0	1	0	10
Length of attributes	a_i	8	6	4	2	

Figure 1은 문제의 해법과정을 나타낸다. 속성을 단편에 할당할 때, 임의의 속성 하나를 하나의 단편에 고정하여 할당할 수 있다. 따라서 속성 1을 항상 단편 F_1에 할당 시킬 수 있다. 뿌리 노드가 이러한 경우를 나타내며, 이 때의 node_cost는 트랜잭션 1과 3이 속성 1을 사용하므로 (8 + 2) * 30 + (8 + 2) 25 = 550이 된다. Level 1의 첫 번째 노드의 node_cost는 속성 1과 2가 단편 F_1에 할당 되었을 때의 액세스 비용을 나타낸다. 최적해는 깊이 우선순위를 적용하여 잎새 노드인 가능해 중에서 선택되므로, 네번째 잎새 노드 값(Z*=node_cost=1,000)이다. 그 때의 최적 분할 단편은 F_1={1,2}, F_2={3,4} 가 된다. 이와 같이 분할할 경우 트랜잭션 1의 액세스 비용은, 단편 F_1만 액세스 하면 되므로, A_1 = (8 + 6 + 2) * 30 = 480 이 된다. 나머지 트랜잭션의 액세스 비용도 같은 방법으로 구하고, 이것들을 합하면 총 액세스 비용(1,000)이 된다. 최적해는 액세스 비용이 터플의 길이에 비례하는 특성을 이용 하였으므로, 이 방법으로 분할하면 segment scan과 clustered index scan을 사용할 때 액세스 횟수를 줄일 수 있다.

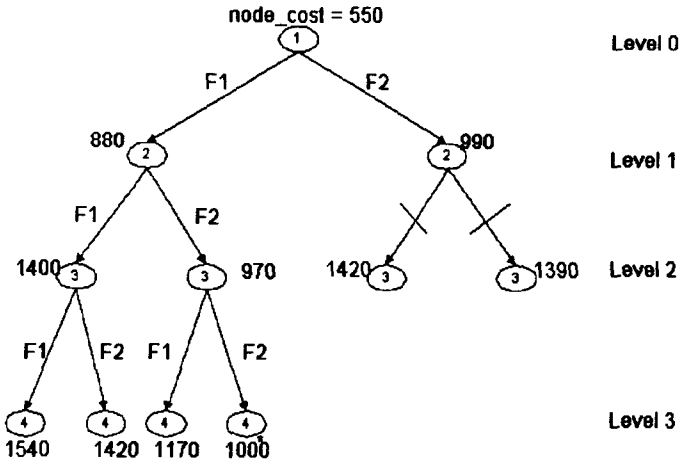

Figure 1. Search tree by branch and bound algorithm

4.2 알고리듬 수행 분석

본 연구의 이진 수직분할 알고리듬의 효율을 평가하기 위하여 기존의 트랜잭션에 기반한 Chu의 연구와 비교한다. 본 알고리듬의 계산량은 속성의 수 m에, Chu는 트랜잭션의 수 n에 각각 영향을 받는다. 현실 문제에서 릴레이션을 이용하는 트랜잭션의 수는 주어진 릴레이션의 속성 수 보다 많거나 적을 수 있다. 그러므로 일반성을 유지하기 위하여 문제의 크기는 m과 n에 대하여 같게 만들어 비교한다. 문제는 Table 2에 근거하여 각각 10 문제씩 임의로 만들었다. 이러한 문제를 IBM 호환 (펜티엄급, CPU clock 90Mhz) 컴퓨터를 사용하고, C/C++ 언어로 구현하여 푼 결과가 평균 실행시간으로 Table 3에 나타나 있다. Table 3의 30*30문제에 대한 Chu의 결과는 구할 수 없었다. 이와 같은 결과로부터 본 연구에서 제시하는 분지한계법은 Chu의 트랜잭션을 중심으로한 이진 수직분할 방법보다 효율적임을 알 수 있다. Chu는 알고리듬의 효율을 향상 시키기 위한 발견적 기법을 제시 하였으나, 이 방법을 사용하면 최적해를 ㄱ 구할 수 없으므로 액세스 비용이 증가한다. 두 방법은 액세스 비용을 최소화 하는 최적 이진 수직분할 해를 구하므로 같은 해를 얻는다. 이러한 문제에 대한 액세스 비용 감소(reduction of accesses cost) 효과를 분석하기 위해 Table 6에 나타냈다. 전체 40문제에서 분할하지 않을 때에 대한 이진 분할하였을 때의 비용 절감률은 최소 11%, 최대 44% 였다. 또한 문제 크기가 커질수록 절감률은 감소하여 이진 분할 보다 더 분할하므로써 절감효과를 얻을 수 있음을 알 수 있다. 또한 분할 단편의 개수는 속성 사용 행렬의 "1"의 밀도에 따라 민감하게 달라진다. 예를 들어 "1"의 밀도가 높으면 높을수록 분할의 절감률은 둔화되고, 높으면 많은 액세스 비용을 절감할 수 있다.

Table 2. Parameters used in generation of the relations and transactions

Parameters	values
ID	10
Length of transactions(a_i)	$1 \le a_i \le 15$
Selectivity of transactions j	$S_j = 1$
Frequency of transactions(f_j)	$5 \le f_j \le 50$
The probability of an attribute accessed by a transaction(P_r)	0.25
Cardinality	5,000

Table 3. Execution time(sec.) resulted from each problem size

Problem size(n*m)	Branch and bound	Chu's algorithm
15*15	0.28	4.83
20*20	6.05	260.00
21*21	9.42	626.00
22*22	15.28	1444.00
30*30	947.00	-

Table 4. Comparison of the reduction rate(%) of access cost for each problem sizes

Problem sizes	10×10	15×15	20×20	30×30
Average	36	28.6	19.8	13.4
Min	31	22	11	12
Max	44	36	29	16

5. 결 론

본 연구는 관계형 데이타베이스에서 디스크 액세스 비용을 최소화하는 0-1 정수 계획 모형을 개발하고, 이진 수직분할 분지한계법을 제시하였다. 관계형 데이타베이스에서 segment scan과 clustered index scan방법을 사용할 경우, 본 연구에서 제시한 이진 수직분할 최적해를 사용하면 기존의 분할 방법을 사용할 때 보다 디스크 액세스 횟수를 줄이는 데 더 효과적임을 보여 주었다. 또한 더 분할할 경우는 이진분할을 반복적으로 적용할 수 있다. 문제를 통하여 본 연구의 앨고리듬은 이진 최적해를 기존 연구보다 효율적으로 구할 수 있었다. 실험 결과에서 속성 사용 행렬의 "1"의 밀도가 낮으면 단편으로 분할하여 많은 액세스 비용을 감소 시킬 수 있음을 알 수 있었다. 또한 문제 크기가 클수록 많은 단편으로 분할하여야 액세스 비용을 많이 절감할 수 있음을 알 수 있었다.

참 고 문 헌

1. Ceri, S., S. Navathe and G. Wiederhold,"Distribution design of logical database schemas," *IEEE Trans. Software Eng.*,vol. SE-9,no.4,1983.
2. Cheng Ch,"Algorithms for vertical partitioning in database physical design," *Omega, Int. J. Mgnt. Sci.*, vol.22, no.3, pp.291-303,1994.
3. Chu W.W. and I. T. Ieong,"A transaction-based approach to vertical partitioning for relational database systems," *IEEE Trans. on Software Eng.*, vol. 19, no. 8, pp. 804-812, 1993.
4. Cornell D.W. and P.S. Yu, "An Effective approach to Vertical Partitioning for Physical Design of Relational Database," *IEEE Trans. Software Eng.*, vol. 16, no. 2, pp. 248-258, 1990.
5. Eisner M. and D.G. Severance,"Mathematical techniques for efficient record segmentation in large shared database," *J. ACM*, vol.23, no.4, pp.619-635, Oct. 1976.
6. Elmasri R. and S. Navathe, *Fundamentals of database systems*. Benjamin/Cummings, second edition,1994.

7. Hoffer J. A., "An integer programming formulation of computer database design problems," *Inform. Sci.* vol. 11, pp.29-48, 1976.
8. Hoffer J. A. and D. G. Severance,"The use of cluster analysis in physical database design," *in Proc. First VLDB*, 1975.
9. Kusiak A. and W. S. Chow,"An efficient cluster identification algorithm," *IEEE Trans. on Syst., Man and Cybern.*,vol.SMC-17, no.4,pp.696-699,1987.
10. Livadas P. E., *File Structures:Theory and Practice*. Englewood Cliffs, NJ : Prentice-Hall, 1990.
11. Navathe S. and M. Ra,"Vertical Partitioning for database design: A graphical algorithm," *in Proc. ACM SIGMOD Int. Conf. Management Data*, 1989.
12. Navathe S., S. Ceri, G. Wiederhold, and J. Dou, " Vertical partitioning algorithms for database design," *ACM Trans. Database Syst.*, vol.9, no.4, pp.680-710,1984.