트랜잭션 중심의 발견적 화일 수직분할 방법

박 기택, 김 재련

한양대학교 산업공학과

ABSTRACT

관계형 데이타베이스 환경에서 데이타 분할은 트랜잭션 혹은 질의에 요구되는 데이타량과 직접적인 관련이 있다. 본 논문에서 고려하는 데이타 분할은 중복이 없는 수직 분할로 다음 두 단계로 이루어져 있다.

첫째 단계에서는, 각 속성들간의 친밀도를 최대화시키는 0-1 정수 모형으로 속성들을 클러스터링한다. 이 단계의 결과를 초기 단편이라 한다.

두번째 단계에서는, 트랜잭션에 기반한 분할 방법을 이용하여 비용요소가 직접적으로 고려되지 않은 초기 단편을 변환시킨다. 트랜잭션에 기반한 분할 방법이란 트랜잭션 위주로 속성들을 나누는 것이다. 이 단계에서는 트랜잭션 수행에 요구되는 논리적인 액세스량을 비교 척도로 한다.

즉, 이 논문에서 제안한 수직 분할은 친밀도를 최대로 하는 최적화 모형으로 초기 분할을 한 후, 트랜잭션에 근거한 분할 방법을 이용한 발견적 기법으로 해를 개선시켜 나간다.

1. 서론

관계형 데이타베이스의 트랜잭션/질의 처리에 있어 사용자에게 관심의 대상이 되는 비교적 적은 양의 정보를 검색 또는 갱신하기 위해 많은 양의 데이타를 액세스할 필요가 종종 발생한다. 액세스의 지역성(locality of access)은 전체 릴레이션에 해당하기 보다는 그 부분집합에만 해당하기 때문에 트랜잭션에 의해 자주 요구되는 속성들은 디스크에서 메인메모리로 옮겨지는 페이지 수를 줄이기 위해 함께 그룹핑될 수 있다. 더우기 릴레이션을 단편(fragment)으로 분해시킴으로써, 각각의 단편만을 액세스하는 트랜잭션들을 동시에 수행할 수도 있다.

수직 분할은 데이타베이스 설계 시 트랜잭션 처리시간 향상을 위해 사용된다. 단편들은 비교적 적은 크기의 레코드로 구성되며, 따라서 트랜잭션 처리를 위해 디스크의 적은 페이지가 액세스되는 것이다. 자료를 메모리 계층(memory hierachy)에 할당할 때도 수직 분할은 가장 많이 액세스되는 속성들을 가장 빠른 메모리에 저장하기 위해 사용된다. 분산 데이타베이스 환경에서는 단편들이 여러 사이트에 중복이 가능하게 할당된다. 수직 단편은 궁극적으로 어떤 물리적 화일 구조를 사용해서 데이타베이스에 저장된다. 그러나 단편들에 대한 실제적인 저장구조로의 이행은 본 논문에서는 거론되지 않는다.

수직 분할은 릴레이션의 속성들을 서로 다른 단편에 할당하는 것이다. m개의 속성을 가진 릴레이션은 B(m)가지로 나뉘어 질 수 있다. 여기서 B(m)은 m번째 Bell number로 큰 수 m에 대해 B(m)은 m^m에 접근한다[1]. 비교적 적은 수 m에 대해서도 exhaustive approach로 이 문제를 푸는 것은 가능하지 않다.

Hoffer[7]는 수직 분할을 결정하기 위해 비선형 0-1 정수계획 모형을 개발하였다. 각각의 서브화일에 주어진 용량제약하에서 저장, 검색, 갱신 비용을 최소화시키는 클러스터 분석을 사용하여 해를 찾는다.

Eisner and Severance[8]는 primary memory에 저장되어 있고 모든 사용자에 의해 액세스 가능한 집합과, secondary memory에 저장되어 각 사용자에 의해 추가 비용으로 검색되는 집합들 중에서 레코드를 세그먼팅하는 수리적 기법을 제시하였다. 비용은 primary segment의 길이와 secondary segment의 사용을 요구하지 않는 트랜잭션 런의 수에 대한 선형함수로 표현된다.

Hoffer and Severance[12]는 속성들을 조합해서 높은 친밀도를 갖는 것들을 그룹핑하는 앨고리듬을 개발하였다. 속성들간의 친밀도는 트랜잭션에 의해 함께 참조되는 정도로 세가지의 친밀도를 제시하였다.

Navathe et al.[1]은 위의 작업을 확장하여 2단계 수직 분할 앨고리듬을 제안하였다. 즉 첫째 단계인 직관적인 설계 단계와, 둘째 단계인 비용에 근거한 설계 단계이다. 직관적인 설계 단계에서는 비용 요인에 대한 자세한 정보 없이 설계자가 설계에 대한 결정을 하도록 한다. 분할에 대한 입력 파라미터는 속성 사용 정보와 각 트랜잭션의 논리적 액세스 횟수(logical access frequency)이다. 먼저 속성 친밀도 행렬(attribute pairwise affinity matrix)을 만드는데 두 속성간의 친밀도는 다음과 같이 표현된다. $aff_{ij} = \sum_k acc_{kij}$. 여기서 acc_{kij}는 속성 i와 속성 j를 동시에 참조하는 트랜잭션 k의 빈도수이다. 그다음 행렬을 가능한한 block diagonal형태로 만들기 위한 행과 열을 조합하는 클러스터링 앨고리듬이 제안되었다. 행렬은 비관련 속성들의 액세스를 최소화시키기 위해 두 개의 집합으로 분리된다. 나누어진 세그먼트의 속성들은 중복될 수도 있다.

속성 친밀도 행렬로 부터 얻은 두 속성 사이의 친밀도를 edge 값으로 나타낸 친밀도 그래프 방법도 제안되었다[2]. 속성 친밀도 행렬을 linearly connected spanning tree로 만든 후 하나의 사이클을 하나의 단편으로 보는 이 앨고리듬은 한번에 모든 의미있는 단편들을 생성해 낸다.

Cornell and Yu[4]는 관계형 데이타베이스의 물리적 설계에 기반한 수직 분할법을 개발하였다. 속성들을 물리적인 세그먼트에 할당함으로써 디스크 액세스 수를 최소화시키는 최적 이진분할 앨고리듬을 선형 정수계획법을 사용하여 개발하였다.

Chu[3]는 관계형 데이타베이스에서 트랜잭션에 근거한 새로운 수직 분할법을 제안하였다. 여기서는 트랜잭션별로 속성들을 분석하여 화일을 분할한다. 분지한계법에 기초한 최적 이진분할 앨고리듬(OBP)과 트랜잭션의 수가 많을 때 적용하는 이진 분할 앨고리듬(BP_i)이 개발되었다.

이전 연구에서는 주로 속성들의 친밀도를 크게 하는데 촛점을 맞추거나, 인덱스등 액세스 방법을 고려한 이진 분할 앨고리듬을 사용하였다. 거의 모든 경우에 있어 속성 수나 트랜잭션의 수에 따라 그 계산량이 급격히 증가하는 문제점을 가지고 있다. 본 연구는 속성 친밀도를 최대화시키는 초기 분할 후, 트랜잭션에 근거하여 합리적인 방법으로 초기 분할된 단편들을 개선함으로써 속성 수, 트랜잭션 수에 큰 제약없이 적용가능 할 뿐 아니라, 여러개의 단편으로도 분할 가능하게끔 한다.

2. 수직 분할

데이타베이스는 릴레이션들로 구성되어 있고, 각 릴레이션은 길이가 알려진 속성들로 구성되어 있다. 또한 시스템의 주요 트랜잭션들도 미리 알려져 있으며, 이 트랜잭션들은 80-20 법칙을 따른다는 것을 가정한다. 80-20 법칙이란 20%의 주요 트랜잭션이 80%의 업무를 수행한다는 것으로 대부분의 시스템에 적용되고 있다. 수직 분할에 관련된 트랜잭션 상세는 아래와 같다.

① 트랜잭션의 발생 횟수(단위 시간당).
② 각 트랜잭션의 검색이나 갱신이 필요한 속성들.
③ 각 트랜잭션이 한번 수행될 때, 선택되는 릴레이션의 평균 인스턴스 수.

본 논문은 logical decision(단편의 구조와 구성에 관한)에 중점을 두고 있으며 physical decision(저장구조나 각 단편에 대한 액세스 방법)은 이후에 고려될 수 있음을 가정한다. 그러나 설계자가 물리적인 비용 요소에 대한 약간의 사전 지식을 이용하게끔 허용한다.

다음은 본 논문에서 제시한 전반적인 해법을 나타낸 것이다.

그림 1.

초기 설계에서는 트랜잭션의 논리적 액세스 횟수에 기반하여, 각 속성들을 클러스터링하는 0-1 정수 모형을 수립한다. 이 결과로 속성간 친밀도를 최대로 하는 초기 단편들이 생성된다. 속성간 친밀도가 높은 것끼리 같은 단편에 속하고, 낮은 것끼리는 서로 다른 단편에 속하게 된다. 특히 이 방법은 이전까지의 반복적인 분할 방법과는 달리 한번에 최적해에 근사하게 도달할 수 있다.

트랜잭션에 기반한 설계에서는 비용요소를 비교 척도로 하여, 초기 단편을 트랜잭션 위주로 분석하여 개선시켜 나간다.

초기 설계 단계에서 "논리적 액세스(logical access)"란 개념은 트랜잭션에 의해 검색되는 record occurrence를 의미한다. 이 개념은 일반적으로 개개의 속성을 메인 메모리로 부터 불러오는 것은 불가능하고, 따라서 하나의 레코드가 불려질 때 그 속에 포함된 모든 속성들이 액세스된다는 것을 모형화한 것이다.

트랜잭션에 기반한 설계 단계에서는 구체적인 화일의 특성이나 데이타베이스 시스템의 구조에 의존하지 않는 비용요소로 논리적인 액세스량을 고려하였다. 제시할 모델은 저장구조(storage

structure), 화일구조(file organization), 액세스 방법(access method), 트랜잭션 처리전략(transaction processing strategy) 등의 구체적인 정보를 사용하여 확장될 수 있다. 그러나, 그러한 구체적인 모델은 특별한 환경을 위해서만 사용될 수 있다. 여기서는 구체적인 물리적 환경에 대한 상세가 없는 일반적인 방법으로 분할 문제에 접근하다.

1) 초기 설계

수직 분할을 위하여 Navathe et al.[1]의 데이타를 사용하여 초기 설계 방법을 설명한다. 먼저 각 트랜잭션과 그에 필요한 속성, 그리고 그 빈도수를 나타낸 속성 사용 행렬(attribute usage matrix)로 부터 속성 친밀도 행렬(attribute affinity matrix)를 구한다. 속성 친밀도 행렬은 그 (i, j) 원소가 속성 i와 속성 j간의 친밀도를 표시하는데, 친밀도란 속성 i와 j를 동시에 참조하는 트랜잭션의 총 접근 횟수로 나타내어진다. 다음은 속성 사용 행렬과 속성 친밀도 행렬이다.

속성	1	2	3	4	5	6	7	8	9	10	액세스
길이	10	8	4	6	15	14	4	5	9	12	횟수
T1	1	0	0	0	1	0	1	0	0	0	25
T2	0	1	1	0	0	0	0	1	1	0	50
T3	0	0	0	1	0	1	0	0	0	1	25
T4	0	1	0	0	0	0	1	1	0	0	35
T5	1	1	1	0	1	0	1	1	1	0	25
T6	1	0	0	0	1	0	0	0	0	0	25
T7	0	0	1	0	0	0	0	0	1	0	25
T8	0	0	1	1	0	1	0	0	1	1	15

그림 2. 속성 사용 행렬

속성	1	2	3	4	5	6	7	8	9	10
1	75	25	25	0	75	0	50	25	25	0
2	25	110	75	0	25	0	60	110	75	0
3	25	75	115	15	25	15	25	75	115	15
4	0	0	15	40	0	40	0	0	15	40
5	75	25	25	0	75	0	50	25	25	0
6	0	0	15	40	0	40	0	0	15	40
7	50	60	25	0	50	0	85	60	25	0
8	25	110	75	0	25	0	60	110	75	0
9	25	75	115	15	25	15	25	75	115	15
10	0	0	15	40	0	40	0	0	15	40

그림 3. 속성 친밀도 행렬

[그림 3]의 속성 친밀도 행렬의 원소를 계수로 하여 아래의 수리모형을 수립한다. 이 모형으로 초기단편들이 생성된다.

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} aff_{ij} \cdot x_{ij} \qquad (1)$$

$$s.t \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } all\ i=1,2,\cdots,n \qquad (2)$$

$$\sum_{j=1}^{n} x_{jj} \le F \qquad (3)$$

$$x_{ij} \le x_{jj} \qquad \text{for } all\ i=1,2,\cdots,n,\ j=1,2,\cdots,n \qquad (4)$$

$$x_{ij} = 0 \qquad (5)$$

여기서 n은 속성의 개수, F는 예상되는 최대 단편의 수, aff_{ij}는 속성 i와 속성 j간의 친밀도로서 속성 친밀도 행렬의 i행, j열 원소이다. 단, i=j 인 경우 속성 i와 속성 j 사이의 친밀도는 아무 의미가 없으므로 aff_{ij}의 값을 0으로 둔다. 결정 변수 x_{ij}는 속성 i와 속성 j가 같은 단편에 속하면 1이고 그렇지 않으면 0이다.

목적식 (1)은 속성 i와 속성 j간의 친밀도를 최대로 한다. 제약식 (2)는 속성 i는 오직 하나의 단편에만 속한다는 제약이고, (3)은 단편의 수는 F개 이하라는 제약이고, (4)는 단편이 생성되지 않으면 속성이 단편에 속할 수 없다는 제약이며, (5)는 0,1 정수 제약이다.

2) 트랜잭션에 기반한 설계

트랜잭션에 기반한 접근법은 Chu[3]가 제안한 방법이다. 개개의 속성을 조작단위로 고려한 이전까

지의 연구와는 달리, 하나의 트랜잭션에 의해 액세스되는 속성들을 하나의 조작단위로 고려하였다.

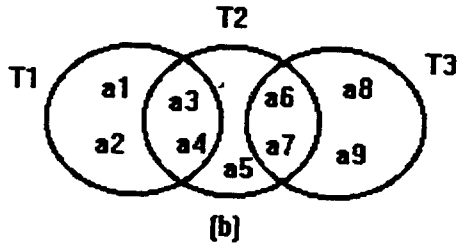
Chu는 분지한계법을 이용하여 최적 이진 분할 앨고리듬을 제안하였으나, 트랜잭션의 수가 많아지면 계산량이 급격히 증가하고 이진 분할만을 허용하므로 최적해에 도달하는 데에 한계가 있었다. 본 앨고리듬은 친밀도를 최대로 하는 초기 단편으로부터 트랜잭션에 기반한 분할 방법을 도입하여 적은 계산량으로 위의 단점을 해결하고자 한다.

2-1) Reasonable & Unreasonable Cuts

아홉개의 속성으로 이루어진 릴레이션에 T_1, T_2, T_3 세개의 트랜잭션이 있다고 가정하자. 각 트랜잭션이 액세스하는 속성을 나타내는 표가 [그림 4(a)]에 있다. [그림 4(b)]는 [그림 4(a)]의 속성들을 트랜잭션별로 나타낸 속성 교집합 그래프(attribute intersection graph)이다.

	a1	a2	a3	a4	a5	a6	a7	a8	a9
T1	1	1	1	1	0	0	0	0	0
T2	0	0	1	1	1	1	1	0	0
T3	0	0	0	0	0	1	1	1	1

(a)



(b)

그림 4. (a)액세스 패턴, (b)속성 교집합 그래프

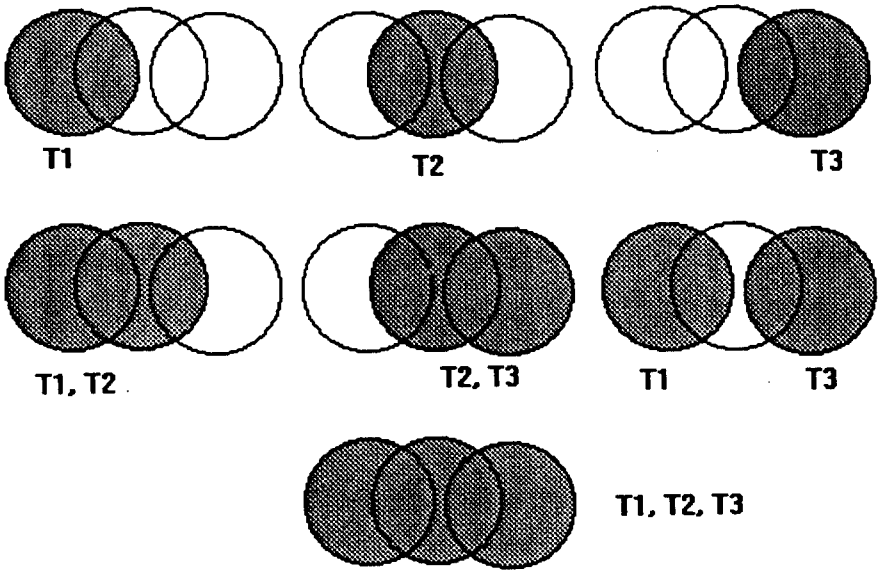


그림 5. reasonable cuts

정의) self-contained fragment, T_i, 는 트랜잭션 i가 액세스하는 속성들의 집합이다. 그러한 self-contained fragment의 합집합은 contained fragment라 한다. 속성들을 두 개의 집합으로 나누는 이진 분할을 고려할 때, 두 집합 중 적어도 한 집합이 contained fragment이면 이러한 이진 분할을 reasonable cut이라 한다.

정의) reasonable cut이 아닌 모든 이진 분할을 unreasonable cut이라 한다.

[그림 4(b)]에서 트랜잭션 1에 대한 self-contained fragment T_1은 속성 a_1, a_2, a_3, a_4로 구성되어 있다. self-contained fragment T_1과 T_2의 합집합이 contained fragment이다. 그림 5는 그림 4에 대한 이진 분할 중 reasonable cut을 보여주고 있다.

Chu[3]는 아래의 Theorem을 증명하여 reasonable cut에 의한 분할 앨고리듬을 개발하였다.

Theorem) x를 튜플 길이라 하고, $f_i(x)$를 트랜잭션 i에 대한 액세스 비용함수라 하자. 만약 모든 i에 대해서 $f_i(x)$의 이계도함수 $f_i''(x) < 0$ (concave downward) 이면, 주어진 unreasonable cut에 대해 이 unreasonable cut의 비용보다 적거나 같은 비용이 드는 reasonable cut이 적어도 하나 존재한다.

Chu[3]는 이진 분할시의 reasonable cut과 unreasonable cut을 정의하였으나 본 논문은 이것을 n 분할로 확장하여 아래의 앨고리듬을 전개한다.

2-2) 앨고리듬

같은 트랜잭션에 사용된 것들끼리는 상호 친밀도가 높고, 초기 분할은 속성간 친밀도를 최대화시킨 것이므로, 초기 분할은 트랜잭션별로 나누어 클러스터링된다. 즉 초기 분할은 reasonable cut의 형태이다. 그러나 앞에서도 언급하였듯이 이러한 초기 분할은 디스크 액세스를 직접적으로 고려하지 못하였기에 개선의 여지가 크다. 여기서는 논리적인 액세스량을 비교 척도로 하여 초기에 이루어진 단편들을 변화시킨다.

● Reasonable cut의 변화

[그림 6(a)]의 속성 교집합 그래프를 고려해보자.

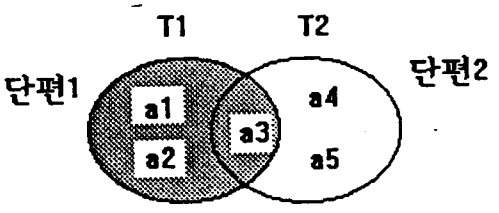


그림 6(a)

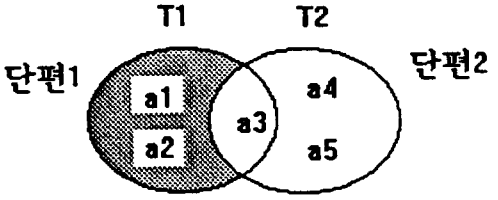


그림 6(b)

초기 분할은 단편 1={ a_1 ,a_2, a_3 } , 단편 2={ a_4, a_5 } 이다. 두 단편을 모두 액세스하는 트랜잭션은 T_1, T_2로 공통트랜잭션이라 한다. 여기서 두 트랜잭션의 공통 속성인 a_3는 속성 a_1 ,a_2와 함께 단편을 이루지만 속성 a_4 ,a_5와도 단편을 이룰 수 있다. 후자의 경우 단편은 { a_1 ,a_2 }, { a_3, a_4, a_5 }으로 구성되고 reasonable cut의 위치는 변화된다.([그림 6(b)])

정의) 두 단편 사이에 공통트랜잭션이 존재할 경우, 단편의 구성을 새롭게 함으로써 reasonable cut의 위치를 바꾸는 것을 reasonable cut의 변화라 한다.

본 연구는 두가지 경우의 reasonable cut의 변화를 고려하였다.

① 삭제: 두 단편 조합 (i, j) 사이에 공통트랜잭션이 존재할 경우, 두 단편을 통합시킴으로써 reasonable cut을 변화시킨다. [그림 7(b)]

② 이동: 단편 조합 (i, j) 사이에 공통트랜잭션이 존재할 경우, 그 트랜잭션의 공통 속성을 상대편 단편에 속하게 함으로써 reasonable cut을 변화시킨다. [그림 7(c)]

삽입과 삭제의 두 경우라도, 단편 조합 (i, j)가 서로 독립(모든 트랜잭션에 대해 공통으로 사용되는 속성이 없음)이면 reasonable cut의 변화가 고려되지 않음은 물론이다.

우리가 초기해의 변화로 reasonable cut의 이동과 삭제를 고려하는 것은 함께 사용되는 속성들끼리 묶어둔다는 화일 분할의 원리에 근거한다. 따라서, 위의 예에서 a_2를 단편 2에 속하게 하는 cut은 고려하지 않는다.

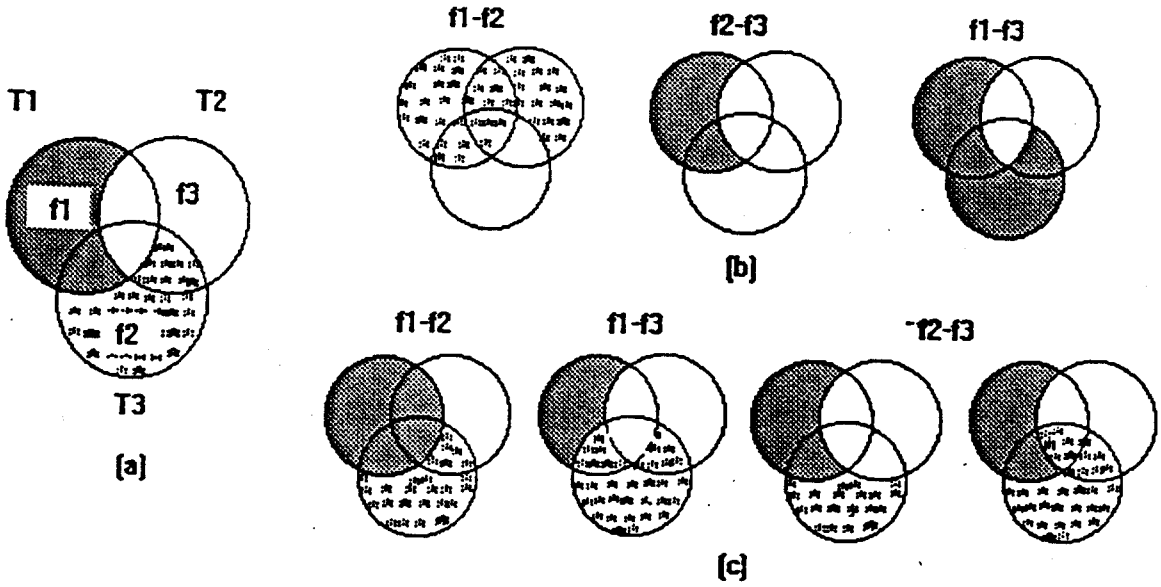


그림 7. (a)초기 단편 · (b) reasonable cut 삭제 (c) reasonable cut 이동

위에서 고려한 두 가지 reasonable cut의 변화는 적은 계산량으로 더 좋은 해를 찾기 위해 선택한 것이므로 더욱 다양한 reasonable cut의 변화(삭제 후 이동등)를 생각해 볼 수도 있다.

그러나 위의 앨고리듬은 트랜잭션이 많거나 속성 교집합 그래프가 복잡한 경우에는 많은 수의 reasonable cut의 삭제와 이동을 고려해야 하므로, 선택적으로 변환시 성능 향상에 영향을 크게 미칠 트랜잭션들만 고려할 필요가 있다. 이러한 트랜잭션을 대상트랜잭션이라 하고 아래의 T_{cand} 값이 큰 순서로 변환 대상트랜잭션을 선택한다.

$T_{cand} = freq_i \cdot length_i$

$freq_i$는 트랜잭션 i의 빈도수이고, $length_i$는 트랜잭션 i가 액세스하는 속성들의 길이의 합이다.

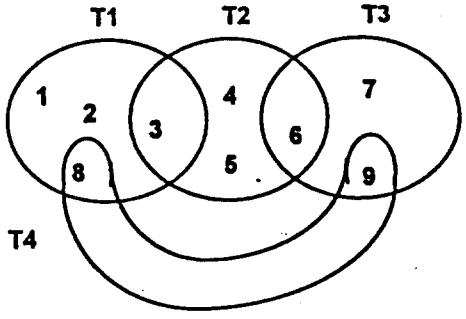


그림 8. 대상트랜잭션

[그림 8]에서 T_4의 T_{cand} 값이 T_1, T_1, T_3의 T_{cand} 값들보다 적을 경우, T_4가 단편 1,3간 공통 트랜잭션이지만 T_4에 대한 reasonable cut의 변화를 생략할 수 있다. 만약 T_4를 대상트랜잭션으로 선택한다면, 단편 (1, 3)간 reasonable cut은 {1, 2}, {7, 8, 9}로 분할하거나, {1, 2, 8, 9}, {7}로 분할하는 것으로 변화될 수 있다.

3) 비용 분석

시스템에서 디스크 I/O의 수는 트랜잭션 액세스 패턴, 액세스 빈도, 액세스 방식에 달려 있다. 모든 트랜잭션에 대한 디스크 I/O 수는 각 트랜잭션에 의한 디스크 I/O의 합이다. 수직 분할의 목적

은 릴레이션을 여러개의 단편으로 나눔으로써 총 디스크 액세스 수를 최소화시키는 것이다. 즉,

$$\min \sum_{i=1}^{n} \sum_{j=1}^{d} f_i(l(I+F_j))$$

여기서 d는 분할 후 생성된 단편의 수, I는 원래의 튜플을 확인하기 위해 모든 단편의 각 튜플에 덧붙여지는 primary key 혹은 tuple identifier이다.

$l(I+F_j)$는 단편 F_j의 속성과 I의 길이의 합이고, $f_i(l(I+F_j))$는 트랜잭션 i가 단편 F_j에 액세스하는 비용 함수이다.

본 논문에서 제시한 초기 단편과 reasonable cut 변화 후의 단편을 비교할 비용함수는 논리적인 액세스량이며 아래와 같이 표현된다.

$$\text{cost} = \sum_{i=1}^{n} \sum_{j=1}^{d} l(I+F_j) \times freq_i \times SIZE\ , \qquad A_i \cap F_j \neq \varnothing \qquad (6)$$

A_i는 트랜잭션 i에 의해 액세스되는 속성들의 집합이고, $freq_i$는 트랜잭션 i의 빈도수이며, SIZE는 릴레이션의 튜플수이다.

위의 비용함수는 논리적인 설계결정만을 가능하게 하지만, 저장구조, 화일구조, 액세스 방법이 구체적으로 명시된 시스템에서는 이들을 고려하여 물리적인 설계결정을 내릴 수 있다.

3. 수치 예제

Nava[1]에서 사용한 예제로 본 논문에서 제시한 앨고리듬을 설명한다.

3-1) 초기설계

[그림 3]의 속성 친밀도 행렬을 계수로 하여 0-1 정수모형을 수립한 후, LINDO 패키지로 초기 단편을 구한다. F=10 일때, 모형의 해는 다음과 같다.

$$X_{3,2} = X_{4,10} = X_{5,1} = X_{6,10} = X_{7,2} = X_{8,2} = X_{9,2} = X_{1,1} = X_{2,2} = X_{10,10} = 1$$

나머지 변수는 모두 0 이다.

이것은 다음의 초기단편을 만든다.

단편 1 : {1, 5}　　단편 2 : {2, 3, 7, 8, 9}　　단편 3 : {4, 6, 10}

이 결과를 [그림 2]의 속성 사용 행렬을 이용하여 속성 교집합 그래프를 그리면 [그림 9]와 같다. 그림에서 다른 트랜잭션의 부분집합인 T_3, T_6, T_7과, 합집합인 T_5는 표현하지 않았다.

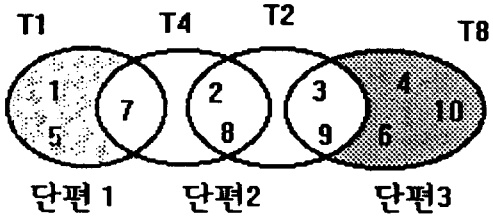


그림 9

3-2) 트랜잭션에 기반한 설계

이 단계에서는 초기 단편을 reasonable cut을 변화시킴으로써 해를 발견적으로 개선시킨다. 식 (6)이 비용으로 사용된다. 속성 1을 primary key로 SIZE는 100으로 하였다.

1. 삭제 : 단편 1,2 통합과 단편 2,3 통합 두가지이다. [그림 10]
2. 이동 : 단편 1-2간과 단편 2-3간 reasonable cut 이동 두가지이다. [그림 11]

위에서 단편 1-3간 reasonable cut의 변화는 두 단편이 독립이므로 고려하지 않았다.

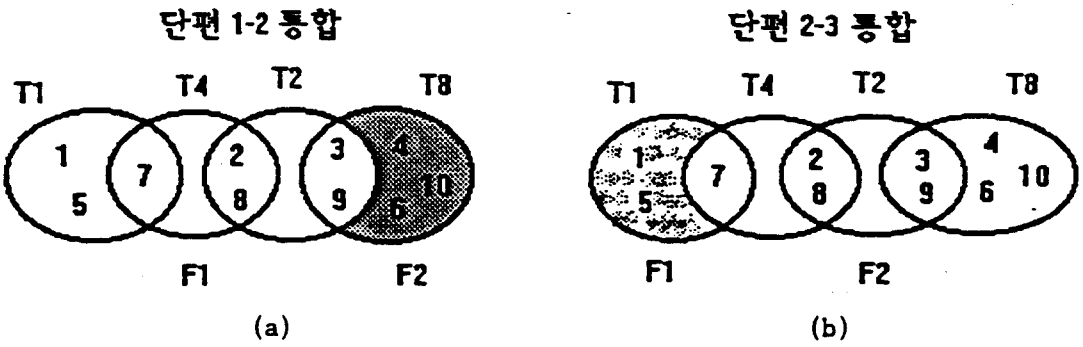


그림 10. reasonable cut 삭제

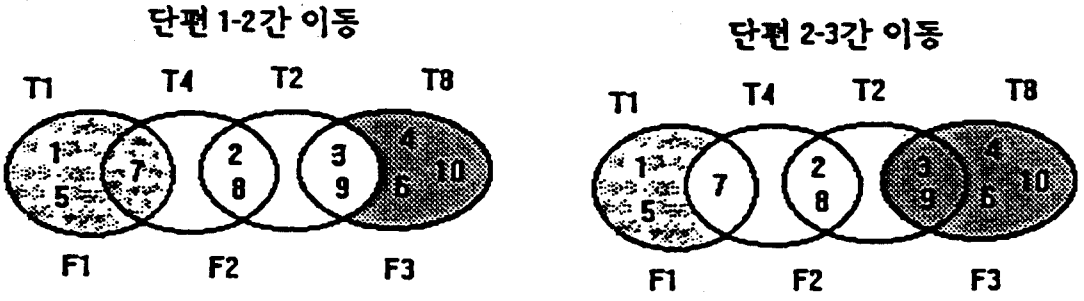


그림 11. reasonable cut 이동

초기단편	reasonable cut 변화			
	삭 제		이 동	
	단편 1-2	단편 2-3	단편 1-2	단편 2-3
1,038,000	1,248,000	1,607,500	1,016,000	1,308,500

그림 12 분할 간 비용 분석

각각의 대안에 대해 식(6)을 적용하면 [그림 12]의 결과를 얻는다. 초기 단편은 비용이 1,038,000이고 가장 적은 비용은 1,016,000 이다. 위 표는 [그림 11(a)]로 분할하는 것이 가장 효율적임을 보여준다.

4. 결론

이상에서 우리는 트랜잭션에 기반한 화일 수직분할에 대하여 논의하였다. 본 논문이 제시한 수직분할은 초기 설계와 트랜잭션에 기반한 설계로 구성되어 있다. 초기 설계에서는 속성간 친밀도를 최대로 하는 0-1 정수모형을 사용해서 반복적인 작업없이 최적해에 근사한 초기 단편을 생성해내고, 트랜잭션에 기반한 설계에서는 논리적 액세스량을 비용함수로 하여 초기 단편을 트랜잭션에 근거한 분할 방법으로 개선한다. 물론 저장구조나 액세스 방법이 명시된 시스템에서는 비용함수로 사용한 논리적 액세스량을 물리적인 측면을 고려한 함수로 대체시킬 수 있다.

제시된 알고리듬은 화일 분할 문제의 계산량에 가장 큰 영향을 미치는 속성 수나 트랜잭션 수에 매우 둔감하므로 큰 문제를 비교적 빠른 시간 내에 해결할 수 있다.

REFERENCES

[1] S.Navathe, S.Ceri, G.Wiederhold, and J.Dou, "Vertical Partitioning Algorithms for Database Design," ACM Trans. Database Systems, Vol.9, No.4, pp.690-710, Dec. 1984.

[2] S.Navathe, M.Ra, "Vertical Partitioning for Database Design: A Graphical Algorithm," Proc. ACM SIGMOD Intl. conf. Management Data, pp.440-450, May 1989.

[3] W.Chu, I.Ieong, "A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems," IEEE Trans. Software Eng., Vol.19, No.8, pp.804-812, Aug. 1993.

[4] D.W.Cornell, P.S.Yu, "An Effective Approach to Vertical Partitioning for Physical Design of Relational Databases," IEEE Trans. Software Eng., Vol.16, No.2, pp.248-258, Feb. 1990.

[5] S.T.March, "Techniques for Structuring Database Records," ACM Comput. Surveys, Vol.15, No.1, pp.45-79, Mar. 1983.

[6] A.Kusiak, "The Generalized group technology concept," Int. J. Prod. Res., Vol.25, No.4, pp.561-569, July 1986.

[7] J.Hoffer, "An integer programming formulation of computer database design problems," Inform. Sci., Vol.11, pp.29-48, 1976.

[8] M.Eisner, D.Severance, "Mathematical techniques for efficient record segmentation in large shared databases," J. ACM, Vol.23, No.4, pp.619-635, Oct. 1976.

[9] S.Ceri, S.Navathe. G.Wiederhold, "Distribution Design of Logical Database Schemas," IEEE Trans. Software Eng., Vol.SE-9, No.4, pp.487-504, July 1983.

[10] S.Navathe, M.Ra, R.Varadarajan, K.Karlapalem, K.Sreewastav, "A Mixed Partitioning Methodology for Distributed Database Design," UF-CIS TR 90-17, Univ. Florida, 1990.

[11] R.Elmasri, S.Navathe, Fundamentals of Database Systems, Benjamin/Cummings Publishing, 1989.

[12] J.Hoffer, D.Severance, "The use of cluster analysis in physical database design," Proc. First VLDB, 1975.