

CHƯƠNG 2: TẬP PHỔ BIẾN

2.1 Tập phổ biến

2.1.1 Định nghĩa:

Định nghĩa độ phổ biến:

Cho CSDL bao gồm:

Tập các danh mục I , tập các giao dịch D và tập danh mục $X \subseteq I$

Độ phổ biến của X trong D - ký hiệu là $\text{sup}(X)$ - được định nghĩa là số giao dịch mà X xuất hiện trong D .

Ví dụ:

Với CSDL mẫu trong bảng 1.1, ta có:

Tập danh mục $I = \{ A, C, D, T, W \}$ và tập giao dịch D gồm có 6 giao dịch: $\{ACTW, CDW, ACTW, ACDW, ACDTW, CDT\}$

Độ phổ biến của tập danh mục $X_1 = \{ A \}$ là số giao dịch trong D có chứa $\{ A \}$, do đó $\text{sup}(X_1) = 4$

Tương tự, $X_2 = \{ A, T \} \Rightarrow$ độ phổ biến của X_2 là $\text{sup}(X_2) = 3$

Định nghĩa tập phổ biến:

Tập $X \subseteq I$ được gọi là tập phổ biến nếu $\text{sup}(X) \geq \text{minSup}$, với minSup là giá trị do người dùng chỉ định.

Ví dụ:

Cũng với CSDL mẫu trong bảng 1.1, với $\text{minSup} = 3$ (50%) thì tập $X_2 = \{ A, T \}$ là tập phổ biến vì có $\text{sup}(X_2) = 3 \geq \text{minSup}$. Tương tự, với $X_3 = \{ A, C, T \}$ thì $\text{sup}(X_3) = 3 \geq \text{minSup}$ và X_3 cũng là tập phổ biến. Ngược lại, với $X_4 = \{ C, D, T \}$ thì $\text{sup}(X_4) = 2 < \text{minSup}$, do đó X_4 không phải là tập phổ biến.

2.1.2 Các tính chất:

Tính chất 1:

Độ phổ biến của tập con lớn hơn tập cha.

Cho hai tập phổ biến X, Y với $X \subset Y$ thì $\text{sup}(X) \geq \text{sup}(Y)$

Tính chất 2 :

Mọi tập con của một tập phổ biến đều là tập phổ biến.

X là tập phổ biến và $Y \subset X$ thì $\text{sup}(Y) \geq \text{sup}(X) \geq \text{minSup}$, vì vậy Y cũng là tập phổ biến.

Tính chất 3 :

Mọi tập cha của một tập không phổ biến thì cũng không phổ biến.

X là tập không phổ biến và $Y \supset X$ thì $\text{sup}(Y) \leq \text{sup}(X) < \text{minSup}$, vì vậy Y cũng không phải là tập phổ biến.

2.1.3 Cách bố trí dữ liệu :

Trong các cơ sở dữ liệu quan hệ thông thường sẽ lưu trữ dữ liệu theo chiều ngang. Tức là các bảng dữ liệu hai chiều sẽ gồm N dòng tương ứng với các giao dịch, và M cột tương ứng với các danh mục. Việc bố trí theo chiều ngang giúp cho việc xác định các danh mục thuộc về một giao dịch đơn giản nhanh chóng. Tuy nhiên khi cần xác định một danh mục cụ thể thuộc vào những giao dịch nào thì cách bố trí theo chiều ngang lại gây ra khó khăn, khi đó ta phải duyệt tất cả các giao dịch có trong CSDL và ghi nhận những giao dịch có chứa danh mục cụ thể đó.

Ví dụ :

Trong CSDL mẫu ở bảng 1.1, muốn biết danh mục A nằm trong những giao dịch nào thì ta phải duyệt hết tất cả các danh mục xem giao dịch nào chứa A thì liệt kê ra, kết quả các giao dịch chứa A là tập $\{ 1, 3, 4, 5 \}$

Có thể nhận thấy với các CSDL lớn thì việc xác định những giao dịch có chứa một danh mục là rất tốn kém. Trong lúc đó việc xác định những giao dịch có chứa một danh mục cụ thể lại là công việc thường xuyên để tính độ phổ biến của một tập danh mục. Để tăng tốc độ khai thác tìm tập phổ biến, chúng ta có thể sử dụng cách bố trí dữ liệu theo chiều dọc. Nghĩa là bảng dữ liệu có sự đảo chiều, các dòng biến thành các cột và ngược lại :

Ví dụ : CSDL mẫu trong bảng 1.1 được bố trí theo chiều dọc như sau :

Bảng 2.1 Bảng dữ liệu bố trí theo chiều dọc

Mã danh mục	Mã các giao dịch					
C	1	2	3	4	5	6
W	1	2	3	4	5	
A	1	3	4	5		
D	2	4	5	6		
T	1	3	5	6		

Lúc này việc xác định xem danh mục A có độ phổ biến bao nhiêu trong CSDL bố trí theo chiều dọc trở nên đơn giản bằng cách xác định dòng trong bảng tương ứng với mã danh mục A, kết quả là tập giao dịch {1, 3, 4, 5} và $\text{sup}(A) = 4$

Tuy nhiên cách bố trí dữ liệu theo chiều dọc sẽ gây khó khăn trong việc quản lý bộ nhớ vì bảng dữ liệu gia tăng kích thước theo chiều ngang thay vì theo chiều dọc.

2.2 Tập phổ biến đóng

2.2.1 Kết nối Galois

Quan hệ hai ngôi:

Cho $I = \{ i_1, i_2, \dots, i_n \}$ là tập tất cả các danh mục. $T = \{ t_1, t_2, \dots, t_m \}$ là tập tất cả các giao dịch trong CSDL giao dịch D .

CSDL được cho là quan hệ hai ngôi $\delta \subseteq I \times T$. Nếu mục $i \in I$ xảy ra trong giao dịch $t \in T$ thì ta viết là: $(i, t) \in \delta$ kí hiệu $i \delta t$.

Ví dụ:

Xét CSDL mẫu trong bảng 1.1, với quan hệ hai ngôi:

- Giao dịch thứ nhất được biểu diễn là $\{ A\delta 1, C\delta 1, T\delta 1, W\delta 1 \}$
- Giao dịch thứ hai được biểu diễn là $\{ C\delta 2, D\delta 2, W\delta 2 \}$
- Giao dịch thứ ba được biểu diễn là $\{ A\delta 3, C\delta 3, T\delta 3, W\delta 3 \}$
- Giao dịch thứ tư được biểu diễn là $\{ A\delta 4, C\delta 4, D\delta 4, W\delta 4 \}$
- Giao dịch thứ năm được biểu diễn là $\{ A\delta 5, C\delta 5, D\delta 5, T\delta 5, W\delta 5 \}$
- Giao dịch cuối cùng được biểu diễn là $\{ C\delta 6, D\delta 6, T\delta 6 \}$

Bảng 2.2 Bảng minh họa quan hệ 2 ngôi

Mã giao dịch	Nội dung giao dịch
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

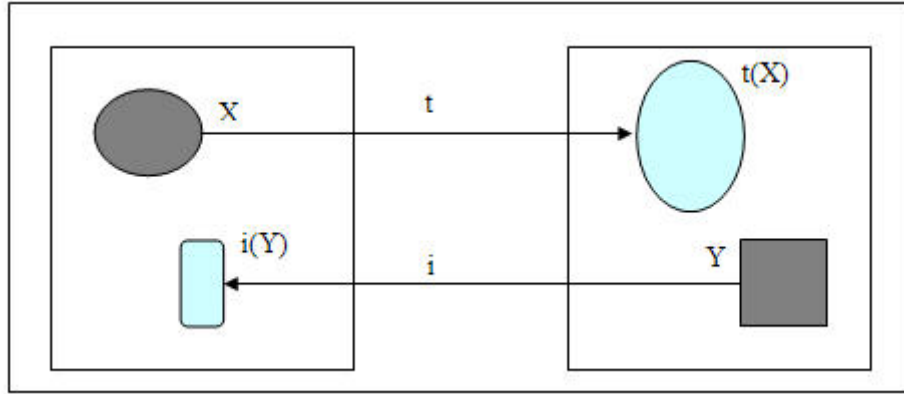
Mã danh mục	Các giao dịch có chứa danh mục
A	1, 3, 4, 5
C	1, 2, 3, 4, 5, 6
D	2, 4, 5, 6
T	1, 3, 5, 6
W	1, 2, 3, 4, 5

Định nghĩa kết nối Galois:

Cho quan hệ hai ngôi $\delta \subseteq I \times T$ chứa CSDL cần khai thác. Đặt $X \subseteq I$ và $Y \subseteq T$. Với $P(S)$ gồm tất cả các tập con của S . Ta định nghĩa hai ánh xạ giữa $P(I)$ và $P(T)$ được gọi là kết nối Galois như sau :

$$a. t : P(I) \rightarrow P(T), t(X) = \{ y \in T \mid \forall x \in X, x\delta y \}$$

$$b. i : P(T) \rightarrow P(I), i(Y) = \{ x \in I \mid \forall y \in Y, x\delta y \}$$



Hình 2.1 Kết nối Galois

Hai ánh xạ trên được minh họa trong hình 2.1, trong đó ánh xạ $t(X)$ là tập tất cả các giao dịch có chứa itemset X (hay còn gọi là **Tidset** của **Itemset** X) và $i(Y)$ là tập tất cả các danh mục có trong tất cả các giao dịch trong Y .

Ký hiệu itemset X và tập các giao dịch tương ứng với nó $t(X)$ là : $X \times t(X)$ và được gọi là **IT-pair**. Tương tự với tập giao dịch Y và $i(Y)$ là $i(Y) \times Y$.

Ví dụ :

$$t(ACW) = t(A) \cap t(C) \cap t(W) = 1345 \cap 123456 \cap 12345 = 1345$$

$$i(245) = i(2) \cap i(4) \cap i(5) = CDW \cap ACDW \cap ACDTW = CDW$$

2.2.2 Toán tử đóng và tập đóng:

Toán tử đóng:

Cho $X \subseteq I$ và ánh xạ $c: P(I) \rightarrow P(I)$ với $c(X) = i(t(X))$. Ánh xạ c được gọi là toán tử đóng.

Ví dụ:

Xét CSDL mẫu cho trong bảng 1.1 có:

$$c(AW) = i(t(AW)) = i(1345) = ACW$$

$$c(ACW) = i(t(ACW)) = i(1345) = ACW$$

Tập đóng:

Cho $X \subseteq I$. X được gọi là tập đóng khi và chỉ khi $c(X) = X$.

Ví dụ:

Xét CSDL mẫu cho trong bảng 1.1, ta có:

$$\text{Do: } c(AW) = i(t(AW)) = i(1345) = ACW \Rightarrow AW \text{ không phải là tập đóng}$$

$$\text{Do: } c(ACW) = i(t(ACW)) = i(1345) = ACW \Rightarrow ACW \text{ là tập đóng}$$

2.2.3 Định nghĩa tập phổ biến đóng:

X được gọi là tập phổ biến đóng nếu X phổ biến và X là tập đóng.

Ví dụ:

Xét CSDL mẫu cho trong bảng 1.1, với $\text{minSup} = 50\% = 50\% * 6 = 3$, ta có:

$$\text{Do } t(AW) = 1345 \Rightarrow \text{sup}(AW) = 4 > \text{minSup} \Rightarrow AW \text{ là tập phổ biến}$$

$$\text{Do } t(ACW) = 1345 \Rightarrow \text{sup}(ACW) = 4 > \text{minSup} \Rightarrow ACW \text{ là tập phổ biến}$$

Tuy nhiên chỉ có ACW là tập phổ biến đóng còn AW thì không phải.

Nói tóm lại, tập danh mục X là tập phổ biến đóng khi không tồn tại tập cha X' sao cho $X \subset X'$ và $\text{sup}(X) = \text{sup}(X')$.

Ý nghĩa của tập phổ biến đóng:

Trong những CSDL lớn, số lượng tập phổ biến đóng ít hơn rất nhiều so với số lượng tập phổ biến thông thường. Do đó để giảm thời gian trong công đoạn 1 trong khai thác luật kết hợp, chúng ta có thể đi tìm các tập phổ biến đóng thay cho các tập phổ biến, và thực hiện rút trích luật kết hợp trên các tập phổ biến đóng.

2.2.4 Tính chất của tập phổ biến đóng:

Tính chất 1 (item merging):

Giả sử X là một tập phổ biến và tất cả các giao dịch $Trans$ có chứa tập danh mục X , đồng thời mọi giao dịch này cũng chứa tập danh mục $Y \neq \emptyset$ với $Y \cap X = \emptyset$ và không tồn tại tập Y' tương tự như Y với $Y \subset Y'$ (có nghĩa là Y là tập lớn nhất có thể có). Thì có thể kết luận là tập $X \cup Y$ là một tập phổ biến đóng có $\text{sup}(X \cup Y) = |Trans|$; và những tập phổ biến chứa X mà không chứa Y thì không thể là tập phổ biến đóng.

Ví dụ:

Trong bảng 1.1 cho thấy những giao tác có chứa tập danh mục $\{W\}$ thì cũng chứa tập danh mục $\{C\}$ và cũng không tìm được tập danh mục Y sao cho $\{C\} \subset Y$ với những giao tác có chứa tập danh mục $\{CW\}$ thì cũng chứa tập danh mục Y . Vậy có thể kết luận là tập danh mục $\{CW\}$ là tập đóng. Ngoài ra cũng nhận thấy một điều: tập phổ biến $\{TW\}$ không là tập phổ biến đóng.

Tính chất 2 (sub-itemset pruning):

Tập danh mục X là tập phổ biến và tập danh mục Y với $Y \subset X$ và $\text{sup}(Y) = \text{sup}(X)$ thì có thể khẳng định là những tập phổ biến đóng có chứa Y thì chắc chắn sẽ chứa luôn X hoặc những tập chỉ chứa Y không chứa X thì không thể là tập phổ biến đóng.

Ví dụ:

Trong bảng 1.1 cho thấy tập danh mục $\{CTW\}$ là tập phổ biến với độ phổ biến là 3, và tập con $\{TW\}$ cũng có độ phổ biến là 3. Vậy tập phổ biến đóng $\{ACTW\}$ chứa tập danh mục $\{TW\}$ thì cũng chứa tập danh mục $\{CTW\}$ và các tập danh mục $\{ATW\}$, $\{DTW\}$ cũng không thể là tập phổ biến đóng.

2.3 Các phương pháp tìm tập phổ biến

2.3.1 Phương pháp sinh ứng viên – thuật toán Apriori:

Phương pháp sinh ứng viên để tìm tập phổ biến được Agrawal [13] đề xuất từ năm 1993 với thuật toán Apriori. Ý tưởng của thuật toán Apriori dựa trên kết luận: *nếu một tập danh mục là phổ biến thì tất cả tập con của nó cũng phải phổ biến (tính chất 2 – tập phổ biến)*. Do vậy không thể có trường hợp một tập phổ biến có tập con là không phổ biến hay nói cách khác tập phổ biến nhiều danh mục hơn chỉ có thể được tạo ra từ các tập phổ biến ít danh mục hơn. Và đó là cách hoạt động của thuật toán Apriori :

- Bắt đầu từ các tập phổ biến chỉ có một danh mục
- Tạo ra các tập phổ biến có k danh mục từ những tập phổ biến có (k-1) danh mục

Nội dung thuật toán :

L_k : Tập hợp của các tập phổ biến k danh mục (k-itemset) . Mỗi phần tử của tập hợp này có hai trường: tập danh mục và độ phổ biến.

C_k : Tập hợp của các tập ứng viên k danh mục. Mỗi phần tử của tập hợp này có hai trường: tập danh mục và độ phổ biến.

$L_1 = \{\text{tập hợp 1 danh mục}\};$

For ($k = 2; L_{k-1} \neq \emptyset; k++$) do begin

$C_k = \text{apriori-gen}(L_{k-1});$ // Tạo ứng viên mới.

Forall giao tác $t \in D$ do begin //duyệt CSDL

**$C_t = \text{subset}(C_k, t);$ //các tập danh mục ứng viên có trong
//giao tác t**

Forall ứng viên $c \in C_t$ do

$c.\text{count} ++;$

end

$L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$

end

$\text{Answer} = \bigcup_k L_k;$ // Trả về tập hợp của các tập phổ biến

Diễn giải thuật toán :

- Bước đầu tiên của thuật toán đơn giản chỉ tính các danh mục xuất hiện để xác định tập hợp của các tập phổ biến 1 danh mục.
- Lặp bước k :
 - + Tập hợp L_{k-1} được sử dụng để tạo nên tập ứng viên C_k : sử dụng hàm **apriori-gen** được miêu tả là hàm lấy L_{k-1} (tập hợp của các tập phổ biến k-1 danh mục) là đầu vào và trả về C_k là tập hợp của tất cả các tập chứa k danh mục phát sinh từ tập hợp L_{k-1} bằng cách hợp các tập k-1 danh mục trong tập hợp L_{k-1} . Chú ý một ứng viên thuộc C_k thì tất cả các tập con của ứng viên đó phải có mặt trong L_{k-1} (theo tính chất 2 của tập phổ biến)
 - + Bước kế tiếp, duyệt CSDL để tính độ phổ biến của các ứng viên trong tập hợp C_k . Từ đó, tính được tập hợp L_k .
 - + Nếu $L_k = \emptyset$ thì dừng lại.
- Hợp của các L_k chính là các tập phổ biến cần tìm.

Ví dụ minh họa : Xét CSDL mẫu trong bảng 1.1 với $\text{minSup} = 50\% = 50\% * 6 = 3$

Bước 1 : Duyệt CSDL để tìm ra L_1 là các tập phổ biến có 1 danh mục có độ phổ biến ≥ 3 :

Bảng 2.3 Các tập phổ biến có 1 danh mục

Database (D)				L_1	
TID	Nội dung			Danh mục	Độ phổ biến
1	A, C, T, W			A	4
2	C, D, W			C	6
3	A, C, T, W			D	4
4	A, C, D, W			T	4
5	A, C, D, T, W			W	5
6	C, D, T				

Bước 2 : tính tập ứng viên C_2 dựa trên L_1 bằng phép hợp. Sau đó duyệt CSDL tính độ phổ biến của các ứng viên thuộc C_2 , loại bỏ những ứng viên có độ phổ biến bé hơn minSup , để tạo thành L_2 là các tập phổ biến có 2 danh mục.

Bảng 2.4 Các tập phổ biến có 2 danh mục

C_2				L_2	
Danh mục	Độ phổ biến			Danh mục	Độ phổ biến
AC	4			AC	4
<u>AD</u>	<u>2</u>			AT	3
AT	3			AW	4
AW	4			CD	4
CD	4			CT	4
CT	4			CW	5
CW	5			<u>CT</u>	<u>2</u>
<u>CT</u>	<u>2</u>			DW	3
DW	3			TW	3
TW	3				

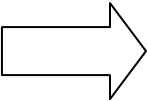
Bước 3 : Tương tự bước 2 tạo C_3 và L_3 từ L_2 . Chú ý khi tạo C_3 thì loại bỏ những ứng viên có tập con 2 danh mục không nằm trong L_2 . Từ L_2 để tạo C_3 có thể bằng cách từ tập hợp tất cả danh mục có trong L_2 , rồi lần lượt kiểm tra các tập danh mục 3 thuộc tính trong tập hợp đó theo chú ý ở dòng trên.

Bảng 2.5 Các tập phổ biến có 3 danh mục

C_3				L_3	
Danh mục	Độ phổ biến			Danh mục	Độ phổ biến
ACT	3			ACT	3
ACW	4			ACW	4
ATW	3			ATW	3
CDW	3			CDW	3
CTW	3			CTW	3

Bước 4 : Tạo C_4 và L_4 từ L_3

Bảng 2.6 Các tập phổ biến có 4 danh mục

C_4			L_4	
Danh mục	Độ phổ biến		Danh mục	Độ phổ biến
ACTW	3		ACTW	3

Bước 5 : Tạo C_5 và L_5 từ L_4 . Ta có $C_5 = \emptyset$.

Như vậy thuật toán dừng lại ở bước 5 vì $L_5 = \emptyset$

Kết quả : với $\text{minSup} = 50\%$ thì :

$$\begin{aligned} \text{Tập hợp các tập phổ biến} &= L_1 \cup L_2 \cup L_3 \cup L_4 \\ &= \{A, C, D, T, W, AC, AT, AW, CD, CT, CW, DW, TW, ACT, \\ &\quad ACW, ATW, CDW, CTW, ACTW\} \end{aligned}$$

Hạn chế của thuật toán Apriori:

Để xác định độ phổ biến của các tập ứng viên, thuật toán Apriori phải quét lại toàn bộ giao dịch trong CSDL, do đó sẽ tiêu tốn rất nhiều thời gian, đặc biệt khi số danh mục lớn.

Trong bài báo [12] các tác giả đề xuất thuật toán Apriori –Tid với sự khác biệt cơ bản là không sử dụng lại toàn bộ CSDL của lần duyệt thứ nhất mà kể từ bước thứ hai thuật toán apriori-Tid sử dụng tập C'_k . Việc tính độ phổ biến sẽ dựa vào các tập C'_k và như vậy tránh được việc phải đọc nhiều lần CSDL. Tuy nhiên trong quá trình xét duyệt khởi tạo kích thước của C'_k là rất lớn và hầu hết là tương đương với kích thước của CSDL gốc. Do đó thời gian tiêu tốn cũng sẽ tương đương với thuật toán Apriori, ngoài ra thuật toán Apriori- Tid còn phải gánh chịu thêm chi phí phát sinh nếu C'_k vượt quá bộ nhớ mà phải sử dụng kèm bộ nhớ ngoài.

Như vậy, nhìn chung các phương pháp sinh ứng viên để tìm tập phổ biến đều không hiệu quả vì phải đọc CSDL nhiều lần và phải phát sinh và kiểm tra một lượng lớn các ứng viên. Những hạn chế này sẽ được khắc phục với các phương pháp không sinh ứng viên sẽ tìm hiểu ở phần tiếp theo.

2.3.2 Phương pháp dựa trên cây FP-Tree

2.3.2.1 Cấu trúc cây FP – Tree

Cấu trúc FP-Tree (Frequent Pattern tree) được giới thiệu lần đầu tiên bởi các tác giả J.Han, J.Pei và Y.Yin trong bài báo [5] đã khắc phục được nhược 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 chung 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 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 danh mục.

2.3.2.2 Xây dựng cây FP – Tree

Thuật toán tạo cây FP-tree:

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 lớn 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 là “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 tử 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 ($p.item-name = child.item-name$) thì tăng chỉ số đếm của 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 thể hiện qua ví dụ xây dựng cây tương ứng với CSDL mẫu ở bảng 1.1 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 2.7 Mảng thứ tự các 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 2.6. Ta có CSDL sau khi sắp xếp như sau:

Bảng 2.8 CSDL sau khi sắp xếp theo thứ tự trong f-list

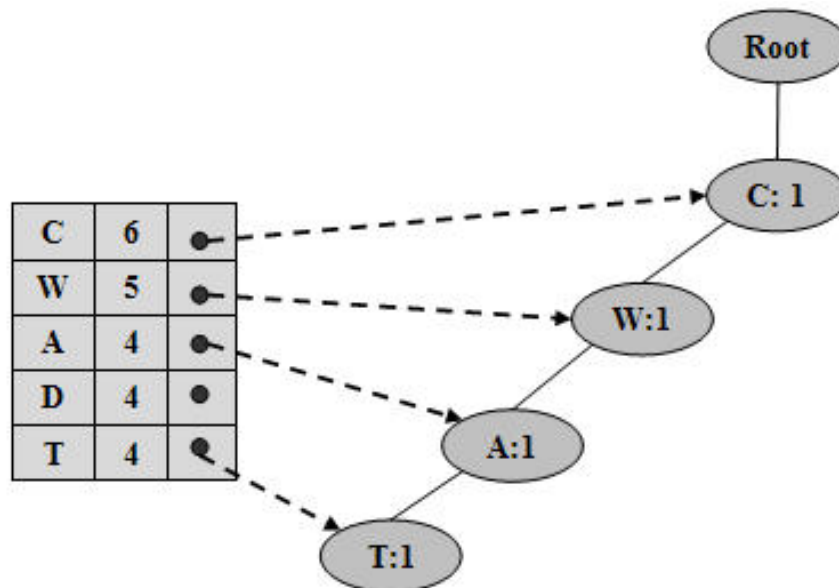
Mã giao dịch	Nội dung giao dịch	Nội dung giao dịch sau khi sắp xếp theo thứ tự mới
1	A, C, T, W	C, W, A, T
2	C, D, W	C, W, D
3	A, C, T, W	C, W, A, T
4	A, C, D, W	C, W, A, D
5	A, C, D, T, W	C, W, A, D, T
6	C, D, T	C, D, T

Cây FP-tree khi mới khởi tạo:



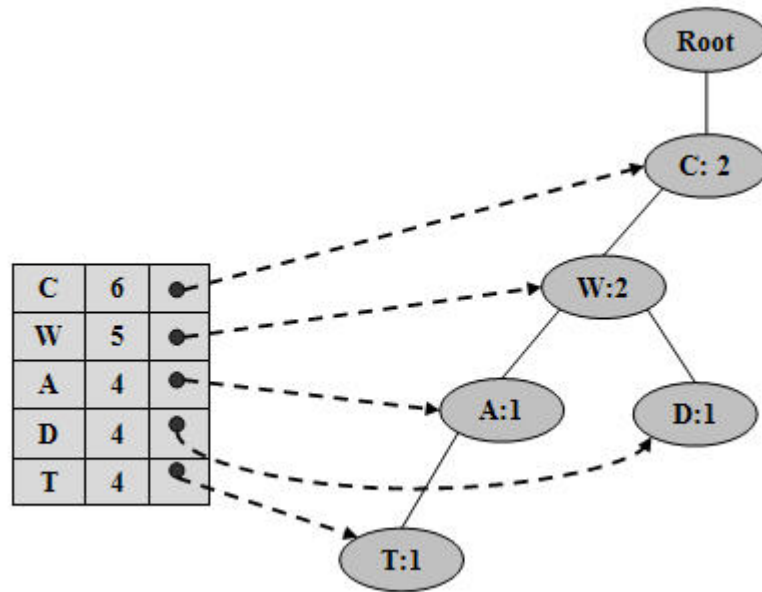
Hình 2.2 Cây FP-tree mới khởi tạo

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



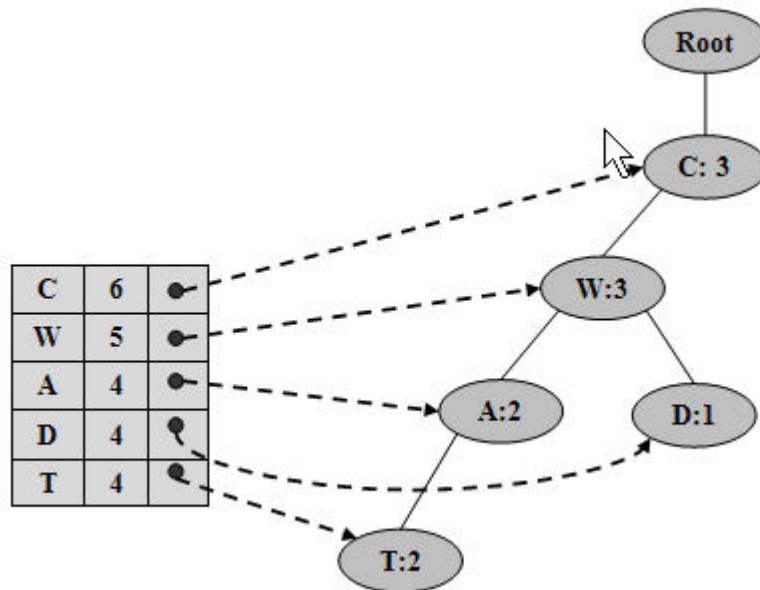
Hình 2.3 Cây FP-tree sau khi đọc xong giao dịch CWAT

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



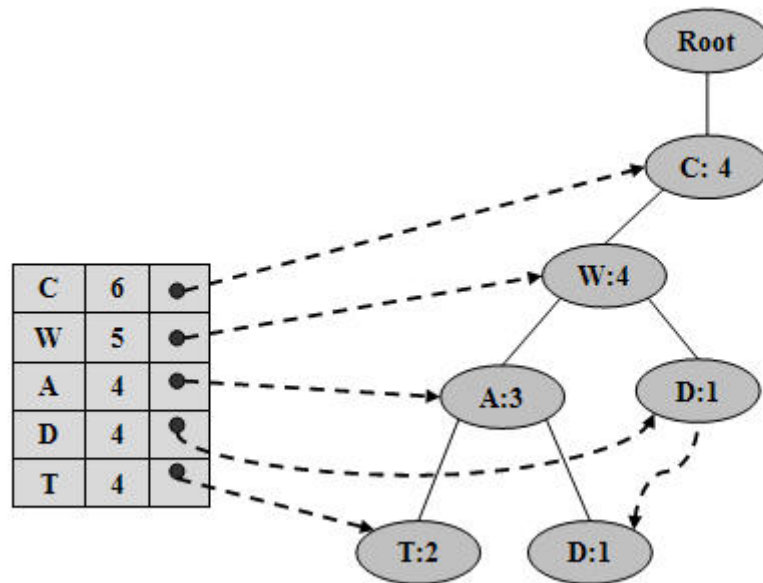
Hình 2.4 Cây FP-tree sau khi đọc giao dịch CWD

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



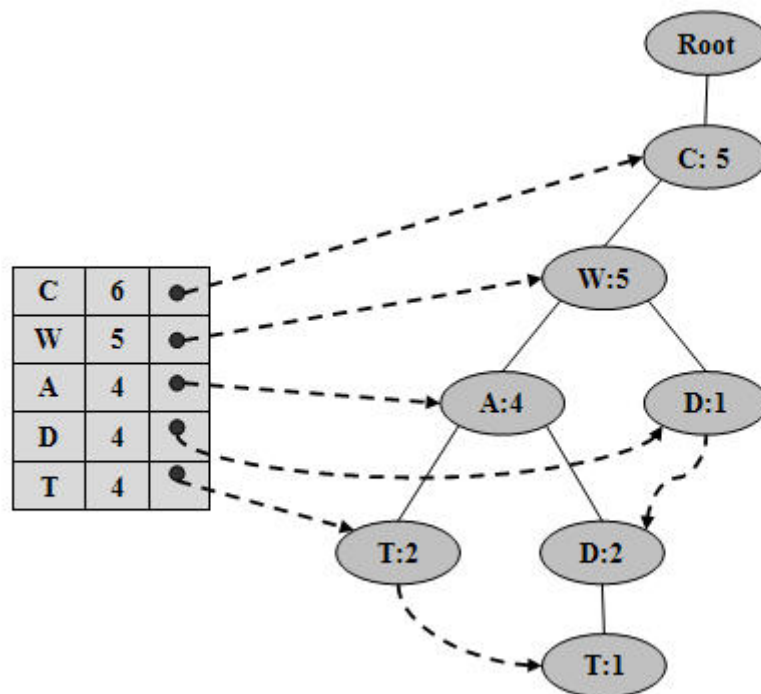
Hình 2.5 Cây FP-tree sau khi đọc giao dịch CWAT

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



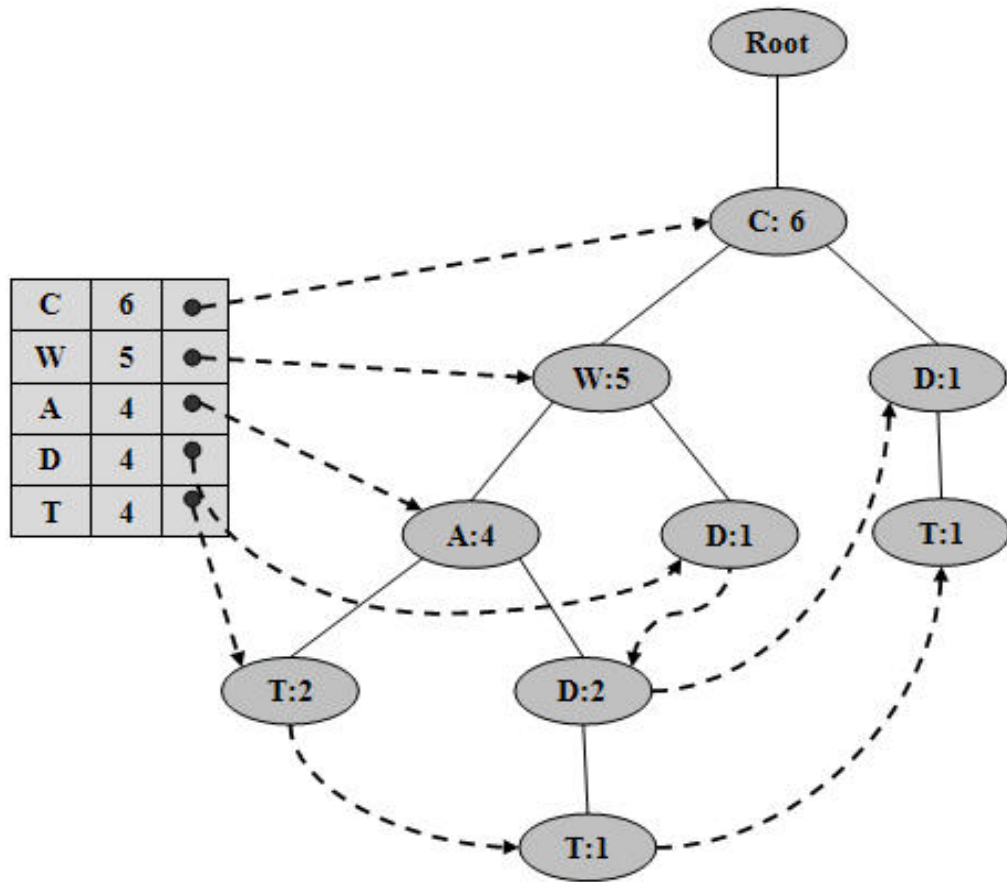
Hình 2.6 Cây FP-tree sau khi đọc giao dịch CWAD

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



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 $\text{minSup} = 2$:



Hình 2.8 Cây FP-tree toàn cục

2.3.2.3 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 nhiều vào 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 hai 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 của 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

2.3.2.4 Tìm các tập phổ biến với thuật toán FP-growth:

Trong bài báo [5] 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 đề 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** $_{\{prefix \cup a\}}$

Bước 2.2: $Tree_{\{prefix \cup a\}} = createFPtree(CSDL_{\{prefix \cup a\}}, minSup)$

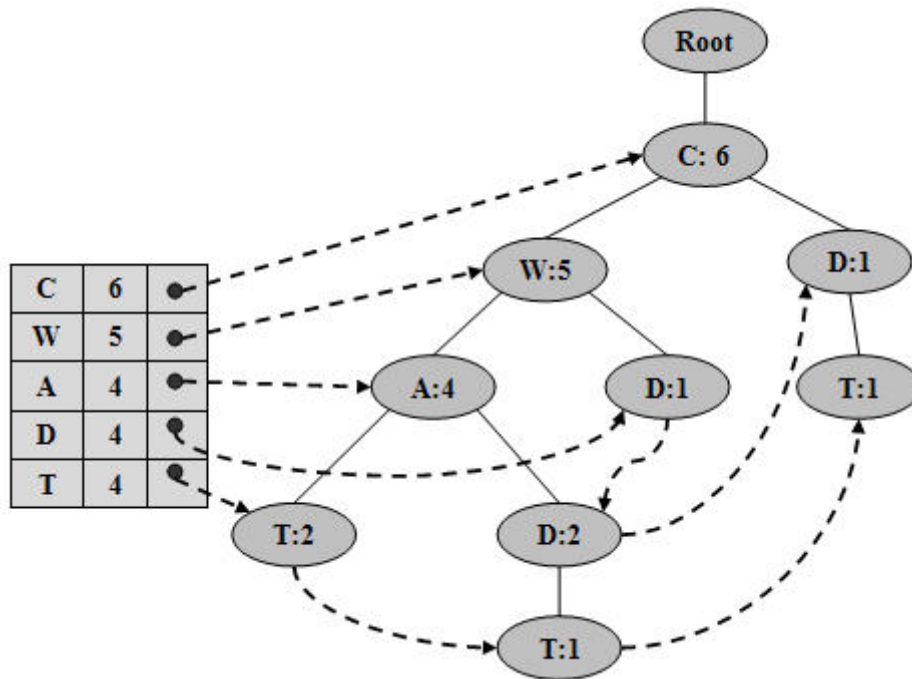
Bước 2.3: Nếu $Tree_{\{prefix \cup a\}} \neq \emptyset$ thì gọi hàm:

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 1.1 với ngưỡng phổ biến là $\text{minSup} = 2$. 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 ra $\text{CSDL}_{\{T\}}$:

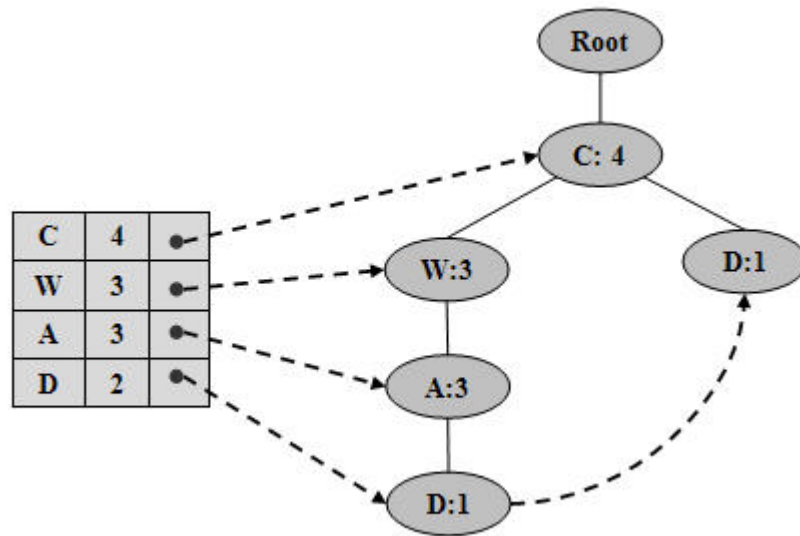
Chiếu trên tập danh mục (T:4)

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

Bảng 2.9 Nội dung $\text{CSDL}_{\{T\}}$

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

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

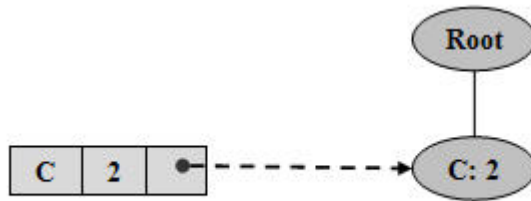
Chiều trên tập danh mục ($TD: 2$) :

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

Bảng 2.10: Nội dung $CSDL_{\{TD\}}$

Nội dung còn lại của các 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 $Tree_{\{TD\}} \neq \emptyset$ nên gọi đệ quy hàm $FP\text{-}growth (Tree_{\{TD\}}, (TD), \text{minSup})$

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 ($TA : 3$).

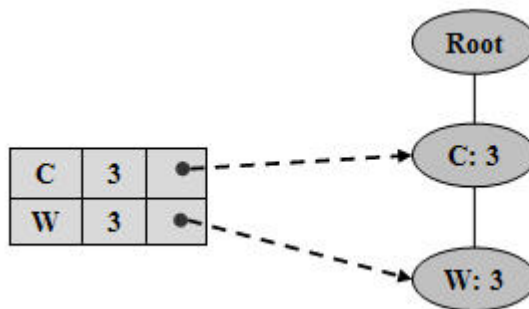
Chiếu trên tập danh mục (TA : 3) :

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

Bảng 2.11: 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 với $CSDL_{\{TA\}}$

Gọi hàm đệ quy $FP\text{-}growth (Tree_{\{TA\}}, (TA), \text{minSup})$ 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).

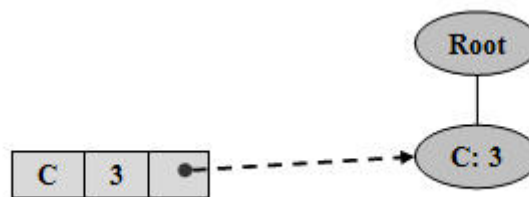
Chiếu trên tập danh mục (TW :3) :

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

Bảng 2.12: 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 $FP\text{-}growth (Tree_{\{TW\}}, (TW), minSup)$ 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) :

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

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)

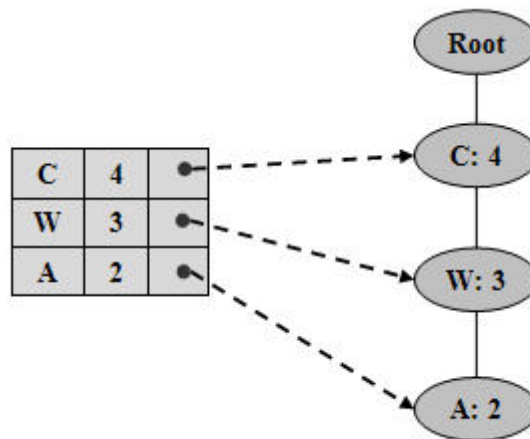
Chiều trên tập danh mục ($D:4$):

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

Bảng 2.13: 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 $FP\text{-}growth (Tree_{\{D\}}, (D), minSup)$ 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 $Tree_0$ và phát sinh tập phổ biến $(A: 4)$

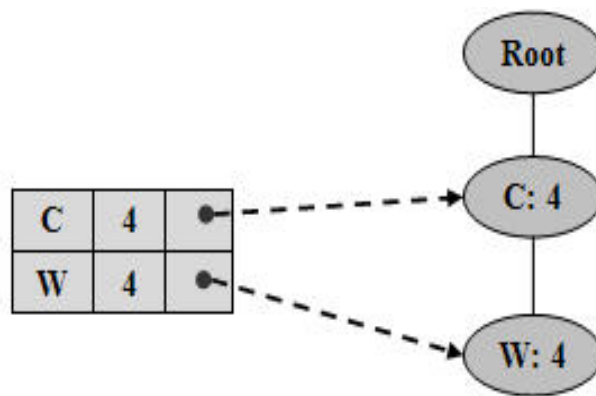
Chiều trên tập danh mục (A :4) :

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

Bảng 2.14: 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 $Tree_0$ và phát sinh tập phổ biến (W :5)

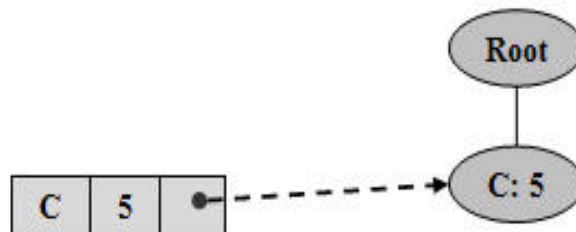
Chiều trên tập danh mục (W :5) :

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

Bảng 2.15: 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 hàm đệ quy $FP\text{-}growth (Tree_{\{W\}}, (W), minSup)$ và phát sinh được tập phổ biến : (WC : 5)

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

Chiều trên tập danh mục (C :6) :

Cơ sở dữ liệu cục bộ của các giao dịch có chứa tập danh mục { C } : $CSDL_{\{C\}} = \emptyset$

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

Thuật toán kết thúc

Kết quả tập phổ biến thu được khi thực hiện thuật toán FP-growth :

Bảng 2.16 : Kết quả tập phổ biến thỏa ngưỡng minSup = 2

STT	Tập phổ biến	Độ phổ biến (số giao dịch)	Độ phổ biến (%)
1.	T	4	66.67%
2.	T, D	2	33.33%
3.	T, D, C	2	33.33%
4.	T, A	3	50.00%
5.	T, A, W	3	50.00%
6.	T, A, C	3	50.00%
7.	T, A, W, C	3	50.00%
8.	T, W	3	50.00%
9.	T, W, C	3	50.00%
10.	T, C	4	66.67%
11.	D	4	66.67%
12.	D, A	2	33.33%
13.	D, W	3	50.00%
14.	D, W, A	2	33.33%
15.	D, C	4	66.67%
16.	D, A, C	2	33.33%
17.	D, W, C	3	50.00%
18.	D, A, W, C	2	33.33%
19.	A	4	66.67%
20.	A, W	4	66.67%
21.	A, C	4	66.67%
22.	A, C, W	4	66.67%
23.	W	5	83.33%
24.	W, C	5	83.33%
25.	C	6	100.00%

Nhận xét :

Thuật toán FP-growth khá hiệu quả vì sử dụng cấu trúc cây FP-tree và duyệt theo chiều sâu với mô hình chia để trị. Tuy nhiên vì FP-growth làm việc trên tất cả các tập phổ biến nên vẫn chậm hơn một số thuật toán khác sử dụng cây FP-tree nhưng chỉ làm việc trên các tập phổ biến đóng như CLOSET và CLOSET+.

2.3.2.5 Tìm các tập phổ biến đóng với thuật toán CLOSET+

Trong các bài báo [6], [4] các tác giả đã dựa trên ý tưởng là khai thác luật kết hợp thông qua các tập phổ biến đóng và trình bày các thuật toán CLOSET và CLOSET+ để tìm tập phổ biến đóng. Số lượng tập phổ biến đóng ít hơn rất nhiều so với số lượng các tập phổ biến thông thường, bên cạnh đó các tập phổ biến đóng vẫn chứa đựng tất cả các luật kết hợp có trong CSDL, do đó việc khai thác luật kết hợp dựa trên các tập phổ biến đóng là rất hiệu quả. Sau đây chúng ta tìm hiểu về thuật toán CLOSET+ dùng để tìm ra các tập phổ biến đóng trên cây FP-tree.

Dựa vào tính chất của tập phổ biến đóng có một số nhận xét sau:

Nhận xét 1: Nếu một danh mục phổ biến đều có xuất hiện trong nhiều cấp f-list cục bộ với cùng độ phổ biến thì ta có thể loại bỏ và không xét đến danh mục này trong những cấp f-list cục bộ trước đó.

Nhận xét 2: Trong quá trình phát sinh tập phổ biến đóng, chúng ta đảm bảo tính đóng của tập phổ biến bằng 2 phép kiểm tra:

- + **Superset checking:** Kiểm tra trong các tập phổ biến đóng cũ nếu có tập nào là tập con của tập phổ biến mới và có cùng độ phổ biến thì loại bỏ tập đó.
- + **Subset checking:** Kiểm tra trong các tập phổ biến đóng cũ nếu có tập nào là tập cha của tập phổ biến mới và có cùng độ phổ biến thì tập phổ biến mới không phải là tập phổ biến đóng.

Thuật toán CLOSET + sử dụng chiến lược duyệt theo chiều sâu và mô hình chia để trị cùng với việc áp dụng tính chất của tập phổ biến đóng nên không cần phép **Superset checking** vẫn đảm bảo tính đóng của các tập phổ biến tìm thấy.

Thuật toán CLOSET+**Hàm CLOSET+ ()****INPUT:** CSDL các giao dịch và ngưỡng phổ biến \minSup **OUTPUT:** Tập các tập phổ biến đóng FCI thỏa ngưỡng phổ biến \minSup **Các bước thực hiện:****Bước 1:** Duyệt CSDL và xây dựng cây FPtree: $Tree_0$ *Chú ý khi xây dựng cây $Tree_0$ tính số tích lũy trung bình của một nốt. Dựa vào đó để đánh giá xem CSDL có tỷ lệ nén cao hay thấp.***Bước 2:** Nếu $Tree_0$ có tỉ lệ nén cao:

- + Sử dụng phép chiếu từ dưới lên để khai thác tập phổ biến
- + Trong quá trình khai thác tập phổ biến sử dụng tính chất 1 và tính chất 2 của tập phổ biến đóng để giới hạn không gian khai thác.
- + Gọi thủ tục **Checking_BottomUp** mỗi lần phát sinh một ứng viên tập phổ biến đóng để kiểm tra tính đóng của kết quả.

Bước 3: Nếu $Tree_0$ có tỉ lệ nén thấp:

- + Sử dụng phép chiếu từ trên xuống để khai thác tập phổ biến
- + Trong quá trình khai thác tập phổ biến sử dụng tính chất 1 và tính chất 2 của tập phổ biến đóng để giới hạn không gian khai thác.
- + Gọi thủ tục **Checking_TopDown** mỗi lần phát sinh một ứng viên tập phổ biến đóng để kiểm tra tính đóng của kết quả.

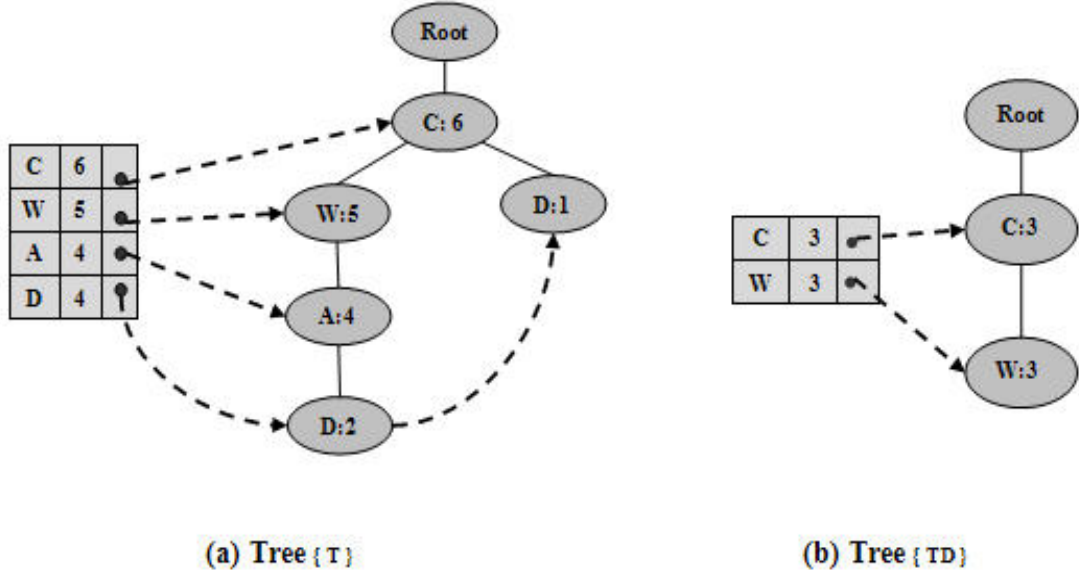
Giải thích thuật toán:

Với CSDL có tỉ lệ nén cao, FPtree toàn cục có thể được nén lại nhỏ hơn nhiều lần so với CSDL gốc. Tương tự, các phép chiếu trên cây FPtree cũng tạo thành các FPtree cục bộ nhỏ hơn nhiều so với chính nó. Vì vậy quá trình khai thác tập phổ biến khá hiệu quả. Bên cạnh đó thuật toán sử dụng thêm các tính chất 1 (item merging) và tính chất 2 (sub-itemset pruning) để giảm bớt không gian tìm kiếm.

Ví dụ:

Với CSDL mẫu trong bảng 1.1 và $\minSup = 2$.

Hình 2.21(a) là $Tree_{\{T\}}$ sau khi chiếu FPtree toàn cục lên tập danh mục $\{T\}$



Hình 2.17: Áp dụng tính chất của tập phổ biến đóng

Khi thực hiện phép chiếu từ dưới lên với tập danh mục $\{TA: 3\}$ ta có $Tree_{\{TD\}}$ như hình 2.21(b) và $CSDL_{\{TA\}} = (CW : 3)$.

Vì mọi giao dịch chứa tập phổ biến $\{TA:3\}$ cũng chứa $\{CW:3\}$ và $\{CW\}$ lớn nhất có thể nên theo tính chất **item merging** thì:

- $\{TA\}$ và $\{CW\}$ hợp thành tập phổ biến đóng $\{TACW : 3\}$.
- Không phải xét tới các tập phổ biến $\{TAC\}$ và $\{TAW\}$ vì chúng chắc chắn không phải là tập phổ biến đóng.

Tiếp đó, khi xét đến tập danh mục $\{TW : 3\}$ thì theo tính chất **sub-itemset pruning** thì do $\{TW\}$ là tập con của tập phổ biến đóng $\{TACW\}$ vừa tìm thấy ở trên và lại có cùng độ phổ biến là 3, do đó không cần thực hiện phép chiếu với $\{TW\}$

Có thể nhận thấy, khi CSDL có tỷ lệ nén cao thì nhiều khả năng xuất hiện các FPtree cục bộ là đường đơn, và do đó tính chất item merging sẽ có cơ hội áp dụng nhiều lần để làm giảm không gian tìm kiếm.

Sau khi tìm được một ứng viên tập phổ biến đóng thì ta sử dụng thủ tục **Checking_BottomUp** hoặc **Checking_TopDown** để kiểm tra tính đóng của ứng viên đó.

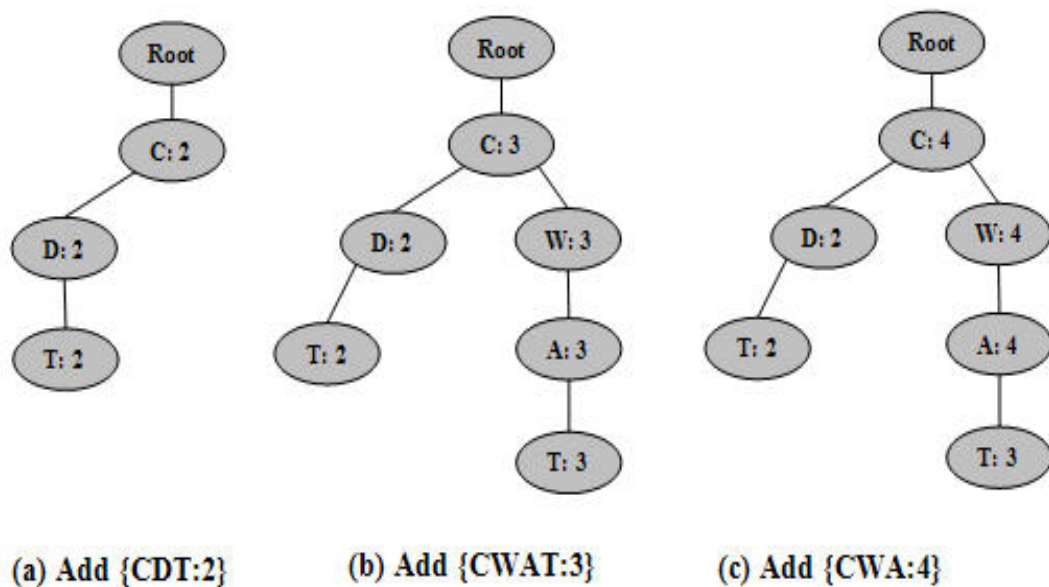
Thủ tục **Checking_BottomUp** làm việc dựa trên cây kết quả:

Cây kết quả là cấu trúc dữ liệu dùng để lưu trữ các tập phổ biến đóng, nó bao gồm 2 cấp chỉ mục:

- Một là tên danh mục theo thứ tự trong f-list
- Hai là độ phổ biến của danh mục, căn cứ trên độ phổ biến lớn nhất của danh mục đó trong các tập phổ biến đóng tham gia vào cây.

Khi bổ sung tập phổ biến đóng mới vào cây, duyệt từ gốc nếu nút đã có thì lấy giá trị lớn nhất thay vì tính tích lũy, nếu nút chưa có thì tạo nút mới với độ phổ biến bằng độ phổ biến của tập đóng thêm vào cây.

Ví dụ: Từ các tập phổ biến đóng { CDT:2}, {CWAT:3}, {CWA:4} ta xây dựng cây kết quả như sau theo thứ tự trong f-list là {CWADT} :



Hình 2.18: Minh họa xây dựng cây kết quả

Để kiểm tra tính đóng của tập phổ biến $\{CWT:3\}$ ta tìm vị trí xuất hiện trên cây của phần tử cuối theo thứ tự f-list là nút $\{T:3\}$, sau đó duyệt dần về gốc kiểm tra thấy tất cả các phần tử của $\{CWT:3\}$ đều xuất hiện hết trong cây, do đó $\{CWT:3\}$ không phải là tập phổ biến đóng.

Hàm Checking_BottomUp()

INPUT: Cây kết quả RTree lưu trữ tập phổ biến đóng, tập phổ biến X

OUTPUT: Kết luận tập X có phải là tập phổ biến đóng hay không

Các bước thực hiện:

Bước 1: Chọn phần tử cuối cùng trong tập danh mục của X , xác định vị trí xuất hiện của nó trên cây có giá trị trùng $\text{sup}(X)$.

Bước 2: Xác định nút trên cây và duyệt dần về gốc để kiểm tra xem tất cả các danh mục trong X có lần lượt xuất hiện hết trong cây hay không?

Bước 3: Nếu không hết, kết luận X là tập phổ biến đóng và phải bổ sung vào cây RTree. Ngược lại thì không phải là tập phổ biến đóng.

Khi CSDL có tỷ lệ nén thấp, chúng ta sử dụng thủ tục Checking_TopDown để kiểm tra một tập phổ biến có phải là tập phổ biến đóng hay không dựa trên cây FP-tree toàn cục.

Hàm Checking_BottomUp()

INPUT: Cây FP-Tree toàn cục, tập phổ biến X

OUTPUT: Kết luận tập X có phải là tập phổ biến đóng hay không

Các bước thực hiện:

Bước 1: max = thứ tự lớn nhất trong số các danh mục trong X theo f-list

Bước 2: Duyệt lần lượt các giao dịch T_i có chứa tập phổ biến X dựa trên cây FP-tree và ghi nhận những danh mục có thứ tự nhỏ hơn max theo f-list thì tích lũy độ phổ biến của danh mục đó.

Bước 3: Nếu có danh mục nào ở bước 2 có độ phổ biến bằng $\text{sup}(X)$ thì X không phải tập phổ biến đóng. Ngược lại X là tập đóng.

2.3.3 Phương pháp dựa trên cây IT-Tree

2.3.3.1 Cấu trúc IT-tree

Cho $X \subseteq I$, ta định nghĩa hàm $P(X, k) = X[1:k]$ gồm k phần tử đầu của X và quan hệ tương đương dựa vào tiền tố như sau :

$$\forall X, Y \in P(I), X \equiv_{\theta_k} Y \Leftrightarrow p(X, k) = p(Y, k)$$

Mỗi nút trên IT-tree gồm hai thành phần:

- Itemset X
- Tidset $t(X)$: là tập các giao dịch có chứa X

Cặp $X \times t(X)$ được gọi là **IT-pair**.

Các tính chất của IT-pair:

Cho $X_i \times t(X_i)$ và $X_j \times t(X_j)$ là hai **IT-pair**. Ta có 4 tính chất sau

1. Nếu $t(X_i) = t(X_j)$ thì $c(X_i) = c(X_j) = c(X_i \cup X_j)$
2. Nếu $t(X_i) \subset t(X_j)$ thì $c(X_i) \neq c(X_j)$ nhưng $c(X_i) = c(X_i \cup X_j)$
3. Nếu $t(X_i) \supset t(X_j)$ thì $c(X_i) \neq c(X_j)$ nhưng $c(X_j) = c(X_i \cup X_j)$

$$4. \text{ Nếu } \begin{cases} t(X_i) \not\subseteq t(X_j) \\ t(X_i) \not\supseteq t(X_j) \end{cases} \quad \text{Thì } c(X_i) \neq c(X_j) \neq c(X_i \cup X_j)$$

2.3.3.2 Xây dựng cây IT-tree

Xét CSDL mẫu ở bảng 2.18, với minSup =3 (50%)

Bảng 2.17 Bảng CSDL minh họa xây dựng IT-tree

Mã giao dịch	Nội dung giao dịch	Mã danh mục	Các giao dịch có chứa danh mục
1	A, C, T, W	A	1, 3, 4, 5
2	C, D, W	C	1, 2, 3, 4, 5, 6
3	A, C, T, W	D	2, 4, 5, 6
4	A, C, D, W	T	1, 3, 5, 6
5	A, C, D, T, W	W	1, 2, 3, 4, 5
6	C, D, T		

Ta xây dựng cây IT-tree như sau :

Nút gốc Root là nút rỗng

Lớp 1 là các nút con có itemset chỉ là 1 danh mục dựa trên các danh mục phổ biến thỏa minSup.

$A \times t(A) = A \times 1345$; $C \times t(C) = C \times 123456$; $D \times t(D) = D \times 2456$

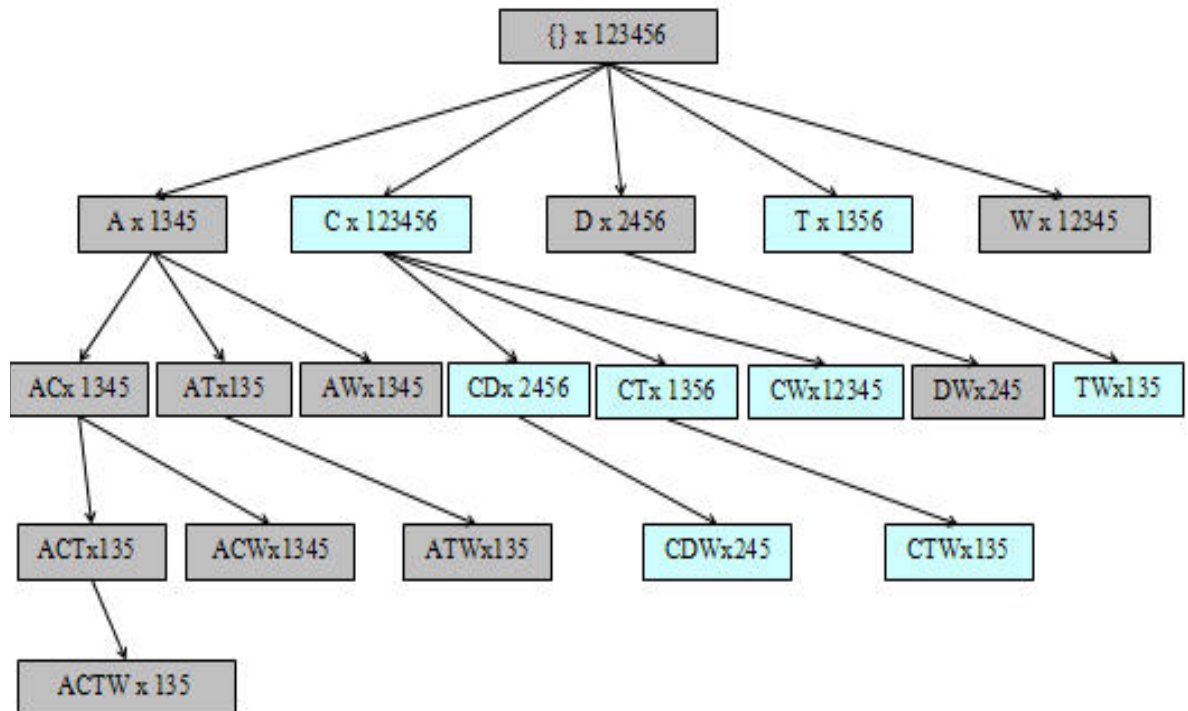
$T \times t(T) = T \times 1356$; $W \times t(W) = W \times 12345$

**Hình 2.19: Lớp 1 của cây IT-tree**

Lớp 2 là các nút con có itemset gồm 2 danh mục, được xây dựng bằng sự tổ hợp các itemset của các nút lớp 1 với Tidset được tính dựa trên Tidset của các nút lớp 1 :

$t(AC) = t(A) \cap t(C) = 1345 \cap 123456 = 1345 \Rightarrow \text{sup}(AC) > \text{minSup} \Rightarrow AC \times 1345$
tạo thành một nút mới ở lớp 2.

$t(AD) = t(A) \cap t(D) = 1345 \cap 2456 = 45 \Rightarrow \text{sup}(AD) < \text{minSup} \Rightarrow AD \times 45$ không
tạo thành nút mới ở lớp 2.



Hình 2.20: Cây IT-tree dùng Tidset với minSup =3

Như vậy có thể thấy các tập phổ biến được liệt kê hết trong các nút của cây IT-tree, và trong quá trình xây dựng cây It-tree chính là chúng ta đã đi tìm các tập phổ biến của CSDL.

2.3.3.3 Tìm tập phổ biến trên cây IT-tree

Thuật toán:

Hàm createITtree ()

$[Gen] = \{ i \in I \mid \text{sup}(i) \geq \text{minSup} \}$

enumerateFI ([Gen])

Hàm enumerateFI ([P])

for all $p_i \in [P]$ do

$[P_i] = \emptyset$

for all $p_j \in [P]$ with $j > i$ do

$I_{ij} = p_i \cup p_j$

$T_{ij} = t(p_i) \cap t(p_j)$

if $|T_{ij}| \geq \text{minSup}$ then

$[P_i] = [P_i] \cup \{ I_{ij} \times T_{ij} \}$

enumerateFI([P_i])

Giải thích thuật toán:

Thuật toán bắt đầu với việc sinh tập Gen là các tập phổ biến chỉ có 1 danh mục. Tập Gen tạo thành lớp thứ nhất của cây IT-tree.

Lớp thứ k của cây IT-tree được tạo thành việc xét tổ hợp theo thứ tự từng cặp phần tử của lớp thứ k-1, tính độ phổ biến dựa vào Tidset nếu thỏa mãn ngưỡng minSup thì bổ sung cặp IT-pair vào lớp k.

Nếu lớp k không rỗng thì gọi đệ quy để tính lớp k+1

Nhận xét:

Thuật toán dựa vào phần giao giữa các Tidset để tính nhanh độ phổ biến nên chỉ cần quét CSDL một lần, tuy nhiên lại tốn không gian để lưu trữ Tidset. Khi số tập phổ biến lớn thì thời gian khai thác luật sẽ lớn. Chúng ta có thể áp dụng khái niệm Diffset để tính nhanh độ phổ biến nhằm làm giảm không gian lưu trữ Tidset.

Khái niệm Diffset:

Diffset của X so với Y, ký hiệu $d(X, Y)$ được định nghĩa:

$$d(XY) = t(X) - t(Y)$$

Như vậy Diffset của X so với Y là các giao dịch có chứa X nhưng không chứa Y.

Các tính chất của Diffset:

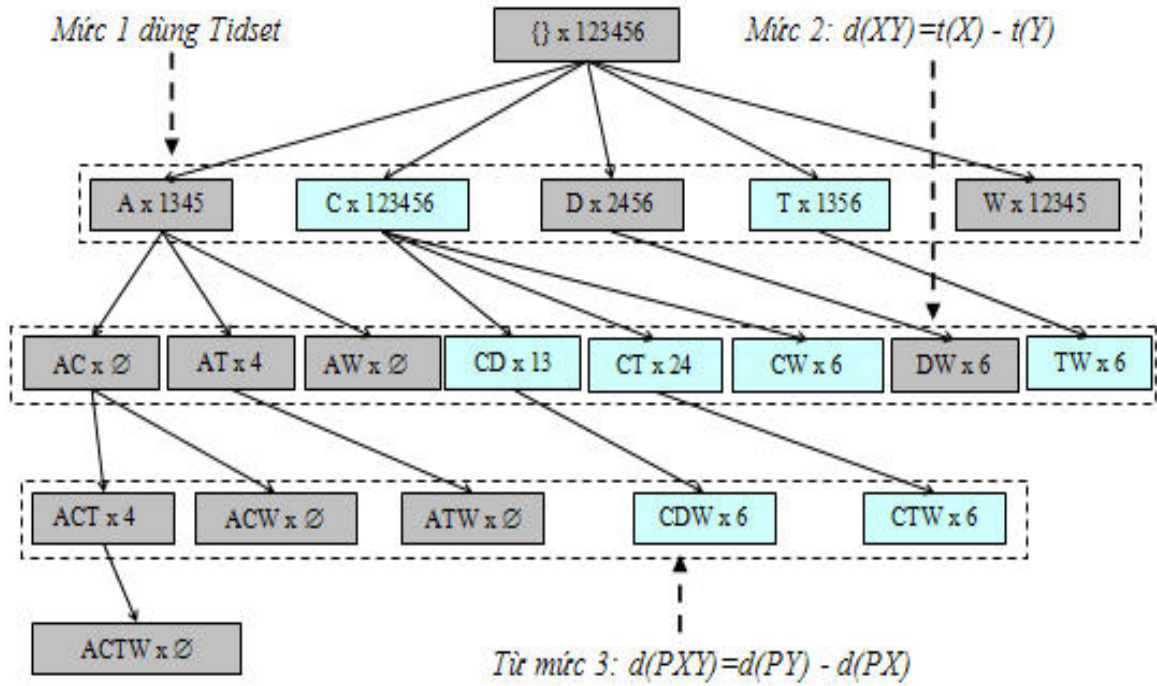
$$(1). \text{sup}(XY) = \text{sup}(X) - |d(XY)|$$

$$(2). d(PXY) = d(PY) - d(PX)$$

(3). Diffset thường khá nhỏ so với Tidset.

Từ các tính chất (1), (2), (3) ta có thể sử dụng Diffset để thay thế Tidset trong quá trình tạo cây IT-tree.

Ví dụ minh họa:



Hình 2.21: Cây IT-tree dùng Diffset với minSup =50%

2.3.3.4 Tìm tập phổ biến đóng trên cây IT-tree

Nhận xét về IT-pair:

Dựa trên các tính chất của IT-pair ta có các nhận xét sau:

(1). Nếu $t(X_i) = t(X_j)$ thì

$$|t(X_i)| = |t(X_j)| = |t(X_i \cup X_j)|$$

$$X_i \subset (X_i \cup X_j)$$

$$X_j \subset (X_i \cup X_j)$$

X_i, X_j không phải là các tập đóng.

(2). Nếu $t(X_i) \subset t(X_j)$ thì

$c(X_i) = c(X_i \cup X_j) \Rightarrow X_i$ không là tập đóng.

$t(X_i) \neq t(X_j) \Rightarrow X_i, X_j$ thuộc về hai tập đóng khác nhau

(3) Nếu $t(X_i) \supset t(X_j)$ thì

$c(X_j) = c(X_i \cup X_j) \Rightarrow X_j$ không là tập đóng.

$t(X_i) \neq t(X_j) \Rightarrow X_i, X_j$ thuộc về hai tập đóng khác nhau

(4). Nếu $\begin{cases} t(X_i) \not\subseteq t(X_j) \\ t(X_i) \not\supseteq t(X_j) \end{cases}$ Thì $c(X_i) \neq c(X_j) \neq c(X_i \cup X_j)$
nên X_i, X_j và $X_i \cup X_j$ sẽ thuộc về 3 tập đóng khác nhau

Dựa trên các nhận xét trên các tác giả M.J Zaki và C.Hsiao trong bài báo [9] đã trình bày thuật toán CHARM để tìm các tập phổ biến đóng trên cây IT-tree.

Thuật toán CHARM:

CHARM (D, minSup)

$[\emptyset] = \{ k_i \times t(k_i) : k_i \in I \wedge \text{sup}(k_i) \geq \text{minSup} \}$

CHARM-EXTEND ([\emptyset], C = \emptyset)

return C

CHARM-EXTEND ([P], C)

for each $k_i \times t(k_i)$ in [P] do

$P_i = P \cup k_i$ and $[P_i] = \emptyset$

for each $k_j \times t(k_j)$ in [P] with $j > i$ do

$X = k_j$

$Y = t(k_i) \cap t(k_j)$

CHARM-PROPERTY ($X \times Y, k_i, k_j, [P_i], [P]$)

SUBSUMPTION-CHECK (C, P_i)

CHARM-EXTEND ([P_i], C)

delete ([P_i])

CHARM-PROPERTY ($X \times Y, k_i, k_j, [P_i], [P]$)

```

if  $\text{sup}(X) \geq \text{minSup}$  then
    if  $t(k_i) = t(k_j)$  then
        Remove  $k_j$  from  $[P]$ 
         $P_i = P_i \cup k_j$ 
    elseif  $t(k_i) \subset t(k_j)$  then
         $P_i = P_i \cup k_j$ 
    elseif  $t(k_i) \supset t(k_j)$  then
        Remove  $k_j$  from  $[P]$ 
        Add  $X \times Y$  to  $[P_i]$ 
    else
        Add  $X \times Y$  to  $[P_i]$ 

```

SUBSUMPTION-CHECK (C, P)

```

for all  $Y \in \text{HASHTABLE} [ /t(P) / ]$  do
    if  $\text{sup}(P) = \text{sup}(Y)$  and  $\text{not}(P \subset Y)$  then
         $C = C \cup P$ 

```

Giải thích :

Ở bước thêm tập phổ biến vào cây IT-tree thì thuật toán CHARM sử dụng hàm **CHARM-PROPERTY ($X \times Y, k_i, k_j, [P_i], [P]$)** để kiểm tra xem tập phổ biến đó có khả năng là tập đóng hay không dựa vào các nhận xét về IT-pair ở trên.

Sau đó sử dụng tiếp hàm **SUBSUMPTION-CHECK (C, P)** để kiểm tra tính đóng của tập P dựa trên bảng băm **HASHTABLE**

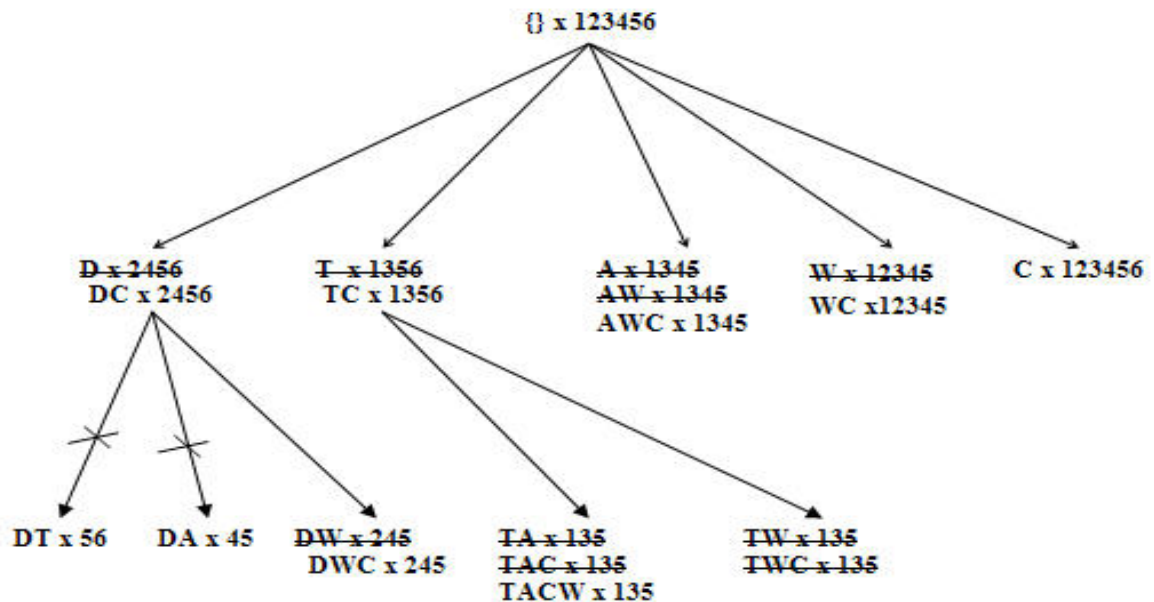
Minh họa thuật toán :

Sử dụng CSDL cho ở bảng 2.19 với $\text{minSup} = 50\%$

Bảng 2.18 Bảng CSDL minh họa tìm tập phổ biến đóng trên IT-tree

Mã giao dịch	Nội dung giao dịch	Mã danh mục	Các giao dịch có chứa danh mục
1	A, C, T, W	A	1, 3, 4, 5
2	C, D, W	C	1, 2, 3, 4, 5, 6
3	A, C, T, W	D	2, 4, 5, 6
4	A, C, D, W	T	1, 3, 5, 6
5	A, C, D, T, W	W	1, 2, 3, 4, 5
6	C, D, T		

Cây IT-tree lưu giữ các tập phổ biến đóng FCI sẽ được xây dựng như hình 2.38

**Hình 2.22: Minh họa xây dựng IT-tree bằng Charm**

Đầu tiên, tập $I = \{A, C, D, T, W\}$ sẽ được sắp xếp theo chiều tăng dần của độ phổ biến thành $K = \{D, T, A, W, C\}$.

Với $k_i = D$:

Kết hợp với các $k_j \in \{T, A, W, C\}$ thành các nút con $\{DT, DA, DW, DC\}$

Do $\text{sup}(DT) = \text{sup}(DA) = 2 < \text{minSup}$ nên không được sinh ra ở mức kế tiếp.

Do $\text{sup}(DW) = 3 = \text{minSup}$ nên nút $(DW \times 245)$ được thêm vào cây.

Do $t(D) \subset t(C)$ nên theo tính chất 2 của IT-pair thì D không thể là tập đóng và ta thay $(D \times 2456)$ bằng $(DC \times 2456)$ và $(DW \times 245)$ bằng $(DWC \times 245)$

Với $k_i = T$:

Kết hợp với các $k_j \in \{ A, W, C \}$ thành các nút con $\{TA, TW, TC\}$

Do $\text{sup}(TA) = 3 = \text{minSup}$ nên nút $(TA \times 135)$ được thêm vào cây

Do $\text{sup}(TW) = 3 = \text{minSup}$ nên nút $(TW \times 135)$ được thêm vào cây

Do $t(T) \subset t(C)$ nên theo tính chất 2 của IT-pair thì T không thể là tập đóng và ta thay $(T \times 1356)$ bằng $(TC \times 1356)$, $(TA \times 135)$ bằng $(TAC \times 135)$ và $(TW \times 135)$ bằng $(TWC \times 135)$.

Do $t(TAC) = t(TWC)$ nên thay $(TAC \times 135)$ bằng $(TACW \times 135)$ và xóa $(TWC \times 135)$

Với $k_i = A$:

Kết hợp với các $k_j \in \{ W, C \}$ thành các nút con $\{AW, AC\}$

Do $t(A) \subset t(W)$ nên xóa $(A \times 1345)$ thay bằng $(AW \times 1345)$

Do $t(AW) \subset t(C)$ nên xóa $(AW \times 1345)$ thay bằng $(AWC \times 1345)$

Với $k_i = W$:

Kết hợp với các $k_j \in \{ C \}$ thành các nút con $\{WC\}$

Do $t(W) \subset t(C)$ nên xóa $(W \times 12345)$ thay bằng $(WC \times 12345)$

Kết quả sinh ra được 7 tập phổ biến đóng thỏa ngưỡng $\text{minSup} = 50\%$ lần lượt là:

$\{ C:6; DC:4; TC:4; CW:5; AWC:4; DWC:3; TAWC:3 \}$

Nhận xét: Số lượng tập phổ biến đóng thường nhỏ hơn nhiều so với số tập phổ biến vì thế việc khai thác luật từ chúng sẽ hiệu quả hơn. Thêm nữa mức tìm kiếm trên IT-tree đối với FCI thấp hơn so với tìm FI nên yêu cầu bộ nhớ cho quá trình gọi đệ quy sẽ nhỏ hơn.

2.4 Kết luận

Giai đoạn tìm tập phổ biến là một giai đoạn quan trọng và tốn thời gian nhất trong quá trình khai khoáng luật kết hợp. Do đó có rất nhiều nghiên cứu tập trung vào vấn đề này. Hầu hết đều tập trung giải quyết các nhiệm vụ sau :

- Đọc CSDL càng ít lần càng tốt
- Sử dụng bộ nhớ cho quá trình gọi đệ quy càng ít càng tốt.
- Tìm các tập phổ biến đóng thay cho các tập phổ biến nói chung để dễ dàng hơn cho quá trình khai thác luật.

Với họ các thuật toán Apriori là cách thức nguyên thủy do Agrawal [13] đề xuất thực hiện tìm tập phổ biến dựa trên việc đọc lại CSDL nhiều lần để phát sinh ứng viên, nên cách thức này tốn thời gian và không hiệu quả. Để khắc phục các hạn chế của phương pháp sinh ứng viên, các nhà nghiên cứu đề xuất ra các cấu trúc cây FP-tree và IT-tree để khai thác tập phổ biến trên đó hiệu quả hơn nhiều.

Cấu trúc cây FP-tree được tác giả J.Han và các đồng sự giới thiệu trong [5] dùng để tổ chức lại CSDL thuận lợi hơn cho quá trình tìm tập phổ biến. Lợi ích của cấu trúc FP-tree nằm ở chỗ toàn bộ thông tin trong CSDL được lưu trữ trong cây với tỉ lệ nén cao và thuận lợi cho quá trình duyệt theo chiều sâu cũng như áp dụng mô hình chia để trị. Sau đó của J.Han và đồng sự trong các bài báo [6] [4] áp dụng các tính chất của tập đóng vào cây FP-tree để khai thác các tập phổ biến đóng với các thuật toán CLOSET và CLOSET+. Việc chỉ khai thác các tập phổ biến đóng giới hạn được không gian tìm kiếm nhỏ hơn và hiệu quả hơn khi khai thác luật

Cấu trúc cây IT-tree lưu trữ trực tiếp các tập phổ biến trên các nút của cây. Việc xây dựng cây IT-tree dựa vào phần giao giữa các Tidset để tính nhanh độ phổ biến nên khá hiệu quả. Có thể sử dụng Diffset thay cho Tidset để giảm không gian lưu trữ. Việc tìm tập phổ biến đóng với cây IT-tree được tác giả M.J Zaki trình bày trong [9] với thuật toán CHARM. Ngoài ra, để thuận lợi cho quá trình khai thác luật, các nhà nghiên cứu còn xây dựng dàn tập phổ biến và dàn tập phổ biến đóng với các thuật toán như CHARM-L hay phương pháp lai ghép.