FP-Tree 및 DRFP 의 성능 개선 기법

조경수*, 정재호, 김영희, 김응모
*성균관대학교 정보통신공학부
e-mail : kisschks@hotmail.com

Technique for Improving performance of FP-Tree and DRFP

Kyung Soo Cho*, Jae-ho Jeong, Young Hee Kim, Ung-mo Kim
*School of Information and Communication Engineering,
Sungkyunkwan University

요 약

FP-tree는 연관성 규칙 알고리즘 전체의 성능을 향상 시키며 DB 스캔을 단 2회로 줄였다. 하지만 빈발 항목과 모든 트랜잭션의 tree 정보를 메모리에 상주 시키면서 많은 메모리 공간을 요구했다. 그래서 나온 DRFP알고리즘은 메모리 요구 문제를 저장장치에 저장함으로 해결 하였으나 FP-tree와는 달리 시간 성능에서의 문제점을 가졌다. 그래서 우리는 이러한 문제점을 보완할 NRFP-tree(Nare disc-Resident Frequent pattern Tree)를 제안한다.

1. 서론

이전의 마이닝 알고리즘들은 많은 양의 계수 공간을 필요로 하며 계수 공간 마련을 했어도 대응하는 계수 지점을 찾기 위해 많은 시간을 소요 한다. 고객 트랜잭션 데이터베이스에서 패턴을 찾기 위해 후보 항목 집합을 형성하고 연관성 규칙을 적용해서 결과를 만들기 위해 계산을 반복하는 시간 및 공간의 비효율적인 구조를 가지고 있었다. [1] 이것을 개선하기 위해 좀 더 빠른 알고리즘으로 Apriori 알고리즘을 제안했다. [2] 이후 Apriori 알고리즘을 기반으로 한 CLOSET+ [3] 이나 MAFIA [4] 같은 알고리즘들이 나왔으나 Apriori가 가진 비효율적인 구조적 문제를 해결하지는 못했다. 이러한 Apriori 및 유사-Apriori 알고리즘들이 가진 문제점을 해결하고자 FP-tree가 제안되었다. FP-tree는 기존의 알고리즘들이 요구 했던 여러 번의 DB스캔을 2회로 단축 하면서 소요 시간을 획기적으로 줄일 수 있었다. [5][6]

FP-tree는 이러한 장점에도 불구하고 DB의 크기가 커지면 많은 공간의 메모리를 필요로 하게 되는 단점을 지녔다. 그리하여 낮은 support threshold 가 주어지게 되거나 데이터베이스가 커지게 되면 FP-tree는 패턴 마이닝을 수행하지 못하게 되는 경우가 발생한다. 이것을 해결하기 위한 방법으로 Muhaimenul Adnan 과 Reda Alhajj는 DRFP(Disk-resident FP-tree) 알고리즘을 제안했다.[7] DRFP-tree는 FP-tree의 단점을 보완한 것으로 메모리 병목 현상이 발생하면 tree 구조를 유지하면서 동시에 부분화 시켜 2차 저장장치에 나누어 저장한다. 그래서 낮은 support threshold나 큰 데이터베이스 환경에서도 빈발 패턴 마이닝을 할 수 있게 되었다. 하지만 FP-tree에서는 존재하지 않던 알고리즘 구조로 인해 계산에 걸리는 시간이 FP-tree에 비해 크게 늘어나게 된다. 그리하여 본 논문에서는 DRFP-tree에서 시간 성능을 향상 시키는 방법을 연구 하였다.

본 논문에서는 DRFP가 가진 빈발 패턴 마이닝 계산에 걸리는 시간을 줄일 수 있는 기법을 제안한다. 본 논문에서 제안되어진 NRFP(Nare disc-Resident Frequent Pattern Tree) 는 효율적인 Tree 운영 기법이다.

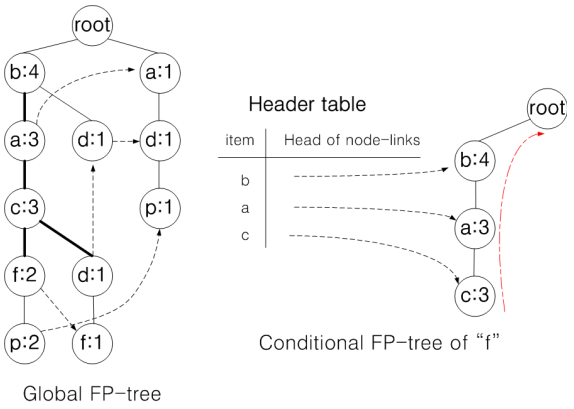
또한 본 논문은 다음과 같이 구성된다. 2장에서는 FP-tree에 대한 설명과 FP-tree와 DRFP-tree의 비교한다. 3장에서 NRFP-Tree의 구현 방식에 대해 설명하고 4장에서 결론과 향후 연구 과제를 언급하며 논문을 마친다.

2. 관련연구

2.1 FP-tree

FP-tree는 Jiawei Han, Jian Pei, and Yiwen Yin 이 제안한 빈발 패턴 마이닝 알고리즘이다. 그림1 에서는 FP-tree와 조건 FP-tree에 대해 설명을 하고 있다. FP-tree는 상향식 방법으로 tree를 탐험하여 FP-tree에서 빈발 항목 집합들을 생성하는 알고리즘이다. 문제를 더 작은 부분문제들로 분해하는 분할-정복 전략을 적용함으로써 특정 접미사로 끝나는 모든 빈발 항목을 찾는다. 효율적인 빈발 패턴 마이닝을 위한 소형 데이터 구조로 FP-tree를 구성하며 패턴을 찾는 방법으로 Pattern-growth가 있다. DB 스캔으로 빈발 아이템 리스트를 작성 하여 아이템을 빈번도 내림 차순 서열로 정렬한다. FP-tree 구성은 tree의 루트를 만들고 null이라는 라벨을 붙인 후 첫 번째 트랜잭션의 스캔은 tree의 첫 가지들을 형성하게 한다. 두 번째 트랜잭션에서는 공통된 경로가 존재하기 때문에 공통 부분인 전위 개수를 1씩 늘려주고 새로운 노드를 자식 노드로 추가한다. FP-tree는 하나

의 루트, 루트의 자식인 아이템 전위 하위tree들의 집합, 그리고 빈발 아이템의 헤더 테이블로 구성된다. 아이템 전위 하위tree에서 노드는 Item-name, count, 그리고 nodelink. 영역으로 구성된다.

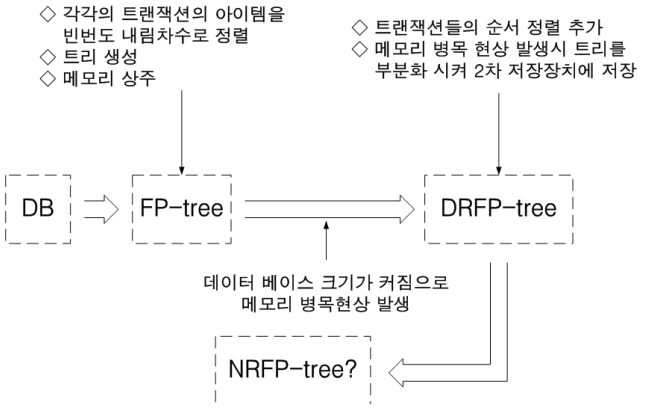


<그림 1> FP-tree와 조건 FP-tree 구성

FP-tree는 기존의 Apriori 알고리즘과 비교해서 효율적인 성능을 보여준다. 하지만 데이터베이스가 커질 경우 메모리에 tree형태로 상주시키는 알고리즘의 특성 때문에 병목 현상이 발생하며 결국 작업을 수행하지 못하는 단점을 지니게 된다.

2.2 FP-tree와 DRFP-tree의 비교

DRFP 알고리즘은 FP-tree에서 가장 큰 문제점이었던 메모리 상주시 발생하는 문제를 해결했다. FP-tree에서 하던 데이터 전처리 과정에 트랜잭션들의 순서를 정렬 하는 것도 추가 하였다. 또한 FP-tree와 달리 Bottom-up 방식으로만 FP-growth를 생성한다. 그리고 메모리에 노드 정보를 상주시키는 FP-tree 구조를 이용시 병목 현상이 발생하면 FP-tree는 계산이 불가능하게 변하던 것과는 달리 DRFP-tree에서는 tree들을 부분화 시켜 2차 저장 장치에 저장을 하게 된다. 이때 전체적인 tree 구조를 변형하지 않고 저장을 하게 되어 빈발 패턴을 찾는데 문제가 발생하지 않는다.



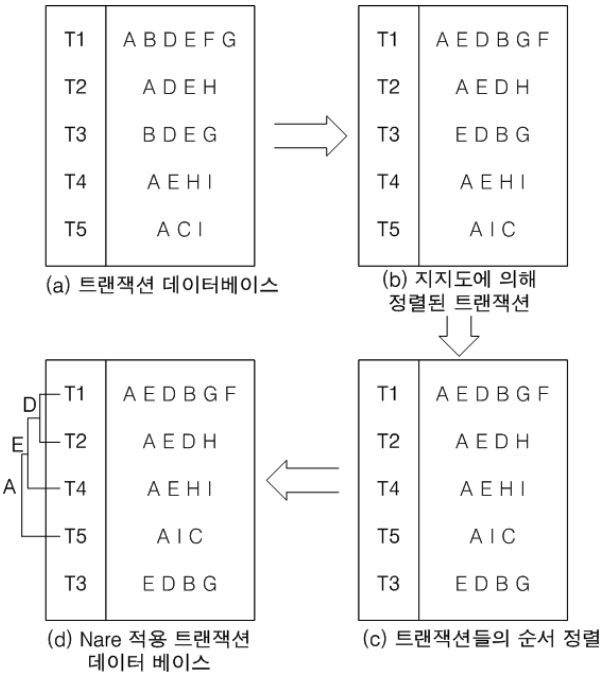
<그림 2> FP-tree와 DRFP-tree의 관계도

그림 2는 FP-tree, DRFP 그리고 NRFP 알고리즘이 어떤 관계 인지를 보여준다. FP-tree의 메모리 관련 문제를 DRFP-tree가 해결 했음에도 DRFP-tree는 FP-tree가 지니지 않은 새로운 문제들이 발생 하였다. 작은 메모리 환경이나 큰 규모의 데이터베이스에서도 빈발 패턴 마이닝이 가능하지만 전체 성능은 시스템이 발생시키는 I/O시간으로 인해 저하됐다.

3. 제안내용

3.1 데이터 전처리 과정

(a)처음 DB에서 데이터 전처리를 통해서 (b)각각의 트랜잭션을 support 순서대로 정렬을 하고 (c)각각의 트랜잭션들 끼리 비교 정렬을 다시 실행한다. (d)그 후에 Nare 개념을 적용하여 DB에 Nare 정보를 추가한다. 이런 순서로 이루어지는데 (b)와 (c)의 과정은 DRFP-tree와 동일하다. (d)단계의 경우는 본 논문에서 제안하는 Nare라는 새로 제안하는 개념이 적용된다. 이것은 비슷한 것 끼리 묶어두는 개념으로 그림을 통해 설명을 하겠다.

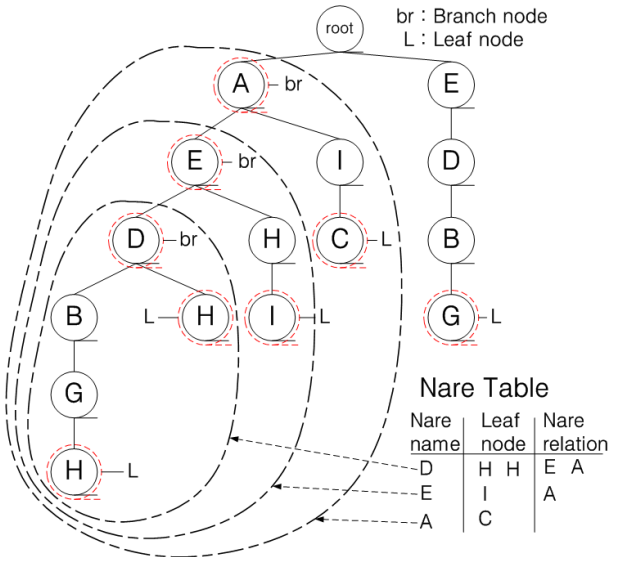


<그림 3> Nare 데이터베이스 전처리 과정

그림3에 나와 있는대로 (a), (b) 와 (c)의 과정을 거친 후 (d)와 같이 Nare로 묶는다. 이것은 나중에 NRFP-tree를 구축 할 때 겹치는 과정을 과감하게 생략하기 위함이다. 겹치는 과정을 생략하게 됨으로써 계산에 걸리는 시간을 단축 할 수 있다. 작은 DB에서는 큰 차이가 발생하지 않으나 DB의 크기가 커질수록 계산에 소모되는 시간을 단축하는 효과가 커질 것으로 기대 된다.

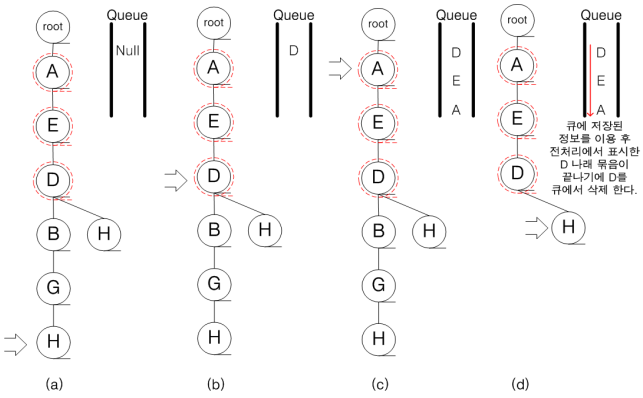
3.2 NRFP 구성 및 NRFP-Growth

제안된 NRFP-tree는 기존의 FP-tree와 달리 DRFP-tree와 동일한 방법으로 트리를 구성한다. 트리 구성시 소규모 집단화 하는 과정을 수행 후 패턴 탐색시 DRFP-tree 에서는 Bottom-up 방법에 의해 수행한다. 따라서 이 역시 NRFP-tree에서도 수행한다. NRFP-tree는 DRFP-tree와 달리 각 집단의 마지막 노드는 분화가 일어나거나 리프노드 조건을 갖는다. 그렇기 때문에 이들 노드에게 분화 또는 리프 노드 여부를 판단하는데 분화가 일어나는지 아닌지와 리프 노드인지 아닌지를 구별해야 하는 이유는 데이터 전처리를 통해 적용된 Nare 개념을 NRFP-growth 에서 적용하기 위함이다. 패턴을 찾기 위해 NRFP-growth를 진행하는 동안 첫 번째로 발견되는 분화 노드는 Nare의 가장 첫 번째에 있는 정보를 가진 것으로 판단하게 되어 이후 정보를 큐에 저장하게 된다. 그리고 큐에 저장된 내용은 다음 NRFP-growth 과정에서 공통된 노드정보를 포함 시 계산하지 않고 꺼내서 쓰게 된다.



<그림 4>NRFP-tree와 집단의 마지막 대상 노드

그림4는 그림3에서 예시로 나온 DB를 tree로 구성 한 것이다. 각각의 노드들 중 분화가 일어나면 br 정보가 추가되고 리프 노드인 경우에는 L정보가 추가 된다. 데이터베이스 전처리에서 묶은 Nare를 기반으로 트리 구성시에 Nare Table이 작성된다. Nare Table을 활용하여 그림4에서처럼 트리에 가상의 묶음을 형성한다. 그림4에서 Nare-D에는 B-G-H 와 H 가 A-E-D 라는 동일한 전위 노드들을 가지게 된다. 그래서 패턴 탐색시에 B-G-H를 탐색한 후 D-E-A 순으로 탐색 할 때 해당 Nare 상태와 br 정보를 가진 노드의 상태가 같은가를 확인 후 사실이라고 판단되면 이후 노드들의 정보를 큐에 저장한다. 큐에 저장된 이 정보들은 이후 B-G-H와 동일한 전위 노드들을 가진 H를 탐색할 때 D-E-A 노드들을 재탐색 하지 않고 큐에 저장된 정보를 활용해서 패턴을 찾아내게 된다.



<그림 5 > 큐에 노드 정보를 저장하는 과정

그림5는 큐에 노드 정보를 저장하는 과정 이다. NRFP-growth중 제일 처음 나오는 Branch node가 전처리 과정에서 묶은 Nare의 정보와 일치하는지를 보고 그 이후 얻어지는 노드들의 정보를 큐에 저장한다. 그림5에서 과정을 살펴 보면 (a)에서는 Growth가 시작되고 있다. (b)에서 D를 읽을 때 D는 Nare와 일치하는지 확인한다. 이후 탐색되는 노드들의 정보를 (c)처럼 큐에 저장한다. 끝까지 읽은 후 다음 리프 노드인 H로 커서가 이동 했을 때 이후 다시 D노드에 도착하게 되면 (d)에서처럼 이전에 큐에 저장된 정보들을 활용하게 된다. 정보를 모두 활용하고 난 후 전처리 과정에서 표시 한 Nare D묶음이 종료되게 되고 큐에서 D노드와 관련된 정보를 삭제한다. 삭제 후에는 큐에 E와 A 노드와 관련된 정보만 남고 다음 패턴 탐색 시에 동일한 Nare 하에 있는 노드들의 패턴 탐색시에 활용된다.

4. 결론 및 향후 연구 방향

본 논문에서는 추가적인 전처리 단계와 tree 구성 및 growth 방법에 대해 간단히 언급만 하였다. 이는 FP-tree나 DRFP-tree에 동시에 적용 가능한 성능 향상 기법이다. 향후에는 DRFP가 가진 구조적인 문제, 즉, I/O 시간에 대한 제거 또는 회피하는 방법에 대해 모색 할 것이다. DRFP 알고리즘의 I/O 시간문제를 해결 한다면 FP-tree를 큰 데이터베이스나 작은 support threshold에서도 사용할 수 있으며 FP-tree가 가진 장점을 그대로 가질 것이다. 따라서 현재 제안된 NRFP-tree는 DRFP-tree가 가진 문제점을 해결 한 알고리즘이 아니기 때문에 향후 연구를 통해 DRFP-tree가 가진 문제점을 해결한 완벽한 형태의 NRFP-tree 알고리즘을 연구하고 실제 사용가능한 형태로 만드는 것이 목표 이다.

감사의 글

이 논문은 2009 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. 2009-0075771)

참고문헌

[1] R. Agrawal, T. Imielinski and A. Swami, "*Mining Association Rules between Sets of Items in Large Databases*", Proceedings of ACM SIGMOD on Management of Data, pp.207~216, 1993.

[2] R. Agrawal and R. Srikant, "*Fast Algorithms for Mining Association Rules*", Proceedings of the 20th International Conference on Very Large Databases, pp.487-499, 1994.

[3] J. Wang, J. Han, J. Pei "CLOSET+: searching for the best strategies for mining frequent closed itemsets" International Conference on Knowledge Discovery and Data Mining archive, Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Pages: 236 - 245, 2003

[4] Doug Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, Tomi Yiu, "MAFIA: A Maximal Frequent Itemset Algorithm," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 11, pp. 1490-1504, Nov. 2005, doi:10.1109/TKDE.2005.183

[5] Jiawei Han, Jian Pei, and Yiwen Yin "Mining frequent patterns without candidate generation", Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000

[6] J. Han1 , J. Pei, Y. Yin and R. Mao "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" Data Mining and Knowledge Discovery, 2004 - Springer

[7] Muhaimenul Adnan and Reda Alhajj "DRFP-tree: disk-resident frequent pattern tree", Applied Intelligence, 2007