퍼지 그래프 기반의 수직 분할 알고리즘
(A Vertical Partitioning Algorithm based on Fuzzy Graph)

손 진 현 † 최 경 훈 † 김 명 호 ††
(Jin Hyun Son) (Kyung Hoon Choi) (Myoung Ho Kim)

요 약 사용자의 질의 요청을 보다 빨리 지원하고 시스템 전체 처리량을 증가시키기 위한 하나의 방법으로 데이타 스키마의 수직 분할 문제가 많이 연구되어 왔다. 수직 분할의 대표적인 응용 예로는 중앙 집중 시스템에서의 파일 분할, 분산 데이타베이스에서의 데이타 분산, 메모리 계층 사이의 데이타 분할 등이 있다. 일반적으로 수직 분할 알고리즘은 모든 유용한 단편들의 생성과 임의 분할 지원 등의 두가지 기능을 효율적으로 지원할 수 있어야 한다. 그러나, 기존의 제안된 방법들은 대부분 첫번째 기능에 중점을 두고 있어 임의 분할 기능을 지원하는데 많은 제한이 있다. 그리고 수직 분할 알고리즘에서 데이타 속성들이 포함될 단편을 결정할 때 기본적으로 모호성 문제를 가지고 있기 때문에 이에 대한 효과적인 처리가 필요하다. 본 논문에서는 퍼지 이론에 기반한 효율적인 수직 α-분할 알고리즘을 제안한다. 이 방법은 퍼지 그래프 이론을 바탕으로 수직 분할에서의 모호성 문제를 해결하여 복잡한 수학적 계산 없이 모든 유용한 단편들을 생성할 수 있다. 또한, 범용 임의 분할 기능도 효과적으로 지원할 수 있다.

Abstract The concept of vertical partitioning has been discussed so far in an objective of improving the performance of query execution and system throughput. It can be applied to the areas where the match between data and queries affects performance, which includes partitioning of individual files in centralized environments, data distribution in distributed databases, dividing data among different levels of memory hierarchies, and so on. In general, a vertical partitioning algorithm should support n-ary partitioning as well as a globally optimal solution for the generation of all meaningful fragments. Most previous methods, however, have some limitations to support both of them efficiently. Because the vertical partitioning problem basically includes the fuzziness property, the proper management is required for the fuzziness problem. In this paper we propose an efficient vertical α-partitioning algorithm which is based on the fuzzy theory. The method can not only generate all meaningful fragments but also support n-ary partitioning without any complex mathematical computations.

1. 서 론

일반적으로 정보 시스템에서는 데이타 속성(data attribute)들의 집합인 데이타 스키마(data schema)를 기반으로 정보를 관리하고 있으며, 다양한 사용자들의 요구사항들을 만족시키기 위한 하나의 방법으로 데이타 분할(data partitioning) 기법들이 연구되어 왔다. 데이타 분할 기법에는 수평 분할(Horizontal Partitioning), 수직 분할(Vertical Partitioning), 그리고 혼성 분할(Hybrid Partitioning) 등이 있다[1][2]. 수평 분할은 데이타 튜플(tuple)들 사이의 친밀도를 바탕으로 동일한 데이타 스키마를 가지는 여러 개의 데이타 튜플 집합들을 생성하는 것이고, 수직 분할은 데이타 속성들 사이의 친밀도를 기준으로 여러 개의 데이타 속성 집합들을 생성하는 것이다. 그리고 혼성 분할은 수평 분할과 수직 분할을 병행해서 사용하는 방법으로 혼용 분할(Mixed Partitioning) 혹은 중첩 분할(Nested Partitioning)이라고 부르기도 한다. 이러한 분할 기법들은 데이타 트랜잭션 혹은 질의 패턴에 기반을 두고 있다. 데이타 분할에 의해서 생성되는

· 본 연구는 HVcenter의 지원으로 수행되었음.
† 비 회 원 : 한국과학기술원 전산학과
jhson@dbserver.kaist.ac.kr
khchoi@dbserver.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2000년 11월 29일
심사완료 : 2001년 5월 14일

데이타 단편(data fragment)들은 최종적으로 네트워크로 연결된 여러 사이트에 효과적으로 할당되어야 한다. 이러한 단편 할당 문제는 사용자 요구 사항들을 지원하는 과정에서 발생하는 비용 함수를 최소화시키는데 그 목적이 있다. 본 논문에서는 효과적인 수직 데이타 분할 방법에 대해서 제안하며, 단편 할당 문제는 논외로 한다.

수직 분할의 개념은 데이타와 트랜잭션이 성능 향상에 상관 관계를 가지는 모든 응용 분야에 활용될 수 있다. 이러한 분야로는 중앙 집중 시스템에서의 파일 분할, 분산 데이타베이스에서의 데이타 분산, 메모리 계층 사이의 데이타 분할 등이 고려될 수 있다[3]. 수직 분할의 근본적인 목적은 사용자의 질의 요청을 보다 빨리 지원하는 것과 시스템 전체 처리량을 증가시키는 것이다. 예를 들면, 한 사이트에서 데이타베이스를 수직 분할하면 메모리와 보조 기억 장치 사이의 불필요한 블록 전송을 줄일 수 있기 때문에 데이타베이스 시스템의 성능을 향상시킬 수 있다[2][3][4][5][6]. 그리고, 분산 정보 시스템에서 친밀도가 높은 데이타 속성들을 그룹화하고 이를 통하여 데이타를 각 사이트에 적절히 분할 배치하면, 많은 사용자 질의들의 병렬 수행이 가능하며 결과적으로 시스템 전체의 처리량을 증가시킬 수 있다[1][7][8][9][10].

최적의 수직 분할은 응용 프로그램들의 수행 시간을 최소화시킬 수 있는 데이타 속성들의 집합들을 생성하는 것이다. 본 논문에서는 수직 분할의 결과인 데이타 속성들의 집합을 단편(fragment)이라고 부른다. 일반적으로 범용 수직 분할 문제를 지원하는 수직 분할 방법은 다음과 같은 두 가지 기능을 지원할 수 있어야 한다. 하나는 모든 유용한 단편들을 생성할 수 있어야 하고, 다른 하나는 임의 분할(n-ary Partitioning) 기능을 지원해야 한다. 그러나, 기존의 제안된 대부분의 수직 분할 방법들은 위의 두 가지 기능을 모두 효율적으로 지원하는데 다소 제한이 있다. 이에 본 논문에서는 복잡한 수학적 계산 없이 퍼지 그래프에 기반하여 모든 유용한 단편들을 동시에 생성할 뿐만 아니라 임의 분할을 지원할 수 있는 효율적인 수직 분할 방법을 제안하고자 한다.

지금까지 수직 분할 문제와 관련하여 많은 연구가 이루어져 왔다. 기본적으로 수직 분할 방법은 하나의 사용자 질의에서 사용되는 속성들을 동일한 단편에 포함시키는 것을 원칙으로 하기 때문에, [11]에서는 [12]에서 제안된 결합 에너지 알고리즘(Bond Energy Algorithm : BEA)을 이용하여 속성들 간의 친밀도를 계산하여 친밀도가 높은 속성들을 그룹화 하였다. [13]에서는 파일의 사용 패턴과 데이타 특성에 기반하여 파일의 속성을 분할하는 최적 분할 방법을 제안하였다. 이 방법에서는 두 가지 경험적(heuristic) 방법들을 제안하였는데, 속성들을 분할한 결과가 더이상 향상되지 않을 때까지 이 방법들을 번갈아 적용시킨다. 그리고, [13]에서는 기본적으로 모든 데이타 속성들이 독립적으로 존재하는 것이 최적 수직 분할에 가깝다고 생각하여, 각 데이타 속성이 개별적인 그룹을 구성하고 제안된 경험적 방법들을 반복적으로 적용함으로써 그룹들을 점차적으로 합병한다. 한편, [2]에서는 결합 에너지 알고리즘 기반의 클러스터 알고리즘과 이진 분할 알고리즘을 바탕으로 수직 분할 알고리즘을 제안하였다. 이 알고리즘에서는 [13]와는 반대로 최적의 분할은 모든 속성이 한 그룹에 포함된 쪽에 가깝다고 생각하고 있기 때문에, 하나의 큰 데이타 속성 그룹을 반복적으로 분할함으로써 최종적으로 여러 개의 의미 있는 데이타 속성 단편들을 생성한다. 그러나, [2]에서 제안된 이진 분할 방법은 모든 유용한 단편들이 결정될 때까지 복잡한 목적 함수(objective function)를 반복적으로 계산해야 하며, 클러스터링 역시 재귀적으로 반복되어야 하는 문제점들이 있다. 이를 해결하기 위해 [3]에서는 간단한 계산과 그래프를 이용하여 모든 유용한 분할들을 동시에 결정할 수 있는 새로운 수직 분할 알고리즘을 제안하였다.

데이타베이스의 분할과 네트워크에 있는 프로세서들에게 이를 할당하는 문제가 데이타베이스 설계에 있어서 매우 중요하다는 생각에서 [8]은 데이타베이스의 분할 및 할당 알고리즘을 제안하였다. 이 알고리즘은 탐욕(Greedy) 알고리즘과 우선 적합(First-Fit) 알고리즘으로 구성된다. [14]에서 Hufnagel과 Browne은 객체 지향 시스템의 대중화를 막는 요인이 과부하로 인한 비효율적인 수행 때문이라고 생각하여 객체 지향 시스템의 설계 및 구현을 위한 수직 분할 구조를 제안하였다. [5]에서는 재귀적으로 적용될 수 있는 최적의 이진 분할 알고리즘을 이용하여 질의 분석 단계와 분할 단계로 구성된 접근 방법을 제안하였다. 이 알고리즘은 질의를 처리할 때의 디스크 접근을 최소화하기 위한 정수 계획법(integer programming)에 기반을 두고 있다. 데이타 단편화(fragmentation) 문제는 분산 관계 데이타베이스 시스템의 성능을 저하시키는 요인으로 오랫동안 인식되어 왔다. 그러나 객체 지향 데이타베이스가 대중화 되면서 [9]에서는 복잡한 속성과 메쏘드를 가지는 분산 객체 데이타베이스 시스템에서 데이타 전송을 최소화하기 위한 클래스 분할 및 할당 기법을 제안하였다. 데이타베이스 시스템 이외에도 [10]에서는 분산 워크플로우 관리 시스템(workflow management system)에서 워크플로우의

각 요소 및 데이타의 분산이 필요함을 언급하였다.

본 논문의 구성은 다음과 같다. 2절에서는 본 연구의 동기를 설명한다. 그리고 3절에서는 수직 분할을 위한 효율적인 α-분할 알고리즘을 제안하고 4절에서는 임의 α-분할 알고리즘을 다룬다. 5절에서는 본 논문에서 제안한 알고리즘을 분석하고 실제 예제를 통하여 기존의 제안된 알고리즘과 비교 평가한다. 마지막으로 6절에서는 본 연구의 공헌과 앞으로 해야할 일에 대해 언급하면서 결론을 맺는다.

2. 연구 동기

앞에서 언급했듯이 일반적으로 수직 분할 알고리즘은 모든 유용한 최적의 단편들의 생성과 범용의 수직 분할 문제를 다룰 수 있는 임의 분할 등의 두 가지 기능을 효율적으로 지원할 수 있어야 한다. 이와 관련하여 아래에서 언급하는 두 가지 이유에서 본 연구의 동기를 찾을 수 있다. 첫째, 기존의 제안된 방법들은 수직 분할 알고리즘의 기본적인 두 가지 기능을 모두 효율적으로 지원하지 못한다. 지금까지 제안된 대부분의 방법들은 모든 유용한 단편들을 생성하는 기능에 초점을 두어 상대적으로 임의 분할 기능에 많은 제한을 가지고 있다. 한편, [2]에서는 이진 분할 알고리즘을 반복적으로 수행하여 임의 분할 기능을 지원하고자 하였다. 그러나, 이진 분할 알고리즘의 특성상 목적 함수의 값이 음수일 경우에는 더 이상의 분할이 불가능하고, 각 단계에서 이진 분할 알고리즘을 적용할 때 세부적으로 분할할 단편들을 선택하는 기준이 명확하지 않다. 따라서 이진 분할 알고리즘의 반복적인 수행은 임의 분할 기능을 완전히 지원하지는 못한다고 할 수 있다. 둘째, 수직 분할의 문제는 본질적으로 모호성(fuzziness)을 가지고 있다. 수직 분할은 서로 친밀도가 높은 속성들로 구성된 단편을 생성하는 과정이므로 임의의 두 속성 사이의 친밀도가 단편들을 결정하는데 중요한 요소가 된다. 그림 1과 같이 3개의 단편이 있고 att_7은 att_2, att_4, att_5와 각각 0.5, 0.4, 0.5의 친밀도를 가진다고 가정하면, att_7이 포함될 단편을 결정할 때 모호성에 직면하게 된다. 따라서 수직 분할 알고리즘은 이러한 본질적인 모호성을 효율적으로 반영하고 이용할 수 있어야 한다.

본 논문에서는 퍼지 그래프에 기반한 효율적인 수직 분할 알고리즘인 α-분할 알고리즘을 제안하고자 한다. 이 알고리즘은 단편 내에서의 불확실성과 단편들 사이의 불확실성을 최소화하면서 모든 유용한 단편들을 생성할 뿐만 아니라 복잡한 수학적 계산 없이 효율적으로 임의 분할 기능을 지원한다. 이때, 제안된 수직 분할 방법에 의해서 최종적으로 생성되는 단편들은 서로 동일한 데이타 속성을 공유하지 않는 비겹침 단편(non-overlapping fragment)들이다.

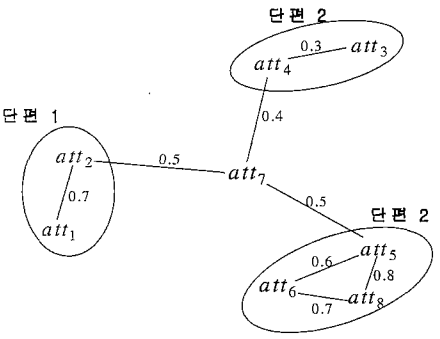


그림 1 수직 분할에서의 모호성

3. α - 분할 알고리즘

3.1 기반 정보

데이타 스키마의 수직 분할은 데이타에 대한 사용자 질의 패턴과 밀접한 관련이 있다. 다시 말해서, 동일한 사용자 질의에서 접근하는 데이타 속성들은 서로간의 친밀도가 높으므로 가능한 한 같은 단편에 포함되도록 해야 한다. 수직 분할에서 가장 기본이 되는 요소인 두 속성 사이의 친밀도는 두 속성이 질의에서 같이 사용되는 빈도로 나타낼 수 있는데, 이는 속성 친밀도 행렬(AAM : Attribute Affinity Matrix)로 표현된다.

$R=\{a_1, a_2, ..., a_n\}$은 n개의 속성들로 구성된 데이타 스키마이고, $Q=\{q_1, q_2, ..., q_l\}$는 R에 대한 사용자의 데이타 질의 집합이며, $S=\{s_1, s_2, ..., s_m\}$는 시스템을 구성하는 사이트 집합이라고 하자. 결국, 사이트 s_l에서 수행된 질의 q_i는 R의 임의의 부분집합에 속하는 속성들을 사용한다. 그러므로, 데이타 스키마 R에 대한 수직 분할의 결과로 여러 개의 단편들이 생성되며, 이때 사용하는 속성들 사이의 친밀도 계산은 아래의 식을 이용한다.

$$use(q_i, a_j)=\begin{cases} 1 & \text{질의 } q_i\text{가 속성 } a_j\text{를 사용할 경우,} \\ 0 & o.w \end{cases}$$

$freq_1(q_k)$ = 사이트 s_l에서 수행되는 질의 q_k의 빈도

$$acc_l(q_k)=\begin{cases} 1 & \text{사이트 } s_l\text{에서 질의 } q_k\text{가 수행되는 경우,} \\ 0 & o.w \end{cases}$$

라고 정의할 때, 속성 a_i와 a_j사이의 친밀도 $aff(a_i, a_j)$는 다음과 같이 정의된다.

$$= \begin{cases} \dfrac{\sum_{k\ for\ use(q_k,a_i)=1 \wedge use(q_k,a_j)=1} \sum_{\forall s_l} freq_l(q_k) * acc_l(q_k)}{\sum_{k\ for\ use(q_k,a_i)=1 \vee use(q_k,a_j)=1} \sum_{\forall s_l} freq_l(q_k) * acc_l(q_k)} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

AAM =

	a_1	a_2	a_3	a_4	a_5
a_1	1	0.7	0	0	0.6
a_2	0.7	1	0.2	0	0.8
a_3	0	0.2	1	0.8	0
a_4	0	0	0.8	1	0
a_5	0.6	0.8	0	0	1

그림 2 속성 친밀도 행렬의 예

이 식의 계산 결과인 $n \times n$ 행렬이 속성 친밀도 행렬(AAM)이고, 이 행렬의 각 원소 AAM[i, j]는 $aff(a_i, a_j)$의 값을 가진다. 그림 2는 5개의 데이타 속성들로 구성된 AAM의 한 예이다. 그림 2에서 보듯이 $aff(a_i, a_j)$는 $aff(a_j, a_i)$와 동일하기 때문에 행렬 AAM은 대칭적(symmetric) 특징을 가진다. 여기서, 아래와 같이 정의되는 퍼지 함수 X를 고려하자.

$aff(a_i, a_j)$

$$X : R \rightarrow R,\ R = \{a_1, a_2, \ldots, a_n\}$$
$$\mu_X : R \times R \rightarrow [0, 1]$$
$$\mu_X(a_i, a_j) = aff(a_i, a_j)$$

퍼지 관계 함수인 $\mu_X(a_i, a_j)$를 AAM[i, j]의 친밀도 값 $aff(a_i, a_j)$로 정의할 때, 속성 친밀도 행렬 AAM은 함수 μ_X를 가지는 퍼지 속성 친밀도 행렬로 간주될 수 있다.

3.2 퍼지 α - 분할

3.1장에서 정의된 퍼지 행렬 AAM은 그림 3과 같은 퍼지 그래프로 표현될 수 있다. 행렬의 각 데이타 속성는 퍼지 그래프에서 노드가 되고, 행렬의 각 항목값 $aff(a_i, a_j)$는 노드 a_i와 a_j를 연결하는 에지의 결합력이 된다.

퍼지 그래프에서 두 노드 a_i와 a_j가 같은 단편에 포함되는지를 결정하는데 다음의 정의들을 활용한다.

정의 3.1 *퍼지 그래프에서 노드 a_i와 a_j를 연결하는 에지의 결합력, 즉 $aff(a_i, a_j)$를 a_i와 a_j사이의* ***결합 확실성 (cohesion certainty)****, 간단히 $Cohesion(a_i, a_j)$라고 한다.*

정의 3.2 *퍼지 그래프에서 노드 a_i와 a_j 사이의 $1 - aff(a_i, a_j)$를 a_i와 a_j 사이의* ***분리 확실성(discohesion certainty)****, 간단히 $Discohesion(a_i, a_j)$라고 한다.*

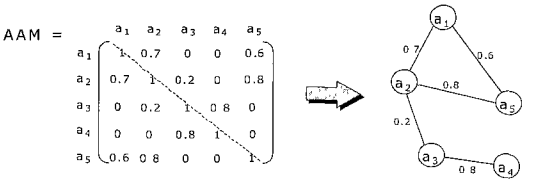


그림 3 퍼지 그래프

임의의 두 노드 사이의 결합 확실성과 분리 확실성의 합은 항상 1임을 알 수 있다. 따라서, $aff(a_i, a_j)$이 1에 가까워지면, $Cohesion(a_i, a_j)$은 커지고 $Discohesion(a_i, a_j)$은 작아진다. 반대로, $aff(a_i, a_j)$가 0에 가까워지면, $Cohesion(a_i, a_j)$은 작아지고 $Discohesion(a_i, a_j)$은 커진다. 정의 3.1과 3.2는 단편 내에서의 결합 확실성과 단편들 사이의 분리 확실성으로 각각 확장될 수 있다. 여기서 우리는 다음의 사실에 주목할 필요가 있다. 두 그룹 사이의 결합력 $aff(a_i, a_j)$ 값이 0 혹은 1에 가까울수록 더욱 확실한 정보를 준다는 것이다. 다시 말해서, $aff(a_i, a_j)$ 값이 0에 가까우면 두 노드 a_i와 a_j는 서로 다른 단편에 속할 가능성이 높다는 것이고, $aff(a_i, a_j)$ 값이 1에 가까우면 두 노드 a_i와 a_j는 동일한 단편에 속할 가능성이 높다는 것이다. 반면에 $aff(a_i, a_j)$ 값이 0.5에 가까우면 두 노드 a_i와 a_j에 대한 판단에 상당한 모호성이 있다. 이러한 사실에 기반하여 두 노드 a_i와 a_j 사이의 확실성을 의미하는 $Certainty(a_i, a_j)$를 아래와 같이 정의할 수 있다.

정의 3.3 $Certainty(a_i, a_j) = Max\{Cohesion(a_i, a_j), Discohesion(a_i, a_j)\}$

$Certainty(a_i, a_j)$는 항상 0.5보다 크거나 같고, 1보다 작은 값을 가진다. $aff(a_i, a_j)$ 값이 1 또는 0에 가까워지면 $Certainty(a_i, a_j)$는 1에 가까워지며, 이는 노드 a_i와 a_j가 같은 단편에 포함될지를 결정할 때 더 확실한 정보를 제공한다. 반대로 $aff(a_i, a_j)$ 값이 0.5에 가까워지면 $Certainty(a_i, a_j)$도 역시 0.5에 가까워지기 때문에 위의 사항을 결정하기가 어려워진다. 본 논문에서 제안하는 α-분할 알고리즘은 이러한 확실성의 개념을 바탕으로 퍼지 그래프를 여러 개의 하위 퍼지 그래프들로 분할하며 이들은 최종적으로 수직 분할의 결과물인 단편

들에 대응된다.

퍼지 그래프를 분할할 때 α-cut라는 개념을 사용한다. 퍼지 그래프에서 α-cut을 수행하면 α 값보다 결합력이 작거나 같은 모든 에지가 제거되어 여러 개의 하위 퍼지 그래프들이 생성된다. 예를 들어 그림 4에서 α-cut(α=0.6)를 수행하면, 두개의 에지가 제거되어 두개의 하위 퍼지 그래프들이 생성되며 이들은 각각 단편 $f_1=\{a_1, a_2, a_5\}$와 $f_2=\{a_3, a_4\}$에 대응된다. 따라서 α-분할 알고리즘에서 가장 중요한 기능은 전체적으로 최적인 α 값을 선택하는 것이다.

α 값을 선택하는 기준으로 α-cut에 의해서 생성되는 하위 퍼지 그래프(즉, 단편) 내부의 불확실성과 하위 퍼지 그래프들(즉, 단편들) 사이의 불확실성을 나타내는 목적 함수를 정의하고 이 목적 함수 값을 최소로 하는 α 값을 최종적으로 선택하여 수직 분할의 단편들을 생성한다. 퍼지 그래프에 α-cut을 수행하면 여러 개의 하위 퍼지 그래프들이 생성되는데 하위 퍼지 그래프를 구성하는 에지들의 결합력 값의 집합을 g라고 할 때, 각 하위 그래프에 대응되는 집합 g의 모임을 G_α라 하자. 본 논문에서는 G_α를 α-cut에 의한 결합도 멱집합(power set)이라고 부른다. G_α에 대해 불확실성을 나타내는 목적 함수 D_α는 다음과 같이 정의된다.

$$D_\alpha = \sum_{g \in G_\alpha} \sum_{a,b \in g} |a-b|$$

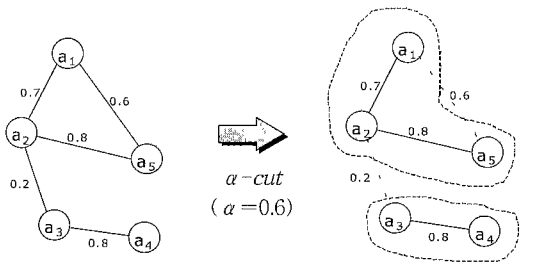


그림 4 퍼지 그래프에서의 α-cut

그림 4의 퍼지 그래프에서 α-cut(α=0.6)에 의해 생성되는 두개의 하위 퍼지 그래프를 구성하는 결합력 값들의 집합은 $g_1=\{0.78, 0.8\}$와 $g_2=\{0.8\}$이다. 그리고 추가적으로 g'라는 집합을 생성하는데 이는 α에 의해 제거된 에지들의 분리 확실성, 즉 $1-aff(a_i, a_j)$ 값들로 구성된다. 그림 4에서 $g'=\{0.4, 0.8\}$이고 $G_{0.6}=\{g_1, g_2, g'\}$이다. 불확실성 목적 함수 D_α에서 식 $\sum_{a,b \in g} |a-b|$ 는 퍼지 집합의 모호성을 나타내는 해밍 거리(Hamming distance)

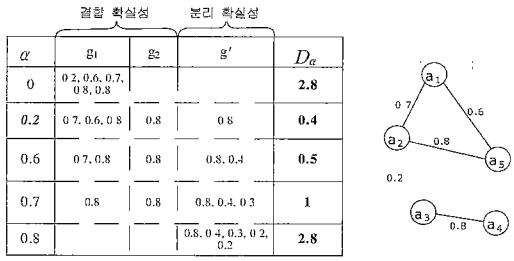


그림 5 α-분할 알고리즘의 예

이다. $G_{0.6}=\{g_1, g_2, g'\}$에서 g_1과 g_2의 해밍 거리는 하위 퍼지 그래프(단편) 내부의 결합 확실성을 나타내고, g'의 해밍 거리는 하위 그래프들(단편들) 사이의 분리 확실성을 나타낸다. 결국, 본 논문의 α-분할 알고리즘은 불확실성 목적 함수 D_α를 최소화시킬 수 있는 α 값을 결정함으로써 단편 내부의 결합 확실성과 단편들 사이의 분리 확실성을 최대로 하는 수직 분할 단편들을 생성한다.

그림 5에서 α-분할의 전체적인 예를 보여준다. 각 에지의 결합력 α 값에 대해 $G_\alpha=\{g_1, ..., g_n, g'\}$가 생성되고, 최종적으로 불확실성 목적 함수 D_α 값을 최소로 하는 α 값을 선택한다. 바꾸어 말하면, 선택된 α 값은 $g_i(i=1, ..., n)$에 의해서 표현되는 단편 내부의 결합 확실성과 g'에 의해 표현되는 단편들 사이의 분리 확실성을 최대화시킨다. 그림 5에서 $D_{0.2}$가 최소이므로 α-cut (α=0.2)를 통해 최종적으로 두개의 단편 $f_1=\{a_1, a_2, a_5\}$과 $f_2=\{a_3, a_4\}$가 생성된다.

```
For(퍼지 그래프에서 각 α에 대해) {
    -퍼지 그래프에 α-cut을 수행하여 그룹들의 집합 Gα
     를 생성한다.
    -불확실성 Dα를 계산한다.
}
-Min(Dα)인 α를 선택하여 모든 유용한 단편들을 생성한다.
```

그림 6 α-분할 알고리즘

α-분할 알고리즘은 그림 6에 설명되어 있다. 퍼지 그래프에서 서로 다른 에지의 결합력 값 α의 개수를 n이라 할 때, 이 알고리즘의 계산 복잡도는 $O(n)$이다. 본 논문의 α-분할 알고리즘은 다음과 같은 특징들을 관찰할 수 있다.

관찰 3.1 *a_i와 a_j(단, $a_i \le a_j$)로 α-분할을 수행하면, a_i-cut에 의해서 생성되는 단편의 개수는 a_j-cut에 의해서 생성되는 단편의 개수보다 작거나 같다.*

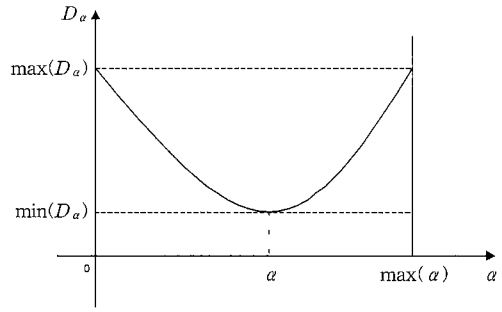


그림 7 α와 D_α의 관계

관찰 3.2 *동일한 D_α 값을 가지는 α 값은 하나 이상 존재할 수 있다.*

위의 관찰들을 통하여 α와 D_α의 계략적인 관계는 그림 7에서 보듯이 $min(D_\alpha)$ 값을 가지는 α를 기준으로 증가 함수와 감소 함수로 구성된다. α(=0)-cut은 결합 확실성을 나타내는 g_1만을 생성하고 α(= $max(\alpha)$)-cut은 분리 확실성을 나타내는 g'을 생성하므로 모두 동일한 $max(D_\alpha)$ 값을 가진다.

그리고, 관찰 3.2로부터 $min(D_\alpha)$ 값을 가지는 α 값은 여러 개가 인접하여 존재할 수 있다. 이 경우에 가장 작은 α 값을 선택하여 α-분할을 수행하는데, 가능한 한 적은 수의 단편들을 생성하기 위해서이다. 이는 모든 속성을 한 단편에 포함하는 것이 각각의 속성을 서로 다른 단편으로 만드는 것보다 최적에 가깝다는 [2]의 생각에 기반한 것이다.

4. 임의 α-분할 알고리즘

분할된 단편들 사이에 속성이 중복되지 않는 임의 α-분할 알고리즘의 계산 복잡도는 n을 데이타 속성의 개수라고 하면 $O(2^n)$임이 알려져 있다 [1][2][9]. 본 논문에서는 3장에서 다룬 α-분할 알고리즘에 기반하여 간단하고 효율적인 임의 α-분할 알고리즘을 제안한다.

임의 α-분할 알고리즘은 그림 8에서 설명되어 있다. 생성하고자 하는 단편의 개수 n이 정해지면, 먼저 n에 가장 가까운 m개의 단편을 생성하는 α를 찾는다. m개의 단편을 생성하는 α가 여러 개 존재할 수 있으므로 단계 1에서 집합 $A=\{\alpha \mid \alpha\text{-}cut$이 m개의 단편을 생성$\}$를 정의한다. 원하는 단편의 개수 n이 m과 같을 경우에는, A에서 가장 작은 α_i를 선택하고 α-cut을 수행하여 n개의 단편을 생성한다. A에서 가장 작은 α를 선택하는 이유는 3장에서 언급한 이유와 같다. n이 m보다

단계 1 : n에서 가장 가까운 m개의 단편을 생성하는 α를 찾는다.
$A=\{\alpha \mid \alpha\text{-}cut$이 m개의 단편을 생성$\}$
단계 2 : IF ($n=m$)
-A에서 가장 작은 α_i로 퍼지 그래프를 분할한다.
단계 3 : ELSE IF($n>m$)
-A에서 가장 큰 α_i로 퍼지 그래프를 분할한다.
-분해 단계
단계 4 : ELSE IF($n<m$)
-A에서 가장 작은 α_i로 퍼지 그래프를 분할한다.
-병합 단계

그림 8 임의 α-분할 알고리즘

단계 1: α_i보다 큰 α 중에 가장 작은 α'를 찾는다.
단계 2: α'를 포함하고 가장 작은 해밍 거리를 가지는 g_i를 선택하여 분할한다.
여기서, g_i는 α_i-cut으로 생성된 집합이다.
단계 3 : n개의 단편이 생성될 때까지 단계 2를 반복한다.

그림 9 분해 단계

단계 1: α_i보다 작은 α 중에 가장 큰 α'를 찾는다.
단계 2: 친밀도 $aff(g_i, g_j)$가 α'이고 병합으로 인한 해밍 거래의 증가를 최소로 하는 두 집합 g_i와 g_j를 선택하여 병합한다.
단계 3 : n개의 단편이 생성될 때까지 단계 2를 반복한다.

그림 10 병합 단계

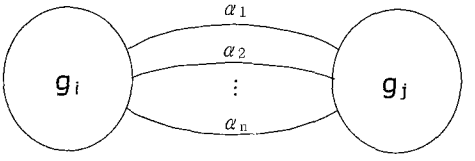
를 경우에는, 먼저 A에서 가장 큰 α_i로 퍼지 그래프를 분할한 후 분해 단계(decomposition phase)를 수행하여 m개의 단편을 n개로 분할한다. n이 m보다 작을 경우에는, A에서 가장 작은 α_i로 퍼지 그래프를 분할한 후 병합 단계(composition phase)를 수행하여 m개의 단편을 n개로 병합한다.

그림 9와 그림 10은 임의 α-분할 알고리즘에서 요구하는 분해 단계와 병합 단계를 각각 설명하고 있다. 분해 단계를 수행하는 동안 이미 생성된 m개의 단편은 n개의 단편을 생성하기 위해 더 분할된다. 우선 임의 α-분할 알고리즘에서 언급된 α_i 보다 큰 α 중에서 가장 작은 α'를 찾는데, 이는 관찰 3.1의 사실에 기반한 것이다. 그리고 α_i-cut에 의해 생성된 집합들 중에서 α'를 포함하고 가장 작은 해밍 거리를 가지는 집합을 선택하여 α'-cut을 수행하여 분할한다. 이 단계를 n개의 단편이 생성될 때까지 반복하면 된다.

병합 단계를 언급하기 전에 그림 11과 같이 퍼지 그래프에서 서로 다른 두 집합 사이의 친밀도를 아래와

같이 정의한다.

정의 4.1 *집합 g_i와 g_j가 결합력이 $\alpha_1, \alpha_2, ..., \alpha_n$인 에지들로 연결되어 있다고 하면, 두 집합 사이의 친밀도는 $off(g_i, g_j) = \max\{\alpha_1, \alpha_2, ..., \alpha_n\}$로 정의된다.*



$off(g_i, g_j) = \max\{\alpha_1, \alpha_2, ..., \alpha_n\}$

그림 11 서로 다른 두 집합 사이의 친밀도

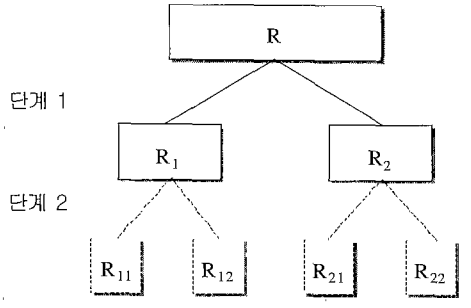


그림 12 이진 수직 분할의 반복 수행

병합 단계에서는 임의 α-분할 알고리즘에서 언급된 α_i보다 작은 α 중에서 가장 큰 α'를 찾는데, 이 역시 관찰 3.1의 사실에 기인한다. 그리고 α_i-cut에 의해 생성된 집합들 중에서 친밀도 $off(g_i, g_j)$가 α'이고 병합으로 인한 해밍 거리의 증가를 최소로 하는 두 집합을 선택하여 병합한다. 이 단계는 n개의 단편이 생성될 때까지 반복된다.

5. 분석 및 실험

지금까지 연구된 수직 분할 방법들 중에서 현재 가장 많이 참조되는 방법은 [2]에 의해서 제안된 이진 수직 분할 알고리즘(Binary Vertical Partitioning Algorithm: BVP)이다[1]. 이에 본 절에서는 본 논문의 α-분할 알고리즘과[2]에서 제안된 알고리즘을 비교 평가하고자 한다.

먼저 시간 복잡도(Time Complexity) 측면에서 두 알고리즘을 비교하자. BVP 알고리즘의 시간 복잡도는 $O(n^2)$으로 n은 분할하고자 하는 데이타 속성들의 개수이다. 한편, α-분할 알고리즘은 $O(e)$의 복잡도를 가지며 e는 그림 3과 같은 퍼지 그래프에서 에지의 개수이다. 데이타 속성들의 개수가 n이라고 할 때, 일반적으로 에지의 개수 e의 최대값은 모든 데이타 속성들 사이의 친밀도가 존재하는 $\frac{n(n-1)}{2}$이다. 그러므로 α-분할 알고리즘은 기존의 BVP 알고리즘보다 효율적임을 알 수 있다. *n-ary* 수직 분할 문제에 대해서 [2]에서는 이진 수직 분할 알고리즘인 BVP의 반복적인 적용으로 지원하고 있으며 이에 대한 시간 복잡도는 $O(2^n)$이다. 한편, 본 논문에서 제안한 *n-ary* α-분할 알고리즘은 $O(e) \approx O(n^2)$이므로 더 효율적이다. 여기서 주목할 점은 [2]에서 제안된 *n-ary* 수직 분할 알고리즘이 가지는 제한점이다. 그림 12에서 보듯이 데이타 속성들의 집합 R을 *n-ary* 수직 분할하고자 할 때 [2]에서는 BVP을 반복적으로 적용한다. 만약 *3-ary* 수직 분할 문제인 경우에 그림 12에서 R_1과 R_2 둘 중 하나만을 이진 분할해야 한다. 그러나 [2]에서 제안된 방법은 이를 효과적으로 결정할 수 없으므로 임의의 선택해야 한다는 문제가 있다. 반면에 본 논문에서 제안한 *n-ary* α-분할 알고리즘은 항상 최적의 분할을 제공한다. 이는 아래의 예에서 확인할 수 있다.

위에서 언급한 두 알고리즘을 [2]에서 사용한 예를 통해 비교해 보자. 분할하고자 하는 데이타 집합 R은 10개의 데이타 속성들을 포함한다(즉, $R=\{a_1, a_2, ...a_{10}\}$). 그리고 집합 R에 대해 8개의 질의(즉, $Q=\{q_1, q_2, ...a_8\}$)가 수행되고 이들 질의들은 한 사이트에서 발생한다고 하자. 이 때 그림 13은 각 질의가 접근하는 데이타 속성들과 발생 빈도수를 보여 준다. 예를 들면, 질의 q_1은 데이타 a_1, a_5, a_7을 접근하며 25회 발생함을 의미한다. 이 예제에 대한 본 논문의 α-분할 알고리즘과 [2]의 BVP 알고리즘의 결과는 그림 14와 같이 4개의 단편들을 생성하는 경우를 제외하고는 동일하다. [2]의 방법은 두개의 서로 다른 4개 단편들 $F_1=\{R_{111}(a_1,a_5), R_{112}(a_7), R_{12}(a_2,a_3,a_8,a_9)$,

$use(q_i, a_j)$ $freq_i(q_k)$

데이터속성 / 질의	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	접근빈도수
q_1	1	0	0	0	1	0	1	0	0	0	$freq(q_1) = 25$
q_2	0	1	1	0	0	0	0	1	1	0	$freq(q_2) = 50$
q_3	0	0	0	1	0	1	0	0	0	1	$freq(q_3) = 25$
q_4	0	1	0	0	0	0	1	1	0	0	$freq(q_4) = 35$
q_5	1	1	1	0	1	0	1	1	1	0	$freq(q_5) = 25$
q_6	1	0	0	0	1	0	0	0	0	0	$freq(q_6) = 25$
q_7	0	0	1	0	0	0	0	0	1	0	$freq(q_7) = 25$
q_8	0	0	1	1	0	1	0	0	1	1	$freq(q_8) = 15$

그림 13 수직 분할을 위한 기반 정보

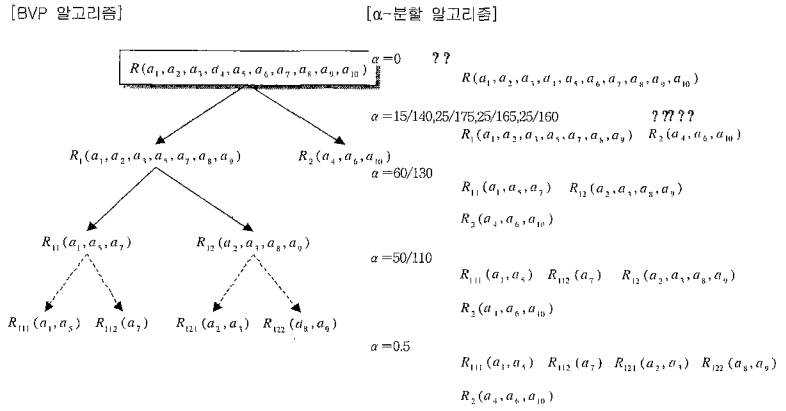


그림 14 수직 분할 결과

$R_2(a_4,a_6,a_{10})$}과 F_2={$R_{11}(a_1,a_5,a_7)$,$R_{121}(a_2,a_3)$,$R_{122}(a_8,a_9)$, $R_2(a_4,a_6,a_{10})$}을 생성하지만 본 논문의 α-분할 알고리즘은 F_1만을 생성한다. 수직 분할의 근본적인 목적은 임의의 질의가 접근하는 데이타 속성들이 가능하면 동일한 그룹(즉, 단편)에 속하도록 하는 것이므로, 두 분할 F_1과 F_2에 대해 질의들이 서로 다른 단편들을 접근하는 횟수는 그림 15와 같으므로 분할 F_1이 더 좋은 분할임을 알 수 있다. 이를 통해 분할 F_1만을 생성하는 본 논문의 α-분할 알고리즘이 기존의 방법보다 더 효율적이다.

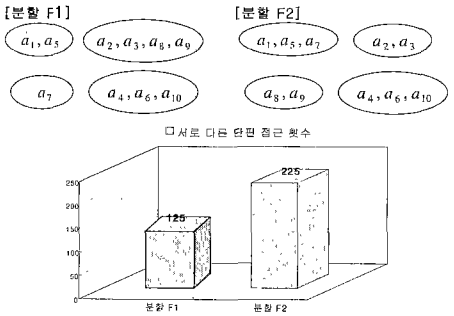


그림 15 수직 분할 결과 비교

본 논문에서 제안한 수직 분할 알고리즘은 전통적인 분산 데이타베이스 환경에서뿐만 아니라 현재 산업체 및 공공부문으로부터 많은 관심을 받고 있는 워크플로우 영역에서도 효과적으로 적용될 수 있다. 워크플로우 개념은 기업의 업무를 자동화하는 것으로 일반적으로 분산 워크플로우 시스템을 기반으로 한다. 기업의 업무는 여러 단계로 구성되어 있으며 각 단계는 분산 환경에 존재하는 다양한 자료를 활용하여 수행된다. 이때 워크플로우 업무 처리를 위해 필요한 자료를 본 논문에서 제안한 알고리즘을 활용하여 효과적으로 분할 배치한다면 워크플로우 수행 성능을 개선시킬 수 있을 것이다.

6. 결 론

수직 분할은 데이타와 질의가 시스템의 성능 향상에 상관관계를 가지는 분야에 효율적으로 활용될 수 있다. 예를 들면, 중앙 집중 시스템에서의 파일 분할, 서로 다른 메모리 계층에 저장된 데이타베이스, 분산 데이타베이스 등이 있다. 수직 분할의 기능은 사용자 질의에서 같이 사용되는 데이타 속성들을 동일 단편에 포함되게 함으로써 효율적인 사용자 질의 처리와 시스템 전체 처리량을 증가시키는 것이다. 이러한 개념은 기존의 데이타베이스 분야에 대부분 적용되어 왔지만, 트랜잭션의 병렬 처리, 워크플로우 같은 새로운 응용 분야를 지원하기 위해 확장될 수 있다.

본 논문에서는 퍼지 그래프 기반의 효율적인 수직 분할 방법인 α-분할 알고리즘을 제안하였다. 이 알고리즘은 단편 내부의 결합 확실성과 단편들 사이의 분리 확실성을 최대로 하면서 모든 유용한 비겹침 단편들을 생성할 뿐만 아니라 복잡한 수학적 계산 없이 효율적으로 범용 임의 분할 기능 역시 지원할 수 있다. 그리고, 데이타 속성 분할에서 발생하는 모호성을 효율적으로 반영하고 이용하기 위해 퍼지 개념을 적절히 활용한다.

앞으로는 본 논문에서 제안한 α-분할 알고리즘과 잘 부합할 수 있는 수평 분할 알고리즘에 대한 연구와 데이타 분할에 의해서 생성되는 단편들의 할당 문제에서 발생할 수 있는 모호성을 적절히 처리하기 위한 효율적인 퍼지 할당 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] M. Tamer Ozsu and Patrick Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1999.

[2] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou, "Vertical Partitioning Algorithms for Database Design," ACM Transactions on Database Systems, Vol. 9, No. 4, pp.680-710, December, 1984.

[3] Shamkant B. Navathe and Minyoung Ra, "Vertical Partitioning for Database Design: A Graphical Algorithm," ACM SIGMOD, pp.440-450, 1989.

[4] C. Meghini and C. Thanos, "The Complexity of Operations on a Fragmented Relation," ACM Transactions on Database Systems, Vol. 16, No. 1, pp.56-87, March, 1991.

[5] Douglas W. Cornell and Philip S. Yu, "An Effective Approach to Vertical Partitioning for Physical Design of Relational Databases," IEEE Transactions on Software Engineering, Vol. 16, No. 2, pp.248-258, February, 1990.

[6] Wesley W. Chu and Ion Tim Leong, "A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems," IEEE Transactions on Software Engineering, Vol. 19, No. 8, pp.804-812, August, 1993.

[7] Vasanth Balasundaram, Geoffrey Fox, Ken Kennedy, and Ulrich Kremer, "An Interactive Environment for Data Partitioning and Distribution," In the proceed- ings of the Fifth Distributed Memory Computing Conference, pp.1160-1170, 1990.

[8] Domenico Sacca and Gio Wiederhold, "Database Partitioning in a Cluster of Processors," ACM Transactions on Database Systems, Vol. 10, No. 1, pp.29-56, March, 1985.

[9] Ladjel Bellatreche and Ana Simonet, "Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Methods," The Seventh International Workshop on Database and Expert Systems Applications, pp.15-21, 1996.

[10] San-Yih Hwang and Chi-Ten Yang, "Component and Data Distribution in a Distributed Workflow Management System," In the proceedings of IEEE Software Engineering Conference, pp.244-251, 1998.

[11] Hoffer, J.A., and Severance, D.G., "The use of cluster analysis in physical database design," In proceedings of first international conference on Very Large Database, 1975.

[12] McCormick, W.T., Schweitzer, P.J., and White, T.W., "Problem decomposition and data reorganization by a clustering technique," Operation Research, Vol. 20, No. 5, pp.993-1009, September, 1972.

[13] Michael Hammer and Bahram Njamir, "A Heuristic Approach to Attribute Partitioning," ACM SIGMOD, pp.93-101, 1979.

[14] Stephen P. Hufnagel and James C. Browne, "Performance Properties of Vertically Partitioned Object-Oriented Systems," IEEE Transactions on Software Engineering, Vol. 15, No. 8, pp.935-946, August, 1989.

손 진 현

1996년 서강대학교 전산학과 학사. 1998년 한국과학기술원 전산학과 석사. 2001년 한국과학기술원 전산학과 박사. 2001년 ~ 현재 한국과학기술원 전산학과 연구원. 관심분야는 분산데이타베이스, 멀티데이타베이스, 미들웨어, 워크플로우, 데이타웨어하우징, 분산 스케줄링, CORBA

최 경 훈

2000년 한국과학기술원 전산학과 학사. 2000년 ~ 현재 한국과학기술원 전산학과 석사과정 재학중. 관심 분야는 분산데이타베이스, 워크플로우

김 명 호

정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조