BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

HOÀNG TRUNG THÔNG

PHƯƠNG PHÁP PHÂN VÙNG PHÂN CẤP TRONG KHAI THÁC TẬP PHỔ BIẾN

LUẬN VĂN THẠC SĨ

Chuyên ngành: **Công Nghệ Thông Tin**

Mã số ngành: **60480201**

TP. HỒ CHÍ MINH, tháng 03 năm 2015

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

HOÀNG TRUNG THÔNG

PHƯƠNG PHÁP PHÂN VÙNG PHÂN CẤP TRONG KHAI THÁC TẬP PHỔ BIẾN

LUẬN VĂN THẠC SĨ

Chuyên ngành: **Công Nghệ Thông Tin**

Mã số ngành: **60480201**

CÁN BỘ HƯỚNG DẪN KHOA HỌC: **PGS.TS. LÊ TRỌNG VĨNH**

TP. HỒ CHÍ MINH, tháng 03 năm 2015

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

Cán bộ hướng dẫn khoa học : **PGS.TS. LÊ TRỌNG VĨNH**
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Luận văn Thạc sĩ được bảo vệ tại Trường Đại học Công nghệ TP. HCM ngày 11 tháng 04 năm 2015

Thành phần Hội đồng đánh giá Luận văn Thạc sĩ gồm:
(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ Luận văn Thạc sĩ)

TT	Họ và tên	Chức danh Hội đồng
1	**PGS.TS. Đỗ Phúc**	Chủ tịch
2	**TS. Võ Đình Bảy**	Phản biện 1
3	**TS. Lư Nhật Vinh**	Phản biện 2
4	**PGS.TS. Lê Hoàng Thái**	Ủy viên
5	**TS. Lê Tuấn Anh**	Ủy viên, Thư ký

Xác nhận của Chủ tịch Hội đồng đánh giá Luận sau khi Luận văn đã được sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM
PHÒNG QLKH – ĐTSĐH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

TP. HCM, ngày 14 tháng 03 năm 2015

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: **Hoàng Trung Thông** Giới tính: Nam

Ngày, tháng, năm sinh: **06 / 09 /1975** Nơi sinh: Sơn La

Chuyên ngành: **Công Nghệ Thông Tin** MSHV: 1341860025

I- Tên đề tài: PHƯƠNG PHÁP PHÂN VÙNG PHÂN CẤP TRONG KHAI THÁC TẬP PHỔ BIẾN

II- Nhiệm vụ và nội dung:

Phân vùng thứ bậc để khai thác tập phổ biến trong những cơ sở dữ liệu lớn:

- Khai thác tập phổ biến, các cách tiếp cận
- Cơ sở dữ liệu có kích thước lớn
- Phương pháp phân vùng, phân cấp dữ liệu trên hệ thống nhiều máy
- Áp dụng phương pháp phân vùng phân cấp vào bài toán khai thác tập phổ biến
- Xây dựng chương trình demo

III- Ngày giao nhiệm vụ: 18/08/2014

IV- Ngày hoàn thành nhiệm vụ: 14/03/2015

V- Cán bộ hướng dẫn: PGS.TS. LÊ TRỌNG VĨNH

CÁN BỘ HƯỚNG DẪN **KHOA QUẢN LÝ CHUYÊN NGÀNH**

PGS. TS. Lê Trọng Vĩnh

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi, với sự hướng dẫn của Thầy PGS.TS. LÊ TRỌNG VĨNH và sự đóng góp ý kiến của thầy TS. CAO TÙNG ANH. Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này đã được cảm ơn và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

(Ký và ghi rõ họ tên)

Hoàng Trung Thông

LỜI CÁM ƠN

Lời đầu tiên tôi xin cám ơn chân thành và sâu sắc nhất đến Thầy PGS.TS. LÊ TRỌNG VĨNH , Thầy đã dành rất nhiều thời gian hướng dẫn tôi một cách tận tâm, sâu sát và giúp tôi vượt qua những thời điểm khó khăn nhất về luận văn này. Tôi cũng xin gởi lời cảm ơn đến thầy TS. CAO TÙNG ANH đã có những đóng góp ý kiến quý báu cho luận văn này.

Tiếp theo tôi xin gởi lời cám ơn chân thành và trân trọng nhất đến quý Thầy Cô Khoa CNTT Trường Đại Học Công Nghệ Tp.HCM đã truyền đạt nhiều kiến thức quý báu cho tôi trong suốt quá trình học tập tại trường.

Xin cám ơn gia đình, các bạn học, bạn hữu, đồng nghiệp đã có những góp ý và động viên trong suốt thời gian qua.

TP. Hồ Chí Minh, tháng 03/2015

TÓM TẮT

Mặc dù có nhiều phương pháp đã được đề xuất để nâng cao hiệu quả khai thác dữ liệu nhưng chỉ có ít nghiên cứu về khả năng mở rộng - đó là vấn đề khai thác tập phổ biến khi kích thước của CSDL là rất lớn. Nghiên cứu [14] đề xuất một phương pháp là phân vùng thứ bậc để khai thác tập phổ biến trong CSDL lớn dựa trên một cấu trúc dữ liệu mới gọi là Danh sách mẫu phổ biến (FPL). Một trong những tính năng chính của FPL là khả năng phân vùng cơ sở dữ liệu để chuyển đổi CSDL thành một tập các CSDL con có kích thước có thể quản lý được. Kết quả là một cách tiếp cận chia để trị có thể được phát triển để thực hiện nhiệm vụ khai thác dữ liệu mong muốn. Kết quả cho thấy phân vùng thứ bậc có khả năng khai thác tập phổ biến và tập phổ biến đóng trong CSDL rất lớn.

ABSTRACT

Although many methods have been proposed to enhance the efficiencies of data mining, little research has been devoted to the issue of scalability – that is, the problem of mining frequent itemsets when the size of the database is very large. This study proposes a methodology, hierarchical partitioning, for mining frequent itemsets in large databases, based on a novel data structure called the Frequent Pattern List (FPL). One of the major features of the FPL is its ability to partition the database, and thus transform the database into a set of sub-databases of manageable sizes. As a result, a divide-and-conquer approach can be developed to perform the desired data-mining tasks. Experimental results show that hierarchical partitioning is capable of mining frequent itemsets and frequent closed itemsets in very large databases.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

DB	Cơ sở dữ liệu giao dịch
Conf	Độ tin cậy
CSDL	**C**ơ **s**ở **d**ữ **l**iệu
Item	Mục
Itemset	Tập mục
FPL	**F**requent **P**attern **L**ist
FI	**F**requent **I**temset
KDD	**K**nowledge **D**iscovery and **D**ata Mining
LSB	**L**east **S**ignificant **B**it
MSB	**M**ost **S**ignificant **B**it
Minsup	**min sup**port
minconf	Ngưỡng tin cậy tối thiểu (**min**imum **conf**idence)
Supp	**Supp**ort
Sub-DB	Cơ sở dữ liệu con (phụ)
TID	**T**ransaction **Id**entification

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH ẢNH

MỞ ĐẦU

1. Đặt vấn đề

Trong thời đại ngày nay, với sự phát triển vượt bậc của công nghệ thông tin và sự phổ biến của Internet. Lượng dữ liệu tại các hệ thống thông tin này ngày càng trở nên phong phú, đa dạng và thực sự khổng lồ. Trong tình hình đó, việc chắt lọc những thông tin quý giá từ những dữ liệu khổng lồ này càng có ý nghĩa hơn bao giờ hết, nó đóng vai trò chìa khóa thành công cho sự phát triển của các tổ chức, cá nhân. Các thông tin tìm được có thể được vận dụng để cải thiện hiệu quả hoạt động của hệ thống thông tin ban đầu, cải thiện thời gian tìm kiếm, hay đưa ra những dự đoán giúp cải thiện những quyết định trong tương lai… Các kỹ thuật khai thác dữ liệu (data mining) ngày càng được quan tâm và ứng dụng rộng rãi trong nhiều lĩnh vực của cuộc sống như kinh tế, giáo dục, y tế, trong siêu thị,…

2. Tính cấp thiết của đề tài.

Vì sự "bùng nổ" của thông tin như vậy nên ta phải có phương pháp hiệu quả nhất để khai thác thông tin đó và chúng ta phải cân nhắc những yếu tố gì, tiêu chí nào để lựa chọn khai thác thông tin một cách hiệu quả nhất và nhanh nhất?.

Một trong những công nghệ hiệu quả nhất là khai thác dữ liệu, đó là công nghệ dùng khai thác các mẫu hữu ích hay những kiến thức có ích từ cơ sở dữ liệu lớn [7].

Nhiệm vụ cơ bản và quan trọng nhất của khai thác dữ liệu là khai thác tập phổ biến, đó là tập các mặt hàng được thường xuyên mua cùng với nhau trong một giao dịch, những công cụ trong khai thác tập phổ biến điển hình như phân tích so sánh, phân tích mẫu, phân loại, gom cụm và lưu trữ dữ liệu [7].

Để nâng cao hiệu quả trong khai thác tập phổ biến, một số phương pháp mới và cấu trúc dữ liệu linh hoạt đã được phát triển, chẳng hạn như FP-tree, FP-Growth [6], Danh sách mẫu phổ biến và danh sách mẫu giao dịch [9], [10], [11], hay Khai thác mẫu khổng lồ một cách hiệu quả trong bộ dữ liệu lớn [8]. Tuy nhiên, khi cơ sở dữ liệu (viết tắt là CSDL) ngày càng phình to ra, thậm chí cấu trúc dữ liệu linh hoạt sẽ phát triển ra khỏi dung lượng bộ nhớ thì khả năng mở rộng các phương pháp khai thác dữ liệu là một vấn đề phải được giải quyết. Các phương pháp thông thường sử dụng một lược đồ phân vùng phẳng để phân vùng CSDL ban đầu thành một tập các CSDL con nhỏ hơn ở cùng cấp độ và sau đó tìm các tập phổ biến cục bộ trong các CSDL con này. Xử lý cuối cùng là quét lại CSDL ban đầu để kiểm tra xem các tập phổ biến cục bộ có phải là phổ biến toàn cục hay không. Điều này, tất nhiên phải mất thêm thời gian và tình hình sẽ xấu đi khi khai thác tập phổ biến đóng bởi vì không chỉ tần số của tập phổ biến được tính toán mà việc kiểm tra tập hợp con cũng phải được thực hiện.

Đặc biệt những dữ liệu lớn vượt khỏi tầm kiểm soát của bộ nhớ máy tính, vậy ta phải sắp xếp, phân vùng phân cấp sao cho nhỏ hơn hoặc bằng bộ nhớ máy tính yêu cầu và tốc độ máy tính cũng sẽ nhanh gấp nhiều lần. Trong nghiên cứu này, tác giả luận văn sẽ tìm hiểu phương pháp phân vùng phân cấp để khai thác tập phổ biến trong những CSDL lớn

3. Mục tiêu của đề tài

Mục tiêu của đề tài tìm hiểu việc khai thác các tập phổ biến (frequent item sets) trong cơ sở dữ liệu lớn, dựa trên cấu trúc dữ liệu mới hay gọi là danh sách mẫu phổ biến FPL (Frequent Pattern List). Phương pháp này phân vùng không gian tìm kiếm và chia cơ sở dữ liệu thành một tập các cơ sở dữ liệu con có kích thước có thể quản lý được. Kết quả thu được là, tiếp cận phương pháp chia để trị để khai thác dữ liệu mong muốn mà không cần phải quét lại dữ liệu ban đầu. Phương pháp này được gọi là "Phương pháp phân cấp trong khai thác tập phổ biến", nó có thể cải thiện tốc độ và hiệu suất đáng kể trong khai thác tập phổ biến từ cơ sở dữ liệu lớn.

4. Bố cục của luận văn

Luận văn được chia làm 5 phần:

- Mở đầu
- Chương 1: Giới thiệu về khai thác dữ liệu, cơ sở dữ liệu kích thước lớn.
- Chương 2: Khai phá tập phổ biến.
- Chương 3: Phương pháp phân vùng, phân cấp trong khai phá tập phổ biến
- Chương 4: Kết luận và hướng phát triển trong tương lai

CHƯƠNG 1 GIỚI THIỆU VỀ KHAI THÁC DỮ LIỆU, CƠ SỞ DỮ LIỆU KÍCH THƯỚC LỚN

1.1 Tổng Quan về khai thác dữ liệu

1.1.1 Mục tiêu của khai thác dữ liệu

Với sự phát triển của phần mềm và phần cứng máy tính và số lượng khổng lồ và tăng tốc của dữ liệu. Từ khối dữ liệu rất lớn như vậy, cần phải có những công cụ tự động rút trích các thông tin và tri thức có ích, đó là khai thác dữ liệu (Data mining).

Khai thác dữ liệu là quá trình tìm kiếm các mẫu mới, những thông tin tiềm ẩn trong các khối dữ liệu khổng lồ, khai thác có thể dự đoán những xu hướng trong tương lai, hay giúp cho các công ty kinh doanh ra các quyết định kịp thời, hay dựa trên những sự kiện trong quá khứ của các hệ hỗ trợ ra quyết định (decision support systems - DSSs). Với các ưu điểm trên, khai thác dữ liệu được ứng dụng rộng rãi trong các lĩnh vực như thương mại, tài chính, y học, giáo dục và các lĩnh vực khác.

Khai thác dữ liệu được định nghĩa, hay cách gọi khác của một thuật ngữ rất thông dụng là khám phá tri thức trong cơ sở dữ liệu (Knowledge Discovery in databases - KDD): là việc trích ra các tri thức chưa được nhận ra, tiềm ẩn trong các tập dữ liệu lớn một cách tự động [1]

- Một ví dụ tiêu biểu cho việc khai thác tập phổ biến [7] là phân tích giỏ hàng. Tiến trình này phân tích thói quen mua sắm của khách hàng bằng cách tìm ra sự kết hợp giữa các danh mục khác nhau từ trong giỏ hàng của họ. Việc khám phá ra những sự kết hợp này giúp ích cho các nhà bán lẻ mở rộng phân phối sản phẩm bởi họ thấu hiểu được những lợi nhuận có được từ những danh mục được khách hàng mua thường xuyên. Cho một ví dụ thực tiễn hơn, nếu khách hàng

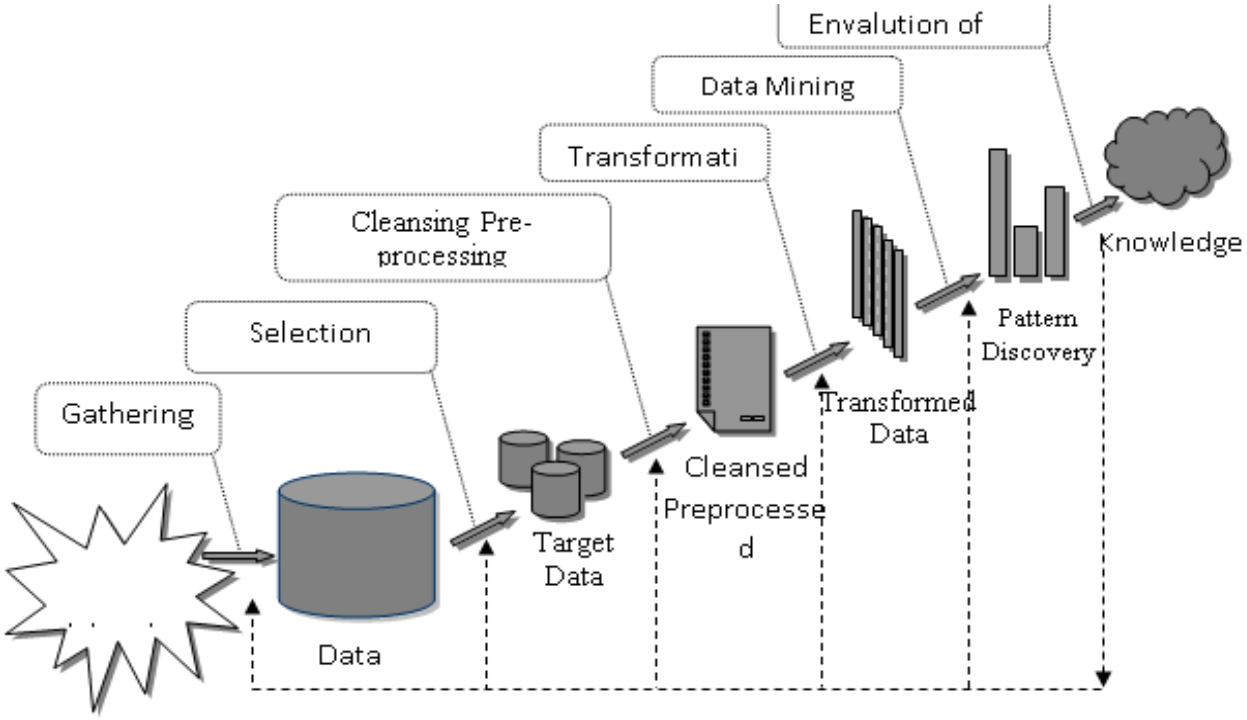
mua sữa, khả năng họ mua bánh mì trên cùng một lần đi siêu thị là như thế nào? Những thông tin này sẽ giúp cho các nhà bán lẻ tăng doanh thu và giúp họ

- Lựa chọn kế hoạch tiếp thị và trưng bày sản phẩm.
- Kết quả phân tích giỏ hàng có thể giúp bạn lên kế hoạch tiếp thị, chiến lược quảng cáo, trưng bày sản phẩm hay lập danh mục bán hàng giảm giá …Ví dụ, kết quả phân tích cho thấy nếu khách hàng mua một máy vi tính thì có thể mua kèm phần mềm diệt vi rút. Từ đó, bạn sẽ có kế hoạch trưng bày sản phẩm hợp lý hơn (Thông tin về máy tính được hiển thị kèm theo phần mềm diệt vi rút được khuyến khích mua).
- Từ phân tích giỏ hàng bạn cũng có thể tìm ra một số quy tắc hay luật kết hợp có ích. Ví dụ, thông tin khách hàng mua máy vi tính và cũng mua phần mềm diệt vi rút đã đưa ra luật kết hợp như sau:

  Computer ➔ antivirus_software [support = 2%, confidence = 60%]

- Độ hỗ trợ (support) và độ tin cậy (confidence) của luật là hai độ đo được quan tâm nhất. Luật có support = 2%, nghĩa là số lần giao dịch mà máy vi tính và phần mềm diệt vi rút được mua cùng nhau chiếm 2% trong tổng số các giao dịch; confidence=60%, nghĩa là có 60% khách hàng mua máy vi tính thì cũng sẽ mua phân mềm diệt vi rút.
- Luật kết hợp được quan tâm nếu nó thỏa mãn cả hai ngưỡng độ hỗ trợ nhỏ nhất (minimum support threshold) và độ tin cậy nhỏ nhất (minimum confidence threshold).

1.1.2 Các bước chính của quá trình khai thác dữ liệu [12]



Hình 1.1: Quá trình khai thác tri thức

- **Gom dữ liệu** (Gathering): Tập hợp dữ liệu là bước đầu tiên trong quá trình khai phá dữ liệu. Đây là bước được khai thác trong một cơ sở dữ liệu, một kho dữ liệu và thậm chí các dữ liệu từ các nguồn ứng dụng Web.

- **Trích chọn dữ liệu** (data selection): Ở giai đoạn này dữ liệu được lựa chọn hoặc phân chia theo một số tiêu chuẩn nào đó, ví dụ chọn tất cả những người có tuổi đời từ 25 – 35 và có trình độ đại học.

- **Tiền xử lý dữ liệu** (data preprocessing): Giai đoạn thứ ba này là giai đoạn hay bị sao lãng, nhưng thực tế nó là một bước rất quan trọng trong quá trình khai phá dữ liệu. Một số lỗi thường mắc phải trong khi gom dữ liệu là tính không đủ chặt chẽ, logic. Vì vậy, dữ liệu thường chứa các giá trị vô nghĩa và không có khả năng kết nối dữ liệu. Giai đoạn này sẽ tiến hành xử lý những dạng dữ liệu không chặt chẽ nói trên. Những dữ liệu dạng này được xem như thông tin dư thừa, không có giá trị. Bởi vậy, đây là một

quá trình rất quan trọng vì dữ liệu này nếu không được "làm sạch - tiền xử lý - chuẩn bị trước" thì sẽ gây nên những kết quả sai lệch nghiêm trọng

- **Biến đổi dữ liệu** (data transformation): Tiếp theo là giai đoạn chuyển đổi dữ liệu, dữ liệu đưa ra có thể sử dụng và điều khiển được bởi việc tổ chức lại nó. Dữ liệu đã được chuyển đổi phù hợp với mục đích khai thác

- **Khai thác dữ liệu** (data mining): Đây là bước mang tính tư duy trong khai phá dữ liệu. Ở giai đoạn này nhiều thuật toán khác nhau đã được sử dụng để trích ra các mẫu từ dữ liệu. Thuật toán thường dùng là nguyên tắc phân loại, nguyên tắc kết hợp hoặc các mô hình dữ liệu tuần tự, …

- **Đánh giá và biểu diễn tri thức** (knowledge representation & evaluation): Đây là giai đoạn cuối trong quá trình khai phá dữ liệu. Ở giai đoạn này, các mẫu dữ liệu được chiết xuất ra bởi phần mềm khai phá dữ liệu. Không phải bất cứ mẫu dữ liệu nào cũng đều hữu ích, đôi khi nó còn bị sai lệch. Vì vậy, cần phải ưu tiên những tiêu chuẩn đánh giá để chiết xuất ra các tri thức (Knowledge).

Trên đây là 6 giai đoạn trong quá trình khai phá dữ liệu, trong đó giai đoạn 5 là giai đoạn được quan tâm nhiều nhất, đó là khai phá dữ liệu.

1.1.3 Các dạng dữ liệu có thể khai thác được [12]

- Cơ sở dữ liệu quan hệ (relational databases)

- Cơ sở dữ liệu đa chiều (multidimention databases)

- Cơ sở dữ liệu giao tác (transactional databases)

- Cơ sở dữ liệu quan hệ - hướng đối tượng (object relation databases)

- Dữ liệu không gian và dữ liệu chuỗi theo thời gian (spatial and time-series data)

- Cơ sở dữ liệu đa phương tiện (multimedia databases)

1.1.4 Hướng tiếp cận và các kỹ thuật trong khai thác dữ liệu [12]

- **Phân lớp và dự đoán** (Classification & prediction): là quá trình xếp một đối tượng vào một trong những lớp đã biết trước. Ví dụ như: phân lớp các bệnh nhân theo dữ liệu hồ sơ bệnh án, phân lớp loại cước hoặc loại dịch vụ sử dụng dựa trên số máy bị gọi của cuộc gọi, phân lớp giờ cao điểm, thấp điểm dựa trên lưu lượng giao thông hàng ngày, phân lớp vùng địa lý theo thời tiết… Đối với hướng tiếp cận này thường sử dụng một số kỹ thuật của học máy như cây quyết định (decision tree), mạng nơron nhân tạo (neural network), hay còn được gọi là học có giám sát – học có Thầy (supervised learning).

- **Luật kết hợp** (association rules): là dạng luật biểu diễn tri thức ở dạng tương đối đơn giản. Ví dụ: 85% sinh viên đăng ký học môn Kỹ thuật lập trình thì có tới 65% trong số họ đăng ký học môn Cơ sở dữ liệu, hay có 60% khách hàng gọi điện thoại liên tỉnh thì 85% họ gọi nội tỉnh… Luật kết hợp được ứng dụng nhiều trong lĩnh vực kinh doanh, y học, tin sinh học, giáo dục, viễn thông, tài chính,…

- **Khai thác mẫu tuần tự/ chuỗi thời gian** (sequential/temporal patterns): Cũng tương tự như khai phá dữ liệu bằng luật kết hợp nhưng có thêm tính thứ tự và tính thời gian. Một luật mô tả mẫu tuần tự có dạng tiêu biểu X → Y, phản ánh sự xuất hiện của biến cố X sẽ dẫn đến việc xuất hiện biến cố Y. Hướng tiếp cận này được ứng dụng nhiều trong lĩnh vực tài chính và thị trường chứng khoán bởi chúng có tính dự báo cao.

- **Phân cụm** (clustering/segmentation): Sắp xếp các đối tượng theo từng cụm dữ liệu tự nhiên, tức là số lượng và tên cụm chưa được biết trước. Các đối tượng được gom cụm sao cho mức độ tương tự giữa các đối tượng trong cùng một cụm là lớn nhất, và mức độ tương tự giữa các đối tượng nằm trong các cụm khác nhau là nhỏ nhất. Lớp bài toán này còn được gọi là học không giám sát - học không thầy (unsupervised learning)

- **Mô tả khái niệm** (concept description & summarization): Lớp bài toán này thiên về mô tả, tổng hợp và tóm tắt khái niệm (Ví dụ: tóm tắt văn bản).

- **Phân vùng phân cấp** (the hierarchical partitioning approach): là khai thác tập phổ biến trong những cơ sở dữ liệu lớn, dựa trên một cấu trúc dữ liệu mới gọi là danh sách mẫu phổ biến. Phương pháp này phân vùng không gian tìm kiếm và không gian giải pháp và do đó chia cơ sở dữ liệu thành một tập các cơ sở dữ liệu con có kích thước có thể quản lý được.[14]

1.1.5 Phân loại các hệ thống khai thác dữ liệu[3]

- Phân loại dựa trên kiểu dữ liệu được khai thác: CSDL quan hệ, CSDL giao tác, CSDL hướng đối tượng, …

- Phân loại dựa trên dạng tri thức được khám phá: tóm tắt và mô tả, luật kết hợp, phân lớp, phân cụm,…

- Phân loại dựa trên lĩnh vực được áp dụng: thương mại, tài chính, y học…

- Phân loại dựa trên kỹ thuật được áp dụng: phân tích trực tuyến, máy học

1.1.6 Ứng dụng của khai thác dữ liệu[3]

- Tài chính và thị trường chứng khoán.

- Phân tích dữ liệu và hỗ trợ ra quyết định

- Điều trị y học và chăm sóc y tế

- Sản xuất và chế biến

- Text mining & Web mining

- Lĩnh vực khoa học

- Mạng viễn thông

- Bảo hiểm…

1.2 Cơ Sở Dữ Liệu Kích Thước Lớn.

Với sự tiến bộ của công nghệ thông tin và sự phổ biến của Internet thì dữ liệu được tạo ra và thu thập lại đã gia tăng đáng kể. Sự bùng nổ dữ liệu đã dẫn đến một nhu cầu cấp thiết cho các công nghệ và các công cụ có thể chuyển đổi dữ liệu thành thông tin và kiến thức bổ ích. Một trong những công nghệ hiệu quả nhất là khai thác dữ liệu, đó là công nghệ dùng để lấy ra các mẫu có ích hay tri thức từ số lượng lớn các dữ liệu[7]

Theo [5], Năm 2009 MỘT VI-RÚT CÚM mới được phát hiện. Kết hợp các yếu tố của các vi_rút gây cúm gà, chủng loại mới này, được gọi là H1N1, đã lây lan nhanh chóng. Trong vài tuần, các cơ sở y tế khắp thế giới lo sợ một đại dịch khủng kiếp xảy ra. Hy vọng duy nhất của cơ quan y tế là giảm mức lây lan. Nhưng để làm điều đó, họ cần biết bệnh lây lan tới đâu.

Google có thể đạt được điều này bằng cách xem xét những gì người sử dụng trên Internet. Bởi Google nhận được hơn 3 tỷ câu hỏi tìm kiếm mỗi ngày và lưu giữ tất cả chúng, nên nó có vô số dữ liệu để phân tích. Phần mền của họ tìm thấy sự kết hợp của 45 điều kiện tìm kiếm mà khi sử dụng cùng với mô hình toán học, có một mối tương quan mạnh mẽ giữa phỏng đoán của họ và các số liệu chính thức trên toàn quốc. Do vậy, hệ thống của Google đã chỉ báo có ích hơn và nhanh hơn, bằng cách xây dựng trên "dữ liệu lớn" khả năng của xã hội khai thác thông tin theo những cách thức mới để đưa ra những kiến thức hữu ích hay những sản phẩm và dịch vụ có giá trị đáng kể.

Một ví dụ khác là dự án mang tên Farecast của Oren Etzioni. Bằng cách dự báo giá của một vé máy bay có thể tăng hoặc giảm, và tăng hoặc giảm bao nhiêu. Farecast xử lý gần 200tỷ bản ghi giá vé máy bay để đưa ra các dự báo của nó. Microsoft đã mua lại và tích hợp vào công cụ tìm kiếm Bing, năm 2012 hệ thống đã khuyến cáo đúng 75% và tiết kiệm trung bình 50$ cho mỗi vé.

Với thông tin, cũng như với vật lý, kích thước là quan trọng. Do đó, Google có thể làm điều này bằng cách kết hợp hàng trăm tỷ từ khóa tìm kiếm và nó có thể đưa ra câu trả lời gần như trong thời gian thực, nhanh hơn nhiều các nguồn chính thức. Tương tự như vậy, Farecast của Oren Etzioni có thể dự đoán sự biến động giá của một chiếc vé máy bay và do đó chuyển quyền lực kinh tế đáng kể vào tay người tiêu dùng. Nhưng cả hai đều làm tốt như vậy bằng cách phân tích hàng trăm tỷ điểm dữ liệu.

Hai ví dụ trên cho thấy tầm quan trọng về khoa học và xã hội của dữ liệu lớn cũng như mức độ mà dữ liệu lớn có thể trở thành một nguồn giá trị kinh tế. Chúng đánh dấu hai cách thức mà thế giới dữ liệu đã sẵn sàng để cải tổ tất cả mọi thứ, từ các doanh nghiệp và các ngành khoa học tới chăm sóc sức khỏe, chính phủ, giáo dục, kinh tế, nhân văn, và mọi khía cạnh khác của xã hội.

Để đánh giá mức độ cuộc cách mạng thông tin đã tiến triển tới đâu, ta hãy xem xét các xu hướng xuyên suốt của các lĩnh vực của xã hội. Như trong thiên văn học, Trạm quan sát bầu trời bằng kỹ thuật số Sloan- SDSS (Sloan Digital Sky Survey) ở New Mexico, đến năm 2010 lưu trữ của trạm đã bạt ngàn với con số khổng lồ 140 tera (10 mũ 12) byte thông tin. Dự kiến năm 2016 kính thiên văn Large Synoptic Survey- LSST ở Chile cứ mỗi năm ngày sẽ thu thập được lượng dữ liệu tương đương như thế. Trong Y học, khi các nhà khoa học lần đầu giả mã gen người vào năm 2003, họ đã mất tới một thập kỷ làm việc miệt mài để xác định trình tự cho ba tỷ cặp cơ sở. Bây giờ, sau một thập kỷ một thiết bị đơn lẻ cũng có thể xác định trình tự cho số lượng DNA như vậy chỉ trong một ngày. Trong ngành tài chính, khoảng 7 tỷ cổ phiếu được bán mỗi ngày trên các thị trường chứng khoán Mỹ, trong số đó hai phần ba được giao dịch bằng các thuật toán máy tính dựa trên mô hình toán học xử lý hàng núi dữ liệu để dự đoán lợi nhuận trong khi cố gắng giảm thiểu rủi ro. Còn trong Internet đặc biệt bị tràn ngập. Google xử lý hơn 24 peta (10 mũ 15) byte dữ liệu mỗi ngày. Một nghiên cứu toàn diện của Martin Hilbert (Trường truyền thông và Báo trí Annenberg thuộc Địa học Nam Califonia) thì có hơn 300 exa (10 mũ 18) byte dữ liệu lưu trữ đã tồn tại vào năm 2007 (một exa byte là 1 tỷ giga byte). Facebook nhập hơn 10 triệu ảnh mới được tải lên mỗi giờ, các thành

viên Facebook nhấp nút “like” hoặc gửi lời bình gần ba tỷ lần mỗi ngày. Trong khi đó 800 triệu người sử dung dịch vụ Youtube của Google tải lên hơn một giờ video mỗi giây. Thành viên của mạng Twitter tăng khoản 200% mỗi năm và đến năm 2012 đã có hơn 400 triệu tweet mỗi ngày.

Từ khoa học tới y tế, từ ngân hàng tới Internet, các lĩnh vực có thể khác nhau, nhưng cùng nhau, chúng đều có cùng một câu chuyện tương tự: số lượng dữ liệu trong thế giới tăng rất nhanh, vượt sức không chỉ những chiếc máy tính mà cả trí tưởng tượng của chúng ta.

Như vậy dữ liệu lớn đánh dấu một bước quan trọng trong việc tìm kiếm của con người về định lượng và hiểu thế giới; một ưu thế của những thứ chưa bao giờ đo lường, lưu trữ, phân tích và chia sẻ trước khi dữ liệu hóa. Việc khai thác lượng lớn dữ liệu thay vì chỉ một phần nhỏ, và việc đặc quyền với nhiều dữ liệu với nhiều dữ liệu có độ chính xác thấp hơn, sẽ mở ra cánh cửa tới những cách hiểu biết mới. Nó dẫn xã hội tới việc từ bỏ ưu tiên lâu đời cho nhân quả và trong nhiều trường hợp thu được các lợi ích của mối tương liên.

CHƯƠNG 2 KHAI PHÁ TẬP PHỔ BIẾN

2.1 Phương pháp tìm tập phổ biến

1. Định nghĩa độ phổ biến:[2]

Cho CSDL giao dịch D và tập dữ liệu $X \subseteq I$. Độ phổ biến của X trong D, kí hiệu $\sigma(X)$, được định nghĩa là số giao dịch mà X xuất hiện trong D.

2. Định nghĩa tập phổ biến:

Tập $X \subseteq I$ được gọi là phổ biến nếu $\sigma(X) \geq minSup$ (với minSup là giá trị do người dùng chỉ định).

Một số tính chất:[2]

1. Mọi tập con của tập phổ biến đều phổ biến, nghĩa là $\forall X \subseteq Y$, nếu $\sigma(Y) \geq minSup$ thì $\sigma(X) \geq minSup$

2. Mọi tập cha của tập không phổ biến đều không phổ biến, nghĩa là $\forall Y \supseteq X$, nếu $\sigma(X) < minSup$ thì $\sigma(Y) < minSup$

2.2 Thuật toán Apriori

Input:
D, tập hợp các giao dịch
min_*sup, độ hỗ trợ nhỏ nhất*

Output: L, tập phổ biến trong D

Mô tả:

Bước 1: Với i =1, đếm số support cho mỗi tập C_i gồm một phần tử (*i- item*) và xem chúng như là một Large Itemset, L_i. Và minimum Support của chúng chính là min_sup

Bước 2: Với tập Large Item, L_i, bổ sung vào một item và tạo thành một tập phổ biến có *i+1* phần tử (*(i+1)-itemset*), C_{i+1}, tập này còn gọi là tập ứng viên (Candidate Itemset) và mỗi phần tử *i-itemset* phải thuộc L_i. Đếm số Support cho mỗi phần tử của tập C_{i+1}. Tạo ra tập Large Item L_{i+1}, với mỗi phần tử là một ứng viên thuộc tập C_{i+1} và thỏa min_sup.

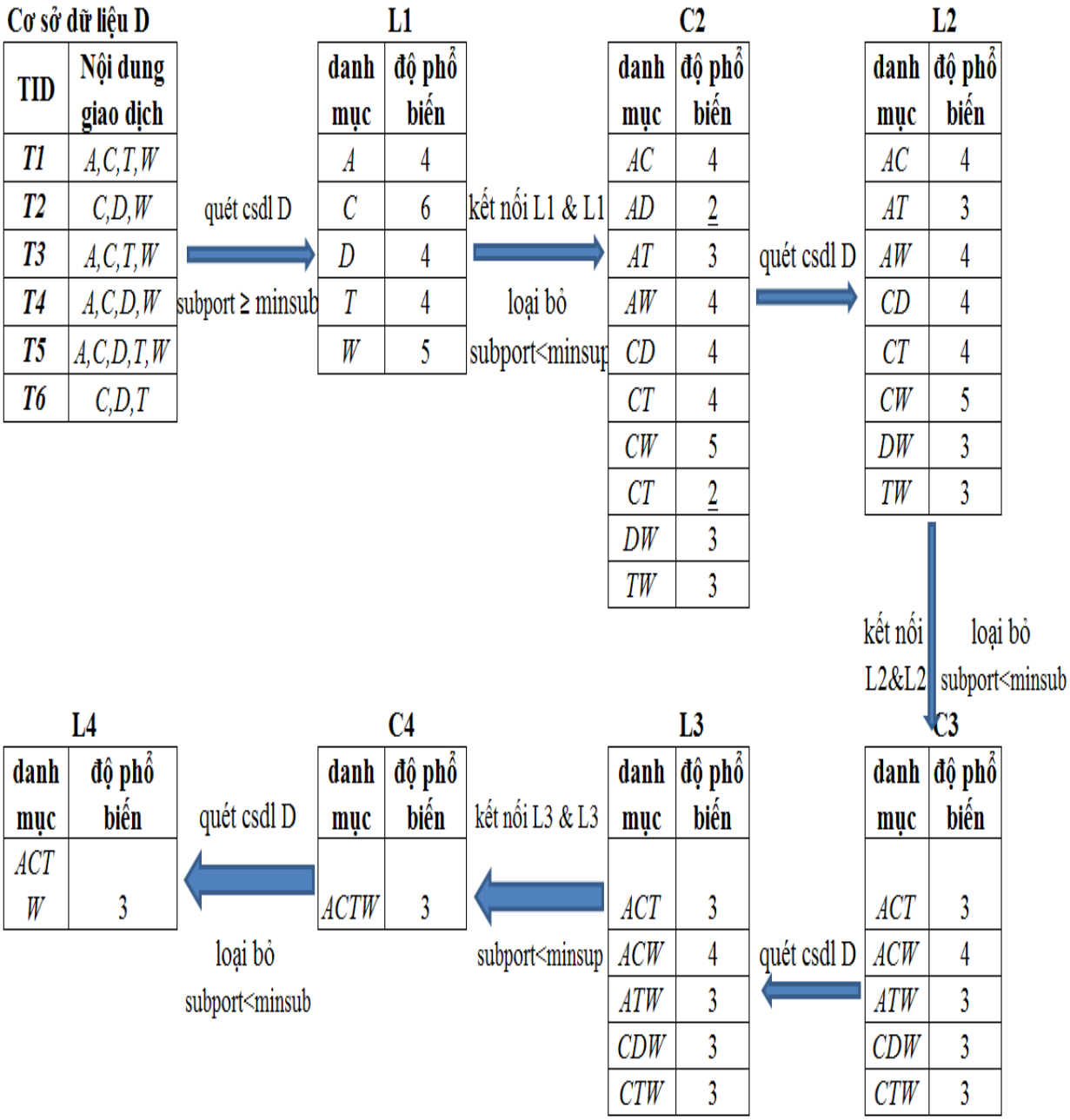
Bước 3: Lặp lại Bước 2, để tạo ra các tập Large Item mới. Thuật toán dừng khi không tạo được tập Large Item nào nữa.

Ví dụ : Giả sử tập các item I = { *a, c, d, t, w* } và cơ sở dữ liệu giao dịch: D = {<1, {*a, c, t, w*}>, <2,{*c, d, w* }>, <3, {*a, c, t, w*}>, <4, {*a, c, d, w*}>}, 5{*a, c, d, t, w*}>, <6, {*c, d, t*}>}.

Với minsup = 0.5 (tức tương đương 3 giao dịch). Khi thực hiện thuật toán Apriori trên ta có:

Bảng 2.1: Cơ sở dữ liệu mẫu

Mã giao dịch	**Nội dung giao dịch**
T1	*A,C,T,W*
T2	*C,D,W*
T3	*A,C,T,W*
T4	*A,C,D,W*
T5	*A,C,D,T,W*
T6	*C,D,T*



Hình 2.1: Minh hoạ thuật toán Apriori tìm tập mục phổ biến

2.3 Phương pháp dựa trên cây FP-Tree

2.3.1 Cấu trúc cây FP-Tree [4], [6]

Cấu trúc FP-Tree (Frequent Pattern tree) được giới thiệu đầu tiên bởi các tác giả J.Han, J.Pei và Y.Yin trong bài báo [6] đã khắc phục được điểm của thuật toán Apriori là phải phát sinh và kiểm tra một lượng lớn các ứng viên.

Cấu trúc FP-tree được dùng để tổ chức lại CSDL cho thuận lợi hơn trong quá trình tìm tập phổ biến, đồng thời các thông tin được nén trong cây FP-tree với tỉ lệ tương đối cao. Những thuận lợi đó là:

Những danh mục không đủ độ phổ biến được loại ngay từ đầu, vì vậy việc tìm tập phổ biến chỉ thao tác trên một số lượng danh mục nhỏ hơn nhiều so với toàn bộ các danh mục.

Nhiều giao dịch sẽ được nén trong cây FP-tree và việc này giúp giảm bớt khá nhiều thao tác trong quá trình xác định độ phổ biến của tập danh mục.

Cấu trúc FP-tree cho phép thực hiện tìm kiếm theo chiều sâu và áp dụng mô hình chia để trị khá hiệu quả.

Cây FP-tree là cấu trúc cây với một số đặc điểm như sau:

Có một nút cha được đánh nhãn NULL, những nút con nối với nút cha là những thành phần chung của nhiều giao dịch được nén lại với nhau (*item prefix subtree*), bên cạnh cũng có một mảng các danh mục đơn phổ biến *(frequent-item header table).*

Mỗi nút trong *item prefix subtree* có ba trường dữ liệu: mã danh mục, số tích lũy và con trỏ liên kết. Mã danh mục tương ứng danh mục mà nút này đại diện, số tích lũy là số giao dịch có chứa chung phần danh mục này, con trỏ liên kết dùng để liên kết 2 nút đại diện chung một mã danh mục ở hai *item prefix subtree* khác nhau. Giá trị con trỏ liên kết mang giá trị rỗng khi là nút cuối cùng trong chuỗi liên kết.

Mỗi phần tử trong *frequent-item header table* gồm 2 trường: mã danh mục và con trỏ liên kết đến đầu nút của chuỗi liên kết các nút cùng đại diện chung cho một mục danh mục.

2.3.2 Xây dựng cây FP-tree

Thuật toán tạo cây FP-tree [4], [6]:

Duyệt toàn bộ CSDL và xác định thứ tự của các danh mục giảm dần theo độ phổ biến và được đưa vào trong f-list. Dựa vào ngưỡng phổ biến người dùng đưa vào sẽ xác định những danh mục nào được tạo trong FP-tree và sắp xếp các danh mục trong từng giao dịch theo thứ tự trong f-list. Sau đó tạo cây FP-tree bằng cách lần lượt xét từng giao dịch trong CSDL đã được sắp xếp và loại bỏ những danh mục không đạt ngưỡng phổ biến.

Hàm createFPtree()

INPUT: *CSDL và ngưỡng phổ biến min_sup*

OUTPUT: *cấu trúc dữ liệu FP-tree của CSDL***.**

Các bước thực hiện:

Bước 1: Duyệt toàn bộ CSDL và tính độ phổ biến của từng danh mục. Sau đó xác định những danh mục có độ phổ biến hơn ngưỡng phổ biến minSup và sắp xếp giảm dần theo độ phổ biến trong f-list.

Bước 2: Tạo cây FPtree chỉ có một nút gốc được gán nhãn "null", ký hiệu Root

Bước 3: Với mỗi giao dịch trong CSDL thực hiện chọn và sắp xếp những danh mục phổ biến theo thứ tự trong *f-list*.

- Giao dịch đang xét được ký hiệu là *[p/rlist]* gồm 2 phần:

p là phần danh mục đầu tiên

rlist là phần danh mục còn lại bên phải của giao dịch

(không kể những danh mục không thỏa ngưỡng phổ biến).

- Gọi hàm **insert_tree***([p / rlist], root***)**

Hàm insert_tree(*List:[p/rlist], Node***)**

Các bước thực hiện:

Bước 1: - So sánh *p* với các nút con của Node *(child)*, nếu nhãn của *p* trùng với nhãn của nút con *(p.item-name = childe.item-name)* thì tăng chỉ số đếm của các nút con thêm 1.

- Nếu *p* khác nhãn các nút con, hoặc nút con rỗng thì tạo một nút con mới, khởi tạo chỉ số đếm là 1, tạo liên kết với nút trong cây có cùng nhãn.

Bước 2: Nếu ***rlist*** chưa rỗng thì gọi hàm **insert_tree** *(rlist, child).*

Ví dụ minh họa xây dựng cây FP-tree:

Quá trình xây dựng cây FP-tree sẽ được thực hiện qua ví dụ xây dựng cây tương ứng với CSDL mẫu ở bảng 2.1 để tìm tập phổ biến thỏa ngưỡng minSup=2

Tìm các danh mục đơn phổ biến

Quét CSDL tính độ phổ biến của từng danh mục và sắp xếp giảm dần trong mảng các danh mục đơn phổ biến **f-list**, xác định những danh mục có độ phổ biến không nhỏ hơn minSup sẽ được tạo trong FP-tree:

Bảng 2.2: Mảng thứ tự danh mục đơn phổ biến f-list

STT	Mã danh mục	Độ phổ biến	Con trỏ liên kết
1	*C*	6	Null
2	*W*	5	Null
3	*A*	4	Null
4	*D*	4	Null
5	*T*	4	Null

Sắp xếp thứ tự các danh mục trong từng giao dịch theo thứ tự trong f-list:

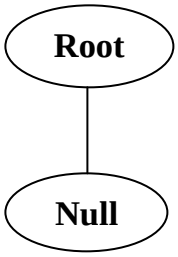
Quét CSDL lần 2, với mỗi giao dịch chọn và sắp lại thứ tự các danh mục theo thứ tự trong bảng 2.1 Ta có CSDL sau khi sắp xếp như sau:

Bảng 2.3: CSDL sau khi sắp xếp theo thứ tự trong f-list

Mã giao dịch	**Nội dung giao dịch**	**Nội dung giao dịch sau khi sắp xếp theo thứ tự mới**
T1	*A,C,T,W*	*C,W,A,T*
T2	*C,D,W*	*C,W,D*
T3	*A,C,T,W*	*C,W,A,T*
T4	*A,C,D,W*	*C,W,A,D*
T5	*A,C,D,T,W*	*C,W,A,D,T*
T6	*C,D,T*	*C,D,T*

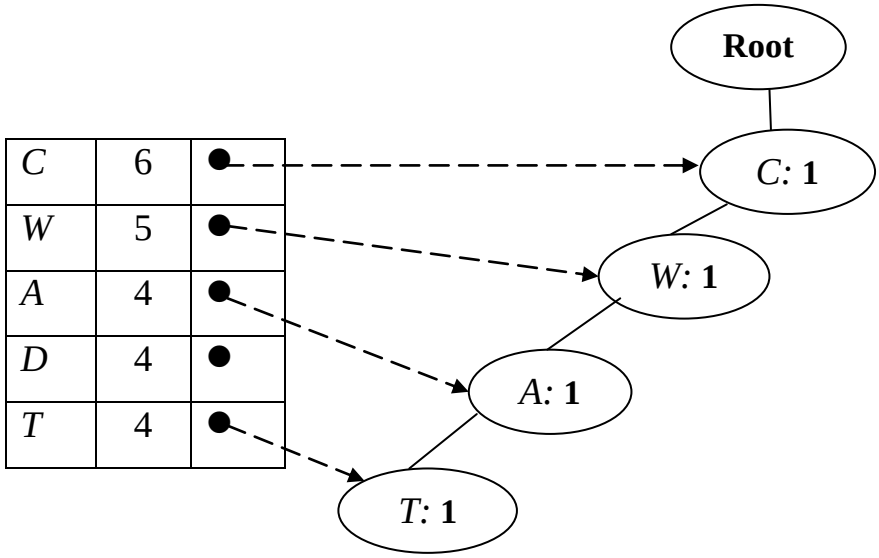
Cây FP-tree khi mới khởi tạo:

C	6	●
W	5	●
A	4	●
D	4	●
T	4	●



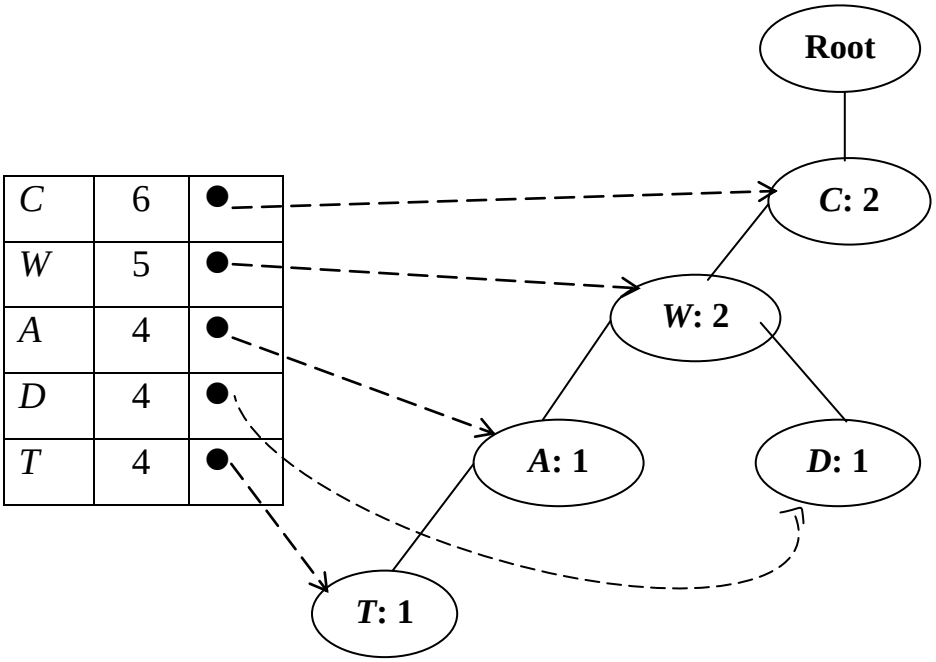
Hình 2.2: Cây FP-tree mới khởi tạo

Cây FP-tree sau khi đọc giao dịch 1: CWAT



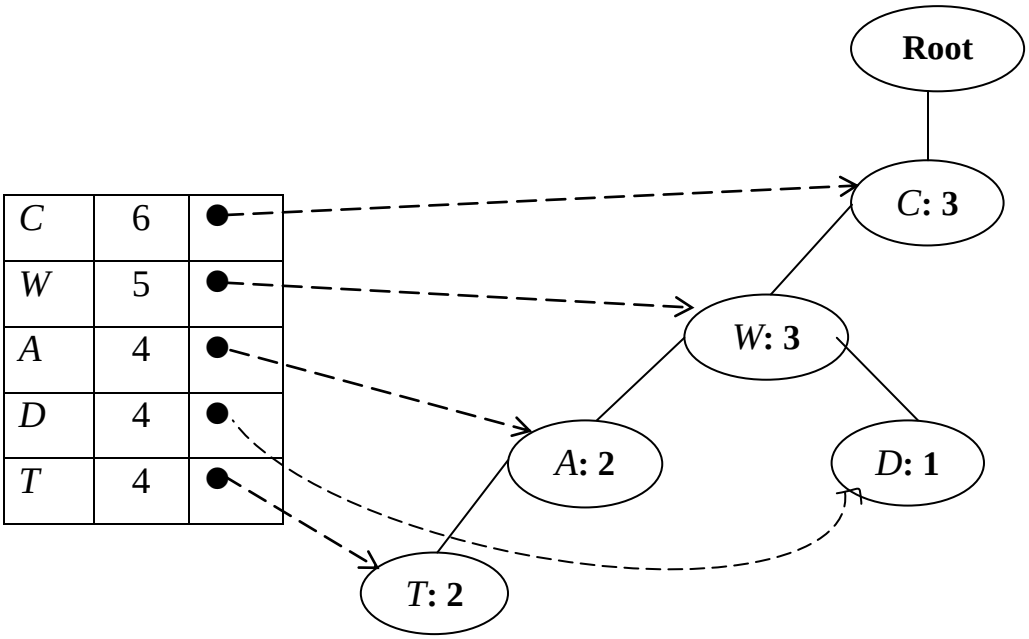
Hình 2.3: Cây FP- tree sau khi đọc giao dịch CWAT

Cây FP-tree sau khi đọc giao dịch 2: CWD



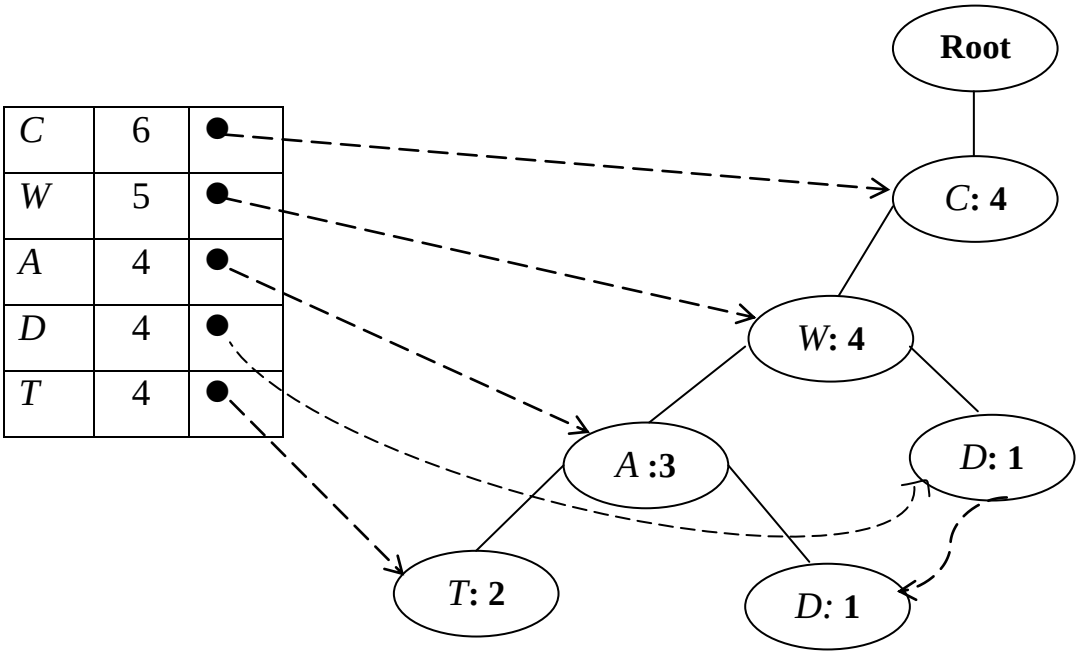
Hình 2.4: Cây FP-tree sau khi đọc giao dịch CWD

Cây FP-tree sau khi đọc giao dịch 3: CWAT



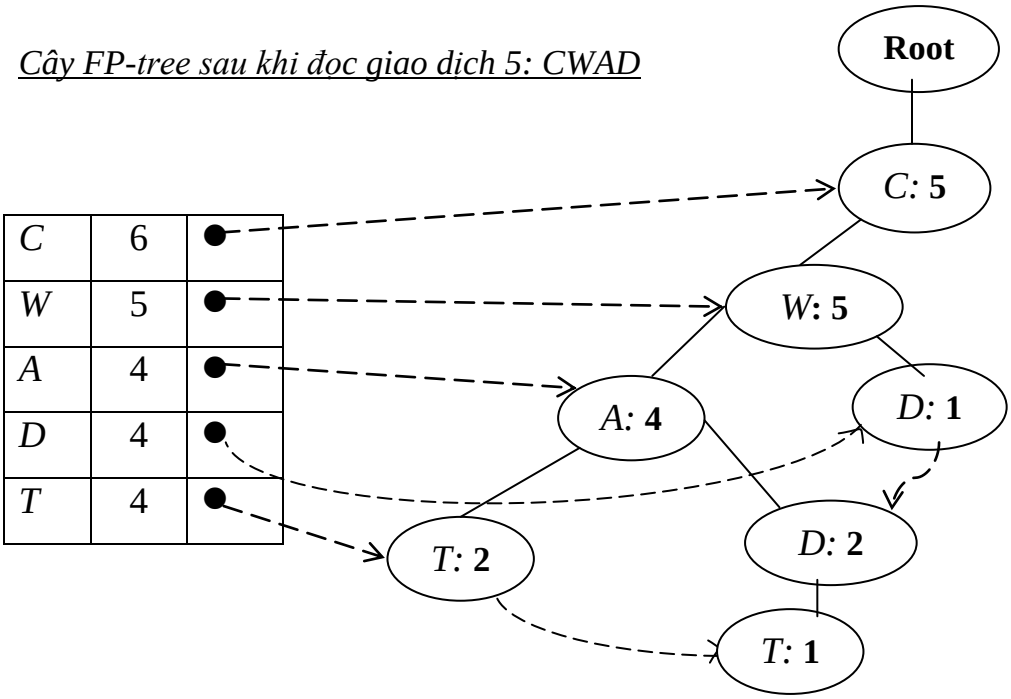
Hình 2.5: Cây FP-tree sau khi đọc giao dịch CWAT

Cây FP-tree sau khi đọc giao dịch 4: CWAD



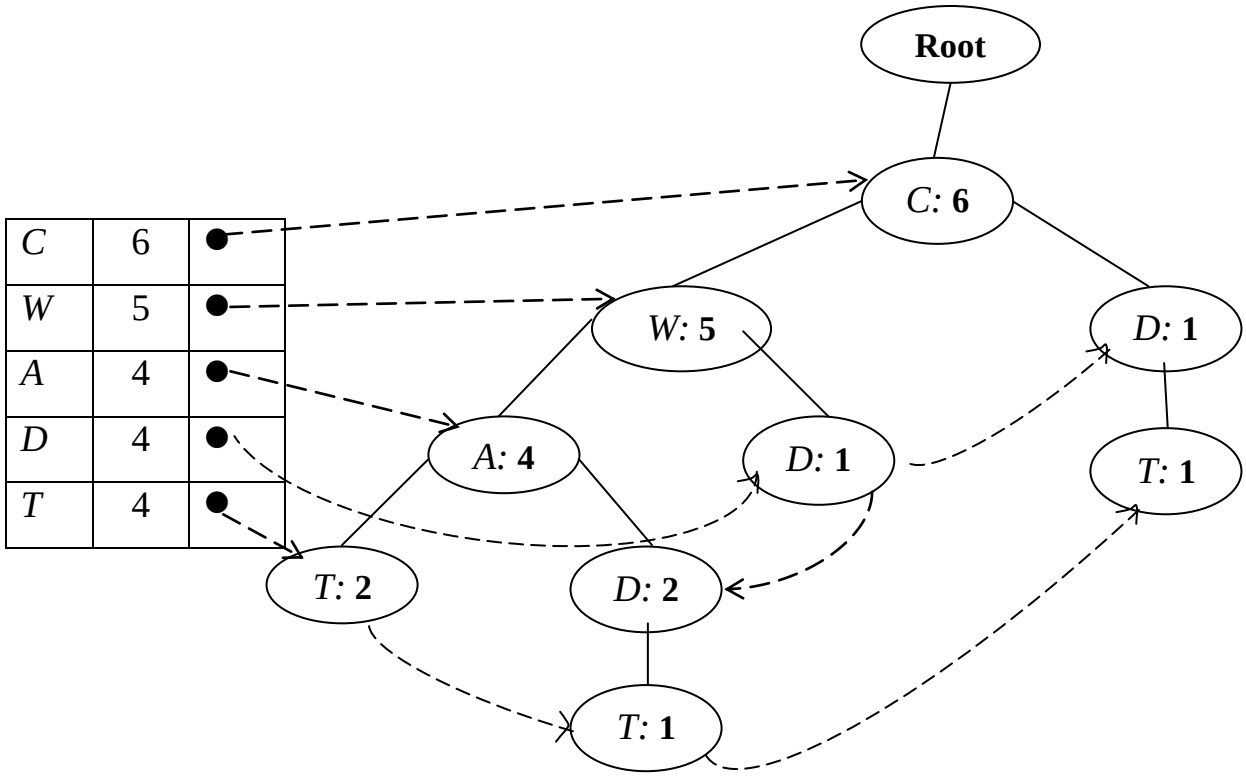
Hình 2.6: Cây FP-tree sau khi đọc giao dịch CWAD

Cây FP-tree sau khi đọc giao dịch 5: CWAD



Hình 2.7: Cây FP-tree sau khi đọc giao dịch CWADT

Sau khi đọc giao dịch cuối cùng CDT ta có cây FP-tree ứng với minSup=2:



Hình 2.8: Cây FP-tree toàn cục

2.3.3 Phép chiếu trên cây FP-tree

Sau khi xây dựng cấu trúc FP-tree cho toàn bộ CSDL chỉ gồm những danh mục đơn thỏa ngưỡng phổ biến, chúng ta phải duyệt cây để tìm ra những tập phổ biến thỏa minSup. Hiệu quả của quá trình khai thác phụ thuộc vào nhiều phương pháp duyệt. Phương pháp duyệt phải thỏa những yêu cầu:

- Đảm bảo kết quả tập phổ biến là đầy đủ
- Kết quả các tập phổ biến không bị trùng lặp
- Những tập phổ biến tạo ra thỏa ngưỡng minSup.

Để duyệt cây FP-tree, ta có thể sử dụng một trong 2 phép chiếu dưới đây:

Phép chiếu từ dưới lên:

Dựa trên thứ tự của f-list, chọn danh mục hạt giống bắt đầu từ danh mục có độ phổ biến nhỏ nhất thỏa minSup cho đến danh mục có độ phổ biến lớn nhất.

Trên cây FP-tree, duyệt từ những nút chứa danh mục hạt giống tiến dần đến nút gốc để xây dựng f-list cục bộ và FP-tree cục bộ của danh mục hạt giống.

Nếu duyệt hết danh mục trong f-list thì quay lui một bước và thực hiện tiếp.

Phép chiếu từ trên xuống:

Dựa trên thứ tự của f-list, chọn danh mục hạt giống bắt đầu từ danh mục có độ phổ biến lớn nhất cho đến danh mục có độ phổ biến nhỏ nhất thỏa minSup.

Trên cây FP-tree, duyệt từ nút chứa danh mục hạt giống tiến dần xuống nút lá cây và xây dựng f-list cục bộ của danh mục hạt giống. Ghi nhận vị trí của nút con trực tiếp của những nút chứa danh mục hạt giống trong f-list cục bộ.

Nếu f-list cục bộ không còn danh mục nào thì quay lui một bước và thực hiện tiếp.

2.3.4 Tìm các tập phổ biến với thuật toán FP-growth

Trong bài báo [6] các tác giả đã giới thiệu thuật toán FP-growth để duyệt cây FP-tree khá hiệu quả. Thuật toán Fp-growth duyệt cây bằng phép chiếu từ dưới lên theo phương pháp duyệt theo chiều sâu và dựa trên mô hình chia để trị.

Thuật toán FP-growth:

Hàm gen-FreqItemset()

INPUT: CSDL các giao dịch và ngưỡng phổ biến minSup

OUTPUT: Tập các tập phổ biến Fi thỏa ngưỡng phổ biến minSup

Các bước thực hiện:

Bước 1: $Tree_0$= createFPtree (CSDL, minSup)

Bước 2: FP-growth ($Tree_0$, null, minSup)

Hàm FP-growth (*Fp-tree, prefix, minSup***)**

Các bước thực hiện:

Bước 1: Nếu FP-tree chỉ có một đường đơn P thì chỉ cần tạo ra những tập phổ biến kết hợp giữa prefix và các tổ hợp của những danh mục trong P và độ phổ biến bằng với giá trị tích lũy nhỏ nhất của những nút tham gia vào tổ hợp. Sau khi phát sinh xong thì kết thúc hàm.

Bước 2: Ngược lại, lần lượt xét từng phần tử a trong f-list của FP-tree và phát sinh tập phổ biến (prefix $\cup$ a) có độ phổ biến bằng sup(a).

Bước 2.1: Duyệt cây FP-tree từ chuỗi các nút đại diện cho danh mục a, bắt đầu bằng con trỏ liên kết của phần tử a trong f-list và hướng lên nút gốc. Sau khi duyệt xong ta có được CSDL các giao dịch chiếu trên tập danh mục (prefix $\cup$ a) , ký hiệu là : **CSDL** $_{\{prefix \cup a\}}$

Bươc 2.2: Tree $_{\{\text{prefix} \cup \text{a}\}}$= createFPtree ($CSDL_{\{prefix \cup a\}}$, minSup)

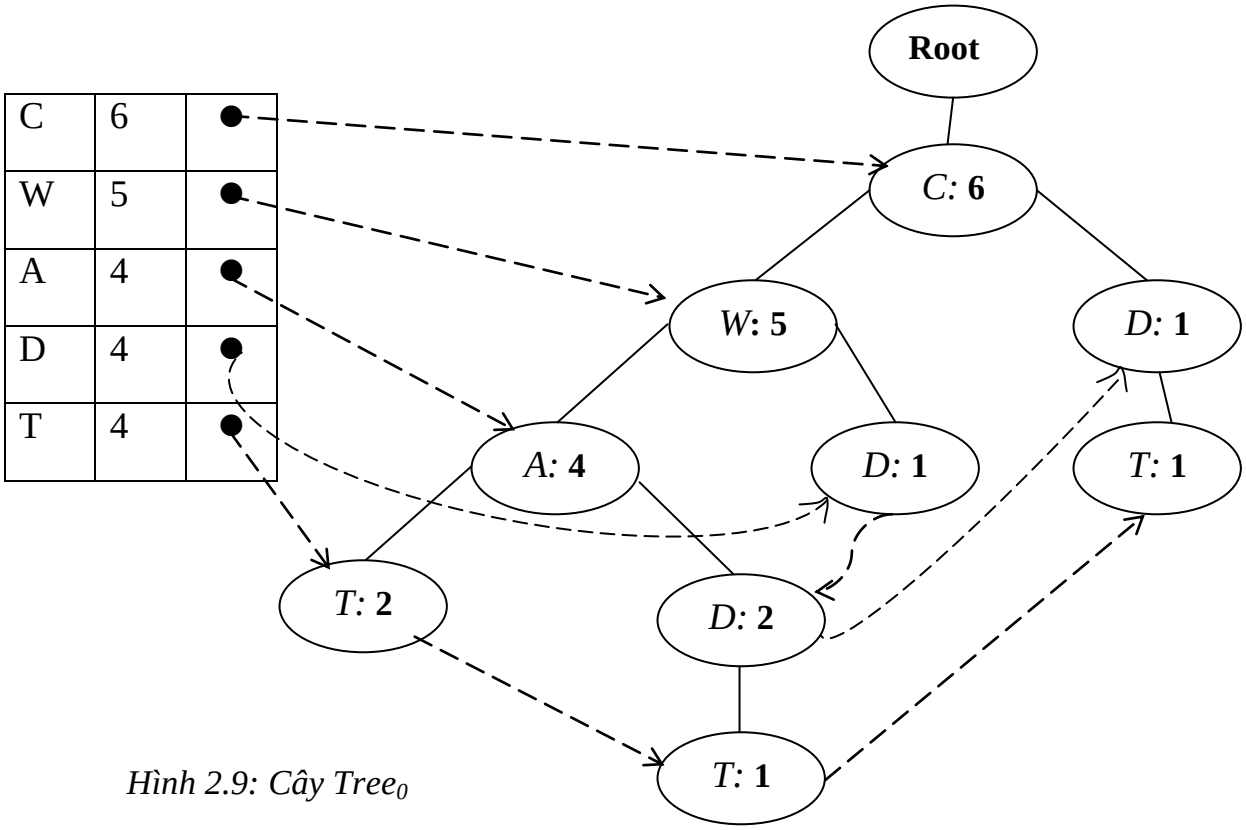
Bước 2.3: Nếu Tree $_{\{prefix \cup a\}} \neq \emptyset$ thì gọi hàm:

FP-growth ($Tree_{\{prefix \cup a\}}$, (prefix $\cup$ a), minSup)

Ví dụ minh họa thuật toán FP-growth:

Thực hiện thuật toán FP-growth để tìm các tập phổ biến trong CSDL mẫu ở bảng 2.1 với ngưỡng phổ biến minSup=2. Sau khi tạo được $Tree_0$, gọi hàm:

FP-growth (*$Tree_0$, null, minSup***)**



Hình 2.9: Cây Tree$_0$

Vì Tree$_0$ không phải đường đơn, nên xét danh mục (T) trong F-list và phát sinh tập phổ biến (T: 4). Sau đó chiếu trên Tree$_0$ để tạo CSDL $_{\{T\}}$:

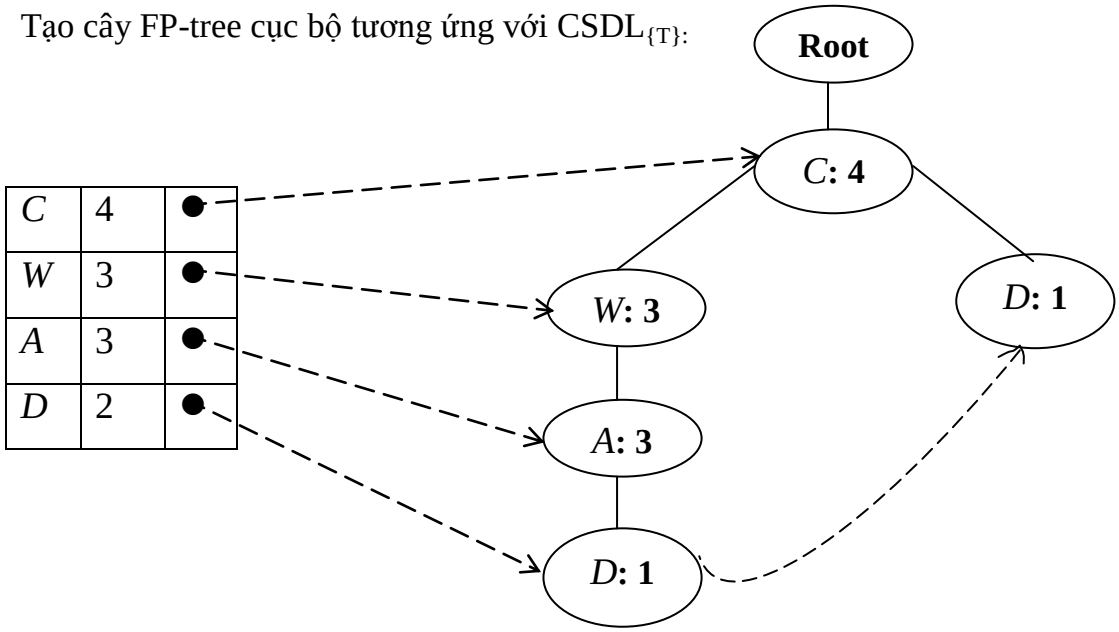
Chiếu trên tập danh mục **(*T*:4)**

Cơ sở dữ liệu cục bộ của các giao dịch chứa tập danh mục **{*T*}**

Bảng 2.4: Nội dung CSDL{T}

Nội dung còn lại của các giao dịch có chứa tập danh mục {T}	**Giá trị tích lũy**
C, W, A	2
C, W, A, D	1
C, D	1

Tạo cây FP-tree cục bộ tương ứng với $CSDL_{\{T\}}$:



Hình 2.10: Tree{T} cục bộ tương ứng với CSDL{T}

Vì $Tree_{\{T\}} \neq \varnothing$ nên gọi đệ quy chiều sâu hàm FP-growth ($Tree_{\{T\}}$, (T), minSup)

Vì $Tree_{\{T\}}$ không phải đường đơn, nên xét danh mục (D) trong f-list và phát sinh tập phổ biến (TD: 2)

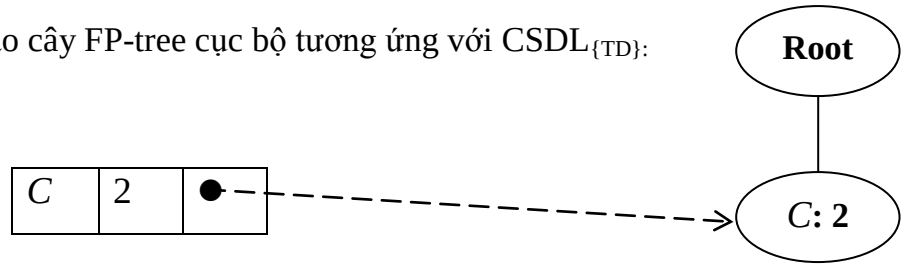
Chiếu trên tập danh mục (TD: 2)

Cơ sở dữ liệu cục bộ của các giao dịch chứa tập danh mục {*T, D*}

Bảng 2.5: Nội dung CSDL{TD}

Nội dung còn lại của các tập giao dịch có chứa tập danh mục {T, D}	**Giá trị tích lũy**
C	1
C,W,A	1

Tạo cây FP-tree cục bộ tương ứng với CSDL$_{\{TD\}}$:



Hình 2.11: Tree{TD} cục bộ tương ứng với CSDL{TD}

Vì Tree$_{\{TD\}} \neq \varnothing$ nên gọi đê quy hàm FP-growth (Tree $_{\{TD\}}$, (TD) , minSup)

VÌ Tree$_{\{TD\}}$ là đường đơn nên phát sinh tập phổ biến (TDC : 2)

Quay trở lại với Tree$_{\{T\}}$ xét tiếp danh mục (A) trong f-list của Tree{T} và phát sinh tập phổ biến (TA:3)

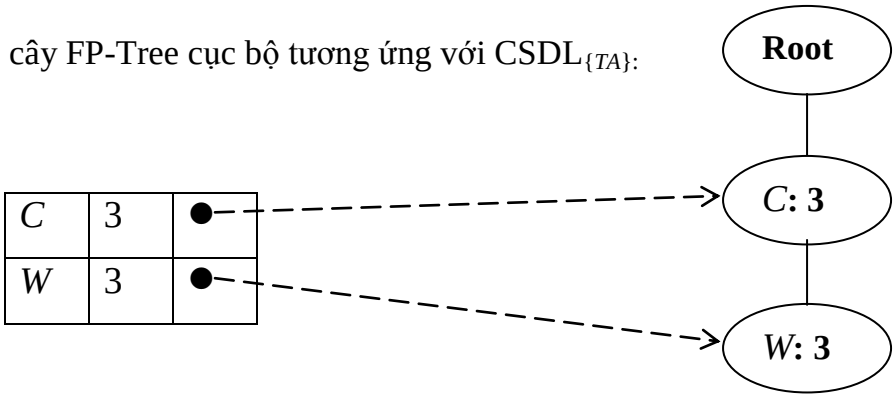
Chiếu tập danh mục (TA: 3):

Cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục {*T,A*}

Bảng 2.6: Nội dung CSDL{*TA*}

Nội dung còn lại của các giao dịch có chứa tập danh mục {*T, A*}	**Giá trị tích lũy**
C, W	3

Tạo cây FP-Tree cục bộ tương ứng với CSDL$_{\{TA\}}$:



Hình 2.12: Tree{TA} cục bộ tương ứng CSDL{TA}

Gọi hàm đệ quy FP-growth (*Tree* $_{\{TA\}}$ **,** *minSup và phát sinh được các tập phổ biến (TAC :3) ; (TAW : 3); (TACW :3***)**

Quay trở lại với Tree$_{\{T\}}$ **xét tiếp danh mục (***W***) trong f-list của Tree{***T***} và phát sinh tập phổ biến (***TW* **:3)**

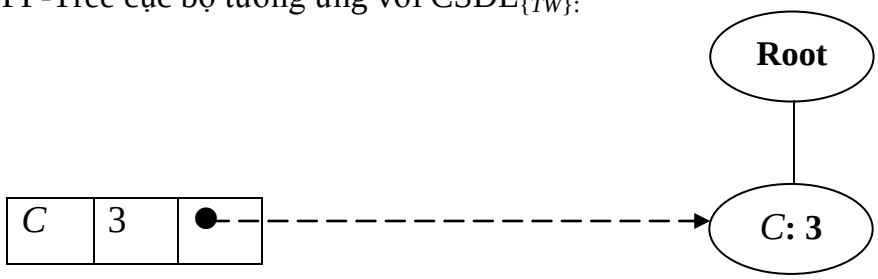
Chiếu trên tập danh mục (TW:3):

Cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục {*T, W*}

Bảng 2.7: Nội dung CSDL{*TW*}

Nội dung còn lại của các giao dịch có chứa tập danh mục {*T, W*}	**Giá trị tích lũy**
C	3

Tạo cây FP-Tree cục bộ tương ứng với CSDL$_{\{TW\}}$:



Hình 2.13: Tree{TW} cục bộ tương ứng với CSDL{TW}

Gọi hàm đệ quy FP-growth (*Tree*$_{\{TW\}}$**,** *(TW), minSup) và phát sinh được các tập phổ biến (TWC : 3)*

Quay trở lại với Tree$_{\{T\}}$ **xét tiếp danh mục (***C***) trong f-list của Tree**$_{\{T\}}$ **và phát sinh tập phổ biến (** *TC* **:4)**

Chiếu trên tập danh mục (TC :4)

$CSDL_{\{TC\}}= \varnothing$ dẫn đến $Tree_{\{TC\}} =\varnothing$

Quay trở lại với $Tree_0$ xét tiếp danh mục (*D*) trong f-list của $Tree_0$ và phát sinh tập phổ biến (*D*: 4)

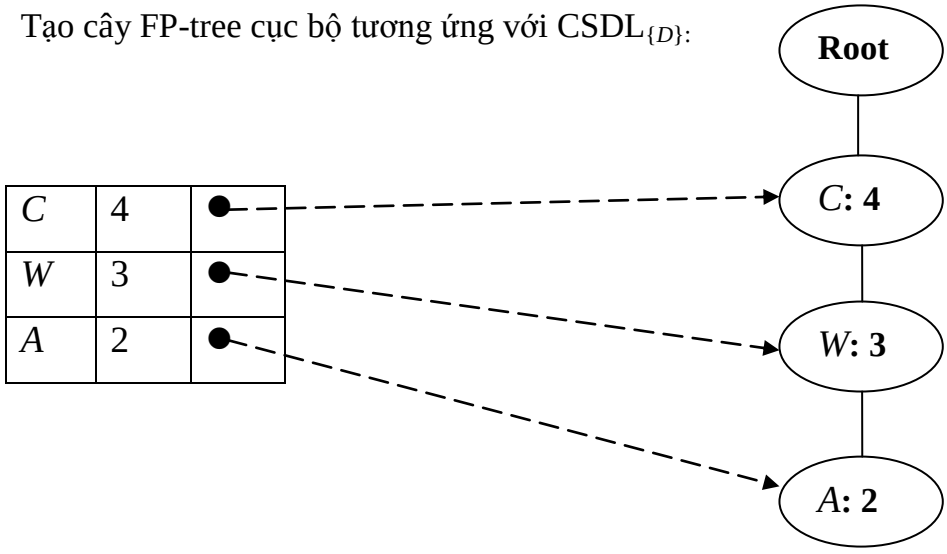
Chiếu trên tập danh mục (D :4):

Ta có cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục {*D*}

Bảng 2.8: Nội dung CSDL{D}

Nội dung còn lại của các giao dịch có chứa tập danh mục { *D*}	**Giá trị tích lũy**
C, W, A	2
C, W	1
C	1

Tạo cây FP-tree cục bộ tương ứng với $CSDL_{\{D\}}$:



Hình 2.14: Tree{D} cục bộ tương ứng với CSDL{D}

Gọi hàm đệ quy FP-growth ($Tree_{\{D\}}$, (*D*) , *minSup*) và phát sinh được các tập phổ biến : (*DA* :2) ; (*DW* :3) ; (*DC* :4) ; (*DAW* :2) ; (*DAC* :2) ; (*DWC* :3) ; (*DAWC* :2)

Quay trở lại với $Tree_0$ xét tiếp danh mục (*A*) trong f-list của Tree0 và phát sinh tập phổ biến (*A*:4) :

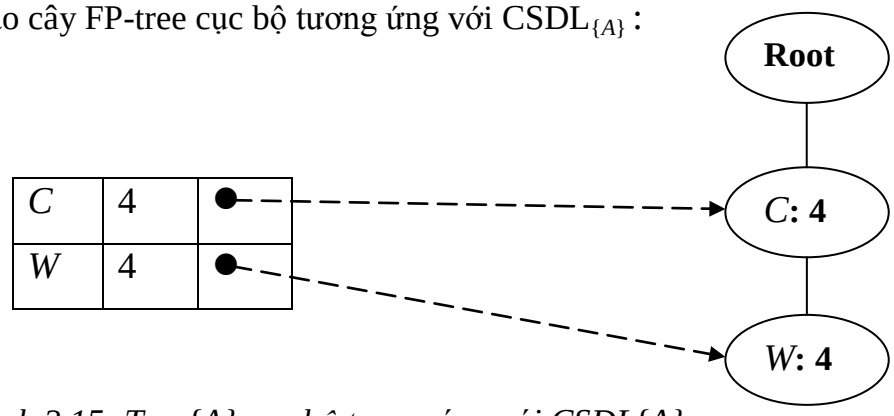
Chiếu trên tập danh mục (A:4):

Ta có cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục {*A*}

Bảng 2.9: Nội dung CSDL{*A*}

Nội dung còn lại của các giao dịch có chứa tập danh mục {*A*}	**Giá trị tích lũy**
C, W	4

Tạo cây FP-tree cục bộ tương ứng với $CSDL_{\{A\}}$:



Hình 2.15: Tree{A} cục bộ tương ứng với CSDL{A}

Gọi hàm đệ quy FP-growth (*Tree* $_{\{A\}}$, *(A) , minSup*) và phát sinh được các tập phổ biến : (*AW* : 4) ; (*AC* :4) ; (*AWC* :4)

Quay trở lại với $Tree_0$ xét tiếp danh mục (*W*) trong *f-list* của Tree0 và phát sinh tập phổ biến (*W* :5)

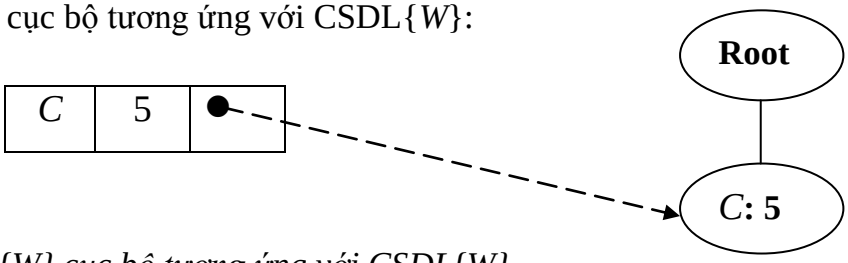
Chiếu trên tập danh mục (W :5):

Ta có cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục {*W*}

Bảng 2.10: Nội dung CSDL{*W*}

Nội dung còn lại của các giao dịch có chứa tập danh mục {*W*}	**Giá trị tích lũy**
C	5

Tạo cây FP-tree cục bộ tương ứng với CSDL{*W*}:



Hình 2.16: Tree{W} cục bộ tương ứng với CSDL{W}

Gọi đệ hàm quy FP-growth (*$Tree_{\{W\}}$, (W), minSup*) và phát sinh được tập phổ biến : (*W* :5)

Quay trở lại với Tree0 xét tiếp danh mục (*C*) trong *f-list* của $Tree_0$ và phát sinh tập phổ biến (*C* :6)

Chiếu trên tập danh mục (C :6)

CSDL cục bộ của các giao dịch có chứa tập danh mục {*C*} : CSDL{*C*} = ∅

Cấu trúc cây FP-tree cục bộ tương ứng với $CSDL_{\{C\}}$: $Tree_{\{C\}}$ = ∅

Thuật toán kết thúc.

CHƯƠNG 3 PHƯƠNG PHÁP PHÂN VÙNG, PHÂN CẤP TRONG KHAI PHÁ TẬP PHỔ BIẾN

3.1 Giới thiệu

Theo [14], với sự tiến bộ của công nghệ thông tin và sự phổ biến của Internet thì dữ liệu được tạo ra và thu thập lại đã gia tăng đáng kể. Sự bùng nổ dữ liệu đã dẫn đến một nhu cầu cấp thiết cho các công nghệ và các công cụ có thể chuyển đổi dữ liệu thành thông tin và kiến thức bổ ích. Một trong những công nghệ hiệu quả nhất là khai thác dữ liệu, đó là công nghệ dùng để lấy ra các mẫu có ích hay tri thức từ số lượng lớn các dữ liệu.

Nhiệm vụ cơ bản và quan trọng nhất của khai thác dữ liệu là khai thác tập phổ biến, đó là tập các mặt hàng được thường xuyên mua cùng với nhau trong một giao dịch. Tập phổ biến đại diện cho các thuộc tính nội tại và quan trọng của CSDL giao tác và cung cấp nền tảng cho nhiều tác vụ khai thác dữ liệu thiết yếu như sự kết hợp phân tích tương quan, phân tích mẫu, phân loại, phân tích cụm và lưu trữ dữ liệu. Để nâng cao hiệu quả trong khai thác tập phổ biến, một số phương pháp mới và cấu trúc dữ liệu linh hoạt đã được phát triển, chẳng hạn như FP-tree, danh sách mẫu phổ biến và danh sách mẫu giao dịch

Tuy nhiên, khi CSDL ngày càng phình to ra, thậm chí cấu trúc dữ liệu linh hoạt sẽ phát triển ra khỏi dung lượng bộ nhớ thì khả năng mở rộng các phương pháp khai thác dữ liệu là một vấn đề phải được giải quyết. Các phương pháp thông thường sử dụng một lược đồ phân vùng phẳng để phân vùng CSDL ban đầu thành một tập các CSDL con nhỏ hơn ở cùng cấp độ và sau đó tìm các tập phổ biến cục bộ trong các CSDL con này. Xử lý cuối cùng là quét lại CSDL ban đầu để kiểm tra xem các tập phổ biến cục bộ có phải là phổ biến toàn cục hay không. Điều này, tất nhiên phải mất thêm thời gian và tình hình sẽ xấu đi khi khai thác tập phổ biến đóng bởi vì không chỉ tần số của tập phổ biến được tính toán mà việc kiểm tra tập hợp con cũng phải được thực hiện

Trong nghiên cứu này, một phương pháp thay thế là phương pháp phân vùng thứ bậc được đề xuất để khai thác tập phổ biến trong những cơ sở dữ liệu lớn, dựa trên một cấu trúc dữ liệu mới gọi là danh sách mẫu phổ biến FPL (Frequent Pattent List-FPL). Phương pháp này phân vùng không gian tìm kiếm và không gian giải pháp và do đó chia cơ sở dữ liệu thành một tập các cơ sở dữ liệu con có kích thước có thể quản lý được. Kết quả là một cách tiếp cận chia để trị có thể được phát triển để thực hiện nhiệm vụ khai thác dữ liệu mong muốn mà không cần quét lại cơ sở dữ liệu ban đầu để tìm giải pháp cuối cùng. Vì các cơ sở dữ liệu con trong phân vùng thứ bậc chỉ chứa các mẫu phổ biến nên việc quét lại cơ sở dữ liệu để kiểm tra ngưỡng hỗ trợ tối thiểu là không cần thiết như trong phân vùng phẳng. Do đó, phân vùng thứ bậc có thể cải thiện tốc độ trong khai thác tập phổ biến từ cơ sở dữ liệu lớn.

Định nghĩa: (khai thác tập phổ biến) Cho I = {i1, i2, ..., in} là một tập hợp các mục. Một tập các mục X là một tập con không rỗng của I. Một CSDL giao tác DB là một tập các giao dịch. Mỗi giao dịch Tx = <tid, X>, với tid là một định danh giao dịch duy nhất và X là một tập các mục. Một giao dịch Tx = <tid, X> chứa một tập các mục Y, nếu Y $\subseteq$ X. Độ hỗ trợ (số lần xuất hiện) của một tập các mục Y, ký hiệu là sup (Y), là số lượng các giao dịch có chứa Y. một tập các mục Y là phổ biến nếu độ hổ trợ của nó không ít hơn độ hỗ trợ tối thiểu *t*. Nghĩa là, Y là phổ biến nếu sup(Y) >= *t*.

Cho một cơ sở dữ liệu giao tác DB và một độ hỗ trợ tối thiểu t, vấn đề của việc tìm kiếm tập hoàn chỉnh của các tập phổ biến được gọi là vấn đề khai thác tập phổ biến.

3.2 Danh sách mẫu phổ biến (FPL) dùng để khai thác tập phổ biến

Trong luận văn này trình bày ngắn gọn về Danh sách mẫu phổ biến (FPL) [9] [11] dùng để khai thác tập phổ biến. Các FPL là một danh sách tuyến tính các nút mục (item node), mỗi nút mục tương ứng với một mục phổ biến trong cơ sở dữ liệu. Các giao dịch trong CSDL được chuyển đổi thành chuỗi bit và được gọi là ký hiệu giao dịch (transaction signatures), với bit 1 đại diện cho sự có mặt và bit 0 đại diện cho sự

vắng mặt của các mục phổ biến trong các giao dịch. Sau đó, các ký hiệu giao dịch được gián tới các nút mục tùy theo vị trí của bit 1 cuối cùng của chúng.

Một ví dụ để minh họa cách xây dựng FPL. Dữ liệu trong Bảng 3.1 là một CSDL ký hiệu giao dịch (ký hiệu là DB), trong đó có 10 giao dịch (từ T1 đến T10) và 6 mục riêng biệt (mục *f, c, a, b, m, p*). Độ hỗ trợ tối thiểu cho tập phổ biến được thiết lập là 3, có nghĩa là nếu một tập các mục là một tập phổ biến thì nó phải xuất hiện trong 3 giao dịch hoặc nhiều hơn trong DB. Để thuận tiện, chỉ có các mục phổ biến mới được liệt kê trong Bảng 3.1 và chúng ta thêm vào chuỗi bit đại diện cho các giao dịch để tạo thuận lợi cho việc giải thích sau này.

Bảng 3.1: ví dụ CSDL giao tác DB

Transaction ID	**Frequent items**	***Bit-string representation***					
		f	**c**	**a**	**b**	**m**	**p**
T1	**f, c, a, m, p**	1	1	1	0	1	1
T2	**f, c, a, b, m**	1	1	1	1	1	0
T3	**f, b**	1	0	0	1	0	0
T4	**c, b, p**	0	1	0	1	0	1
T5	**f, c, a, m, p**	1	1	1	0	1	1
T6	**f, c, a, b, m, p**	1	1	1	1	1	1
T7	**f, c, a, b, m, p**	1	1	1	1	1	1
T8	**f**	1	0	0	0	0	0
T9	**c**	0	1	0	0	0	0
T10	**f, a**	1	0	1	0	0	0
Item frequency		**8**	**7**	**6**	**5**	**5**	**5**

Sau đó, quá trình xử lý quét DB lần thứ hai. Đối với mỗi giao dịch Tx trong DB, các mục phổ biến được lựa chọn và sắp xếp theo tần số của chúng trong FPL, và một chuỗi bit, được gọi là một ký hiệu giao dịch (transaction signature) được xây dựng từ MSB (Most Significant Bit - bit quan trọng nhất) đến LSB (Least Lignificant Bit - bit ít quan trọng nhất) như sau: duyệt qua FPL theo thứ tự giảm dần tần số của các mục. Tại

mỗi nút mục, nếu mục tương ứng của nó chứa trong Tx thì đặt bit là 1, ngược lại đặt bit là 0. Có nghĩa là các bit trong ký hiệu giao dịch cho thấy sự xuất hiện hay không xuất hiện của các mục phổ biến trong Tx. Chúng ta giữ lại các bit từ MSB đến bit 1 cuối (bit ngoài cùng bên phải) và cắt bỏ tất cả các ký hiệu giao dịch có bit cuối là bit 0. Sau đó, ký hiệu giao dịch Tx có bit cuối là 1 được lưu trữ trong nút mục của FPL. Các FPL được xây dựng từ CSDL DB được trình bày trong hình 3.1.

Item Node	**node 1**	**node 2**	**node 3**	**node 4**	**node 5**	**node 6**
Item-name	f	**c**	a	b	m	P
Count	8	7	6	5	5	5
Transaction Set (transaction signatures)	T8 1	T9 01	T10 101	T3 1 0 0 1	T2 1 1 1 1 1	T1 1 1 1 0 1 1 T4 0 1 0 1 0 1 T5 1 1 1 0 1 1 T6 1 1 1 1 1 1 T7 1 1 1 1 1 1

Hình 3.1: Các FPL được xây dựng từ DB trong Bảng 3.1.

Các FPL có một số tính chất cần thiết, một tính chất quan trọng là tính phân vùng cơ sở dữ liệu. Có nghĩa là mỗi nút mục của FPL có thể được coi như là một CSDL con của toàn bộ CSDL ban đầu. Bằng cách này, FPL có thể phân vùng CSDL thành một tập các CSDL con. Ví dụ, CSDL DB ban đầu trong bảng 3.1 được phân chia bởi các FPL trong hình 3.1 thành một tập 6 CSDL con. CSDL con đầu tiên (nút mục 1) chứa giao dịch T8, CSDL con thứ hai chứa giao dịch T9 và CSDL con cuối (nút mục 6) chứa các giao dịch T1, T4, T5, T6, T7. Cũng nhận thấy rằng mục p chỉ được chứa bởi các giao dịch trong nút mục 6. Vì vậy, để khai thác các tập phổ biến có chứa mục p thì chỉ cần CSDL con cuối (nút mục 6). Các tính chất này mở đường cho việc khai thác tập phổ biến với FPL bằng cách chia để trị (phân vùng của phân vùng), phương pháp tiếp cận dựa trên sự lặp đi lặp lại, đó là cơ sở để phân vùng thứ bậc (được mô tả trong phần sau)

Ba hoạt động cơ bản - đếm bit, cắt tỉa ký hiệu và di chuyển ký hiệu - được thực hiện trên nút mục cuối cùng của FPL để tìm các tập phổ biến. Việc đếm bit cho biết số lượng bit 1 trong các ký hiệu giao dịch để tính toán độ hỗ trợ của các mục. Việc cắt tỉa ký hiệu cắt tỉa bit 1 cuối cùng của các ký hiệu giao dịch và sau đó cắt tỉa các bit 0 theo sau. Cuối cùng, việc di dời ký hiệu di chuyển các ký hiệu được cắt tỉa từ nút mục cuối cùng đến các nút khác tùy theo vị trí của bit 1 cuối cùng trong ký hiệu đó.

Với các tính chất ở trên và các hoạt động cơ bản của FPL, việc khai thác các tập phổ biến có thể được thực hiện lặp đi lặp lại thao tác phân vùng của phân vùng. Nghĩa là từ các ký hiệu giao dịch trong phân vùng cuối cùng (nút mục cuối cùng) của FPL, chúng tôi đếm số bit 1 trên mỗi vị trí bit để tính số lần xuất hiện của mục tương ứng. Những mục này có số lượng vượt qua ngưỡng hỗ trợ tối thiểu, được kết hợp lại để tạo ra các tập phổ biến có chứa mục cuối cùng của FPL.

Sau đó, các bit ít quan trọng nhất (LSBs) của các ký hiệu giao dịch này được cắt tỉa bởi vì những bit này cho biết sự có mặt của mục cuối cùng, mà mục này đã được bao gồm trong các tập phổ biến đã được tìm ra trước đó. Các bit 0 theo sau của ký hiệu đã được cắt tỉa cũng được loại bỏ bởi vì các mục tương ứng không có mặt trong các giao dịch này để tạo ra các tập phổ biến. Sau đó, các ký hiệu đã được cắt tỉa xong ở trên được chuyển qua các phân vùng khác (các nút mục) tùy theo vị trí của bit 1 ít quan trọng nhất của chúng.

Vì tất cả các tập phổ biến có chứa các mục cuối cùng đã được tìm ra nên nút mục cuối cùng có thể được loại bỏ, kết quả dẫn đến một FPL ngắn hơn. Tiếp tục trên FPL ngắn hơn này, chúng tôi truy cập nút mục cuối cùng (mới) của nó để tìm các tập phổ biến có chứa mục cuối cùng (mới). Quá trình tiếp tục cho đến khi tất cả các nút mục của FPL đều được truy cập.

3.3 Phân vùng thứ bậc với danh sách mẫu phổ biến

Khi CSDL càng ngày càng lớn hơn, bất kỳ cấu trúc dữ liệu linh hoạt nào cũng sẽ phát triển ra khỏi dung lượng bộ nhớ. Trong trường hợp này, chúng ta có thể sử dụng tính chất phân vùng CSDL (như mô tả trong phần 3.2) của FPL để phát triển các phương pháp tiếp cận phân vùng thứ bậc nhằm khai thác các tập phổ biến trong CSDL lớn.

Các nguyên tắc được đưa ra như sau: khi CSDL ban đầu được đại diện bởi FPL toàn cục của nó, nó được sắp xếp lại và phân chia thành một tập các CSDL (cấp đầu tiên) có kích thước nhỏ hơn. Điều này có nghĩa là, có một tính đối ngẫu giữa các nút mục trong FPL và các CSDL con: một nút của một FPL có thể được coi như là một CSDL con thường trú trong bộ nhớ, trong khi một CSDL con có thể được coi như là một nút mục của FPL thường trú trên đĩa. Kết quả là, ba hoạt động cơ bản – đếm bit, cắt tỉa ký hiệu, di chuyển ký hiệu - được thực hiện trên FPL, cũng có thể được thực hiện tương tự trên các CSDL con khi chúng ta phân vùng thứ bậc CSDL để khai thác tập phổ biến.

Trong quá trình khai thác, chỉ có CSDL con cuối cùng là cần thiết để khai thác tập phổ biến. Có nghĩa là, mỗi nút mục của FPL toàn cục sẽ được coi như là một CSDL con (cấp đầu tiên), và chỉ có CSDL con cuối cùng (nút mục cuối cùng) là phải được đặt vào trong bộ nhớ để khai thác các tập phổ biến, với một FPL cục bộ (cấp đầu tiên) được xây dựng cho CSDL con cuối cùng. Vì vậy, chúng ta không cần phải đặt toàn bộ FPL vào bộ nhớ, mỗi lần chúng ta chỉ đặt một nút mục vào bộ nhớ.

Nếu FPL cục bộ (cấp đầu tiên) của CSDL con cuối cùng (cấp đầu tiên) vẫn còn vượt quá bộ nhớ, chúng ta lại coi nút mục cuối cùng của FPL cục bộ (cấp đầu tiên) này như là một CSDL con (cấp thứ 2) và xây dựng FPL cục bộ (cấp thứ 2) tương ứng của nó. Một lần nữa, chỉ có nút mục cuối cùng của FPL cục bộ cấp thứ 2 này là phải được đưa vào bộ nhớ. Quá trình này có thể được tiếp tục cho đến khi FPL cục bộ hiện hành

có thể vừa với bộ nhớ. Sau đó, có thể dùng bất kỳ thuật toán nào dựa trên năng lực của bộ nhớ, chẳng hạn như FP-growth hoặc FPL-Mining, để khai thác các tập phổ biến.

3.3.1 Một ví dụ về phân vùng thứ bậc

Chúng ta sử dụng CSDL toàn cầu mẫu trong Bảng 3.1 để minh họa quá trình phân vùng thứ bậc cơ sở dữ liệu. Độ hỗ trợ cho tập phổ biến được thiết lập là 2. Dung lượng bộ nhớ tối đa được giả định là 24 bit cho các ký hiệu giao dịch được lưu trữ trong FPL. Dưới đây là quá trình phân vùng thứ bậc cơ sở dữ liệu:

Bước 1: Kiểm tra các chuỗi bit đại diện cho các giao dịch trong Bảng 3.1, chúng ta thấy rằng tổng số bit là 60, cao hơn so với dung lượng bộ nhớ là 24 bit. Vì vậy, chúng ta phải phân vùng CSDL toàn cầu trong bảng 3.1 thành một tập các CSDL con cấp đầu tiên, sử dụng FPL liên kết của nó trong hình 3.1 như là một lược đồ cho việc phân vùng. Kết quả được hiển thị trong hình 3.2. Để duy trì các cấu trúc file trong khi phân vùng thứ bậc, chúng ta lưu giữ một cấu trúc dữ liệu gọi là FileHeader như hình 3.3.

Sub-database	Sub-DB_f f: 8	Sub-DB_c c: 7	Sub-DB_a a: 6	Sub-DB_b b: 5	Sub-DB_m m: 5	Sub-DB_p p: 5
Transactions	T8 1	T9 01	T10 101	T3 1001	T2 11111	T1 111011 T4 010101 T5 111011 T6 111111 T7 111111

Hình 3.2: Các CSDL con cấp đầu tiên từ DB của Bảng 3.1.

Partition Level	Sub-database pointers						Parent itemset
1	Sub-DB_f	Sub-DB_c	Sub-DB_a	Sub-DB_b	Sub-DB_m	Sub-DB_p	Φ (null)

Hình 3.3: FileHeader sau khi phân vùng cấp đầu tiên từ DB của Bảng 3.1.

Từ hình 3.3, chúng ta thấy FileHeader có ba trường: cấp độ của phân vùng (partition level), con trỏ cơ sở dữ liệu con (sub-database pointers) và tập các mục cha (parent itemset). Mỗi khi việc phân vùng của CSDL được thực hiện ở mức độ sâu hơn, một bản ghi (record) mới được nối thêm vào FileHeader. Tuy nhiên, lưu ý là các CSDL con không cần phải được lưu trữ trong các tập tin riêng biệt. Thay vào đó, một số CSDL con có thể chia sẻ cùng một tập tin nếu kích thước của chúng không quá lớn. Do đó, con trỏ CSDL con của FileHeader có thể là con trỏ tập tin hoặc là chỉ mục trong một tập tin.

Bước 2: Kiểm tra Sub-DBp trong hình 3.2, chúng ta lưu ý rằng mỗi giao dịch trong Sub-DBp chứa mục *p*. Vì vậy, trong cột đại diện của mình, mục *p* có thể được bỏ qua, với mục *p* là tập các mục cha của các cấp phân vùng sâu hơn.

Điều này được thực hiện như sau: đối mỗi giao dịch trong Sub-DBp, chúng ta loại bỏ mục cuối cùng (mục *p*) để có được một phiên bản rút gọn, ký hiệu là Sub-DB'p và được trình bày trong bảng 3.2.

Bảng 3.2: CSDL con cấp đầu tiên đã được rút gọn Sub-DB'p

Transaction ID	Bit-string representation				
	f	*c*	*a*	*b*	*m*
T1	1	1	1	0	1
T4	0	1	0	1	0
T5	1	1	1	0	1
T6	1	1	1	1	1
T7	1	1	1	1	1
Item frequency	4	5	4	3	4

Bây giờ chúng ta sẽ quyết định có nên xây dựng một FPL trực tiếp cho Sub-DB'p hoặc phân vùng nó một lần nữa. Chúng ta thấy rằng có tổng cộng 25 bit trong Sub-DB'p, vẫn còn cao hơn so với dung lượng bộ nhớ là 24 bit. Do đó, chúng ta lại phải

phân vùng Sub-DB’p thành một tập các CSDL con cấp thứ 2, được mô tả trong bước tiếp theo.

Bước 3: Vì FPL có thể phân vùng cơ sở dữ liệu nên quá trình phân vùng CSDL con Sub-DB’p cũng giống như xây dựng FPL cho Sub-DB’p, ngoại trừ các nút mục được coi như là các CSDL con cấp thứ 2 và được lưu trữ trong đĩa chứ không phải là trong bộ nhớ. Sau khi quét Sub-DB'p hai lần (bởi quá trình xây dựng FPL) để tìm và sắp xếp các mục phổ biến, chúng ta thấy mục c có tần số cao nhất là 5, và mục b có tần số thấp nhất là 3.

Do đó, chúng ta phân vùng cấp thứ hai cho Sub-DB'p thành 5 CSDL con như hình 3.4. Theo đó, FileHeader phải được cập nhật như hình 3.5.

Sub-database	$Sub\text{-}DB_{pc}$ c: 5	$Sub\text{-}DB_{pf}$ f: 4	$Sub\text{-}DB_{pa}$ a: 4	$Sub\text{-}DB_{pm}$ m: 4	$Sub\text{-}DB_{pb}$ b: 3
Transaction Set	Empty	Empty	Empty	**T1** 1 1 1 1 **T5** 1 1 1 1	**T4** 1 0 0 0 1 **T6** 1 1 1 1 1 **T7** 1 1 1 1 1

Hình. 3.4: Phân vùng cấp thứ hai cho CSDL con Sub-DB'p trong Bảng 2.

Partition Level	**Sub-database pointers**						Parent itemset
1	$Sub\text{-}DB_f$	$Sub\text{-}DB_c$	$Sub\text{-}DB_a$	$Sub\text{-}DB_b$	$Sub\text{-}DB_m$	$Sub\text{-}DB_p$	Φ
2	$Sub\text{-}DB_{pc}$	$Sub\text{-}DB_{pf}$	$Sub\text{-}DB_{pa}$	$Sub\text{-}DB_{pm}$	$Sub\text{-}DB_{pb}$		{ p}: 5

Hình 3.5: FileHeader sau khi phân vùng cấp thứ hai cho CSDL con Sub-DB'p

Lưu ý rằng trong Hình 3.5, tập các mục cha của phân vùng cấp thứ hai được mô tả là {p}: 5. Điều này là do: như được chỉ ra trong bước (2), tất cả các CSDL con cấp thứ hai có nguồn gốc từ CSDL con cấp đầu tiên Sub-DBp và trong Sub-DB'p, tất cả năm giao dịch (T1, T4, T5, T6, T7) đều chứa mục p.

Bước 4: Bây giờ, hãy truy cập CSDL con cấp thứ hai cuối cùng Sub-DBpb trong hình 3.4. Tương tự như bước (2), chúng ta loại bỏ mục cuối cùng (mục b) để có được một CSDL con rút gọn là Sub-DB'pb, được trình bày trong Bảng 3.3.

Bảng 3.3: CSDL con Sub-DB'pb

Transaction ID	*Bit-string representation*			
	c	**f**	**a**	**m**
T4	1	0	0	0
T6	1	1	1	1
T7	1	1	1	1

Từ bảng 3.3, chúng ta quyết định xây dựng một FPL trực tiếp cho Sub-DB'pb hoặc phân vùng nó một lần nữa. Chúng ta thấy chỉ có 12 bit trong Sub-DB'pb, đã vừa với dung lượng bộ nhớ là 24 bit. Do đó, chúng ta không cần phải phân vùng Sub-DB'pb một lần nữa. Thay vào đó, một FPL có thể được xây dựng cho Sub-DB'pb và được lưu trữ trong bộ nhớ như Hình 3.6, với FileHeader cập nhật như Hình 3.7.

Item Node	Node 1 c: 3	Node 1 f: 2	Node 1 a: 2	Node 1 m: 2
Transaction Set	**T4** 1			**T6** 1111 **T7** 1111

Hình 3.6: FPL của CSDL con Sub-DB'pb trong Bảng 3.3.

Partition Level	**Sub-database pointers**						Parent itemset
1	Sub-DB_f	Sub-DB_c	Sub-DB_a	Sub-DB_b	Sub-DB_m	Sub-DB_p	Φ
2	Sub-DB_{pc}	Sub-DB_{pf}	Sub-DB_{pa}	Sub-DB_{pm}	Sub-DB_{pb}		{ p}: 5
Null	Pointer to the FPL for Sub-DB'_{pb}						{ p, b}: 3

Hình 3.7: FileHeader sau khi FPL được xây dựng cho Sub-DB'pb

Lưu ý rằng trong bản ghi cuối cùng của FileHeader trong hình 3.7, cấp độ phân vùng trở thành Null, bởi vì CSDL con Sub-DBpb không được phân vùng lần nữa. Thay vào đó, FPL thường trú trong bộ nhớ được xây dựng. Ngoài ra, tập các mục cha trong bản ghi cuối cùng của FileHeader được mô tả là {p, b}: 3 bởi vì FPL này được xây dựng cho Sub-DB'pb, và trong Sub-DB'pb, tất cả ba giao dịch đều chứa mục b, và Sub-DB'pb đã có một tập các mục cha là {p}: 5. Do đó, chúng ta có các nguyên tắc sau: trong phân vùng thứ bậc ở mức sâu hơn, tập các mục cha được hình thành bằng cách ghép các mục cuối cùng dọc theo tất cả các cấp phân vùng.

Bước 5: Từ FPL trong hình 3.6, chúng ta có thể sử dụng các thuật toán dựa trên năng lực bộ nhớ như FPL-Mining để tìm các tập phổ biến.

Bước 6: Sau này, chúng ta thực hiện việc cắt tỉa và di chuyển ký hiệu trong CSDL con Sub-DBpb trong hình 3.4. Trước tiên, chúng ta loại bỏ các bit 1 cuối cùng của các giao dịch T4, T6, T7. Sau đó, di chuyển T4 đến Sub-DBpc, di chuyển T6 và T7 đến Sub-DBpm. Các CSDL con kết quả (sau khi thực hiện di chuyển và loại bỏ) được trình bày trong hình 3.8. Sau đó, File Header trong hình 3.7 phải được cập nhật bằng cách loại bỏ bản ghi cuối cùng và vô hiệu hóa con trỏ chỉ đến Sub-DBpb. FileHeader kết quả được trình bày trong hình 3.9.

Sub-database	Sub-DB$_{pc}$ c: 5	Sub-DB$_{pf}$ f: 4	Sub-DB$_{pa}$ a: 4	Sub-DB$_{pm}$ m: 4
Transaction Set	**T4** 1	Empty	Empty	**T1** 1 1 1 1 **T5** 1 1 1 1 **T6** 1 1 1 1 **T7** 1 1 1 1

Hình 3.8: CSDL con cấp thứ 2 sau khi cắt và di chuyển trên Sub-DBpb trong hình.3.4

Partition Level	**Sub-database pointers**						Parent itemset
1	Sub-DB$_f$	Sub-DB$_c$	Sub-DB$_a$	Sub-DB$_b$	Sub-DB$_m$	Sub-DB$_p$	Φ
2	Sub-DB$_{pc}$	Sub-DB$_{pf}$	Sub-DB$_{pa}$	Sub-DB$_{pm}$	**Null**		{ p}: 5

Hình 3.9: File Header sau khi cắt và di chuyển trên Sub-DBpb trong hình.3.4.

Sau đó, quá trình tiếp tục với CSDL con Sub-DBpm trong hình 3.8 cho đến khi các CSDL con đều được truy cập để khai thác các tập phổ biến.

3.3.2 Các thuật toán để phân vùng thứ bậc CSDL và khai thác tập phổ biến

Thuật toán đầu tiên là FPL_HPDB [14], thuật toán này phân vùng thứ bậc CSDL giao tác cho đến khi các CSDL con cuối cùng cho phép một cấu trúc dữ liệu thường trú trong bộ nhớ (ví dụ: FPL) được xây dựng. Thuật toán thứ hai là FPL_HP_Mining [14], thuật toán này khai thác các tập phổ biến từ các CSDL được phân vùng thứ bậc. Lưu ý rằng một khi một cấu trúc dữ liệu thường trú trong bộ nhớ (ví dụ: FPL) có thể được xây dựng cho các CSDL con thì bất kỳ thuật toán nào dựa trên bộ nhớ để khai thác các tập phổ biến đều có thể được sử dụng. Trong mô tả thuật toán, chúng ta chọn thuật toán FPL-Mining dựa trên phương pháp FPL cho tác vụ này. Hai thuật toán FPL_HPDB và FPL_HP-Mining, được mô tả tương ứng trong hình 3.10 và 3.11

Algorithm FPL_HPDB (DB, t, *FileHeader, PartitionLevel, parent_itemset*)
/* Các tham số: (1) DB: CSDL giao dịch; (2) t: ngưỡng hỗ trợ tối thiểu; (3) *FileHeader*: cấu trúc dữ liệu lưu giữ cấu trúc file của các csdl con; (4) *PartiLevel:* cấp độ của các phân vùng csdl, khởi tạo là 1; (5) *parent_set*: itemset cha của DB. Trong lời gọi ban đầu, nó là 1 tập rỗng (null). */

begin /* bắt đầu thuật toán */

1. Quét csdl để tìm tất cả các items phổ biến và tần xuất của. Có *n* items phổ biến và đặt các items này vào 1 danh sách **L-items**, theo *thứ tự giảm dần của tần xuất*.
2. Quét csdl lần thứ 2 để tạo 1 csdl được cắt tỉa *DB-trimmed* bằng cách giữ lại các items phổ biến và cắt tỉa các items không phổ biến, và sắp xếp các items phổ biến theo thứ tự của chúng trong **L-items** cho mỗi giao dịch.
3. **If** (*DB-trimmed* có thể vừa với bộ nhớ) **then**
   xây dựng 1 FPL cho DB, và lưu giữ con trỏ chỉ tới FPL vào trong *FileHeader*, với *PartiLevel* thiết lập là Null;
   else do begin /* phân vùng *DB-trimmed* thành 1 tập của n csdl con, $n \geq 1$ */
   (1). Tạo *n* csdl con $Sub\text{-}DB_1$ đến $Sub\text{-}DB_n$ bằng cách làm theo các bước giống như khi xây dựng FPL, với các nút được xem như là các csdl con và được lưu trên.
   (2). Lưu các con trỏ file chỉ tới *n* csdl con này vào trong *FileHeader* và lưu *PartiLevel* và tập các *parent_* vào trong *FileHeader*.
   (3). Tăng *PartiLevel* lên 1
   end.
4. Đếm *số lượng các giao dịch* (*m*) trong $Sub\text{-}DB_n$, và loại bỏ item cuối cùng (item *n*) cho mỗi giao dịch trong $Sub\text{-}DB_n$, để thu được csdl rút gọn $Sub\text{-}DB_n'$;
5. Gọi thủ tục /* phân vùng thứ bậc $Sub\text{-}DB_n'$ */
   FPL_HPDB *(*$Sub\text{-}DB_n'$*, t, FileHeader, partiLevel, parent_set* $\cup$*{ item n}: m);*

end /* kết thúc thuật toán */.

Hình 3.10: Thuật toán FPL_HPDB.

Algorithm FPL_HP-Mining (*FileHeader, t*)
/* Khai thác mẫu phổ biến bằng cách phân vùng thứ bậc CSDL */
/* Các tham số: (1) *FileHeader*: cấu trúc dữ liệu lưu giữ cấu trúc file của các csdl đã được phân vùng; (2) *t:* ngưỡng hỗ trợ tối thiểu */

begin /* bắt đầu thuật toán */

6. Lấy FPL và *itemset* cha của nó (S) từ *FileHeader*, và gọi **FPL-Mining** (*FPL, n, t, S*) để tìm các itemsets phổ biến. /* n: độ dài của *FPL* */
7. Tạo một itemset phổ biến từ *itemset cha* của *FPL* (ví dụ, *S*), và sau đó xóa bản ghi cuối (bản ghi cho *FPL*) từ *FileHeader*.
8. **while** (*Fileheader* không rỗng) **do begin**

(1). Thực hiện việc cắt tỉa ký hiệu và di chuyển trên csdl con cuối cùng trong cấp độ phân vùng sâu nhất;
(2). **if** (chỉ có một csdl con (đối với item *x*) *Sub-DB*$_x$) **then**

i. Đếm số lượng các giao dịch của nó (*c*) và tạo một itemset phổ biến bằng cách ghép *itemset cha* của *Sub-DB*$_x$ với item *x*, tổng số itemset phổ biến này gán vào *c*;
ii. Tạo một itemset phổ biến từ *itemset cha* của *Sub-DB*$_x$
iii. Xóa bản ghi cho *Sub-DB*$_x$ từ *FileHeader*;

else /* Phân vùng thứ bậc và khai thác csdl con mới và cuối cùng */

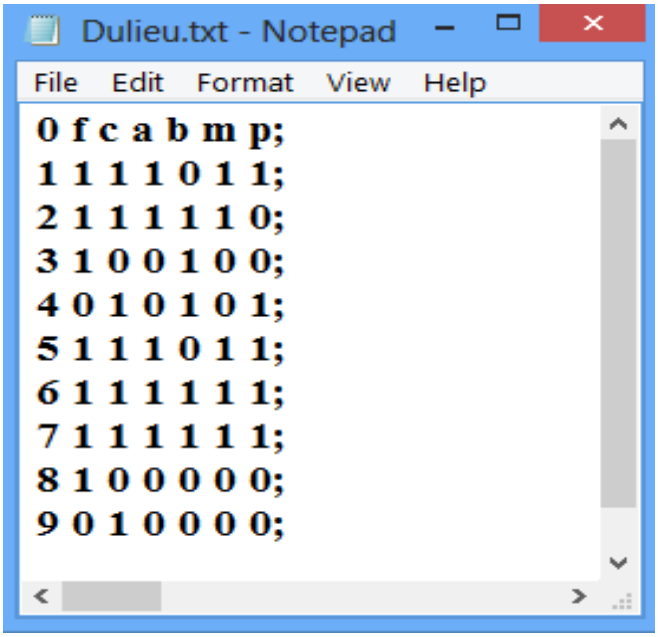
i. Từ *FileHeader*, lấy ra csdl con cuối cùng (*Sub-DB*$_y$) trong cấp độ phân vùng sâu nhất (*level_x*), và tìm *itemset cha* của nó (*S*);
ii. Đếm *số lượng các giao dịch* (*m*) trong *Sub-DB*$_y$, và với mỗi giao dịch Tx trong *Sub-DB*$_y$, loại bỏ item cuối cùng (item *n*) để thu được CSDL con rút gọn *Sub-DB*$_y$*'* ;
iii. Gọi thủ tục
FPL_HPDB *(Sub-DB*$_y$*' , t, FileHeader, Level_x +1, S* $\cup$ *{ item n}: m);*
iv. Gọi thủ tục **FPL_HP-Mining** (*FileHeader, t*);

end /* kết thúc while *FileHeader* không rỗng */
end /* kết thúc thuật toán */.

Hình 3.11: Thuật toán FPL_HP-Mining.

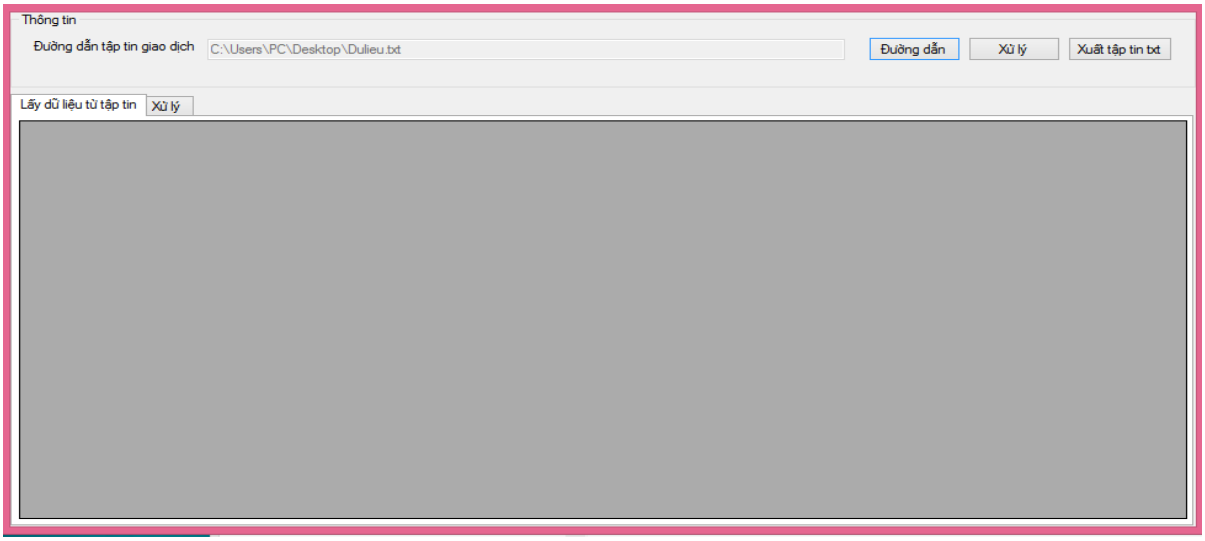
3.4 Kết quả thực nghiệm phân vùng phân cấp

Trong luận văn nghiên cứu phương pháp phân vùng phân cấp này này, chúng ta tiến hành thực nghiệm khi CSDL đã được mã hóa thành chuỗi các bit nhị phân, với bit 1 là đại diện cho sự xuất hiện của các mặt hàng, bit 0 là mặt hàng chưa xuất hiện. Dữ liệu được mã hóa và lưu vào file có dạng .txt, trong Hình 3.12 là tập tin CSDL đã được mã hóa, trong đó có 10 giao dịch (từ 1 đến 10) và có 6 mục (item) riêng biệt (f, c, a, b, m, p). Như vậy chúng ta thấy có tổng số bit là 60, dung lượng bộ nhớ tối đa được giả định là 24 bit.



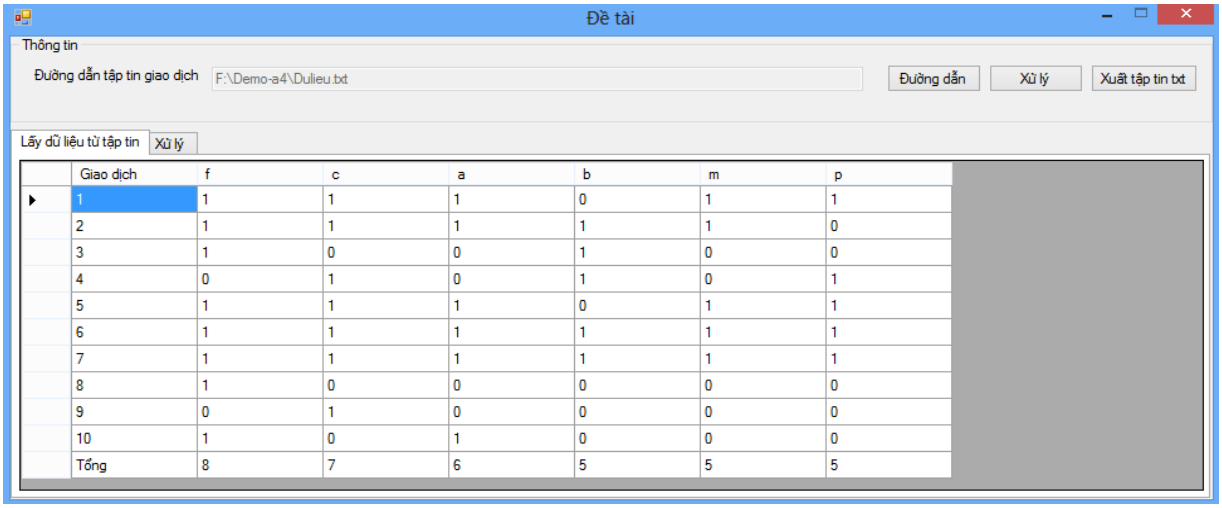
Hình 3.12: tập tin CSDL đã được mã hóa

Trước khi phân vùng dữ liệu để lấy ra các giao dịch phổ biến, chúng ta tạo đường dẫn như hình 3.13 để lấy tập tin cần xử lý.



Hình 3.13: tạo đường dẫn để lấy dữ liệu

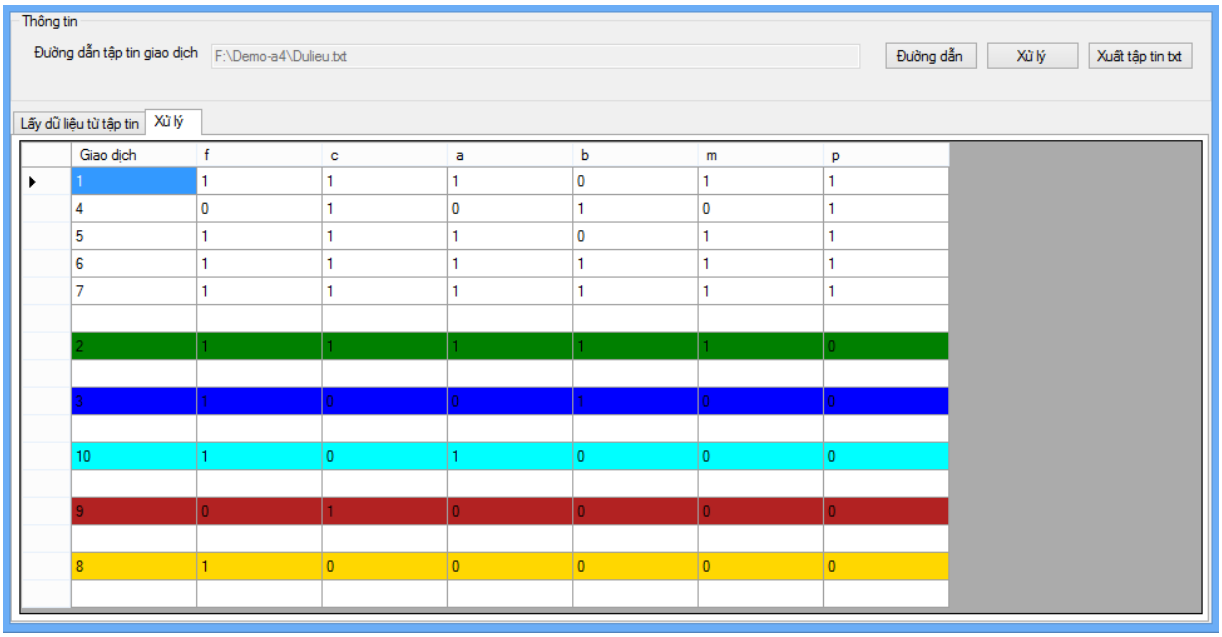
Sau đó chương trình sẽ xử lý sắp xếp theo tần số xuất hiện trong FPL và duyệt chúng theo thứ tự giảm dần tần số của các mục. Như hình. 3.14



Giao dịch	f	c	a	b	m	p
1	1	1	1	0	1	1
2	1	1	1	1	1	0
3	1	0	0	1	0	0
4	0	1	0	1	0	1
5	1	1	1	0	1	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	0	0	0	0	0
9	0	1	0	0	0	0
10	1	0	1	0	0	0
Tổng	8	7	6	5	5	5

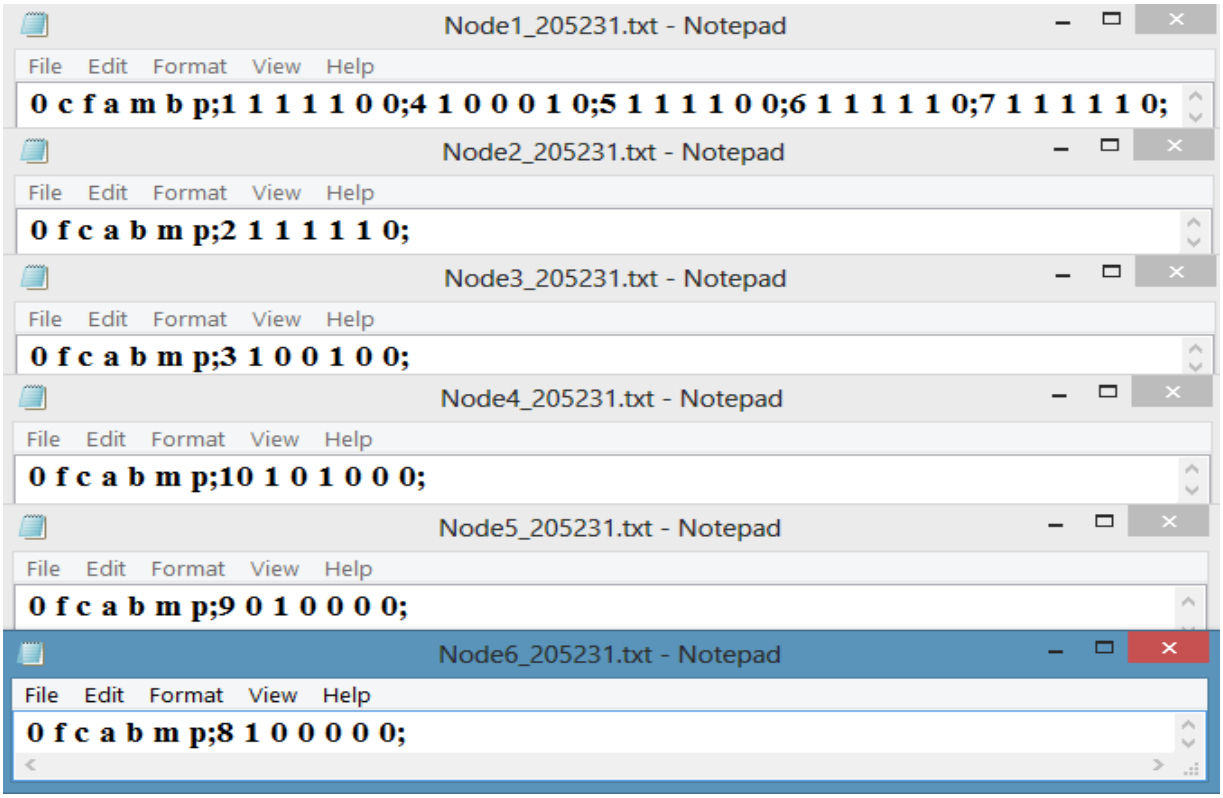
Hình 3.14: duyệt và sắp xếp danh sách

Hình 3.15 đã được xử lý và phân vùng thành tập các CSDL con cấp đầu tiên, trong đó các giao dịch 2, 3, 10, 9, 8 (các dòng hiển thị tô màu) đã được tối ưu và phổ biến trong FPL tương ứng với các node 2, 3, 4, 5, 6 và các giao dịch 1, 4, 5, 6, 7 là CSDL con đầu tiên hay node 1 như hình 3.16 và đây cũng là CSDL con cấp thứ 2 cần phải xử lý phân vùng tiếp theo.



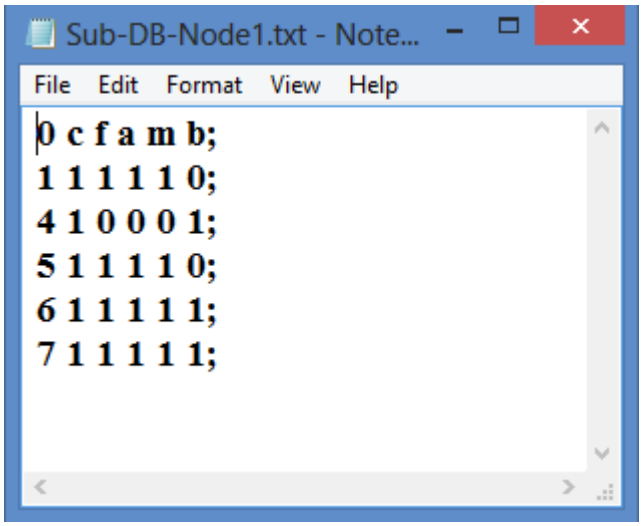
Giao dịch	f	c	a	b	m	p
1	1	1	1	0	1	1
4	0	1	0	1	0	1
5	1	1	1	0	1	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
2	1	1	1	1	1	0
3	1	0	0	1	0	0
10	1	0	1	0	0	0
9	0	1	0	0	0	0
8	1	0	0	0	0	0

Hình 3.15: phân vùng thành tập các CSDL con cấp đầu tiên (các node)



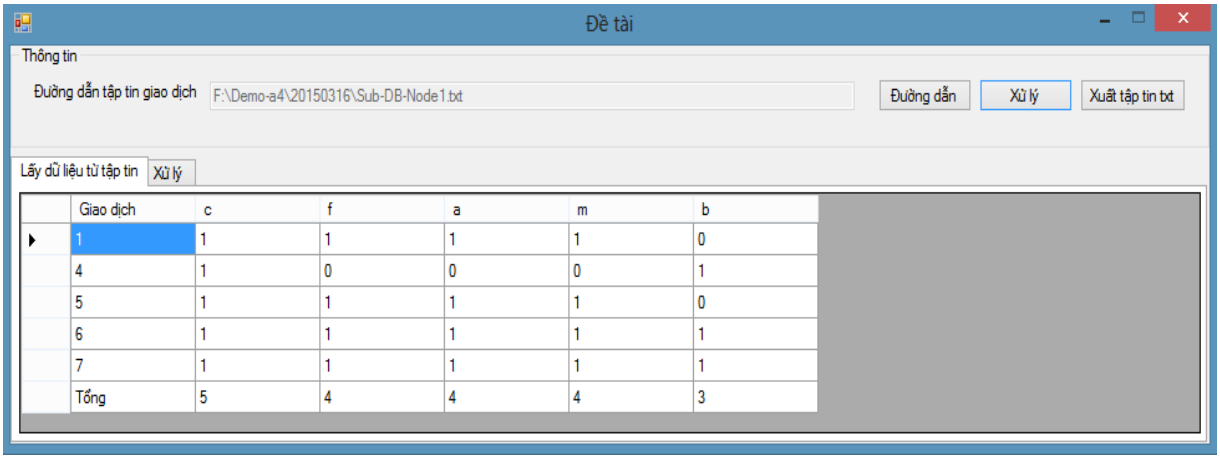
Hình 3.16: hiển thị các node sau khi phân vùng CSDL

Như vậy, sau khi đã loại bỏ các mục cuối cùng của node 1 (p) để có được một danh sách ngắn ngọn hơn. Chúng ta thấy rằng có tổng cộng 25 bit, vẫn còn cao so với yêu cầu dung lượng của bộ nhớ. Hình 3.17 (CSDL con cấp đầu tiên đã rút gọn)



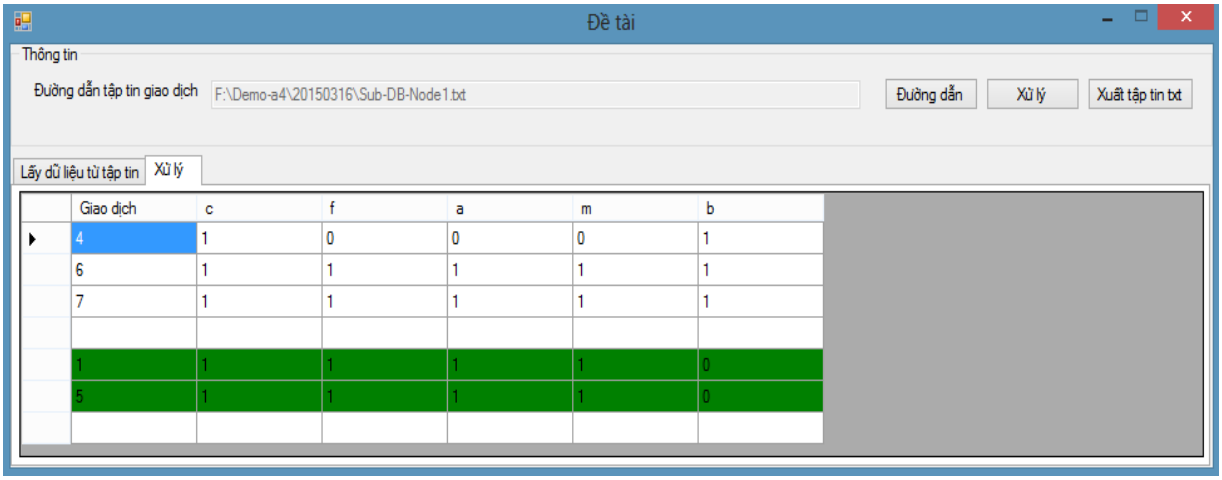
Hình 3.17: CSDL con cấp đầu tiên

Chương trình tiếp tục xử lý như bước ở trên, như hình.3.18



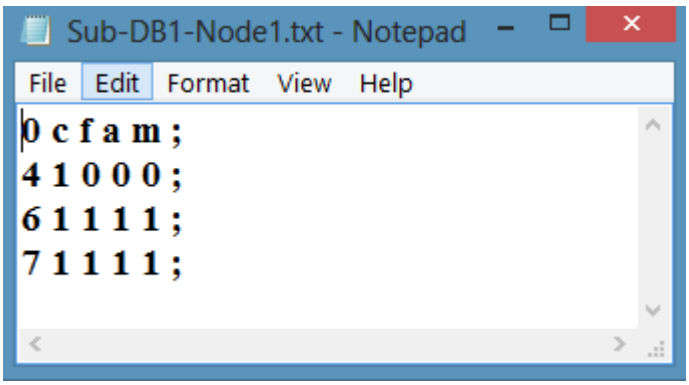
Hình 3.18: duyệt và sắp xếp danh sách CSDL cấp thứ nhất

Phân vùng dữ liệu con cấp thứ 2, hình 3.19



Hình 3.19: phân vùng thành tập các CSDL con cấp thứ 2 (các node)

Loại bỏ các mục cuối cùng của Sub-DB-node1(b) để có được một danh sách ngắn ngọn hơn



Hình 3.20: danh sách CSDL cấp thứ 2

Chúng ta thấy chỉ có 12 bit trong Sub-DB1- node1 tương ứng với các giao dịch 4, 6, 7 như hình 3.20, đã vừa dung lượng bộ nhớ là 24 bit. Do đó chúng ta không cần phải phân vùng thêm một lần nữa, và cũng là tìm tập phổ biến tại đây.

CHƯƠNG 4 KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TRONG TƯƠNG LAI

Nghiên cứu này giới thiệu một phương pháp tiếp cận hiệu quả là phương pháp phân vùng thứ bậc để khai thác tập phổ biến trong CSDL lớn. Phương pháp này dựa trên hai nguyên tắc. Nguyên tắc đầu tiên là tính chất phân vùng của Danh sách mẫu phổ biến (FPL), danh sách này phân vùng không gian tìm kiếm (cơ sở dữ liệu) và không gian giải pháp (tập hoàn chỉnh của các tập phổ biến hoặc FCIs). Vì vậy, một cách tiếp cận chia để trị có thể được áp dụng một cách có thứ bậc cho các CSDL để khai thác dữ liệu. Nguyên tắc thứ hai là tính đối ngẫu giữa các nút mục của FPL và CSDL con: một nút của FPL có thể được coi là một CSDL con thường trú trong bộ nhớ, và một CSDL con có thể được coi như là một nút mục của FPL thường trú trên đĩa. Vì vậy, các thao tác và các kỹ thuật tối ưu hóa cho FPL cũng tương tự và có thể được áp dụng cho CSDL con.

Một khi CSDL con có thể đặt vừa trong bộ nhớ thì bất kỳ thuật toán nào dựa trên hiệu quả của bộ nhớ đều có thể được sử dụng để khai thác các mẫu phổ biến. Do đó, phân vùng thứ bậc là một cách tiếp cận chung có thể được sử dụng với bất kỳ thuật toán dựa trên năng lực bộ nhớ nào để khai thác dữ liệu trong CSDL lớn. Ngoài ra, các tập phổ biến được tạo ra từ phân vùng thứ bậc cũng là tập phổ biến toàn cục. Do đó, không mất thêm chi phí để quét lại CSDL ban đầu nhằm kiểm tra tần số toàn cục. Kết quả thực nghiệm cho thấy phân vùng thứ bậc cải thiện hiệu suất đáng kể trong khai thác tập phổ biến và tập phổ biến đóng trong CSDL lớn.

Trong thời đại của thương mại điện tử và kinh doanh, kích thước của CSDL giao tác sẽ tăng lên một cách dễ dàng và nhanh chóng. Kết quả là, khả năng mở rộng của các thuật toán khai thác dữ liệu là một vấn đề có tầm quan trọng ngày càng tăng. Các thuật toán khai thác dựa trên phân vùng thứ bậc có thể được sử dụng để khai thác dữ liệu nhằm hiểu được hành vi của người tiêu dùng trực tuyến, bao gồm người mua

sắm trực tuyến và các game thủ trực tuyến. Đồng thời, vì CSDL luôn được cập nhật và tích lũy, nội dung của CSDL luôn thay đổi và kích thước của nó cũng luôn tăng.

Cho nên, các nghiên cứu trong tương lai có thể được dành cho sự phát triển của các thuật toán mở rộng nhằm tăng khả năng khai thác tập phổ biến trong CSDL lớn

Phân chia CSDL ra làm nhiều máy để xử lý, dùng phương pháp xử lý song song để giao nhận dữ liệu khi xử lý./.

TÀI LIỆU THAM KHẢO

TÀI LIỆU TIẾNG VIỆT

[1] Lê Hoài Bắc (2013), *Bài giảng môn Data Mining*, Đại học KHTN (Đại học Quốc gia Tp.HCM).

[2] Võ Đình Bảy (2013), *Bài giảng Luật Kết Hợp*, Đại học KHTN (Đại học Quốc gia Tp.HCM).

[3] Hồ Anh Tài (2004), *Ứng dụng kỹ thuật khai khoáng dữ liệu trong nghiệp vụ xử lý cước điện thoại tại bưu điện tỉnh Ninh Thuận*, Luận văn Thạc Sỹ, Đại Học KHTN TP. HCM, TP.HCM.

[4] Bùi Danh Hường (2010), *Ứng dụng khai mỏ trên cơ sở dữ liệu tai nạn giao thông*, Luận văn Thạc Sỹ, Đại Học KHTN TP. HCM, TP.HCM.

[5] Viktor Mayer – Schönberger, Kenneth Cukier; Vũ Duy Mẫn dịch, *"Dữ Liệu Lớn"*, NXB Trẻ 2014.

TÀI LIỆU TIẾNG ANH

[6] Han, J., Pei, J., & Yin, Y. (2000), *Mining frequent itemsets without candidate generation.* In: Proc. 2000 ACM SIGMOD int. conf. on management of data (SIGMOD'00), pp. 1–12

[7] Han, J., Kamber, M., & Pei, J. (2011), *Data mining: Concepts and techniques*(3rd ed.). San Francisco, CA: Morgan Kaufmann.

[8] Mohammad Karim Sohrabi, Ahmad Abdollahzadeh Barforoush (2012), *Efficient colossal pattern mining in high dimensional datasets*. Knowledge –Based Systems Vol.33, pp.41–52.

[9] Tseng, F.-C. (2012). *An adaptive approach to mining frequent itemsets efficiently*. Expert Systems with Applications Vol.39, pp.13166–13172.

[10] Tseng, F.-C., & Hsu, C.-C. (2001), *Generating frequent itemsets with the frequent pattern list*.Lecture Notes in Artificial Intelligence, LNAI Vol.2035, pp.376–386, Springer - Verlag

[11] Tseng, F.-C., Hsu, C.-C., & Fu, K.-S. (2005a), *The frequent pattern list: Another framework for mining frequent itemsets*. International Journal of Electronic Business Management Vol.3, 104–115.

[12] Jiawei Han and Micheline Kamber (2002), *DataMining:Concepts and Techniques*, University of Illinois, Morgan Kaufmann Publishers.

[13] Jiawei Han, Micheline Kamber & Jian Pei (2012), *DataMining:Concepts and Techniques*. 3rd ed., Morgan Kaufmann, USA, pp. 243-276

[14] Tseng,F.-C.(2013), *Mining frequent itemsets in large databases: The hierarchical partitioning approach*. Expert Systems with Applications Vol.40, Issue 5, pp.1654-1661.