PHỤ LỤC

LỜI MỞ ĐẦU

Với sự phát triển của công nghệ thông tin thì khối lượng dữ liệu lưu trữ ngày càng lớn, và giữa những lượng dữ liệu khổng lồ đó lại ẩn chứa một số thông tin được coi là chìa khóa dẫn đến thành công của mọi lĩnh vực từ hoạt động sản xuất đến kinh doanh. Việc khai thác, chiếc lọc thông tin ứng dụng vào cuộc sống của con người không chỉ dừng lại là một kĩ thuật đơn thuần, nó đòi hỏi sự ra đời của ngành khoa học mới: khoa học về phát hiện tri thức và khai phá dữ liệu (Knowledge Discovery and Data Mining - KDD).

Khai phá dữ liệu là ngành khoa học đang ngày được quan tâm nghiên cứu và phát triển do những ứng dụng thiết thực mà nó mang lại. Khai phá dữ liệu là phần cốt lõi của phát hiện tri thức, trong khai phá dữ liệu phát hiện các luật là một trong những nội dung cơ bản và phổ biến nhất. Các phương pháp phát hiện luật nhằm tìm ra sự phụ thuộc giữa các tính chất của các đối tượng hay các thuộc tính trong cơ sở dữ liệu.

Trên cơ sở đó bài thu hoạch tập trung tìm hiểu một trong hướng tiếp cận khai phá dữ liệu thông qua thuật toán Apriori và một số thuật toán xuất phát từ Apriori.

Em xin cảm ơn những kiến thức nền quý báo của **GS. TSKH Hoàng Kiếm** đã truyền đạt cho em, để em có cơ sở nghiên cứu và tìm hiểu nhiều hơn, sâu hơn.

Do quá trình nghiên cứu cũng như kiến thức và tài liệu còn nhiều hạn chế nên bài viết còn nhiều thiếu sót, chưa được đầy đủ. Em mong nhận được sự góp ý của Thầy để bài viết được thực sự hoàn chỉnh hơn.

PHẦN 1: PHÁT HIỆN TRI THỨC VÀ KHAI PHÁ DỮ LIỆU

I. PHÁT HIỆN TRI THỨC (KNOWLEDGE DISCOVERY)

1. Phát hiện tri thức

Chúng ta có thể xem tri thức như là các thông tin tích hợp, bao gồm các sự kiện và các mối quan hệ giữa chúng. Các mối quan hệ này có thể được hiểu ra được phát hiện hoặc cũng có thể được học. Nói cách khác tri thức có thể được coi là dữ liệu có độ trừu tượng và tổ chức cao.

Phát hiện tri thức trong các cơ sở dữ liệu là một quy trình nhận biết các mẫu hoặc các mô hình trong dữ liệu với các tính năng: hợp thức, mới, khả ích và có thể hiểu được. Còn khai thác dữ liệu là một bước trong quy trình phát hiện tri thức: gồm các thuật toán khai thác dữ liệu chuyên dùng dưới một số quy định về hiệu quả tính toán chấp nhận được để tìm các mẫu các mô hình trong dữ liệu. Nói một cách khác mục đích của phát hiện tri thức và khai phá dữ liệu chính là tìm ra các mẫu và các mô hình đang tồn tại trong cơ sở dữ liệu nhưng bị che khuất bởi hàng núi dữ liệu.

2. Quá trình phát hiện tri thức

a. Làm sạch dữ liệu (Data cleaning):

Là quá trình loại bỏ nhiễu - những bộ dữ liệu không bình thường, không tuân theo quy luật, nguyên tắc hay mô hình dữ liệu (còn gọi là các phần tử ngoài cuộc), và dữ liệu không nhất quán.

b. Tích hợp dữ liệu (Data intergation):

Dữ liệu có thể được thu thập từ nhiều nguồn khác nhau, hoặc có thể thu thập dữ liệu nhiều lần. Dữ liệu cuối của quá trình có có thể là kết quả của việc tổ hợp lại những lần thực hiện thu thập dữ liệu.

c. Lựa chọn dữ liệu (Data selection):

Kết quả đạt được của quá trình này là những dữ liệu thích hợp với nhiệm vụ phân tích được trích rút từ cơ sở dữ liệu.

d. Chuyển đổi dữ liệu (Data transformation):

Dữ liệu được chuyển đổi hay được hợp nhất về dạng thích hợp cho việc khai phá.

e. Khai phá dữ liệu (Data mining):

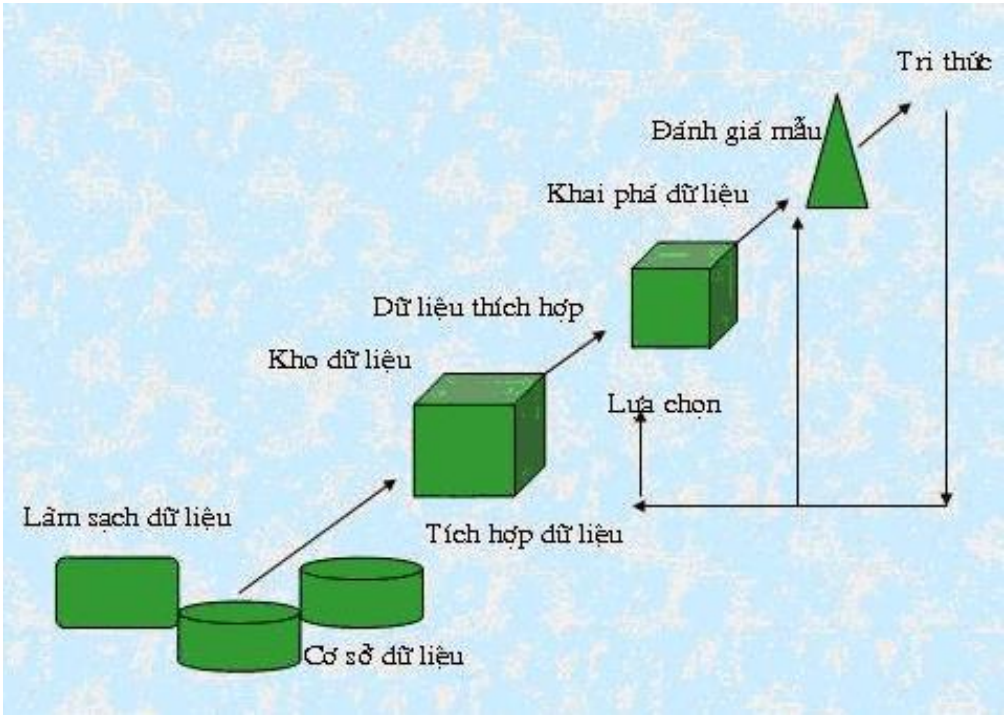
Đây là một tiến trình cốt yếu trong đó các phương pháp thông minh được áp dụng nhằm trích ra các mẫu dữ liệu.

f. Đánh giá mẫu (Pattern evaluation):

Dựa trên một số độ đo nào đó xác định lợi ích thực sự, độ quan trọng của các mẫu biểu diễn tri thức.

g. Biểu diễn tri thức (Knowledge presentation):

Ở giai đoạn này, các kĩ thuật biểu diễn và hiển thị được sử dụng để đưa tri thức đã lấy ra được cho người dùng.



II. KHAI PHÁ DỮ LIỆU (DATA MINING):

1. Khai phá dữ liệu

Ở một mức độ trừu tượng nhất định có thể định nghĩa về khai phá dữ liệu (*Data Mining)* là một quá trình tìm kiếm, phát hiện các tri thức mới, tiềm ẩn, hữu dụng trong CSDL lớn.

2. Mục đích của việc khai phá dữ liệu

- Khai phá dữ liệu cung cấp những thông tin giúp hỗ trợ ra quyết định.
- Cung cấp những thông tin giúp dự báo: Ví dụ dự báo dân số thế giới căn cứ vào số liệu của dân số thế giới những năm trước đó.
- Có thể giúp khái quát dữ liệu.

3. Các ứng dụng trong khai phá dữ liệu

Khai phá dữ liệu (KPDL) đang được áp dụng một cách rộng rãi trong nhiều lĩnh vực kinh doanh và đời sống khác nhau: marketing, tài chính, ngân hàng và bảo hiểm, khoa học, y tế, an ninh, internet… Rất nhiều tổ chức và công ty lớn trên thế giới đã áp dụng kĩ thuật khai phá dữ liệu vào các hoạt động sản xuất kinh doanh của mình và thu được những lợi ích to lớn. Các công ty phần mềm lớn trên thế giới cũng rất quan tâm và chú trọng tới việc nghiên cứu và phát triển kĩ thuật khai phá dữ liệu: Oracle tích hợp các công cụ khai phá dữ liệu vào bộ Oracle9i, IBM đã đi tiên phong trong việc phát triển các ứng dụng khai phá dữ liệu với các ứng dụng như Intelligence Miner ...

PHẦN 2: THUẬT TOÁN APRIORI VÀ CÁC THUẬT TOÁN XUẤT PHÁT TỪ APRIORI

I. THUẬT TOÁN APRIORI:

Apriori là thuật toán được Rakesh Agrawal, Tomasz Imielinski, Arun Swami đề xuất lần đầu vào năm 1993. Bài toán được phát biểu: Tìm t có độ hỗ trợ s thỏa mãn $s \geq s_0$ và độ tin cậy $c \geq c_0$ (s_0, c_0 là hai ngưỡng do người dùng xác định và s_0=minsupp, c_0 =minconf) . Ký hiệu L_k tập các tập k - mục phổ biến, C_k tập các tập k-mục ứng viên.

Bài toán đặt ra là:

1) Tìm tất cả các tập mục phổ biến với minsupp nào đó.

2) Sử dụng các tập mục phổ biến để sinh ra các luật kết hợp với độ tin cậy minconf nào đó.

1. NGUYÊN TẮC APRIORI

- Đếm số lượng của từng Item, tìm các Item xuất hiện nhiều nhất.
- Tìm các cặp ứng viên: Đếm các cặp => cặp item xuất hiện nhiều nhất.
- Tìm các bộ ba ứng viên: Đếm các bộ ba => bộ ba item xuất hiện nhiều nhất. Và tiếp tục với bộ 4, bộ 5, …
- Nguyên tắc chủ yếu: Mọi tập con của tập phổ biến phải là tập con phổ biến.

2. MÔ TẢ THUẬT TOÁN APRIORI

- **Bước 1**: Đếm số support cho mỗi tập gồm một phần tử và xem chúng như một Large itemset. Support của chúng là minsup.
- **Bước 2**: Với mỗi tập Large item bổ sung các item vào và tạo một Large itemset mới, tập này được gọi là tập ứng viên (Candidate itemset - C). Đếm số support cho mỗi tập C trên cơ sở dữ liệu, từ đó quyết định tập C nào là *Large Item* thực sự, và ta dùng làm hạt giống cho bước kế tiếp.
- **Bước 3**: Lặp lại bước 2 cho đến khi không còn tìm thấy thêm, một tập Large itemset nữa.

3. NỘI DUNG THUẬT TOÁN APRIORI:

Input: Tập các giao dịch D, ngưỡng support tối thiểu minsup

Output: L- tập mục phổ biến trong D

Method:

1. L_1=***Large_1_ItemSets***()
2. **for** (k=2; $L_{k-1} \neq \varnothing$; k++) **do**
3. **begin**
4. C_k=apriori-gen(L_{k-1});
5. **for** (mỗi một giao dịch $T \in D$) **do**
6. **begin**
7. C_T = subset(C_k, T);
8. **for** (mỗi một ứng cử viên $c \in C_T$) **do**
9. c.count++;
10. **end;**
11. $L_k = \{c \in C_k | c.count \geq minsup\}$
12. **end;**
13. return $\cup_k L_k$

– Hàm ***Large_1_ItemSets***() trả về các *Item* có số *support* lớn hơn hay bằng minsup.

1. ***for all*** *transaction* $t \in D$ ***do***
2. ***for all*** *item* $i \in t$ ***do***

3. i.*count* ++;

4. L1={i | i.*count* ≥ *minsup*};

– Hàm ***Apriori_Gen*** (L_{k-1}) thực hiện việc kết các cặp (*k-1*) *ItemSet* để phát sinh các tập *k_ItemSet* mới. Tham số của hàm là L_{k-1} – tập tất cả các (*k-1*)-*ItemSet* và kết quả trả về của hàm là tập các *k-ItemSet*.

1. ***Join*** L_{k-1} ***with*** L_{k-1};
2. ***Insert into*** C_k
3. ***select*** $p.item_1, p.item_2, \ldots p.item_{k-1}, q.item_{k-1}$
4. ***from*** L_{k-1} ***as*** p, L_{k-1} ***as*** q;
5. ***where*** $(p.item_1 = q.item_1) \wedge \ldots \wedge (p.item_{k-2} = q.item_{k-2}) \wedge (p.item_{k-1} < q.item_{k-1})$;

   Điều kiện $(p.item_{k-1} < q.item_{k-1})$ sẽ bảo đảm không phát sinh các bộ trùng nhau.

4. MINH HỌA THUẬT TOÁN APRIORI:

Minh họa 1: Cho một ví dụ tập các giao dịch từ các hóa đơn mua hàng như sau:

TID	**Các món hàng được mua (Item)**
1	{ b, m, t, y }
2	{ b, m }
3	{ p, s, t }
4	{ a, b, c, d }
5	{ a, b }
6	{ e, t, y }
7	{ a, b, m }

Cho *Min Support = 30%, Min Confidence = 60%*

Tính tập Large 1-item, ta có F1:

Tập Item	Số lần xuất hiện
{a}	3
{b}	5
{m}	3
{t}	3

Ở bước kết Từ F1 trên ta có tập C2 gồm các cặp 2-item:

{{a, b}, {a, m}, {a,t}, {b,m}, {b,t}, {m,t}}

Tính tập Large 2-item, ta có F2:

Tập Item	Số lần xuất hiện
{a, b}	3
{a, m}	1
{a, t}	0
{b, m}	3
{b, t}	1
{m, t}	1

Chỉ lấy các cặp 2-items có Support > Min Support (= 30%) gồm: {a, b} và {b, m}

Phát sinh luật:

a $\rightarrow$ b có độ Confidence 3/3 = 100%

b $\rightarrow$ a có độ Confidence 3/5 = 60%

b $\rightarrow$ m có độ Confidence 3/5 = 60%

m $\rightarrow$ b có độ Confidence 3/3 = 100%

Ở bước lược bỏ ta có F2 = {{a, b}, {b,m}}

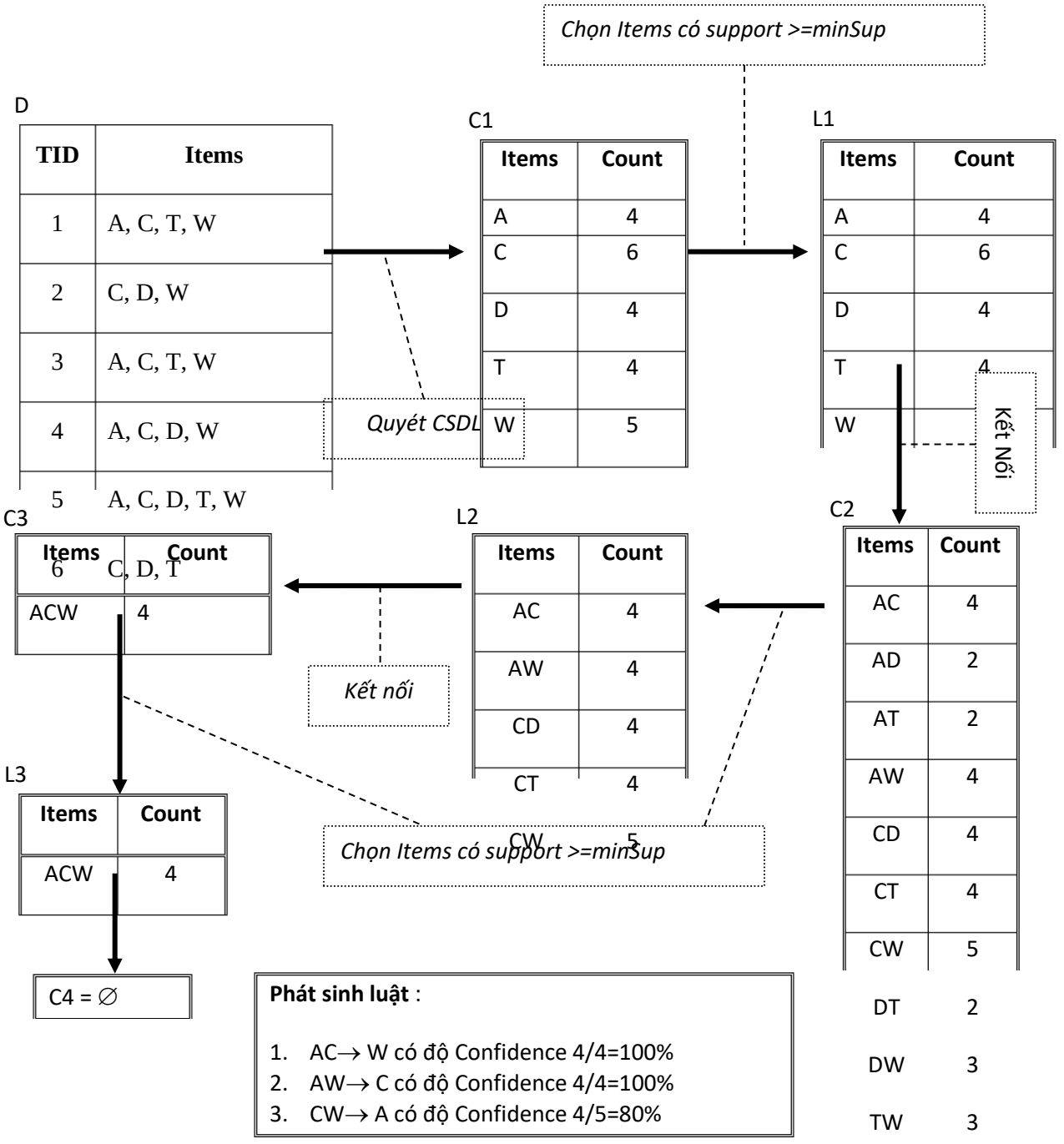
Ở bước kết Từ F2 ta có tập C3 gồm các cặp 3-item là {∅}

Thuật toán kết thúc.

Minh họa 2: Xét cơ sở dữ liệu mẫu như sau

TID	**Item**
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

minSup = 60%
minConf = 80%


Chọn Items có support >=minSup
D
TID
Items
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
C1
Items
Count
A
4
C
6
D
4
T
4
W
5
Quyét CSDL
L1
Items
Count
A
4
C
6
D
4
T
4
W
Kết Nối
C2
Items
Count
AC
4
AD
2
AT
2
AW
4
CD
4
CT
4
CW
5
DT
2
DW
3
TW
3
L2
Items
Count
AC
4
AW
4
CD
4
CT
4
CW
5
Kết nối
C3
Items
Count
ACW
4
L3
Items
Count
ACW
4
C4 = ∅
Chọn Items có support >=minSup
Phát sinh luật :
1. AC→ W có độ Confidence 4/4=100%
2. AW→ C có độ Confidence 4/4=100%
3. CW→ A có độ Confidence 4/5=80%

II. THUẬT TOÁN APRIORITID:

Giải thuật **AprioriTID** là phần mở rộng theo hướng tiếp cận cơ bản của giải thuật Apriori. Thay vì dựa vào cơ sở dữ liệu thô giải thuật AprioriTID biểu diễn bên trong mỗi giao tác bởi các ứng viên hiện hành.

1. THUẬT TOÁN APRIORITID:

- Thuật toán *AprioriTID* sử dụng hàm *Apriori_Gen* để tạo các tập *ItemSet* ứng viên. Thuật toán này không dùng cơ sở dữ liệu D để đếm *support* kể từ bước thứ hai, thay vào đó là sử dụng tập C_k cho mục đích này. Mỗi thành viên của tập C_k có dạng <TID, $\{X_k\}$> với X_k là tập *k-ItemSet* thể hiện một phần giao tác t có mã là TID, hay ta có thể viết <t.TID, $\{c \in C_k \mid c$ có trong t$\}$>.
- Nếu một giao tác không chứa bất kỳ một tập *k-ItemSet* ứng viên nào, thì giao tác này không được đưa vào $\overline{C}_k$. Do đó, số lượng ứng viên được đưa vào $\overline{C}_k$ có thể nhỏ hơn số lượng các giao tác trong cơ sở dữ liệu.

2. MÔ PHỎNG THUẬT TOÁN APRIORI-TID

- **Bước 1**: Quét tất cả các giao dịch để tìm tất cả các item có độ Support lớn hơn Min Support và đưa tập Large 1-Item vào F1
- **Bước 2**: Đưa toàn bộ các Tid của giao dịch cùng các Items vào C'1 dưới dạng <Tid,{X1}>
- **Bước 3**: Xây dựng các cặp 2-items từ F1 đưa vào tập ứng viên C2. Quét tất cả các giao dịch trong C'1 để tìm tất cả các tập Large 2-Item từ C2 đưa vào C'2 dưới dạng <Tid,{X2}>, đồng thời đưa các tập Large 2-Item ứng viên vào F2.
- **Bước 4**: Phát sinh Luật. Xây dựng các cặp k items từ Fk-1 đưa vào tập ứng viên Ck. Quét tất cả các giao dịch trong C'k-1 để tìm tất cả các tập Large k-Item từ Ck và đưa vào C'k dưới dạng <Tid,{Xk}>, đồng thời đưa các tập Large k-Item vào Fk. Lặp lại Bước 4 cho đến khi hết ứng viên mới.

3. NỘI DUNG TỐI ƯU THUẬT TOÁN APRIORI-TID

1. L_1 = ***Large_1_ItemSets*** ();
2. $\overline{C}_1$ = Database D;
3. ***for*** (k=2; $L_{k-1} \neq \emptyset$; k++) ***do begin***
4. C_k = ***Apriori_Gen***(L_{k-1});
5. $\overline{C}_k$ = $\emptyset$;
6. ***for all*** t$\in$ $\overline{C}_{k-1}$ ***do begin***
7. C_t = {c $\in$ C_k | (c-c[k]) $\in$ t.Set_of_ItemSets ^
8. (c-c[k-1] $\in$ t.Set_of_ItemSets};
9. ***for all*** candidate c $\in$ C_t ***do***
10. c.count ++;
11. ***if*** ($C_t \neq \emptyset$) ***then*** $\overline{C}_k$ += < t.TID, C_t >;
12. ***End***
13. L_k = {c $\in \overline{C}_k$ | c.count $\geq$ minsup}
14. ***End***
15. Answer = $\cup_k L_k$;

4. CẤU TRÚC LƯU TRỮ:

– Mỗi tập *ItemSet* ứng viên sẽ được gán cho một mã số duy nhất, gọi là ***ID***. Mỗi tập *ItemSet* C_k được lưu trong một mảng. Một thành viên của $\overline{C}_k$ bây giờ có dạng <TID, ID >, mỗi $\overline{C}_k$ được lưu trong một cấu trúc tuần tự.

– Hàm ***Apriori_Gen*** phát sinh một tập các *k-ItemSet* ứng viên C_k bằng cách kết hai tập *Large (k-1)-ItemSets*. Mỗi *ItemSet* ứng viên ta thêm hai trường:

(i) generators.

(ii) extensions.

- Trường *generators* của tập *ItemSet* c_k lưu các ID của hai tập *Large (k-1)-ItemSet* kết với nhau để phát sinh c_k.
- Trường *extensions* của tập *ItemSet* c_k lưu những ID của các tập *Large (k+1)-ItemSet* kết với nhau để phát sinh c_k.
- Khi một *ItemSet* c_k ứng viên được phát sinh bằng cách kết 1^1_{k-1} và 1^2_{k-1}, thì các ID của 1^1_{k-1} vaø 1^2_{k-1} sẽ được lưu vào trường *generators* của c_k, đồng thời ID của ck được lưu vào trường *extension* của 1^1_{k-1}.
- Với cấu trúc lưu trữ này thì câu lệnh

$C_t = \{c \in C_k \mid (c\text{-}c[k]) \in t.\textit{Set\_of\_ItemSets} \wedge (c\text{-}c[k-1] \in t.\textit{Set\_of\_ItemSets}\}$;

sẽ được thực hiện như sau: trường t.*Set-of-ItemSets* của bản ghi *t* thuộc $\overline{C}_{k-1}$ lưu các ID của tập ứng viên (*k-1*)-*ItemSet* chứa trong giao tác t.TID. Với mỗi c_{k-1}, trường *extensions* chứa tập T_k là tập các ID của tất cả các tập *k-ItemSet* ứng viên được mở rộng từ c_{k-1}. Mỗi c_k trong T_k, trường *generators* chứa các ID của hai tập *ItemSet* dùng để phát sinh ra c_k. Nếu những tập *itemSet* này nằm trong danh sách các tập *ItemSet* của bản ghi *t*, thì có thể kết luận c_k thuộc giao tác t.TID, và c_k được thêm vào tập C_t.

5. MINH HỌA THUẬT TOÁN APRIORI-TID:

Cho một ví dụ tập các giao dịch Tid với các Items như sau:

Tid	**Items**
100	{1, 3, 4}
200	{2, 3, 5}
300	{1, 2, 3, 5}
400	{2, 5}

Cho *Min Support* = 50%, *Min Confidence* = 60%

Tính tập Large 1-item, ta có F1:

Tập 1-item	**Số lần xuất hiện**
{ 1 }	2
{ 2 }	3
{ 3 }	3
{ 5 }	3

Lấy toàn bộ <Tid,{X1}> đưa vào C'1

Tid	**Tập 1-Item**
100	{{1 }, {3}, {4}}
200	{{2}, {3}, {5}}
300	{{1}, {2}, {3}, {5}}
400	{{2}, {5}}

Ở bước kết Từ F1 trên ta có tập C2 gồm các cặp 2-item:

{{1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}}.

Xác định ứng viên từ C2 khi duyệt Tid trong C'1 và đưa vào C'2

Tid	**Tập 2-Item**
100	{{1,3}}
200	{{2,3}, {2,5}, {3,5}}
300	{{1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}}
400	{{2,5}}

Ở bước kết Từ F1 trên ta có tập C2 gồm các cặp 2-item:

{{1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}}.

Tính tập Large 2-Item, ta có F2

Tập 2-Item	**Số lần xuất hiện**
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Ở bước kết Từ F2 ta có tập C3 gồm cặp 3-item

{{2,3,5}}.

Xác định ứng viên từ C3 khi duyệt Tid trong C'2 và đưa vào C'3

Tid	**Tập 3-Itims**
200	{{2, 3, 5}}
300	{{2, 3, 5}}

Tính tập Large 3-Item, ta có F3:

Tập 3- Item	**Số lần xuất hiện**
{{2, 3, 5}}	2

Phát sinh luật:

2,3 → 5 có độ Confidence 2/2 = 100%

2,5 →3 có độ Confidence 2/3 = 66,66%

3,5 → 2 có độ Confidence 2/2 = 100%

Ở bước kết Từ F3 ta có tập C4 gồm các cặp 4-item là {∅}

Thuật toán kết thúc.

III. SO SÁNH THUẬT TOÁN APRIORI VÀ APRIORI-TID

1. Khuyết điểm của apriori:

Để xác định độ Support của các tập ứng viên, thuật toán luôn luôn phải quét lại toàn bộ các giao tác trong CSDL. Do vậy sẽ tiêu tốn rất nhiều thời gian khi số k-items tăng (số lần xét duyệt các giao tác tăng).

2. Khuyết điểm của apriori-Tid:

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 bằng 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 trong mà phải sử dụng kèm bộ nhớ ngoài.

IV. THUẬT TOÁN APRIORI-HYBRID

Thuật toán Apriori-Hybrid được coi như kết hợp giữa Thuật toán Apriori và thuật toán Apriori-TID.

Trong thuật toán Apriori-Hybrid, được sử dụng khi tổ chức lặp và chuyển sang Apriori-TID khi đã chắc chắn rằng tập C k đã vào bộ nhớ chính. Thuật toán Apriori-Hybrid được coi là tốt hơn so với Apriori và AprioriTID. Nhờ có nhận xét tinh tế là thuật toán Apriori chạy khá nhanh ở nhữngbước đầu tiên, còn thuật toán Apriori-TID chạy nhanh ở những bước sau (chạy khá chậm ở những bước đầu tiên), Agrawal đề nghị phương án lai ghép: không nhất thiết phải chạy tất cả các bước cùng một thuật toán giống nhau. Những bước đầu tiên, ông cho chạy thuật toán Apriori, sau đó khi tập các ứng cử viên khá lớn, sắp chứa đầy trong bộ nhớ tính toán, mới dùng thuật toán Apriori-TID.

Srikant đưa ra thêm một nhận xét: thời gian chuyển từ thuật toán Apriori sang thuật toán Apriori-TID tương đối tốn kém. Và thuật tóab lai ghép Apriori-Hybrid chỉ tỏ ra hiệu quả khi sự chuyển mạch này diễn ra ở gần cuối quá trình tìm kiếm tập xuất hiện σ thường xuyên.

TÀI LIỆU THAM KHẢO

[1] GS.TSKH Hoàng Kiếm. Bài giảng cao học môn học cơ sở tri thức và ứng dụng. ĐHKHTN-TPHCM.

[2]GS.TSKH Hoàng Kiếm, TS. Đỗ Văn Nhơn, Th.sĩ Đỗ Phúc. Giáo trình Các hệ cơ sở tri thức. Đại Học Quốc Gia TPHCM – 2002

[3] GS.TSKH Hoàng Kiếm, Th.sĩ Đinh Nguyễn Anh Dũng. Giáo trình Trí tuệ nhân tạo. Đại Học Quốc Gia TPHCM – 2002

[4]. Giáo trình khai thác dữ liệu, PGS.TS. Đỗ Phúc, Trường ĐH CNTT, ĐHQG TP.HCM, Nhà xuất bản ĐHQG TP.HCM, 2006