대용량 공간 데이터로 부터 빈발 패턴 마이닝
(Mining Frequent Pattern from Large Spatial Data)

이 동 규* 이 경 민* 정 석 호* 이 성 호** 류 근 호***
(Dong Gyu Lee) (Gyeong Min Yi) (Jung Suk Ho) (Seong-Ho Lee) (Keun Ho Ryu)

요 약 공간 및 비 공간 데이터에서 알지 못했던 패턴을 탐사하는 빈발 패턴 탐사 기법은 마이닝 분야에서 가장 핵심적인 부분으로 많은 연구가 활발히 진행되고 있다. 기존의 자료구조들은 트리 구조 및 배열 구조로써 밀집 또는 희소 빈발 패턴에서 성능 저하를 보인다. 대용량의 공간 데이터는 밀집 및 희소 빈발 패턴을 둘 다 가지므로 단일 알고리즘으로 빠르게 탐사 하는 것은 중요하다. 본 논문에서는 단일 알고리즘을 사용하면서도 밀집 및 희소 빈발 패턴 모두에 대해 빠르게 빈발 패턴을 마이닝할 수 있는 압축된 패트리샤 빈발 패턴 트리라는 새로운 자료구조와 이를 사용한 빈발 패턴 마이닝 알고리즘을 제안한다. 실험 평가는 제안한 알고리즘이 대용량 희소 및 밀집 빈발 데이터에서 기존의 FP-Growth 알고리즘 보다 약 10배 정도 빠르게 빈발 패턴을 탐사하는 것을 보인다.

키워드 : 공간 데이터 마이닝, 데이터 마이닝, 빈발 패턴 마이닝, FP-Growth

Abstract Many researches of frequent pattern mining technique for detecting unknown patterns on spatial data have studied actively. Existing data structures have classified into tree-structure and array-structure, and those structures show the weakness of performance on dense or sparse data. Since spatial data have obtained the characteristics of dense and sparse patterns, it is important for us to mine quickly dense and sparse patterns using only single algorithm. In this paper, we propose novel data structure as compressed patricia frequent pattern tree and frequent pattern mining algorithm based on proposed data structure which can detect frequent patterns quickly in terms of both dense and sparse frequent patterns mining. In our experimental result, proposed algorithm proves about 10 times faster than existing FP-Growth algorithm on both dense and sparse data.

Keywords : Spatial Data Mining, Data Mining, Frequent Pattern Mining, FP-Growth

1. 서론

최근 GIS의 급속한 발달로 인하여 많은 공간 데이터가 축척되고 있다. 축척된 공간 데이터베이스의 공간 데이터와 비공간 데이터에서 알려지지 않은 새로운 지식을 탐사하는 공간 데이터 마이닝에 관한 연구가 활발히 진행되고 있다.

특히 공간 데이터와 비 공간 데이터에서 패턴을 탐사하는 빈발 패턴 마이닝은 마이닝 분야에서 가장 핵심적인 부분으로 많은 연구가 활발히 진행되고 있다. 빈발 패턴 마이닝은 상호관계(correlations), 인과관계(causality), 순차 패턴(sequential patterns), 에피소드(episodes), 다차원 패턴(multi-dimensional patterns), 최대 패턴(max-patterns), 부분 주기성(partial periodicity), 신흥 패턴(emerging patterns) 등 일반적인 데이터 마이닝과 공간 데이터 마이닝 분야에서 다양하게 활용되고 있다[1, 2, 3, 4, 5, 6, 7, 8].

빈발 패턴 마이닝은 트랜잭션 데이터베이스에서 전체 트랜잭션 개수 대비 패턴이 발생한 트랜잭션 개수 비율이 특정 지지도 임계값보다 큰 지지도를 갖는 모든 패턴을 탐사하는 것이다.

본 논문에서는 단일 알고리즘을 사용하면서도 밀집 및 희소 빈발 패턴 모두에 대해 효율적이고 빠르게 빈발 패턴을 마이닝할 수 있는 압축된 패트리샤 빈발 패턴 트리(이하 CPFP-Tree라 칭함)라는 새로운 자료구조와 이를 사용한 빈발 패턴 마이닝 알고리즘을 제안한다.
본 논문에서는 제안하는 CPFP-Tree를 이용한 빈발 패턴 마이닝 알고리즘이 기존의 알고리즘들 보다 더 나은 성능을 보인다는 것을 실험 평가를 통하여 확인하였다. 실험 평가는 제안하는 알고리즘과 기존의 대표적인 빈발

† 이 논문은 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(과제번호:07국토정보C05)과 2009년 정부(교육과학기술부)의 지원 (지역거점연구단육성사업/충북BIT연구중심대학육성사업단)에 의해 수행되었습니다.

* 충북대학교 데이터베이스연구실, {dglee, min9709, sukhojung}@dblab.chungbuk.ac.kr
** 한국전자통신연구원(ETRI : Electronics and Telecommunications Research Institute), sholee@etri.re.kr
*** 충북대학교 전자정보대학 컴퓨터공학부 교수, khryu@dblab.chungbuk.ac.kr(교신저자)

논문접수 : 2010.01.26 수정일 : 1차 2010.03.04 / 2차 2010.03.17 심사완료 : 2010.03.19

패턴 마이닝 알고리즘인 Apriori와 FP-Growth 알고리즘에 대하여 다양한 최소 지지도 변화에 따른 수행시간을 비교 하였다. 실험 평가에 사용한 데이터는 빈발 패턴 마이닝 실험에서 많이 사용되는 T10I4D100K와 Chess 데이터를 사용하였다. 실험 평가를 통하여 제안하는 알고리즘이 기존의 FP-Growth 알고리즘 보다 약 10배정도 빠르게 빈발 패턴을 탐사하였다.

따라서 제안한 알고리즘은 빈발 패턴을 탐사하는 다양한 분야에 활용될 수 있다. 또한 제안한 알고리즘은 밀집 및 희소 빈발 패턴 모두를 빠르게 탐사하는 효과도 기대할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 빈발 패턴 알고리즘들에 대하여 배열기반과 트리기반으로 분류하여 설명한다. 3장에서는 새롭게 제안하는 CPFP-Tree 자료구조와 이를 사용한 빈발 패턴 탐사 기법에 대하여 자세히 설명한다. 4장에서는 성능평가를 통하여 기존 알고리즘들 보다 새롭게 제안하는 알고리즘이 더 우수함을 보이며, 5장에서는 결론에 대하여 논한다.

2. 관련 연구

빈발 패턴과 빈발 패턴 알고리즘을 분류하는 방법은 다양하다[3, 9, 10, 11, 12, 13]. 본 논문에서 빈발패턴은 일반적으로 많이 분류되는 방법으로 빈발 패턴의 특성에 따라 밀집도가 낮은 희소 빈발 패턴과 밀집도가 높은 밀집 빈발 패턴으로 분류한다. 빈발 패턴 마이닝 알고리즘은 사용하는 자료구조에 따라 배열기반 알고리즘과 트리기반 알고리즘으로 분류한다.

최초의 배열기반 알고리즘은 Apriori 알고리즘으로 후보 집합을 생성하여 빈발 패턴을 탐사하는 알고리즘이다[8]. 첫 번째 트랜잭션 데이터베이스 스캔에서 1빈발 패턴을 탐사한다. 탐사한 1빈발 패턴을 이용하여 2후보 빈발 패턴을 생성한다. 다시 트랜잭션 데이터베이스를 탐사하여 2후보 빈발 패턴을 점검하여 빈발하지 않은 패턴들의 제거를 통하여 2빈발 패턴을 생성한다. 생성된 2빈발 패턴을 이용하여 3후보 빈발 패턴을 생성한다. 이와 같은 과정을 반복하여 전체 빈발 패턴을 모두 탐사한다. 따라서 빈발 패턴의 최대 길이가 n이라면 n번 또는 n+1번의 전체 트랜잭션 데이터베이스의 탐사를 필요로 한다.

DHP 알고리즘은 Apriori 알고리즘을 확장한 알고리즘이다[15]. Apriori 알고리즘의 후보 패턴을 생성하는 과정에 해싱을 접목시켜 Apriori 알고리즘의 성능을 향상시켰다. 특히 2와 3후보 빈발 패턴을 생성하는데 많은 시간이 소요되는데 해싱을 이용하여 후보 패턴 생성 시간을 단축시켜 성능을 향상 시켰다.

H-Mine 알고리즘은 H-struct라는 자료구조를 이용한다[13]. H-struct 자료구조는 배열과 하이퍼링크(Hyper-link)의 조합들로 구성되며, 헤더 테이블과 빈발 투영도(Frequent Projection)로 이루어진다. H-Mine 알고리즘은 전체 트랜잭션 데이터베이스를 H-struct 자료구조로 변환한 후, 재귀적 탐사과정을 통해 전체 빈발 패턴을 탐사한다. 특히 H-Mine 알고리즘은 희소 빈발 패턴을 효율적으로 빠르게 탐사할 수 있다.

FP-Growth 알고리즘은 트리기반 알고리즘으로 FP-Tree라는 자료구조를 이용한다[14]. FP-Tree는 공통된 이전 항목들에 대해 같은 노드를 사용하는 전위(Prefix) 트리이다. FP-Growth 알고리즘은 전체 트랜잭션 데이터베이스를 FP-Tree 자료구조로 변환한 후, 재귀적 탐사과정을 통해 전체 빈발 패턴을 탐사한다. FP-Growth 알고리즘은 공통된 이전 항목들에 대해 같은 노드를 사용하여 밀집 빈발 패턴의 경우 매우 효율적이고 빠르게 빈발 패턴을 탐사한다.

CT-PRO 알고리즘은 FP-Tree를 확장한 CFP-Tree라는 자료구조를 이용한다[10]. FP-Tree의 노드가 카운트 정보만 가지는 것에 비해 CFP-Tree의 노드는 레벨과 카운트 정보를 가지고 있다. 따라서 공통된 이전 항목뿐만 아니 같은 레벨을 가지는 항목들에 대해 같은 노드가 사용된다. CT-PRO 알고리즘은 전체 트랜잭션 데이터베이스를 CFP-Tree 자료구조로 변환한 후, 재귀적 탐사과정을 통해 전체 빈발 패턴을 탐사한다. CT-PRO 알고리즘은 FP-Growth 알고리즘보다 트리의 크기를 축소시킬 수 있다. 따라서 밀집 빈발 패턴의 경우, 사용하는 메모리도 적고 효율적으로 빠르게 빈발 패턴을 탐사한다.
Patricia 알고리즘은 Patricia Tries라는 자료구조를 사용한다[11]. Patricia Tries는 압축된 접미트리로써 트리의 크기를 감소시켜 사용하는 메모리의 크기를 크게 줄였다. 특히 밀집 빈발 패턴의 경우 효율적으로 메모리의 크기를 줄인다.

트리기반 알고리즘들은 전체적으로 밀집 빈발 패턴을 효율적이고 빠르게 빈발 패턴을 탐사한다. 반면에 배열기반 알고리즘은 희소 빈발 패턴을 효율적이고 빠르게 빈발 패턴을 탐사한다.

밀집 및 희소 빈발 패턴 모두를 효율적으로 탐사 하고자 하는 최초의 시도는 OP(Opportunistic Projection) 알고리즘이다[12]. OP 알고리즘은 능동적으로 밀집도가 높으면 FP-Growth 알고리즘을 사용하고 밀집도가 낮으면 H-Mine을 사용한다. 따라서 밀집 및 희소 빈발 패턴 모두를 효율적으로 탐사한다. 그러나 밀집도에 대한 명확한 기준 제시가 어려우며, 밀집도가 높아 FP-Growth 알고리즘을 사용할 경우 FP-Growth 알고리즘의 단점을 내제하고, 밀집도가 낮아 H-Mine 알고리즘을 사용할 경우 H-Mine 알고리즘의 단점을 내제한다. 즉 2가지 종류의 알고리즘을 동시에 사용하므로 각각의 알고리즘을 사용할 정확한 기준 제시가 어렵고, 2가지 알고리즘의 단점을 내포한다.

밀집 및 희소 빈발 패턴 모두를 단일 알고리즘으로 빠르게 탐사 하는 것은 중요하다. 따라서 본 논문에서는 단일 알고리즘을 사용하면서도 밀집 및 희소 빈발 패턴 모두에 대해 효율적이고 빠르게 빈발 패턴을 마이닝할 수 있는 CPFP-Tree라는 새로운 자료구조와 이를 사용한 빈발 패턴 마이닝 알고리즘을 제안한다.

3. CPFP 알고리즘

본 논문에서 새롭게 제안하는 CPFP-Tree라는 자료구조와 이를 이용한 전체 빈발 패턴 마이닝 알고리즘의 이해를 돕기 위해서 표1의 간단한 샘플 트랜잭션 데이터베이스를 이용하여 단계별로 알고리즘을 설명한다. 샘플 트랜잭션 데이터베이스는 14개의 항목들로 총 5개의 트랜잭션으로 구성되어 있다. 빈발 트랜잭션은 원본 트랜잭션에서 최소 지지도 3을 만족하지 않는 항목들이 제거된 항목들로 이루어져 있다. 정렬된 트랜잭션은 빈발 트랜잭션에서 항목들의 빈발한 순으로 정렬된 트랜잭션으로 예를 들어 설명할 CPFP-Tree의 구성에 사용된다.

표 1. 샘플 트랜잭션 데이터베이스

TID	원본 트랜잭션	빈발 트랜잭션	정렬된 트랜잭션
1	2, 3, 4, 5, 7	2, 3, 4, 7	3, 2, 4, 7
2	1, 3, 4, 6, 7, 8	1, 3, 4, 7	3, 1, 4, 7
3	2, 3, 9, 10	2, 3	3, 2
4	1, 2, 3, 4, 7, 11, 14	1, 2, 3, 4, 7	3, 1, 2, 4, 7
5	1, 12, 13	1	1

제안하는 CPFP-Tree 자료구조를 이용한 빈발 패턴 마이닝 알고리즘은 크게 CPFP-Tree를 구성하는 단계와 구성된 CPFP-Tree를 이용하여 빈발 패턴을 탐사하는 단계으로 구성된다. CPFP-Tree를 구성하는 알고리즘은 그림1에 자세히 나타난다. CPFP-Tree를 이용하여 빈발 패턴을 탐사하는 알고리즘은 그림 8에 자세히 나타난다.

CPFP-Tree를 구성하는 과정은 원본 트랜잭션 데이터베이스 내의 모든 트랜잭션의 항목을 카운트하여 헤더 테이블을 구성하는 1단계(알고리즘 01-14)와 구성된 헤더 테이블을 이용하여 CPFP-Tree를 구성하는 2단계(알고리즘 15-23)로 이루어져 있다. CPFP-Tree 구성 알고리즘에서 트랜잭션 데이터베이스와 최소 지지도 임계값이 입력 값이고 CPFP-Tree가 출력 값이 된다. 헤더 테이블 구성 단계(알고리즘 01-14)에서는 먼저 트랜잭션 데이터베이스를 읽어 들인 후(알고리즘 02) 전체 트랜잭션을 탐사하면서 신규항목은 추가 시키고 기존 항목은 카운트를 증가시킨다(알고리즘 03-12). 구성한 헤더 테이블에서 빈발하지 않은 항목을(최소 지지도 임계값을 만족하지 못하는 항목) 제거하고(알고리즘 13) 빈발한 순으로 정렬한다(알고리즘 14).

CPFP-Tree 구성 단계(알고리즘 15-23)에서는 먼저 헤더 테이블을 이용하여 트리를 초기화하고(알고리즘 16), 트랜잭션 데이터베이스를 읽어 들인 후(알고리즘 17), 전체 트랜잭션을 탐사하면서 트리를 구성한다(알고리즘 18~23). 트랜잭션 t를 이용하여 노드의 분할여부를 판단한다(알고리즘 20). 분할이 일어나면 노드를 분할하고 해당노드의 트랜잭션 t의 레벨 카운트를 1증가 시킨다(알고리즘 21). 분할이 일어나지 않으면 트랜잭션 t의 항목들을 신규 노드로 추가한다(알고리즘 22).

```
Input:
    D: 트랜잭션 데이터베이스;
    Min_sup: 최소 지지도 임계값;

Output:
    CPFP: CPFP-Tree;

01: // 1단계: 헤더 테이블 구성
02: 트랜잭션 데이터베이스 D를 탐사;
03: foreach (트랜잭션 t in D)
04: {
05:     foreach (항목 i in t)
06:     {
07:         if (헤더 테이블에 항목 i의 존재)
08:             i의 카운트 1증가;
09:         else
10:             헤더 테이블에 항목 i추가;
11:     }
12: }
13: 헤더 테이블의 빈발하지 않은 항목들을 제거;
14: 항목의 순서를 내림차순으로 정렬;

15: // 2단계: CPFP-Tree 구성
16: 헤더 테이블을 이용하여 CPFP-Tree 초기화;
17: 트랜잭션 데이터베이스 D를 탐사;
18: foreach (트랜잭션 t in D)
19: {
20:     if (노드가 분할 가능)
21:         노드를 분할하고 t레벨 카운트 1증가;
22:     else t를 이용하여 신규노드 추가;
23: }
```

그림 1. CPFP-Tree 구성 알고리즘

헤더 테이블은 항목과 빈발 카운터의 리스트로 구성된다. 항목은 트랜잭션 데이터베이스의 각각의 항목이며 빈발 카운트는 각각의 항목이 나타난 횟수를 의미한다. 트리의 노드는 항목을 나타내며 추가적으로 레벨과 카운트를 가지고 있다. 레벨은 트랜잭션이 시작 항목의 헤더 테이블 위치를 나타내며 카운트는 트랜잭션 데이터베이스에서 항목이 나타난 횟수를 의미한다. 그리고 헤더 테이블의 각각의 항목은 항목을 포함하는 노드들의 링크를 가지고 있다.

초기화된 트리는 각각의 항목에 대하여 레벨0과 카운트 0을 가지는 단일 항목 노드들로 단일링크로 구성된 트리이다. 이와 같이 초기화된 트리를 구성하는 이유는 이후 트랜잭션 데이터를 이용하여 트리를 구성하는 과정을 원활하게 하기 위해서다. 트리의 구성과정에서 단순히 노드의 삽입만 발생하는 것이 아니라 경우에 따라서는 노드의 분할이 발생할 수 있다. 노드의 분할이 일어날 경우 노드들의 연결정보를 재구성 하여야하는데 초기화된 트리를 구성하여 이를 더 원활하게 수행할 수 있다. 이를 위하여 노드들의 연결정보를 헤더 테이블이 모두 가지고

있다. 이는 또한 노드의 초기 삽입위치를 헤더 테이블을 통하여 바로 알 수 있어 트리의 구성 시간을 줄여줄 수 있다. 이해를 돕기 위한 샘플 트랜잭션 데이터베이스를 이용하여 구성된 헤더 테이블과 초기화된 트리의 모습은 그림 2와 같다.

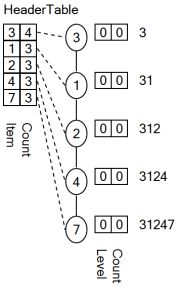


그림 2. 헤더 테이블 및 초기화된 CPFP-Tree

지금 부터는 트리의 구성 과정의 이해를 돕기 위해 샘플 트랜잭션 데이터베이스를 이용하여 예를 들어 설명한다. 각각의 트랜잭션들을 이용하여 어떻게 트리가 구성되고 분할되는지 순서대로 설명한다.

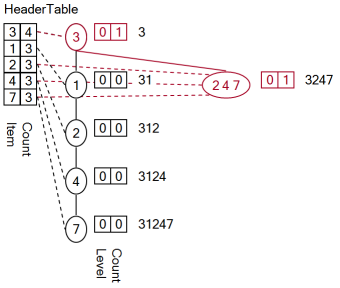


그림 3. TID1(3, 2, 4, 7) 삽입으로 신규 노드 생성

TID1(3, 2, 4, 7)이 트리에 삽입된 후의 트리구성은 그림3이다. TID1은 항목 3부터 시작하는 트랜잭션으로 항목3은 헤더 테이블의 첫 번째 항목으로 레벨은 0이다. 따라서 첫 번째 노드인 노드3의 레벨0의 카운트를 1증가한다. 노드3의 하위 노드 중에 2로 시작하는 노드가 없으므로 2, 4, 7로 구성된 새로운 노드를 추가하고 레벨0의 카운트를 1증가한다. 그리고 헤더 테이블의 2, 4, 7 항목에 새로 추가된 2, 4, 7 노드의 링크를 추가적으로 연결한다.

TID2(3, 1, 4, 7)가 트리에 삽입된 후의 트리구성은 그림 4이다. TID2 역시 항목 3부터 시작하는 트랜잭션으로 항목3은 헤더 테이블의 첫 번째 항목으로 레벨은 0이다. 따라서 첫 번째 노드인 노드3의 레벨0의 카운트를 1증가시켜 2가된다. 노드3의 하위 노드 중에 1로 시작하는 노드가 있으므로 노드1의 레벨0의 카운트를 1증가 시킨다. 노드1의 하위 노드들 중에 항목 4로 시작하는 노드가 없으므로 4, 7로 구성된 새로운 노드를 추가하고 레벨0의 카운트를 1증가한다. 헤더 테이블의 4와 7의 항목에 연결되어 있던 2, 4, 7 노드의 링크를 제거하고 4, 7항목에 새로 추가된 4, 7노드의 링크를 연결한다.

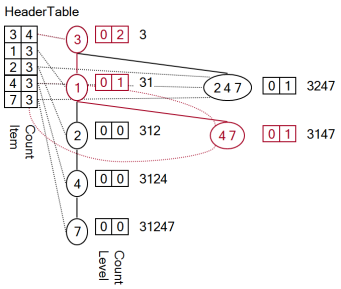


그림 4. TID2(3, 1, 4, 7) 삽입으로 신규 노드 생성

TID3(3, 2)이 트리에 삽입된 후의 트리구성은 그림5이다. TID3 역시 항목 3부터 시작하는 트랜잭션으로 레벨은 0이다. 첫 번째 노드인 노드3의 레벨0의 카운트를 1증가 시켜 3이다. 노드3의 하위 노드 중에 2로 시작하는 노드가 있으나 TID3은 항목 2만 있지만 하위 노드는 2, 4, 7로 구성되어 있다. 이 경우는 노드의 분할이 이루어진다. 즉, 2로만 구성된 노드와 4, 7로 구성된 노드로 분할이 되어 4, 7로 구성된 노드는 노드 2의 하위 노드로 삽입된다. 마찬가지로 노드 3의 레벨0의 카운트를 1증가시켜 2가된다.

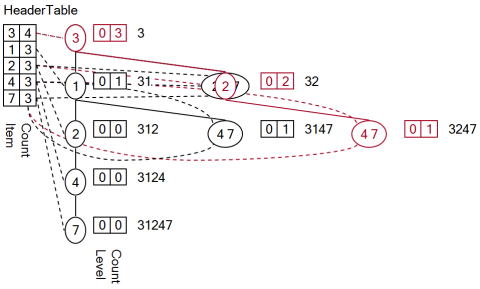


그림 5. TID3(4, 3) 삽입으로 노드의 분할

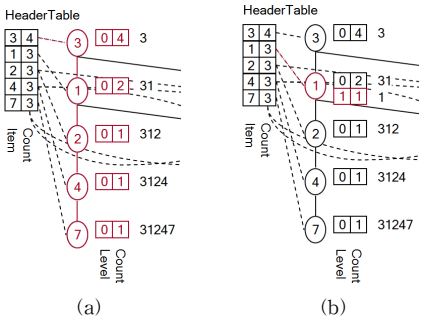


그림 6. TID 4, 5의 삽입으로 카운트와 레벨 수정

TID4(3, 1, 2, 4, 7)가 트리에 삽입된 후의 트리구성은 그림6의 (a)이다. TID4 역시 항목 3부터 시작하는 트랜잭션으로 레벨은 0이다. 첫 번째 노드인 노드3의 레벨0의 카운트를 1증가 시켜 4이다. 노드3의 하위 노드 중에 1로 시작하는 노드가 있어 노드1의 레벨0의 카운트를 1증가 시켜 2가된다. 마찬가지로 항목 2, 4, 7이 하위노드에 존재함으로 레벨0의 카운트를 1증가 시킨다.

TID5(1)가 트리에 삽입된 후의 트리구성은 그림5의 (b)이다. TID2는 항목 1부터 시작하는 트랜잭션으로 항목1은 헤더 테이블의 두 번째 항목으로 레벨은 1이다. 따라서 헤더 테이블 1의 첫 번째 노드인 노드1에 레벨1의 카운트를 증가시켜야 한다. 그러나 노드1에는 레벨1이 없으므로 새로운 레벨1을 추가 시키고 카운트를 1증가 시켜 1이다.

이와 같이 5개의 트랜잭션을 삽입한 후의 트리 구성은 그림6이다. 기존의 FP-Tree와 제안하는 CPFP-Tree를 비교해 보면, FP-Tree는 하나의 노드에 하나의 항목만을 가지고 있는 것에 반하여 CPFP-Tree는 하나의 노드에 여러 항목을 가질 수 있다. 이는 희소 빈발 패턴의 경우에 트리의 크기가 지나치게 커지고 가지의 수가 지나치게 많아지는 것을 방지한다. 또한 FP-Tree는 동일한 항목의 노드들의 연결을 각각의 노드들이 가지고 있지만 CPFP-Tree는 헤더 테이블이 모두 가지고 있다. 하나의 노드에 여러 항목을 가지고 있는 경우 분할이 발생할 수 있는데 이때 제안하는 자료구조는 노드들 간의 연결을 보다 쉽게 할 수 있다. 이는 트리의 구성 및 탐사 시간을 단축한다. 구성된 CPFP-Tree의 재귀적인 탐사과정을 통하여 전체 반발 항목 집합을 탐사한다.

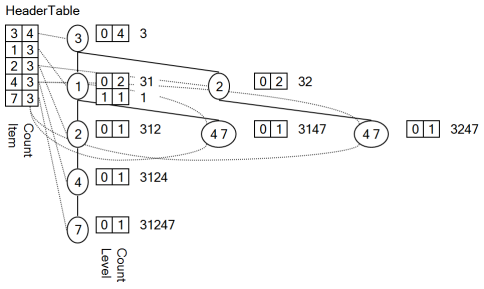


그림 7. 완성된 CPFP-Tree 구조

CPFP-Tree를 이용하여 빈발 패턴을 탐사하는 과정은 기존의 FP-Growth와 유사하며 보다 자세한 내용은 그림 8에 자세히 나타난다. 입력받은 구성된 전체 CPFP-Tree를 Mine 함수의 매개변수로 호출한다(알고리즘 01). 빈발 패턴을 탐사하는 Mine 함수에서는 입력받은 트리가 단일 경로로 구성되어 있는지 판단한다(알고리즘 05). 단일 경로로 구성되어 있다면 트리를 탐사하여 빈발항목 집합을 FISL에 추가한다. 단일 경로로 구성되어 있지 않다면 헤더 테이블의 각각의 항목에 대하여 개별 트리를 구성한다(알고리즘 09-14). 재귀적으로 개별트리에 대하여 빈발 패턴 탐사 과정을 수행한다(알고리즘 12).

```
Input:
    CPFP: CPFP-Tree;
    Min_sup: 최소 지지도 임계값;

Output:
    FISL: 빈발 패턴 리스트;

01: Mine(CPFP);

02: // 빈발 패턴 마이닝 함수 Mine
03: function void Mine(localTree)
04: {
05:     if (localTree가 단일 경로로 구성)
06:         FISL에 빈발 패턴 추가;
07:     else
08:     {
09:         foreach (항목 i in localTree)
10:         {
11:             localTree = i의 개별 트리;
12:             Mine(localTree); // 재귀호출
13:         }
14:     }
15: }
```

그림 8. 빈발 패턴 마이닝 알고리즘

빈발 패턴 마이닝 알고리즘은 기존의 FP-Growth와 유사하나 기존의 FP-Growth는 지역 트리 생성할 때 FP-Tree를 사용하지만 제안하는 알고리즘은 CPFP-Tree를 사용한다. 따라서 제안하는 CPFP-Tree의 하나의 노드에 여러 개의 항목, 레벨, 카운트 정보를 가지는 특징으로 인하여 트리의 크기와 트리 순회 시간이 감소될 수 있다.

4. 성능평가

이번 장에서는 본 논문에서 제안하는 알고리즘과 다른 알고리즘들과의 성능을 비교 평가한다. 성능 평가에 사용된 시스템의 사양은 CPU: Intel Core 2 Duo E4500 2.2GHz, Memory: 2GB DDR2 main memory, HDD: 320GB 7200rpm(8M), OS: Windows XP SP3이다. 제안하는 알고리즘은 Microsoft Visual Studio 2008 C#으로 구현 하였으며, 비교 알고리즘들은 Bart Goethals's web site의 Apriori와 FP-Growth 알고리즘을 이용하였다[16].

성능평가에 사용된 트랜잭션 데이터베이스는 FIMI Repository의 T10I4D100K와 Chess를 이용하였다[17]. T10I4D100K는 IBM-Artificial 트랜잭션 데이터베이스로 IBM Almaden의 데이터 생성기로 생성한 인공적인 트랜잭션 데이터베이스이다[18]. Chess는 Roberto Bayardo

의 트랜잭션 데이터베이스이다. 보다 자세한 내용은 표2에 기술되어 있다.

표 2. T10I4D100K와 Chess의 기본적인 정보

Datasets	#Trans	#Items	Max#Trans	Avg#Trans
T10I4D100K	100,000	1,000	29	10
Chess	3,196	75	37	37

T10I4D100K는 100,000개의 트랜잭션과 1,000개의 항목으로 구성되어 있으며, 최대 트랜잭션의 길이는 29이고 평균 트랜잭션의 길이는 10이다. Chess는 3,196개의 트랜잭션과 75개의 항목으로 구성되어 있으며, 최대 트랜잭션의 길이는 37이고 평균 트랜잭션의 길이는 37이다.

T10I4D100K의 최소 지지도의 변화에 따른 빈발 패턴의 개수는 표3에 자세히 기술되어 있으며, 낮은 최소 지지도에도 빈발 패턴이 적은 희소 빈발 패턴 데이터라는 것을 확인할 수 있다.

표 3. T10I4D100K의 최소 지지도에 따른 빈발 패턴 개수

Min_sup(%)	0.9	0.8	0.7	0.6	0.5
#Freq.items	421	494	603	772	1,073
Min_sup(%)	0.4	0.3	0.2	0.1	
#Freq.items	2,001	4,552	13,255	27,532	

표 4. Chess의 최소 지지도에 따른 빈발 패턴 개수

Min_sup(%)	95	90	85	80	75
#Freq.items	77	622	2,669	8,227	20,993
Min_sup(%)	70	65	60	55	50
#Freq.items	48,731	111,239	254,944	574,998	1,272,982

Chess의 최소 지지도의 변화에 따른 빈발 패턴의 개수는 표 4에 자세히 기술되어 있으며, 높은 최소 지지도에도 빈발 패턴이 많은 밀집 빈발 패턴 데이터라는 것을 확인할 수 있다.

본 논문에서 제안하는 CPFP-Tree를 이용한 알고리즘과 기존의 알고리즘인 Apriori와 FP-Growth 알고리즘을 희소 빈발 패턴인 T10I4D100K와 밀집 빈발 패턴인 Chess를 이용하여 최소지지도의 변화에 따른 수행시간을 비교하였으며 그림 9과 10의 그래프는 최소 지지도에 따른 수행시간을 나타낸다.

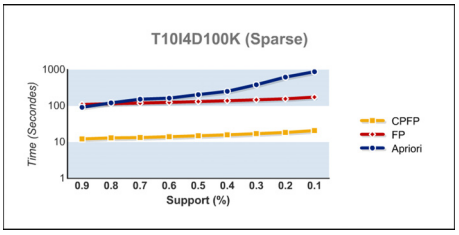


그림 9. 희소 빈발 패턴인 T10I4D100K의 수행시간 비교

희소 빈발 패턴인 T10I4D100K에 대하여 CPFP, FP-Growth, Apriori 알고리즘의 수행시간을 비교해 보면, 최소 지지도가 0.9%이면 전체 빈발 패턴을 탐사하는데 소요된 시간은 CPFP는 12.2초, Apriori는 91.8초, FP-Growth는 109.3초이다. 최소 지지도가 0.5%이면 전체 빈발 패턴을 탐사하는데 소요된 시간은 CPFP는 15초, FP-Growth는 131.4초, Apriori는 204.5초이다. 최소 지지도가 0.1%이면 CPFP는 20.8초, FP-Growth는 175.3초, Apriori는 881.1초이다. 최소 지지도가 0.6%까지는 Apriori와 FP-Growth는 비슷한 수행시간을 보였으며 0.9%이면 Apriori가 FP-Growth보다 조금 더 빠른 수행시간을 보였다. CPFP 알고리즘은 FP-Growth 알고리즘보다 약 10배 정도 빠르게 빈발 패턴을 탐사하였다.

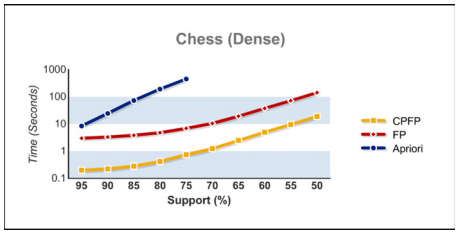


그림 10. 밀집 빈발 패턴인 Chess의 수행시간 비교

밀집 빈발 패턴인 Chess에 대하여 CPFP, FP-Growth, Apriori 알고리즘의 수행시간을 비교해 보면, 최소 지지도가 95%이면 전체 빈발 패턴을 탐사하는데 소요된 시간은 CPFP는 0.2초, FP-Growth는 3.0초, Apriori는 8.5초이다. 최소 지지도가 75%이면 CPFP는 0.7초, FP-Growth는 7.0초, Apriori는 453.4초이다. 최소 지지도가 50%이면 CPFP는 19초, FP-Growth는 142.8초이다. 최소 지지도가 75%보다 작은 경우에는 Apriori 알고리즘의 수행시간이 매우 커져 생략하였다. CPFP는 FP-Growth 알고리즘 보다 약 10배 정도 빠르게 빈발 패턴을 탐사하였다.

희소 빈발 패턴인 T10I4D100K와 밀집 빈발 패턴인 Chess에 대하여 CPFP, FP-Growth, Apriori 알고리즘의 수행시간을 비교해본 결과 CPFP는 다른 알고리즘보다 약 10배 정도 빠르게 빈발 패턴을 탐사하였다. 따라서 본 논문에서 제안하는 CPFP 알고리즘은 밀집 및 희소 빈발 패턴 모두를 빠르게 탐사할 수 있다.

5. 결론

본 논문에서는 단일 알고리즘을 사용하면서도 밀집 및 희소 빈발 패턴 모두에 대해 빠르게 빈발 패턴을 마이닝할 수 있는 압축된 패트리샤 빈발 패턴 트리(CPFP-Tree)라는 새로운 자료구조와 이를 사용한 빈발 패턴 마이닝 알고리즘을 제안했다. 실험 평가를 통하여 제안하는 알고리즘이 기존의 FP-Growth 알고리즘 보다 약 10배 정도 빠르게 빈발 패턴을 탐사하였다.

따라서 제안한 CPFP 알고리즘은 빈발 패턴을 탐사하는 다양한 분야에 활용될 수 있다. 또한 제안한 알고리즘은 밀집 및 희소 빈발 패턴 모두를 빠르게 탐사하는 효과도 기대할 수 있다. 따라서 제안한 알고리즘은 사전에 패턴의 빈발 및 희소 여부를 알 수 없는 환경에서 활용될 수 있다. 즉 제안한 알고리즘은 일반적인 빈발 패턴 마이닝뿐만 아니라 공간 데이터나 데이터스트림 같은 사전에 탐사될 패턴의 특징을 예측할 수 없는 환경에서 활용될 수 있다.

참 고 문 헌

[1] 박성승, 손호선, 이동규, 지은미, 김희석, 류근호, "u-City응용에서의 시간 패턴을 이용한 단기 전력 부하 예측," 한국공간정보시스템학회 논문지, 제11권, 제2호, 2009, pp. 177-181.

[2] 서성보, 이용미, 이준욱, 남광우, 류근호, 박진수, "RFID 데이터 스트림에서 이동궤적 패턴의 탐사," 한국공간정보시스템학회 논문지, 제11권, 제1호, 2009, pp. 127-136.

[3] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," Proceedings of the Data Mining and Knowledge Discovery, Springer Netherlands, 2007.

[4] 김의찬, 김계현, 이철용, 박은지, "비트 클러스터링을 이용한 빈발 패턴 탐사의 성능 개선 방안," 한국공간정보시스템학회 논문지, 제9권, 제1호, 2007, pp. 105-115.

[5] 홍남관, 최진우, 양영규, "퍼지논리와 교통패턴을 이용한 유고검지 모형에 관한 연구," 한국공간정보시스템학회 논문지, 제9권, 제1호, 2007, pp. 79-90.

[6] 박지웅, 김동오, 홍동숙, 한기준, "시공간 이동 패턴 추출을 위한 효율적인 알고리즘," 한국공간정보시스템학회 논문지, 제8권, 제2호, 2006, pp. 39-52.

[7] 홍성한, 황병연, "수직구조 기반의 빈발 이동 객체 패턴 탐색 기법," 한국공간정보시스템학회 2005년도 추계학술대회, 2005, pp. 75-79.

[8] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, 1994, pp. 487-499.

[9] 이경민, 정석호, 신동문, Ibrahim Musa Ishag Musa, 이동규, 손교용, 류근호, "효율적인 밀집 및 희소 빈발 항목 집합 탐색 방법," 제31회 정보처리학회 춘계학술대회 논문집, 제16권 제1호, 2009, pp. 375-376.

[10] R. P. Gopalan and Y. G. Sucahyo, "High Performance Frequent Pattern Extraction using Compressed FPTrees," Proceedings of the SIAM International Workshop on High Performance and Distributed Mining (HPDM), Orlando, USA, 2004.

[11] A. Pietracaprina, and D. Zandolin, "Mining Frequent Itemsets using Patricia Tries," Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, Florida, USA, 2003.

[12] J. Liu, Y. Pan, K. Wang, and J. Han, "Mining Frequent Item Sets by Opportunistic Projection," Proceedings of ACM SIGKDD, Edmonton, Alberta, Canada, 2002.

[13] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases," Proceedings of the IEEE International Conference on Data Mining (ICDM), San Jose, California, USA, 2001, pp. 441-448.

[14] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 2000, pp. 1-12.

[15] J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash based algorithm for mining association rules," Proceedings of the 1995 ACM SIGMOD international conference on Management of data, New York, NY, USA, 1995, pp. 175-186.

[16] B. Goethals, Home page of Bart Goethals, http://www.adrem.ua.ac.be/~goethals/.

[17] FIMI, Frequent Itemset Mining Implementations Repository, http://fimi.cs.helsinki.fi/.

[18] Z. Zheng, R. Kohavi, and L. Mason, "Real World Performance of Association Rule Algorithms," Proceedings of the Seventh ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2001, pp. 1-14.

이 동 규

2005년 서원대학교 정보통신공학과 (공학사)

2007년 충북대학교 대학원 전자계산학과 (공학석사)

2007년~현재 충북대학교 대학원 전자계산학과(박사수료)

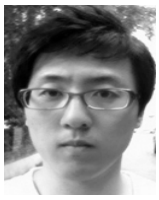
관심분야는 데이터베이스, 지리정보시스템, 데이터마이닝, 센서네트워크, 상황인식, 메디칼 인포매틱스

이 경 민

2008년 충북대학교 컴퓨터공학과 (공학사)

2008년~현재 충북대학교 대학원 전자계산학과(석박사통합과정)

관심분야는 데이터베이스, 데이터 마이닝, 공간 데이터 마이닝, GIS, 센서 네트워크, 메디컬 데이터 마이닝



정 석 호

2009년 극동대학교 멀티미디어학과 (공학사)

2009년~현재 충북대학교 대학원 컴퓨터과학과 (석사과정)

관심분야는 관계형 데이터베이스, 시공간 데이터베이스, 지식기반 정보검색 시스템, 데이터마이닝, 바이오 인포메틱스, 바이오 메디컬

이 성 호

1997년 충북대학교 전자계산학과 (공학사)

2000년 충북대학교 대학원 전자계산학과 (공학석사)

2000년~현재 한국전자통신연구원(ETRI) 공간정보연구팀(Spatial Information Research Team) 근무

관심분야는 spatiotemporal database, GIS, GeoWeb

류 근 호

1976년 숭실대학교 전산학과(이학사)

1980년 연세대학교 공학대학원 전산전공 (공학석사)

1988년 연세대학교 대학원 전산전공 (공학박사)

1976년~1986년 육군군수 지원사 전산실(ROTC 장교), 한국전자통신연구원(연구원), 한국방송통신대 전산학과(조교수) 근무

1989년~1991년 Univ. of Arizona Research Staff (TempIS 연구원, Temporal DB)

1986년~현재 충북대하교 전자정보대학 컴퓨터공학부 교수

관심분야는 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 지식기반 정보검색 시스템, 유비쿼터스컴퓨팅 및 스트림데이터 처리, 데이터마이닝, 데이터베이스 보안, 바이오 및 메데칼 인포메틱스