工業經營學會誌 第 20卷 第 43輯 1997年 8月 265

데이타 마이닝에서 기존의 연관 규칙을 갱신하는 앨고리듬 개발
- An Algorithm for Updating Discovered Association Rules in Data Mining -

이 동 명*
Lee, Dong-Myoung
지 영 근**
G, Young-Gun
황 종 원***
Hwang, Jong-Won
강 맹 규****
Kang, Maing-Kyu

ABSTRACT

There have been many studies on efficient discovery of association rules in large databases. However, it is nontrivial to maintain such discovered rules in large databases because a database may allow frequent or occasional updates and such updates may not only invalidate some existing strong association rules but also turn some weak rules into strong ones. The major idea of updating algorithm is to resuse the information of the old large itemsets and to integrate the support information of the new large itemsets in order to substantially reduce the pool of candidate sets to be re-exmained.

In this paper, an updating algorithm is proposed for efficinet maintenance of discovered assocation rules when new transaction data are added to a transaction database. And superiority of the proposed updating algorithm will be shown by comparing with FUP algorithm that was already proposed.

1. 서론

최근 들어 데이타 마이닝(data mining)은 의사결정 지원, 시장 전략 등을 포함한 많은 분야에서 그 다양한 적용성 때문에 데이타베이스 분야에서 많은 주목을 끌고 있다.

데이타베이스 분야에서 연관 규칙을 발견하는 문제는 Agrawal 등[3]에서 처음으로 다루어졌다. 이 논문에서 연관 규칙을 발견하는 문제는 두개의 하위 문제로 나뉘어짐을 보이고 있다.

* 한양대학교 산업공학과 석사과정
** 한양대학교 산업공학과 박사과정
*** 한양대학교 산업공학과 박사과정
**** 한양대학교 산업공학과 교수

첫번째로 최소 지지도(minimum support) 보다 큰 다량의 트랜잭션에 포함된 항목집합(itemsets)을 규명해야 한다. 두번째로 일단 모든 빈발 항목집합들이 얻어지면 원하는 연관 규칙을 간단하게 만들어낼 수 있다. 대부분의 논문들은 이 방법을 따랐고 주로 빈발 항목집합을 찾는 것에 초점을 두었다.

연관 규칙을 발견하는 문제는 트랜잭션에 빈번히 나타나는 항목들의 모임인 빈발 항목집합들을 찾아내는 일로 바꿀 수 있다. 빈발 항목집합들을 찾아내는 문제는 항목집합들의 후보집합을 생성하고 그 가운데서 빈발 항목집합의 조건을 충족시키는 항목집합들을 추출해냄으로써 해결된다.

지금까지의 연구에 의하면 데이타 마이닝의 관건은 대용량의 데이타를 처리하고, 많은 수의 규칙을 관리하고, 발견된 규칙을 매우 긴 시간에 걸쳐 유지, 보수하는 기법에 달려있다는 사실을 보여준다. 따라서, 데이타베이스 마이닝이 의미를 가지기 위해서는 다음의 두 가지 문제가 필수적으로 해결되어야 한다.

1. 상이한 종류의 규칙이나 패턴을 발견하는 효율적 앨고리듬의 개발
2. 기발견된 규칙을 갱신, 유지, 관리하는 효율적 앨고리듬의 개발

첫 문제는 데이타 마이닝에서 많은 연구가 이루어지고 있다[3, 4, 5, 7]. 그러나 기발견된 규칙을 갱신하는 연구는 거의 없었다[6].

둘째 문제는 데이타베이스가 추가되어 전체 데이타베이스가 갱신됨에 따라 새로운 연관 규칙이 발견되거나 기존의 연관 규칙이 소멸될 수 있기 때문에 대용량 데이타베이스에서 연관 규칙을 갱신하는 효율적 앨고리듬에 관한 연구는 매우 중요하다. 이는 본 연구에서 다루는 주제이다.

본 연구에서는 트랜잭션 데이타베이스에 새로운 트랜잭션들이 추가될 때 기발견된 연관 규칙을 효율적으로 유지, 보수하는 갱신 앨고리듬을 제시한다. 그리고 기존의 갱신 앨고리듬인 FUP와 실험결과를 비교함으로써 제안하는 앨고리듬의 우수성을 보인다.

2. 기본 개념

2.1데이타 마이닝

데이타베이스가 여러 분야에 응용됨에 따라 데이타베이스의 규모가 대용량화되고 있다. 데이타베이스가 대용량화 됨에 따라 데이타베이스 모델링에서는 현실 세계의 모든 규칙성(regularity)을 반영하지 못하고 있다. 이런 규칙성을 발견하기 위해 데이타베이스 모델링에 반영되지 못하고 감추어져 있는 규칙성을 발견하는 연구가 활발히 시작되고 있다.

일례로, 바코드(bar-code) 기법의 발전은 소매 조직으로 하여금 방대한 양의 판매 데이타의 수집 및 저장을 가능하게 하였다. 그리고 이와 같이 수집된 다량의 데이타를 다루기 위한 요구는 인공지능 분야의 연구에서 구분되어 데이타베이스 분야의 데이타 마이닝(data mining)으로 특징짓게 되었다. 마이닝에서 발견된 규칙은 마케팅, 주가 예측, 시장 전략, 공정 관리 등 업무 분야의 특성을 효과적으로 대변하여 의사결정에 유용한 정보를 제공한다[1, 2].

데이타베이스에서 연관 규칙을 발견하는 문제는 가장 중요한 데이타 마이닝 문제들 중 하나이다. 대용량의 트랜잭션들이 데이타베이스에 누적된 환경에서 사건 부류 혹은 트랜잭션간의 상호 관계를 발견하는 작업을 데이타베이스 상의 연관 규칙 발견(mining association rules)이라 한다. 예를 들어 슈퍼마켓의 경우 상품 판매가 카운터에서 트랜잭션 별로 기록되어지고 이런 트랜잭션을 바탕으로 95%의 고객이 상품 A, B를 구입하면 상품 C, D를 구입한다는 연관 규칙이 생성될 수 있다. 연관 규칙은 트랜잭션이 발생한 업무 분야의 특성을 효과적으로 대변

하여 의사 결정에 유용한 정보를 제공하기 때문에, 대용량의 트랜잭션으로 구성된 데이타베이스로부터 연관 규칙을 발견하기 위한 연구가 최근 활발히 이루어지고 있다.

2.2. 연관 규칙

$I = \{ i_1, i_2, \ldots, i_m \}$를 항목(item)이라 불리는 문자들의 집합, D는 트랜잭션들의 집합이고 각 트랜잭션 T는 $T \subseteq I$인 항목들의 집합이라고 하자. 한 트랜잭션에서 구입하는 항목들의 양에는 상관하지 않고, 각 항목은 항목의 구입 여부를 나타낸다. 그리고 각 트랜잭션은 TID라 불리는 식별자와 연계되어 있다. X를 항목들의 집합(itemset)이라 하고 트랜잭션 T가 X를 포함한다고 하면, $X \subseteq T$이라 한다. 연관 규칙(association rules)은 $X \Rightarrow Y$의 형식으로 표시되고, 여기서 $X \subset I$, $Y \subset I$, 및 $X \cap Y = \varnothing$ 이다. $X \Rightarrow Y$는 신뢰도(confidence) c를 가지고 있다는 것은 X를 포함하는 D의 트랜잭션들 중 c%가 Y도 포함하고 있음을 의미한다. 또한, $X \Rightarrow Y$는 트랜잭션 집합 D내에 지지도(support) s%를 갖는데 이는 D의 트랜잭션들 중 s%는 $X \cup Y$를 포함하고 있음을 뜻한다. 최소 지지도 이상을 갖는 항목집합을 빈발 항목집합(large itemset 또는 frequently itemset)이라 한다. k개의 항목들로 이루어진 빈발 항목집합을 빈발 k-항목집합이라 한다. 빈발 k-항목집합들의 집합을 L_k라 하고 이를 위한 후보 k-항목집합들의 집합을 C_k라 한다.

연관 규칙을 발견하는 과정은 다음의 두 단계로 구성된다.

단계1. 빈발 항목집합들을 찾는다. 예컨대 주어진 최소 지지도 이상의 트랜잭션들의 지지를 갖는 항목집합들의 모든 집합을 찾는다.

단계2. 데이타베이스에 대한 연관 규칙을 찾기 위해 빈발 항목집합들을 사용한다.

발견된 연관 규칙은 지지도 및 신뢰도를 가지는 수치값으로 정량화된다.

3. 기존 연구

3.1 앨고리듬 Apriori[3]

Apriori 앨고리듬은 각각의 반복시행에서 빈발 항목집합들에 대한 후보 집합을 생성하고 각 후보 항목집합의 빈도수를 계산한 후 주어진 최소 지지도 이상의 빈발 항목집합들을 결정하게 된다. 최초의 반복시행에서 Apriori는 각 항목에 대한 빈도수를 계산하기 위해 모든 트랜잭션을 읽는다. 트랜잭션 데이타베이스가 그림 3.1과 같이 주어졌을 때 데이타베이스 D를 검색하여 1-항목집합들의 빈도수를 계산하여 후보 1-항목집합들의 집합 C_1이 얻어진다. 최소 지지도가 2라고 하면, 최소 지지도를 가진 후보의 1-항목집합들로 구성된 빈발 1-항목집합들 L_1이 결정된다. 빈발 2-항목집합들의 집합 L_2을 찾을 때, 빈발 항목집합들의 모든 부분 집합들이 최소 지지도 이상을 가져야 한다는 사실로부터, Apriori는 항목집합들의 후보 집합인 C_2를 생성하기 위해 $L_1 * L_1$를 사용한다. 여기서 $*$는 접속(concatenation)연산이다.

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

그림3.1 데이타 마이닝을 위한 예제 트랜잭션 데이타베이스 D

다음으로 데이터베이스 D내의 4개 트랜잭션들을 읽고 C_2의 각 후보 항목집합의 지지도를 세어 본다. 빈발 2-항목집합들의 집합 L_2는 C_2의 후보 2-항목집합의 지지도를 기반으로 결정된다.

후보 3-항목집합들의 집합 C_3는 다음과 같이 L_2로부터 생성된다. {BC}, {BE}와 같이 동일한 첫번째 항목을 가지고 있는 두개의 빈발 2-항목집합들을 찾는다. 그리고 Apriori는 이들의 두 번째 항목들로 구성된 {CE}가 빈발 2-항목집합들을 이루는지 여부를 검사한다. {CE} 자체가 빈발 항목집합이고, 따라서 {BCE}의 모든 부분 집합들이 빈발 항목집합이므로 {BCE}는 후보3-항목집합이 된다. 이와 같은 후보 생성방법의 제시가 Apriori 앨고리듬에서 기여한 가장 중요한 것으로 평가 받고 있다. L_2로부터 그 밖의 다른 후보 3-항목집합이 나오지 않는다. 이 때 Apriori는 모든 트랜잭션들을 읽어 빈발 3-항목집합들인 L_3를 찾는다. L_3로부터는 후보 4-항목집합들이 나오지 않으므로 Apriori는 빈발 항목집합들을 찾는 작업을 끝낸다. 그림 3.2는 전체적인 과정을 도시한 것이다.

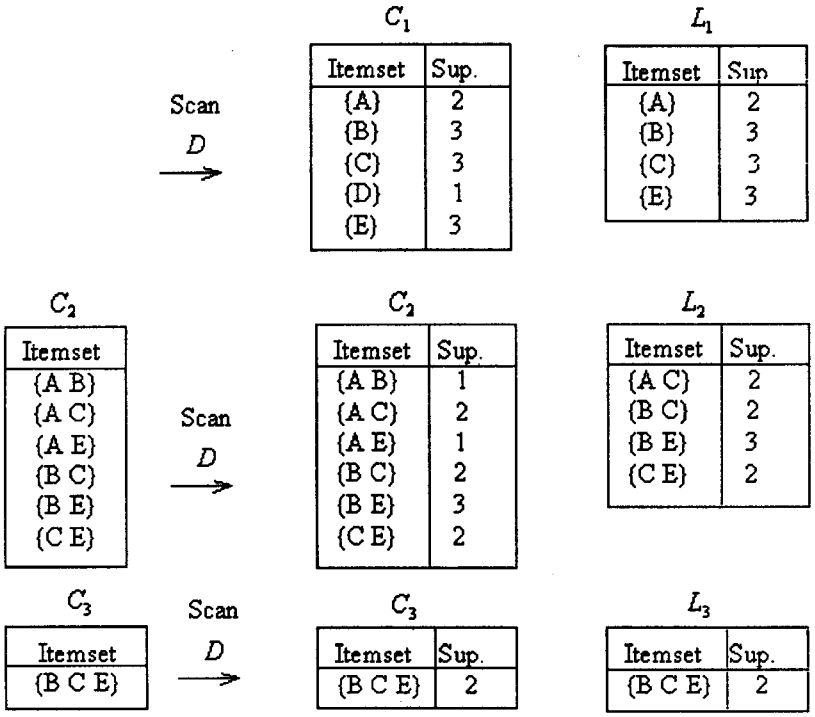

그림 3.2 후보 항목집합들과 빈발 항목집합들의 생성

3.2 앨고리듬 DHP[4]

DHP 앨고리듬은 다음과 같은 두가지 특징을 가지고 있다. 그것은 후보 항목집합들의 효율적인 생성과 트랜잭션 데이타베이스 크기의 효과적인 감소이다.

대용량의 데이타베이스가 주어졌을 때 초기에 데이타베이스로부터 유용한 정보를 얻는 것은 가장 높은 비용을 요한다. 처음 두번의 반복시행에서의 연산 비용 즉, L_1와 L_2를 구하는 과정에서의 연산비용은 총 연산비용을 좌우하는 것으로 실험을 통해 입증되었다. 이것은 다음과 같은 이유로 설명된다. 최소 지지도가 주어졌을 때, 일반적으로 시행되어야 할 다수의 C_2 항목집합들을 만들어내는 L_1이 있게 된다.

Apriori에서는 $|C_2| = \binom{|L_1|}{2}$ 이므로 C_2에 의해 구축된 해시 트리에 대해 데이타베이스의 모든 트랜잭션들을 검색하여 L_2를 결정하는 단계의 비용이 매우 커진다. 이것에 비해, DHP는 해시 테이블 H_2를 이용하여 훨씬 더 적은 C_2를 구성함으로써 L_2를 결정하는 과정에서 항목들의 전지(prune)로 인하여 다음 단계의 빈발 항목집합을 찾기 위한 데이타베이스 D_3는 작은 크기가 된다. C_2가 L_2와 거의 비슷하게 적어질 수 없다면, 데이타베이스의 효과적인 감소는 기대할 수 없다. 이 단계 후에 L_k의 크기는 k가 증가함에 따라 급속히 감소한다. 보다 적은 L_k는 보다 적은 C_{k+1}을 만들어냄으로써 연산 비용도 줄일 수 있다. 이와 같은 관점에서 앨고리듬 DHP는 초기의 반복 시행시에 C_k의 항목들의 수를 현저하게 감소시킬 수 있도록 고안되었다.

또한, 해시 기법(hash technique)은 DHP로 하여금 후보 항목집합들의 생성을 매우 효과적이게 한다. 특히 빈발 2-항목집합에서 더욱 그러하다. DHP에 의해 생성된 후보 항목집합의 수는 양적으로 기존 방법에 비해 더 적기 때문에 전체 프로세스의 수행상의 병목을 개선한다. 덧붙여 DHP는 트랜잭션 데이타베이스 크기를 점차적으로 감소시키기 위해 효과적인 전지 기법을 사용한다. DHP에 의한 보다 적은 수의 후보 집합의 생성은 반복 시행의 초기 단계에서는 트랜잭션 데이타베이스를 효과적으로 감소할 수 있게 하며, 그러므로써 계산 비용을 줄여 준다. 연관 규칙의 특성을 이용함으로써 트랜잭션의 수 뿐만 아니라 각 트랜잭션 항목의 수를 감소시킬 수 있음을 보여 주었다.

3.3 앨고리듬 SS[5]

SS(selective scan) 앨고리듬은 DHP에서 생성된 C_2를 기반으로 모든 단계의 후보 k-항목집합을 한꺼번에 생성하는 방법을 이용하여 효율적인 데이타베이스 검색을 가능하게 한다.

DHP에서와 같이 빈발 1-항목집합 L_1과 해시 테이블 H_2에서 후보 2-항목집합 C_2를 생성시킨다. 그 다음 후보 k-항목집합 C_k를 그 이전의 후보 $(k-1)$-항목집합 C_{k-1}로부터 생성시킨다. 이러한 후보 생성방법이 기존의 Apriori와 DHP의 방법과 다른 점이다. 이렇게 함으로써 두 번의 데이타베이스 검색으로 모든 빈발 항목집합 L_k를 구할 수 있다.

또한, C_{k-1}로부터 C_k를 생성했을 때 L_{k-1}로부터 L_k를 생성하는 것보다 그 수가 급격히 증가하는 경우를 고려할 수 있는데 이 때에는 그 단계에서만 C_k를 L_{k-1}로부터 생성하여 후보 항목집합의 수가 급격히 늘어나는 현상을 방지한다.

3.4 앨고리듬 FUP[6]

데이타베이스에 새로운 트랜잭션들이 추가됨에 따라 기존의 연관 규칙에 변화를 가져오게 된다. 갱신 문제를 풀기위한 하나의 접근 방법은 연관 규칙을 발견하는 기존의 앨고리듬을 갱신된 데이타베이스 전체에 대해 재수행하는 것이다. 이는 갱신 상황에 대한 어떠한 점도 고려하지 않는다.

Cheung 등[6]은 갱신 문제의 특성을 반영하는 앨고리듬 FUP(fast update)를 제시하였다. FUP는 다음과 같이 이미 만들어진 빈발 항목집합을 재사용하여 새로운 빈발 항목집합을 찾을 때 후보 항목집합이 현저하게 전지될 수 있게 한다.

최소 지지도 s, 트랜잭션 수 D인 데이타베이스 DB에서 빈발 항목집합 L의 원소 $X \in L$에 대해 지지 횟수를 $X.support$ 라고 하면 새로운 트랜잭션 수 d로 구성된 데이타베이스 db가 기존의 데이타베이스 DB에 추가된다. 여기서 최소 지지도 s에서 갱신된 전체 데이타베이스 $DB \cup db$의 항목집합 X가 $X.support \geq s \times (D+d)$이면 빈발 항목집합이 된다. 즉, 갱신된 전체 데이타베이스 $DB \cup db$에서 빈발 항목집합 L^*를 찾아낸다.

FUP 앨고리듬은 Apriori나 DHP를 재수행하는 것보다 2~16배 빠른 것으로 나타났다.

4. 제안하는 앨고리듬

본 연구에서 제안하는 갱신 앨고리듬은 FUP 앨고리듬에서 제시한 기존 정보를 재사용하는 방법을 기반으로 SS 앨고리듬에서의 후보 k-항목집합을 생성하는 방법을 이용함으로써 데이타베이스가 추가될 때 연관 규칙의 효율적인 갱신이 이루어지도록 한다.

4.1 기호 설명

본 연구에서 사용되는 기호는 다음과 같다.

L_k	:	기존 데이타베이스의 빈발 k-항목집합들의 집합
L_k^*	:	갱신된 전체 데이타베이스의 빈발 k-항목집합들의 집합
C_k	:	후보 k-항목집합들의 집합
H_2	:	2-항목집합에 대한 해시 테이블
s	:	최소 지지도
DB	:	기존 데이타베이스
db	:	추가되는 데이타베이스
$DB \cup db$	:	전체 데이타베이스
D	:	기존 데이타베이스의 트랜잭션의 개수
d	:	추가되는 데이타베이스의 트랜잭션의 개수
$X.support$	:	항목집합 X의 지지 횟수(지지도)
W_k	:	지지 횟수가 s d 이상인 후보 항목집합들의 집합
$\mid W_k \mid$	:	W_k 의 항목집합의 개수

4.2 연관 규칙의 갱신

갱신문제는 다음과 같은 특징을 가진다. 첫째, 전체 데이타베이스에서 새로운 빈발 항목집합을 찾는 문제이다. 둘째, 기존의 빈발 항목집합은 갱신된 데이타베이스에서 소멸될 수 있는 가능성과 그 반대의 가능성을 가진다. 셋째, 새로운 빈발 항목집합을 찾기 위해 기존 데이타베이스를 포함한 갱신된 데이타베이스의 모든 레코드를 모든 후보 항목집합들에 대해서 탐색해야 한다.

표 4.1은 FUP 앨고리듬의 수행 과정을 전체 데이타베이스의 빈발 항목집합을 찾는 단계별로 도시한 것이다. 여기서 $|db_k|$, $|DB_k|$, $|Cut_k|$, $|Win_k|$는 각각 k단계에서 추가되는 데이타베이스의 크기, 기존 데이타베이스의 크기, 지지도를 만족하지 않아 전지되는 후보 항목집합의 개수, 기존 데이타베이스를 검색해야 하는 후보 항목집합의 개수를 나타낸다.

FUP에서의 갱신 과정을 분석한 결과로부터 갱신 문제에서 추가되는 후보 항목집합은 k가 증가할수록 점차 기존의 데이타베이스에서 추출된 빈발 항목집합에 근사하는 것으로 나타났다.

즉, k가 증가할수록 첫번째 단계에서 발견된 새로운 빈발 항목집합으로 인하여 그 다음 단계에서 추가되거나 삭제될 가능성이 있는 빈발 항목집합은 점차로 줄어들어 기존의 데이타베이스에서의 빈발 항목집합으로 근사하게 된다.

결국 추가되는 새로운 가능 빈발 항목집합의 후보 항목집합을 한꺼번에 생성해도 그 개수가 그렇지 않은 경우에 비해 상대적으로 많지 않고 이로 인해 데이타베이스를 한꺼번에 탐색해도 전체 수행능력에 영향을 주지 않는 것으로 나타났다.

k	1	2	3	4	5	6
$\|db_k\|$	1,000	1,000	311	127	31	24
$\|DB_k\|$	100,000	95,868	29,084	10,748	3,101	1,783
$\|Cut_k\|$	575	23	0	0	0	0
$\|Win_k\|$	166	2	0	0	0	0

표4.1 FUP에서의 갱신과정 분석

갱신문제에서 데이타베이스가 추가되어 기존의 빈발 항목집합을 갱신할 때 전체 데이타베이스에서의 빈발 항목집합을 탐색 하는 과정을 분석하였다.

이를 바탕으로 효율적인 검색이 가능하고 생성되거나 소멸되는 빈발 항목집합의 관계를 전체 데이타베이스에서의 빈발 항목집합과 연계하여 후보 항목집합의 개수를 현저히 감소시킬 수 있다.

4.3 SS를 적용한 갱신 앨고리듬

지금까지 설명한 개념을 구체화하는 앨고리듬은 그림 4.1과 같이 정리된다.
먼저 기존 데이타베이스 DB와 추가되는 데이타베이스 db를 검색하여 전체 데이타베이스의 빈발 1-항목집합 L_1^*와 해시 테이블 H_2를 찾는다. 그리고 L_1^*과 H_2에서 후보 2-항목집합C_2를 생성한다. 이때의 C_2는 L_1^*를 Apriori에서 생성한 것보다 훨씬 L_2^*에 근사하는 후보 항목집합이다.

여기까지는 DHP 앨고리듬과 동일하다. 그 다음 단계에서, 생성된 C_2를 기반으로 모든 단계의 후보 항목집합 C_3 , C_4 , ... , C_k을 생성한다. 이 때 C_{k+1}의 후보 항목집합은 C_k의 Apriori에서 제시한 것과 같은 방법을 따른다. C_2로부터 후보 항목집합을 생성하는 것은 데이타베이스의 검색 횟수를 최대한 줄이기 위함이다.

이렇게 생성된 후보 항목집합 C_k ($k \geq 2$)와 기존 데이타베이스의 빈발 항목집합 L_k ($k \geq 2$)에 대해 추가되는 데이타베이스 db를 검색하여 항목집합들의 지지 횟수를 계산한다. 그리고 L_k의 최소 지지도 s를 만족하는 모든 항목집합들을 갱신된 전체 데이타베이스 $DB \cup db$의 빈발 항목집합 L_k^*로 한다. 그리고 C_k에 포함된 항목집합들 중 기존 데이타베이스의 빈발 항목집합 L_k에 포함되지 않은 것들에 대해 지지 횟수 $s \times d$ 를 만족한다면 이를 W_k로 채택한다. 그리고 이 W_k가 공집합이 아니라면 기존 데이타베이스 DB를 읽어 W_k에 포함된 모든 항목집합의 지지 횟수를 계산한다. 그리고 이 항목집합들 중 지지 횟수가 $s \times (D+d)$ 이상인 것들을 L_k^*에 포함시킨다. 이렇게 구해진 L_k^*가 데이타베이스 db가 추가된 전체 데이터베이스$DB \cup db$의 최소 지지도를 만족하는 빈발 항목집합이 된다.

그림 4.2는 이러한 과정을 간략히 나타낸 것이다. 이를 보아 알 수 있듯이 제안하는 앨고리듬은 기존 데이타베이스 DB와 추가되는 데이타베이스 db를 각각 두 번씩 검색함으로써 모든 빈발 항목집합을 찾아내어 효율을 보장할 수 있는 것이다. 그리고 기존 데이타베이스의 빈발 항목집합 L_k에 대한 정보를 최대한 이용하여 W_k를 선정함으로써 기존 데이타베이스 DB를 가능한 한 적게 읽을 수 있도록 한다. 이는 W_k가 공집합일 경우 기존 데이타베이스 DB를 앨고리듬의 전체 수행 과정 중에서 단지 한번만 검색하기 때문이다.

Scan DB, db to find L_1^*, H_2

Generate C_2 based on L_1^*, H_2

$k = 2$

For ($k = 2$; $C_k \neq \varnothing$; k++) {

    Generate C_{k+1} from C_k by Apriori

}

Scan db to find supports of itemsets in L_k, C_k for $k \geq 2$

For ($k = 2$; $L_k \neq \varnothing$; k++) {

    Put all itemsets, that satisfy support, in L_k into L_k^*

    $C_k = C_k - L_k$

}

Generate W_k from C_k, L_k

If ($\sum_{k=2} |W_k| > 0$) {

    Scan DB to count supports of itemsets in W_k for $k \geq 2$

    put all itemsets, that satisfy support, in W_k into L_k^* for $k \geq 2$

}

그림 4.1 제안하는 앨고리듬

현실적으로 기존 데이타베이스 DB에 비해 추가되는 데이타베이스 db의 크기가 작은 경우에 이로 인한 새로운 빈발 항목집합의 출현을 기대하기는 매우 어렵다. 이는 앨고리듬의 수행
과정 중 기존 데이타베이스를 검색하여야 그 지지 횟수를 알 수 있는 W_k가 공집합이 되는 형태로 나타나고 DB를 한번만 읽을 가능성이 매우 높게 된다. 이러한 현상은 추가되는 데이타베이스 db의 크기가 기존 데이타베이스 DB의 크기보다 훨씬 작고 비슷한 형태의 연관 규칙을 가지고 있을 때 더욱 잘 나타나는 경향이 있다.

이처럼 제안하는 앨고리듬은 FUP 앨고리듬에서 제시한 기존 정보를 재사용하는 갱신 문제의 일반적인 해법을 기반으로 갱신 문제의 특성을 분석하여 연관 규칙의 갱신에 SS 앨고리듬의 후보 생성 방법을 적용시킴으로써 최대한의 효율 향상을 가져오게 할 수 있다.

5. 실험 및 결과

제안한 앨고리듬의 성능평가를 위해 많은 실험을 수행하였다. 프로그래밍 언어는 JAVA이며 컴파일러는 Microsoft사의 Visual J++ 1.1을 기계는 CPU 166MHz, 주메모리 32MB를 사용하였다.

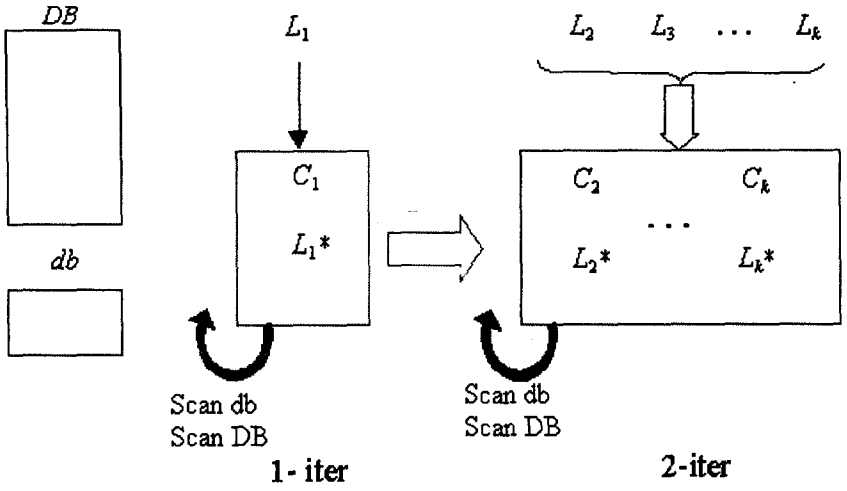

그림 4.2 제안하는 앨고리듬의 과정 도해

5.1 실험 데이타의 생성

본 연구에서 실험 데이타를 생성하기 위해 사용한 방법은 Park 등[4]과 Agrawal 등[7]에 제시된 것과 같다. 표 5.1은 본 실험에서 사용된 파라미터들의 목록이다.

D	트랜잭션들의 개수
d	추가된 트랜잭션들의 개수
|T|	트랜잭션의 항목들의 평균 개수
|I|	최대 잠재적인 빈발 항목집합들의 평균 크기(개수)
|L|	최대 잠재적인 빈발 항목집합들의 개수
N	항목들의 개수

표 5.1 실험 데이타 생성에 사용된 파라미터 목록

각 트랜잭션은 일련의 잠재적인(potential) 빈발 항목집합들로 구성되며 여기서 각 항목들은 항목집합 L에서 선택된다. L에 있는 각각의 잠재적인 빈발 항목집합들의 최대 크기는 평균값이 | I |인 포아송 분포로부터 결정된다.

L의 항목집합들은 다음과 같이 만들어진다. 첫번째 항목집합에 있는 항목들은 N개의 항목들로부터 임의의 방법으로 선택되어진다.

이어지는 항목집합들에서 공통의 항목을 갖기위해, 항목들의 일부가 앞에서 만들어진 항목집합으로부터 선택되어지며 그밖의 다른 항목들은 임의로 선택되어진다. 그런 일부 항목들의 선택은 상관 계수에 의해서 결정되는데 이는 평균값이 0.5인 지수. 분포로부터 선택되어진다.

Agrawal 등[7]에서와 같이, 빈발 항목집합들의 모든 항목들이 항상 한꺼번에 선택되지 않는 현상을 구현하기 위해, 트랜잭션을 만드는 동안 몇 개의 항목들을 탈락시키는데 이런 항목들의 개수는 탈락 수준(corruption level)을 정하는 분포 함수에 따른다. 각 트랜잭션은 <트랜잭션 식별자, 항목들의 수, 항목들>의 형식으로 하나의 파일시스템에 저장된다. 그림 5.1은 실험 데이타 생성의 과정을 개략적으로 보여준다.

실험에 사용하는 해시 테이블의 크기는 생성되는 C_2의 양에 영향을 준다. C_2가 L_2에 비해서 너무 커지면 다음 단계의 후보 항목 집합들의 크기가 급격하게 증가되어 전체 수행 성능이 나빠지게 된다. 이런 현상을 방지하기 위해 해시 테이블의 크기는 Park 등[4]에서 제시된 것과 같이 대략 2^{19}개의 버켓으로 구성되게 하였다.

실험에서 사용된 파라미터의 값은 D = m (천단위), d = n(천단위), N = 1000, | L | = 2000, | T | = 4, | I | = 10을 사용하였다[6]. 예를 들어, 데이타베이스 T10.I4.D100.d1은 각각 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이타베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이타베이스의 트랜잭션들의 개수가 1,000임을 의미한다.

5.2 실험결과

실험은 FUP와의 비교를 위해 Cheung 등[6]에서 수행한 것과 같은 방법을 따랐다. 상대시간은 FUP에서의 수행시간을 100으로 보았을 때의 제안하는 앨고리듬에서의 수행시간이며 {(제안하는 앨고리듬의 수행시간 / FUP의 수행시간) × 100}으로 계산된다.

그림 5.2는 데이타베이스 T10.I4.D100.d1에 대해서 최소 지지도를 0.5%에서 4%까지 단계적으로 변화 시켰을 때의 상대시간을 표시한 것이다. 그림에서 보여지듯이 제안하는 앨고리듬

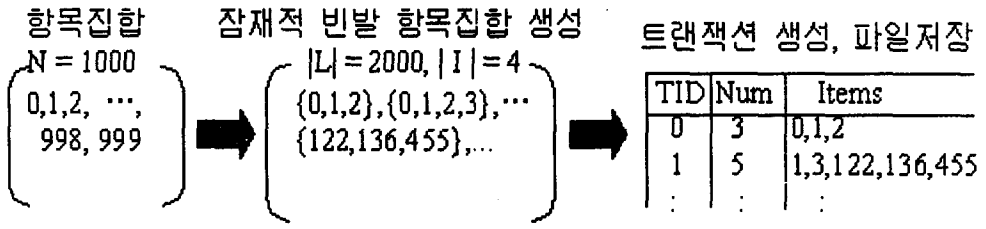

그림 5.1 실험 데이타 생성 과정

서 일정한 성능 개선이 이루어지고 있음을 알 수 있다.

그림 5.3는 기존의 데이타베이스의 크기는 고정시키고 추가되는 데이타베이스의 크기를 변화 시켰을 때의 수행시간을 보여준다. 일반적으로 추가되는 데이타베이스의 크기가 커지면 갱신에 수행되는 시간도 증가한다. 추가되는 데이타베이스의 크기를 10K, 50K, 100K로 증가 시키고 각각의 경우에 최소 지지도 1%와 2%에 대한 수행시간을 비교하였다. 그림에서 보여지듯이 추가되는 데이타베이스의 크기가 커지더라도 일정한 성능 개선이 이루어짐을 알 수 있다.

마지막으로 그림 5.4는 전체 데이타베이스의 크기를 매우 크게 증가시켰을 때의 결과를 보여준다. 처음에 수행한 실험에서 사용된 데이타베이스의 크기를 10배 확대시킨 데이터베이스 T10.I4.D1000.d10은 110만건의 트랜잭션을 포함하며 이를 파일 시스템에 저장했을 때 약 48메가바이트의 용량을 차지한다. 그림에서 보여지듯이 일정한 성능 개선이 이루어짐을 알 수 있으며 이는 제안하는 앨고리듬이 매우 큰 대용량의 데이타베이스에 대해서도 효율적인 갱신을 가

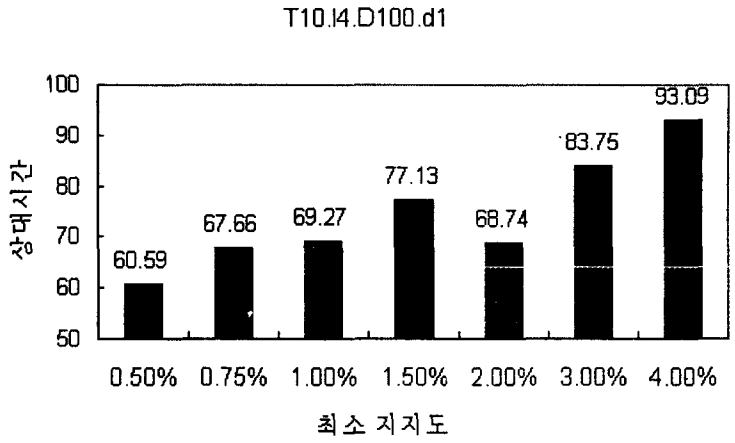

그림 5.2 최소 지지도의 변화에 따른 수행시간 비교

능하게 해준다는 것을 보여준다.

이상의 실험결과에서 보여주듯이 제안한 앨고리듬이 FUP 앨고리듬에 비해서 수행시간을 향상시켰음을 알 수 있다. 제안한 앨고리듬은 대용량의 데이타베이스에서 갱신이 발생할 때 빈발 항목집합들을 찾는 효율적인 방법이라고 할 수 있다.

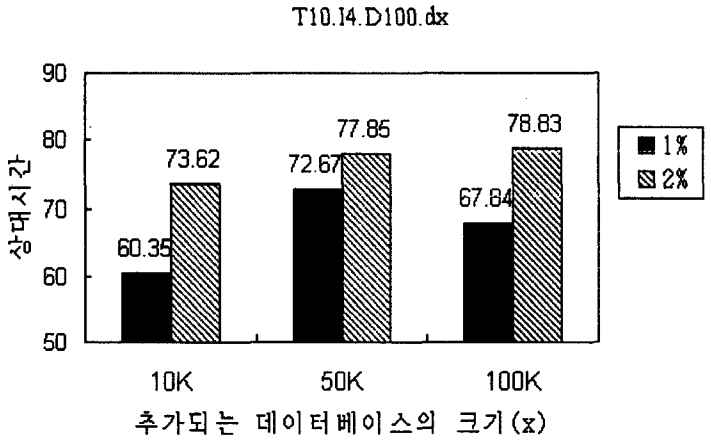

그림 5.3 추가되는 데이타베이스 크기변화에 따른 수행시간 비교

6. 결론 및 추후 연구과제

대용량의 데이타베이스에서 항목들간의 연관 규칙을 발견하는 문제는 기존의 데이타베이스에 새로운 트랜잭션이 추가됨에 따라 새로운 연관 규칙이 생성되거나 기존의 연관 규칙이 소멸될 수 있기 때문에 대용량 데이타베이스에서 연관 규칙을 갱신하는 효율적 앨고리듬을 제

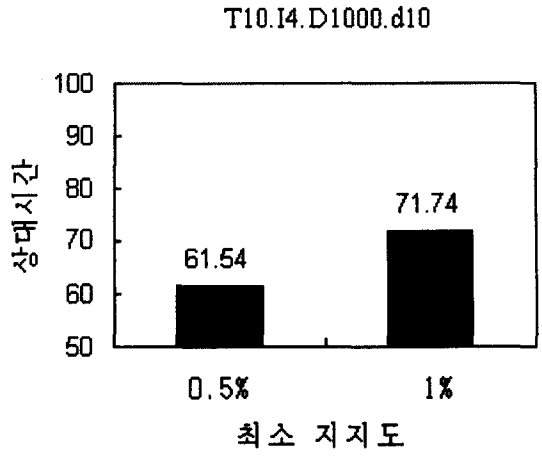

그림 5.4 데이타베이스 크기를 증가시킨 수행시간 비교

시하였다.

초기의 후보 집합 생성시에 2-후보 항목집합을 기반으로 모든 단계의 후보 항목집합을 생성하고 추가된 데이타베이스와 기존 데이타베이스를 한번씩 검색하여 모든 빈발 항목집합을 구하게 된다. 실험결과 제안하는 앨고리듬이 FUP의 성능보다 우수하였다.

대용량 데이타베이스에서 연관 규칙을 갱신하는 문제는 매우 중요하며 이에 대한 지속적인 연구가 필요하다. 추후 연구과제로 후보 항목집합 크기가 급격히 증가하는 경우를 고려한 연구가 기대된다.

참 고 문 헌

1. Adriaans, P. and D. Zantinge, *Data Mining,* Addison-Wesley, England, 1996.
2. Chen, M. S., J. Han, and P. S. Yu, "Data Mining:An Overview from Database Perspective," *IEEE Transaction on Knowledge and Data Engineering*, Vol.8, No.6, December, 1996.
3. Agrawal, R., T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proceedings of the 1993 ACM SIGMOD Conference*, May, 1993.
4. Park, J. S., M. S. Chen, and P. S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules," *Proceedings of ACM SIGMOD International Conference on Management of Data*, May, 1995.
5. Chen, M. S., J. S. Park, and P. S. Yu, "Data Mining for Path Traversal Patterns in a Web Environment", *Proceedings of the 16th International Conference on Distributed Computing Systems*, May, 1996.
6. Cheung, D. W., J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of Discovered Rules in Large Databases: An Incremental Updating Technique," *Proceedings of 12th International Conference on Data Engineering*, February, 1996.
7. Agrawal, R. and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proceedings of the 20th VLDB Conference*, 1994.