Journal of the Korean Institute of Industrial Engineers
Vol. 27, No. 1, pp. 30-36, March 2001.

트리 구조를 이용한 연관규칙의 효율적 탐색

김창오 · 안광일 · 김성집 · 김재련†

한양대학교 산업공학과

An Efficient Tree Structure Method for Mining Association Rules

Chang-Oh Kim · Kwang-Il Ahn · Seong-Jip Kim · Jae-Yearn Kim

Department of Industrial Engineering, Hanyang University, Seoul

We present a new algorithm for mining association rules in the large database. Association rules are the relationships of items in the same transaction. These rules provide useful information for marketing. Since Apriori algorithm was introduced in 1994, many researchers have worked to improve Apriori algorithm. However, the drawback of Apriori-based algorithm is that it scans the transaction database repeatedly. The algorithm which we propose scans the database twice. The first scanning of the database collects frequent length 1-itemsets. And then, the algorithm scans the database one more time to construct the data structure *Common-Item Tree* which stores the information about frequent itemsets. To find all frequent itemsets, the algorithm scans *Common-Item Tree* instead of the database. As scanning *Common-Item Tree* takes less time than scanning the database, the algorithm proposed is more efficient than Apriori-based algorithm.

Keywords : data mining, association rule, apriori algorithm, tree data structure, hash table

1. 서 론

대용량의 데이터베이스에 숨어 있는 정보를 발견해 내는 연구 분야를 데이터 마이닝(data mining)이라고 한다. 데이터 마이닝에서 사용되는 기법으로는 시장 바구니 분석(market basket analysis), 군집화(clustering), 신경망 분석(artificial neural network), 그리고 의사결정 나무(decision tree) 등이 있다(M. Berry and G. Linoff, 1997). 최근 들어서는 바코드 기술의 발전, 신용카드 거래의 증가로 인해 고객들이 어떤 제품을 구입하는지에 대한 많은 세일즈 정보들이 생성, 수집되어 데이터베이스에 저장되고, 기업들은 이러한 정보들을 마케팅에 적용하기 위해 노력하고 있다. 특히 고객이 함께 구입하는 제품들 간에 존재하는 연관규칙을 찾아내는 연구를 '시장 바구니 분석'이라고 부른다. 예를 들어, '빵을 구입하는 고객 중 80%는 우유를 함께 구입한다'와 같은 규칙을 발견해 내는 것이다.

데이터베이스에서 항목집합(itemset)에서의 연관규칙을 발견해 내는 문제는 Agrawal *et al.* (1993)에 의해 처음 소개되었고, 연관규칙을 발견해내는 Apriori 알고리듬(Agrawl and Srikant, 1994)이 발표된 이후로 Apriori를 근거로 한 효율적인 알고리듬들이 다수 제시되었다. DHP 방법(J. Park *et al.*, 1995)은 해시 테이블을 이용해 Apriori 알고리듬 중 길이가 2인 후보 항목집합(candidate itemset)생성의 효율을 높였고, DIC(Brin *et al.*, 1997)와 DICIP(Tang, 1998)는 데이터베이스를 여러 개로 분할(partition)하여 데이터베이스를 검색하였다. 그리고 범주적 속성(categorical attributes)의 데이터 뿐만 아니라 수치적 속성(numeric attributes) 데이터의 연관규칙을 찾는 알고리듬(Srikant and Agrawal, 1996), 계층도(taxonomy : is-a hierarchy)를 이용해 일반화된 연관규칙을 찾는 알고리듬(Srikant and Agrawal, 1995), 항목간의 순차적 패턴(sequential pattern)을 찾는 알고리듬(Srikant and Agrawal, 1996; Zaki, 1998), 갱신과 유지(update and maintenance)를 위한 알고리듬(Cheung *et al.*, 1996), 사용자 제약(user-defined constraints)을 고려한 알고리듬(Srikant *et al.*, 1997), 항목간의 주기적 속성(cyclic attributes)

† 연락저자 : 김재련 교수, 133-791, 서울 성동구 행당동 17 한양대학교 산업공학과, Fax : 02-2299-0889, e-mail : jyk@email.hanyang.ac.kr
1999년 12월 접수, 1회 수정 후, 2000년 11월 게재 확정.

을 찾는 알고리듬(Özden *et al.*, 1998; Han *et al.*,1998), 마케팅의 세일즈 데이터(sales data) 뿐만 아니라 인구조사 데이터(census data)와 같이 상대적으로 패턴 길이가 긴 경우에 적합한 알고리듬(Bayardo Jr. and Agrawal, 1998) 등 다양한 연구가 이루어져왔다.

그러나 Apriori를 근거로 한 기존의 알고리듬들은 빈발 항목집합을 찾기 위해 데이터베이스를 여러 번 검색하기 때문에 효율적이지 못하다. 본 연구에서는 트리 구조를 이용해 데이터베이스 검색을 두 번만 수행하여 연관관계를 찾는 효율적인 방법을 제시한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 연관규칙 탐사 문제를 정의하고 연관규칙 탐사에서 사용하는 기본적인 용어를 설명한다. 3장에서는 제안하는 알고리듬에 대해 설명하고, 4장에서 실험을 통해 제안한 알고리듬의 효율성을 기존의 해법들과 비교 분석한다. 마지막으로 5장에서 결론을 맺는다.

2. 연관규칙 탐사

2.1 빈발항목 집합

$I = \{i_1, i_2, \cdots, i_m\}$를 항목(item)들의 전체 집합이라고 할 때, 항목집합(itemset)은 I의 부분집합을 말한다. 트랜잭션 T는 TID라는 유일한 식별자를 가지고 있는 항목집합이며, 검색하고자 하는 데이터베이스는 트랜잭션 T로 구성된 집합이다. 어떤 항목집합을 X라고 했을 때, T가 X를 포함하면, 즉 $X \subseteq T$를 만족시키면 트랜잭션 T는 항목집합 X를 지지한다고 한다. 데이터베이스에서 항목집합 X를 지지하는 트랜잭션의 수를 X의 지지수(support count)라고 부르며, 트랜잭션의 총 개수에 대한 X의 지지수의 비율을 X의 지지도(support)라고 부른다. 연구자가 지정한 최소 지지도(minimum support)를 만족하는 항목집합을 빈발 항목집합(frequent (혹은 large) itemset)이라고 정의하며, k개의 항목으로 이루어진 빈발 항목집합을 빈발 k - 항목집합(frequent k - itemset)이라고 부른다.

2.2 연관규칙

X, Y가 항목집합일 때, 연관규칙은 $X \rightarrow Y$의 형태로 표현하며, 한 트랜잭션에 X가 존재하면 Y도 함께 존재한다는 것을 의미한다. 여기서 $X \subseteq I$, $Y \subseteq I$, 그리고 $X \cap Y = \varnothing$이다. 연관규칙 $X \rightarrow Y$의 지지도는 전체 트랜잭션에서 $X \cup Y$를 포함하고 있는 트랜잭션의 비율을 말한다. 그리고 $X \rightarrow Y$의 신뢰도는 X의 지지도에 대한 $X \cup Y$의 지지도의 비율을 말한다. 연관규칙은 빈발 항목집합을 이용하여 쉽게 생성할 수 있기 때문에 연관규칙 탐사 문제는 빈발 항목집합을 발견하는 문제로 축소할 수 있다(Park *et al.*, 1995).

3. 공통항목트리(Common-Item Tree) 알고리듬

빈발 항목집합을 찾기 위해 데이터베이스를 여러 번 검색하는 기존의 Apriori기반의 알고리듬과는 달리 본 논문에서 제안한 방법은 데이터베이스를 여러 번 검색하는 대신 공통항목트리를 만들어 이것을 검색한다. 공통항목트리는 데이터베이스의 요약정보로서 크기가 데이터베이스보다 작은 데이터구조이다.

본 장은 다음과 같이 구성되어 있다. 3.1절에서는 공통항목트리와 해시 테이블의 생성과정을 설명하고 3.2절에서는 공통항목트리를 검색하여 항목집합의 지지수를 계산하는 과정을 설명한다. 그리고 3.3절에서는 제안한 알고리듬의 전 과정을 보여주고, 3.4절에서 제안한 알고리듬의 효율성을 설명한다.

본 알고리듬에서 사용하는 기호 및 용어를 정리하면 다음과 같다.

minsup : 최소 지지도(minimum support)
C_k : 빈발 가능성이 있는 후보 k - 항목집합들의 집합 (candidate k - itemsets)
C_{max} : 최대 후보 빈발 항목집합(maximal candidate frequent itemset) 즉, 빈발 1 - 항목집합들의 합집합
L_k : 빈발 k - 항목집합들의 집합
최대 공통항목집합(Maximal Common Itemset) : 트랜잭션과 C_{max} 사이의 최대 공통집합 (간단히 *MCI*로 사용)
결여 항목집합(Absent itemset) : 항목집합과 C_{max}를 비교했을 때 결여된 항목들의 집합. 즉, C_{max} - 항목집합 (간단히 *A_Items*로 사용)

3.1 공통항목트리 및 해시 테이블의 생성

이 절에서는 공통항목트리 생성에 대해 설명한다. <표 1>은 예제 데이터베이스이다. 최소 지지도는 40 %라고 가정한다. 즉, 5개의 트랜잭션 중에서 2개 이상의 트랜잭션에 존재하는 항목집합이 빈발 항목집합이다(최소 지지수 = 2).

공통항목트리는 다음 두 단계를 거쳐 만들어진다. 첫 단계는 데이터베이스를 검색하여 빈발 1-항목집합들을 찾고, 최대 후보 빈발 항목집합 C_{max}를 생성하여 공통항목트리의 뿌리 노드로 만든다. <그림 1> 예제 데이터베이스의 빈발 1-항목집합은 {1}, {2}, {3}, {4}, {5}이고, C_{max}는 빈발 1-항목집합의 합집합인 {1, 2, 3, 4, 5}이다.

표 1. 예제 데이터베이스

TID	Items
100	1 2 3 5
200	2 3 5
300	2 3
400	1 3 4 5 6
500	1 3 4 5

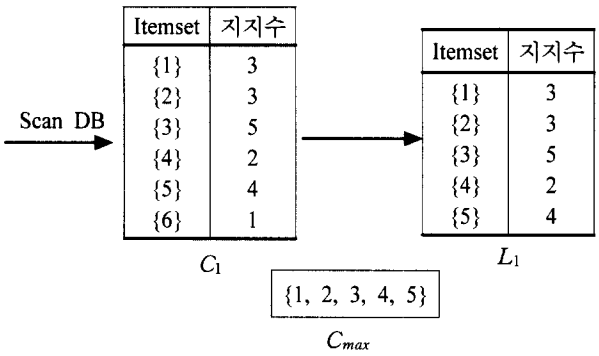

그림 1. 예제 데이터베이스의 빈발 1-항목집합과 C_{max}.

둘째 번 단계는 데이터베이스를 한번 더 검색하여 공통항목트리를 작성한다. 첫 단계에서 만들어진 뿌리 노드로부터 시작하여 다음의 네 과정을 모든 트랜잭션에 대해서 수행한다.

과정 1 : 트랜잭션과 C_{max} 사이의 최대 공통항목집합(*MCI*)을 찾는다.

과정 2 : 최대 공통항목집합을 C_{max}와 비교하여 결여 항목집합(*A_Items*)을 찾는다.

과정 3 : 공통항목트리를 검색하여 최대 공통항목집합과 같은 노드가 있는지 찾는다. 검색은 뿌리 노드부터 결여 항목집합의 항목들의 사전적 크기 순서에 따라 이루어진다. 만약 찾으면 노드의 횟수를 1 증가시키고 못 찾으면 새 노드를 만들어 횟수를 1로 설정한다. <그림 2>는 과정 3을 수행하는 공통항목트리 작성 알고리듬이다.

```
// ptr은 뿌리 노드를 가리키는 포인터
Insert(ptr, A_Items){
   IF (A_Items= ∅) {
      ptr이 가리키는 노드의 횟수를 1 증가시킨다. return;
   }ELSE IF (ptr 노드에 연결된 노드 중에 A_Items의 첫 항목이
   결여된 노드가 있다){
      ptr이 찾은 노드를 가리키도록 한다.
      A_Items에서 첫 항목을 제거한다.
      Insert(ptr, A_Items)
   }ELSE{
      ptr이 가리키는 항목집합노드에서 A_Items의 첫 항목이 결
      여된 노드를 생성하여 ptr이 가리키는 노드와 연결시킨다.
      그리고 ptr이 이 새 노드를 가리키게 한다. 새로운 노드의 횟
      수는 0으로 설정한다.
      A_Items에서 첫 항목을 제거한다.
      Insert(ptr, A_Items)
   }
}
```

그림 2. 공통항목트리 작성 알고리듬.

과정 4 : 최대 공통항목집합으로부터 2-항목집합을 만들어 해시 테이블에 할당한다(Park *et al.*, 1995).

<표 1>에서 트랜잭션 *TID* 100과 C_{max} 사이의 최대 공통항목집합은 {1, 2, 3, 5}이다. 그리고 최대 공통항목집합을 C_{max}와 비교해보면 항목 {4}가 결여되어있다. 그러므로 뿌리 노드에서 항목 {4}가 결여된 자식노드를 찾는다. 아직 공통항목트리는 뿌리 노드밖에 없기 때문에 상응하는 노드가 존재하지 않는다. 따라서 <그림 3>에서처럼 {4}가 결여된 {1, 2, 3, 5}를 만들고 부모 노드인 뿌리 노드와 연결한다. 횟수는 1이 된다. 공통항목트리에서 노드의 오른쪽 상단의 숫자는 트랜잭션의 발생 횟수를 의미하고 연결선 위의 숫자(예 : ~4)는 결여된 항목을 나타낸다.

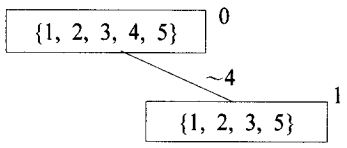

그림 3. TID 100을 공통항목트리에 삽입.

TID 200의 최대 공통항목집합은 {2, 3, 5}이고, C_{max}와 비교할 때 {1}과 {4}가 결여되어 있다. 결여된 항목들의 사전적 크기 순서에 따라서 먼저 뿌리 노드에서 항목 {1}이 결여된 노드 {2, 3, 4, 5}가 있는지 찾는다. 찾는 노드가 존재하지 않기 때문에 {2, 3, 4, 5} 노드를 생성하고 횟수를 0으로 설정한 후 뿌리 노드에 연결한다. 이 노드에서 항목 {4}가 결여된 {2, 3, 5} 노드를 찾는다. 이 노드도 존재하지 않기 때문에 항목집합 {2, 3, 5} 노드를 만들고 부모 노드인 {2, 3, 4, 5}에 연결한다. 횟수는 1로 설정한다. <그림 4>는 *TID* 200을 공통항목트리에 삽입시킨 결과이다.

<그림 5>는 <표 1>의 모든 트랜잭션을 공통항목트리에 삽입한 결과이다.

본 알고리듬에서는 공통항목트리 생성과 동시에 2-항목집합에 대해서 해시 테이블을 사용한다. 해시 테이블을 사용하는 이유는 후보 2-항목집합의 수를 줄여 알고리듬의 효율을 높이기 위한 것이다(Park *et al.*, 1995). <표 1>의 *TID* 100의 경우에 최대 공통항목집합이 {1, 2, 3, 5}이므로, 이 항목들을 조합하면 2-항목집합은 {1, 2}, {1, 3}, {1, 5}, {2, 3}, {2, 5}, {3, 5}가

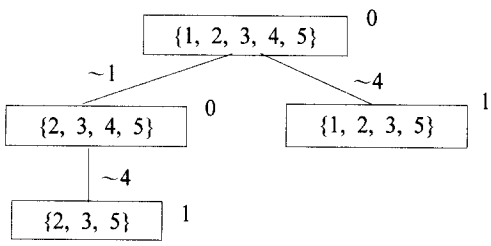

그림 4. TID 200을 공통항목트리에 삽입.

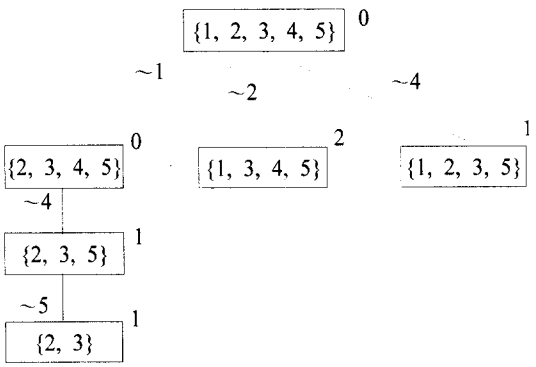

그림 5. 예제 데이터베이스에 대한 공통항목트리.

항목집합		{2,5}		{3,5}	{1,2}	{1,3}		{2,3} {1,5}
항목집합수	0	1	0	1	1	1	0	2
함수값	0	1	2	3	4	5	6	7

그림 6. *TID* 100에 대한 해시 테이블의 예.

된다. 예를 들어, 해시함수가 (2-항목집합의 첫번째 항목값 × 10 + 2-항목집합의 두번째 항목값)을 8로 나눈 나머지 값을 반환하는 함수라면, 항목집합 {1, 2}의 해시 함수값은 (1×10 +2)를 8로 나눈 나머지이므로 4가 되고, {1, 3}의 해시 함수값은 5가 된다. *TID* 100의 모든 2-항목집합에 대해 해시함수를 적용하여 해시 테이블에 할당하면 <그림 6>과 같이 된다.

그리고 <그림 7>은 <표 1> 예제 데이터베이스의 2-항목집합에 대해 완성한 해시 테이블이다. 2-항목집합 중에서 해시 테이블의 항목집합수가 최소 지지수(= 2)를 넘지 못하면 후보 2-항목집합에서 제외하여 후보 항목집합수를 줄인다. 따라서 항목집합 {1, 2}(항목집합수 = 2)는 후보 2-항목집합에서 제외된다.

항목집합		{2,5} {2,5}	{3,4} {3,4}	{3,5} {3,5} {3,5}	{1,2}	{4,5} {1,3} {4,5} {1,3} {1,3}	{1,4} {1,4}	{1,5} {1,5} {2,3} {2,3} {2,3} {1,5}
항목집합수	0	2	0	4	1	5	2	6
함수값	0	1	2	3	4	5	6	7

그림 7. 예제 데이터베이스의 2-항목집합에 대한 해시테이블.

3.2 빈발 항목집합의 탐색

빈발 항목집합을 찾기 위해서 본 알고리듬은 공통항목트리를 검색하여 후보 항목집합의 지지수를 계산한다. 공통항목트리에서 부모 노드는 자식 노드를 포함하는 항목집합이다. 따라서 후보 항목집합의 지지수는 자신의 지지수 뿐만 아니라 후보 항목집합을 포함하는 모든 상위 항목집합의 지지수를 더한 값이 된다. 그러므로 후보 항목집합의 지지수를 계산하기 위해서는 상위 항목집합을 먼저 구해야한다. <그림 8>은 후보 항목집합의 상위 항목집합을 찾고 항목집합의 지지수를 계산하는 함수이다. 트리 생성 과정(3.1절 참조)과 마찬가지로 결여 항목집합을 사용하여 트리를 검색한다.

```
Count_Function(후보 항목집합){
    Support_Count = 후보 항목집합 노드에 저장된 트랜
                    잭션의 발생횟수
    A_Items = Cmax − 후보 항목집합
    Subset = A_Items 자신을 제외한 A_Items의 모든
             부분집합들
    for all subset ∈ Subset do begin
        subset의 결여항목의 사전적 크기 순서에 따라 공
        통항목트리를 검색하여 상위 항목집합 노드를 찾
        는다.
        Support_Count에 찾은 노드에 저장된 트랜잭션의
        발생횟수를 합산한다.
    end
    return Support_Count
}
```

그림 8. 후보 항목집합의 지지수 계산.

<그림 5>에 있는 항목집합 {2, 3}의 지지수를 계산하는 과정은 다음과 같다. 먼저 항목집합 {2, 3}에 대한 결여 항목집합 (*A_Items*)을 구하면 {1, 4, 5}가 된다. 그리고 결여 항목집합의 부분집합들은 ∅, {1}, {4}, {5}, {1, 4}, {1, 5}, {4, 5}이다. 그리고 {2, 3}의 상위 항목집합은 C_{max} 자신(트랜잭션의 발생횟수 = 0)과 C_{max}로부터 {1}이 결여된 {2, 3, 4, 5}(트랜잭션의 발생횟수 = 0), {4}가 결여된 {1, 2, 3, 5}(트랜잭션의 발생횟수 = 1), {5}가 결여된 {1, 2, 3, 4}(트랜잭션의 발생횟수=0), {1, 4}가 결여된 {2, 3, 5}(트랜잭션의 발생횟수=1), {1, 5}가 결여된 {2, 3, 4}(트랜잭션의 발생횟수=0), 그리고 {4, 5}가 결여된 {1, 2, 3}(트랜잭션의 발생횟수=0)이다. 따라서 항목집합 {2, 3}의 지지수는 {2, 3} 자신의 횟수(트랜잭션의 발생횟수 = 1)와 상위 항목집합들의 횟수를 더한 값이므로 1 + 0 + 0 + 1 + 0 + 1 + 0 + 0 = 3이 된다.

3.3 제안하는 알고리듬

본 알고리듬의 전체적인 과정은 Apriori 알고리듬에 기반을 두고 있다. 그러나 빈발항목을 찾는 과정에서 Apriori는 전체 데이터베이스를 여러 번 검색하지만, 제안한 방법은 데이터베이스 대신 공통항목트리를 검색한다(3.1, 3.2절 참조). 본 알고리듬의 단계를 표시하면 다음과 같다.

단계 1. 데이터베이스를 검색하여 빈발 1-항목집합을 찾는다.
단계 2. 데이터베이스를 검색하여 공통항목트리와 해시 테

이블을 만든다. //(3.1절 참조)

단계 3. 해시 테이블에서 항목집합수가 최소 지지수를 만족하는 항목집합들을 C_2에 등록한다.

단계 4. 공통항목트리를 검색하여 C2의 지지수를 계산한다. //(3.2절 참조)

단계 5. C_2 중에서 최소 지지도를 만족하는 항목집합들을 L_2에 등록한다. k=3으로 설정한다.

단계 6. apriori-gen 함수에 의해 C_k를 생성한다(Agrawl and Srikant, 1994).

단계 7. C_k가 ∅ 이면 종결하고 아니면 단계 8로 간다.

단계 8. 공통항목트리를 검색하여 C_k의 지지수를 계산한다.

단계 9. C_k 중에서 최소 지지도를 만족하는 항목집합들을 L_k에 등록한다.
$k = k + 1$ 로 설정하고 단계 6으로 간다.

<표 1> 예제 데이터베이스에 대해서 제안하는 알고리듬을 적용하면 다음과 같다. 먼저 단계 1, 단계 2에 의해 빈발 1-항목집합을 찾은 후 공통항목트리와 해시 테이블을 완성하면 <그림 5>와 <그림 7>이 된다. <그림 7>의 해시 테이블에서 항목집합 {1, 2}는 항목집합수가 최소 지지수를 만족시키지 못하므로 C_2에서 제외된다. 따라서 <그림 9>의 C_2 테이블에 항목집합 {1, 2}가 존재하지 않는다. <그림 9>에서 C_2 테이블의 둘째 번 필드값은 공통항목트리를 검색하여 얻은 값이다. C_2 중에서 최소 지지도(minsup = 40 %, 즉 최소 지지수 = 2)를 만족시키는 빈발 2-항목집합을 찾으면 <그림 9>의 L_2 가 된다.

<그림 10>은 apriori-gen 함수(Agrawl and Srikant, 1994)에 의해 C_3을 구하고 공통항목트리를 검색하여 빈발 3-항목집합을 찾은 결과이다.

공통항목 트리검색 →

Itemset	지지수
{1, 3}	3
{1, 4}	2
{1, 5}	3
{2, 3}	3
{2, 5}	2
{3, 4}	2
{3, 5}	4
{4, 5}	2

C_2

Itemset	지지수
{1, 3}	3
{1, 4}	2
{1, 5}	3
{2, 3}	3
{2, 5}	2
{3, 4}	2
{3, 5}	4
{4, 5}	2

L_2

그림 9. 빈발 2-항목집합.

공통항목 트리검색 →

Itemset	지지수
{1, 3, 4}	2
{1, 3, 5}	3
{1, 4, 5}	2
{2, 3, 5}	2
{3, 4, 5}	2

C_3

Itemset	지지수
{1, 3, 4}	2
{1, 3, 5}	3
{1, 4, 5}	2
{2, 3, 5}	2
{3, 4, 5}	2

L_3

그림 10. 빈발 3-항목집합.

공통항목 트리검색 →

Itemset	지지수
{1, 3, 4, 5}	2

C_4

Itemset	지지수
{1, 3, 4, 5}	2

L_4

그림 11. 빈발 4-항목집합.

<그림 11>은 빈발 4-항목집합을 찾은 결과이다. 빈발 4-항목집합이 하나이므로 후보 5-항목집합을 만들 수 없다. 따라서 알고리듬을 종결한다.

3.4 공통항목트리 알고리듬의 효율성

공통항목트리 생성 과정에서 볼 수 있듯이, 본 알고리듬은 전체 데이터베이스를 두 번 검색한다. 한 번은 빈발 1-항목집합을 찾기 위한 것이고, 두번째 검색은 공통항목트리를 만드는 과정에 필요하다. 생성된 공통항목트리는 데이터베이스에 존재하는 빈발항목들의 집합으로 구성되기 때문에 데이터베이스에 존재하는 연관규칙을 찾기 위한 정보를 모두 저장하고 있다. 따라서 데이터베이스 대신에 공통항목트리를 검색해도 찾은 빈발 항목집합은 동일하다.

공통항목트리의 크기는 데이터베이스의 크기보다 작다. 왜냐하면 비빈발항목에 대한 데이터가 제외되었을 뿐만 아니라 (<그림 1> 참조) 최대 공통항목집합이 여러 트랜잭션에 동시에 나타나는 경우에 이들은 한 노드의 지지수로 압축되어 표시되기 때문이다. <표 2>는 C_{max} = {1, 2, 3, 4, 5}일 때, 세 개의 트랜잭션이 같은 최대 공통항목집합 {2, 3, 5}를 갖게 됨을 보여준다. 따라서 공통항목트리에서는 {2, 3, 5} 노드의 횟수를 3으로 설정함으로 세 개의 트랜잭션을 압축하여 한 노드로 표시할 수 있다.

표 2. 같은 최대 공통항목집합을 갖는 트랜잭션들

TID	Items	MCI
600	2 3 5 6	2 3 5
700	2 3 5 7 8	2 3 5
800	2 3 5 8	2 3 5

빈발 k-항목집합을 찾기 위해서 Apriori 기반의 알고리듬은 데이터베이스를 k번 검색한다. 그러나 본 방법은 데이터베이스를 두 번 검색하고 데이터베이스보다 크기가 작은 공통항목트리를 $(k-1)$ 번 검색한다. 따라서 제안하는 알고리듬은 기존의 Apriori 방법에 비해 빈발 항목집합 탐색시간이 단축된다(4절 참조).

4. 실험 결과 및 분석

본 장에서는 제안하는 공통항목트리 알고리듬을 기존의

표 3. 매개변수

$\|T\|$	트랜잭션의 평균 길이
$\|I\|$	항목 수
$\|P\|$	최대 빈발 항목집합의 평균 길이
$\|D\|$	트랜잭션 수
S	최소지지도(%)

Apriori 알고리듬(Agrawl and Srikant, 1994)과 Max-Miner 알고리듬(Bayardo Jr. and Agrawal, 1998)과 비교한다. 실험에 사용된 컴퓨터는 Windows 98 환경에서 CPU는 펜티엄 II 400MHz이며 메모리는 64MByte이다. 알고리듬은 Visual C++ 5.0으로 구현하였다. 알고리듬의 수행도를 실험하기 위해 IBM의 Quest group이 구현한 실험 데이터 생성 프로그램을 이용했다(Agrawl and Srikant, 1994). 실험 데이터를 생성하기 위해 이용된 매개변수는 <표 3>에 나타나 있다. 데이터베이스 T10.I20.P10.S10.D100k는 트랜잭션의 평균 길이가 10, 항목 수가 20, 최대 빈발 항목집합의 평균 길이가 10, 최소지지도가 10 %, 그리고 트랜잭션 수가 100000 개라는 것을 의미한다.

본 연구에서는 세 가지를 비교 실험하였다. 첫째는 최대 빈발 항목집합의 평균 길이의 증가에 따른 실행시간 변화이다. 둘째 번은 트랜잭션의 수의 증가에 따른 실행시간의 변화이다. 마지막으로 최소지지도의 감소에 따른 실행시간의 변화이다

<그림 12>와 <그림 13>은 항목 수가 20, 트랜잭션 수가 100000, 그리고 최소 지지도는 각각 10 %와 20 %인 가공 데이터에서 최대 빈발 항목집합의 평균 길이의 변화에 따른 실험의 결과이다. 최대 빈발 항목집합의 평균 길이가 증가할 때, Apriori 알고리듬의 실행시간은 지수적으로 증가함을 알 수 있다. 이에 비해 공통항목트리 알고리듬과 Max-Miner의 실행시간의 변화는 작다.

<그림 14>와 <그림 15>는 트랜잭션의 평균 길이가 10, 항목 수가 20, 최대 빈발 항목집합의 평균 길이가 10, 그리고 최소 지지도는 각각 10 %와 20 %인 가공 데이터에서 트랜잭션 수의 변화에 따른 실험의 결과이다. 트랜잭션 수가 증가하더라도 공통항목트리 알고리듬은 데이터베이스를 두 번만 검색하고 트리를 탐색하기 때문에 Apriori나 Max-Miner보다 실행시간의 변화가 완만함을 볼 수 있다.

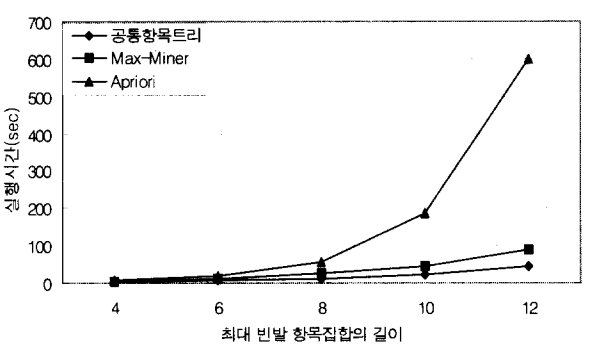

그림 12. I20.S10.D100k.

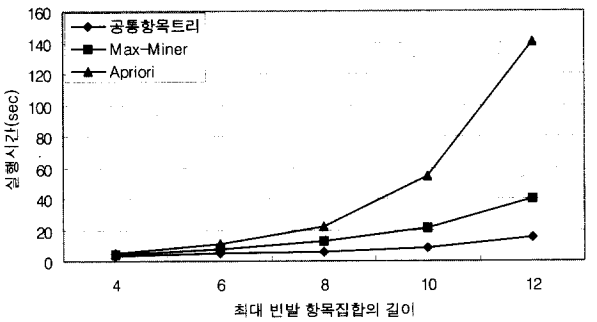

그림 13. I20.S20.D100k.

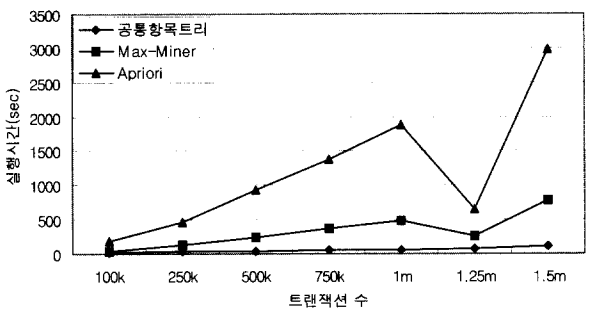

그림 14. T10.I20.P10.S10.

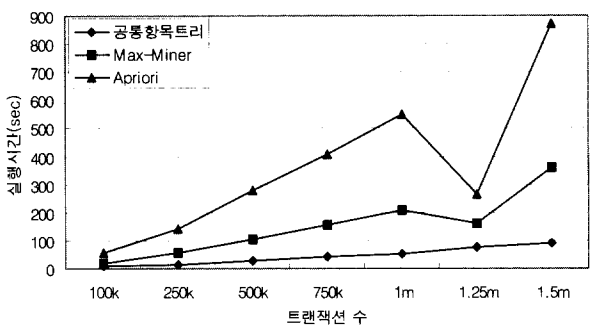

그림 15. T10.I20.P10.S20.

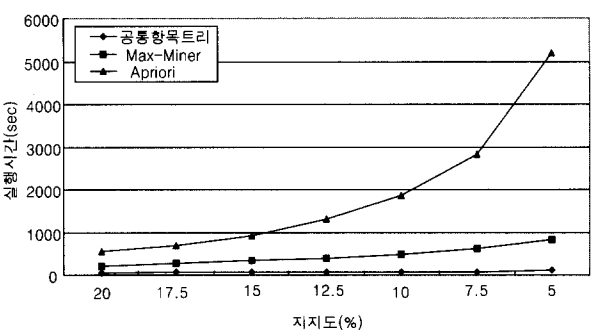

그림 16. T10.I20.P10.D1m.

<그림 16>과 <그림 17>은 트랜잭션의 평균 길이가 10, 최대 빈발 항목집합의 평균 길이가 10, 트랜잭션 수가 1000000, 그리고 항목 수가 각각 20, 30인 가공 데이터에서 최소 지지도의

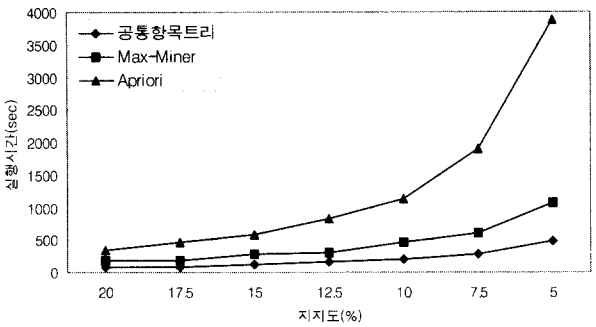

그림 17. T10.I30.P10.D1m.

변화에 따른 실험의 결과이다. Apriori의 실행시간은 최소 지지도가 낮아짐에 따라 실행시간이 지수적으로 증가함을 알 수 있다. 반면에, 공통항목트리는 비교한 두 알고리듬보다 최소 지지도의 변화에 따른 실행시간의 변화가 가장 작음을 알 수 있다.

5. 결 론

연관규칙을 찾는 알고리듬에서 데이터베이스의 검색은 계산 시간 중에서 가장 많은 비중을 차지하는 요소이다. 찾고자하는 항목집합의 크기가 k라면 기존의 Apriori 기반의 알고리듬들은 데이터베이스를 k 번 검색하므로 빈발 항목집합의 길이가 길면 길수록 많은 시간이 소요된다. 이러한 단점을 보완하기 위해 본 연구에서는 공통항목트리를 이용하여 검색시간을 단축시키는 알고리듬을 제시하였다.

제안한 알고리듬에서는 데이터베이스를 검색하여 트랜잭션의 요약 데이터인 공통항목트리를 생성하고 데이터베이스 대신 이 트리를 검색하여 빈발항목을 찾는다. 본 알고리듬은 데이터베이스를 두 번만 검색하기 때문에 빈발 항목집합의 길이에 관계없이 선형적인 수행속도를 갖는다. 이는 공통항목트리 알고리듬이 빈발 항목집합의 길이가 긴 경우에 효율적이며, 트랜잭션의 수가 큰 데이터베이스에서 빈발 항목집합을 찾는 데 효율적이라는 것을 의미한다.

참고문헌

Agrawal, R. and Srikant, R. (1994), Fast Algorithm for Mining Association Rules in Large Databases, *In Proc. of the 20th Int'l Conference on Very Large Data Bases*, Santiago de Chile, Chile, 487-499.

Agrawal, R., Imielinski, T. and Swami, A. (1993), Mining Association Rules between Sets of Items in Large Databases, *In Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Washington, D. C., 207-216.

Bayardo, Jr. R. J. and Agrawal, R. (1998), Efficiently Mining Long Patterns from Databases, *In Proc. of the ACM SIGMOD Int'l Conference on management of Data*, Seattle, Washington, 85-93.

Berry, M. and Linoff, G. (1997), Data Mining Techniques for Marketing, Sales, and Customer Support, John Wiley & Sons, Inc.

Brin, S., Motwani, R., Ullman, J. and Tsur, S. (1997), Dynamic Itemset Counting and Implication Rules for Market Basket Data, *In Proc. of the ACM SIGMOD Int'l Conference on Management of* Data, Tucsan, Arizona, 255-264.

Cheung, D., Han, J., Ng, V. and Wong, C. (1996), Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique, *In Proc. of the 12th Int'l Conference on Data Engineering*, New Orleans, Louisiana, 106-114.

Han, J., Gong, W. and Yin, Y. (1998), Mining Segment-Wise Periodic Patterns in Item-Related Databases, *In Proc. of the 4th Int'l Conference on Knowledge Discovery and Data Mining*, New York City, New York, 214-218.

Özden, B., Ramaswamy, S. and Silberschatz, A. (1998), Cyclic Association Rules, *In Proc. of the 14th Int'l Conference on Data Engineering*, Orlando, Florida, 412-421.

Park, J., Chen, M. and Yu, P. (1995), An Effective Hash-Based Algorithm for Mining Association Rules, *In Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, San Jose, California, 175-186.

Srikant, R. and Agrawal, R. (1995), Mining Gernalized Association Rules, *In Proc. of the 21st Int'l Conference on Very Large Data Bases*, Zurich, Switzerland, 407-419.

Srikant, R. and Agrawal, R. (1996), Mining Quantitative Association Rules in Large Relational Tables, *In Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Montreal, Canada, 1-12.

Srikant, R. and Agrawal, R. (1996), Mining Sequential Patterns: Generalizations and Performance Improvements, *In 5th Int'l Conference on Extending Database Technology*, Avignon, France, 3-17.

Srikant, R., Vu, Q. and Agrawal, R. (1997), Mining Association with Item Constraints, *In Proc. of the 3rd Int'l Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, 67-73.

Tang, J. (1998), Using Incremental Pruning to Increase the Efficiency of Dynamic Itemset Counting for Mining Association Rules, *In Proc. of the ACM CIKM Int'l Conference on Information and Knowledge Management*, Bethesda, Maryland, 273-280.

Zaki, M. (1998), Efficient Enumeration of Frequent Sequences, *In Proc. of the ACM CIKM Int'l Conference on Information and Knowledge Management*, Bethesda, Maryland, 68-75.