Journal of the Society of Korea Industrial and Systems Engineering
Vol. 25, No. 1, pp.1-8, February 2002.

CP-Tree 구조를 이용한 높은 신뢰도를 갖는 연관 규칙의 효율적 탐색 방법

송한규* · 김재련**

*한양대학교 산업공학과 석사과정 · ** 한양대학교 산업공학과 교수

An Efficient Search Method for High Confidence Association Rules Using CP(Confidence Pattern)-Tree Structure

Han Gyu Song · Jae Yearn Kim

Dept. of Industrial Engineering HanYang University

The traditional approaches of association rule mining have relied on high support condition to find interesting rules. However, in some application such as analyzing the web page link and discovering some unusual combinations of some factors that have always caused some disease, we are interested in rules with high confidence that have very low support or need not have high support. In these cases, the traditional algorithms are not suitable since it relies on first satisfying high support. In this paper, we propose a new model, CP(Confidence Pattern)-Tree, to identify high confidence rule between 2-items without support constraint. In addition, we discuss confidence association rule between two more items without support constraint.

Keywords : Association rule, High Confidence Constraint, CP-Tree

1. 서론

전통적인 시장바구니분석(Market Basket Analysis)에서 연관 규칙을 찾는 대표적인 알고리듬인 Apriori 알고리듬[1]은 높은 지지도를 기반으로 하고 있다. 그러나 지지도가 최소지지도보다 낮지만 매우 유용한 정보, 즉 신뢰도가 높은 정보는 얻을 수 없게 된다. 정보를 얻기 위해 최소지지도를 낮추게 되면 후보항목집합과 그로부터 발견되는 규칙의 수가 매우 많아지게 되며 이를 모두 열거하기 위해 상당히 긴 처리시간과 I/O 시간이 필요하게 된다[3,7,11]. 빈발항목을 찾는 Apriori알고리듬을 개선한 기존의 해법[2,6,9,10,13]들 대부분이 규칙을 찾기 위해 지지도 조건을 필요로 하므로 이와 같은 문제점은 공통적으로 발생하게 된다. 지지도가 낮으면서 신뢰도가 높은 경우에 대해 규칙을 효율적으로 발견하기 위한 연구[3,4,7,11,12]가 최근 활발히 이루어지고 있으며 여러 응용분야들이 소개되고 있다. 예를 들어 '캐비어와 보드카'처럼 항상 같이 구매되어 의미 있는 규칙을 제공해 줄 수 있지만 두 항목을 같이 구매하는 사람이 적어 지지도가 낮은 규칙을 발견하는데는 유용하지 않은 경우, web-page-link분석과 많이 발생하지 않는 희귀한 질병의 여러 발병 원인을 발견하는 일과 같은 분야등과 같이 지지도를 고려하지 않고 높은 신뢰도를 갖는 규칙을 발견하는 것이 필요한 많은 응용분야가 존재한다. 본 연구는 지지도 제약조건 없이 신뢰도 조건만을 가지고 2-항목집합간 규칙을 발견하는데 CP(Confidence Pattern)-Tree 구조를 제안한다. 그리고 Tree구조를 이용해 지지도 제약조건 없이 임의의 신뢰도를 갖는 3-항목집합이상간의 규칙을 찾는 방법을 제안한다. 본 논문은 다음과 같이 구성되어 있다. 2장에서 트리 구조를 이용한 알고리

듬을 제시하고 3장에서는 3-항목집합이상에서 연관 규칙을 찾는 방법에 대해 기술한다. 마지막으로 결론에 대하여 기술한다.

2. CP(Confidence Pattern)-Tree 알고리듬

본 논문에서 제안하고자 하는 CP-Tree알고리듬은 데이터베이스를 검색하면서 트리 구조를 구성한다. 각 column에 대한 발생횟수 ones(c_i)의 계산과 이로부터 maxmis(c_i)를 계산하기 위해 데이터베이스를 한번 검색하고 트랜잭션의 발생 항목들을 트리에 저장하기 위해 데이터베이스를 한번 더 검색한다. 트리를 구성해 나가면서 miss pattern의 수를 세어 maxmis(c_i)를 초과하는 노드는 삭제하게 되며 2-항목집합에 대한 연관 규칙뿐만 아니라 100% 신뢰도를 만족하는 3-항목집합의 연관규칙도 찾을 수 있다. 또한 밀집된 데이터베이스의 경우 트리 크기에 영향을 적게 받는다.

알고리듬 단계

[단계1] CP-Tree List 생성
- 1'st DB Scan
- maxmis(c_i)계산, maxmis(c_i)의 내림차순으로 List 생성

[단계2] CP-Tree 구성
- 2'nd DB Scan
- 각 노드는 link로 연결
- Hit count 와 Miss count 갱신
- Miss count 가 maxmis(c_i)를 초과하는 노드 삭제

[단계3] 연관 규칙 발견
- Tree의 leaf 노드에서부터 가지별로 규칙을 발견

2.1 CP-Tree list 구성

CP-Tree list를 구성하기 위해 사용되는 기호는 다음과 같다.

ones(c_i) : The number of 1's for each column c_i
maxmis(c_i) : The maximum number of isses for each column

* : list에서 한번이라도 발생한 column에 표시

maxmis(c_i)는 최소신뢰도(minimum confidence)가 주어지면 식(1)에 의해서 계산할 수 있다.

$$\text{maxmis}(c_i) = \lfloor (1 - \min conf) \times ones(c_i) \rfloor \quad (1)$$

주어진 데이터베이스를 한번 검색하여 각 속성에 대한 ones(c_i)를 구한 후, maxmis(c_i)를 계산하여 list를 구성한다. list는 각 column c_i와 거기에 해당하는 maxmis(c_i)의 쌍으로 표현되며 maxmis(c_i)에 대해 내림차순으로 구성한다. maxmis(c_i)가 같을 경우 알파벳 내림차순에 의해서 list를 구성한다. [그림1-b]은 [그림1-a]의 예제 데이터베이스에 대해 최소신뢰도는 80%인 경우 CP-Tree list를 구성한 것이다. 데이터베이스에 대한 첫 번째 검색을 통해서 list를 구성한 후 두 번째 검색에서 트랜잭션을 읽어나가며 CP-Tree구조를 형성해 나간다.

	a	b	c	d	e
r_1			1		1
r_2		1			1
r_3			1		1
r_4			1	1	
r_5			1	1	
r_6	1			1	1
r_7		1		1	1
r_8			1	1	1
r_9	1	1		1	1

(a) 예제 데이터베이스

e/1
d/1
c/1
b/0
a/0

(b) CP-Tree list

[그림1] 예제 데이터베이스와 CP-Tree list

2.2 CP-Tree 구성

CP-Tree를 구성하기 위해 사용되는 기호는 다음과 같다.

{ } : root node of Tree
$c_{i\,\bullet}^{\,a}$: node of Tree. (a : hit count, b : miss count, c_i : column)
• : 트리에서 maxmis(c_i)를 초과한 중간노드에 표시

트리 구조를 형성할 때는 ones(c_i)가 큰 순서대로 노드를 구성해 나간다. 결국 하나의 트랜잭션에서 가장 작은 ones(c_i)값을 가지는 column c_i가 leaf node를 이루게 된다. 만약 ones(c_i) = ones(c_j)일 경우는 알파벳순서(i < j)에 의해 알파벳순서가 먼저인 column c_i를 leaf node 쪽으로 구성한다. ones(c_i) < ones(c_j) 이면 식(2)에 의해 conf(c_j,c_i) < conf(c_i,c_j)를 만족하므로 트리 구조가 leaf node에서 root node로 올라갈수록 ones(c_i) ≤ ones(c_j) 만족하기 때문에 트리의 아래에서부터 위로 향하면서

주어진 신뢰도를 확실히 만족하는 $c_i \rightarrow c_j$ 규칙을 찾을 수 있게 된다.

$$\text{신뢰도} = conf(c_i, c_j) = \frac{ones(i \cap j)}{ones(i)} \qquad (2)$$

트리에 노드가 새로 생기게 되면 같은 항목의 노드는 link로 연결되게 된다. List를 구성한 후 두 번째 데이터베이스 스캔을 통해서 트리를 구성해 나간다. 트랜잭션이 발생하면 해당 노드의 Hit count는 하나씩 증가하고 해당 노드가 link신하게 된다. 로 연결되어 있는 경우 Miss count 값을 갱link가 연결되는 경우는 크게 두 가지 경우로 나눌 수 있다. 첫째는 list로부터 link가 연결되는 경우와 트리에 이미 존재하는 노드로부터 link가 연결되는 경우이다. list로부터 link가 연결되는 경우는 다시 두 가지 경우로 나뉠 수가 있는데 list에 처음으로 연결되는 경우, 즉 * 표시가 없는 list의 c_i으로부터 연결되는 경우와 트리 에서 삭제되었다가 다시 list로부터 연결되는 즉, * 표시가 있는 list의 c_i으로부터 link가 연결되는 경우이다. 각 경우에 대한 값은 식(3)과 같다.

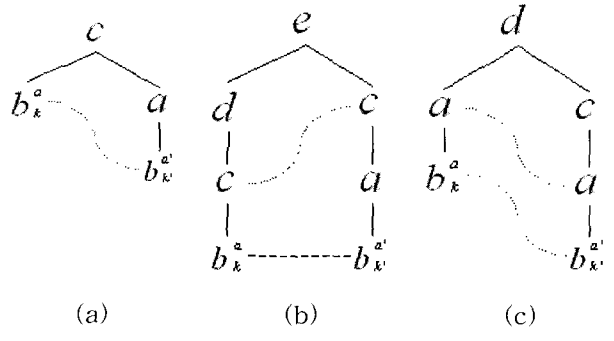
$$c_b^a = c_k^1 \qquad (3)$$

k = 0 : c_i 가 * 표시되어 있지 않은 경우
k = 해당 트랜잭션의 바로 상위 노드의 hit count-1
: c_i 가 * 표시되어 있는 경우

CP-Tree list로부터 노드의 link가 연결되는 경우 외에 노드와 노드간에 link가 연결될 수 있다. 노드간 link로 연결될 경우 두 노드에 대한 miss count를 수정해 줘야 하며 기준이 되는 조건은 새롭게 link로 연결되는 노드의 상위노드가 링크로 연결되는 다른 쪽 노드의 가지에 존재하는가 존재하지 않는가가 판단 기준이 된다.

i) 존재하는 경우

[그림2-a, b]처럼 한쪽만 존재하는 경우와 [그림2-c]



[그림2] 같은 노드가 존재하는 경우

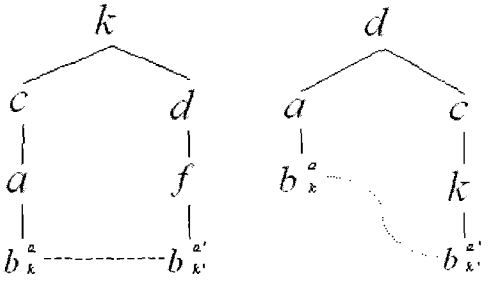
처럼 양쪽에 존재하는 경우가 있다.

한쪽에만 존재하는 경우에는 식(4)에 의해 miss count 값이 수정되며 양쪽에 존재하는 경우에는 식(5)과 같이 수정된다.

$$k' = a + k' \qquad (4)$$

$$k' = k \qquad (5)$$

만약 해당 트랜잭션에서 새롭게 link 연결이 생기지 않으면서 hit count 만 증가하면 한쪽만 존재하는 경우에



[그림3] 같은 노드가 존재하지 않는 경우

는 해당 트랜잭션이 아닌 link로 연결된 노드의 miss count 값은 1증가하게 되며 양쪽에 모두 존재하는 경우에는 k′ 와 k 값 중 큰 값으로 k′=k 가 결정된다.

ii) 존재하지 않는 경우

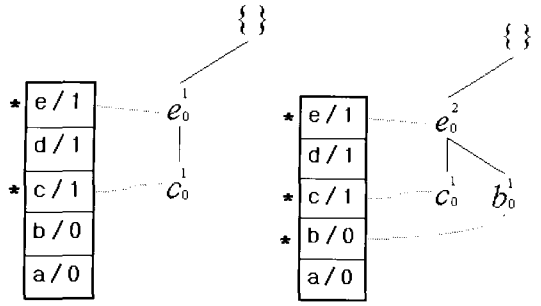
[그림3]처럼 링크로 연결된 두 노드에 같은 노드가 전혀 없는 경우는 식(6)과 같이 양 노드의 miss count 값을 수정해줘야 한다.

$$k' = a + k' \qquad k = a' + k \qquad (6)$$

만약 해당 트랜잭션에서 새롭게 link 연결이 생기지 않으면서 hit count 만 증가하는 경우에는 해당 트랜잭션이 아닌 곳에 위치한 노드의 miss count 만 1증가시킨다.

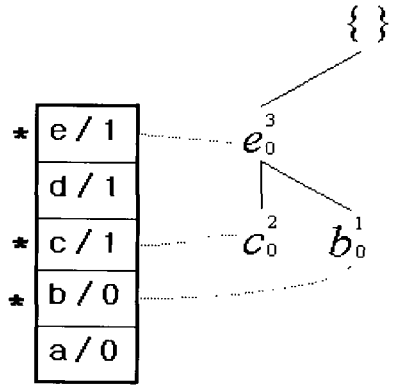
2.3 예제분석

[그림4]는 [그림1-a]에 대해 CP-Tree알고리듬을 수행하는 과정 중 r_1,r_2을 검색한 후의 결과이다. * 표시가 없는 column으로부터 노드가 처음으로 연결될 때는 c_b^a 값이 항상 c_0^1으로 된다. 예를 들어 [그림4-b]의 b_0^1는 트랜젝션에서 b가 한번 발생했으며 b가 발생하면 e도 발생했다는 것을 보여준다 (hitcount=1, miss count=0). root 노드 바로 밑의 노드는 그 위에 노드가 더 이상 노드가 없으므로 miss가 전혀 발생하지 않아 항상 0값을



(a) CP-Tree after r1 (b) CP-Tree after r2

[그림4] CP-Tree 구성: Scan r_1, r_2

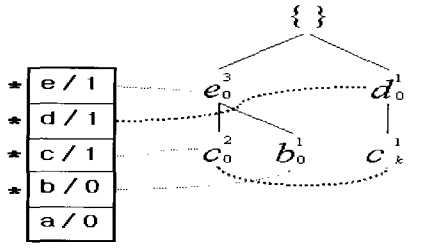


[그림5] CP-Tree 구성 : Scan r_3

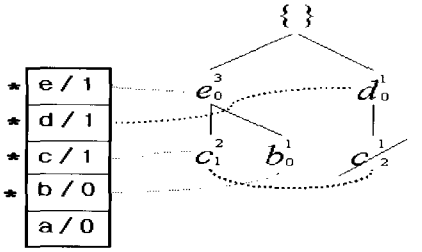
miss count 값으로 갖게 된다. [그림5]는 r3트랙잭션에 대해 CP-Tree를 구성한 것이다. 기존의 트리 구조에 영향을 미치지 않으므로 해당하는 트랜잭션 가지에 대해 hit count만 하나씩 증가하게 된다.

[그림6]은 트랜잭션 r_4를 검색했을 때 트리 구조가 변하는 과정을 나타낸 것이다. [그림6-a]처럼 두 개의 새로운 링크가 트리 구조에 만들어진다. "항목d"는 처음으로 나타나므로 CP-Tree list에 * 표시가 붙으면서 list에서 연결된 노드는 d_0^1 값을 갖는다. 나머지 새롭게 노드 c_0^2 에서 노드 c_k^1 로 연결되는 링크는 [그림3]의 경우처럼 같은 노드가 존재하지 않는 경우로 [그림6-b]처럼 c_k^1 의 k값은 c_0^2 의 hit count 값인 2와 초기 k=0 값을 더해서 k=2+0=2 로 수정되고 c_0^2 의 miss count 0 역시 c_k^1 의 hit count 1을 반영하여 0+1=1 로 수정된다. 그러면 c_k^1 노드의 miss count 가 2가 되어 maxmis(c)=1을 초과하므로 삭제되어 [그림6-c]와 같은 트리 구조가 된다.[그림7]은 나머지 트랜잭션에 대해 CP-Tree를 적용시킨 과정을 트랜잭션별로 보여준다. [그림 7-b]를 보면 r6를 검색하면 두 개의 새로운 링크가 생기는데, 특히 "항목d"에 대한 링크는 두 노드 모두 miss count 값이 식(6)에 의해 수정되어야 하나 d_0^2 의 miss count 는 수정되지 않는다. root 노드 바로 밑의 노드는 그 위에 아무 노드도 없기 때문에 항상 miss count 값이 0을 갖기 때문이다. d_2^1 노드는 "항목d"의 maxmis(d)를 초과했지만 자식노드를 갖고 있기 때문에 삭제될 수 없으며 노드에 "•" 를 해서 maxmis(d)를 초과했음을 표시한다. [그림 7-d]에서 "항목c"에 대한 노드는 list로부터 다시 연결되나 이미 * 표시가 있으므로 바로 윗 노드인 "항목d"의 hit count에서 자신의 hit count를 뺀 값을 miss count 값으로 갖게 된다. [그림7-d]에서처럼 maxmis(c)=2 값을 초과하므로 노드는 삭제된다. 트랜잭션 r9까지의 검색과정을 모두 거치면 [그림7-e]와 같은 트리 구조가 완성되며 이로부터 2-항목집합간 신뢰도 80%를 만족하는 연관 규칙을 찾을 수 있다.

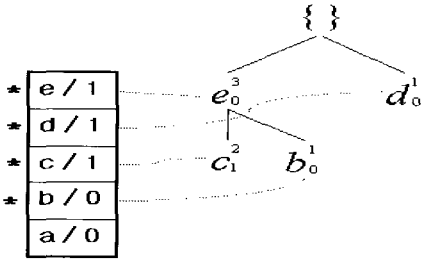
완성된 트리에서의 연관 규칙은 leaf 노드를 기준으로 하며 각 가지별로 연관 규칙을 찾게 된다. [그림7-e]로부터 a → d, a → e , b → e인 연관 규칙을 찾을 수 있다. d → e 인 연관 규칙은 "d"의 miss count가 maxmis(d) = 1을 초과했기 때문에 규칙으로 발견될 수



(a)

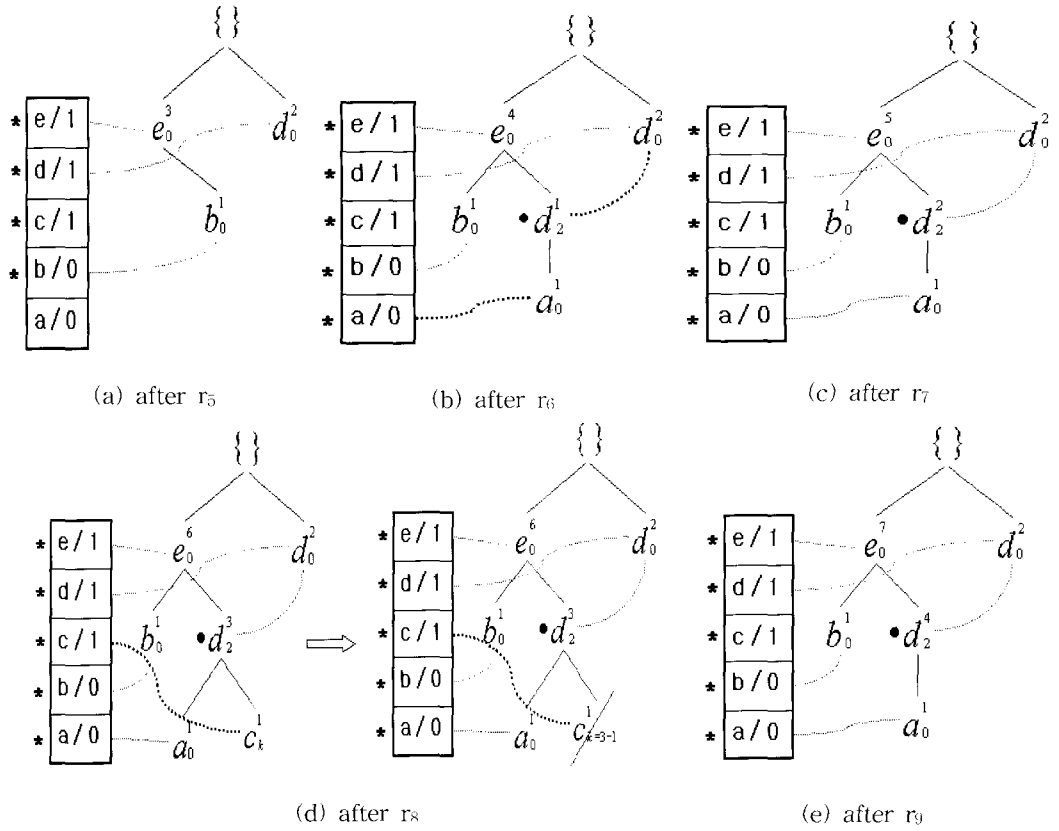


(b)



(c)

[그림6] CP-Tree 구성 : Scan r4



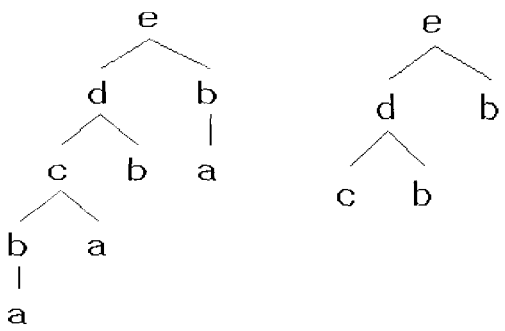
(a) after r_5 (b) after r_6 (c) after r_7

(d) after r_8 (e) after r_9

[그림7] CP-Tree 구성 : Scan r_5 - r_9

없다. 또한 위에서 발견한 규칙을 갖고 100%신뢰도를 만족하는 규칙을 발견할 수 있다. a → d, a → e 두 연관 규칙은 신뢰도 100%이므로 ad → e, ae → d 역시 신뢰도 100%를 갖게 되어 3-항목집합 이상의 연관 규칙도 찾을 수 있다. 그러나 ab → d 인 연관 규칙도 [그림1-a]에서 보듯이 100%의 신뢰도를 갖고 있지만 [그림7-e]를 통해서는 발견할 수 없다. 왜냐하면 트리에서 "항목a"와 "항목d"가 같은 가지에 있지 않기 때문이다. 3-항목집합이상의 모든 연관 규칙을 찾는 방법은 3장에서 논한다. 우리 주변의 많은 데이터들은 수백 개의 속성(column)을 갖고 있다. 그러나 [그림8-a]처럼 밀집된 경우는 거의 없고 수백 개의 속성 중 각 트랜잭션당 10개에서 20정도의 속성 값을 갖는 경우가 대부분이다.
만약 밀집된 형태의 데이터베이스로부터 지지도를 고려하지 않고 연관 규칙을 찾기 위해 DMC알고리듬[4]을 적용시킨다면 무수히 많은 join이 일어나게 된다.

	a	b	c	d	e
r_1	0	0	1	1	1
r_2	0	1	1	1	1
r_3	1	0	1	1	1
r_4	1	1	1	1	1
r_5	1	1	0	0	1
r_6	0	1	0	1	1



(a)밀집된 데이터베이스 (b)일반적 트리모양 (c)CP-Tree

[그림8] 밀집된 데이터베이스의 경우

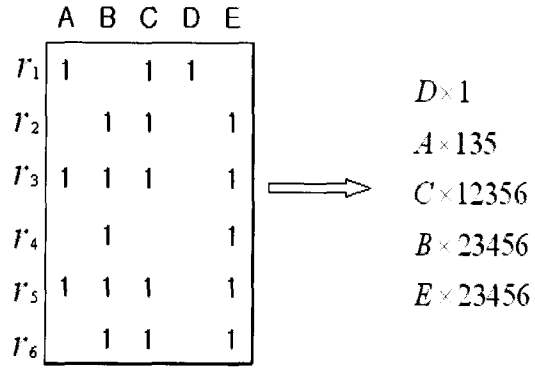
그러나 트리 구조를 사용하면 [그림8-c]처럼 트리 구조에 중복되어 표현되기 때문에 간단하게 트리 구조로 표현할 수 있다.

2.4 알고리듬 분석

Apriori 알고리듬[1]에서 2^m(m: number of items)의 개의 모든 빈발항목집합을 찾는데 전체적으로 $O(r \cdot n \cdot 2^l)$의 계산량 (r: number of maximal frequent itemsets, n: number of transaction, l: length frequent itemset)이 필요하며 NP-Complete[8]문제로 알려져 있다. 또한 발견한 빈발항목집합으로부터 연관규칙을 찾는 것은 $O(s \cdot 2^l)$의 계산량(s: number of frequent itemsets, l: longest frequent itemset)이 다시 필요하다[13]. 그러나 2-항목집합에 대한 경우 2^m개의 빈발항목집합을 찾기 위해 m(m-1)/2번의 join 과정만이 필요하게 된다. DMC알고리듬[4]은 최악의 경우 m^2의 비교횟수가 발생하지만 신뢰도조건이 높을수록 Apriori보다 더 좋은 결과를 보인다. 본 논문에서 제안한 CP-Tree알고리듬 역시 최악의 경우 m^2의 비교횟수가 발생하나 트리 구조에 표현된 3-항목집합이상의 연관 규칙과 밀집된 데이터베이스인 경우 DMC보다 좋은 결과를 보인다.

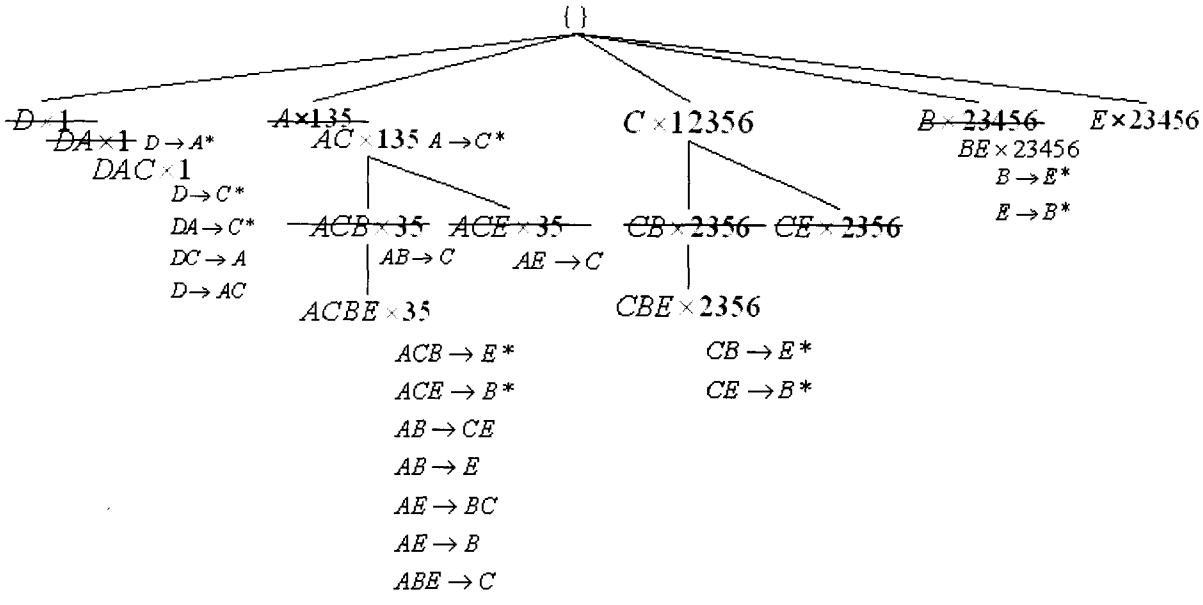
3. 3-항목집합이상에 대한 연관 규칙 탐색 방법

본 논문에서 제안한 CP-Tree알고리듬은 3-항목집합



[그림9] Vertical form

이상의 연관 규칙도 발견할 수 있으나 100% 신뢰도를 가지면서 트리 구조에 표현된 노드에 대해서만 발견할 수 있다. 일반적으로 3-항목집합 이상에 대한 연관규칙을 지지도 제약조건 없이 찾기 위해서는 지수적인 계산량이 필요하다. 왜냐하면 속성(column)들의 모든 조합을 생각해야 하기 때문이다. 3장에서는 트리 구조를 사용해서 좀 더 간단히 3-항목집합 이상에 대한 연관 규칙을 찾는 방법을 제안한다. 그렇지만 트리 구조를 사용하더라도 최악의 경우 지수적 계산량을 갖게 된다. 이 방법은 Galois connection[5]의 closure와 규칙들간의 포함관계, 그리고 그룹화기법을 통해 구성된다. 지지도제약조건 없이 신뢰도만을 고려해 연관 규칙을 찾기 위해서는 많은 경우를 고려해야 하며 따라서 트리에 많은 정보를 필요로 한다. 그러나 트리를 그룹화 시켜서 closure와



[그림10] CHARM을 이용한 3-항목집합이상의 연관규칙 탐색

발견과 윗 노드에서 발견한 규칙을 기준으로 하위노드에서 규칙을 찾는다면 모든 발생 가능한 경우를 검토하는데 필요한 join 수보다 더 적은 수의 join수가 소요되며 트리 크기도 줄일 수 있다. [그림9]처럼 전체 데이터베이스의 검색을 통해 각 column의 ones(c_i)를 계산한 후 vertical form으로 데이터베이스를 변환한다. [그림10]과 같이 ones(c_i)가 작은 순서대로 트리 구조를 만들며 트리의 노드에는 트랜잭션정보(Tid list)를 포함시킨다. 만약 [그림10]의 "C", "B", "E"처럼 ones(ci)가 같은 경우 포함하고 있는 Tid list의 첫 트랜잭션 Tid가 가장 작은 것을 갖고 있는 c_i를 트리의 왼쪽에 구성한다. 트리를 구성하는 과정은 CHARM[13]알고리듬과 같다. CHARM알고리듬은 Galois connection의 closure을 이용해 빈발항목집합을 찾는 방법이다. 그러나 최소지지도를 만족하는 ci에 대해서만 알고리듬을 수행하게 된다. 최소지지도를 고려하지 않고 신뢰도만을 고려해 연관 규칙을 찾으려면 모든 c_i에 대해 트리를 구성해야 한다. 앞서 언급했듯이 3-항목집합이상의 연관 규칙을 찾기 위해서는 모든 c_i에 대한 정보를 갖고 있어야 하므로 지수적 계산량의 join이 필요하다. 그러나 CHARM을 이용해 필요한 join 수를 줄일 수 있다. 그러나 최악의 경우는 역시 지수적 계산량을 나타내는 단점이 있다. 각 c_i에 대해 그룹화 시키며 closure발견과 윗 노드에서 발견한 규칙을 기준으로 하위노드에서 규칙을 찾으면 주어진 신뢰도를 만족하는 모든 연관 규칙을 찾을 수 있다. [그림10]은 [그림9]의 데이터를 갖고 100%신뢰도를 만족하는 연관 규칙을 찾기 위해 CHARM의 방법을 사용하여 트리를 구성한 예이다. * 표시가 붙은 연관 규칙은 Galois connection의 closure개념으로 찾을 수 있는 규칙이고 그렇지 않은 규칙들은 윗 노드에서 발견한 규칙을 기준으로 하위노드에서 규칙을 찾은 경우이다. 각 노드가 Tid정보를 갖고 있으므로 임의의 신뢰도를 만족하는 연관 규칙을 찾고자 할 때 Tid 정보를 이용하여 miss count를 계산할 수 있다.

4. 결론

Apriori 알고리듬[1]에서 2-항목간의 빈발항목만 발견하는 경우 m(m-1)/2번의 join 과정과 2번의 데이터베이스 검색과정이 필요하다. DMC알고리듬[4]은 최악의 경우 m^2의 비교횟수가 발생하며 가장 큰 단점은 지지도 조건 없이 2-항목집합에 대한 연관 규칙만을 찾을 수 있다는 점이다. 제안한 알고리듬 CP-Tree는 트리 구조를 구성하여 2-항목집합간 규칙뿐만 아니라 3-항목집합 이상에서 100% 신뢰도를 갖는 규칙도 발견할 수 있으며 트랜잭션당 항목의 발생빈도가 높은 경우에도 효율적이다. 트리 구조를 통해 지지도 제약조건 없이 3-항목집합 이상을 갖는 연관 규칙을 찾는 문제는 closure를 이용해 해결할 수 있다. 최종적으로, CP-Tree는 지지도 조건 없이 신뢰도 조건만을 갖고 2-항목집합에 대한 연관 규칙을 찾는 문제, 데이터베이스내의 트랜잭션이 밀집되어 있는 문제 그리고 최소신뢰도가 높은 문제에 있어 기존의 지지도 또는 신뢰도를 기반으로 한 알고리듬보다 더 좋은 결과를 보인다.

참고문헌

[1] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules in Large Databases", *In Proc. of the 20th Int'l Conference on VLDB*, pp. 487-499, September 1994

[2] R. Bayrardo Jr., "Efficiently Mining Long Patterns from Database", *In Proc. of the ACM SIGMOD Int'l Conference on management of Data*, pp. 85-93, June 1998

[3] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, et al, "Finding Interesting Associations without Support Pruning", *IEEE Transactions on Knowledge and Data Engineering*, 13, pp 64-78, 2001.

[4] S. Fujiwara, J. D. Ullman, R. Motwani, "Dynamic Miss-Counting Algorithms: Finding Implication and Similarity Rules with Confidence Pruning", *Proc. of the 16th ICDE*. 2000.

[5] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[6] J. Han, J. Pei, Y. Yin., "Mining Frequent Patterns without Candidate Generation", *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data(SIGMOD'00)*, May 2000.

[7] J. Li, X. Zhang, G. Dong, K. Ramamohanarao, and Q. Sun, "Efficient Mining of High Confidence Association Rules without Support Thresholds", *Proc. of 3*rd European Conference on Principles and Practice of Knowledge Discovery in Database, September 1999.

[8] M. Ogihara and M.J.Zaki "Theoretical foundations of association rules". *In 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, June 1998.

[9] J. Park, M. Chen, and P. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", *In Proc. of the ACM SIGMOD Int'l*

Conference on Management of Data, pp.175-186, May 1995

[10] P. Pei, J. Han, and R. Mao, "Closet: an efficient algorithm for mining frequent losed itemsets", *Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery(DMKD'00)*, May 2000.

[11] W. Perrizo, Qin Ding, Qiang Ding, and A. Roy, "Deriving High Confidence Rules from Spatial Data using Peano Count Trees", *Proc. of the 2nd International Conference* 2001.

[12] K. Wang, S. Zhou, Y. He, "Growing Decision Trees on Support-less Association Rules", *KDD*, August 2000.

[13] J. Zaki, Ching-Jui Hsiao, "CHARM: An Efficient Algorithm for Closed Association Rule Mining", RPI Technical Report 99-10, 1999.