Journal of Korea Multimedia Society Vol. 14, No. 10, October 2011(pp. 1323-1334)

민감한 빈발항목집합을 숨기기 위한 경계기반 HSFI 알고리즘

이단영[†], 안형근[††], 고재진[†††]

요 약

민감한 정보 숨기기 알고리즘은 민감한 정보를 보호하기 위하여 트랜잭션 데이터베이스를 삭제한다. 데이터 변경은 삭제 접근 방법들 중 하나이다. 민감한 정보를 숨기는 이전 연구들은 결과 데이터베이스의 품질을 유지하기 위해 서로 다른 휴리스틱 알고리즘을 적용했다. 그러나 민감한 정보를 숨기는 과정에서 변경되는 항목집합에 대한 영향을 평가하거나 숨겨지는 항목을 감소시키는 연구들은 미흡하였다. 본 논문에서는 민감한 빈발 항목집합을 숨기기 위하여 경계기반의 HSFI(Hiding Sensitive Frequent Itemsets) 알고리즘을 제안한다. 본 알고리즘에서 FP-Tree의 노드 정보는 기존과는 다르게 빈발 항목집합 생성단계에서 트랜잭션 정보와 민감 정보, 경계 정보를 모두 구성하며, 숨기는 과정에서 비민감한 빈발 항목집합의 영향을 줄이기 위하여 경계를 사용하였다. 본 논문의 예시 트랜잭션 데이터베이스에 HSFI를 적용한 결과, 손실 항목을 크게 감소시킴으로써 기존 방법들에 비해 효과적임을 증명하였고, 보다 개선된 데이터베이스의 품질을 유지할 수가 있었다.

Border-based HSFI Algorithm for Hiding Sensitive Frequent Itemsets

Dan-Young Lee[†], Hyoung-Keun An[††], Jae-Jin Koh[†††]

ABSTRACT

This paper suggests the border based HSFI algorithm to hide sensitive frequent itemsets. Node formation of FP-Tree which is different from the previous one uses the border to minimize the impacts of nonsensitive frequent itemsets in hiding process, including the organization of sensitive and border information, and all transaction as well. As a result of applying HSFI algorithms, it is possible to be the example transaction database, by significantly reducing the lost items, it turns out that HSFI algorithm is more effective than the existing algorithm for maintaining the quality of more improved database.

Key words: Data Mining(데이터마이닝), FP-Tree(빈발 패턴 트리), FP-Growth(빈발 패턴 성장), Sensitive Frequent ItemSets(민감 빈발 항목집합)

1. 서 론

개인 사생활 정보 보호를 위한 데이터 마이닝이 최근의 새로운 연구 방향이 되고 있다. 개인 사생활 정보 보호 데이터 마이닝의 고려사항으로는 첫째, 이름, 주소 및 식별자와 같은 민감한 데이터는 데이터 수신자가 다른 사람의 프라이버시를 침해할 수 없도록, 원본 데이터베이스에서 민감한 데이터가 수정 또

※ 교신저자(Corresponding Author) : 고재진, 주소 : 울산광역시 남구 대학로 93 울산대학교 7호관205호(680-749), 전화 : 052)259-2216, FAX : 052)259-1687, E-mail : jjkoh@ulsan.ac.kr
접수일 : 2011년 5월 19일, 수정일 : 2011년 8월 8일
완료일 : 2011년 9월 2일
[†] 정회원, 울산대학교 전기공학부
(E-mail : danyoung64@ulsan.ac.kr)
[††] 정회원, 울산대학교 전기공학부
(E-mail : hkahn@ulsan.ac.kr)
[†††] 울산대학교 전기공학부
※ 이 논문은 현대중공업 지원에 의한 울산대학교 전기공학부 일류화 연구비에 의하여 연구되었음.

는 삭제와 같이 변경되어 마이닝되지 않아야 한다. 둘째, 지식 또한 데이터의 프라이버시를 침해할 수 있기 때문에 데이터 마이닝 알고리즘을 사용하여 데이터베이스에서 마이닝 할 수 있는 민감한 지식의 데이터는 사전에 제외되어야 한다. 이에 최근 개인 정보는 빈번하게 마이닝 할 가능성이 높기 때문에 민감하고 비밀스러운 데이터가 있다면 이를 제공하기 전에 데이터베이스를 수정하여 마이닝 못하도록 할 필요성이 대두되고 있다[1].

빈번하게 마이닝 되는 중요 민감 데이터를 숨기기 위한 대표적인 데이터 마이닝 방법으로 트랜잭션 데이터베이스(Transaction Database: TDB)로부터 빈발 항목집합(Frequent ItemSets: FIS)을 찾는 것이며, 그 중의 하나는 FP-Tree기반의 빈발 패턴 성장(Frequent Pattern Growth, FP-Growth)을 이용하는 것이다[2]. 보고된 이전 연구의 민감 데이터 숨기기는 대부분 작업 후 데이터베이스의 수정된 결과 트랜잭션 데이터베이스(Result Transaction Database, RTDB)의 품질에만 관심을 가졌으며, 숨기기 과정에서 발생하는 항목손실의 평가 연구는 미흡한 편이었다[3,4].

따라서, 본 논문에서는 마이닝 과정에서 빈발하면서 정보 보호의 중요성이 강조되는 항목집합인 민감한 빈발 항목집합(Sensitive Frequent Itemsets: SFIS)를 효과적으로 숨기고, 숨기는 과정에서 발생할 수 있는 비민감 빈발 항목집합(Non-Sensitive Frequent Itemsets: NSFIS)의 손실을 최소화하는데 목표를 두고 연구하였다. 이를 위하여 SFIS와 NSFIS 항목집합들 간의 상대적인 지지도를 줄임으로써 빈발한 민감 항목을 숨기기 위한 HSFI(Hiding Sensitive Frequent Itemsets) 알고리즘을 제안하고자 한다. HSFI 알고리즘에서는 SFIS와 NSFIS와의 상대적 지지도와 영향(impact)[1]을 평가하기 위하여 경계(border)를 이용하였다[5].제안하는 HSFI 알고리즘은 SFIS 숨기기 과정에서의 최대 문제인 항목손실의 문제점을 개선하여 부작용을 줄이고 최종적인 RTDB의 품질상태를 향상시킬 수 있었으며, 항목들 간의 연관규칙 및 신뢰도 또한 향상된 것을 실험을 통하여 확인할 수 있었다.

이후 본 논문의 구성은 다음과 같다. 1장 서론에 이어 2장에서는 관련연구로 숨기기 알고리즘의 선행연구와 숨기기 전략에 대하여 살펴보고 숨기기 후에 평가 대상이 되는 RTDB의 RFIS에 대하여 간략하게 기술한다. 3장에서는 FP-Tree에 의해서 생성되는 빈발항목과 FIS, SFIS, NSFIS에 대하여 살펴보고, 상대적 지지도를 위한 경계에 대하여 간략하게 기술한다. 4장에서는 경계기반의 R_SFIS, R_NSFIS 경계요소 항목집합과 이를 이용한 가중치, 지지도에 대하여 살펴보고, 실제 SFIS 숨기기 알고리즘에 적용하여 예를 들어 기술한다. 5장에서는 예시 TDB를 구현된 프로그램에 적용한 후 선행연구들과 손실항목에 대한 비교평가를 하고 마지막 6장에서 결론을 맺는다.

2. 관련연구

2.1 선행연구

데이터베이스로부터 발견되는 관계들은 대부분이 빈발 패턴 또는 연관 규칙 형태로 표현된다. 발견된 빈발 패턴 또는 연관 규칙 중에서 민감한 데이터를 숨기는 문제는 [6]의 연구에서 처음 제시되었다. 이 연구에서는 SFIS를 숨기기 위해 연관 규칙의 분석을 통하여 데이터베이스에서 민감 항목을 찾아 최적으로 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 [7]의 연구에서 민감한 연관 규칙과 항목집합을 숨기기 위하여 항목들의 삽입, 삭제를 통하여 TDB를 수정하는 방법을 채택하여 이전 연구와는 다른 휴리스틱 알고리즘을 기술하였다. 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 상태

1) 숨기기 작업과정에서 SFIS에 의하여 손실되는 NSFIS의 상태

에 부작용이 나타났다. 또한 연관 규칙에 의한 데이터베이스의 항목을 찾기 위하여 많은 검색을 필요로 한다는 문제점을 보였다.

항목 검색에서 문제점을 보인 [7]의 연구를 보완하기 위한 또 다른 알고리즘도 제시되었다[8]. 이 연구의 알고리즘에서는 [7]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 빈발항목을 테이블로 표현하여 검색하는데 효과적이었다. 하지만 이전 연구와 마찬가지로 숨기기 작업과정에서 항목 손실로 인한 RTDB의 종합적인 품질 상태의 부작용은 높은 편이었다.

[9]의 연구에서는 숨기기 후 RTDB의 부작용을 줄이기 위해 제한된 민감한 연관 규칙을 찾고 이를 이용하여 숨길 수 있는 알고리즘을 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. 그러나 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.

민감한 연관 규칙을 이용한 숨기기 작업은 연관 규칙에 따른 항목들 간의 포함관계로 인하여 항목 손실이 높았으며 RTDB 품질상태에 부작용이 나타났을 뿐만 아니라, 숨기기 작업과정에서 데이터베이스 항목을 스캔하는데 많은 횟수가 걸린다는 문제점을 보였다. 또한, 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 가지 목표를 두고 진행을 하였다. 첫째는 숨기기 작업과정에서 손실될 수 있는 항목을 최소화하고, 둘째는 숨기기 한 후의 RTDB의 품질상태를 최적으로 유지하는 것이다.

2.2 숨기기의 기본 전략

앞 절 선행 연구들은 SFIS를 숨기기 위한 빈발패턴과 그에 따른 연관 규칙을 발견하고 지지도와 신뢰도 값을 기본으로 하는 휴리스틱 알고리즘이 대부분이었다. [10]의 연구에서는 데이터베이스 D에서 규칙 X→Y가 주어질 때 숨기기 전략으로 두 가지 방법을 소개하고 있다. 첫째, 규칙에서 X의 지지도를 증가하고 Y의 지지도를 감소하여 규칙의 신뢰도를 감소시킨다. 둘째, 규칙에서 X, Y 지지도 중에서 어느 한 항의 지지도를 감소 시켜서 전체적인 지지도를 감소시킨다. 선행연구와 [10]의 숨기기 전략을 기반으로 아래와 같이 신뢰도와 지지도를 이용한 전략을 다음과 같이 네 가지로 기술할 수 있고, 본 논문의 HSFI 알고리즘은 아래 (2)의 숨기기 알고리즘 기반이라 할 수 있다.

(1) 규칙의 신뢰도가 최소 임계치 아래로 감소될 때까지 항목 지지도를 증가시켜 숨긴다.
(2) 규칙의 신뢰도, 지지도 중 어느 한 쪽이 최소 임계치 아래로 감소될 때까지 항목 지지도를 감소시켜 숨긴다.
(3) 규칙의 신뢰도 또는 지지도 중 어느 한 쪽이 최소 임계치 아래가 될 때까지 민감 규칙의 신뢰도를 감소시켜 숨긴다.
(4) 지지도가 최소 지지도 임계값 이하까지 생성된 항목집합들의 지지도 감소에 의해 규칙들을 숨긴다.

위 모든 전략에서의 문제점은 숨기기 과정에서 발생하는 항목의 손실이다. 이로 인하여 결과에서 생각지 않은 연관 규칙이 삭제되는 경우가 발생하게 되는 것이다. Apriori 원리[11]에 의하면 만약, 한 항목집합이 빈발하면, 그것을 포함하고 있는 부분집합들 역시 빈발해야만 한다. 따라서 SFIS의 한 항목집합이 민감하기 때문에 삭제되어야 한다면 SFIS의 항목집합을 포함하고 있는 NSFIS의 부분집합은 민감하므로 일부가 삭제되어야 하는 것이다. 손실의 문제는 최종 숨기기 과정 후 RTDB 품질과 연관되어 있기 때문에 아주 중요한 문제 중에 한 부분이라 할 수 있다.

이러한 문제를 해결하기 위하여 본 논문의 HSFI 알고리즘에서는 경계 접근법[5]을 이용하였다. 경계는 SFIS와 NSFIS의 항목들의 상대적인 지지도와 Apriori 원리에 의한 항목집합을 고려하고 있기 때문에 경계의 요소 항목들의 상대적인 지지도를 변경하

면서 숨기기 과정이 이루어진다. 또한, 숨기는 과정에서의 경계 요소 항목의 영향을 평가하여 손실을 최소화하고 RTDB의 품질 상태를 최적화 할 수 있다.

2.3 RFIS와 RTDB의 품질

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 개념은 다음과 같다.

TDB의 항목들 중에서 최소 지지도 임계치 이상의 항목집합을 FIS라 하고, 이를 구성하는 항목집합은 SFIS와 NSFIS로 이루어진다. 이 중에서 민감 데이터를 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 1과 같이 TDB의 FIS 중에서 SFIS를 숨기기 한 후 결과 NSFIS이며, 이는 RTDB의 FIS인 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 원리에 의하여 숨기지 않아야 하는 항목집합의 손실 부분이 발생하는 것이며 결과적으로 RTDB는 |FIS-SFIS-X| (X : SFIS를 포함하는 부분 항목집합)가 되어야 옳을 것이다. 따라서 본 논문에서는 항목의 손실을 최소화하기 위하여 SFIS와 NSFIS 사이에 경계를 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. FP-Tree와 FIS, SFIS, NSFIS

FIS를 찾아내는데 유용한 빈발 패턴 마이닝 알고리즘으로 Apriori 알고리즘[12]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. 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]. 본 장에서는 FP-Tree에 대하여 간략하게 살펴보고 HSFI 알고리즘에 효과적으로 활용할 수 있는 트랜잭션 리스트기반의 FP-Tree에 대하여 기술한다. 그리고 예시 TDB를 이용하여 FIS, SFIS, NSFIS 등을 간단하게

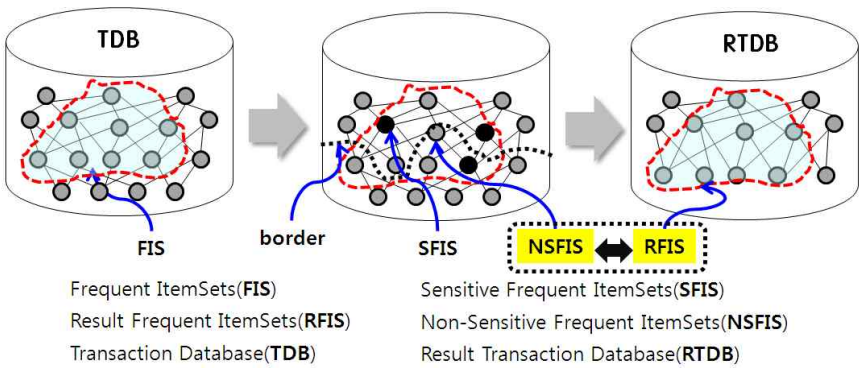

그림 1. TDB(FIS, SFIS, NSFIS)와 RTDB(RFIS)

살펴본다.

3.1 트랜잭션기반 FP-Tree와 FIS생성

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 2와 같으며, 지지도 3을 기준으로 표 1의 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.

아래 표 1의 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.

트랜잭션기반 FP-Tree는 FP-Tree와 유사하지만 트리를 구성하고 있는 노드는 FP-Tree와 차이를 보인다. 트랜잭션기반 FP-Tree의 노드 구성은 항목명과 <T_i, S_i, B_i>로 표현된다. 여기서 T_i는 해당 항목

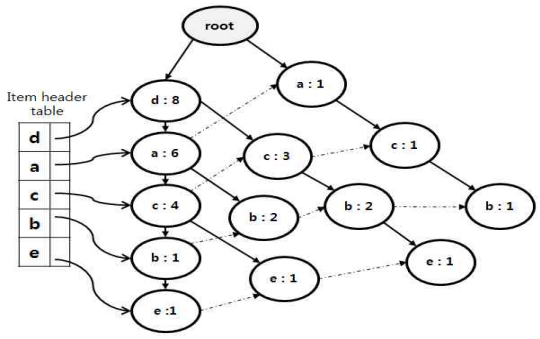

그림 2. 예시 TDB를 이용한 FP-Tree

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

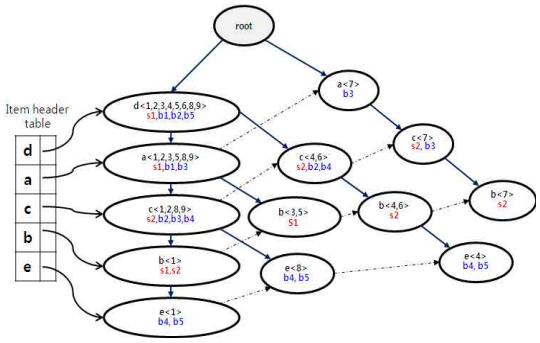

그림 3. 트랜잭션기반 FP-Tree

을 포함하고 있는 트랜잭션 리스트(T_1, T_2,…T_n)를 의미하며 FP-Tree 초기 생성과정 노드에 기록된다. S_i는 숨기기 위한 민감 항목, B_i는 경계 요소 항목을 의미한다. 트랜잭션기반 FP-Tree의 예시는 그림 3과 같으며, 지지도 3을 기준으로 표 1의 예시 TDB로부터 생성되었다. 트랜잭션기반 FP-Tree 노드는 트랜잭션 정보, 민감 정보, 경계 정보를 구성하고 있기 때문에 숨기기 작업과정에서 데이터베이스를 반복 스캔할 필요가 없다.

위 그림 3의 트랜잭션기반 FP-Tree를 참고하여 빈발 항목별 트랜잭션 리스트를 정리하면 아래 표 2와 같이 기술할 수 있다.

표 2. 항목별 트랜잭션 리스트

item	frequent	Transaction ID lists
d	8	T_1,T_2,T_3,T_4,T_5,T_6,T_8,T_9
a	7	T_1,T_2,T_3,T_5,T_7,T_8,T_9
c	7	T_1,T_2,T_4,T_6,T_7,T_8,T_9
b	6	T_1,T_3,T_4,T_5,T_6,T_7
e	3	T_1,T_4,T_8

3.2 FIS와 경계기반 R_SFIS, R_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 3의 결과 트리에서 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 패턴 베이스를 구성한다. 항목 e의 CFP-Tree는 주어진 최소 지지도 임계값보다 작은 경로를 제외한 후, 단일 경로의 빈발패턴인 모든 조합 ((dc:3), (d:3), (c:3))|e를 생성한다. 나머지 항목에 대한 방법은 재귀호출을 통하여 구한 후 표 3과 같은 최종적인 FIS를 구할 수 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 3은 예시 TDB로부터 생성된 FIS와 NSFIS를 보여주고 있다. SFIS는 입력 항목집합이다.

Apriori 원리에 의해 SFIS의 부분집합에 속하는 NSFIS 일부가 숨기는 과정에서 손실되는 경우가 발생하여 RTDB의 RFIS 결과에 영향을 주게 된다. 이러한 과정에서 삭제의 영향을 받게 되는 NSFIS의 모든 항목집합에 대해서 비교, 평가하는 것은 비효율적이라는 문제점이 있다. 이러한 문제를 해결하기 위해서 서론에서 소개한 경계를 이용한다. 경계는 NSFIS 중에서 SFIS와 포함관계에 의하여 표현되는 것으로 항목집합 서로 간에 밀집한 관계를 가지게 된다. 따라서 SFIS의 숨기는 작업과 손실을 줄일 수 있는 항목의 영향 평가를 경계요소 항목에 대해서만 하는 것이다. 그림 4는 SFIS와 NSFIS의 상대적인 관계를 경계를 통하여 보여주고 있다.

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

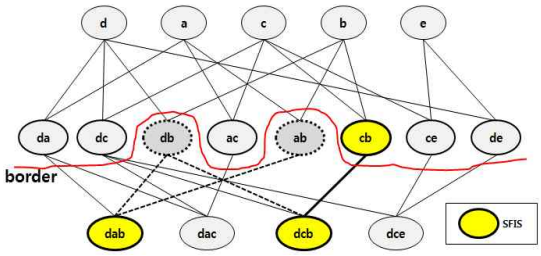

그림 4. FIS에서의 경계 구분

위 그림 4에서 모든 타원형의 항목집합들은 FIS이며, 굵은선 타원은 SFIS이다. FIS에서 SFIS를 제외한 나머지 항목집합은 NSFIS이고 이 중에서 SFIS와 완전 포함관계에 있는 항목집합은 점선의 타원인 {db, ab}이다. 이 항목집합은 SFIS와 포함관계에 있기 때문에 영향 평가에는 제외되며 그 외 나머지 항목집합 {da, dc, ac, ce, de}은 경계요소 항목집합이 된다. 경계요소 항목집합의 경우 SFIS 항목집합 중에서 부분 포함관계이고 숨기기 작업에서 손실이 발생할 수 있는 항목집합이기 때문에 SFIS 숨기기 작업에서 영향 평가의 대상된다. 숨기기 작업의 대상이 되는 SFIS에서 {dcb}의 경우 {cb}와 완전 포함관계에 있기 숨기기 대상에서 제외가 된다. 따라서 경계요소의 항목집합은 아래 표 4와 같이 {dab, cb}만 대상이 되며 굵은 테두리 부분이다.

표 4. SFIS 트랜잭션 리스트

item	frequent	TIDS
dab	3	T_1,T_3,T_5
dcb	3	T_1,T_4,T_6
cb	4	T_1,T_4,T_6,T_7
⇓		
dab	3	T_1,T_3,T_5
cb	4	T_1,T_4,T_6,T_7

숨기기 과정에서 영향 평가의 대상이 되는 NSFIS 경계요소 항목집합은 {da, dc, ac, ce, de}이며 항목집합이 포함하고 있는 트랜잭션 리스트는 아래 표 5와 같이 기술할 수 있다.

표 4와 표 5에서 최종적인 SFIS, NSFIS의 상대적인 경계요소를 R_SFIS(Relative Sensitive Frequent Itemsets), R_NSFIS(Relative Non Sensitive Frequent Itemsets)이라 표기하고 해당 항목집합은 R_SFIS={dab, cb}, R_NSFIS={da, dc, ac, ce, de}와 같이 나타낼 수 있다.

표 5. NSFIS 트랜잭션 리스트

item	frequent	TIDS
da	6	T_1,T_2,T_3,T_5,T_8,T_9
dc	6	T_1,T_2,T_4,T_6,T_8,T_9
ac	5	T_1,T_2,T_7,T_8,T_9
ce	3	T_1,T_4,T_8
de	3	T_1,T_4,T_8

4. 경계기반 상대적 지지도를 이용한 SFIS 숨기기

Apriori 원리에 의해면 SFIS의 부분집합에 속하는 NSFIS 일부가 숨기는 과정에서 손실되는 경우가 발생하여 RTDB의 RFIS 결과에 영향이 있다는 것을 앞 절을 통하여 살펴보았다. 이 과정에서 삭제 손실이 있는지를 파악하기 위하여 SFIS와 포함관계에 있는 NSFIS의 모든 항목에 대해서 비교 평가하는 것은 비효율적인 NP-hard 문제이다. 본 논문에서는 이러한 NP-hard의 문제점을 효율적으로 해결하기 위하여 Apriori 원리를 활용한 경계를 이용하였다. 본 장에서는 SFIS와 NSFIS의 경계요소의 영향 평가와 제안하는 HSFI 알고리즘에 대하여 기술하고자 한다.

4.1 경계요소와 영향 평가(impact)

앞 절에서 생성된 경계요소인 R_SFIS와 R_NSFIS를 포함하고 있는 트랜잭션을 비트 패턴으로 표현한 테이블은 표 6과 같으며, s_{ij}는 R_SFIS의 항목이며 숨기기 위한 SFIS의 후보항목을 의미한다. 여기서 i는 R_SFIS의 요소인 s_i의 순번이며, j는 s_i가 포함하고 있는 항목 순번을 의미한다. 예를 들어 s_{12} 경우는 R_SFIS={dab, cb}의 첫 번째 경계요소 s_1(abd)에서 두 번째 항목을 의미하며 결과 s_{12}={b}가 된다. b_{ij}는 R_NSFIS의 항목집합이며, i는 R_NSFIS의 b_i의 순번이며, j는 해당 b_i가 포함하고 있는 항목 순번을 의미한다. 예를 들어 b_{22}의 경우는 R_NSFIS={da, dc, ac, ce, de}의 두 번째 경계요소 b_2(dc)에서 두 번째 항목을 의미하며 결과 b_{22}={c}가 된다.

SFIS 숨기기는 R_SFIS={dab, cb}에 대한 impact 값을 구하고 그 결과 중에서 최소 impact 값을 찾아서 해당 트랜잭션의 항목을 삭제하는 것이다. 여기서 impact 값은 R_SFIS 후보항목을 포함하고 있는 R_NSFIS의 지지도를 적용한 가중치(weight)를 구한 후 항목들의 가중치 전체 합으로 구할 수 있으며, 가중치는 R_SFIS와 R_NSFIS의 상대적 지지도를 유지하기 위함이다. 가중치는 숨기기 과정동안 변경되는 지지도에 의해서 다시 계산하여 반영된다. 가중치 합에 의한 impact 값이 작을수록 NSFIS 손실이 최소화되고 숨기게 될 확률이 높은 SFIS가 되는 것이다. [5]의 연구에 의하면 impact 값을 구하기 위한 R_NSFIS 항목집합 B의 가중치(w(B))는 아래 조건

표 6. R_SFIS와 R_NSFIS의 트랜잭션 비트 패턴 테이블

s_{ij}	R_SFIS	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	supp(S)
s_1	dab	1		1		1					3
s_2	cb	1			1		1	1			4

b_{ij}	R_NSFIS	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	supp(B)
b_1	da	1	1	1		1			1	1	6
b_2	dc	1	1		1		1		1	1	6
b_3	ac	1	1					1	1	1	5
b_4	ce	1			1				1		3
b_5	de	1			1				1		3

에 의해서 구해진다.

if Psupp(B)≥(m+1) then // m(최소 지지도 임계값)
w(B)=supp(B)-Psupp(B)+1 / (supp(B)-m);
if 0≤Psupp(B)≤m then
w(B)=n+m-Psupp(B); // 0≤n≤∞(n : 임의의 큰 정수)

여기서 제시된 TDB에서 R_NSFIS의 경계요소 항목집합 B의 지지도는 supp(B)이고, 수정 처리 중인 TDB를 PTDB이라하고, Psupp(B)는 PTDB의 지지도이다.

4.2 SFIS 숨기기 알고리즘: HSFI(Hiding Sensitive Frequent Itemsets)

본 논문의 SFIS 숨기기 작업은 가중치 합에 따른 impact 값을 계산하여 R_NSFIS에 최소의 영향을 주는 R_SFIS의 항목을 찾아서 삭제하는 것이다. SFIS의 항목집합의 숨기기 후보항목을 s_i라 하고, $w(b_i)$는 각 $b_i \in R_NSFIS|_s$의 가중치라 한다면 impact는 $I(T_i, s_{ij}) = \sum w(b_i)$와 같이 계산할 수 있다. 여기서 T_i는 민감한 항목 s_{ij}를 포함하고 있는 트랜잭션을 말한다. impact 값에 의한 민감한 항목 삭제를 위한 알고리즘은 그림 5와 같다.

```
Input :
    TDB : Transaction Database;
    m : Minimal Support Threshold;
    SFIS : Sensitive Frequent Itemsets;
Output :
    RTDB : Result Transaction Database;
Method :
01:  Compute FIS and FIS transaction list;
02:  Compute NSFIS and SFIS, NSFIS transaction list;
03:  Compute border itemset of NSFIS, SFIS;
04:  Compute R_SFIS, R_NSFIS tansaction list;
05:  for each Xi∈R_SFIS do
06:      Compute R_NSFIS|s  and w(bj) where bj∈R_NSFIS|s;
07:      Initialize C(C is the set of hiding candidate of X);
08:      until(supp(X)<m) do
09:          Find ui=(Ti, xi) such that I(ui)=Min {I(u)|u∈C};
10:          Update C=C- {(T,x)|T=Ti};
11:          Update w(bj) where bj∈R_NSFIS|s;
12:      end
13:  end
14:  Update database TDB;
15:  Output RTDB=TDB;
```

그림 5. HSFI 알고리즘

위 그림 5에서 기술된 HSFI 알고리즘의 숨기기 과정을 간략하게 살펴보면 다음과 같다.

01~04 라인에서는 숨기기 작업에 필요한 항목집합을 구하기 위한 선수작업으로 TDB에서 최소 지지도 임계값 이상인 FIS로 구성된 트랜잭션 리스트를 구성하고, 결과 FIS에서 입력된 SFIS를 제외한 나머지 항목의 NSFIS 트랜잭션 리스트를 구한다. Apriori 원리에 따라 만들어진 트랜잭션 리스트 중 SFIS 항목의 포함 관계에 따른 R_NSFIS의 트랜잭션 리스트와 R_SFIS 리스트를 구성한다. 05~13 라인까지 impact 값을 구하여 민감한 항목을 숨기기하는 반복 작업 부분이다. 구체적으로 기술하면, 먼저 R_NSFIS의 가중치 $w(b_j)$를 계산하고 R_SFIS의 숨기기 후보 집합을 초기화한다. 이후 R_SFIS의 민감한 항목 s_{ij}를 포함하고 있는 모든 R_NSFIS의 impact인 $I(T_i, s_{ij})$를 계산한다. 각 T_i별 impact 결과 중에서 가장 작은 impact의 $I(T_i, s_{ij})$를 찾은 다음 숨기기 후보 집합에서 s_{ij}를 삭제한다. 만약, 결과 중에서 동일한 값의 impact $I(T_i, s_{ij})$가 여러 개 있는 경우에 트랜잭션 크기가 큰 것을 선택한다. 트랜잭션 크기까지 같을 경우 민감한 항목 s_{ij}의 지지도가 큰 것을, 둘 다 같은 경우는 트랜잭션 순서대로 선택한다. 숨기기 작업 후에 항목의 변동으로 인하여 R_NSFIS 가중치 $w(b_i)$를 다시 갱신한다. R_SFIS의 지지도가 임계값 아래가 될 때까지 08번 라인에서부터 반복 수행한다. 알고리즘의 마지막 14~15 라인은 R_SFIS의 지지도가 임계값 아래가 되면 최종 TDB를 수정하고, 수정된 결과인 TDB는 숨기기 후의 RTDB가 된다.

위 표 6의 R_NSFIS, R_SFIS 비트 패턴 테이블과 그림 5의 알고리즘을 이용하여 숨기기 항목으로 R_SFIS의 첫 번째 경계 항목집합인 {dab}를 실례로 적용고자 한다.

① 민감한 빈발항목 s_i의 항목 s_1(dab)을 포함하는 트랜잭션 리스트를 구한다. 결과는 T_1,T_3,T_5 이다.
② 각 T_i별 b_i 리스트를 구한다.
T_1 : b_1,b_2,$b_3$$b_4$,$b_5$,$b_6$ T_3 : b_1 T_5 : b_1
③ 각 b_i들의 가중치 $w(b_i)$를 계산한 후, s_i의 각 항목별 impact $I(T_i, s_{ij})$를 계산한다.
i) 가중치 $w(b_i)$를 구한다.
- $W(b_1::(da:6))=(6-6+1)/(6-3)=1/3$
- $W(b_2::(dc:6))=(6-6+1)/(6-3)=1/3$

- $W(b_3::(ac:5))=(5-5+1)/(5-3)=1/2$
- $W(b_4::(ce:3))=n+3-3=n$
- $W(b_5::(de:3))=n+3-3=n$

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

- $I(T_1, d)==w(da)+w(dc)+w(de)=1/3+1/3+n$
- $I(T_1, a)==w(da)+w(ac)=1/3+1/2$
- $I(T_1, b)==$ null
- $I(T_3, d)==w(da)=1/3$
- $I(T_3, a)==w(da)=1/3$
- $I(T_3, b)==$ null
- $I(T_5, d)==w(da)=1/3$
- $I(T_5, a)==w(da)=1/3$
- $I(T_5, b)==$ null

④ ③의 impact $I(T_i, s_{ij})$ 결과 중에서 가장 작은 $I(T_3, d)$, $I(T_3, a)$, $I(T_5, d)$, $I(T_5, a)$중 하나를 선택한다.

⑤ Impact 값이 동일한 경우는 우선 트랜잭션 크기가 큰 것을 선택하여 삭제한다. 트랜잭 션 크기까지 같을 경우 해당 s_{ij}의 지지도가 큰 것을, 둘 다 같은 경우는 트랜잭션 순서대로 선택한다. 현재 결과에서는 트랜잭션의 크기가 크면서 민감한 항목 s_{ij}의 지지도가 큰 $I(T_3, d)$를 삭제를 수행한다.

⑥ T_3에서 d를 삭제한 후 가중치 $w(b_i)$를 수정하고, 트랜잭션 데이터베이스를 수정한다.

⑦ R_SFIS의 지지도가 임계값 아래가 될 때까지 ①부터 반복 수행한다.

그림 6은 HSFI 알고리즘을 적용한 숨기기 후의 반영된 결과를 보여주고 있다.

그림 6. R_SFIS={dab, cb} 숨기기 후 반영된 결과

5. 프로그램 적용과 성능 평가

5.1 프로그램 적용 후의 RTDB 품질평가

본 논문에서는 빈발 항목 중에서도 민감한 데이터를 가지고 있는 SFIS를 숨기기 위해 FIS와 NSFIS 등을 추출하고 NSFIS에 대한 영향(impact)을 줄이기 위해 경계를 이용하였다. 상대적인 지지도를 유지하기 위하여 경계요소 항목집합인 R_NSFIS와 R_SFIS를 구하고 이를 이용하는 HSFI 알고리즘을 제안하였다. 본 논문의 표 1의 예시 TDB를 이용하여 SFIS를 숨기기 한 후 RTDB로 변경되었을 때의 최종적인 품질상태를 비교, 평가하기 위하여 프로그램을 구현하였다. 민감한 데이터를 포함하고 있는 SFIS를 숨기기 한 후, TDB의 NSFIS와 RTDB의 RFIS를 비교하면 RTDB의 RFIS는 SFIS를 부분집합을 포함하는 NSFIS를 제외한 것이 되는 것이며, 결과적으로 RTDB의 RFIS는 표 7과 같이 |FIS-SFIS-X| (X 는 SFIS를 포함하는 부분 항목집합)가 된다.

예시 TDB의 적용과 기본적인 실험으로 RTDB의 결과를 확인하고 연관 규칙을 평가하기 위하여 그림 7과 같이 프로그램을 구현하였으며, 표 1의 예시 TDB는 9개 트랜잭션과 8개 항목으로 구성하여 프로그램에 적용하였다. 초기 신뢰도와 지지도는 각각 80%와 30%를 기준으로 실험하였다. 구현된 프로그램을 통하여 숨기기 후의 손실 정보를 확인하였다.

표 7. SFIS 숨긴 후 TDB와 RTDB의 결과 비교

	TDB
FIS	a:7, b:6, c:7, d:8, e:3 de:3, ce:3, cd:6, bd:5, bc:4, ad:6, ac:5, ab:4, abd:3, acd:4, bcd:3, cde:3
SFIS	abd:6, bcd:3, bc:4
NSFIS	a:7, b:6, c:7, d:8, e:3 de:3, ce:3, cd:6, bd:5, ad:6, ac:5, ab:4, acd:4, cde:3

⇒

	RTDB
RFIS	a:7, b:4, c:7, d:7, e:3 de:3, ce:3, cd:6, bd:3, ad:5, ac:5, ab:3, acd:4, cde:3

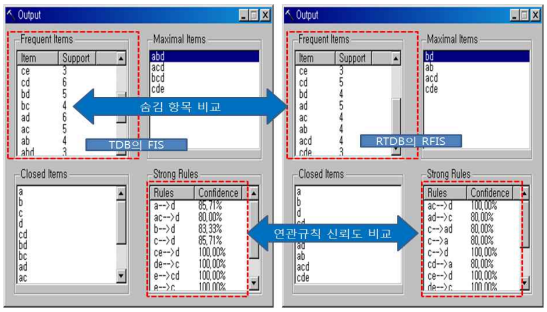

그림 7. 프로그램 적용 후의 TDB의 FIS와 RTDB의 RFIS 결과비교

실험 결과 숨기기 과정 발생하는 손실항목의 비율이 낮아 항목 숨기기의 정확성이 높다는 것을 알 수 있었고 연관규칙의 신뢰도가 숨기기 전보다 크게 향상된 것을 볼 수가 있었다. 위 그림 7의 구현 프로그램 화면에서 점선 사각형 영역은 숨기기 전 TDB의 NSFIS와 숨기기 후 RTDB의 RFIS의 항목집합 비교 결과화면이며, 하단에 연관규칙 신뢰도 부분을 확인할 수 있다.

5.2 성능 평가

제안된 HSFI 알고리즘을 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[14]를 이용하였다. 실험의 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 5개, 10개의 조건으로 생성된 TDB에 적용하였으며, 그 결과의 값들을 비교하였다. 비교되는 실험 결과는 숨기는 과정에서 손실되는 항목수와 수정되는 항목수, TDB의 FIS와 RTDB의 RFIS이다. 그림 8은 SFIS가 5개인 경우의 실험 결과 테이블이며, 그림 9는 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	435.7	3	129
10000	417	414	3.1	295.2
15000	426	421.5	3	498
20000	442	439.2	3.4	696.3
25000	432	427	3.6	887.4
30000	443	438.4	3.2	848.8

그림 8. |SFIS|=5에 대한 실험 적용결과

TDB	FIS	RFIS	#loss itemsets	#modified entries
5000	439	433.1	4.6	204.6
10000	417	419	6.5	585.4
15000	426	416	5.9	852.6
20000	442	434	6	1106
25000	432	421.6	5.9	1392.6
30000	443	433.8	6.8	1493.2

그림 9. |SFIS|=10에 대한 실험 적용결과

그림 10은 본 논문의 HSFI 알고리즘과 선행연구인 DSR 알고리즘[8], Association Rule 알고리즘[9]에 이전 생성된 TDB와 SFIS 항목 5개를 적용하여 손실된 항목의 비교 결과를 나타내고 있다.

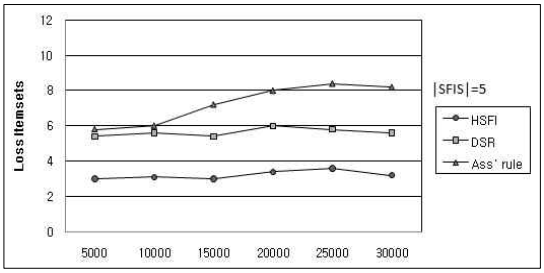

그림 10. |SFIS|=5에 대한 손실항목 비교 결과

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 HSFI 알고리즘을 이용할 경우 DSR 알고리즘보다 손실 항목이 많이 줄었으며, TDB 크기에 따른 손실 항목의 수에 큰 차이가 없음을 알 수 있었다. 따라서 HSFI 알고리즘을 이용한 후의 RTDB 품질상태가 더 개선되었음을 알 수 있었다.

6. 결 론

최근 개인 정보의 민감한 정보를 포함한 정보를 빈번하게 마이닝 될 수 있다는 문제점을 가지게 되었다. 따라서 민감하고 비밀스러운 데이터가 있다면 이를 제공하기 전에 데이터베이스를 수정하여 마이닝 하지 못하도록 미리 숨기는 것이 필요하게 되었다. 기존 연구들은 민감한 숨기기 작업에서 Apriori 원리에 의해 항목 손실이 생기는 문제점 있어 RTDB 품질상태에 부작용이 나타났다.

본 논문에서는 기존 연구의 문제점인 숨기기 작업 과정에서 손실될 수 있는 항목을 최소화하고 RTDB의 품질상태를 최적으로 유지하기 위한 HSFI 알고리즘을 제안하였고, NP-hard의 문제점을 해결하기 위하여 경계를 하여 SFIS와 NSFIS의 경계요소 항목 집합들의 상대적인 지지도를 기반으로 숨기기 작업이 이루어졌다. HSFI 알고리즘은 민감 정보 숨기기 과정에서의 최대 문제인 항목손실의 문제점을 개선하여 부작용을 줄이고 최종적인 RTDB의 품질상태를 향상 시킬 수 있었다. 본 논문에서 제안하는 알고리즘의 성능 평가를 위하여 프로그램을 구현하였으며 기본적으로 논문에서 제시하는 예시 TDB를 적용하여 논문에서 기술되어진 이론의 정확함을 보였으며, 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 HSFI 알고리즘을 적용한 RTDB 품질상태가 개선되었음을 입증하였다. 또한 숨기기 후 항목들 간의 연관규칙 및 신뢰도의 향상을 프로그램 구현 및 실행을 통하여 검증하였다.

참 고 문 헌

[1] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, “State-of-the-Art in Privacy Preserving Data Mining,” *SIGMOD Record*, 33(1), pp. 50-57, 2004.

[2] J. Han, J. Pei, Y. Yin, and 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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[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] 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.

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

[13] 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.

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

이 단 영

1994년 울산대학교 교육대학원 전자계산학과(교육학 석사)
2002년 울산대학교 대학원 컴퓨터정보통신공학부(박사과정 수료)
2002년~2004년 울산대학교 IT 교육원 객원교수
2006년~2009년 울산대학교 컴퓨터정보통신공학부 객원교수
2011년~현재 울산대학교 전기공학부 외래강사
관심분야 : 데이터마이닝, DB분석/설계, ERP, e-Business

안 형 근

2003년 울산대학교 정보통신대학원 정보통신공학과(공학석사)
2008년 울산대학교 대학원 컴퓨터정보통신공학부(공학박사)
1997년~2004년 현대오토시스템 기술지원부
2004년~2006년 (주)CFIC 기업부설연구소 연구소장
2008년 ~2010년 울산대학교 컴퓨터정보통신공학부 객원교수
2011년~현재 울산대학교 전기공학부 외래강사
관심분야 : 멀티미디어DB, DB설계/분석, ERP, BPM, Workflow

고 재 진

1972년 서울대학교 응용수학과(공학사)
1981년 서울대학교 대학원 계산통계학과(이학석사)
1990년 서울대학교 대학원 컴퓨터공학과(공학박사)
1975년~1979년 한국후지쯔(주) 기술개발부 사원
1979년~2010년 울산대학교 컴퓨터정보통신공학부 교수
2011년~현재 울산대학교 전기공학부 교수
관심분야 : DB시스템, 전문가 시스템, DB설계, ERP