DOI: 10.3745/KIPSTD.2011.18D.3.169

민감한 빈발 항목집합 숨기기 위한 확장 빈발 패턴 트리

이 단 영[†] · 안 형 근[†] · 고 재 진[††]

요 약

최근 기업 간 또는 기관 사이의 데이터 공유는 업무 협력을 위해서 필요한 사안이 되고 있다. 이 과정에서 기업이 데이터베이스를 계열회사에 공개했을 때 민감한 정보가 유출되는 문제점이 발행할 수도 있다. 이런 문제를 해결하기 위해서 민감한 정보를 데이터베이스로부터 숨기는 일이 필요하게 되었다. 민감한 정보를 숨기는 이전 연구들은 결과 데이터베이스의 품질을 유지하기 위해 다른 휴리스틱 알고리즘을 적용했다. 그러나 민감한 정보를 숨기는 과정에서 변경되는 항목집합에 대한 영향을 평가하거나 숨겨지는 항목을 최소화하는 연구들은 미흡하였다.

본 논문에서는 민감한 빈발 항목집합을 숨기기 위하여 FP-Tree(Frequent Pattern Tree)기반의 확장 빈발 패턴트리(Extended Frequent Pattern Tree, eFP-Tree)를 제안한다. eFP-Tree의 노드 구성은 기존과는 다르게 빈발 항목집합 생성단계에서 트랜잭션 정보와 민감 정보, 경계 정보를 모두 구성하며, 숨기는 과정에서 비민감한 빈발 항목집합의 영향을 최소화하기 위하여 경계를 사용하였다. 본 논문의 예시 트랜잭션 데이터베이스에 eFP-Tree를 적용한 결과, 손실 항목을 평균 10%이하로 최소화하여 기존 방법들에 비해 효과적임을 증명하였고, 데이터베이스의 품질을 최적으로 유지할 수가 있었다.

키워드 : 데이터마이닝, 빈발 패턴 트리, 빈발 패턴 성장, 민감 빈발 항목집합

An Extended Frequent Pattern Tree for Hiding Sensitive Frequent Itemsets

Lee Dan Young[†] · An Hyoung Geun[†] · Koh Jae Jin[††]

ABSTRACT

Recently, data sharing between enterprises or organizations is required matter for task cooperation. In this process, when the enterprise opens its database to the affiliates, it can be occurred to problem leaked sensitive information. To resolve this problem it is needed to hide sensitive information from the database. Previous research hiding sensitive information applied different heuristic algorithms to maintain quality of the database. But there have been few studies analyzing the effects on the items modified during the hiding process and trying to minimize the hided items.

This paper suggests eFP-Tree(Extended Frequent Pattern Tree) based FP-Tree(Frequent Pattern Tree) to hide sensitive frequent itemsets. Node formation of eFP-Tree uses border to minimize impacts of non sensitive frequent itemsets in hiding process, by organizing all transaction, sensitive and border information differently to before. As a result to apply eFP-Tree to the example transaction database, the lost items were less than 10%, proving it is more effective than the existing algorithm and maintain the quality of database to the optimal.

Keywords : Data Mining, FP-Tree, FP-Growth, Sensitive Frequent ItemSets

1. 서 론

최근 기업 간에 정보를 공유하기 위한 목적으로 데이터베이스를 공개할 때 민감한 정보가 유출되는 것을 막기 위해서 데이터베이스를 공개하기 전에 민감한 정보를 숨기는 것이 중요한 이슈로 되고 있다. 데이터 마이닝 기술이 발전함에 따라서 추출 가능한 여러 종류의 민감한 정보 및 지식들이 생기게 된 것은 좋은 사례이다. 이러한 민감한 정보로는 개인에 관한 정보, 질병에 대한 의료정보, 특정 물품의 판매 패턴에 대한 정보 등이 [1]의 연구에서 소개되었다.

[1]에서 민감한 정보를 갖고 있는 데이터베이스를 다른 기업에 공개하는 것이 기업의 비즈니스에 중대한 위협이 될 수 있다는 시나리오 또한 제시하였다. 따라서 기업 간의 정보 공유에서 빈번하게 마이닝할 가능성이 있는 민감하고 비밀스러운 정보들이 있다면 미리 숨기는 것이 필요하게 되었

※ 이 논문은 2008년 울산대학교 연구비에 의하여 연구되었음.
† 정 회 원 : 울산대학교 전기공학부 외래강사
†† 정 회 원 : 울산대학교 전기공학부 교수(교신저자)
논문접수 : 2011년 2월 21일
수 정 일 : 1차 2011년 3월 26일, 2차 2011년 4월 28일
심사완료 : 2011년 4월 28일

으며, 이런 민감한 정보를 숨기기 위한 방법으로 기존 데이터베이스의 일부분을 수정해 마이닝되지 못하도록 하는 것이다. 빈번하게 사용되는 중요 민감 정보를 찾기 위한 대표적인 데이터 마이닝 방법으로 트랜잭션 데이터베이스(Transaction Database, TDB)로부터 빈발 항목집합(Frequent ItemSets, FIS)을 찾는 것이며, 그 중의 하나는 FP-Tree 기반의 빈발 패턴 성장(Frequent Pattern Growth, FP-Growth)을 이용하는 것이다[2]. 민감한 정보를 숨기는 이전 연구들은 미흡한 편이지만, 보고된 연구를 보면 수정된 결과 트랜잭션 데이터베이스(Result Transaction Database, RTDB)의 종합적인 품질 상태의 부작용을 최소화하여 유지하려는 제한된 부분에만 관심을 가졌으며, 이들 연구 방법의 대부분이 서로 다른 휴리스틱 알고리즘만을 적용하였다[3,4,10-12].

본 논문은 FIS 중에서도 중요성이 강조되는 항목집합인 민감한 빈발 항목집합(Sensitive Frequent ItemSets, SFIS)을 효과적으로 숨기기 위한 FP-tree 기반의 확장 빈발 패턴 트리(eFP-Tree)를 제안한다. eFP-Tree의 경우 지지도(support) 값을 노드에 저장하는 기존 FP-Tree 구성과는 달리 SFIS의 효과적인 숨김 작업을 위하여 각 항목의 노드에 트랜잭션 정보, 민감 정보, 경계 정보를 기록하였다. 본 논문의 eFP-Tree를 이용한 SFIS 숨기는 과정에서 비민감한 빈발 항목집합(Non-Sensitive Frequent ItemSets, NSFIS)들에 대한 영향(impact)을 평가하였으며, 이를 위하여 경계(border)를 이용하였다[7]. SFIS 숨기기는 모든 eFP-Tree의 항목집합을 고려하는 것이 아니라 NSFIS들의 경계기반 항목들을 활용하였으며 이로 인하여 RTDB의 종합적인 품질 상태를 최적으로 유지를 하였다. eFP-Tree는 숨기기 작업 과정에서 데이터베이스를 다시 스캔할 필요가 없고 SFIS 및 경계 요소에 해당하는 항목집합을 효과적으로 관리할 수 있다. 결과적으로 제안하는 eFP-Tree를 이용하여 숨겨지는 항목들의 손실을 줄이고 RTDB 품질상태를 향상 시킬 수 있었다. 더불어 변경 후 결과 항목들 간의 연관규칙 및 신뢰도 또한 향상된 것을 실험을 통하여 확인할 수 있었다.

이 후 본 논문의 구성은 다음과 같이 전개된다. 1장 서론에 이어 2장에서는 관련연구로 숨기기 알고리즘의 선행연구와 RTDB의 RFIS에 대하여 간략하게 살펴본다. 3장에서는 논문에서 제안하는 eFP-Tree 구성과 FIS, NSFIS, SFIS 생성에 대하여 기술하며 4장에서는 eFP-Tree를 이용한 SFIS 숨기기 알고리즘에 대하여 살펴보고 예시를 적용한다. 5장에서는 예시 TDB를 구현된 프로그램에 적용한 후 선행연구들과 손실항목에 대한 비교평가를 하고 마지막 6장에서 결론을 맺는다.

2. 관련 연구

2.1 선행연구

데이터베이스로부터 발견되는 관계들은 대부분이 빈발 패턴 또는 연관 규칙 형태로 표현된다. 발견된 빈발 패턴 또는 연관 규칙 중에서 민감한 정보를 숨기는 문제는 [3]의 연구에서 처음 제시를 하였다. 이 연구에서는 연관 분석을 통하여 항목집합이 주어진 최소 지지도 임계값 이상이면 FIS라 하고 이들 중에서 공개에 위험이 있는 항목집합이 최소 지지도 임계값 이상인 경우는 SFIS이라 정하고 분류하였다. 이 연구에서 SFIS를 숨기기 위해 연관 규칙 분석을 통하여 데이터베이스에서 숨길 항목을 찾아 최적으로 완전 삭제하여 숨긴다는 것은 NP-Hard임을 증명하였고, 항목들과 트랜잭션을 찾기 위한 휴리스틱 그리디 알고리즘(Heuristic greedy algorithms)을 제안하였다. 하지만 민감한 빈발 패턴을 최소 지지도 임계값 이하로 낮추지 못하여 항목의 숨김이 제한적이었다.

또 다른 [10]의 연구에서 민감한 연관 규칙과 항목집합을 숨기기 위하여 항목들의 삽입, 삭제를 통하여 TDB를 수정하는 방법을 채택하여 이전과는 다른 휴리스틱 알고리즘을 제시했다. 이 연구의 트랜잭션 t의 구성은 t=<TID, values_of_items, size> 표현하였으며, 예를 들어 트랜잭션 t의 TID가 t6번이고 항목집합 I={A,B,C}일 경우 t=<T6,[111],3>로 나타내었다. 민감한 연관 규칙에 의해서 항목 B를 숨기기 위하여 삭제할 경우 해당 트랜잭션에서 항목 B의 비트 패턴 값을 0으로 하는 트랜잭션 t=<T6,[101],3>로 표현하여 숨기는 작업을 수행하였다. 이 결과 민감한 연관 규칙에 의한 삭제 항목과 연관된 모든 항목의 삭제로 인하여 손실 정도가 높아 RTDB의 품질 상태에 부작용이 나타났다. 또한 연관 규칙에 의한 데이터베이스의 항목을 찾기 위하여 많은 검색을 필요로 한다는 문제점을 보였다.

데이터베이스 검색에서 단점을 보인 민감한 연관 규칙을 이용한 숨기기 알고리즘 대신에 민감한 항목집합 숨기기에 대한 알고리즘도 제시되었다[11]. 이 연구의 알고리즘에서는 [10]의 연구에서 표현한 트랜잭션 t의 구성을 테이블로 표현하였으며, 지지도와 신뢰도의 최소 임계값을 비교하여 민감한 항목을 숨기는 알고리즘으로 DSR(Decrease Support of Right Hand Side), ISL(Increase Support of Left Hand Side)을 제시하였다. 트랜잭션 데이터베이스의 빈발 항목을 테이블로 표현하여 검색하는데 효과적이었다. 하지만 이전 연구와 마찬가지로 숨기기 작업과정에서 항목 손실로 인한 RTDB의 종합적인 품질 상태의 부작용은 높은 편이었다.

[12]의 연구에서는 RTDB의 부작용을 줄이기 위해 제한된 민감한 연관 규칙을 찾고 이를 이용하여 숨길 수 있는 휴리스틱 알고리즘(Association Rule)을 제시하였다. 연구에서는 TDB의 항목을 비트패턴의 테이블로 구성하고, 민감한 연관 규칙에 의한 숨기기 항목명과 "Del", "Ins"라는 기호를 이용하여 정확한 숨김 작업과 RTDB의 부작용을 최소화하는 시도를 하였다. 하지만 해당 알고리즘은 민감한 규칙을 숨기거나 부작용을 최소화하기 위하여 항목 비교, 검사하는데 많은 시간적인 손실이 있었다. 또한 몇 가지 중요한 규칙 즉, 숨길 항목과 규칙을 숨기지 못하는 결과를 보였다.

위 결과 기존 연구들은 SFIS를 숨기기 위한 빈발패턴과 그에 따른 연관 규칙을 발견하고 지지도와 신뢰도를 기본으

로 하는 휴리스틱 알고리즘이 대부분이었다. 이러한 민감한 연관 규칙을 이용한 숨기기 작업은 연관 규칙에 따른 항목들 간의 포함관계로 인하여 항목 손실이 높았으며 RTDB 품질상태에 부작용이 나타났을 뿐만 아니라, 숨기기 작업 과정에서 데이터베이스 항목을 스캔하는데 많은 횟수가 걸렸다. 그러나 선행 연구들은 RTDB의 종합적인 품질상태의 부작용을 최소화하는데 많은 관심을 가져왔으며 이를 개선하기 위한 알고리즘과 트랜잭션 항목의 표현 및 구성의 변화도 제시를 하였다. 위 결과 본 논문은 다음과 같은 두 가지에 목표를 두고 진행을 하였다. 첫째는 숨기기 작업과정에서 손실될 수 있는 항목을 최소화하고, 둘째는 숨기기 한 후의 RTDB의 품질상태를 최적으로 유지하는 것이다.

2.2 RTDB와 RFIS

본 논문에서 SFIS 숨기기 과정 후 항목의 손실을 최소화하고 RTDB의 종합적인 품질 상태를 최적화 하는데 그 목적임을 기술하였다. 따라서 본 절에서는 RTDB 개요와 RTDB가 최적의 품질상태로 평가되기 위한 조건 등에 대한 기본적인 개념을 살펴보고자 한다.

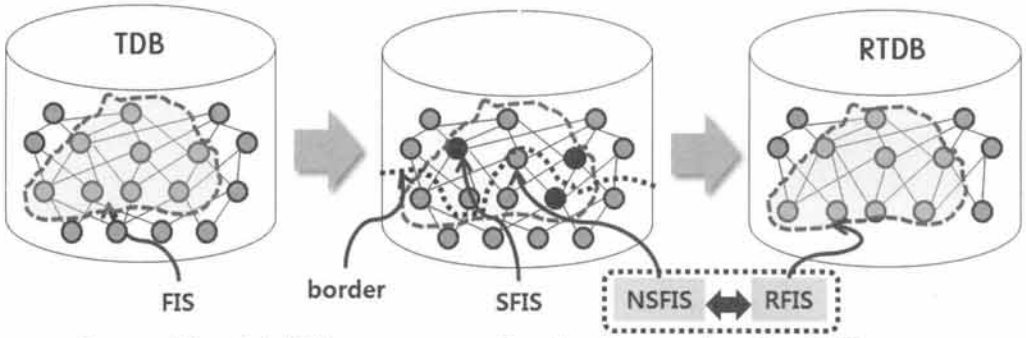
TDB의 항목들 중에서 최소 지지도 임계치 이상의 항목집합을 FIS라 하고, 이를 구성하는 항목집합은 SFIS와 NSFIS로 이루어진다. 이 중에서 민감 정보를 포함하고 있는 SFIS를 숨기기 한 후 남은 TDB의 NSFIS는 결과 빈발 항목집합(Result Frequent Itemsets, RFIS)이며 최종 RTDB가 되는 것이다. 따라서 본 논문에서 제시하는 RTDB는 (그림 1)과 같이 TDB의 FIS 중에서 SFIS를 숨기기 한 후 결과 NSFIS이며, 이는 RTDB의 FIS인 동시에 RFIS가 되는 것이다.

상기 내용의 개념으로 RTDB는 |FIS-SFIS|의 결과 항목집합이라 생각할 수 있다. 그러나 |FIS-SFIS|가 정확한 결과 항목집합은 아니다. SFIS를 숨기는 과정에서 SFIS를 포함하는 항목집합도 숨겨지기 때문에 그러한 손실 부분도 고려해야 한다. Apriori 원리[9]에 의하면 만약 한 항목집합이 빈발하면 그것의 모든 부분집합들 역시 빈발해야만 한다고 하였다. 다시 말해서, 한 항목집합이 민감하여 숨기고자 한다면 그것의 모든 부분집합들 역시 민감하여 숨겨지게 된다는 것이다. 이 원리에 의하여 숨기지 않아야 하는 항목집합의 손실 부분이 발생하는 것이며 결과적으로 RTDB는 |FIS-SFIS-X| (X : SFIS를 포함하는 부분 항목집합)가 되어야 맞을 것이다. 따라서 본 논문에서는 항목의 손실을 최소화하기 위하여 SFIS와 NSFIS 사이에 경계[7]를 이용하여 NSFIS의 상대적 지지도를 고려하였다.

RTDB의 종합적인 품질상태는 TDB의 NSFIS와 RTDB의 RFIS를 서로 비교하여 평가할 수 있으며, 다음 두 가지의 조건일 경우는 최적이 된다고 볼 수가 있다. 첫째로 |NSFIS-RFIS| 했을 때 남은 결과의 항목수가 최소한 0에 가깝거나, 둘째로 RFIS∩SFIS의 결과 항목집합이 공집합(∅)이 되는 것이다.

3. eFP-Tree와 FIS 생성

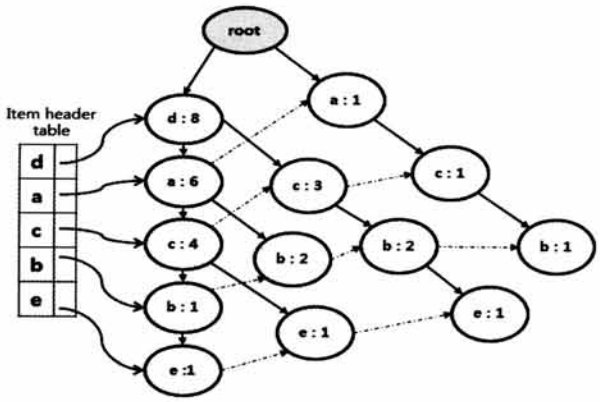
FIS을 찾아내는데 유용한 빈발 패턴 마이닝 알고리즘으로 Apriori 알고리즘[8]이 있다. Apriori 알고리즘은 n번째 항목집합이 n+1번째 항목집합을 생성하기 위한 레벨단위로 진행되는 반복 접근법을 사용한다. Apriori 알고리즘은 k-항목집합이 없을 때까지 진행을 하게 되며 이러한 과정에서 불필요한 후보 패턴을 많이 생성할 뿐만 아니라 생성된 후보 항목집합들 중에서 FIS를 찾아내기 위해 매번 계속적으로 대량의 트랜잭션을 스캔해야 하므로 속도가 느려 잘 활용되지 못하였으며 정확도 및 확장성에 대한 문제도 제기되었다. 이러한 문제점을 해결하고자 FP-Tree를 이용한 FP-Growth 알고리즘은 TDB를 한 번만 스캔하여 빈발 패턴을 찾아내며 많은 성능적인 개선을 가져오게 되었다[2]. 하지만 이러한 알고리즘은 FP-Tree의 각 노드에 항목의 가중치 또는 지지도 값을 기록하고 있고, SFIS 숨기기에 대한 이전 연구 대부분도 지지도만을 기록하고 있다[3,4,10-12]. 본 장에서는 FP-Tree에 대하여 간략하게 살펴보고 SFIS 숨기기에 효과적으로 활용할 수 있는 FP-Tree 기반의 eFP-Tree에 대하여 기술하고자 한다.



(그림 1) TDB(FIS, SFIS, NSFIS)와 RTDB(RFIS)

3.1 eFP-Tree의 구성

집합 I={i_1, i_2 … i_m}는 항목들의 집합(set)이다. 항목집합(itemset) S는 집합 I의 부분집합(subset)인 항목들의 모임이다. 트랜잭션 데이터베이스 TDB={T_1, T_2, … T_n}는 트랜잭션들의 집합이며, 각 트랜잭션 T_i는 집합 I로부터 선택된 항목들의 부분집합을 포함하고 있다. 항목집합 S의 지지도(support) 또는 발생빈도(occurrences frequency)는 TDB에서 항목집합 S를 포함하고 있는 트랜잭션의 개수(supp(S))를 말한다. 빈발 패턴 마이닝은 TDB에 나타난 여러 항목집합 S의 지지도가 주어진 지지도 임계값(m)보다 크거나 같은 패턴을 빈발(supp(S)≥m)이라 하며, 빈발 항목이 아닌 것들은 제외하고 나머지 FIS들만 FP-Tree에 삽입되고 구성된다. FP-Tree의 예시는 (그림 2)와 같으며, 지지도 3을 기준으로 <표 1>의 예시 TDB로부터 생성되었다. FP-Tree의 각 노드는 root로부터 출발하는 트리 고유의 링크와 Item Head Table로부터 출발하는 해당 항목 노드 링크 등 두 개의 링크를 유지하고, 각 노드는 항목명과 지지도 등 두 가지 요소로 구성된다.



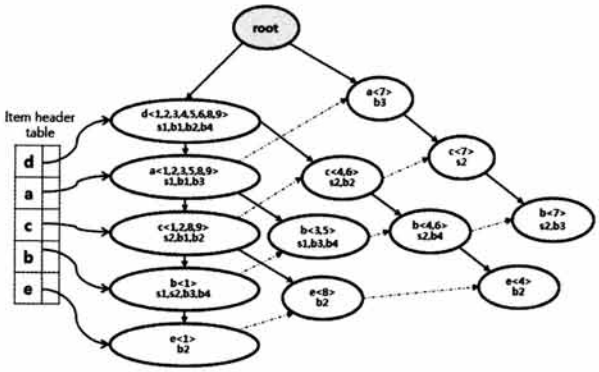
(그림 2) 예시 TDB를 이용한 FP-Tree

아래 <표 1>의 예시 TDB는 집합 I={a, b, c, d, e, f, g, h}와 같이 8개의 항목들과 총 9개의 트랜잭션으로 구성되어 있다. 빈발 항목은 최소 지지도 임계값(m)을 만족하는 항목(supp(S)≥3)들이다. 정렬된 빈발항목은 내림차순 정렬된 항목들이다.

〈표 1〉 예시 트랜잭션 데이터베이스(TDB)

TID	items	frequent items	ordered frequent items
1	a b c d e	a b c d e	d a c b e
2	a c d	a c d	d a c
3	a b d f g	a b d	d a b
4	b c d e	b c d e	d c b e
5	a b d	a b d	d a b
6	b c d f h	b c d	d c b
7	a b c g	a b c	a c b
8	a c d e	a c d e	d a c e
9	a c d h	a c d	d a c

초기 eFP-Tree는 FP-Tree를 생성하는 과정과 동일하다. 그러나 트리를 구성하고 있는 노드는 FP-Tree와 차이를 보인다. eFP-Tree의 노드 구성은 항목명과 <T_i, S_i, B_i>로 표현된다. 여기서 Ti는 해당 항목을 포함하고 있는 트랜잭션 리스트(T_1, T_2,…T_n)를 의미하며 eFP-Tree 초기 생성과정 노드에 기록된다. S_i는 숨기기 위한 민감 항목, B_i는 경계 요소 항목을 의미한다. eFP-Tree의 예시는 (그림 3)와 같으며, 지지도 3을 기준으로 <표 1>의 예시 TDB로부터 생성되었다. eFP-Tree 노드는 트랜잭션 정보, 민감 정보, 경계 정보를 구성하고 있기 때문에 숨기기 작업과정에서 데이터베이스를 반복 스캔할 필요가 없다.



(그림 3) 예시 eFP-Tree

3.2 FIS 생성과 SFIS, NSFIS

FP-Growth 알고리즘은 특정 항목 x_i에 대하여 FP-Tree를 상향식 방법으로 탐색하여 빈발 항목집합들을 생성하는 방법이다. 항목 x_i는 빈발 항목집합의 항목 중에서 빈도수가 가장 작은 항목을 의미한다. FP-Growth 같은 방법으로 (그림 3)의 결과 eFP-Tree에서 FIS를 탐색하는 방법은 특정 항목 x_i에 대한 접미패턴에서 시작하여 조건부 패턴베이스(conditional Pattern-base)를 생성한다. 이 결과를 대상으로 조건부 빈발 패턴 트리(Conditional Frequent Pattern Tree, CFP-Tree)를 구성하고, 재귀호출을 통하여 반복적으로 수행한다[2]. 예를 들어 빈발 항목 중에서 지지도가 가장 작은 항목 e를 접미부로 하는 빈발 항목집합을 찾으면 {(dacbe:1), (dace:1), (dcbe:1)} 등이며, 이 접두부 경로들은 조건부 패턴베이스를 구성한다. 항목 e의 CFP-Tree는 주어진 최소 지지도 임계값보다 작은 경로를 제외한 후, 단일 경로의 빈발 패턴인 모든 조합 ((dc:3), (d:3), (c:3))|e를 생성하게 되며 재귀호출을 통하여 나머지 항목에 대한 반복적인 작업을 한다. 결과 생성된 CFP-Tree는 아래 <표 2>와 같다.

〈표 2〉 CFP-Tree의 생성

item	conditional pattern-base	conditional frequent pattern tree
e	(dacb:1, dac:1, dcb:1)	((dc:3), (d:3), (c:3))|e
b	(dac:1, da:2, dc:1)	((da:3), (dc:3), (d:5), (a:4), (c:4))|b
c	(da:4, d:2, a:1)	((da:4), (d:6), (a:5))|c
a	(d:6)	(d:6)|c
d	∅	∅

<표 2>의 CFP-Tree로부터 숨기기 작업의 대상이 되는 FIS를 구할 수가 있다. 본 예시 TDB에서 생성하고자 하는 FIS는 최소 지지도 3 이상의 값을 가지는 3-FIS를 의미한다. SFIS는 데이터베이스 상황에 따라 입력되는 중요도 높은 FIS이며, NSFIS는 FIS에서 SFIS를 제외한 나머지 항목집합으로 기술된다. 이렇게 분류된 NSFIS가 SFIS 숨김 과정에서 손실 없이 RTDB의 RFIS로 변환되고 NSFIS≡RFIS 관계가 성립된다면 최적화된 숨기기 작업이라 볼 수 있다. 아래 <표 3>은 예시 TDB로부터 생성된 FIS와 NSFIS를 보여주고 있다. SFIS는 입력 항목집합이다.

〈표 3〉 생성된 FIS, SFIS, NSFIS

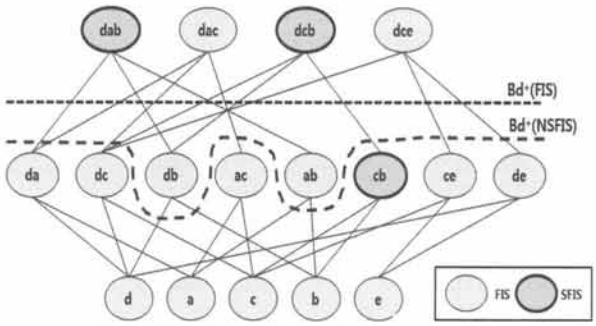
	항목집합 목록
FIS	d:8, a:7, c:7, b:6, e:3 da:6, dc:6, db:5, ac:5, ab:4, cb:4, ce:3, de:3 dab:3, dac:4, dcb:3, dce:3
SFIS	dab:3, dcb:3, cb:4
NSFIS	d:8, a:7, c:7, b:6, e:3 da:6, dc:6, db:5, ac:5, ab:4, ce:3, de:3 dac:4, dce:3

4. eFP-Tree를 이용한 SFIS 숨기기

4.1 경계를 이용한 eFP-Tree 재구성

앞 절의 Apriori 원리에 의해면 SFIS의 부분집합에 속하는 NSFIS 일부가 숨기는 과정에서 손실되는 경우가 발생하여 RTDB의 RFIS 결과에 영향을 주었으며, 이러한 과정에서 삭제의 영향으로 인하여 NSFIS의 모든 항목집합에 대해서 비교 평가하는 것은 비효율적인 문제점이 발생하였다. 이러한 문제를 해결하기 위해서 본 논문에서는 경계 접근법을 이용하였다[7]. 경계는 NSFIS 중에서 최대 NSFIS (Maximal NSFIS)에 집중하여 설정하고 숨기는 작업으로 인한 영향 평가를 경계 요소 항목에 대해서만 하는 것이다. 경계 요소 항목들은 숨기는 작업과 영향 평가의 효율성을 위하여 eFP-Tree의 노드에 저장하게 된다. 숨기는 과정에서 SFIS와 NSFIS 항목의 영향 평가에 필요한 경계요소들은 다음 조건에 의해서 생성된다. 항목집합들의 집합을 S라 할 때, $\forall X, Y \in S$에 대해서 $X \not\subseteq Y$와 $Y \not\subseteq X$를 만족하는 조건이 antichain이면, upperboard($Bd^+(S)$)와 lowerboard($Bd^-(S)$)는 항목집합들의 antichain 집합이다[7]. upperboard의 $Bd^+(S)$은 $\forall X \in S$에 대해서 $X \subseteq Y$를 만족하는 $Y \in Bd^+(S)$가 존재한다는 조건이 만족되면 요소항목이 된다. lowerboard의 $Bd^-(S)$ 요소항목은 $\forall X \in S$에 대해서 $X \supseteq Y$를 만족하는 $Y \in Bd^-(S)$가 존재한다는 조건이 만족되면 요소항목이 된다.

upperboard($Bd^+(S)$)와 lowerboard($Bd^-(S)$) 조건에 의해서 <표 3>의 FIS, NSFIS, SFIS의 각 항목집합에 대한 경계요소는 다음과 같이 구할 수 있다. antichain 조건에 의해 FIS의 경계요소 $Bd^+(FIS)$={dab, dac, dcb, dce}가 되며, SFIS의 경계요소는 $Bd^-(SFIS)$={dab, cb}가 된다. NSFIS 경계요소의 경우 SFIS의 포함 관계의 부분 항목집합으로 부터 유도된다. 각 항목집합 $X \in NSFIS$일 때 NSFIS로부터 해당 항목집합 X가 SFIS의 부분 항목집합인 경우를 제외한 나머지 NSFIS의 모든 항목집합을 경계요소에서 제거함으로써 $Bd^+(NSFIS)$를 얻을 수 있다. 예로 (그림 4)에서 NSFIS의 항목집합 {db, ab}는 SFIS의 항목집합 {dab, dcb}와 포함관계에 있는 부분 항목집합이기 때문에 $Bd^+(NSFIS)$의 경계요소로 남기고 그 외 나머지는 $Bd^-(NSFIS)$에 포함하여 경계요소에서 제거를 한다. 이 결과 NSFIS의 경계요소는 $Bd^+(NSFIS)$={dac, dce, ab, db}가 된다. NSFIS와 SFIS의 경계요소를 이용하는 이유는 숨기는 과정에서의 손실되는 항목수를 줄이기 위함이다. (그림 4)는 FIS 격자표현에서 FIS, SFIS, NSFIS의 포함관계에 따른 항목을 구분하기 위한 경계를 보여주고 있다.

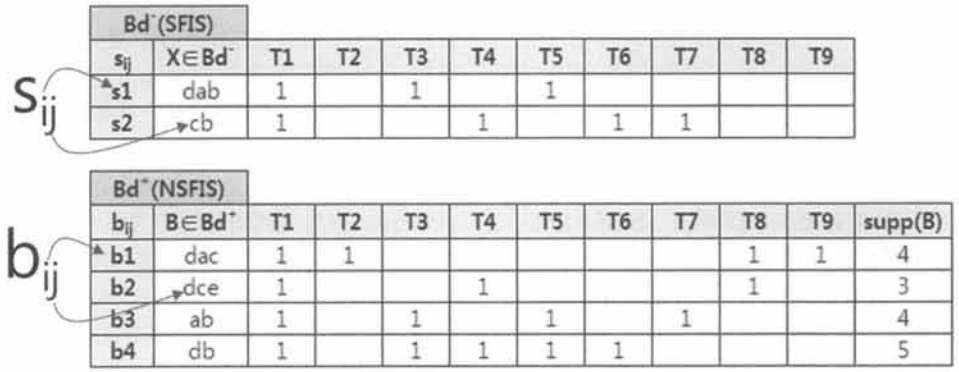


(그림 4) FIS에서의 경계 구분

따라서 경계 결과 SFIS의 모든 항목집합을 숨길 필요 없이 민감성이 있는 $Bd^-(SFIS)$={dab, cb} 경계요소 항목들만을 숨기기하면 된다. 또한 SFIS 숨기기 과정에서 모든 NSFIS에 대한 영향을 평가하지 않고 $Bd^+(NSFIS)$={dac, dce, ab, db}의 경계요소 항목들만 영향을 집중적으로 평가한다. 과정 중에 $Bd^+(NSFIS)$의 한 요소가 비빈발이 되면 다른 요소를 고려해야 하며, 이것은 $Bd^+(NSFIS)$의 요소들 사이의 상대 지지도를 유지하기 위함이다.

생성된 경계요소 $Bd^+(NSFIS)$와 $Bd^-(SFIS)$의 정보를 eFP-Tree의 노드에 기록함으로써 (그림 3)과 같이 본 논문에서 제안하는 최종적인 eFP-Tree를 구성하게 된다. 경계요소 $Bd^+(NSFIS)$와 $Bd^-(SFIS)$를 포함하고 있는 트랜잭션 정보를 비트 패턴 테이블로 나타내면 (그림 5)와 같이 나타낼 수 있다.

경계요소 접근의 예로 (그림 5)에서 s_{ij}는 $Bd^-(SFIS)$ 경계요소 항목이며 숨기기 위한 SFIS의 후보항목을 의미한다. 여기서 i는 $Bd^-(SFIS)$의 민감한 경계요소인 s_i의 순번이며, j는 s_i가 포함하고 있는 항목 순번을 의미한다. 예를 들어 s_{12} 경우는 $Bd^-(SFIS)$={dab, cb}의 첫 번째 경계요소 s_1(dab)에서 두 번째 항목을 의미하며 결과 s_{12}={a}가 된다. b_{ij}는 $Bd^+(NSFIS)$의 경계요소의 항목집합이며, i는 $Bd^+(NSFIS)$의 비민감한 경계요소인 b_i의 순번이며, j는 해당 b_i가 포함하고 있는 항목 순번을 의미한다. 예를 들어 b_{22}의 경우는



Bd^-(SFIS)

s_{ij}	X∈Bd^-	T1	T2	T3	T4	T5	T6	T7	T8	T9
s1	dab	1		1		1				
s2	cb	1			1		1	1		

Bd^+(NSFIS)

b_{ij}	B∈Bd^+	T1	T2	T3	T4	T5	T6	T7	T8	T9	supp(B)
b1	dac	1	1						1	1	4
b2	dce	1			1				1		3
b3	ab	1		1		1		1			4
b4	db	1		1	1	1	1				5

(그림 5) Bd-(SFIS)와 Bd+(NSFIS)의 트랜잭션 정보

Bd^+(NSFIS)={dac, dce, ab, db}의 두 번째 경계요소 b_2(dce)에서 두 번째 항목을 의미하며 결과 b_{22}={c}가 된다.

민감 항목 숨기기는 Bd^-(SFIS)의 s_{ij}에 대한 impact 값을 구하고 그 결과 중에서 최소 impact 값을 찾아서 해당 트랜잭션의 항목을 삭제하는 것이다. 이러한 impact 값은 해당 Bd^+(NSFIS) 경계요소에 가중치(weight)를 적용한 후 항목들의 전체 합으로 구할 수 있으며, 가중치는 경계요소들의 상대적 지지도를 유지할 수 있다. 가중치는 숨기기 과정 동안 변경된 지지도에 의해서 다시 계산하여 반영된다. 가중치 합에 의한 impact 값이 작을수록 NSFIS 항목 손실이 최소화되고 숨기게 될 확률이 높은 SFIS 항목이 되는 것이다. [7]의 연구에 의하면 impact 값을 구하기 위한 Bd^+(NSFIS) 항목집합 B의 가중치(w(B))는 아래 조건에 의해서 구해진다.

```
if Psupp(B)≥(m+1) then                    // m=3
    w(B)=supp(B)-Psupp(B)+1 / (supp(B)-m);
if 0≤Psupp(B)≤m then
    w(B)=n+m-Psupp(B);                    // |Bd+|≤n
    ≤∞(n : 임의의 큰 정수)
```

여기서 TDB는 제시된 데이터베이스이며 B∈Bd^+(NSFIS)는 경계 항목이다. 수정 처리 중인 TDB를 PTDB, Psupp(B)를 PTDB 항목 B의 지지도, supp(B)를 TDB 항목 B의 지지도이다.

4.2 SFIS 숨기기 알고리즘

본 논문의 SFIS 숨기기 작업은 가중치 합에 따른 impact 값을 계산하여 Bd^+(NSFIS)에 최소의 영향을 주는 Bd^-(SFIS)의 항목을 찾아서 삭제하는 것이다. SFIS의 항목 집합 S의 숨기기 후보를 s_i라 하고 숨기기 후보항목이라 한다. w(b_i)는 각 경계요소 b_i∈$Bd^+|_S$의 가중치라 한다면 impact는 $I(T_i, s_{ij}) = \sum w(b_i)$와 같이 값을 계산할 수 있다. 여기서 T_i는 경계요소 s_{ij}를 포함하고 있는 트랜잭션을 말한다. impact 값에 의한 민감한 항목 삭제를 위한 알고리즘은 (그림 6)과 같다.

```
Input :
   TDB : Transaction Database;
   m : Minimal Support Threshold;
   SFIS : Sensitive Frequent Itemsets;
Output :
   RTDB : Result Transaction Database;

Method :
   Compute m-FIS and NSFIS;
   Compute Bd-(SFIS), Bd+(NSFIS);
   Compute Sort Itemsets in Bd-(SFIS);

   for each X∈Bd-(SFIS) do
      Compute Bd+|s and w(bj) where bj∈Bd+|s;
      Initialize C(C is the set of hiding candidate of X);
      for(i=0;i<supp(X)-m+1; I++) do
         Find ui = (Ti, xi) such that I(ui) = Min{I(u)|u∈C};
         Update C = C - {(T, x) | T=Ti};
         Update w(bj) where bj∈Bd+|s;
      Update database TDB;
   Output RTDB = TDB;
```

(그림 6) SFIS 숨기기 알고리즘

위 알고리즘의 숨기기 과정을 간략하게 기술하면 아래와 같다.

① eFP-Tree에서 민감한 항목의 경계요소 Bd^-(SFIS)의 s_i를 포함하고 있는 트랜잭션 리스트들을 구한다. eFP-Tree의 Item Header Table의 접미부분에 해당하는 항목부터 시작하여 상향식으로 트랜잭션의 항목들을 탐색하면서 경계요소 Bd^-(SFIS)의 s_i를 포함하는지 검사한다. 이 과정에서 s_i를 포함하는 트랜잭션 리스트 T_1, T_2,…T_n(n : TID number)을 구한다.

② ①에서 T_i별 경계요소의 항목 s_{ij}를 포함하고 있는 Bd^+(NSFIS)의 impact인 $I(T_i, s_{ij})$를 계산한다.

③ ②에서 구한 각 T_i별 impact 결과 중에서 가장 작은 impact의 $I(T_i, s_{ij})$를 찾는다.

④ 만약 ③결과에서 동일한 값의 impact $I(T_i, s_{ij})$가 여러 개 있는 경우, 트랜잭션 크기가 큰 것을 선택한다. 트랜잭션 크기까지 같을 경우 해당 s_{ij}의 지지도가 큰 것을 최종적으로 선택한다.

⑤ ③ 또는 ④의 결과에서 삭제 후보항목 s_{ij}를 삭제하고 경계요소 bi∈Bd+|s의 가중치 w(bi)를 수정한다.

⑥ Bd^-(SFIS)의 모든 s_i 항목이 삭제될 때까지 ①에서부터 반복 수행한다.

<표 1>에서 제시된 예시 TDB를 이용하여 숨기는 과정 알고리즘까지 살펴보았으며, 숨기기 과정에 필요한 경계정보 Bd^+(NSFIS)={dac, dce, ab, db}, Bd^-(SFIS)={dab, cb} 등을 기술하였다. 위 (그림 5)의 Bd^+(NSFIS), Bd^-(SFIS) 비트 패턴 테이블과 (그림 6) 알고리즘을 이용하여 숨기기 항목으로 Bd^-(SFIS)={dab}를 실례로 적용고자 한다.

① Bd^-(SFIS)의 첫 번째 민감한 경계요소 s_1의 항목 s_1(dab)을 포함하는 트랜잭션 리스트 결과는 T_1, T_3, T_5이다.

② 가중치 $w(b_i)$를 구한 후, s_i의 각 항목별 impact $I(T_i, s_{ij})$를 계산한다.

i) 가중치 $w(b_i)$를 구한다.

- $w(b_1) = (4 - 4 + 1) / (4 - 3) \therefore w(b_1) = 1$
- $w(b_2) = (n + 3 - 3) \therefore w(b_2) = n(n>1)$
- $w(b_3) = (4 - 4 + 1) / (4 - 3) \therefore w(b_3) = 1$
- $w(b_4) = (5 - 5 + 1) / (5 - 3) \therefore w(b_4) = 1/2$

ii) 가중치 $w(b_i)$ 결과값을 참고하여 impact $I(T_i, s_{ij})$를 구한다.

- $I(T_1, d) = w(dac) + w(dce) + w(db) \therefore I(T_1, d) = n + 3/2$
- $I(T_1, a) = w(dac) + w(db) \therefore I(T_1, a) = 2$
- $I(T_1, b) = w(ab) + w(db) \therefore I(T_1, b) = 3/2$
- $I(T_3, d) = w(db) \therefore I(T_3, d) = 1/2$
- $I(T_3, a) = w(ab) \therefore I(T_3, a) = 1$
- $I(T_3, b) = w(ab) + w(db) \therefore I(T_3, b) = 3/2$
- $I(T_5, d) = w(db) \therefore I(T_5, d) = 1/2$
- $I(T_5, a) = w(ab) \therefore I(T_5, a) = 1$
- $I(T_5, b) = w(ab) + w(db) \therefore I(T_5, b) = 3/2$

③ ②의 impact $I(T_i, s_{ij})$ 결과 중에서 가장 작은 $I(T_3, d)$, $I(T_5, d)$ 중 하나를 선택한다.

④ impact 값이 동일한 경우는 우선 트랜잭션 크기가 큰 impact $I(T_3, d)$을 선택하여 삭제를 수행한다.

⑤ 경계요소 $b_i \in Bd^+|_s$의 가중치 $w(b_i)$를 수정한다. (그림 7)은 숨기기 후의 반영된 결과를 나타내고 있다.

⑥ Bd^-(SFIS)의 모든 s_i 항목이 삭제될 때까지 ①에서부터 반복 수행한다.

5. 프로그램 적용과 성능 평가

5.1 예시 TDB의 프로그램 적용

본 논문에서는 SFIS를 숨기기 위하여 FIS와 NSFIS 등을 추출하고 민감 정보와 경계 정보를 노드에 저장하는 eFP-Tree를 제안하였고 숨기기에 대한 알고리즘까지 살펴보았다. 본 절에서는 상기에서 기술한 내용을 바탕으로 본 논문의 <표 1>의 예시 TDB를 이용하여 SFIS를 숨기기 한 후 RTDB로 변경되었을 때의 최종적인 품질상태를 구현된 프로그램을 이용하여 살펴보고자 한다. SFIS의 숨기기 과정에서 NSFIS들이 포함 항목집합의 관계로 인하여 지지도가 여러 수준에서 감소되고 삭제되어 결과 항목의 손실도 발생하였다. 이에 본 논문에서는 eFP-Tree를 이용하여 SFIS을 숨기기한 결과 NSFIS의 손실이 최소화 되었고 RTDB 역시 최적의 품질상태를 유지할 수가 있었다. (그림 8)은 SFIS 숨기기 과정 후 NSFIS와 RTDB의 RFIS의 손실 항목집합의 비교를 보여주고 있다. NSFIS의 한 항목을 제외한 모든 NSFIS는 RTDB의 FIS로 반영되어 RTDB의 품질은 양호함을 유지하였다. 숨기기 과정에서 손실된 항목은 NSFIS{db}이며 (그림 8)의 점선으로 표현된 타원 부분을 말한다.

예시 TDB의 적용과 기본적인 실험으로 RTDB의 결과를 확인하고 연관 규칙을 평가하기 위하여 Microsoft사의 Visual Studio 2010 C#을 이용하여 (그림 9)과 같이 프로그램을 구현 하였으며, <표 1>의 예시 TDB는 9개 트랜잭션과 8개 항목으로 구성되어 있다. 초기 신뢰도와 지지도는 각각 80%와 30%를 기준으로 실험하였다. 구현된 프로그램을 통하여 (그림 7)의 손실 정보를 확인하였으며, SFIS 숨기기한 결과에 강한 규칙의 신뢰도가 많이 향상되었다는 것을 볼 수가 있었다. (그림 9)의 점선 사각형 영역은 TDB의 NSFIS와 RTDB의 RFIS의 결과 항목집합의 비교를 나타내고 있다.

5.2 성능 평가

제안된 eFP-Tree를 이용한 SFIS 숨기기 실험은 Windows XP 운영체제의 노트북(RAM : 1GB, 1.5GHz) 환경에서 이루어지고 성능 평가 되었다. 실험을 위한 TDB를 혼합 생성하는데 IBM data generator[13]를 이용하였다. 실

s_{ij}	$X \in Bd^-$	T1	T2	T3	T4	T5	T6	T7	T8	T9
s1	dab	1		1		1				
s2	cb	1			1		1	1		

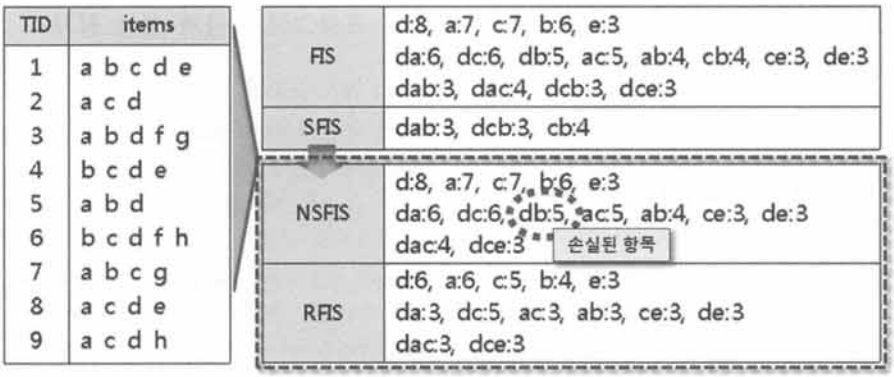
숨긴 항목

변경 가중치

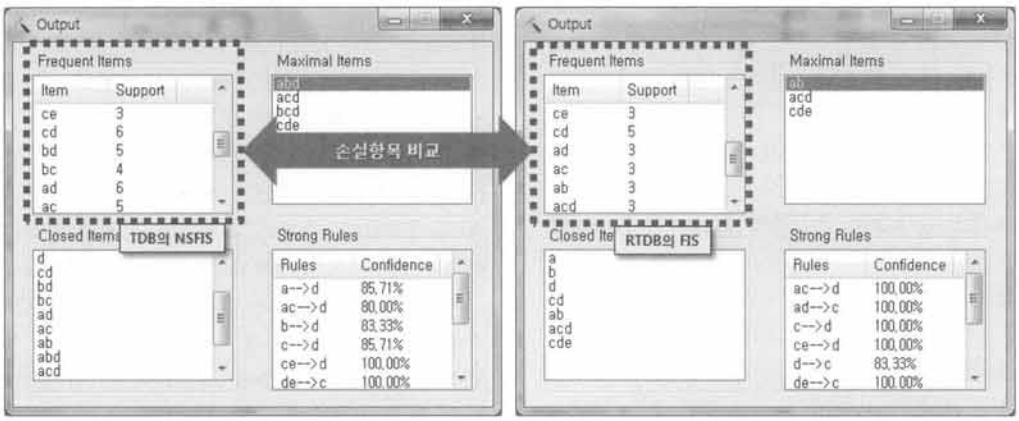
b_{ij}	$B \in Bd^+$	T1	T2	T3	T4	T5	T6	T7	T8	T9	supp(B)	Rsupp(B)
b1	dac	1	1						1	1	4	4
b2	dce	1			1				1		3	3
b3	ab	1		1		1		1			4	4
b4	db	1		1	1	1	1				5	4

숨김으로 인한 평가 항목

(그림 7) Bd-(SFIS)={dab} 숨기기 후 반영된 결과



(그림 8) 숨기기 과정 후 NSFIS, RFIS의 손실항목 비교



(그림 9) 프로그램 적용 후의 NSFIS와 RFIS 결과비교

험의 데이터베이스 크기는 연속적인 5K, 10K, 15K, 20K, 25K, 30K로 생성하여 비교하였다. 생성된 TDB에서 각 트랜잭션의 평균 크기는 10에서 50개의 항목들로 구성되었으며, 최소 지지도 임계값을 30%로 설정하였다.

본 논문에서는 SFIS 숨기기 과정에서 손실되는 항목의 수와 RTDB의 품질상태에 중점을 두고 실험을 하였다. 숨기기 위한 SFIS 항목의 수는 5개와 10개 두 가지 조건으로 생성된 TDB에 적용하였으며, 그 결과의 값들을 비교하였다. 비교되는 실험 결과는 숨기는 과정에서 손실되는 항목수와 수정되는 항목수, TDB의 FIS와 RTDB의 RFIS이다. (그림 10)은 SFIS가 5개인 경우의 실험 결과 테이블이며, (그림 11)은 SFIS가 10개인 경우의 실험 결과 테이블이다. 실험 결과 TDB의 크기에 따라 손실되는 항목의 수가 많은 차이를 보이지 않았으며 숨긴 후 RTDB의 품질상태는 양호함을 보였다.

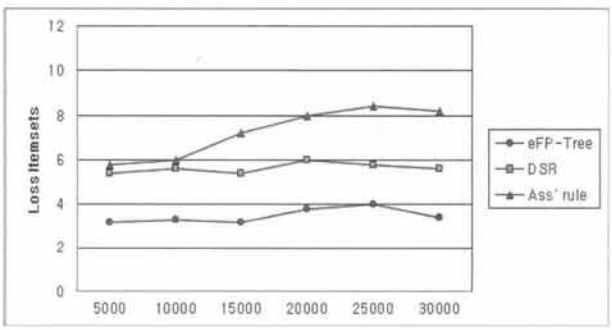
TDB	FIS	RFIS	#loss itemsets	#modified entries
5000	439	432.4	3.2	132
10000	417	410.2	3.3	298.2
15000	426	419.5	3.2	501
20000	442	435.1	3.8	699.3
25000	432	425	4	890.4
30000	443	436.4	3.4	851.8

(그림 10) |SFIS|=5에 대한 실험 적용결과

TDB	FIS	RFIS	#loss itemsets	#modified entries
5000	439	430.2	5.6	202.6
10000	417	418	7.5	583.4
15000	426	417	6.9	850.6
20000	442	433	7	1104
25000	432	423.6	6.9	1390.6
30000	443	434.8	7.8	1491.2

(그림 11) |SFIS|=10에 대한 실험 적용결과

(그림 12)는 본 논문의 eFP-Tree를 이용한 알고리즘과 선행연구인 DSR 알고리즘[11], Association Rule 알고리즘[12]에 이전 생성된 TDB와 SFIS 항목 5개를 적용하여 손실된 항목의 비교 결과를 나타내고 있다.



(그림 12) |SFIS|=5에 대한 손실항목 비교 결과

DSR 알고리즘의 경우 TDB의 크기와는 상관없이 손실되는 항목 수에 차이를 보이지 않았다. 하지만 Association Rule 알고리즘의 경우 TDB의 크기에 따라 손실되는 항목 수가 많은 차이를 보였다. 따라서 RTDB의 품질상태는 DSR 알고리즘이 Association Rule 알고리즘보다 우수하다고 할 수 있다. 하지만, 본 논문에서 제안하는 eFP-Tree를 이용할 경우 DSR 알고리즘보다 손실 항목이 많이 줄었으며, TDB 크기에 따른 손실 항목의 수에 큰 차이가 없음을 알 수 있었다. 따라서 eFP-Tree를 이용한 후의 RTDB 품질상태가 더 최적임을 알 수 있었다.

6. 결 론

회사들 간에 필요한 정보를 이용하기 위해서 서로 데이터베이스를 공유하게 되었으며, 최근 데이터 마이닝 기술이 발전함에 따라서 필요한 정보를 추출하는 것도 가능하게 되었다. 이 과정에서 데이터베이스에 있는 민감한 정보를 포함한 다양한 정보가 마이닝될 수 있다는 문제점을 가지게 되었다. 이 결과로 다른 회사에서 마이닝이 가능한 민감한 정보를 미리 숨기는 것이 필요하게 되었으며, 본 논문에서 중요성이 강조되는 SFIS를 효과적으로 숨기기 위하여 FP-Tree기반의 eFP-Tree를 제안을 하였다. eFP-Tree는 각 항목의 노드에 기존의 가중치, 지지도 값이 기록되는 것이 아니라 트랜잭션 정보, 민감성 정보, 경계 정보를 기록하였다. eFP-Tree는 이러한 참조 정보를 기록하고 유지하기 때문에 작업과정에서 TDB를 재 스캔할 필요 없이 SFIS 및 경계요소에 해당하는 항목집합을 효율적으로 관리할 수 있었으며, SFIS 숨기는 작업에서 효과적으로 이용하였다. 또한 SFIS 숨기는 과정에서 손실되는 항목을 최소화함으로써 RTDB의 품질상태를 최적으로 유지할 수 있었다. 본 논문에서 제안하는 알고리즘의 성능 평가를 위하여 프로그램을 구현하고 기본적으로 논문에서 제시하는 예시 TDB를 적용하여 논문에서 기술되어진 이론의 정확함을 보였으며, 추가적으로 제안된 알고리즘에 다양한 크기(5K ~ 30K)의 TDB를 생성하여 SFIS 항목 수를 5개와 10개로 적용하고, 손실항목 및 RTDB의 결과 값을 비교 실험하였다. 적용 결과 TDB의 각 크기가 증가해도 손실 항목은 큰 차이를 보이지 않았으며 RTDB의 품질상태도 양호함을 보였다. 그리고 동일 TDB와 SFIS의 항목 수 5개를 선행연구 DSR 알고리즘과 Association Rule 알고리즘에 적용하여 비교한 결과 논문에서 제시한 알고리즘과 손실 항목 수에 큰 차이를 보였으며 eFP-Tree를 적용한 RTDB 품질상태가 최적임을 입증하였다. 또한 숨기기 후 항목들 간의 연관규칙 및 신뢰도 또한 향상된 것을 프로그램 구현으로 알 수 있었다.

참 고 문 헌

[1] C. Clifton and D. Marks, "Security and privacy implications of data mining," Data Mining and Knowledge Discovery, Proc. of the ACM workshop Research Issues in Data Mining and Knowledge Discovery, pp.15-19, 1996.

[2] J. Han, J. Pei, Y. Yin, R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," Data Mining and Knowledge Discovery, Vol.8, pp.53-87, 2004.

[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim and V. Verykos, "Disclosure limitation of sensitive rules," Proc. of the IEEE workshop Knowledge and Data Eng. Exchange, pp.45-52, 1999.

[4] E. Dasseni, V. S. Verykios, A. K. Elmagarmid and E. Bertino, "Hiding association rules by using confidence and support," Proc. of the 4th Information Hiding Workshop, pp.369-383, 2001.

[5] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid, "Privacy preserving association rule mining," Proc. of the IEEE workshop Research Issues in Data Eng., 2002.

[6] S. Oliveira, O. Zaiane and Y. Saygin, "Secure association rule sharing," Proc. of the 8th Pacific-Asia Conference Knowledge Discovery and Data Mining, pp.74-85, 2004.

[7] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," Data Mining and Knowledge Discovery, Vol.1, No.3, pp.241-258, 1997.

[8] R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. of the 11th International Conference on Data Engineering (ICDE'95), pp.3-14, 1995.

[9] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," Proc. of the 20th International Conference on Very Large Data Bases, pp.487-499, 1994.

[10] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," IEEE Trans. Knowledge and Data Eng., Vol.16, No.4, pp.434-447, 2004.

[11] Shyue-Liang Wang, "Hiding sensitive predictive association rules," Systems, Man and Cybernetics, 2005 IEEE International Conference on Information Reuse and Integration, Vol.1, pp.164-169, 2005.

[12] Yi-Hung Wu, Chai-Ming Chiang and Arbee L. P. Chen, "Hiding Sensitive Association Rules with Limited Side Effects," IEEE Transactions on Knowledge and Data Engineering, Vol.19, Issue 1, pp.29-42, 2007.

[13] http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml

이 단 영

e-mail : danyoung64@ulsan.ac.kr

1994년 울산대학교 전자계산학과 (교육학 석사)

2002년 울산대학교 컴퓨터정보통신공학부 (박사과정)

2002년~2004년 울산대학교 IT교육원 객원교수

2006년~2009년 울산대학교 컴퓨터정보통신공학부 객원교수

2011년~현 재 울산대학교 전기공학부 외래강사

관심분야 : 데이터마이닝, DB분석/설계, ERP, e-Business

안 형 근

e-mail : hkahn@ulsan.ac.kr

2003년 울산대학교 정보통신공학과 (공학석사)

2008년 울산대학교 컴퓨터정보통신공학부 (공학박사)

1997년~2004년 현대오토시스템 기술지원부

2004년~2006년 (주)CFIC 기업부설연구소 연구소장

2008년~2010년 울산대학교 컴퓨터정보통신공학부 객원교수

2011년~현 재 울산대학교 전기공학부 외래강사

관심분야 : 멀티미디어DB, DB설계/분석, ERP, BPM, Workflow

고 재 진

e-mail : jjkoh@ulsan.ac.kr

1972년 서울대학교 응용수학과(공학사)

1981년 서울대학교 계산통계학과 (이학석사)

1990년 서울대학교 컴퓨터공학과 (공학박사)

1975년~1979년 한국후지쯔(주) 기술개발부 사원

1979년~2010년 울산대학교 컴퓨터정보통신공학부 교수

2011년~현 재 울산대학교 전기공학부 교수

관심분야 : DB시스템, 전문가 시스템, DB설계, ERP