매트릭스를 이용한 빈발 항목집합 생성 알고리즘

채덕진° 황부현
전남대학교 전산학과
{djchai°, bhhwang}@sunny.chonnam.ac.kr

Generation Algorithm of Frequent Itemsets using Matrix

Duckjin Chai° Buhyun Hwang
Dept. of Computer Science, Chonnam National University

요 약

대용량의 데이터베이스에서 최소지지도를 만족하는 항목들의 집합을 빈발 항목집합이라고 한다. 이전에 연구된 대부분의 빈발 항목집합 생성 알고리즘들은 후보 항목집합들을 생성하고 이들 중에서 조건을 만족하는 빈발 항목집합들을 생성하는 과정을 수행하였다. 그러나 이러한 알고리즘들은 모든 k(k≥1)-빈발 항목집합들을 생성하기까지 k를 하나씩 증가하면서 반복적으로 수행되기 때문에 많은 컴퓨팅 시간을 필요로 한다. 본 논문에서는 후보 항목집합들을 생성하지 않고 빈발 항목집합들을 생성할 수 있는 DFG 알고리즘을 제안한다. 각각의 k-빈발 항목집합들에는 데이터베이스의 모든 정보들이 포함되어 있고 하나의 빈발 항목집합은 한 트랜잭션에 존재한다. 본 논문에서는 이러한 성질을 이용하여 먼저 2-빈발 항목집합들을 생성한다. 그리고 2-빈발 항목집합들에 존재하는 한 항목과 나머지 항목들에 대한 매트릭스를 구성하여 최소지지도를 만족하는 빈발 항목집합들을 생성하게 된다. 제안하는 알고리즘은 불필요한 후보 항목집합들을 생성하지 않고 한 번의 데이터베이스 스캔만으로 빈발 항목집합들을 생성할 수 있다.

1. 서 론

연관 규칙 탐사는 데이터베이스를 이루는 트랜잭션들에서 최소지지도(minimum support) 보다 많이 나타나는 빈발항목집합(Large itemset or Frequent itemset)들을 찾는 문제로 정의할 수 있다. 지금까지 빈발항목집합을 찾기 위한 다양한 연관 규칙 탐사 알고리즘들이 제시되었다[1,2,3,4,5,6]. [1]에서 제안된 Apriori 와 [2]에서 제안된 DHP 알고리즘 같은 기존에 제안된 대부분의 알고리즘들은 휴리스틱(heuristic) 방법을 사용했다. 즉, k-후보 항목집합이 존재한다고 할 때, k(k≥1)-후보 항목집합이 최소지지도를 만족하면 k-빈발 항목집합이 된다. 그리고 더 이상의 후보 항목집합들이 생성되지 않을 때까지 이 과정은 반복 수행된다. 기존에 연구된 알고리즘들은 대부분 이러한 휴리스틱 방법을 사용하여 후보 항목집합들을 생성하고 데이터베이스의 크기를 줄이는 데 연구가 집중되었다[5]. 그러나 Apriori와 DHP 같은 알고리즘들은 많은 빈발 항목집합들을 가질 때, 그리고 낮은 최소지지도를 가질 때는 대단히 많은 컴퓨팅 시간을 필요하게 된다.

[5]에서는 후보 항목집합들을 생성하지 않고 빈발 항목집합들을 생성할 수 있는 FP-tree 알고리즘을 제안하였다. FP-tree 알고리즘은 1-빈발 항목집합들을 생성하고 데이터베이스를 1-빈발 항목집합들로만 재구성한다. 이때, 각 항목들의 지지도에 따라 내림차순으로 데이터베이스를 정렬시킨다. 그리고 재구성된 데이터베이스를 스캔함으로써 FP-tree를 생성한다. 자세한 알고리즘 수행 방법은 2장에서 설명할 것이다.

FP-tree 알고리즘은 후보 항목집합들을 생성하지 않고 빈발 항목집합들을 생성할 수 있는 효율적인 알고리즘이다. 그러나 1-빈발 항목집합들의 개수가 많고 트랜잭션을 이루는 항목들의 수가 많으면 그에 따른 컴퓨팅 시간은 증가하게 된다. 왜냐하면, 데이터베이스를 1-빈발 항목집합들의 지지도에 따라 정렬해야하는 컴퓨팅 비용을 수반하기 때문이다.

본 논문에서는 1-빈발 항목집합으로부터 최종 k-빈발 항목집합들을 생성하기까지 기존의 순차적인 방법을 따르지 않고 빈발 항목집합들을 생성할 수 있는 DFG(Direct Frequent itemsets Generation) 알고리즘을 제안한다. 빈발 항목집합들은 다음과 같은 특성을 가진다[2].

- (k-1)-large itemsets ⊂ k-large itemsets
- 임의의 k-large itemset ⊆ one transaction
- Information of all Frequent itemsets ⊆ 임의의 k-large itemsets

첫 번째 특성은 빈발 항목집합들의 부분집합들 또한 최소지지도를 만족한다는 것이고, 두 번째 특성은 임의의 한 빈발 항목집합은 한 트랜잭션의 부분집합이라는 것이다. 다시 말하면 빈발 항목집합은 하나의 트랜잭션에 반드시 존재한다는 것이다. 세 번째 특성은 임의의 k-빈발 항목집합들이 있을 때, 데이터베이스에서 생성하고자하는 모든 빈발 항목집합들의 정보가 k-빈발 항목집합들에 포함되어 있다는 것이다. 위의 세 가지 성질은 다음의 예에서 찾아볼 수 있다. 만약, 3-빈발 항목집합이 {A, B, C}라고 가정하면 이 집합의 부분집합 {A}, {B}, {C}, {A, B}, {A, C}, {B, C}는 모두 빈발 항목집합이 된다. 그리고 {A, B, C}라는 빈발 항목집합이 생성된다면 전체 트랜잭션들 중에서 최소지지도 이상의 트랜잭션들은 반드시 {A, B, C}를 포함하고 있어야 한다. 또, {A, B, C}라고 하는 빈발 항목집합들의

2-빈발 항목집합들과 1-빈발 항목집합들은 모두 A, B, C라고 하는 항목들을 포함하고 있다. 즉, 임의의 k-빈발 항목집합들이라도 데이터베이스에서 생성해야할 빈발 항목집합들에 대한 모든 정보들을 가지고 있다는 것이다.

본 논문에서는 이러한 성질을 이용하여 2-빈발 항목집합들을 생성한다. 그리고 2-빈발 항목집합들을 이루고 있는 각 빈발 항목들에 대해서 한 항목과 나머지 항목들에 대한 매트릭스를 생성하고 그 매트릭스에서 빈발 항목집합들을 생성하게 된다.

본 논문의 이후의 구성은 다음과 같다. 2장 관련 연구에서는 연관규칙 탐사에 대한 개괄적인 소개와 [5]에서 제안된 FP-tree 알고리즘에 대해서 살펴보고 3장에서는 제안하는 알고리즘에 대해 설명한다. 그리고 4장에서는 제안하는 알고리즘과 FP-tree 알고리즘의 성능 상의 차이에 대해 분석하고 5장에서는 본 논문에 대한 결론 및 향후 연구 방향에 대해 논한다.

2. 관련 연구

2.1 연관규칙 탐사

연관 규칙이란 특정 사건 집합의 발생이 다른 사건의 발생을 암시하는 경향을 표현하는 규칙으로 다음과 같이 정의할 수 있다. $I=\{a_1, a_2,..., a_m\}$를 항목(item)들의 집합이라 정의하고 트랜잭션들로 구성된 데이터베이스를 $<T_1, T_2,..., T_n>$라 하자. T_i 가 하나의 트랜잭션일 때, $T_i \subseteq I$ 이다.

$X(X \subseteq T_i)$와 $Y(Y \subseteq T_i)$의 항목 집합에 대한 연관 규칙은 $X \rightarrow Y$로 표현되며, X를 규칙의 조건부(antecedent), Y를 결과부(consequent)라 한다. 이때 $X \cap Y = \emptyset$이다[1].

2.2 FP-tree 생성 알고리즘[5]

FP-tree 알고리즘은 빈발 패턴들에 대한 정보를 가지고 있는 빈발 패턴 트리(frequent pattern tree : FP-tree)를 생성하고 최소지지도를 만족하는 빈발 항목집합들을 생성한다. 또한, 데이터베이스의 반복되는 스캔을 피하고 후보 항목집합들의 생성을 회피함으로써 성능상의 많은 진전을 가져왔다.

빈발 패턴 트리는 다음과 같이 정의된다.

첫째, 빈발 패턴 트리는 "null"로 표기된 하나의 루트 노드, item prefix subtrees, 그리고 header table로 구성된다. 둘째, item prefix subtree에서 각 노드는 표 1과 같이 세 부분으로 구성된다.

표 1. item prefix subtree의 구성요소

Factor	comment
item-name	빈발 항목의 이름
count	빈발 항목의 지지도
node-link	같은 이름을 가진 빈발 항목에 대한 포인터

셋째, header table은 item-name과 head of node-link로 구성된다. item-name은 1-빈발 항목들이고 head of node-link는 1-빈발 항목들이 트리에서 처음 발생한 노드를 가리킨다.

FP-tree 알고리즘은 먼저, 데이터베이스를 스캔하여 1-빈발 항목들과 지지도를 구한다. 그리고 1-빈발 항목들을 지지도가 낮은 순으로 정렬하여 목록을 작성한다. 다음은 루트노드인 "null" 노드를 생성하고 각 트랜잭션을 스캔한다. 각 트랜잭션에 대해서 1-빈발 항목들의 목록에 따라 정렬하고 정렬된 순서대로 트리를 구성해 나간다. 이때, 지지도가 높은 것이 부모 노드가 되고 지지도가 낮은 순으로 자식 노드가 된다. 만약 트랜잭션의 항목과 트리의 노드가 같으면 카운트를 하나 증가시키고 다르면 새로운 노드를 생성한다. 그리고 카운트를 1로 초기화시킨다. 이러한 과정은 모든 트랜잭션에 대해 반복적으로 수행된다.

표 2와 그림 1은 알고리즘의 수행을 보여주는 하나의 예이다. 표 2는 데이터베이스를 나타내고 그림 1은 데이터베이스를 가지고 실제 알고리즘을 수행한 결과를 나타낸다.

표 2. 트랜잭션 데이터베이스의 예

TID	Items Bought	(Ordered) Frequent Items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

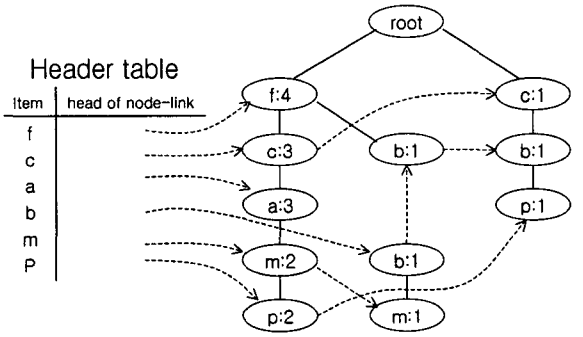

그림 1. The FP-tree in Example

3. DFG(Direct Frequent itemset Generation) 알고리즘

연관규칙 탐사에서 빈발 항목집합을 발견하는데 가장 효율적인 방법은 트랜잭션에 감추어져 있는 빈발 항목집합들을 한 번의 스캔으로 찾아내는 것이라 할 수 있다. 그러나 대용량의 데이터베이스에서 빈발 항목집합들을 눈으로 들여다보듯이 찾아낼 수 없기 때문에 기존의 알고리즘들은 1-빈발 항목집합들로부터 순차적으로 빈발 항목집합들을 찾아야만 했다.

각 k-빈발 항목집합들은 데이터베이스에서 생성하고자 하는 모든 정보들을 가지고 있다. 예를 들어, 3-빈발 항목집합이 {A, B, C}라고 가정하면 2-빈발 항목집합들은 {A, B}, {A, C}, {B, C}가 되고 1-빈발 항목집합들은 {A}, {B}, {C}가 된다. 예에서 볼 수 있듯이 1, 2 그리고 3-빈발 항목집합들은 모두 중복된 정보들을 가지고 있음을 알 수 있다. 본 논문에서는 이러한 성질을 이용하여 먼저 2-빈발 항목집합들을 생성한다. 그리고 2-빈발 항목집합들의 부분집합인 각 항목들에 대해 매트릭스를 형성하여 최소지지도를 만족하는 빈발 항목집합들을 생성한다.

3.1 2-빈발 항목집합 생성

DFG 알고리즘은 2-빈발 항목집합들을 생성하기 위해

해쉬 기법을 사용하여 데이터베이스를 스캔한다. 별도의 비교연산이나 후보 항목집합들을 생성하지 않기 때문에 적은 컴퓨팅 비용으로 2-빈발 항목집합들을 생성할 수 있다.

모든 빈발 항목집합들을 해쉬 기법을 사용하여 생성한다면 알고리즘 수행 시간은 기존의 알고리즘보다 훨씬 적은 시간에 생성할 수 있을 것이다. 왜냐하면 데이터베이스에서 모든 k(k>1)-부분집합들을 버켓(bucket)에 저장하고 단지 이들의 지지도만 최소지지도와 비교하여 빈발 항목집합들을 찾아내면 되기 때문에 빠른 시간 내에 모든 빈발 항목집합들을 찾을 수 있다. 그러나 후보 항목집합들을 생성하는 것은 $\binom{n}{k}$ 연산을 수행하게된다. n은 데이터베이스를 이루는 항목들의 개수이고 k는 후보 항목집합들을 이루는 집합의 원소의 개수이다. 그러므로 k가 커지면 생성되는 버켓의 개수는 지수적으로 증가하게 된다. 이는 시스템 메모리의 한계를 벗어나기 때문에 사용하기 어려운 방법이다. 앞에서 설명한 것처럼 각 k-빈발 항목집합들은 데이터베이스에서 생성할 수 있는 모든 빈발 항목집합들에 대한 정보들을 가지고 있다. 그러므로 본 논문에서는 메모리 오버헤드가 상대적으로 적은 2-빈발 항목집합을 생성한다.

수행 과정은 데이터베이스를 스캔하여 모든 2-부분집합들에 대한 버켓을 생성한다. 그리고 그 버켓에는 지지도가 저장된다. 그림 2는 2-빈발 항목집합을 생성하기 위해 해쉬 함수를 사용한 하나의 예이다.

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

2-부분집합(버켓주소), 지지도					
AC(13)	2	BC(23)	2	CD(34)	1
AD(14)	1	BE(25)	3	CE(35)	1
AB(12)	1				
AE(15)	1				

h(x, y) = (order of x)*10 + (order of y)
2-빈발 항목 집합 : AC, BC, BE

그림 2. 2-빈발 항목집합 생성의 예

3.2 빈발 항목집합 생성

항목집합 X가 한 트랜잭션에 반드시 포함되고 집합 X를 포함한 트랜잭션들이 최소 지지도 이상 존재할 때 집합 X를 빈발 항목집합이라 한다. 이러한 성질 때문에 2-빈발 항목집합들에는 모든 k-빈발 항목집합에 대한 정보가 남아있게 된다. 예를 들면, {B, C, E}는 TID 200과 300인 트랜잭션에 존재한다. 두 개의 트랜잭션에서 {B, C, E}로 생성할 수 있는 2-부분집합의 버켓에는 카운트가 2로 누적되고 단지 TID 300의 A 항목과 나머지 {B, C, E}에 대한 버켓만 생성될 것이다.

알고리즘에서의 매트릭스는 2-빈발 항목집합에 포함된 한 항목과 나머지 항목들간의 관계를 나타낸다. 그림 3의 예에서 2-빈발 항목집합에 포함된 1-항목들은 {A, B, C, E}가 된다. 그림 3은 1-항목들에서 생성될 수 있는 매트릭스들과 지지도를 만족하는 빈발 항목집합을 생성하는 예를 보여준다.

4. FP-tree 알고리즘과의 분석

DFG 알고리즘은 데이터베이스를 한 번 스캔하는데 비해 FP-tree 알고리즘은 데이터베이스를 두 번 스캔한다. 그리고 DFG 알고리즘은 미리 생성된 해쉬 테이블을 사용하여

	B	C	E
A	1	2	1

빈발 항목집합	지지도
{A, C}	2

	C	E
B	2	3

빈발 항목집합	지지도
{B, C, E}	2

	E
C	1

그림 3. 매트릭스와 빈발 항목집합 생성의 예

매트릭스를 생성하므로 별도의 연산 과정을 거치지 않는 반면 FP-tree 알고리즘은 1-빈발 항목집합들을 지지도에 의해 내림차순 정렬한 방법으로 데이터베이스를 재구성한다. 또한, 재구성된 데이터베이스를 스캔하여 트리를 생성하므로 제안하는 알고리즘에 비해 연산의 횟수가 많아진다.

5. 결론 및 향후 연구 방향

본 논문에서는 방대한 데이터베이스에 있는 항목들 중에서 빈발 항목집합을 효율적으로 생성하기 위한 DFG 알고리즘을 제안하였다. 임의의 k-빈발 항목집합들은 모든 빈발 항목집합들에 대한 정보들을 저장하고 있다는 성질을 이용하여 한 번의 데이터베이스 스캔으로 해쉬 테이블을 생성하고 압축된 정보인 2-빈발 항목집합들을 생성한다. 그리고 해쉬 테이블을 재 사용하여 원하는 빈발 항목집합들을 찾아낼 수 있었다.

DFG 알고리즘은 불필요한 연산을 수행하지 않음으로써 빠른 시간 내에 원하는 결과를 얻을 수 있다. 그러나 최소지지도의 임계값에 따라 불필요한 항목들이 빈발 항목집합에 포함될 수 있다. 이는 2-빈발 항목집합들만을 사용하므로 생성하고자 원하는 빈발 항목집합들에 대한 모든 정보들을 추출할 수 있지만 여기에 의미 없는 항목들이 포함될 수 있다. 향후에는 이러한 의미 없는 항목들을 전지할 수 있는 방법에 대한 연구가 진행되어야 하겠다.

6. 참고 문헌

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining Association Rules", In Proceedings of the 20th VLDB Conference, pp. 487-499, 1994.

[2] J.S. Park, M.-S. Chen, and P.S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", In Proceedings of ACM SIGMOD, pp. 175-186, 1995.

[3] B. Lent, A. Swami, and J. Widom, "Clustering association rules", In ICDE, pp. 220-231, 1997.

[4] R. Ng, L. V. S. Lakshmanan, J. Han, and A.Pang, "Exploratory mining and pruning optimizations of constrained associations rules", In SIGMOD, pp. 13-24, 1998.

[5] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", In Proceedings of ACM-SIGMOD, pp. 1-12, 2000.

[6] G. Grahne, L. Lakshmanan, and X. Wang, "Efficient mining of constrained correlated sets", In ICDE, 2000.