DOI: 10.3745/KIPSTD.2008.15-D.4.475

연관 규칙 탐사 응용을 위한 한 번 읽기에 의한 최대 크기 빈발항목 추정기법

한 갑 수†

요 약

최근에는 데이터를 획득 및 처리하는 방법의 향상으로 인하여 연속적이고 실시간으로 발생되는 데이터를 처리하는 응용이 증가하고 있다. 그러한 응용에서 연관규칙을 추출하기 위해서는 새로운 방식을 사용하여 빈발항목집합을 찾아내야 한다. 기존의 빈발항목을 발견하는 방식에서는 전체 데이터베이스를 반복적으로 읽으면서 처리해야 한다. 그러나 실시간이고 연속적으로 발생하는 데이터를 처리하는 응용에서는 반복적으로 여러 번 데이터를 읽을 수 없기 때문에 일정 구간의 데이터를 한 번만 읽고 처리해야 한다. 따라서 본 논문에서는 입력되는 데이터 구간을 한 번만 읽고 최대 빈발항목 집합의 크기와 해당 빈발항목을 추정함으로써 필요한 연관규칙탐사를 가능하게 하는 빈발항목 추정 기법을 제안한다.

키워드 : 빈발항목, 연관규칙탐사, 데이터구간

Approximation of Frequent Itemsets with Maximum Size by One-scan for Association Rule Mining Application

Gab-Soo Han†

ABSTRACT

Nowadays, lots of data mining applications based on continuous and online real time are increasing by the rapid growth of the data processing technique. In order to do association rule mining in that application, we have to use new techniques to find the frequent itemsets. Most of the existing techniques to find the frequent itemsets should scan the total database repeatedly. But in the application based on the continuous and online real time, it is impossible to scan the total database repeatedly. We have to find the frequent itemsets with only one scan of the data interval for that kind of application. So in this paper we propose an approximation technique which finds the maximum size of the frequent itemsets and items included in the maximum size of the frequent itemsets for the processing of association rule mining.

Key Words : Frequent Itemsets, Association Rule Mining, Data Interval

1. 서 론

최근에는 각종 센서에서 분석된 실시간 데이터의 입력이나 통신 전송의 감시 또는 인터넷 사용 로그 처리와 분석 등과 같은 응용에서 연관 규칙 탐사를 처리하기 위해서는 입력되는 데이터를 기존의 방식과 같이 데이터베이스에 저장한 후 이것을 최대 빈발항목의 크기만큼 반복적으로 전체 데이터베이스를 읽어서 처리하는 방식으로는 처리가 불가능하다. 연속적으로 입력되는 데이터로부터 빈발 항목을 찾아서 연관 규칙을 발견하기 위해서는 입력되는 데이터 구간을 한 번만 읽고 빈발항목 집합을 발견해야 한다. 이를 위해서는 기존 응용 시스템 환경에서 주로 사용했던 에러가 전혀 없는 연관 규칙 탐사를 위한 정확한 처리(Exact processing) 보다는 새로운 접근 방법으로서 약간의 에러를 포함하는 추정적 처리(Approximate processing)가 필요하며 이것이 현실적으로도 가능한 방법이다. 따라서 본 논문에서는 데이터 스트림의 일정한 데이터의 구간을 입력되는 동시에 한번의 읽기를 통해서 필요한 빈발항목의 크기와 항목을 추정하여 생성하고 그로부터 연관규칙탐사를 가능하게 하는 새로운 방식을 제안하고자 한다.

본 논문의 구성은 다음과 같다 먼저 2장의 관련 연구에서는 기존의 대표적인 처리 방식인 Apriori[2] 알고리즘과 교집합 처리 방식과 병렬 처리 방식 등에 대한 문제점과 최근의 데이터 스트림 관련 논문을 통한 장단점을 살펴보고, 3

※ 이 논문은 두원공과대학에서 지원한 연구비로 나온 결과임.
† 정 회 원 : 두원공과대학 컴퓨터정보과 교수
논문접수 : 2007년 8월 7일
수 정 일 : 1차 2008년 5월 6일, 2차 2008년 6월 3일
심사완료 : 2008년 6월 4일

장에서는 본 논문에서 새롭게 제안하는 것으로서 문제의 정의와 추정 기법의 주요 내용을 서술한다. 4장에서는 실험에 관련되는 내용과 비교 결과를 서술하며 마지막으로 5장에서 결론을 서술한다.

2. 관련 연구

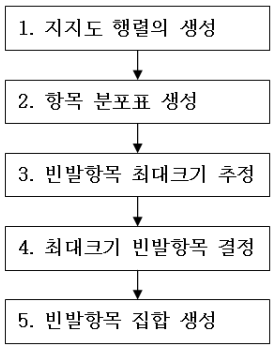
기존의 연관 규칙 처리 방식은 [1]에서 최초로 출현한 이후에 대표적인 단계적 처리(level-wise approach) 방식인 Apriori[2] 알고리즘으로 발전하였다. 단계적 처리를 적용하여 빈발항목을 발견하기 위해서는 먼저 관련 데이터를 기존의 데이터베이스에 저장한 후에 전체를 읽으면서 최초의 빈발항목 집합을 발견하여 그로부터 후보 빈발항목 집합을 구하고 다시 데이터베이스를 읽으면서 다음 크기의 빈발항목 집합을 구하는 방식을 반복적으로 k번 반복 읽기를 해야 크기 k의 최대 빈발항목 집합을 구하는 방식이다. 교집합방식[3]에서는 기본적 크기의 빈발항목 집합에 대한 트랜잭션들의 비트 벡터 리스트를 생성하고 저장한 후에 크기 k의 빈발항목 집합을 발견하려면 기본적 크기의 비트 벡터 리스트를 반복적으로 교집합을 하거나 크기 (k-1)의 빈발항목 집합 들의 트랜잭션 비트 벡터 리스트를 생성하고 저장하여 이것을 추가되는 항목의 비트 벡터 리스트와 반복적으로 교집합 하면서 구하게 된다. 병렬처리 방식[4, 5]은 주로 전체 데이터베이스를 여러 파티션으로 나누고 각각을 한 개의 프로세서에 할당하여 동시에 모든 파티션을 병렬로 처리하면서 증가된 크기 k의 빈발항목 집합을 생성하기 위해 각 파티션으로부터 생성된 크기 k-1의 빈발항목 집합을 통합한 후에 크기 k의 후보 집합을 생성하고 다시 그것을 각 프로세서에게 재분배하면 자기의 파티션을 읽어서 크기 k인 후보들의 지지도를 집계하고 빈발항목 집합을 선별하는 방식이다. 전술한 바와 같은 단계적이고 반복적인 처리방법은 실시간이고 연속적으로 입력되는 데이터 스트림의 특징[6]을 충족시키지 못한다. 그 이유는 반복적으로 읽고 처리하려면 연속적으로 입력되는 데이터가 누수 되는 문제가 발생되기 때문이다.

좀 더 발전된 방식으로서 [11]는 본 논문에서 사용하는 지지도 행렬과 유사한 방식의 지지도 행렬을 제안한 최초의 논문으로서 크기 2 항목 집합으로 구성된 지지도행렬을 먼저 구성한 후에 기본 지지도 행렬을 계속 확장하여 최대 크기까지의 후보 지지도 행렬로 확장하고 최종으로 빈발항목 집합을 결정하기 위해서는 2번째로 한번 더 전체 데이터베이스를 읽는 방법으로서 후보 지지도 행렬로 계속 확장할 때 최소지지도와 각 후보 항목 집합의 지지도만을 사용하기 때문에 탐색공간이 크고 총 2번의 전체데이터를 읽어야 하므로 데이터 스트림에는 적합하지 않다. [7]에서는 먼저 데이터베이스를 읽고 최소지지도 이상인 크기 1의 빈발항목 들을 선발하여 정렬하고 FP-tree라고 부르는 전위 트리(prefix tree)를 구축한 후에 탐색 알고리즘을 이용하여 연관규칙 탐사를 하는 방법이다. 이것은 기존의 단계적 처리 방식보다는 효율적이지만 트리를 구축하기 위해서는 2번의 전체 데이터베이스를 읽어야 하며 또한 지지도가 변경되면 다시 처음부터 새로운 트리를 구축한 후에 연관규칙탐사를 해야 하는 문제점이 있다. 최근의 데이터 스트림 처리 방식[8, 9, 10]에서 연관 규칙 탐사를 위해 FPT와 유사한 전위 트리를 생성하고 이로써 연속적으로 입력되는 데이터 스트림을 저장하고 관리하는 경우에는 저장할 항목의 크기와 데이터의 양이 대용량인 경우에는 주 메모리를 기본으로 한 전위 트리 운용방식에서의 메모리 부담과 입 출력하는 부담을 감당하지 못하는 문제[13]가 발생될 수 있다.

3. 한번 읽기에 의한 빈발항목 추정 기법

본 논문은 데이터 스트림을 처리하는 데 적합한 새로운 방식으로서 연속적으로 입력되는 데이터를 오직 한번만 읽고 빈발항목 집합을 연속적으로 추정하고 처리하는데 있다. 즉 데이터 스트림을 일정 구간으로 나누어서 각 구간이 연속적으로 입력되면 각 구간마다 오직 한 번만 읽고 해당 구간 내의 최대 크기의 빈발항목 집합의 크기와 항목들을 발견한 후에 필요한 모든 크기의 빈발항목 집합을 추정하여 생성하는 추정 기법을 제안한다. 추정 기법의 기본 이론은 암스트롱의 공리[12] 중에서 결합법칙(A ⇒ B이고, A ⇒ C이면 A ⇒ B∪C이다)을 이용하여 크기 2 빈발항목 집합에 최소 지지도와 최소 신뢰도를 적용하면 확장된 항목 집합 {A∪B∪C}을 결정하게 된다. 결합법칙을 사용하면 항상 100% 정확한 결과를 생성하지는 못하지만 그러나 추정 기법으로 사용하기에는 적합함을 보이고자 한다.

본 논문에서 처리하는 방법은 Dist'라고 명명하며 다음과 같은 5단계의 절차로 처리한다.



3.1 문제의 정의

먼저 연관 규칙 탐사 문제에 대한 형식적 정의를 하면 다음과 같다. $I=\{i_1, i_2, \cdots, i_m\}$을 항목들의 집합이라 하고, S를 데이터 스트림이라고 하자. 데이터 스트림 $S=\{S_1, \cdots, S_n\}$은 $1 \le d \le n$일 때 데이터 스트림의 한 구간 S_d는 시간적 단위로 구분되는 시간의 크기 T로 나누고 시간의 크기 T 내에 발생되는 트랜잭션의 개수로 구성한다. 여기서 데이터 스트림

을 나누는 시간의 구간 T는 연속적인 시간 단위 t_i로 구성되므로 즉 $T=\{t_1,\cdots,t_r\}$, $1\le i\le r$이다. 트랜잭션은 시간의 구간 T안에서 임의의 시간 t_i에서 발생되며, 처음에는 t_1로부터 시작하여 $t_i, t_{i+1}, \cdots, t_r$까지 연속적으로 발생되며 한 개의 데이터 스트림 구간 S_d를 형성한다. 각 시간의 구간 t_i마다 한 개의 트랜잭션이 발생된다고 가정하면 트랜잭션은 항목들의 집합으로서 고유한 트랜잭션 번호가 부여되며 $t_i \subseteq I$ 관계가 성립된다. 또한 X, Y⊂I일 때 X⊂t_i와 Y⊂t_i에 있어서 X∩Y=ϕ이면 트랜잭션 t_i는 X와 Y를 지지(support)한다고 하며 지지도는 supp(X) 또는 supp(Y)로 표현하며 이것은 전체 트랜잭션에 대하여 항목 집합 X, Y를 포함한 트랜잭션의 비율을 의미한다. 주어진 임계 값인 최소지지도($0 \le minsupp \le 1$)에 대해 $supp(X) \ge minsupp$이면 X를 빈발항목이라고 한다. 또한 Y도 빈발항목일 때 X와 Y가 동반하여 발생되면, 여기에서 우리는 연관 규칙 X⇒Y (X가 Y를 함축한다) 형태의 관계를 발견할 수 있다. 여기서 X는 독립적으로 발생되는 것으로 피봇 항목 집합이라고 하고 Y는 X가 발생할 때만 발생되거나 또는 발생되지 않을 수도 있는 피피봇 항목 집합이라고 하며, X가 발생하는 경우 Y의 발생 비율을 신뢰도(confidence)라고 하며 이것의 최소 임계 값을 최소신뢰도라고 한다. 최소지지도와 최소신뢰도는 본 논문에서 빈발항목 집합을 발견하기 위한 과정에서 사용하는 중요한 척도이다. 이후에는 피봇 항목 집합을 피봇이라 하고 피피봇 항목 집합을 피피봇이라 한다.

3.2 지지도 행렬의 생성

결합 법칙의 적용을 위해서 먼저 크기 2 항목 집합을 생성하여 기본 정보로 사용하는데 그것은 필요한 모든 크기의 빈발항목 집합은 크기 2 빈발항목 집합으로부터 유도될 수 있다는 동기에서 비롯된다. 따라서 크기 2인 항목집합과 그것의 지지도를 가진 행렬을 먼저 생성한다. 우리는 그것을 지지도 행렬이라고 하며 알고리즘 3-1로 생성한다.

알고리즘 3-1에서는 데이터 스트림을 구성하는 트랜잭션들이 연속적으로 입력된다. 각 트랜잭션은 여러 개의 항목들$\{i_1, i_2, \cdots, i_m\}$로 구성되어 있으며 각 항목의 값은 <표 1>과 같이 입력된다. 만일 트랜잭션이 가질 수 있는 전체 항목의 종류가 {a, b, c, d, e}로서 5개이고 모든 항목이 발생되어 입력된다면 그 순서대로 처음에는 a가 지지도행렬에서 피봇일 때 동시에 발생되는 피피봇 b, c, d, e 항목을 트랜잭션에서 집계하고, b가 피봇일 때 동시에 발생되는 c, d, e항목을 집계한다. 다음은 c가 피봇일 때 d, e항목, 그리고 d가 피봇일 때 e 항목을 집계한다. 그리고 마지막 항목 e가 피봇일 때는 집계하지 않는다. 그 이유는 다른 항목들의 집계 값들이 상 삼각행렬의 원소의 형태로 집계되었으므로 이를 대칭행렬 형태로 주 대각선의 아래 부분을 채우면 모든 집계 값이 구해지므로 고려하지 않는다. 여기서 다른 예로서 특정 트랜잭션에서 입력된 항목 집합이 {a, c, d, e}이라면 해당 트랜잭션은 2진값 {1, 0, 1, 1, 1}로 간주된다. 여기서 발생된 항목은 1로 그리고 발생되지 않은 것은 0으로 간주된다. 이 경우에는 피봇 b와 관련해서는 집계하지 않는다. 그 이유는 b에 해당되는 값이 '0'으로 간주되어 b가 피봇인 경우는 발생되지 않기 때문이다. 여기에서 피봇(pivot)이라는 의미는 지지도행렬에서 집계하기 위한 중심 항목으로서 피봇이 트랜잭션에 입력되어야 동반하여 발생되는 다른 항목에 대한 피피봇을 집계한다. 이런 방식으로 전체의 트랜잭션을 다 읽고 처리하면 각 피봇과 동반하여 발생되는 피피봇 들의 지지도행렬이 집계된다. 이 때에 피봇의 집계 값은 피피봇의 집계 값보다 항상 크거나 같게 된다. 이런 방법으로 데이터 스트림 내의 전체 트랜잭션의 모든 항목 값을 피봇과 상관시켜 집계하여 지지도 행렬을 구성한다. 지지도행렬을 구성하는 알고리즘의 복잡도를 분석하면 전체 트랜잭션의 개수를 n이라고 하고 한 개 트랜잭션의 평균 길이를 e라고 하면 복잡도는 $o(\frac{ne^2}{2})$가 된다.

〈표 1〉 특정 데이터 스트림 구간의 내용

Pro	a b c d e
T1	1 0 1 1 1 {a,c,d,e}
T2	1 1 0 0 1 {a,b,e}
T3	1 0 1 1 0 {a,c,d}
T4	1 1 1 0 0 {a,b,c}
T5	0 1 0 1 1 {b,d,e}
T6	1 0 1 1 1 {a,c,d,e}
T7	0 1 0 1 0 {b,d}
T8	1 0 1 0 0 {a,c}
T9	1 0 1 0 1 {a,c,e}
T10	0 1 0 0 1 {b,e}
.Count..	7 5 6 5 6

알고리즘 3-1: 지지도 행렬 구성

입력: 데이터 스트림 내에 여러 항목을 가진 트랜잭션
출력: 피봇과 피피봇으로 구성된 지지도 행렬

```
1. while 트랜잭션 is not empty in 데이터스트림  do {
2.       for each 항목in 트랜잭션 do {
3.            if 항목is not ZERO in 트랜잭션 then
4.                  set 피봇 of 지지도행렬 by 항목,
5.                    count 피봇 in 트랜잭션,
6.                  for  each  피봇 do {
7.                        count all 피피봇 with 피봇 in 트랜잭션
8.                  }
9.            end-if
10.               }
11.         }
12.       return all count value as a지지도행렬
```

3.2.1 지지도 행렬에서 크기 1, 2 빈발항목 집합 생성

알고리즘 3-1로 생성한 지지도 행렬은 m이 발생 가능 항목의 전체 개수일 때 m개의 행과 m개의 열로 구성되는

〈표 2〉 지지도 행렬

	a	b	c	d	e
a	7	2	6	3	4
b	2	5	1	2	3
c	6	1	6	3	3
d	3	2	3	5	3
e	4	3	3	3	6

m×m 정방행렬이며 원소는 크기 2인 항목집합의 지지도 값으로 구성된다.

원소 값은 알고리즘 3-1을 적용하여 주 대각선 윗부분인 상삼각행렬의 원소만 구한 후에 각 원소 값이 $a_{ij} = a_{ji}$인 대칭 행렬로서 구성하며 2차원 배열로 구현한다. 주 대각선 밑의 부분은 대칭행렬에서 중복되는 부분으로서 지지도를 집계할 때와 그리고 크기 2 빈발 항목을 선정 할 때 고려하지 않으므로 작업량이 절반으로 감소된다. 지지도 행렬을 생성한 후에 행렬의 주 대각선의 원소인 a_{kk}, $1 \le k \le n$ 값은 각 피봇의 지지도로서 최소지지도보다 크거나 같은 값을 크기 1 빈발항목으로 선정한다. 그리고 가로 축으로 각 피봇과 동반하여 발생되는 피피봇 항목들의 누적된 지지도를 최소지지도와 비교하여 크거나 같은 것은 크기 2의 빈발항목으로 선정한다.

3.2.2 크기 2 빈발항목 집합 생성 과정의 예

<표 1>과 같은 특정 데이터 스트림 구간의 트랜잭션 집합이 입력되고 최소지지도가 0.3(개수: 3개)일 때 알고리즘 3-1로 처리하면 <표 2>와 같은 지지도 행렬이 결과로 생성된다. 여기서 가장 왼쪽 부분에 위치한 1열의 각 항목들은 피봇을 나타낸다. 1행에 위치한 항목들은 피봇과 동반하여 발생되는 피피봇들이다. 여기서 a 항목을 피봇으로 할 때 각 피피봇과의 크기 2 항목집합의 지지도 개수는 {a, b : 2}, {a, c : 6}, {a, d : 3}, {a, e : 4}이고, b항목을 피봇으로 할 때 {b, c : 1}, {b, d : 2}, {b, e : 3}이고, c항목을 피봇으로 할 때 {c, d : 3}, {c, e : 3}이고 d항목을 피봇으로 하면 {d, e : 3} 등 10개가 생성된다. 여기에서 최소지지도 개수 3과 비교하여 크거나 같은 것을 선정하면 {a, c : 6}, {a, d : 3}, {a, e : 4}, {b, e : 3}, {c, d : 3}, {c, e : 3}, {d, e : 3}과 같이 크기 2에 해당되는 빈발항목 집합은 총 7개가 생성된다.

3.3 항목 분포표 생성

최소 지지도를 기준으로 하여 지지도 행렬을 생성한 후에 항목별 분포표인 <표 3>의 생성을 위해서는 한번 더 최소신뢰도 값 47% 이상의 값을 가지는 피피봇에 대해서 해당 피봇 행의 열 원소 값을 '1'로 표시하고 최소신뢰도 이하의 피피봇은 '0'으로 표시하여 전지(prune)함으로써 구성한다. <표 3>과 같은 항목별 분포표를 구성한다.

3.4 빈발항목 집합의 최대 크기 추정

항목별 분포표를 생성한 다음에 최대 크기의 빈발항목 집합의 크기를 추정하게 된다. 여기서 분포 값은 어떤 특정 피봇이 크기 2인 빈발항목 집합에서 몇 개의 다른 항목과 관계를 가지는가를 의미한다. 지지도 행렬인 <표 2>에서 보면 최소지지도가 0.3인 경우 피봇 a의 행 값 중에서 최소지지도 개수 3보다 크거나 같고 최소신뢰도를 만족하는 것을 선택하면 분포 값 {a, c : 6} → 1, {a, d : 3} → 1, {a, e : 4} → 1을 가지고 {a, b : 2} → 0는 제외되므로 피봇이 분포된 피피봇의 개수 n({c, d, e}) = 3에 자신을 합하면 발생 가능한 집합의 최대 크기는 4가 된다.

〈표 3〉 항목별 분포표

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

속성 3-1: 결합법칙의 적용 조건

> 크기2 빈발항목 A∪B, A∪C가 있을 때 피봇 A와 피피봇 B, C 간에 최소신뢰도를 만족하는 것에 대해서 결합법칙(union rule)을 적용하여 A⇒B이고, A⇒C이면 A⇒B∪C가 되어 크기3 빈발항목 A∪B∪C를 생성한다.

지지도 행렬을 생성한 후에 최소신뢰도가 0.47이고 최소지지도가 0.3이라고 하면 피봇에 대한 최소신뢰도를 만족하는 항목별 분포표는 <표 3>과 같다. <표 2>에서 {a:7}가 피봇일 때 {a,c:6}->86%, {a,d:3}->43%, {a,e:4}->57%가 선발되지만 d는 최소신뢰도를 만족하지 못하므로 제외된다. 따라서 선택된 크기 2 항목 {a,c}, {a,e}를 속성 3-1의 결합법칙을 적용하면 크기 3인 {a,c,e}가 생성된다.

속성 3-2: 최대 크기 결정 조건

> 항목별 분포표의 피봇 중에서 최대 분포 값 k 을 가지는 피봇의 개수가 k보다 크거나 같으면 빈발항목의 최대 크기를 k로 추정 한다.

속성 3-2를 적용하여 최종적으로 추정된 빈발항목 집합의 최대 크기를 산출하는데 있어서는 모든 피봇의 분포 값 중에서 가장 큰 최대 크기를 사용하지 않고 각 피봇의 분포 값 중에서 최대 크기 k에서 크기 k를 가지는 피봇의 개수가 k개 이상인 조건을 만족하는 k값을 선정한다. 그러나 만일 k 값을 가진 피봇의 숫자가 k개보다 적으면 k=k-1하여 최대값을 1씩 감소하고 피봇의 분포 값과 비교하여 만족되는 값을 찾을 때까지 반복하여 결정함으로써 다수의 피봇이 가지는 최대 크기를 전체의 최대 크기로 선정하는 방식을 사용한다. 그렇게 하는 이유는 크기 2인 빈발 항목 집합으로

구성된 지지도 행렬로부터 만들어지는 항목별 분포표의 각 행을 구성하는 피봇과 피피봇 항목간 결합할 최대값 즉 가장 큰 분포 값에 있어서는 어느 특정 피봇이 가지는 최대 분포 값에 의해서 결합되기 보다는 가장 많은 피봇 행이 가지는 가장 큰 값 즉 최대 분포 값과 최소 분포 값의 중간 정도의 분포로 가장 많은 항목들간에 결합할 가능성이 있기 때문이다. 따라서 가장 큰 분포 값 이면서 동시에 가장 많은 항목들이 가지는 분포 값을 최대크기로 결정한다. 최대 분포 값과 최소 분포 값은 다음의 속성 3-3를 설명할 때 언급한다.

알고리즘 3-2: 발항목 집합 최대 크기 추정 알고리즘

```
1.      set  최대크기값 from 항목별분포표
2.      do {
3.           decrement 최대크기값
4.           for each 피봇 in 항목별분포표  do {
5.              if 최대크기값 == 피봇의분포값 then
6.                   increment  최대 크기값_피봇개수
7.              end-if
8.            }
9.        } while 최대크기값_피봇개수 < 최대크기값
10.       return 최대크기값
```

알고리즘 3-2은 속성 3-2를 구현한 것으로서 최대 크기를 구하는 과정은 먼저 항목별 분포표로부터 모든 분포 값 중에서 가장 최대값을 최대 크기 값으로 정하고 다른 모든 피봇 행이 가진 분포 값과 비교하여 현재의 최대 분포 값을 가진 피봇의 개수가 최대 크기 값보다 크거나 같으면 현재의 최대 크기 값을 확정하고 그렇지 않으면 최대 크기 값에서 하나씩 감소시켜서 조건에 만족하는 최대 크기 값이 결정될 때까지 반복 수행한다.

3.5 최대 크기의 빈발항목 추정

추정 빈발항목 집합의 최대 크기를 구한 후에는 추정 빈발항목 집합 내에 포함되는 항목을 구하여 각 항목을 크기 3에서부터 최대 크기까지 열거하면 구하려는 최대 크기 내의 모든 종류의 추정 빈발 집합이 생성된다. 본 절에서는 최대 크기 안에 포함되는 빈발항목을 추정하는 방법을 서술한다.

보조속성 3-1: 추정 빈발 항목 집합의 지지도 결정

A와 B가 추정 빈발 항목 집합으로서 각 크기가 k이고 추정지지도가 최소지지도를 만족하고 또한 최소 신뢰도를 만족하면 이로부터 확장된 크기 k+1인 추정 빈발항목 집합 A∪B를 생성할 수 있다. 이 때 A∪B의 지지도는 크기 k 추정 빈발 항목 집합들의 추정지지도 중에 최소값에 해당하는 값 Esupp(A∪B) = min(supp(A), supp(B))을 가진다.

추정된 최대 크기의 빈발항목 집합에 포함되는 항목을 발견하기 위해서는 위 보조속성 3-1를 이용하여 각 피봇에 해당하는 최대 크기의 빈발항목 집합을 모두 생성한다. 즉 A와 B가 크기 k인 추정 빈발항목 집합이고 A∪B가 크기 k+1에 해당하는 추정 빈발항목 집합이 되기 위해서는 A와 B 각각이 그들 내부에 피봇에 대한 피피봇이 최소지지도를 초과하는 추정지지도(Esupp ≥ minsupp)를 가지면서 동시에 신뢰도가 최소신뢰도 이상이어야 한다. 그러한 모든 피피봇들을 모두 해당되는 피봇과 결합하여 최대 크기의 항목 집합으로 확장하는 방법을 사용하는데, 이것은 Apriori 알고리즘에서 동일한 크기 k를 가진 피봇과 피피봇을 각각 앞에서부터 k-1까지의 패턴이 서로 동일한 것을 찾아서 상호 조인(join)하여 크기 k+1인 항목집합을 생성하는 방법을 반복하는 것과 같다. 이렇게 확장된 항목 집합의 지지도는 위의 보조속성 3-1에 의해서 결정된다. 보조속성 3-1을 최대 크기까지 반복 적용하여 데이터 스트림 구간 내에서 생성해야 하는 모든 추정 빈발항목 집합을 생성한다. 즉 지지도 행렬로부터 생성한 항목별 분포표에 있는 각 피봇 항목별로 차례로 피봇과 피피봇을 결합하여 크기 3부터 최대 크기까지 확장한다.

속성 3-3: 빈발항목 추정

지지도 행렬에서 각 피봇 항목에 대한 지지도가 최소지지도를 초과하면서 최소신뢰도를 만족하는 피피봇은 최대 크기 빈발항목 집합 내에 포함되는 빈발항목으로 추정될 수 있다.

위의 속성 3-3를 적용하여 우리는 최대 크기의 빈발항목 집합에 포함되는 항목들을 지지도 행렬로부터 추정한다. 예를 들면 크기 2 빈발항목 {a, b}, {a, c}가 있을 때 피봇 a에 대한 피피봇 b 그리고 c의 신뢰도가 최소신뢰도에 포함되면 크기 3인 빈발항목은 크기 2의 항목을 결합하여 {a, b}∪{a, c}에 의해서 크기 3인 {a, b, c} 항목 집합을 구성할 수 있고 크기 3인 집합의 지지도는 보조속성 3-1에 의해서 결정된다. 또한 여기서 크기 3을 구성하는 각 항목을 빈발항목으로 추정하며 이런 방식으로 최대 크기까지 확장된 항목들을 구성한다. 이렇게 최대 크기 내에 포함될 가능한 항목들이 구성되면 3.6절 빈발항목 집합 생성의 내용과 같이 최종적인 항목 집합을 결정하여 생성한다. 빈발항목 집합의 크기와 그 항목을 추정하는 데 있어서는 지지도 행렬에서 쌍을 이루는 2개의 크기 2 빈발항목 집합 {a, b}과 {a, c}가 있을 때 만일 모두 피봇 a에 대한 피피봇 b, c의 신뢰도가 만일 100%라면 이것들을 결합하여 크기를 증가시킨 크기 3인 항목 집합 {a, b, c}의 지지도는 피봇에 대해 100% 신뢰도를 나타낸다. 즉 이 때는 $\frac{supp(a \cup b \cup c)}{supp(a)} = \frac{supp(a \cup b)}{supp(a)} = \frac{supp(a \cup c)}{supp(a)}$ 이 되어 최대 분포 값이 3이 된다. 그러나 만일 피봇 a에 대한 피피봇 b와 c의 신뢰도가 각각 60%라고 하면 크기 2

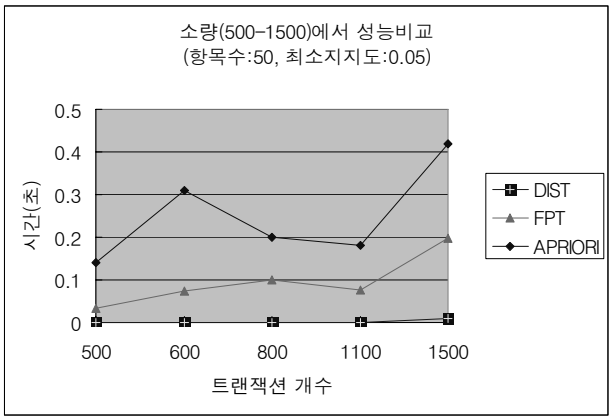
인 두 집합의 합 집합인 크기 3의 지지도가 최대의 경우에는 “피피봇 ⊂ 피봇”이 되어 피봇에 대한 피피봇의 신뢰도가 60%로써 전부가 완전 포함될 수도 있지만, 최소의 경우는 b와 c의 지지도가 분산되어 크기 3 집합 {a, b, c}에서 피피봇 b∪c의 신뢰도는 20%만이 피봇 a에 포함되는데 이런 경우에 해당 피봇의 분포 값은 3과 2 사이 값으로 결정된다. 여기서 피봇 a, b, c가 모두 항목별 분포표에서 분포 값 3을 가지면 항목 집합의 최대 크기는 3이 되고 a, b, c중에 어느 것이 분포 값 2를 가지게 되면 최대크기는 2가 된다. 또한 확장된 집합 {a, b, c}가 가질 수 있는 지지도의 최소값은 $\max(0, \text{supp}(a,b) + \text{supp}(a,c) - \text{supp}(a))$ 이고 최대값은 min(supp({a, b}),supp({a, c}))이 된다. 이와 같이 우리는 항목에 대한 최대, 최소 분포 값과 그리고 최대, 최소 가능 지지도를 최소 신뢰도를 바탕으로 구한다. 최소신뢰도의 결정에 있어서는 47%~61%를 주로 사용하는 것이 효율적임을 <표 5>와 같이 실험 결과에 의해 산출했다. 본 논문에서 사용하는 최소신뢰도는 트랜잭션이 가질 수 있는 항목 개수에 따라서 결정되는데 항목 수가 적은 경우는 작은 최소신뢰도를 사용하고 항목이 많은 경우에는 큰 최소신뢰도를 사용한다. 그 이유는 <표 5>에서 볼 수 있듯이 트랜잭션이 가지는 항목 수가 많을수록 각 항목이 상호 분산될 가능성이 커지므로 결합법칙을 적용할 때 피봇에 대한 최소신뢰도를 올려 주어야 정확도가 높아지기 때문이다. 그러나 여기서 항목이 160개 이상부터는 더 이상 증가하지 않는 것을 볼 수 있는데 그 이유는 일정한 개수 이상으로 항목수가 증가하면 지지도 행렬의 구조에서 각 항목간에 결합되는 조합을 형성하기 위한 제약 조건이 증가 함으로 더 큰 최소신뢰도에 의한 제한이 불필요하게 된다. 따라서 지지도 행렬로부터 생성된 항목별 분포표에 의해서 추정 빈발항목의 최대 크기와 항목을 추정할 때는 트랜잭션에서 출현할 수 있는 최대 항목의 숫자를 기본으로 <표 5>에 의해서 적절한 최소신뢰도를 선정하여 알고리즘 3-2을 사용한다. 예를 들면 <표 2>에 있는 지지도 행렬에서 최소신뢰도가 47%이고 최소지지도 개수가 3일 때 {a : 7}가 피봇인 경우 발생할 수 있는 피피봇과의 조합은 {a,b:2}, {a, c : 6}, {a, d : 3}, {a, e : 4}인데, 최소지지도 개수가 3이므로 {a, c : 6}, {a, d : 3}, {a, e : 4}가 선발되지만 그 중에서도 피봇 a에 대한 피피봇 c, d, e의 신뢰도는 각각 86%, 43%, 57%가 되어 최소지지도 개수와 최소신뢰도 47%를 만족하는 집합에는 {a, c}, {a,e }만 해당된다. 여기서 해당되는 것을 결합하면 항목 집합 {a, c, e}가 생성되고 이것의 추정지지도는 <표 2>의 피봇 a와 같이 발생하는 크기 2 항목의 {a, c}, {a, e}에 대해 보조속성 3-1을 적용하고 계산하면 Esupp{a, c, e} = min(supp (a, c), supp(a, e)) = 4가 된다. 다른 피봇에 대해서도 동일한 방법으로 생성하면 b에 대해서는 {b, e}가 생성되고, c에 대해서는 {c, a}, {c, d}, {c, e}로부터 이를 병합하면 {a, c, d, e}가 생성되고, d에 대해서는 {a, d}, {c, d}, {e, d}로부터 {a, c, d, e}가 생성되고, e에 대해서는 (a, e), {b, e}, {c, e}, {d, e}로부터 {a, b, c, d, e}가 생성된다.

3.6 빈발항목 집합 생성

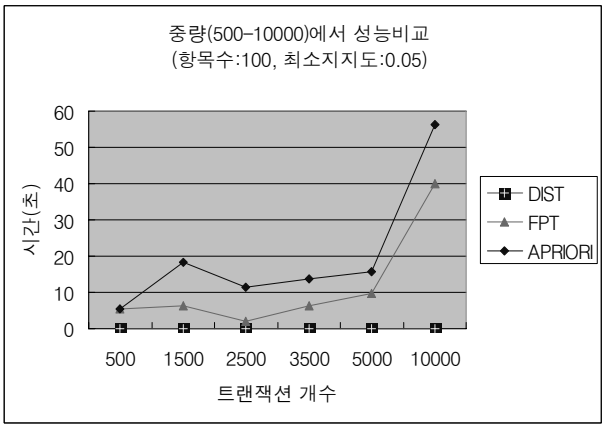
최종적인 최대 크기 빈발항목 집합에 포함되는 항목을 선정하기 위한 최종 작업으로서 만일 추정된 최대 크기의 숫자보다도 적은 개수의 항목이 산출되는 경우에는 1차로 알고리즘 3-1을 적용하여 구성된 모든 항목을 최대 크기 빈발항목으로 추정한다. 그러나 추정된 최대 크기의 숫자보다도 많은 개수의 항목이 산출되는 경우에는 항목별 분포표인 표 3-3에서 ‘1’로 표시된 항목들만을 알고리즘 3-1과 동일한 방법으로 입력하여 집계할 때 최소신뢰도를 적용하여 전지하고 남은 항목 만을 최대 크기에 포함되는 항목으로 결정한다. 그렇게 하는 이유는 알고리즘 3-1에 의한 지지도행렬과 항목별 분포표로써 처리하여 구성한 결과에는 가장 빈발하게 발생되는 가능한 모든 크기에 포함되는 항목들이 모두 포함되기 때문이다. 따라서 구성된 결과 항목 수가 최대 크기 숫자보다 많은 경우에만 2차로 알고리즘 3-1과 동일한 방법을 적용하되 항목별 분포표에 ‘1’로 표시된 항목만을 입력하여 처리하면서 최소신뢰도를 통해 한번 더 빈발 항목을 전지하여 최종 결정한다. 이에 따라서 최종적으로 추정되는 빈발항목 집합의 최대 크기는 3이고 최대 크기의 빈발항목 집합에 포함되는 항목은 {a, c, d, e}로 결정된다. 다음에는 결정된 최대 크기에 의하여 최종 결정된 항목들을 모든 크기의 항목 집합으로 열거하면 최대 크기 내의 모든 크기 별 추정 빈발항목 집합 {a}, {c}, {d}, {e}, {a, c}, {a, d}, {a, e}, {c, d}, {c, e}, {d, e}, {a, c, d}, {a, c, e}, {a, d, e}, {c, d, e}가 생성된다.

4. 실 험

실험 환경으로서 운영체제는 Linux 2.6.9-5 ELsmp이고 구현 언어는 GNU C이며 IBM 펜티엄4에 3.2GHz dual-CPU 주 메모리 1.5GB RAM을 사용했다. 성능비교는 정확한 처리의 대표인 Apriori와 최근에 데이터 스트림 방식에서 주로 사용하는 prefix 트리 방식의 대표인 fp-tree(FPT)와 본 논문의 추정기법인 Dist의 성능을 비교한다. 사용한 언어는 GNU C를 사용했고 프로그램으로는 Apriori는 뉴질랜드의 Canterbury 대학교 전산학과의 Tadao Takaoka 교수의 프로그램을 사용하고 FPT 프로그램은 [14]의 저자인 Yin-Ling Cheung의 프로그램을 사용하여 실험했다. 데이터는 실험용 데이터를 생산하여 사용하였는데 Dist와 Apriori를 위한 데이터는 한 개의 트랜잭션이 가질 수 있는 최대 항목 수를 50, 100개 2가지로 하여 트랜잭션 데이터는 1부터 항목의 개수까지의 숫자 중에서 임의의 숫자를 각 항목으로 최대 항목 수 내에서 임의의 숫자로 구성하는 방식으로 생성했다. FPT용은 Dist와 Apriori와 동일 방식이지만 단지 트랜잭션의 처음 숫자가 해당 트랜잭션이 가지는 전체 항목의 개수로부터 시작하는 것 만은 다르게 생성했다. 속도비교를 위해서는 트랜잭션의 양은 소량 실험을 위해서는 항목 수가 50인 트랜잭션을 500, 600, 800, 1100, 1500개인 파일들을 생성하고 중량 실험을 위해서는 항목수가 100개인 트랜잭션 500, 1500, 2500, 3500, 5000, 10000개인 파일을 생성하고 대량 실험을 위해서는 항



(그림 1) 소량에서의 처리 속도 비교



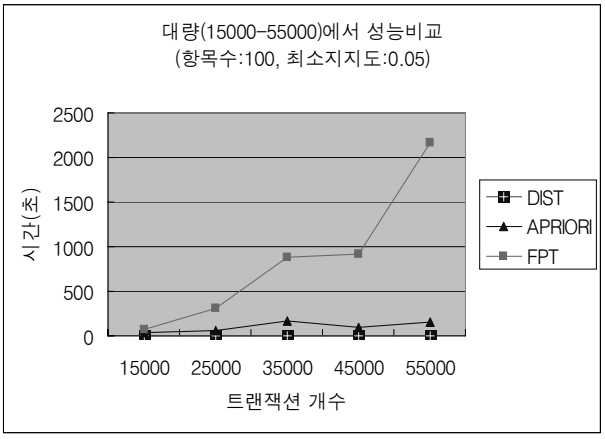
(그림 2) 중량에서의 처리 속도 비교

목 수 100개인 트랜잭션 15000, 25000, 35000, 45000, 55000개인 파일들을 생성하여 실험한다.

위의 (그림 1)은 Apriori, FPT, Dist 방식간의 처리 속도를 비교한 것이며 데이터는 항목수가 50이고 최소지지도는 0.05 그리고 트랜잭션의 양은 소량으로서 500개부터 1500개까지 속도를 비교한 것이다. 소량 실험 결과는 본 논문의 추정방식인 Dist가 가장 빠르고 둘째로 FPT, 셋째로 Apriori의 순서가 된다. 여기서 3종류간의 속도 차이가 큰 것은 Dist는 메모리 내에서 2차원 배열을 구성하여 사용하므로 자료구조를 구성하고 데이터를 저장하는데 거의 시간이 소요되지 않는 반면에 Apriori방식은 트리를 구성하고 전체 데이터를 최대 크기까지 반복하여 읽으면서 처리하기 때문에 최대 빈발항목 수가 작으면 속도가 빠르지만 빈발항목의 최대크기가 커질수록 그 크기만큼 반복해서 전체를 읽어야 하므로 속도가 늦어진다. FPT는 데이터전체를 한번 읽고 크기 1 빈발항목을 찾아서 크기 순으로 정렬한 후에 트리를 뿌리만으로 구성한 후에 한번 더 전체 데이터를 읽어서 트리에 데이터를 채운 다음에 FP-growth단계에서 빈발항목을 처리 할 때 동일한 경로 상의 확장된 다른 패턴을 찾기 위해서 반복적으로 해당 경로를 탐색하므로 데이터 양이 커질수록 FPT의 속도는 늦어지게 된다.

아래의 (그림 2)는 트랜잭션의 양을 늘려서 중량으로서 500, 1500, 2500, 3500, 5000, 10000개까지의 처리속도를 비교한 것으로 소량에서와 동일하게 Dist가 가장 빠르고 둘째로 FPT, 셋째로 Apriori순이다. 여기서는 FPT 와 Apriori의 속도 차이가 평균적으로 소량에서는 2.6배 차이에서 중량에서는 1.7배로 줄어드는 것을 볼 수 있다.

아래의 (그림 3)은 대량으로서 트랜잭션 15000, 25000, 35000, 45000, 55000개에서의 속도비교인데 여기서는 순위가 약간 바뀌어 Dist가 가장 빠르고 둘째는 Apriori 셋째는 FPT순으로 된다. 이것을 보면 데이터가 소량 또는 중량인 경우에는 FPT가 Apriori보다 빠르지만 대용량의 데이터를 처리 할 때 또는 빈발항목의 최대크기가 크지 않은 경우에는 항상 FPT가 항상 빠르지는 않다는 것을 알 수 있다. Dist 경우에는 메모리에서 지지도 행렬을 위한 2차원 배열만을 구성하여 Dist알고리즘을 이용하여 추정하는 방식으로



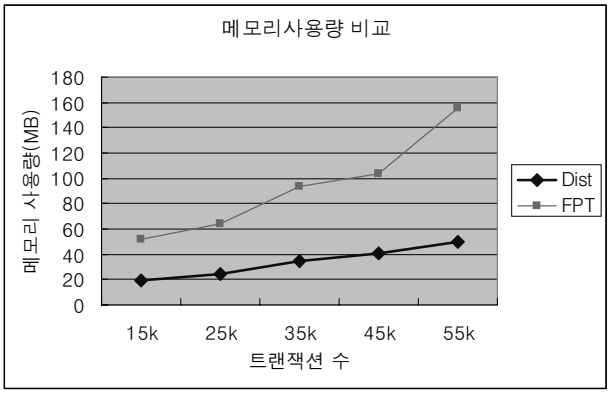
(그림 3) 대량에서의 처리 속도 비교

〈표 4〉 메모리 사용량 비교표

항목 수-트랜잭션 수	FPT 메모리 사용량 (KB) VmSize, Vmlck, VmData, VmStk				Dist 메모리 사용량 (KB) VmSize, Vmlck, VmData, VmStk			
100 - 15000	17660	16352	15988	384	9272	596	6188	1800
100 - 25000	21736	20340	19948	500	11956	300	9996	676
100 - 35000	32248	29324	28924	2036	17112	676	14076	1752
100 - 45000	35200	33440	33016	896	20268	488	18024	960
100 - 55000	52496	50720	50308	900	24404	588	21968	1152

소량, 중량, 대량의 모두의 경우에 1초 미만의 빠른 속도로 처리함을 볼 수 있다.

다음 <표 4>은 위의 대량 데이터를 처리할 때 FPT 와 Dist의 메모리 사용량을 비교한 것이다. 측정 단위는 'cat /proc/{pid}/status' 명령어의 출력 항목으로서 프로세스 번호를 이용하여 해당 파일을 1초 간격으로 메모리 관련 정보를 읽어 파일에 저장하고 최종으로 사용된 크기를 집계하였다. 표에서는 VmRSS: 물리적 메모리 크기, VmExe: 실행코드 영역, VmLib: 동적 라이브러리 영역 등은 유사하여 제외



(그림 4) 대량에서의 메모리 사용량 비교

〈표 5〉 항목 수에 따른 적정 최소 신뢰도의 변화

항목수	20	40	60	80	100	120	140	160	180	200
최소 신뢰도	46.5	51.38	51.8	56.2	57.1	58.8	58.8	60.6	60.6	60.6

하고 VmSize: 가상 메모리 크기, VmLck: 사용중인 메모리 크기, VmData: Heap 영역, VmStk: Stack 영역 등을 비교 표시한다.

모든 측정단위의 합계로써 100개 항목의 트랜잭션 개수별 메모리 사용량을 그래프로 비교하면 (그림 4)와 같다. 여기서는 <표 4>의 항목과 VmRSS(FPT; 0, Dist; 0), VmExe (FPT; 0.01, Dist; 0.005), VmLib(FPT; 1.25, Dist; 1.251)를 포함한 것으로 FPT의 메모리 소요량이 Dist보다 평균적으로 2.8배 많으며 트랜잭션 수가 증가할수록 그 차이가 커지는 것으로 나타난다.

<표 5>은 지지도 행렬로부터 항목별 분포표를 구성할 때 기준에 되는 최소신뢰도 값의 기준 값을 집계한 표로서 이 것은 빈발항목의 최대 크기를 추정할 때와 최대 크기의 빈발항목 집합에 포함되는 항목들을 추정할 때에 사용한다.

<표 5>의 최소신뢰도 값은 주로 트랜잭션이 가질 수 있는 최대 항목 수에 따라서 결정된다. 그러나 데이터 구간의 크기를 결정하는 트랜잭션의 개수에도 약간의 영향을 받는데 즉 데이터 구간의 크기가 변동하면 동일한 항목수라도 그 기준 값을 보정해 주어야 하는데 그 이유는 (최소지지도_개수 = 최소지지도 × 트랜잭션_수)이 되어 데이터 구간의 크기를 결정하는 트랜잭션의 개수가 많아지면 전지(prune)할 때 기준이 되는 최소 지지도는 일정한 비율로 증가하지만 각 항목 집합의 지지도는 동일한 비율로 증가하지 않기 때문에 결국 피봇의 지지도와 피봇과 피피봇의 결합 지지도에 의해서 결정되는 <표 5>의 최소 신뢰도를 보정해야 한다. 보정하는 방법은 다음과 같다.

데이터 구간의 크기 기준은 StdTranNum로 하고, 현재 데이터 구간의 크기는 CurTranNum로 하고, 데이터구간 기준 크기에서 증가된 크기로 인해 발생되는 차이에 대한 평균 보정 값으로서 트랜잭션당 보정계수는 TranRate = 0.0000028로 한다,

〈표 6〉 Apriori과 Dist방식의 추정 정확도 비교 자료

항목수-트랜잭션볼륨		20-1000	40-1000	60-1000	80-1000	100-1000
Apriori의 정확한 결과	최대크기 내의 빈발항목	1, 2, 3, 5, 12, 16, 20	1,3,12,18 22,25,30	10,17,23, 38,40,41, 51	1,4,10,12, 14,19,25,28 31,57,65,66, 72	1,2,8,12,14, 20,28,31,57, 65,66,72
	빈발항목 최대크기	3	3	4	7	7
본 논문의 (Dist)의 추정결과	빈발항목 최대크기	3	3	4	7	7
	빈발항목 추정 False hit	0%	0%	0%	0%	0%
	빈발항목 추정 False Dismissal	14%	14%	28.6%	15%	20%
	정확도	86%	86%	71.4%	85%	80%

최소신뢰도 = minconf × 100이고, pminconf = 1 / minconf 일 때 처음에 pminconf = pminconf + ((CurTranNum - StdTranNum) × TranRate를 계산한 후에 최소신뢰도 = (1 / pminconf) × 100를 계산하여 적용한다. 예를 들어 설명하면 데이터 구간의 크기 기준을 트랜잭션 500개로 하고 항목 수를 100개로 하고 현재 데이터 구간의 크기가 트랜잭션 1000개라고 하면 최소 신뢰도 57.1 = minconf(0.571) * 100을 적용할 때 <표 5>에 있는 그대로 적용하지 않고 현재 pminconf(1.75131) = 100 ÷ 최소신뢰도(57.1)이고 변경된 pminconf = ((현재 데이터구간 크기 - 기준 데이터구간 크기) × 보정수) + 기존 pminconf을 적용하면 ((1000 - 500) × 0.0000028) + 1.75131 = 1.75271이다. 따라서 기존의 pminconf = 1.75131로부터 새로운 pminconf = 1.75271로 보정되어 변경된 최소신뢰도 57.055%(100 ÷ 1.75271)를 적용한다. <표 5>을 구성하기 위해서는 항목 개수를 20개에서 200개를 가진 트랜잭션의 개수 500개와 1000개를 테스트하여 빈발항목의 최대 크기를 산출하여 정확한 최대 크기가 산출될 때까지 반복 조율을 통해 생성한 것으로서 한 개의 데이터 구간을 트랜잭션 500개 기준으로 설정한 최소신뢰도의 평균값이다. 트랜잭션의 항목 수에 따라서 해당 최소신뢰도를 적용하면 false hit과 false dismissal을 줄이고 좀 더 정확한 추정을 할 수 있다.

<표 6>은 빈발항목 집합의 최대 크기와 최대 크기 내의 빈발항목 추정에 대한 정확성 검증에 관한 결과이다. 이것은 Apriori 알고리즘의 결과와 비교 검토 한 것이다. 실험결과는 빈발항목 집합의 최대 크기에서 최소지지도를 적용한 후에 최소신뢰도를 적용할 때 Apriori와 동일한 결과가 산출되었으나 최대 크기의 빈발항목 집합에 포함되는 항목에 대한 추정의 정확도는 약간의 오차가 발생된다. 여기서는 트랜잭션이 가질 수 있는 항목 수 20~80개를 트랜잭션의 개수 1000개로 실험했는데 표는 위 아래 2개의 부분으로 구성되며 윗부분은 Apriori의 정확한 결과 값 중에서 빈발항목의 최대 크기와 최대 크기에 포함되는 빈발항목이 숫자로 표현되어 있고 아랫부분은 본 논문의 Dist로 추정한 결과를 나타낸다.

빈발항목의 최대 크기내의 빈발항목을 추정한 결과에서 잘못된 것이 포함된 false hit비율과 그리고 추정하지 못한 false dismissal 비율에 대한 추정의 정확도를 나타낸다. 정확도는 평균적으로 81.7%로 나타난다

5. 결 론

본 논문은 실시간이고 연속적으로 입력되는 데이터 스트림을 사용하는 응용의 연관 규칙 탐사에서 최대 크기의 빈발항목 집합을 발견하는 방법을 제안했다. 기존의 단계적 처리 방식이나 교집합 처리 방식 또는 병렬 처리 방식에서 단계적 반복적으로 데이터베이스를 읽으면서 빈발항목을 구하는 문제점을 극복하고 또한 기존의 데이터 스트림을 처리하는 방식에서 빈발항목 집합을 구성하고 관리 및 운용하기 위해 필요로 하는 prefix 트리 등과 같은 부담이 크고 처리 시간이 증가하는 저장 구조를 사용하지 않고 데이터 스트림을 읽는 동시에 기본 크기의 항목 집합의 지지도를 구하고 이를 메모리의 배열에 저장하여 그로부터 빈발항목을 효율적으로 추정함으로써 처리속도를 크게 향상시킨 새로운 방법을 제시했다. 즉 각 데이터 스트림을 한 번만 읽고 지지도 행렬을 구성하고 그로부터 크기 1과 크기 2 빈발항목 집합을 동시에 생성하여 크기 2 빈발항목으로부터 빈발항목의 최대 크기를 추정하고 또한 최대 크기 내에 존재할 수 있는 가능한 빈발항목을 발견하는 Dist 알고리즘을 제시했다. 또한 그것으로부터 가능한 모든 크기의 빈발항목 집합을 생성함으로써 각 데이터 스트림을 오직 1회 읽기를 통해서 연관 규칙 탐사를 위한 최대 크기의 빈발항목 집합을 생성할 수 있는 새로운 방법이다. 본 논문에서 제안한 방법은 기존의 데이터베이스 시스템에도 적용 가능하지만 특히 데이터 스트림의 특성을 처리해야 하는 응용에 적용할 기반을 마련한 것이 장점이다. 그러나 단점으로는 본 논문에서 제시한 방법은 추정 기법으로서 정확한 처리가 요구되는 응용에는 기존의 단계적 처리 방법보다 정확도가 낮다는 단점이 있다. 향후에는 현재보다 더욱 향상된 정확도 확보를 위하여 보정 기법을 연구할 예정이며 또한 더욱 데이터 스트림의 특성에 적합한 데이터의 추가, 갱신, 삭제 등에 관한 추정 기법을 연구할 예정이다.

참 고 문 헌

[1] R. Agrawal, T. Imielinski, and A. Swami, “Mining association Rules between sets of items in large databases,” In Proceedings of the ACM SIGMOD international Conference on Management of Data, Washington, DC, pp.207-216, May, 1993.

[2] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” In Proceedings of the 20th VLDB Conference, Santiago, Chile, Sept., 1994.

[3] Ashok Savasere, Edward Omiecinski, Shamkant Navathe, “An Efficient Algorithm for Mining Association Rules in Large Databases,” In Proceeding of the 21st VLDB Conference, Zurich, Swizerland, 1995.

[4] David W. Cheung, Kan Hu, Shaowei Xia, “Asynchronous Parallel Algorithm for Mining Association Rules on a Shared-memory Multi-processors,” SPAA98, Puerto, Vallata, Mexico, 1998.

[5] R. Agrawal, J. C. Shafer, “Parallel Mining of Association Rules,” IEEE Transaction On Knowledge And Data Engineering, Vol.8, pp.962-969, 1996.

[6] Nan Jiang, Le Gruenwald, “Research Issues in Data Stream Association Rule Mining,” Sigmod Record, Vol.35, No.1, Mar., 2006.

[7] Jiawei Han, Jian Pei, Yiwen Yin, Runying Mao, “Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach,” Data Mining and Knowledge Discovery, Vol.8, pp.53-87, 2004.

[8] J.H Chang and W.S.Lee, “Finding Recent Frequent Itemsets Adaptively over Online Data Streams,” In Proceedings of the SIGKDD’03, Washington, DC, USA, August 24-27, 2003.

[9] Hao Huang, Xindong Wu, Rechard Relue, “Association Analysis with One Scan of Databases,” IEEE int’l Conf on Data Mining, December, 2002.

[10] James Cheng, Yiping Ke, and Wilfred Ng, “Maintaining Frequent Itemsets over High-Speed Data Streams,” In Proceedings of the PAKDD2006, April 9-12, Singapore, 2006.

[11] Gab-Soo Han, ”The Research for Efficient Candidate generation in Association Rule Data Mining,” Journal of Doowon Technical college, pp.221-228, December, 2005.

[12] W.W.Armstrong, “Dependency Structures of Data base Relationships,”In Proceedings of the 1974 IFIP Congress, pp.580-583, 1974.

[13] Jiawei Han, Micheline Kamber, “Data Mining Concepts and Techniques,” Morgan Kaufmann Publishers, pp.242, 2001.

[14] Yin-Ling Cheung and Ada Wai-Chee Fu, “Mining Frequent Itemsets without Support Threshold: With and without Item Constraints,” IEEE Transactions on Knowledge and Data Engineering, Vol.16, No.9, September, 2004.

한 갑 수

e-mail : gshan@doowon.ac.kr

1982년 동국대학교 전자계산학과(경영학사)

1990년 연세대학교 산업공학과(공학석사)

1997년 KAIST 정보및통신공학과 컴퓨터공학 (공학석사)

2001년 KAIST 전자전산학과 전산학전공 박사수료

1981년~1985년 대한항공 시스템개발부 DBA

1986년~1994년 삼성SDS 정보기술연구소 수석

1995년 1월 전자계산조직응용 기술사 취득

1994년~1998년 한국오라클 컨설턴트

2007년 4월 정보시스템 수석감리원 취득

1998년~현재 두원공과대학 컴퓨터정보과 교수

관심분야 : 데이터베이스, 데이터마이닝, 정보시스템감리