데이터 마이닝에서 기존의 연관규칙을 갱신하는 효율적인 앨고리듬
- An Efficient Algorithm for Updating Discovered Association Rules in Data Mining -

김 동 필*
Kim, Dong-Pil
지 영 근**
G, Young-Gun
황 종 원**
Hwang, Jong-Won
강 맹 규***
Kang, Maing-Kyu

ABSTRACT

This study suggests an efficient algorithm for updating discovered association rules in large database, 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.

FUP and DMI update efficiently strong association rules in the whole updated database reusing the information of the old large item-sets. Moreover, these algorithms use a pruning technique for reducing the database size in the update process.

This study updates strong association rules efficiently in the whole updated database reusing the information of the old large item-sets. An updating algorithm that is suggested in this study generates the whole candidate item-sets at once in an incremental database in view of the fact that it is difficult to find the new set of large item-sets in the whole updated database after an incremental database is added to the original database. This method of generating candidate item-sets is different from that of FUP and DMI.

After generating the whole candidate item-sets, if each item-set in the whole candidate item-sets is large at an incremental database, the original database is scanned and the support of each item-set in the whole candidate item-sets is updated. So, the whole large item-sets in the whole updated database is found out.

An updating algorithm that is suggested in this study does not use a pruning technique for reducing the database size in the update process. As a result, an updating algorithm that is suggested updates fast and efficiently discovered large item-sets.

---

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

1. 서론

1.1 연구의 배경과 목적

최근 들어 데이터를 생성하고 수집하는 능력은 급속히 증가하고 있다. 바코드 기법의 발전으로 바코드 판독기에 의한 유통, 판매, 생산의 데이터 수집, 기업업무 및 정부업무의 전산처리로 인하여 많은 양의 데이터가 사용자에게 제공되기 때문에 데이터베이스가 기업, 정부, 학교 등에서 사용되고 데이터베이스의 크기도 대용량화 되고 있다.

대용량 데이터베이스는 시장전략 수립, 수요예측을 위한 잠재적 사용가치가 큰 정보를 가지고 있기 때문에 데이터를 정보, 지식으로 변환할 수 있는 새로운 기술 및 도구가 필요하다. 이러한 요구로 태동된 데이터 마이닝(data mining)은 최근 들어 의사결정, 시장전략 수립, 수요예측, 의료진단, 공정관리, 상품진열 등 광범위한 분야에서 유용한 정보를 제공하므로 데이터베이스 분야에서 많은 주목을 받고 있다[6]. 데이터 마이닝은 대용량의 데이터베이스로부터 이러한 사용가치가 큰 정보를 발견하는 과정(knowledge discovery in databases)이라고 할 수 있다.

이러한 정보는 규칙의 종류에 따라서 분류규칙(classification rule), 연관규칙(association rule), 순차패턴(sequential pattern), 군집화(clustering) 등으로 분류할 수 있다. 연관규칙(association rule)은 데이터베이스에서 어떤 사건들이 함께 발생하거나, 또는 하나의 사건이 다른 사건을 암시하는 것과 같은 사건간의 상호관계를 나타낸다. 예를 들어, "편의점에서 빵을 구매하는 고객 중에서 90%의 고객이 우유도 함께 구매한다" 또는 "데이터베이스를 수강하는 학생 중 80%의 학생들은 데이터 마이닝도 수강한다"라는 정보는 연관규칙이다. Agrawal 등[2]은 이러한 연관규칙을 대용량 데이터베이스에서 발견하는 문제를 처음으로 제안하였다.

새로운 트랜잭션 데이터들이 데이터베이스에 지속적으로 누적되어 저장되므로 데이터베이스가 갱신됨에 따라 새로운 연관규칙이 발견되거나 기존의 연관규칙이 소멸될 수 있기 때문에 전체 데이터베이스에서 발견된 규칙을 오랜 시간에 걸쳐 유지, 보수가 필요하다.

데이터 마이닝에서 중요한 문제는 다음 두 문제이다. 첫째, 규칙이나 패턴을 찾는 효율적인 앨고리듬의 개발. 둘째, 발견된 규칙들을 유지, 관리하는 효율적인 앨고리듬의 개발. 첫 번째 문제에서는 특히, 연관규칙의 마이닝에서는 정량적 데이터, 일반화, 다중수준(multiple level), 병렬 처리, 분산 환경, 데이터 큐브(data cube) 등 각 경우를 고려한 앨고리듬들이 제시되었다[2-5, 8-12]. 그러나 두 번째 문제에 대한 연구는 거의 이루어지지 않았다[1, 7]. 두 번째 문제에서는 데이터베이스에서 연관규칙이 발견되어졌다고 가정하고 새로운 트랜잭션 데이터가 데이터베이스에 추가된 후부터 새로운 연관규칙을 발견하거나 기존의 연관규칙을 유지, 보수하는 것이다. 두 번째 문제 즉, 대용량 데이터베이스에서 연관규칙을 갱신하는 효율적인 앨고리듬에 관한 연구는 매우 중요하다. 본 연구에서는 두 번째 문제를 연구한다.

본 연구에서는 첫 번째 문제 대용량의 데이터베이스에서 연관규칙을 발견하는 기존연구와, 두 번째 문제 기존의 연관규칙을 갱신하는 앨고리듬인 FUP[7]와, DMI[1]에 대해 서술한 후 제안하는 갱신 앨고리듬을 서술한다.

1.2 기존연구

1.2.1 연관규칙의 탐사

$I = \{i_1, i_2, ..., i_m\}$는 항목(item)이라 불리는 문자들의 집합이다. D는 트랜잭션들의 집합이고 각 트랜잭션 T는 $T \subseteq I$인 항목들의 집합이다. 각 트랜잭션은 식별자(TID)를 가지고 있다. X는 항목들의 집합(item-set)이다. 트랜잭션 T가 X를 포함한다고 하면, $X \subseteq T$이다. 연관규칙은 $X \Rightarrow Y$의 형식으로 나타내고, 여기서, $X \subset I$, $Y \subset I$, 및 $X \cap Y = \varnothing$이다. $X \Rightarrow Y$는 신뢰도(confidence) c를 가지고 있다라는 조건으로 트랜잭션 집합인 D에 속하는데, 이는 X를 포함하는 D의 트랜

잭션들 중 c%가 Y 또한 포함하고 있음을 의미한다. $X \Rightarrow Y$는 트랜잭션 집합 D에서 지지도(support) s를 갖는데 이는 D의 트랜잭션들 중 s%는 $X \cup Y$를 포함하고 있음을 의미한다.

연관규칙을 찾는 문제는 연관규칙의 신뢰도와 지지도가 최소신뢰도와 최소지지도보다 큰 모든 연관규칙을 찾는 것이다. 이러한 항목집합을 빈발 항목집합(large item-set 또는 frequent item-set)이라 한다. k개의 항목들로 이루어진 빈발 항목집합을 빈발 k-항목집합이라 한다. 빈발 k-항목집합들의 집합을 L_k라 하고 이를 위한 후보 k-항목집합들을 C_k라 한다.

연관규칙을 발견하는 과정은 다음의 두 하위문제로 구성된다[2, 3, 11].

(1)주어진 최소지지도 이상의 트랜잭션들의 지지를 갖는 항목집합들의 모든 집합인 빈발 항목집합들을 찾는다.

(2)빈발 항목집합들을 이용하여 데이터베이스에 대한 연관규칙을 찾는다.

연관규칙을 발견하는 과정 중에서 전체 수행속도에 영향을 미치는 것은 첫 번째 과정이다. 두 번째 과정에서는 단지 빈발 항목집합을 통해 연관규칙을 찾는다. 그러므로 연관규칙을 탐사는 첫 번째 단계에 즉, 빈발 항목집합을 찾는 것에 초점을 두고 있다.

1.2.2 연관규칙의 갱신

연관규칙을 갱신하는 방법 중의 하나는 갱신된 전체 데이터베이스에 대해 연관규칙 탐사 앨고리듬을 재수행(re-running)하는 방법이다. 그러나 이 방법은 기존의 빈발 항목집합을 찾기 위해 수행되었던 계산이 소용없게 되고, 모든 빈발 항목집합을 처음부터 다시 계산되는 단점이 있다. 그러므로 연관규칙을 갱신하는 효율적인 앨고리듬이 필요하다.

L은 데이터베이스에서의 빈발 항목집합, s는 최소지지도, D는 데이터베이스 DB의 트랜잭션의 개수이다. $X \in L$ 인 항목집합 X를 포함하는 데이터베이스의 트랜잭션의 개수는 항목집합 X의 $X_{.support}$ 또는 지지도(support count)이다. 데이터베이스가 갱신된 후, 새로운 트랜잭션 개수 d로 구성된 데이터베이스 db가 데이터베이스 DB에 삽입되었을 때, 같은 최소지지도 s하에서 갱신된 전체 데이터베이스 $DB \cup db$의 항목집합 X는 $X_{.support} \geq s \times (D+d)$이면 빈발 항목집합이다. 연관규칙의 갱신 문제의 핵심은 $DB \cup db$에서의 빈발 항목집합 L'를 찾는 것이다.

연관규칙의 갱신 문제는 다음과 같은 특징을 갖는다[7].

(1)갱신문제는 새로운 빈발 항목집합을 찾는 것으로 줄여질 수 있다. 새로운 연관규칙은 새로운 빈발 항목집합으로부터 만들어 질 수 있다.

(2)기존의 빈발 항목집합이 전체 데이터베이스에서는 빈발 항목집합이 안될 가능성도 있다.

(3)기존에 빈발 항목집합이 아니었던 항목집합이 전체 데이터베이스에서는 빈발 항목집합이 될 수 있다.

(4)새로운 빈발 항목집합을 찾기 위해서 모든 후보 항목집합에 대해 갱신된 데이터베이스 전체를 조사해야 한다.

따라서 연관규칙의 갱신 문제에 있어서의 핵심은 기존의 빈발 항목집합의 정보를 재사용하는 것이다.

1.2.3 연관규칙 탐사 앨고리듬의 기존연구

(1) Apriori 앨고리듬

Apriori 앨고리듬[2]은 각각의 반복수행에서 빈발 항목집합들에 대한 후보 항목집합을 생성하고 각 후보 항목집합의 지지도를 계산한 후 주어진 최소지지도 이상의 빈발 항목집합들을 결정하게 된다.

Apriori는 첫 번째 반복수행에서 각 항목의 지지도를 계산하기 위해 데이터베이스의 모든 트랜잭션을 검색한다. 트랜잭션 데이터베이스가 그림 1.1과 같이 주어졌을 때 데이터베이스 D

를 검색하여 C_1를 결정하고 최소지지도가 2(s = 40%)라 하면 지지도가 최소지지도 이상을 가진 C_1로부터 L_1을 결정한다. 두 번째 반복수행에서는 L_2를 찾을 때 빈발 항목집합의 모든 부분집합들의 지지도가 최소지지도 이상이어야 한다는 사실을 고려해 $L_1 * L_1$를 사용해 C_2를 생성한다. 여기서 *는 접속(concatenation)연산이다. C_2의 크기는 $\binom{|L_1|}{2}$ 이다.

다음으로 데이터베이스 D의 4개의 트랜잭션을 검색하여 후보 항목집합 C_2의 각 항목집합의 지지도를 계산하고 지지도가 최소지지도 이상을 가지는 C_2로부터 L_2를 구한다. 후보 3-항목집합 C_2는 L_2로부터 생성된다. {BC}와 {BE}같이 첫 번째 항목이 같은 빈발 2-항목집합 L_2를 찾고 이들 항목집합들의 두 번째 항목으로 이루어진 {CE}가 L_2의 원소인지를 확인한다. {CE}자체가 빈발 항목집합이므로 {BCE}의 모든 2-항목집합이 빈발 항목집합이므로 {BCE}는 후보 3-항목집합 C_3이 된다. L_2로부터 다른 후보 3-항목집합은 생성되지 않는다. 다음으로 빈발 3-항목집합 L_3을 찾기 위해 모든 트랜잭션을 검색한다. L_3로부터 후보 4-항목집합 C_4가 생성되지 않으므로 빈발 항목집합을 찾는 과정을 중단한다.

그림 1.2는 Apriori의 후보 항목집합 및 빈발 항목집합들의 생성 예를 나타낸다.

(2) DHP 앨고리듬

DHP(direct hashing and pruning)[11]는 다음과 같은 두 가지 특징을 가지고 있다. 후보 항목집합들의 효율적인 생성과 트랜잭션 데이터베이스 크기의 효과적인 감소이다.

C_k에 많은 항목집합들이 있을수록 L_k를 결정하기 위한 많은 연산비용이 필요로 하기 때문에 후보 항목집합의 크기가 작을수록 적은 연산비용이 든다.

DHP는 Apriori와 같이 L_{k-1}로부터 C_k를 생성한다. 그러나 [11]에서 설명된 해시기법은 DHP로 하여금 후보 항목집합들의 생성을 매우 효과적이게 한다. 특히 빈발 2-항목집합 L_2를 위한 C_2의 크기가 기존 방법에 비해 더 적기 때문에 전체 프로세스의 수행상의 병목을 개선한다. 또한 DHP는 트랜잭션 데이터베이스 크기를 점차적으로 감소시키기 위해 효과적인 전지(pruning)기법을 사용한다. DHP에 의한 더 적은 수의 후보 집합의 생성은 반복수행의 초기 단계에서는 트랜잭션 데이터베이스를 효과적으로 감소할 수 있게 하며, 그럼으로써 연산비용을 줄여 준다. 연관규칙의 특성을 이용함으로써 트랜잭션의 수뿐만 아니라 각 트랜잭션 항목의 수 또한 감소시킬 수 있음을 보여 주었다.

(3) SS 앨고리듬

SS(selective scan)[5]는 DHP에서 생성된 C_2를 기반으로 모든 단계의 후보 k-항목집합을 한꺼번에 생성하는 방법을 이용하여 효율적인 데이터베이스 검색을 가능하게 한다. DHP에서와 같이 L_1과 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. 이 생성방법이 기존의 Apriori와 DHP의 방법과 다른 점이다. 이렇게 함으로써 두 번의 데이터베이스 탐색으로 모든 빈발 항목집합 L_k를 구할 수 있다.

1.2.4 연관규칙 갱신 앨고리듬의 기존연구

(1) FUP 앨고리듬

Cheung 등[7]은 갱신 문제의 특성을 반영하는 갱신 앨고리듬인 FUP(fast update)를 제안하였다. FUP는 이미 만들어진 빈발 항목집합을 재사용한다. 새로운 빈발 항목을 찾을 때 후보 항목집합이 현저하게 전지될 수 있게 한다.

FUP에서 제시한 과정은 다음과 같다. 최소지지도 s, 트랜잭션 개수 D인 데이터베이스 DB에서 빈발 항목집합 L의 원소 $X \in L$에 대해 지지도를 $X_{.support}$라 하자. 갱신 작업이 이루어

진 후, 새로운 트랜잭션 개수 d로 구성된 데이터베이스 db가 기존의 데이터베이스 DB에 추가된다. 최소지지도 s하에서 갱신된 전체 데이터베이스 $DB \cup db$의 항목집합 X는 $X.support \geq$

식별자	항목
100	A C D
200	B C E
300	A B C E
400	B E

그림 1.1 데이터 마이닝을 위한 예제 트랜잭션 데이터베이스 D

C_1

항목집합	지지도
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

검색 D →

L_1

항목집합	지지도
{A}	2
{B}	3
{C}	3
{E}	3

C_2

항목집합
{A B}
{A C}
{A E}
{B C}
{B E}
{C E}

검색 D →

C_2

항목집합	지지도
{A B}	1
{A C}	2
{A E}	1
{B C}	2
{B E}	3
{C E}	2

L_2

항목집합	지지도
{A C}	2
{B C}	2
{B E}	3
{C E}	2

C_3

항목집합
{B C E}

검색 D →

C_3

항목집합	지지도
{B C E}	2

L_3

항목집합	지지도
{B C E}	2

그림 1.2 후보 항목집합과 빈발 항목집합들의 생성

$s \times (D+d)$이면 빈발 항목집합이다. 즉, 갱신된 전체 데이터베이스 $DB \cup db$에서 빈발 항목집합 L'를 찾아낸다. Apriori나 DHP를 재수행(re-running)하는 것보다 2~16배 빠른 것으로 밝혀졌다.

(2) DMI 앨고리듬

Lee 등[1]이 제안한 갱신 앨고리듬 DMI는 SS에서 제시한 후보 k-항목집합을 생성하는 방법과 FUP에서 제시한 기존 정보를 재사용하는 방법을 이용하였다.

후보 항목집합을 생성할 때 기존 데이터베이스 DB와 추가된 데이터베이스 db 즉, $DB \cup db$를 검색한 후 $DB \cup db$에서의 빈발 1-항목집합 L_1을 찾고 해시테이블 H_2에 기반하여 L'_2에 근사하는 C_2를 찾는다. C_2로부터 모든 단계의 C_k를 생성한 후 전체 데이터베이스 $DB \cup db$에서 빈발 항목집합 L'_k을 찾는다. 따라서 DMI는 추가된 데이터베이스를 두 번 검색하고, 기존 데이터베이스를 최대 두 번 검색하여 $DB \cup db$에서 빈발 항목집합 L'_k을 찾는다. DMI는 FUP보다 수행시간을 향상시켰다.

2. 제안하는 갱신 앨고리듬

2.1 기호 설명

본 연구에서 사용하는 기호는 다음과 같다.

L_k: 기존 데이터베이스의 빈발 k-항목집합
L'_k: 갱신된 전체 데이터베이스의 빈발 k-항목집합
$L_{k(db)}$: 추가되는 데이터베이스의 빈발 k-항목집합
C_k: 후보 k-항목집합
$C_{k(db)}$: 추가되는 데이터베이스의 후보 k-항목집합
s: 최소지지도
DB: 기존 데이터베이스
db: 추가되는 데이터베이스
D: 기존 데이터베이스의 트랜잭션의 개수
d: 추가되는 데이터베이스의 트랜잭션의 개수
$X.support$: 항목집합 X의 지지횟수(지지도)
$X.support(db)$: db에서 항목집합 X의 지지횟수(지지도)
W_k: db에서 지지회수가 $s \times d$ 이상인 후보 항목집합들의 집합

2.2 제안하는 앨고리듬

본 연구에서 제안하는 갱신 앨고리듬은 SS에서 제시한 후보 k-항목집합을 생성하는 방법과 FUP에서 제시한 기존 정보를 재사용하는 방법을 이용함으로써 데이터베이스가 추가될 때 효율적인 갱신이 이루어지도록 연관규칙을 탐사한다.

DMI는 후보 항목집합을 생성할 때 기존 데이터베이스 DB와 추가된 데이터베이스 db 즉, $DB \cup db$를 검색한 후 $DB \cup db$에서의 빈발 1-항목집합 L_1을 찾고 해시테이블 H_2에 기반하여 C_2를 찾고 C_2로부터 모든 단계의 C_k를 생성하였다.

FUP는 연관규칙을 갱신하는 각 반복수행에서, DMI는 첫 번째 반복수행에서 다음 반복수행에서의 데이터베이스의 검색을 줄이기 위해 DHP와 같이 데이터베이스를 전지한다. 그러나 본 연구에서는 데이터베이스를 전지하지 않는다. 이점이 FUP, DMI와 다른 점이다. 실험결과, 첫 번째 반복수행에서 데이터베이스를 전지하는 것은 수행속도를 증가시켰다.

본 연구에서는 추가된 데이터베이스 db만을 검색하여 db에서의 빈발 1-항목집합 $L'_{1(db)}$을

찾고 H_2에 기반하여 $C_{2(db)}$를 찾고 $C_{2(db)}$로부터 모든 단계의 $C_{k(db)}$를 생성한다. 이점이 FUP, DMI와 다른 점이다. 즉, C_k를 생성할 때 $DB \cup db$를 검색하지 않고 db만을 검색하여 $C_{k(db)}$를 생성하는 것이다. 이는 추가되는 데이터베이스로 인하여 갱신된 전체 데이터베이스에서 새로운 빈발 항목집합의 출현은 어렵기 때문이다. 따라서 DB검색시간을 줄일 수 있다.

이렇게 생성된 후보 항목집합 $C_{k(db)}(k \geq 2)$와 기존 데이터베이스의 빈발 항목집합 $L_k(k \geq 2)$에 대해 추가되는 데이터베이스 db를 검색하여 항목집합들의 지지도를 계산한다. $L_k(k \geq 2)$에 대해 추가되는 데이터베이스 db를 검색하는 이유는 전체 데이터베이스 $DB \cup db$의 L'_k로 채택될 수 있기 때문이다. 그리고 $L_k(k \geq 1)$의 지지도가 $X.support \geq s \times (D+d)$이면 항목집합을 전체 데이터베이스 $DB \cup db$의 L'_k로 채택하다.

다음으로 $L'_{1(db)}$를 $W_k(k=1)$로 $C_{k(db)}$에 포함된 항목집합들 중 기존 데이터베이스의 빈발 항목집합 L_k에 포함되지 않은 항목집합들의 지지도가 $X.support_{(db)} \geq s \times d$인 후보 항목집합들을 $W_k(k \geq 2)$으로 채택한다. 그리고 $W_k \neq \varnothing$이면 기존 데이터베이스 DB를 검색하여 W_k에 포함된 모든 항목집합에 대해 지지도를 계산한다. 이는 추가되는 데이터베이스 db에서 빈발 항목집합만이 즉, db에서의 후보 항목집합들 중 항목집합의 지지도가 $X.support_{(db)} \geq s \times d$인 후보 항목집합만이 $DB \cup db$에서 빈발 항목집합이 될 수 있다는 갱신 문제의 특성을 고려한 것이다. 그리고 이 항목집합들의 지지도가 $X.support \geq s \times (D+d)$이면 항목집합을 L'_k에 포함시킨다. 이렇게 구해진 L'_k가 데이터베이스 db가 추가된 전체 데이터베이스 $DB \cup db$의 최소지지도를 만족하는 빈발 항목집합이 된다.

제안하는 앨고리듬은 다음과 같은 특징을 가진다.

(1)첫 번째 단계에서 데이터베이스 db만을 검색하여 $L_{1(db)}$과 $C_{2(db)}$를 찾아낸다. 여기서 $C_{2(db)}$는 해시테이블 H_2에 기반하여 생성된다. 따라서 데이터베이스 검색을 최소화할 수 있다. DMI 앨고리듬에서는 전체 데이터베이스 $DB \cup db$를 검색하여 C_2를 찾는다.

(2)두 번째 단계에서 $C_{2(db)}$에 기반하여 모든 단계의 $C_{3(db)}$, $C_{4(db)}$, ..., $C_{k(db)}$의 후보 항목집합을 생성한다. 이 점이 기존의 갱신 앨고리듬의 후보 항목집합 생성방법과 다른 점이다. 데이터베이스 db에서 전체 후보 항목집합을 생성하는 이유는 추가되는 데이터베이스로 인하여 갱신된 전체 데이터베이스에서 새로운 빈발 항목집합의 출현은 어렵기 때문이다.

(3)세 번째 단계에서 추가된 데이터베이스 db를 검색하여 후보 항목집합에 있는 항목집합의 지지도가 $X.support_{(db)} \geq s \times d$인 후보 항목집합과 $L_{1(db)}$에 대해서만 W_k로 채택하여, 기존 데이터베이스 DB를 검색한다. 만약 W_k가 공집합이라면 기존 데이터베이스 DB를 검색하지 않는다. DMI 앨고리듬에서도 W_k가 공집합이라면 기존 데이터베이스 DB를 검색하지 않는다.

(4)이미 구해진 L_k를 최대한 이용한다. 이는 갱신문제에서 전체 데이터베이스를 재검색 하지 않고 부분적인 검색으로 빈발 항목집합을 효율적으로 찾아내는 중요한 이점이다.

그림 2.1에서 보인 것과 같이 제안하는 앨고리듬은 추가되는 데이터베이스 db를 두 번, 기존 데이터베이스를 최대 한번만 검색하여 모든 빈발 항목집합을 찾아내므로 효율을 보장할 수 있는 것이다. 그리고 기존 데이터베이스의 빈발 항목집합 L_k를 최대한 이용하여 W_k를 선정함으로써 기존 데이터베이스 DB를 가능한 한 적게 읽을 수 있도록 한다. 이는 W_k가 공집합인 경우 기존 데이터베이스 DB를 앨고리듬의 전체 수행 과정 중에서 한번도 검색하지 않기 때문이다. DMI 앨고리듬은 전체 수행 과정에서 기존 데이터베이스 DB를 한번은 검색한다. 이것이 제안하는 앨고리듬의 장점이다. 즉, 제안하는 앨고리듬은 앨고리듬 수행과정에서 디스크 액세스를 최소화하여 효율을 보장할 수 있는 것이다. 제안하는 앨고리듬의 과정과 code는 각각 그림 2.1과 그림 2.2와 같다.

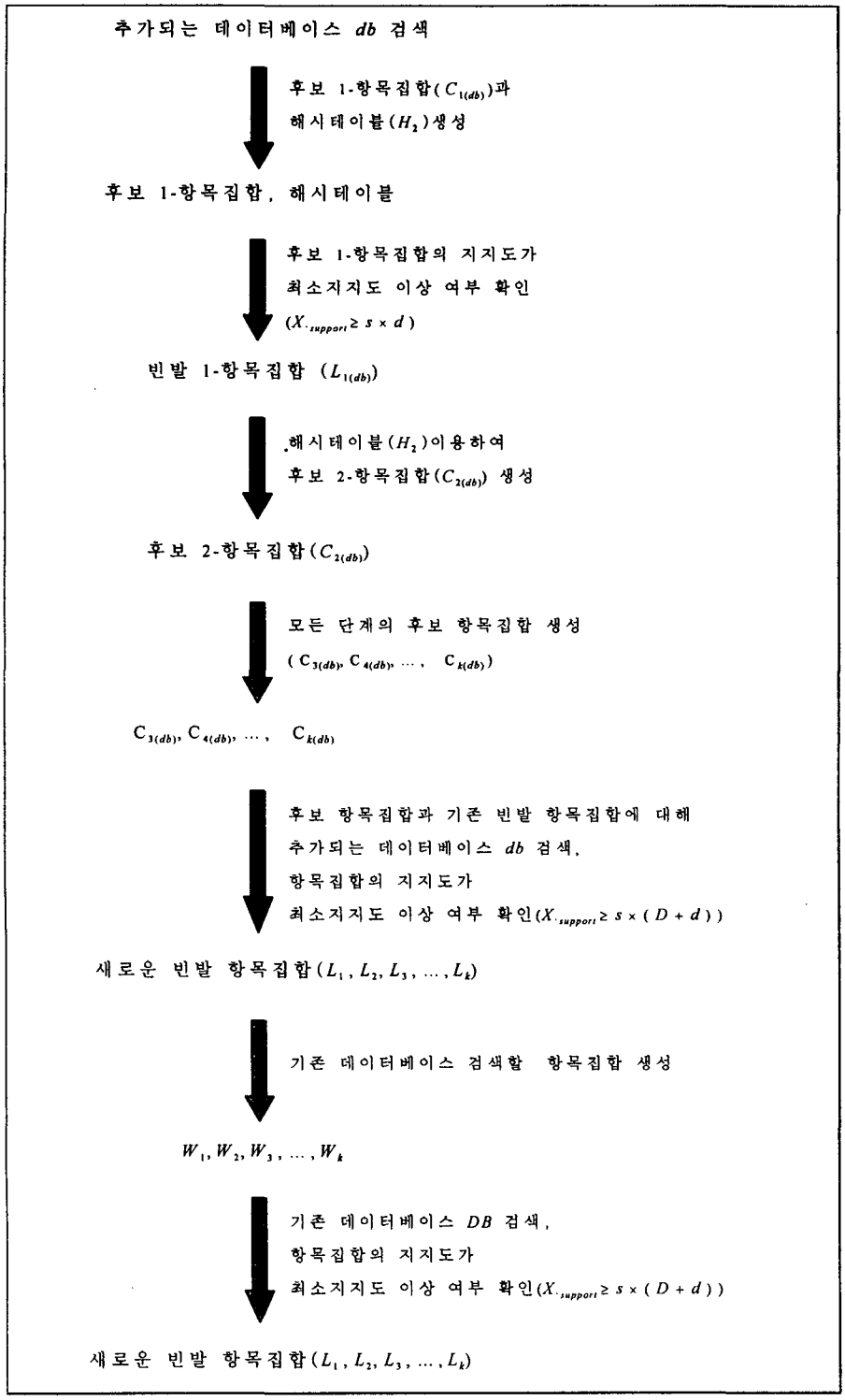

그림 2.1 제안하는 앨고리듬의 과정

```
Scan db to find L'(db);
Generate C2(db) based on L'(db), H2;
k = 2;
While( Ck(db) ≠ ∅ ) {
    Generate Ck+1(db) from Ck(db) by Apriori;
}
L'1(db) = L'1(db) - L1;
For ( k = 2; Ck(db) ≠ ∅; k++ ) {
    Ck(db) = Ck(db) - Lk;
}
Scan db to find supports of itemsets in Lk, Ck(db) for k ≥ 2;
For ( k = 1; Lk ≠ ∅; k++ ) {
    For_all X ∈ Lk
        If X.support ≥ s × (D+d)
            Then L'k = L'k ∪ {X};
}
Generate Wk (k ≥ 1) from L'1(db), Ck(db) for k ≥ 2;
If ( Wk ≠ ∅ ) {
    Scan DB to count supports of itemsets in Wk;
        For_all X ∈ Wk
            If X.support ≥ s × (D+d)
                Then L'k = L'k ∪ {X};
}
```

그림 2.2 제안하는 앨고리듬의 code

3. 실험과 결과

제안하는 앨고리듬의 성능평가를 위해 많은 실험을 수행하였다. Visual C++ 5.0을 사용하여 프로그램하였다. 실험은 CPU 166MHz, 메모리 32MB를 가진 컴퓨터에서 수행하였다.

3.1 실험 데이터의 생성

본 연구에서 실험 데이터를 생성하기 위해 사용한 방법은 [3]과 [11]에 제시된 것과 같다. 그림3.1은 본 실험에서 사용된 파라미터들의 목록이다.

각 트랜잭션은 일련의 잠재적인 빈발 항목집합들로 구성된다. 여기서 각 항목들은 항목집합 L에서 선택된다. L에 있는 각각의 잠재적인 빈발 항목집합들의 크기는 평균이 | I |인 포아송 분포로부터 결정된다.

L의 항목집합들은 다음과 같이 만들어진다. 첫 번째 항목집합에 있는 항목들은 N개의 항목들로부터 임의의 방법으로 선택되어진다. 이어지는 항목집합들에서 공통의 항목을 갖기 위해

항목들의 일부가 앞에서 만들어진 항목집합으로부터 선택되어지며 그 밖의 다른 항목들은 임의로 선택된다. 그런 일부 항목들의 선택은 상관계수(correlation level)에 의해 결정되는데 이는 평균이 0.5인 지수분포로부터 선택되어진다.

Apriori에서와 같이, 빈발 항목집합들의 모든 항목들이 항상 한꺼번에 선택되는 현상을 방지하기 위해 트랜잭션을 만드는 동안 몇 개의 항목들을 탈락시키는데 이런 항목들의 개수는 탈락수준(corruption level)을 정하는 분포함수에 따른다. 각 트랙잭션은 < 트랙잭션 식별자, 항목들의 수, 항목들>의 형식으로 하나의 파일시스템에 저장된다.

실험에 사용되는 해시테이블의 크기는 DHP에서 제시된 것과 같은 2^{19}개의 버켓으로 구성되게 하였다.

실험에서 사용된 파라미터의 값은 [7]에서와 같이 D = m(천 단위), d = n(천 단위), N = 1000, | L | = 2000, | T | = 10, | I | = 4 를 사용하였다. 예를 들어 데이터베이스 T10.I4.D100.d1은 각각 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000임을 의미한다.

D	데이터베이스 *DB* 에서의 트랜잭션들의 개수
d	추가된 데이터베이스 *db* 에서의 트랜잭션들의 개수
|T|	트랜잭션의 항목들의 평균개수
|I|	최대 잠재적인 빈발 항목집합들의 평균 개수
|L|	최대 잠재적인 빈발 항목집합들의 개수
N	항목들의 개수

그림 3.1 실험 데이터 생성에 사용된 파라미터 목록

3.2 실험결과

실험은 FUP와의 비교를 위해 [7]에서 수행한 것과 같은 방법을 따랐다. 상대시간은 FUP와 DMI에서의 수행시간을 100으로 보았을 때의 제안하는 앨고리듬에서의 시간이며 {(제안하는 앨고리듬의 수행시간(KDP) / FUP의 수행시간) × 100}과 {(제안하는 앨고리듬의 수행시간(KDP) / DMI의 수행시간) × 100}으로 계산된다.

그림 3.2는 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000인 데이터베이스 T10.I4.D100.d1에 대해서 최소지지도를 0.4%에서 6%까지 단계적으로 변화시켰을 때의 상대시간을 표시한 것이다. 일정한 성능 개선이 이루어짐을 알 수 있다.

그림 3.3은 기존의 데이터베이스의 크기는 고정시키고 추가되는 데이터베이스의 크기(x)를 변화시켰을 때의 수행시간을 보여준다. 일반적으로 추가되는 데이터베이스의 크기가 커지면 갱신에 수행되는 시간도 증가한다. 추가되는 데이터베이스의 크기를 10K, 50K, 100K로 증가시키고 각각의 경우에 최소지지도 0.5%, 1%, 2%에 대한 수행시간을 비교하였다. 추가되는 데이터베이스의 크기가 커지더라도 일정한 성능 개선이 이루어짐을 알 수 있다.

그림 3.4는 전체 데이터베이스의 크기를 매우 크게 증가시켰을 때의 결과를 보여준다. 처음 실험에서 사용된 데이터베이스의 크기를 10배 확대시킨 데이터베이스 T10.I4.D1000.d10은 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들

의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 1,000,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 10,000이다. 110만 건의 트랜잭션을 포함하며 파일 시스템에 저장했을 때 약 48메가 바이트의 용량을 차지한다. 그림에서 보여지듯이 일정한 성능 개선이 이루어짐을 알 수 있으며 이는 제안하는 앨고리듬이 매우 큰 대용량의 데이터베이스에 대해서도 효율적인 갱신을 가능하게 해준다는 것을 보여준다.

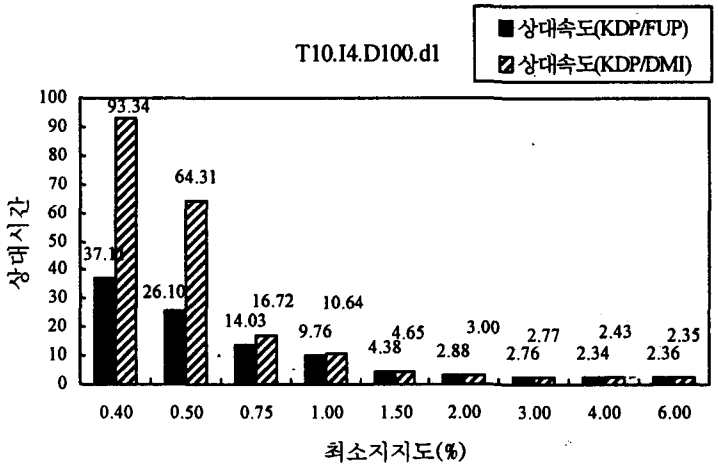

그림 3.2 최소지지도의 변화에 따른 수행시간 비교

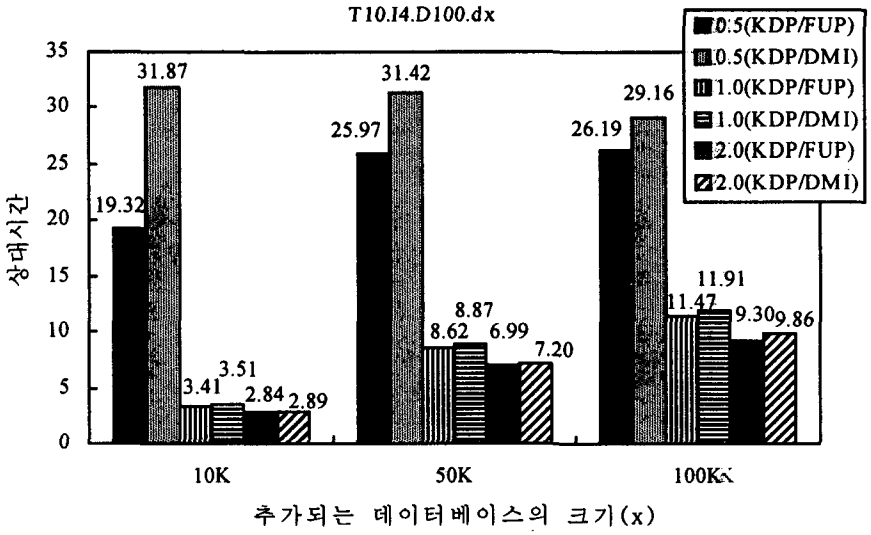

그림 3.3 추가되는 데이터베이스 크기변화에 따른 수행시간비교

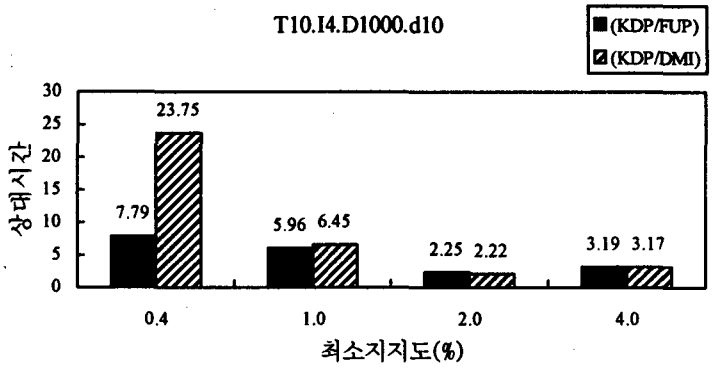

그림 3.4 데이터베이스 크기를 증가시킨 수행시간 비교

이상의 실험결과와 같이 제안한 앨고리듬은 FUP, DMI 앨고리듬보다 수행시간을 향상시켰음을 알 수 있다. 제안한 앨고리듬은 대용량의 데이터베이스에서 갱신이 발생할 때 빈발 항목집합들을 찾는 매우 효율적인 앨고리듬이라고 할 수 있다.

4. 결론

기존 데이터베이스에 새로운 트랜잭션이 추가됨에 따라 새로운 연관규칙이 발견되거나 기존의 연관규칙이 소멸될 수 있기 때문에 대용량 데이터베이스에서 연관규칙을 갱신하는 효율적 앨고리듬의 개발은 매우 중요하다.

제안하는 연관규칙 갱신 앨고리듬은 기존 갱신 앨고리듬과 같이 기 발견된 연관규칙에 대한 정보를 재사용한다.

특히, 본 연구에서는 후보 항목집합을 생성할 때 추가되는 데이터베이스로 인하여 전체 데이터베이스에서 새로운 빈발 항목집합의 출현이 어렵다는 점을 이용하여 추가되는 데이터베이스에서의 후보 2-항목집합을 기반으로 모든 단계의 후보 항목집합을 한번에 생성한다. 이것이 기존의 갱신 앨고리듬과 다른 점이다. 그리고 추가된 데이터베이스 검색한 후 최소지지도를 만족하는 후보 항목집합에 대해 기존의 데이터베이스를 검색하여 전체 데이터베이스에서 갱신된 연관규칙을 찾는다.

갱신 앨고리듬인 FUP는 연관규칙을 갱신하는 각 반복수행에서 다음 반복수행에서의 데이터베이스의 검색을 줄이기 위한 데이터베이스를 전지한다. 또한 갱신 앨고리듬인 DMI는 첫 번째 반복수행에서 데이터베이스를 전지한다. 그러나 제안하는 갱신 앨고리듬은 데이터베이스를 전지하지 않는다. 그리고 제안하는 앨고리듬은 앨고리듬 수행과정에서 추가되는 데이터베이스 db를 두 번, 기존 데이터베이스를 최대 한번만 검색하여 모든 빈발 항목집합을 찾아내므로 효율을 보장할 수 있다. 그러므로 제안하는 앨고리듬은 앨고리듬 수행과정에서 디스크 액세스를 최소화하여 연관규칙의 갱신작업을 빠르게 한다.

제안하는 앨고리듬은 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000인 데이터베이스 T10.I4.D100.d1에 대해서 최소지지도를 0.4%에서 6%까지 단계적으로 변화시키며 갱신작업을 수행할 때 기존 갱신 앨고리듬인 FUP와 DMI보다 평균 91%, 평균 85%의 속도향상을 가져왔다. 또한 추가되는 데이터베이스의 크기를 증가시켰을 때, 전체 데이터베이스의 크기를 증가시켰을 때에도 제안하는 갱신 앨고리듬은 기존 갱신 앨고리듬보다 속도향상을 가져왔다.

참 고 문 헌

[1]이동명, 지영근, 황종원, 강맹규, "데이터 마이닝에서 기존의 연관규칙을 갱신하는 앨고리듬 개발," 공업경영학회, 제 20 권, 제 43 집, pp. 265-276, 1997.

[2]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.

[3]Agrawal, R. and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proceedings of the 20th VLDB Conference*, September 1994.

[4]Agrawal, R. and J. C. Shafer, "Parallel Mining of Association Rules: Design, Implementation and Experience," IBM Research Report RJ 10004, January 1996.

[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]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.

[7]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.

[8]Cheung, D. W., J. Han, V. T. Ng, A. Fu, and Y. Fu, "A Fast Distributed Algorithm for Mining Association Rules," *Proceedings of 4th International Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[9]Han, J. and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," *Proceedings of 21th VLDB Conference*, pp. 420-431, September 1995.

[10]Kamber M., J. Han, and J. Y. Chiang, "Using Data Cubes for Metarule-Guided Mining of Multi-Dimensional Association Rules," Technical Report CS-TR 97-10, School of Computing Science, Simon Fraser University, May 1997.

[11]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.

[12]Srikant, R. and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 1-12, Montreal, Canada, June 1996.