BÀI GIẢNG NHẬP MÔN KHAI PHÁ DỮ LIỆU

CHƯƠNG 3. KHAI PHÁ LUẬT KẾT HỢP

TS. Trần Mai Vũ
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
ĐẠI HỌC QUỐC GIA HÀ NỘI

Chương 4: Khai phá luật kết hợp

- Khai phá luật kết hợp (Association rule)
- Các thuật toán khai phá vô hướng luật kết hợp (giá trị lôgic đơn chiều) trong CSDL giao dịch
- Khai phá kiểu đa dạng luật kết hợp/tương quan
- Khai phá kết hợp dựa theo ràng buộc
- Khai phá mẫu dãy

http://michael.hahsler.net/research/arules_RUG_2015/demo/

Khái niệm cơ sở: Tập phổ biến và luật kết hợp

Một số ví dụ về *"luật kết hợp" (associate rule)*

- "98% khách hàng *mà* mua tạp chí thể thao *thì đều* mua các tạp chí về ôtô" sự ***kết hợp*** giữa "*tạp chí thể thao*" với "*tạp chí về ôtô*"
- "60% khách hàng *mà* mua bia tại siêu thị *thì đều* mua bỉm trẻ em" sự ***kết hợp*** giữa "*bia*" với "*bỉm trẻ em*"
- "Có tới 70% người truy nhập Web vào địa chỉ Url 1 thì cũng vào địa chỉ Url 2 trong một phiên truy nhập web" sự ***kết hợp*** giữa *"Url 1"* với *"Url 2"*. Khai phá dữ liệu sử dụng Web (Dữ liệu từ file log của các site, chẳng hạn được MS cung cấp).
- Các Url có gắn với nhãn "lớp" là các đặc trưng thì có luật kết hợp liên quan giữa các lớp Url này.

Khái niệm cơ sở: Tập phổ biến và luật kết hợp

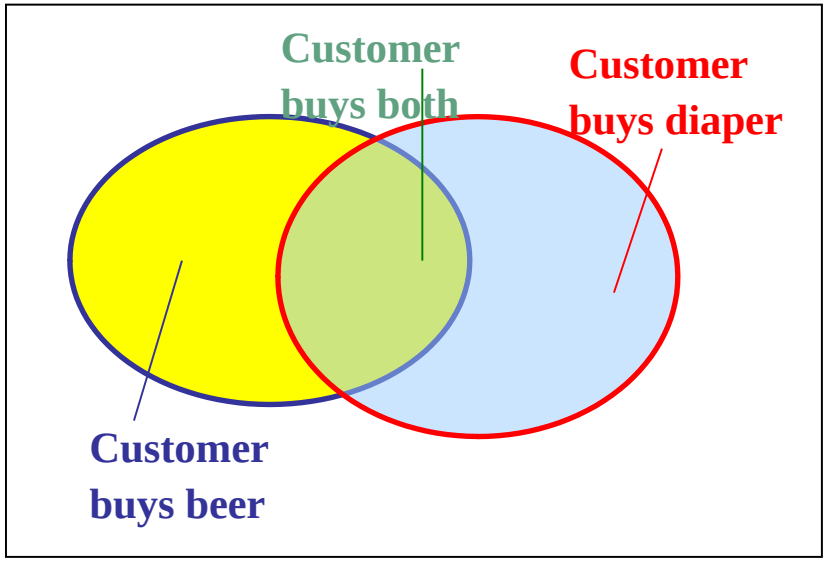
Cơ sở dữ liệu giao dịch (transaction database)

- *Giao dịch*: danh sách các mục (mục: item, mặt hàng) trong một phiếu mua hàng. Giao dịch T là một tập mục.
- Tập toàn bộ các mục I = {i_1, i_2, …, i_k} "tất cả các mặt hàng". Một giao dịch T là một tập con của I: T I. Mỗi giao dịch T có một định danh là T_{ID}.
- A là một tập mục A I và T là một giao dịch: Gọi T chứa A nếu A T.

- Luật kết hợp
  - Gọi A B là một "luật kết hợp" nếu A I, B I và A B= .
  - Luật kết hợp A B có độ hỗ trợ (support) *s* trong CSDL giao dịch D nếu trong D có s% các giao dịch T chứa AB: chính là xác suất P(AB). Tập mục A có P(A) s>0 (với s cho trước) được gọi là *tập phổ biến (frequent set)*. Luật kết hợp A B có độ tin cậy (confidence) *c* trong CSDL D nếu như trong D có c% các giao dịch T chứa A thì cũng chứa B: chính là xác suất P(B|A).
  - Support (A B) = P(A B) : 1 s (A B) 0
  - Confidence (A B) = P(B|A) : 1 c (A B) 0
  - Luật A B được gọi là đảm bảo độ hỗ trợ s trong D nếu s(A B) s. Luật A B được gọi là đảm bảo độ tin cậy c trong D nếu c(A B) c. Tập mạnh.

Khái niệm cơ bản: Mẫu phổ biến và luật kết hợp

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F



- Tập mục I={i_1, …, i_k}. CSDL giao dịch D = {d I}
- A, B I, A B= : *A* B là luật kết hợp
- Bài toán tìm luật kết hợp.

  Cho trước độ hỗ trợ tối thiểu s>0, độ tin cậy tối thiểu c>0. Hãy tìm mọi luật kết hợp mạnh *X Y.*

Giả sử min_support = 50%, min_conf = 50%:

$A \Rightarrow C$ (50%, 66.7%)

$C \Rightarrow A$ (50%, 100%)

- Hãy trình bày các nhận xét về khái niệm luật kết hợp với khái niệm phụ thuộc hàm.
- Các tính chất Armstrong ở đây.

Một ví dụ tìm luật kết hợp

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%
Min. confidence 50%

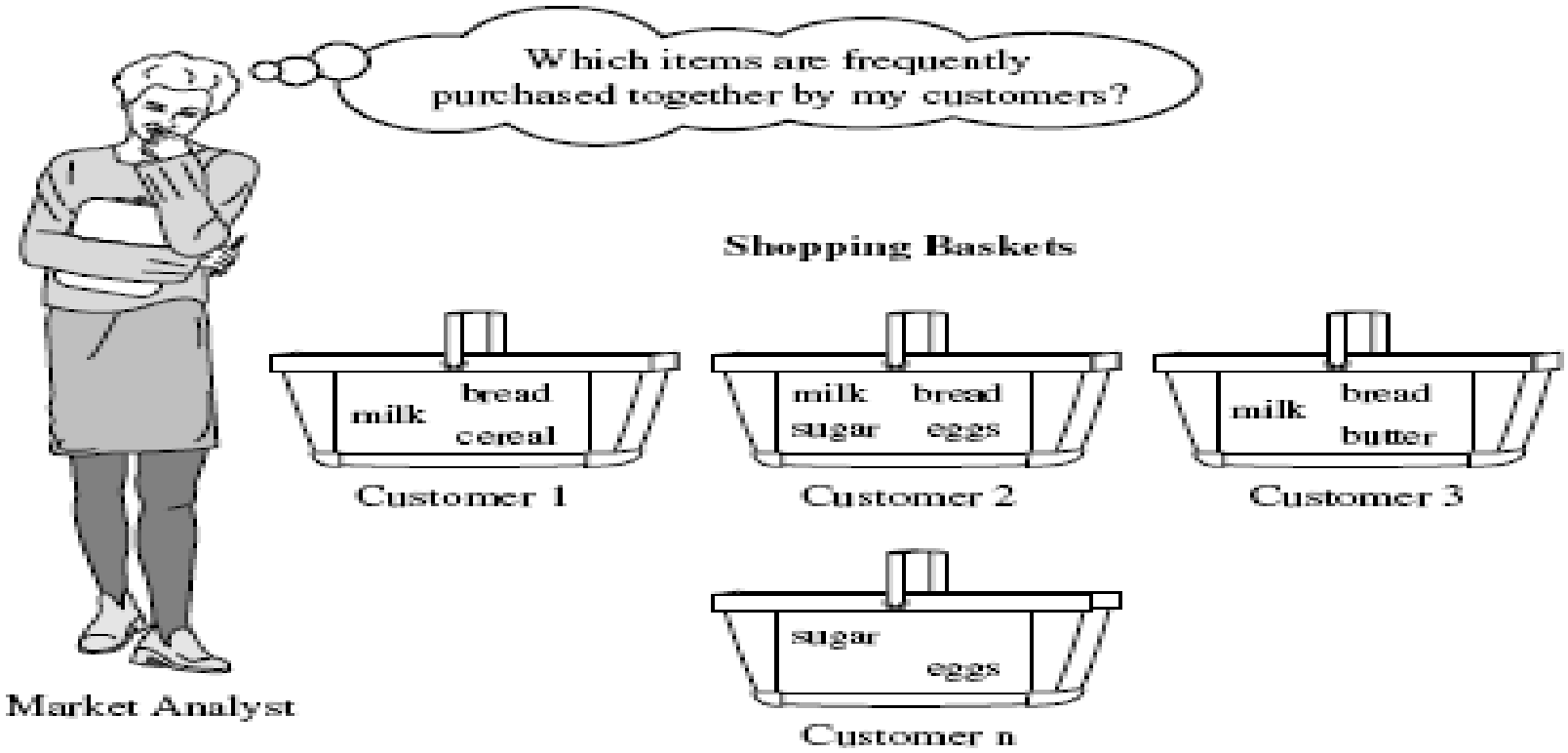
Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

support = support($\{A\}$ $\{C\}$) = 50%

confidence = support($\{A\}$ $\{C\}$)/support($\{A\}$) = 66.6%

hợp



$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

hợp

- Khai phá luật kết hợp:
  - Tìm tất cả mẫu phổ biến, kết hợp, tương quan, hoặc cấu trú nhan-quả trong tập các mục hoặc đối tượng trong CSDL quan hệ hoặc các kho chứa thông tin khác.
  - Mẫu phổ biến (Frequent pattern): là mẫu (tập mục, dãy mục…) mà xuất hiện phổ biến trong 1 CSDL [AIS93]
- Động lực: tìm mẫu chính quy (regularities pattern) trong DL
  - Các mặt hàng nào được mua cùng nhau? — Bia và bỉm (diapers)?!
  - Mặt hàng nào sẽ được mua sau khi mua một PC ?
  - Kiểu DNA nào nhạy cảm với thuộc mới này?
  - Có khả năng tự động phân lớp Web hay không ?

hợp là một bài toán bản chất của khai phá DL

- Nền tảng của nhiều bài toán KPDL bản chất
  - Kết hợp, tương quan, nhân quả
  - Mẫu tuần tự, kết hợp thời gian hoặc vòng, chu kỳ bộ phận, kết hợp không gian và đa phương tiện
  - Phân lớp kết hợp, phân tích cụm, khối tảng băng, tích tụ (nén dữ liệu ngữ nghĩa)
- Ứng dụng rộng rãi
  - Ví dụ: Phân tích DL bóng rổ, tiếp thị chéo (cross-marketing), thiết kế catalog, phân tích chiến dịch bán hàng
  - Phân tích Web log (click stream), Phân tích chuỗi DNA v.v.

Apriori: Một tiếp cận sinh ứng viên và kiểm tra

- Khái quát: Khai phá luật kết hợp gồm hai bước:
  - Tìm mọi tập mục phổ biến: theo min-sup
  - Sinh luật mạnh từ tập mục phổ biến
- Mọi tập con của tập mục phổ biến cũng là tập mục phổ biến
  - Nếu {*bia*, *bỉm*, *hạnh nhân*} là phổ biến thì {*bia*, *bỉm*} cũng vậy: Mọi giao dịch chứa {*bia*, *bỉm*, *hạnh nhân*} cũng chứa {*bia*, *bỉm*}.
- Nguyên lý tỉa Apriori: Với tập mục không phổ biến thì không cần phải sinh ra/kiểm tra mọi tập bao nó!
- Phương pháp:
  - Sinh các tập mục ứng viên dài (k+1) từ các tập mục phổ biến có độ dài k (Độ dài tập mục là số phần tử của nó),
  - Kiểm tra các tập ứng viên theo CSDL
- Các nghiên cứu hiệu năng chứng tỏ tính hiệu quả và khả năng mở rộng của thuật toán
- Agrawal & Srikant 1994, Mannila, và cộng sự 1994

Thuật toán Apriori

Trên cơ sở tính chất (nguyên lý tỉa) Apriori, thuật toán hoạt động theo quy tắc quy hoạch động

- Từ các tập $F_i = \{c_i | c_i$ tập phổ biến, $|c_i| = i\}$ gồm mọi tập mục phổ biến có độ dài i với 1 i k,
- đi tìm tập F_{k+1} gồm mọi tập mục phổ biến có độ dài k+1.

Trong thuật toán, các tên mục $i_1, i_2, \ldots i_n$ ($n = |I|$) *được sắp xếp theo một thứ tự cố định* (thường được đánh chỉ số 1, 2, ..., n).

Thuật toán Apriori

Thuật toán Apriori [WKQ08]:

Input: - Cơ sở dữ liệu giao dịch D = {t| t giao dịch}

- Độ hỗ trợ tối thiểu minsup > 0

Output: - Tập hợp tất cả các tập phổ biến.

0: mincount = minsup * |D|;
1. F_1 = {các tập phổ biến có độ dài 1}
2. for (k=1; $F_k \neq \emptyset$; k++) do begin
3. C_{k+1} = apriori-gen (F_k); // sinh mọi ứng viên độ dài k+1
4. for t ∈ D do begin
5. C_t = {c ∈ C_{k+1} | c ⊆ t}; //mọi ứng viên chứa trong t
6. for c ∈ C_t do
7. c.count ++;
8. end
9. F_{k+1} = {c ∈ C_{k+1} | c.count ≥ mincount} ;
10. end
11. Answer $\cup_k F_k$;

Thuật toán Apriori: Thủ tục con Apriori-gen

Trong mỗi bước k, thuật toán Apriori đều phải duyệt CSDL D.

Khởi động, duyệt D để có được F_1.

Các bước k sau đó, duyệt D để tính số lượng giao dịch t thoả từng ứng viên c của C_{k+1}: mỗi giao dịch t chỉ xem xét một lần cho mọi ứng viên c thuộc C_{k+1}.

Thủ tục con Apriori-gen sinh tập phổ biến: *tư tưởng*

Bước nối: Sinh các tập mục R_{k+1} là ứng viên tập phổ biến có độ dài k+1 bằng cách kết hợp hai tập phổ biến P_k và Q_k có độ dài k và trùng nhau ở k-1 mục đầu tiên:

$$R_{k+1} = P_k \cup Q_k = \{i_1, i_2, ..., i_{k-1}, i_k, i_{k'}\} \text{ với}$$

$$P_k = \{i_1, i_2, ..., i_{k-1}, i_k\} \text{ và } Q_k = \{i_1, i_2, ..., i_{k-1}, i_{k'}\}$$

trong đó $i_1 \le i_2 \le ... \le i_{k-1} \le i_k \le i_{k'}$.

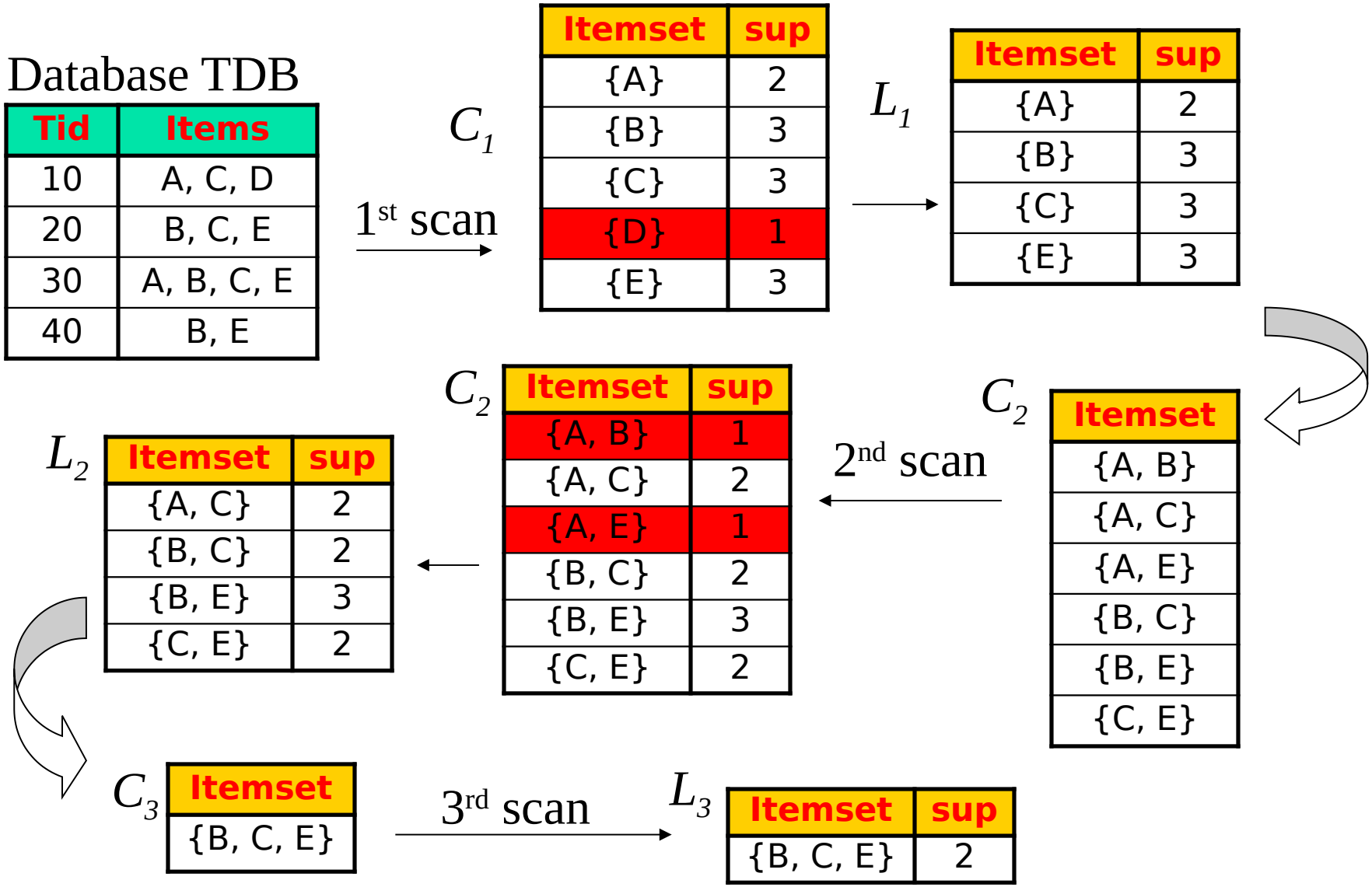
Bước tỉa: Giữ lại tất cả các R_{k+1} thỏa tính chất Apriori ($\forall X \subseteq R_{k+1}$ và $|X|=k \Rightarrow X \in F_k$), nghĩa là đã loại (tia) bớt đi mọi ứng viên R_{k+1} không đáp ứng tính chất này.

Thủ tục con Apriori-gen

(1) for mọi tập mục phổ biến $l_1 \in L_k$
(2) for mọi tập mục phổ biến $l_2 \in L_k$
(3) if $(l_1[1]=l_2[1]) \wedge (l_1[2]=l_2[2]) \wedge \ldots \wedge (l_1[k-1]=l_2[k-1]) \wedge (l_1[k]<l_2[k])$
then {
$c = l_1 \Leftrightarrow l_2$; // join step: generate candidates
// c = $\{l_1[1], l_1[2], \ldots, l_1[k-1], l_1[k], l_2[k]\}$
(5) if has_infrequent_subset(c, L_k) then
(6) delete c; // bước tỉa: bỏ ứng viên không đúng
else add c to C_{k+1};
(8) }
(9) return Ck;

procedure has_infrequent_subset(c: tập ứng viên độ dài $k+1$;
L_k: tập các tập mục phổ biến độ dài k); // tri thức đã có
(1) for mỗi tập con s độ dài k *của* c
(2) if $s \notin L_k$ then
(3) return TRUE;
(4) return FALSE;

Một ví dụ thuật toán Apriori (s=0.5)

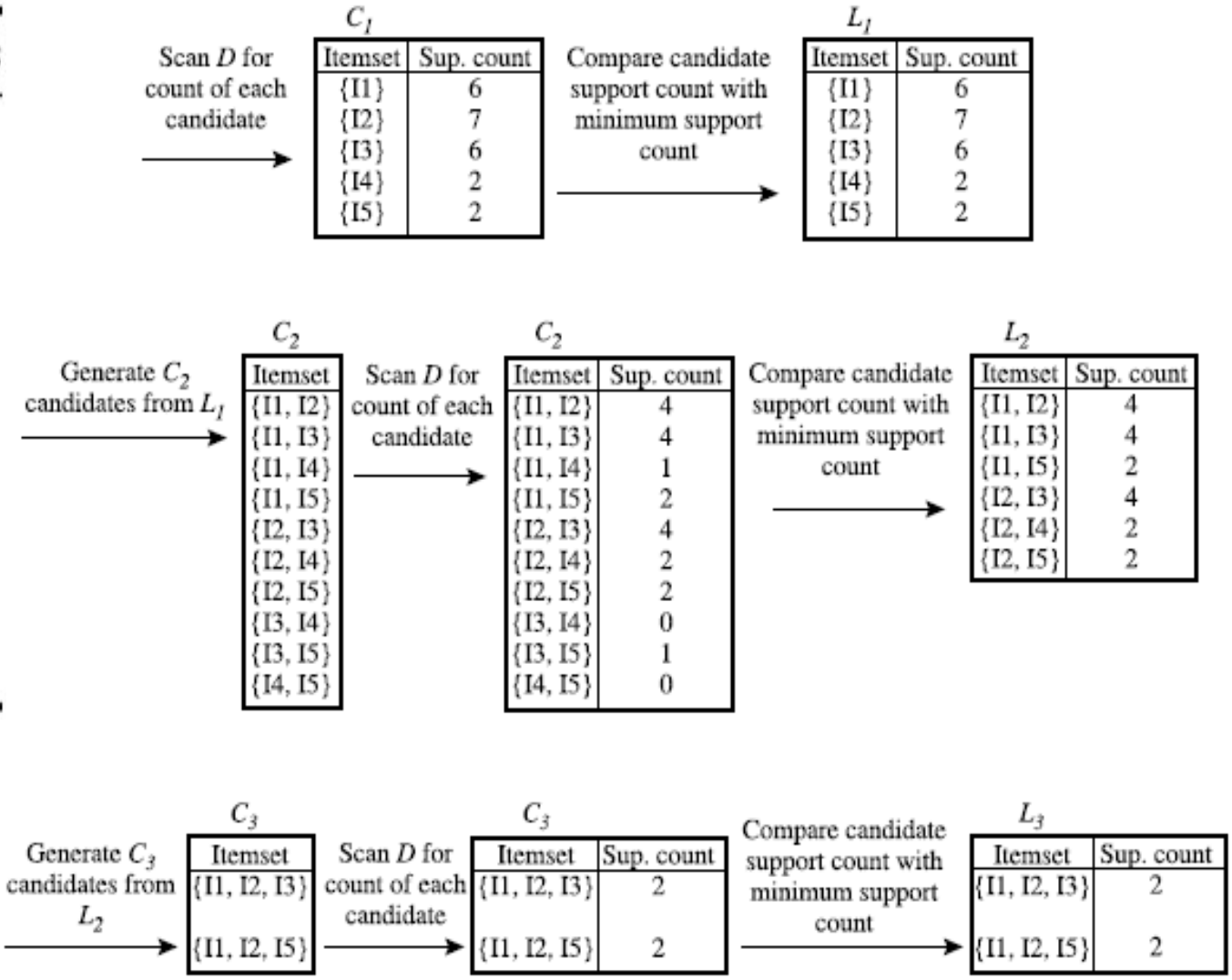


Chi tiết quan trọng của Apriori

- Cách thức sinh các ứng viên:
  - Bước 1: Tự kết nối L_k
  - Step 2: Cắt tỉa
- Cách thức đếm hỗ trợ cho mỗi ứng viên.
- Ví dụ thủ tục con sinh ứng viên
  - L_3={*abc, abd, acd, ace, bcd*}
  - Tự kết nối: L_3*L_3
    - *abcd* từ *abc* và *abd*
    - *acde* từ *acd* và *ace*
  - Tỉa:
    - *acde* là bỏ đi vì *ade* không thuộc L_3
  - C_4={*abcd*}

)

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Sinh luật kết hợp

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Việc sinh luật kết hợp gồm hai bước

- Với mỗi tập phổ biến W tìm được hãy sinh ra mọi tập con thực sự X khác rỗng của nó.
- Với mỗi tập phố biến W và tập con X khác rỗng thực sự của nó: sinh luật X (W – X) nếu P(W-X| X) c.

Như ví dụ đã nêu có L3 = {{I1, I2, I3}, {I1, I2, I5}}

Với độ tin cậy tối thiểu 70%, xét tập mục phổ biến

$I1 \wedge I2 \Rightarrow I5, \quad confidence = 2/4 = 50\%$

$I1 \wedge I5 \Rightarrow I2, \quad confidence = 2/2 = 100\%$

$I2 \wedge I5 \Rightarrow I1, \quad confidence = 2/2 = 100\%$

$I1 \Rightarrow I2 \wedge I5, \quad confidence = 2/6 = 33\%$

$I2 \Rightarrow I1 \wedge I5, \quad confidence = 2/7 = 29\%$

$I5 \Rightarrow I1 \wedge I2, \quad confidence = 2/2 = 100\%$

Cách thức tính độ hỗ trợ của ứng viên

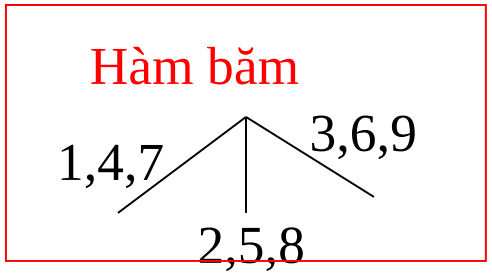
```
4.      for t ∈ D do begin
5.          C_t = {c ∈ C_{k+1} | c ⊆ t}; //mọi ứng viên chứa trong t
6.          for c ∈ C_t do
7.              c.count ++;
8.      end
```

- Tính độ hỗ trợ ứng viên (lệnh 4-8):
  - Số lượng ứng viên là rất lớn
  - Một giao dịch chứa nhiều ứng viên
- Phương pháp: sử dụng cây băm ứng viên
  - Tập các tập mục ứng viên được chứa trong một cây-băm (*hash-tree*)
  - *Lá* của cây băm chứa một danh sách ứng viên và bộ đếm (độ hỗ trợ hiện thời của ứng viên đó)
  - *Nút trong* chứa bảng băm: theo tập các mục I
  - *Hàm tập con*: tìm ứng viên trong tập ứng viên

Tính độ hỗ trợ của ứng viên

- Tập các ứng viên C_k được lưu trữ trong một cây-băm.
  - Gốc cây băm ở độ sâu 1. Lá chứa một danh sách tập mục thuộc C_k.
  - Nút trong chứa một bảng băm (chẳng hạn **mod N**): mỗi ô trỏ tới một nút con (Nút ở độ sâu d trỏ tới các nút ở độ sâu d+1).
  - Khi khởi tạo: gôc là một nút **lá** với danh sách rỗng.
- Xây dựng cây băm - thêm một tập mục c:
  - Bắt đầu từ gốc đi xuống theo cây cho đến khi gặp một lá.
  - Tại nút độ sâu **d**: đi theo nhánh nào: áp dụng hàm băm tới mục thứ **d** của tập mục này.
  - Khi số lượng tập mục tại một lá vượt ngưỡng quy định, lá được chuyển thành nút trong, phân danh sách các tập mục như hàm băm.
- Tính độ hỗ trợ: tìm mọi ứng viên thuộc giao dịch t:
  - Nếu ở nút gốc: băm vào mỗi mục trong t.
  - Nếu ở một lá: tìm các tập mục ở lá này thuộc t và bổ sung chỉ dẫn các tập mục này tới tập trả lời.
  - Nếu ở nút trong và đạt được nó bằng cách băm mục i, trên từng mục đứng sau i trong t và áp dụng đệ quy thủ tục này sang nút trong thùng tương ứng.

Ví dụ: Xây dựng cây băm các ứng viên



Có tập các ứng viên độ dài 3 là 124, 125, 136, 145, 159, 234, 345, 356, 357, 367, 368, 457, 458, 567, 689

1, 4, 7 đi sang trái; 2, 5, 8 dừng ở giữa; 3, 6, 9 đi sang phải

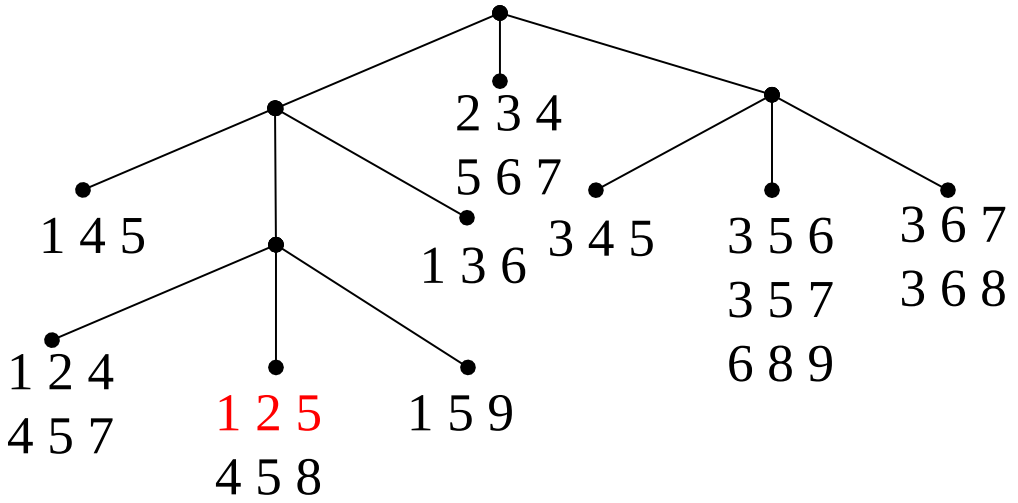
124
125
136

Thêm 145 vượt qua ngưỡng, đưa 4 tập này sang nút con trái. Vì 4 tập này đều vượt qua ngưỡng nên tách thành 145; 124, 125; 136

Thêm 159 bổ sung vào nút giữa cây con trái

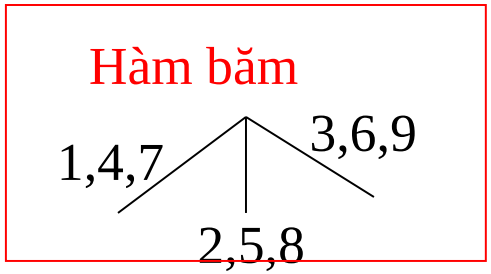
Thêm 234 bổ sung vào nút giữa cây mẹ

Thêm 345 bổ sung vào nút phải cây mẹ; sau đó tách cây con phải 345; 356, 357; 367, 368



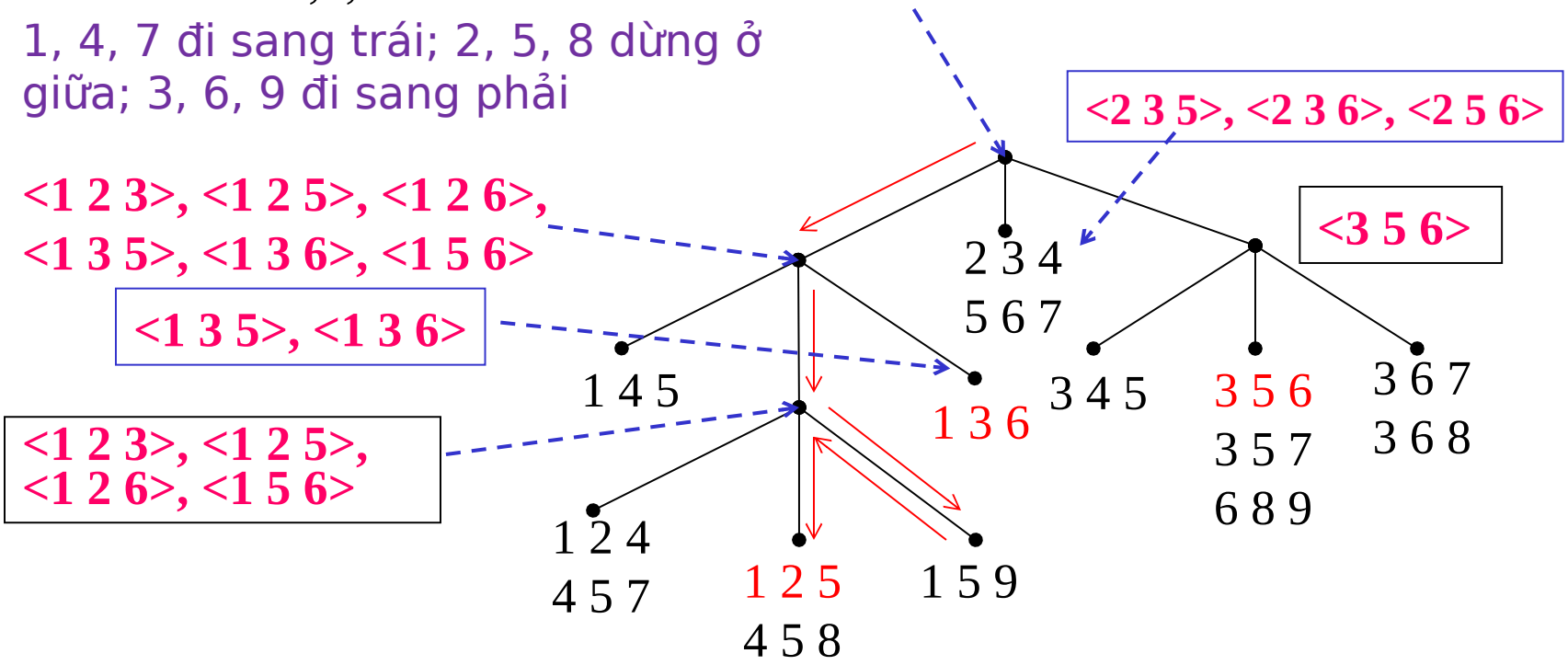
Mối quan hệ gid giữa tập ứng viên hàm băm, cỡ ô chứa, chiều cao cây?

Ví dụ: Tính hỗ trợ các ứng viên



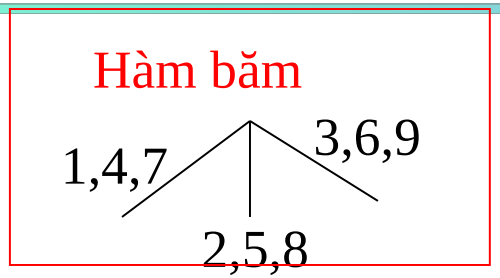
Giao dịch t=1 2 3 5 6 sinh mọi tập mục độ dài 3 : <1 2 3>, <1 2 5>, <1 2 6>, <1 3 5>, <1 3 6>, <1 5 6>, <2 3 5>, <2 3 6>, <2 5 6>, <3 5 6>

1, 4, 7 đi sang trái; 2, 5, 8 dừng ở giữa; 3, 6, 9 đi sang phải



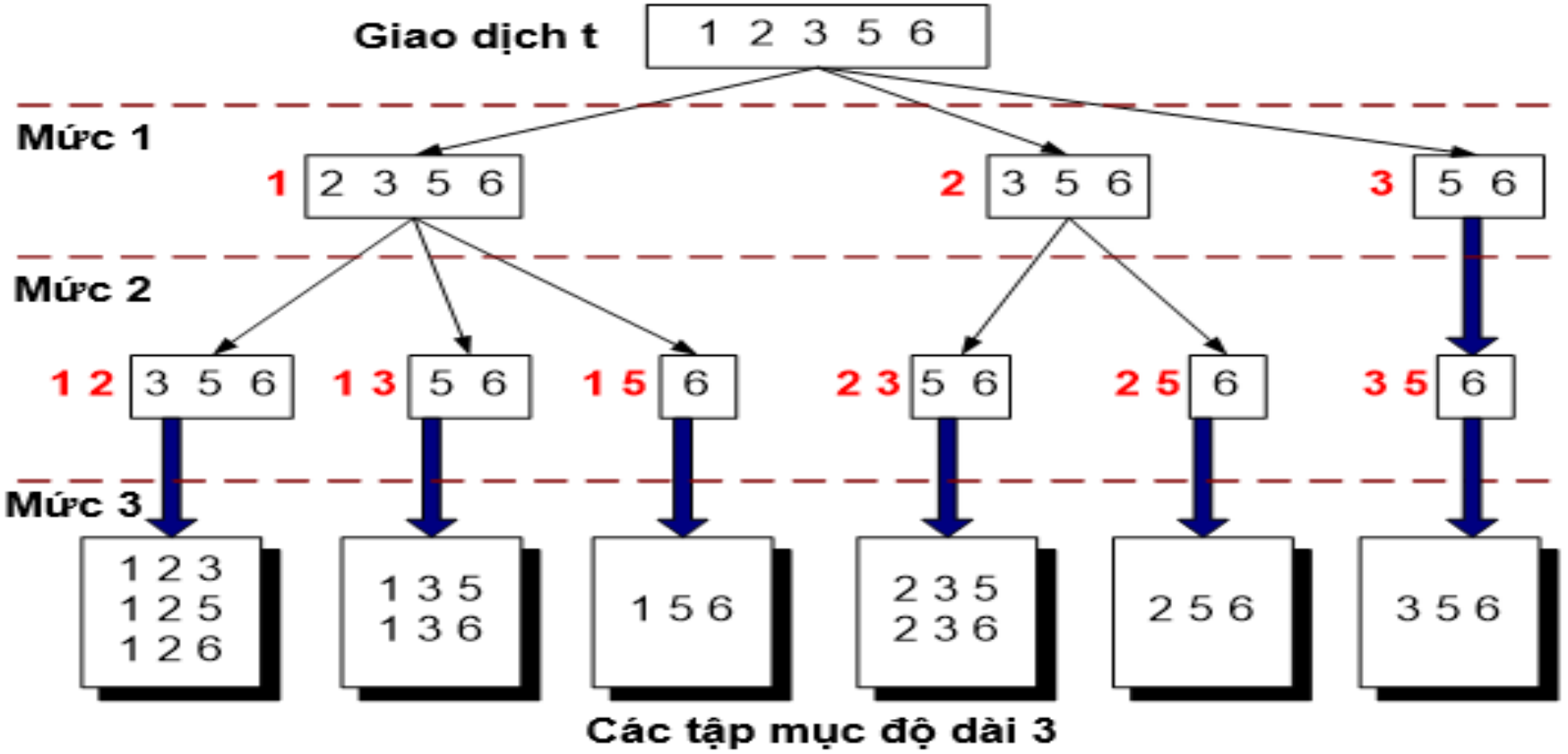
Bộ đếm của ba ứng viên **125**, **136**, **356** được tăng thêm 1 với giao dịch t=12356

Ví dụ: Tính hỗ trợ các ứng viên



1, 4, 7 đi sang trái; 2, 5, 8 dừng ở giữa; 3, 6, 9 đi sang phải

Giao dịch t=1 2 3 5 6 sinh mọi tập mục độ dài 3 : các ứng viên: không bắt đầu bằng mục 5, 6

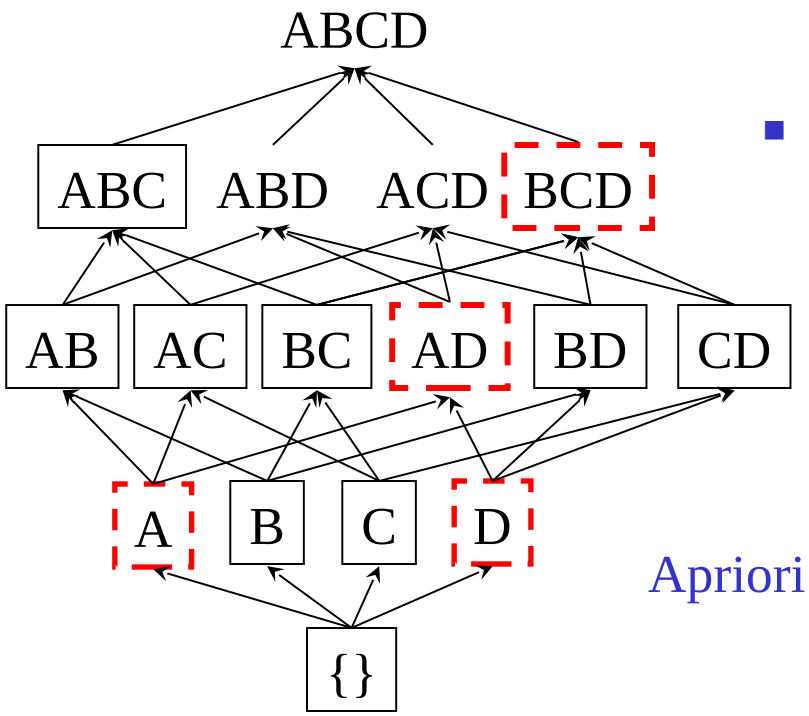


Bộ đếm của ba ứng viên **125**, **136**, **356** được tăng thêm 1 với giao dịch t=12356

Thách thức khai phá mẫu phổ biến 26/10

- Thách thức
  - Duyệt nhiều lần CSDL giao dịch
  - Lượng các ứng viên rất lớn
  - Tẻ nhạt việc tính toán độ hỗ trợ
- Cải tiến Apriori: tư tưởng chung
  - Giảm số lần duyệt CSDL giao dịch
  - Rút gọn số lượng các ứng viên
  - Giảm nhẹ tính độ hỗ trợ của các ứng viên

DIC (Đếm tập mục động): Rút số lượng duyệt CSDL

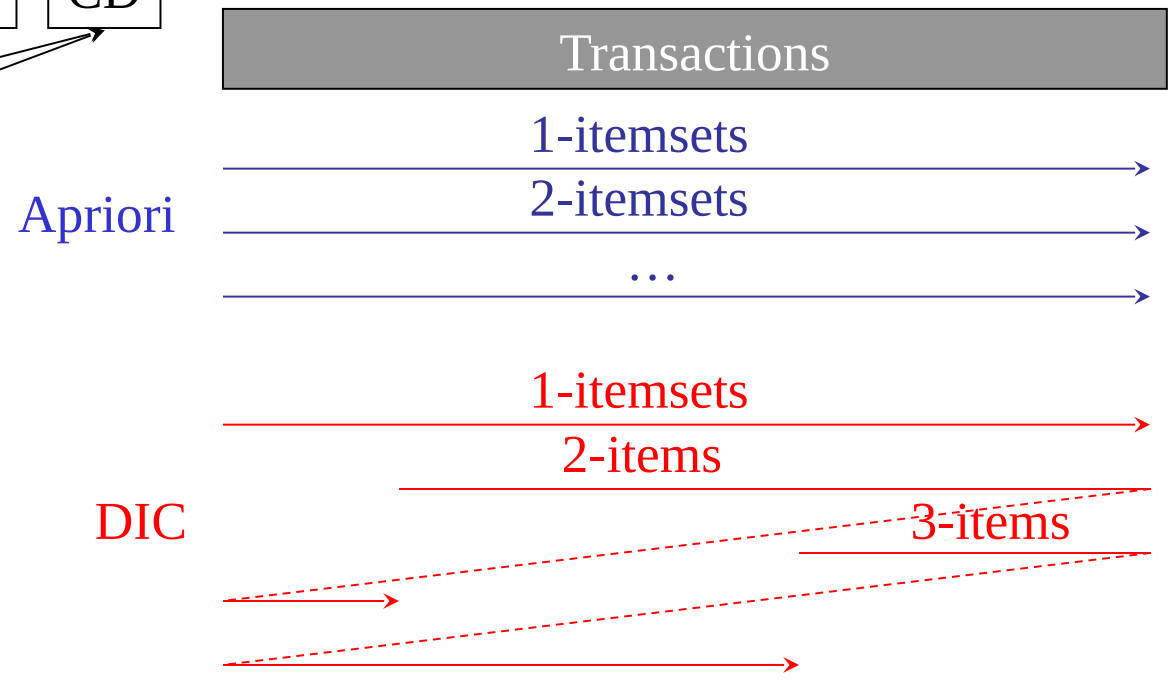


Itemset lattice

S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD'97*

- Xây dựng dàn tập mục
- Khi A và D được xác định là phổ biến thì việc tính toán cho AD được bắt đầu
- Khi mọi tập con độ dài 2 của BCD được xác định là phổ biến: việc tính toán cho BCD được bắt đầu.

Chia D thành k đoạn có M giao dịch.
Mỗi khi vượt qua k*M bỏ đi mức cũ.



Giải pháp Phân hoạch (Partition): Duyệt CSDL chỉ hai lần

- Mọi tập mục là phổ biến tiềm năng trong CSDL bắt buộc phải phổ biến ít nhất một vùng của DB
  - Scan 1: Phân chia CSDL và tìm các mẫu cục bộ
  - Scan 2: Hợp nhất các mẫu phổ biến tổng thể
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

Ví dụ về mẫu phổ biến

- Chọn một mẫu của CSDL gốc, khai phá mẫu phổ biến nội bộ mẫu khi dùng Apriori
- Duyệt CSDL một lần để kiểm tra các tập mục phổ biến tìm thấy trong ví dụ, chỉ có các bao (*borders*) *đóng* của các mẫu phổ biến được kiểm tra
  - Ví dụ: kiểm tra *abcd* thay cho *ab, ac, …, v.v.*
- Duyệt CSDL một lần nữa để tìm các mẫu phổ biến bị mất (bỏ qua)

H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

DHP: Rút gọn số lượng các ứng viên

- Một *k*-tập mục mà bộ đếm trong lô băm tương ứng dưới ngưỡng (=3) thì không thể là tập mục phổ biến
  - Ứng viên: a, b, c, d, e
  - Điểm vào băm: {ab, ad, ae} {bd, be, de} ...
  - 1-tập mục phổ biến: a, b, d, e
  - ab không là một ứng viên 2-tập mục nếu tổng bộ đếm trong lô băm {ab, ad, ae} là dưới ngưỡng hỗ trợ. *Mọi giao dịch có chứa a đều ở lô băm* {ab, ad, ae}.

J. Park, M. Chen, and P. Yu.
An effective hash-based algorithm for mining association rules
In *SIGMOD'95*

Eclat/MaxEclat và VIPER: Thăm dò dạng dữ liệu theo chiều ngang

- Dùng danh sách tid của giáo dịch trong một tập mục
- Nén danh sách tid
  - Tập mục A: t1, t2, t3, sup(A)=3
  - Tập mục B: t2, t3, t4, sup(B)=3
  - Tập mục AB: t2, t3, sup(AB)=2
- Thao tác chính: lấy giao của các danh sách tid
- M. Zaki et al. New algorithms for fast discovery of association rules. In KDD'97
- P. Shenoy et al. Turbo-charging vertical mining of large databases. In SIGMOD'00

biến

- Duyệt CSDL nhiều là tốn kém
- KP mẫu dài cần nhiều bước để duyệt và sinh nhiều ứng viên
  - Để tìm các tập mục phổ biến $i_1 i_2 \ldots i_{100}$
    - # duyệt: 100
    - # ứng viên: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1$ $= 1.27*10^{30}$!
- Thắt cổ chai: sinh ứng viên và kiểm tra
- Tránh sinh ứng viên?

KP mẫu phổ biến không cần sinh ƯV

- Dùng các mục phổ biến để tăng độ dài mẫu từ các mẫu ngắn hơn
  - “abc” là một mẫu phổ biến
  - Nhận mọi giao dịch có “abc”: DB|abc (DB đã luôn có abc: *“có điều kiện”*)
  - “d” là một mục phổ biến trong DB|abc abcd là một mẫu phổ biến

Xây dựng cây FP

Cấu trúc cây FP-tree được định nghĩa như sau:

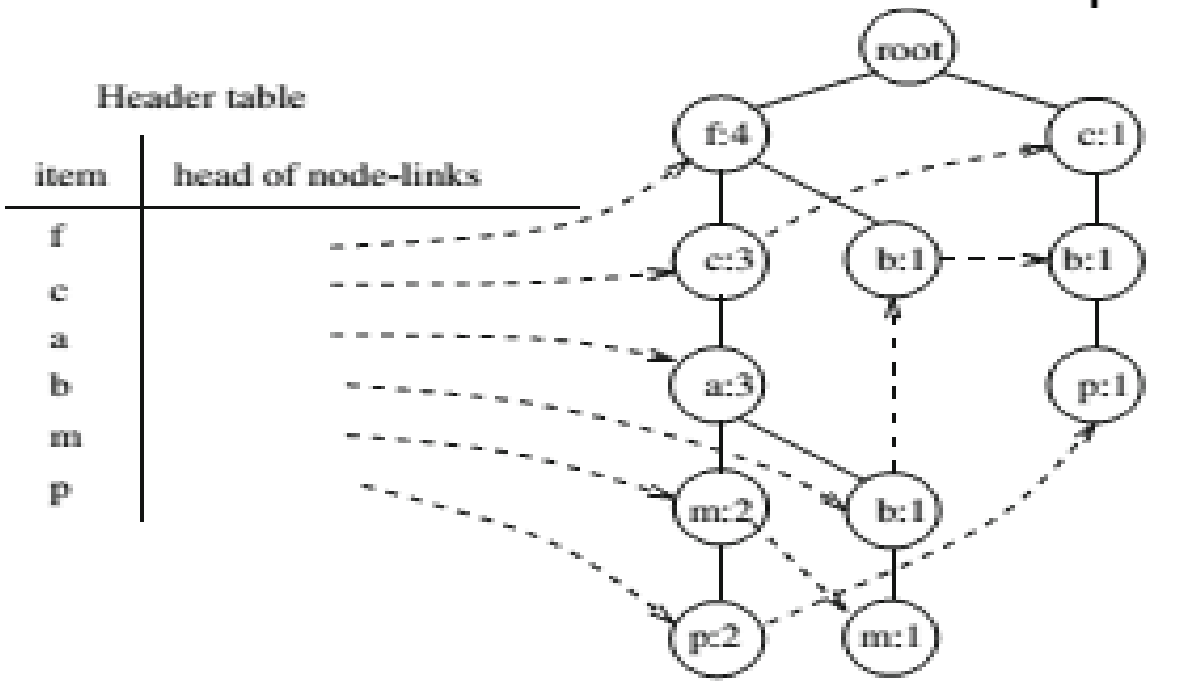
- Gốc của cây nhãn *null*, các đường đi trên cây biểu diễn một tập các tiền tố của một tập mục
- Mỗi nút trong cây có chứa 3 thành phần: tên mục, số lần xuất hiện (*count*), con trỏ. Trong đó, **count** là số lượng xuất hiện của nhánh con (từ **NULL** đến nút này) trong các giao dịch, còn **con trỏ** liên kết (mũi tên nét đứt) đến nút có cùng tên tiếp theo của nó.
- Mỗi dòng trong bảng header chứa 2 trường: tên mục và nút rỗng trỏ tới đến nút đầu tiên cùng một mục trên cây FP

Xây dựng cây FP từ một CSDL giao dịch

TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

min_support = 3

F-list=f-c-a-b-m-p f:4–c:4–a:3–b:3–m:3–p:3



Xây dựng cây FP từ một CSDL giao dịch

1. Duyệt CSDL lần đầu tiên, tìm các 1-tập mục phổ biến (mẫu mục đơn).

   Loại các mục có độ hỗ trợ < minsup.

   Xếp các mục phổ biến theo thứ tự giảm dần về độ hỗ trợ (bậc): Tạo f-list (*).

   Tạo cây FP với gốc nhãn {*root*}

2. Duyệt CSDL lần thứ hai

   Với mỗi giao dịch t:

   Xâu các mục phổ biến theo thứ tự (*) và biểu diễn dưới dạng [p|P] với p là mục đầu tiên, còn P là xâu mục còn lại;

   Gọi insert_tree ([p|P]), T)

3. Tìm tập phổ biến trên cây FP

vào cây

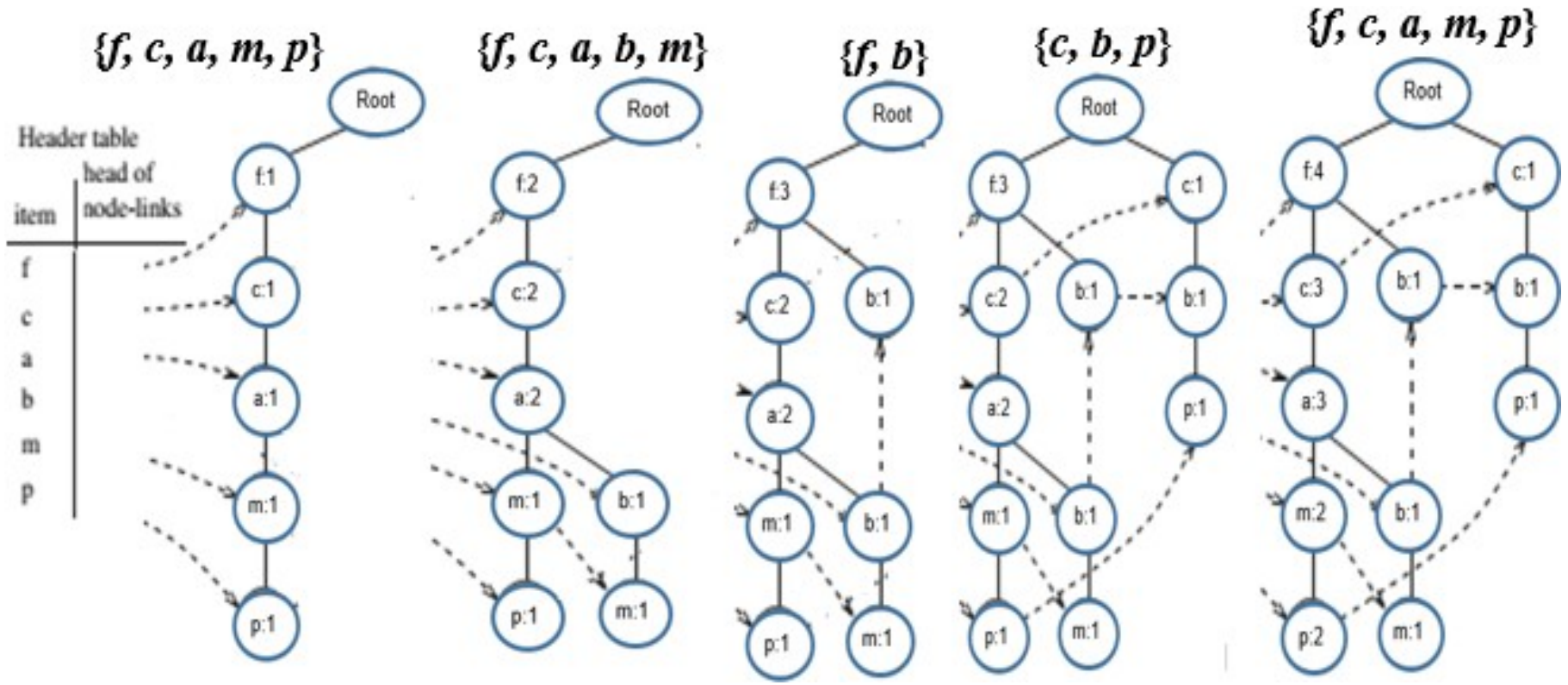
```
Procedure insert_tree(string[p|P], tree có gốc T)

If T có nút con N mà N.itemname=p.itemname
Then N.count++
    else
        Tạo một nút mới N;
        N.itemname:=p.itemname;
        N.count:=1
        Thay đổi nút liên kết cho p bao gồm N;
    End if
If p # rỗng
   insert_tree([p1|P1] ,N);                    // P=p1P1
```

Xây dựng cây FP

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3



Lợi ích của cấu trúc FP-tree

- Tính đầy đủ
  - Duy trì tính đầy đủ thông tin để khai phá mẫu phổ biến
  - Không phá vỡ mẫu dài bới bất kỳ giao dich
- Tính cô đọng
  - Giảm các thông tin không liên quan: mục không phổ biến bỏ đi
  - Sắp mục theo tần số giảm: xuất hiện càng nhiều thì cành hiệu quả
  - Không lớn hơn so với CSDL thông thường

tree

Input: A database *DB*, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold ξ.

Output: The complete set of frequent patterns.

Procedure *FP-growth*(*Tree*, α)

```
{
(1)  if Tree contains a single prefix path     // Mining single prefix-path FP-tree
(2)  then {
(3)      let P be the single prefix-path part of Tree;
(4)      let Q be the multipath part with the top branching node replaced by a null root;
(5)      for each combination (denoted as β) of the nodes in the path P do
(6)          generate pattern β ∪ α with support = minimum support of nodes in β;
(7)      let freq_pattern_set(P) be the set of patterns so generated;       }
(8)  else let Q be Tree;
(9)  for each item a_i in Q do {                          // Mining multipath FP-tree
(10)     generate pattern β = a_i ∪ α with support = a_i.support;
(11)     construct β's conditional pattern-base and then β's conditional FP-tree Tree_β;
(12)     if Tree_β ≠ ∅
(13)     then call FP-growth(Tree_β, β);
(14)     let freq_pattern_set(Q) be the set of patterns so generated;       }
(15) return(freq_pattern_set(P) ∪ freq_pattern_set(Q) ∪ (freq_pattern_set(P)
         × freq_pattern_set(Q)))
}
```

Mẫu cực đại (Max-patterns)

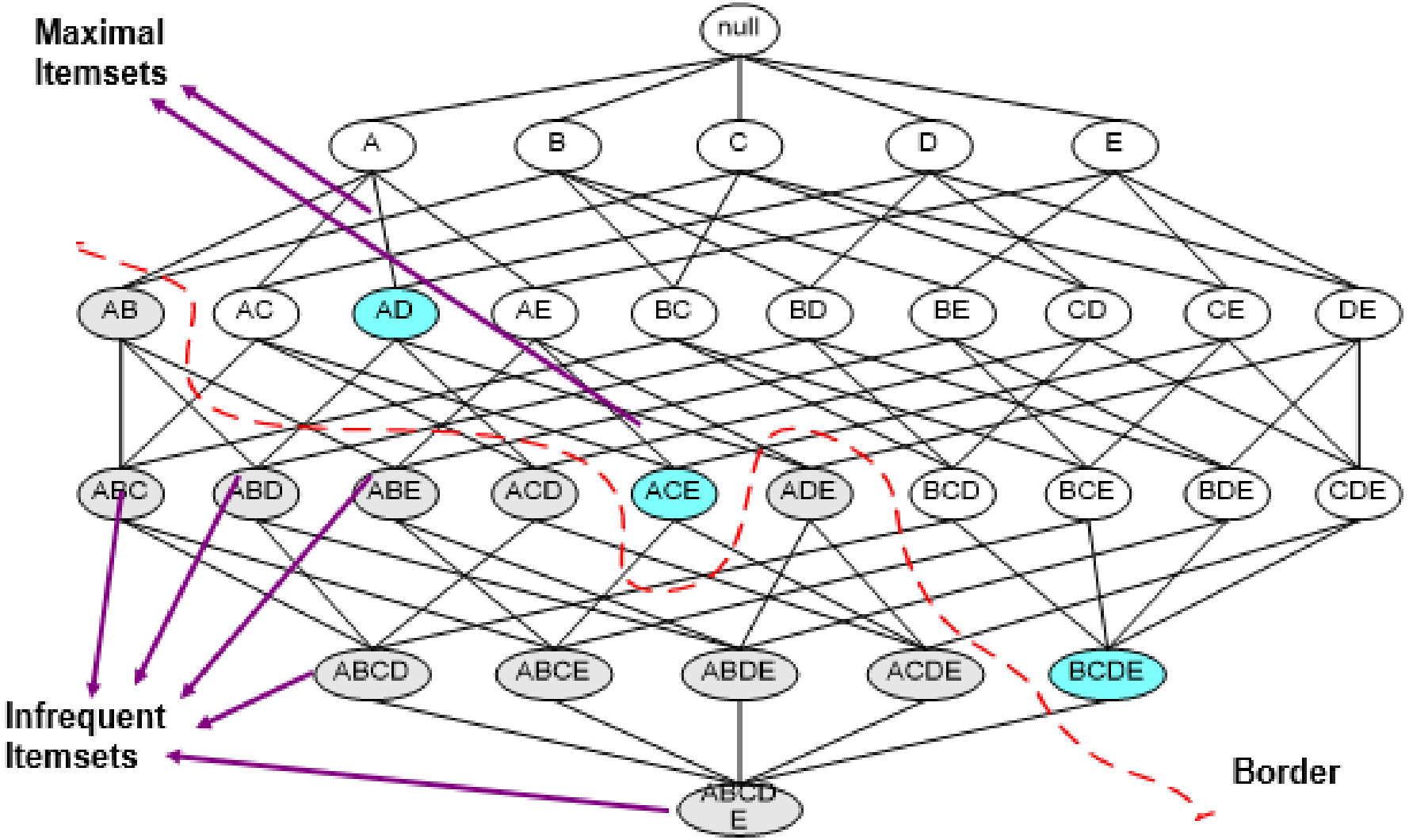
- Mẫu phổ biến $\{a_1, ..., a_{100}\} \Rightarrow \binom{100}{1} + \binom{100}{2} + ... + \binom{100}{100} = 2^{100}-1 = 1.27*10^{30}$ frequent sub-patterns!
- Mẫu cực đại: Mẫu phổ biến mà không là tập con thực sự của mẫu phổ biến khác
  - BCDE, ACD là mẫu cực đại
  - BCD không là mẫu cực đại

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F

Min_sup=2

Tập mục phổ biến cực đại

Tập mục cực đại (Maximal Intemset) là tập mục phổ biến không là tập con thực sự của một tập mục phổ biến khác



Tập mục đóng

- Tập mục đóng là tập mục mà không là tập con thực sự của một **tập mục có cùng độ hỗ trợ**
- X đóng: Y X $s(Y) < s(X)$

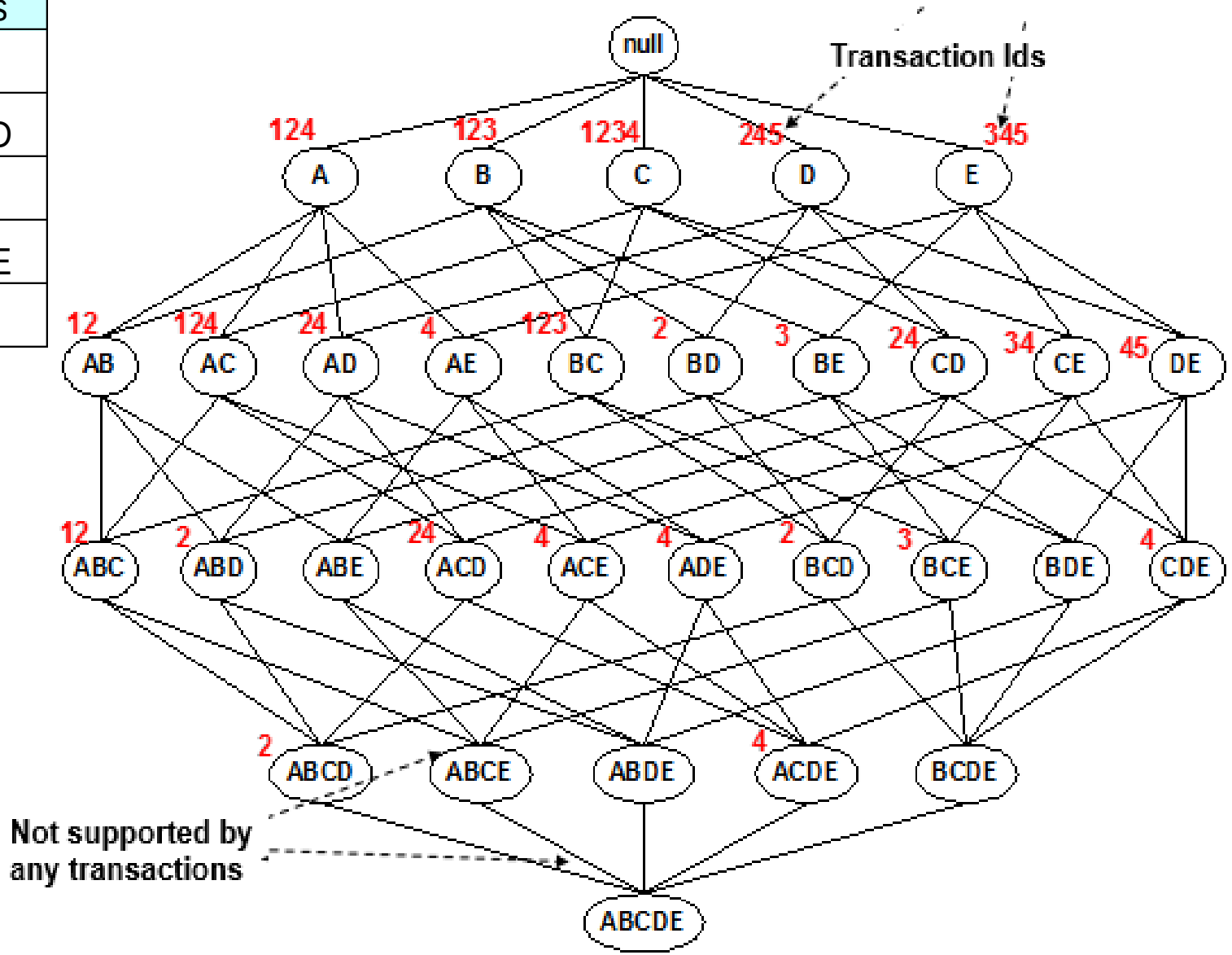
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

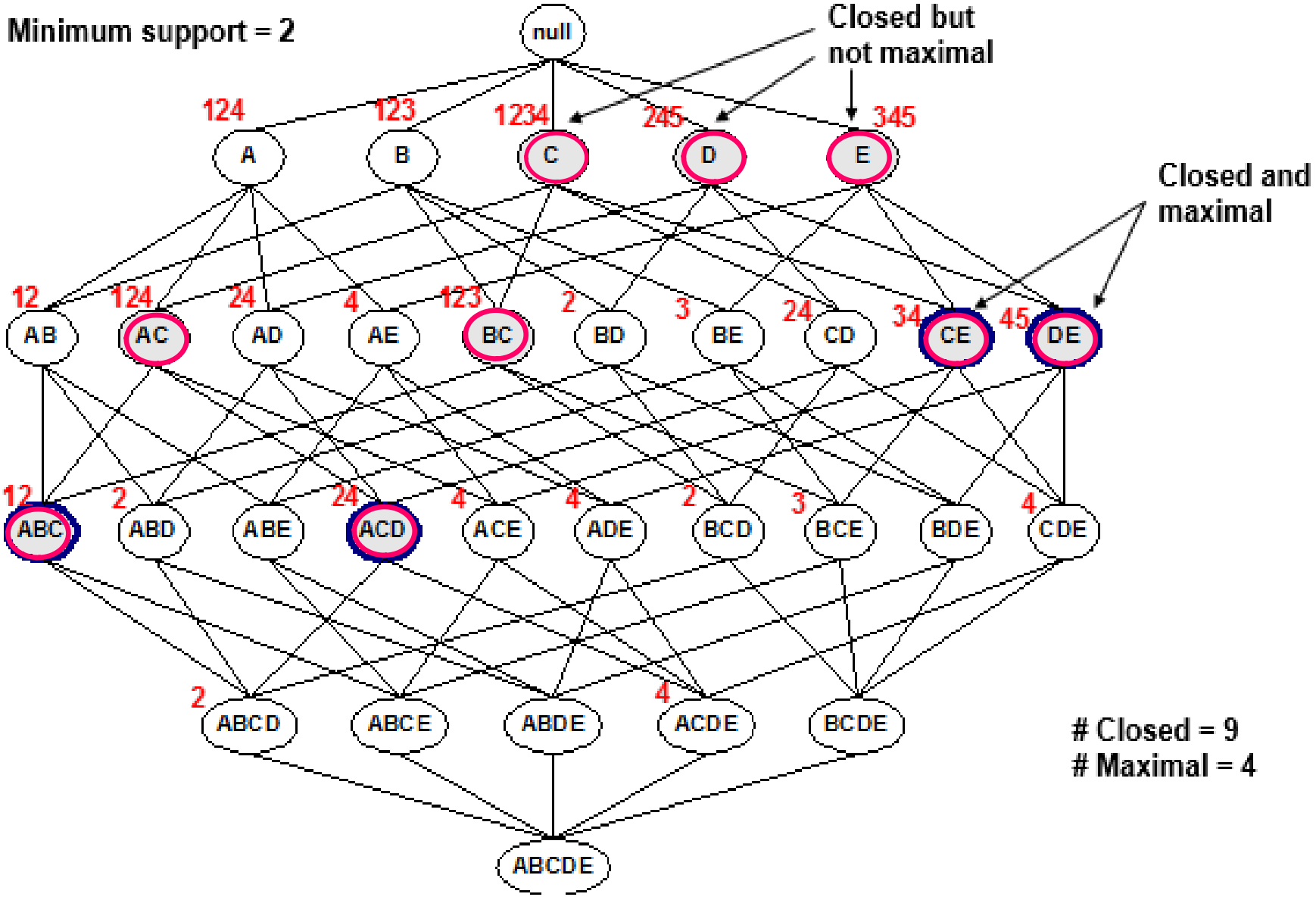
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Phân biệt tập mục cực đại với tập mục đóng

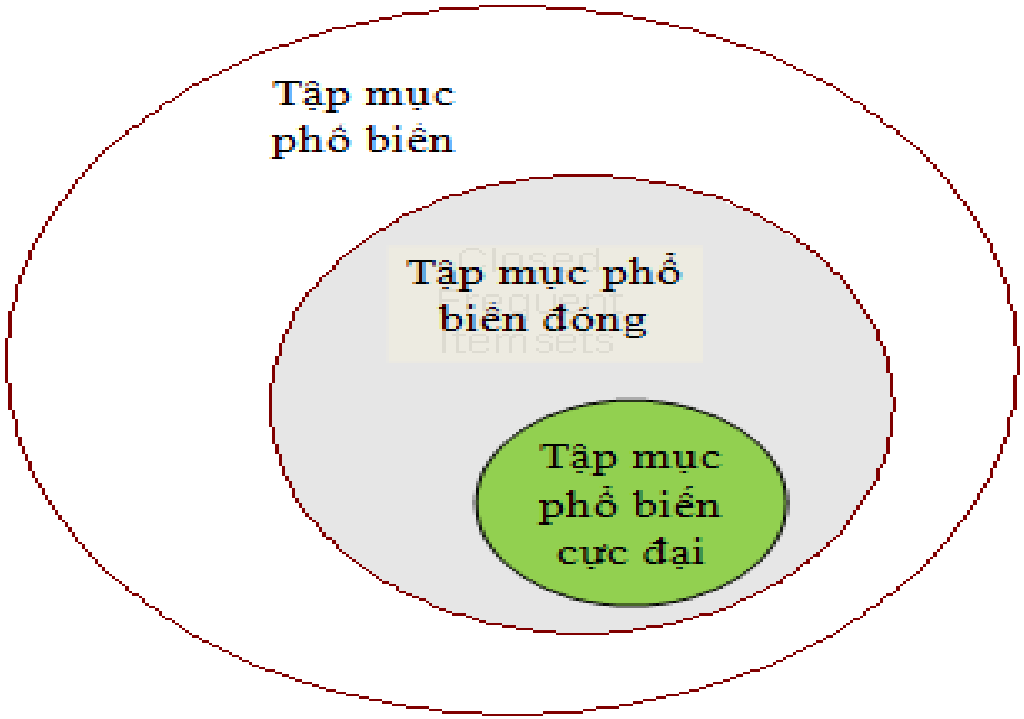
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Tập mục cực đại với tập phổ biến đóng



đóng



Tập mục cực đại với tập mục đóng

DISTINCT DATABASE ITEMS

Jane Austen	Agatha Christie	Sir Arthur Conan Doyle	Mark Twain	P. G. Wodehouse
A	C	D	T	W

DATABASE

Transcation	Items
1	ACTW
2	CDW
3	ACTW
4	ACDW
5	ACDTW
6	CDT

ALL FREQUENT ITEMSETS

MINIMUM SUPPORT = 50%

Support	Itemsets
100% (6)	C
83% (5)	W, CW
67% (4)	A, D, T, AC, AW CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW CDW, CTW, ACTW

Figure 1. *Example DB*

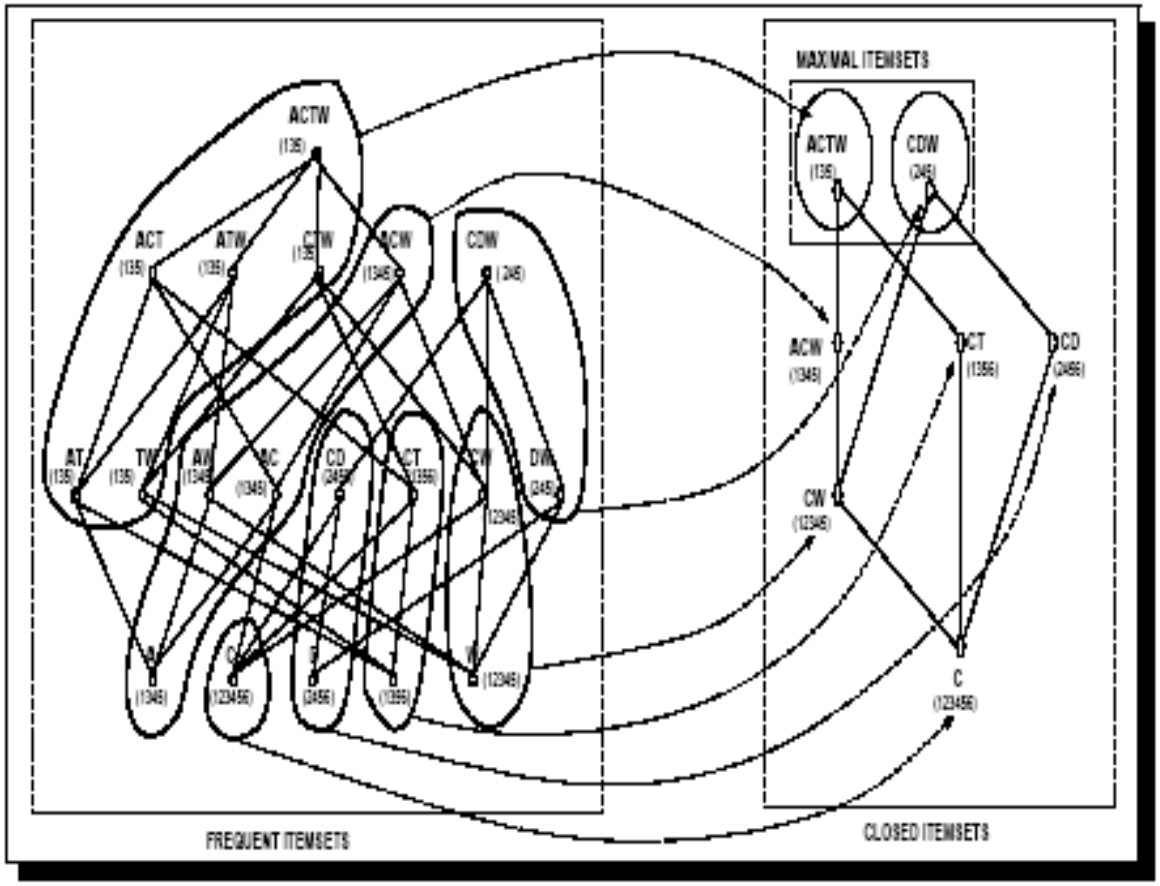


Figure 2. *Frequent, Closed and Maximal Itemsets*

R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD'98*

J. Pei, J. Han & R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00

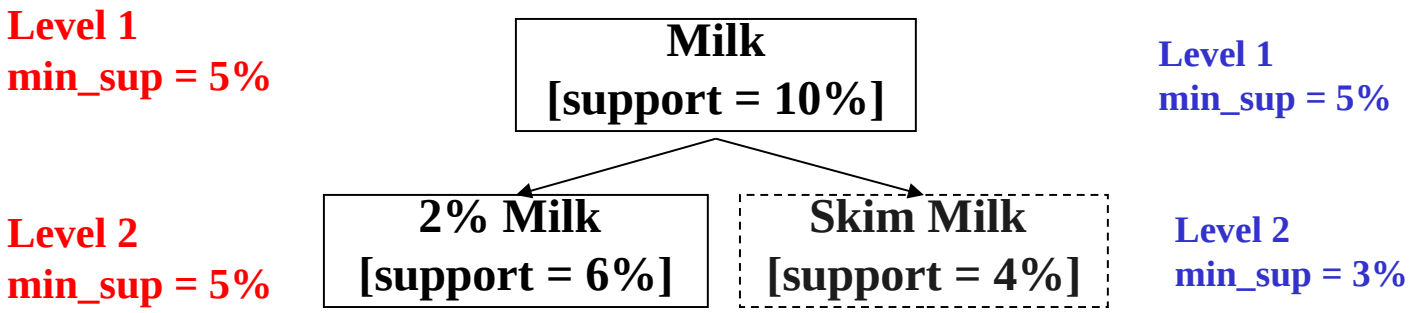
Mohammed Javeed Zaki, Ching-Jiu Hsiao: CHARM: An Efficient Algorithm for Closed Itemset Mining. SDM 2002

Luật kết hợp đa mức

- Các mục có thể phân cấp
- Đặt hỗ trợ linh hoạt: Mục cấp thấp hơn là kỳ vọng có độ hỗ trợ thấp hơn.
- CSDL giao dịch có thể được mã hóa theo chiều và mức
- Thăm dò KP đa mức chia sẻ

uniform support

reduced support



Kết hợp đa chiều

- Luật đơn chiều (viết theo dạng quan hệ (đối tượng, giá trị)):

  buys(X, “milk”) buys(X, “bread”)

- Luật đa chiều: 2 chiều / thuộc tính
  - Luật kết hợp liên chiều (không có thuộc tính lặp)

    age(X,”19-25”) occupation(X,“student”)

    buys(X,“coke”)
  - Luật KH chiều-kết hợp (lai/hybrid) (lặp thuộc tính)

    age(X,”19-25”) buys(X, “popcorn”) buys(X, “coke”)
- Thuộc tính phân lớp
  - Tìm số lượng các giá trị khả năng không được sắp
- Thuộc tính định lượng
  - Số, thứ tự ngầm định trong miền giá trị

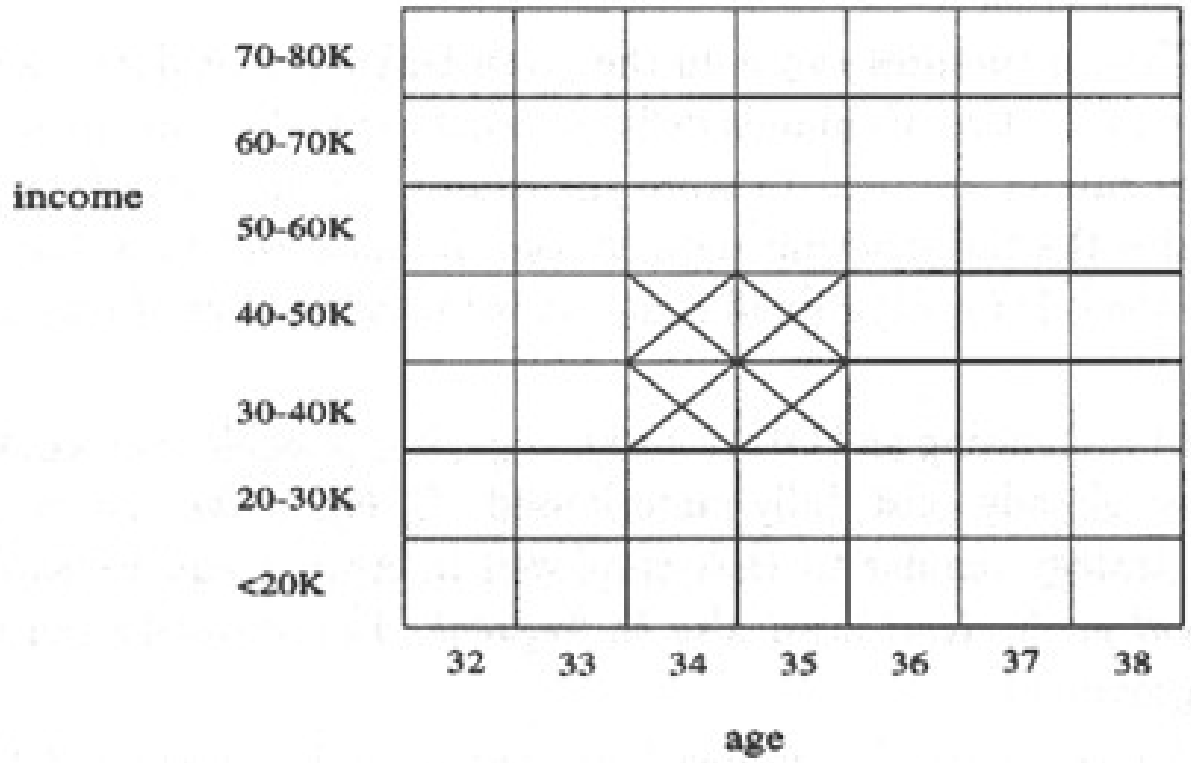
Kết hợp đa mức: Rút gọn lọc

- Trong luật phân cấp, một luật có thể dư thừa do đã có quan hệ giữa “tổ tiên” của các mục.
- Ví dụ
  - milk wheat bread [support = 8%, confidence = 70%]
  - 2% milk wheat bread [support = 2%, confidence = 72%]
- Nói rằng: luật đầu tiên là tổ tiên luật thứ hai.
- Một luật là dư thừa nếu độ hỗ trợ của nó là khít với giá trị “mong muốn”, dựa theo tổ tiên của luật.

Luật kết hợp định lượng

- Thuộc tính số là sự rời rạc hóa động d
  - Độ tin cậy hoặc độ cô đọng của luật là cực đại

- Luật kết hợp định lượng 2-D: A_{quan1} A_{quan2} A_{cat}
- Phân cụm các l
  Liền kề nhau t
  Tổng quát dựa
  Lưới 2-D



- Ví dụ

age(X,"30-34") income(
48K")
buys(X,"high resolu

Khai phá luật KH dựa theo khoảng cách

- Phương pháp đóng thùng không nắm bắt được ngữ nghĩa của dữ liệu khoảng

Price($)	Equi-width (width $10)	Equi-depth (depth 2)	Distance-based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

- Phân vùng dựa trên khoảng cách, rời rạc có ý nghĩa hơn khi xem xét :
  - Mật độ/ số điểm trong một khoảng
  - Tính "gần gũi" của các điểm trong một khoảng

Độ đo hấp dẫn: Tương quan (nâng cao)

- *play basketball* *eat cereal* [40%, 66.7%] là lạc
  - Phần trăm chung của sinh viên ăn ngũ cốc là 75% cao hơn so với 66.7%.
- *play basketball* *not eat cereal* [20%, 33.3%] là chính xác hơn, do độ hỗ trợ và tin cậy thấp hơn
- Độ đo sự kiện phụ thuộc/tương quan: lift (nâng cao)

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

KPDL dựa trên ràng buộc

- Tìm mọi mẫu trong CSDL tự động? — phi hiện thực!
  - Mẫu có thể quá nhiều mà không mục đích!
- KPDL nên là quá trình tương tác
  - Người dùng trực tiếp xác định KPDL gì khi dùng ngôn ngữ hỏi KPDL (hoặc giao diện đồ họa)
- KP dựa theo ràng buộc
  - Linh hoạt người dùng: cung cấp ràng buộc : cái được KP
  - Tối ưu hệ thống: thăm dò các ràng buộc để hiệu quả KP: KP dựa theo ràng buộc

Ràng buộc trong KPDL

- **Ràng buộc kiểu tri thức**
  - classification, association, v.v.
- **Ràng buộc dữ liệu: dùng câu hỏi kiểu SQL**
  - Tìm các cặp sản phẩm mua cùng nhau trong Vancouver vào Dec.'00
- **Ràng buộc chiều/cấp**
  - Liên quan tới vùng, giá, loại hàng, lớp khách hàng
- **Ràng buộc luật (mẫu)**
  - Mua hàng nhỏ (price < $10) nhiều hơn mua hàng lớn (sum > $200)
- **Ràng buộc hấp dẫn**
  - Luật mạng: min_support 3%, min_confidence 60%

KP ràng buộc <> tìm kiếm dựa theo ràng buộc

- **KP ràng buộc tìm/lập luận theo ràng buộc**
  - Cả hai hướng tới rút gọn không gian tìm kiếm
  - Tìm mọi mẫu bảm đảm ràng buộc <> tìm một vài (một_ câu trả lời của tìm dựa theo ràng buộc trong AI (TTNT)
  - Cố tìm theo ràng buộc <> tìm kiếm heuristic
  - *Tích hợp hai cái cho một bài toán tìm kiếm thú vị*
- **KP ràng buộc quá trình hỏi CSDL quan hệ**
  - Quá trình hỏi trong CSDL quan hệ đòi hỏi tìm tất cả
  - KP mẫu ràng buộc chung một triết lý tương tựng như cố gắng chọn về chiều sâu của câu hỏi

KP mẫu PB ràng buộc: tối ưu hóa câu hỏi

- **Cho một câu hỏi KP mấu phổ biến với một tập ràng buộc C, thì thuật toán nên là**
  - Mạnh mẽ: chỉ tìm các tập phố biến bảo đảm ràng buộc *C*
  - đầy đủ: Tìm tất cả tập phổ biến bảo đảm ràng buộc *C*
- **Giải pháp "thơ ngây/hồn nhiên" (naïve)**
  - Tìm tất cả tập PB sau đó kiểm tra ràng buộc
- **Tiếp cận hiệu quả hơn**
  - Phân tích tính chất các ràng buộc một cách toàn diện
  - Khai thác chúng sâu sắc có thể nhất trong tính toán mẫu PB.

buộc

- Chống đơn điệu (Anti-monotonicity)
  - *Một tập mục S **vi phạm** ràng buộc, mọi tập lớn hơn nó cũng vi phạm*
  - *sum(S.Price) v* là chống đơn điệu
  - *sum(S.Price) v* là không chống đơn điệu
- Ví dụ. C: range(S.profit) 15 là chống đơn điệu
  - Tập mục *ab* vi phạm C
  - Cũng vậy mọi tập chứa *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Ràng buộc nào là chống đơn điệu

Ràng buộc	Chống đơn điệu
v S	No
S V	no
S V	yes
min(S) v	no
min(S) v	yes
max(S) v	yes
max(S) v	no
count(S) v	yes
count(S) v	no
sum(S) v (a S, a 0)	yes
sum(S) v (a S, a 0)	no
range(S) v	yes
range(S) v	no
avg(S) v, { , , }	convertible
support(S)	yes
support(S)	no

Tính đơn điệu trong khai thác luật dựa theo ràng buộc

- Tính đơn điệu
  - *Khi một tập mục S **thỏa mãn** ràng buộc, thì mọi tập lớn hơn của nó cũng thỏa mãn*
  - *sum(S.Price) v* là đơn điệu
  - *min(S.Price) v là đơn điệu*
- Ví dụ. C: range(S.profit) 15
  - Tập mục *ab đảm bảo* C
  - Cũng vậy mọi tập chứa *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Ràng buộc đơn điệu

Ràng buộc	Đơn điệu
v S	yes
S V	yes
S V	no
min(S) v	yes
min(S) v	no
max(S) v	no
max(S) v	yes
count(S) v	no
count(S) v	yes
sum(S) v (a S, a 0)	no
sum(S) v (a S, a 0)	yes
range(S) v	no
range(S) v	yes
avg(S) v, { , , }	convertible
support(S)	no
support(S)	yes

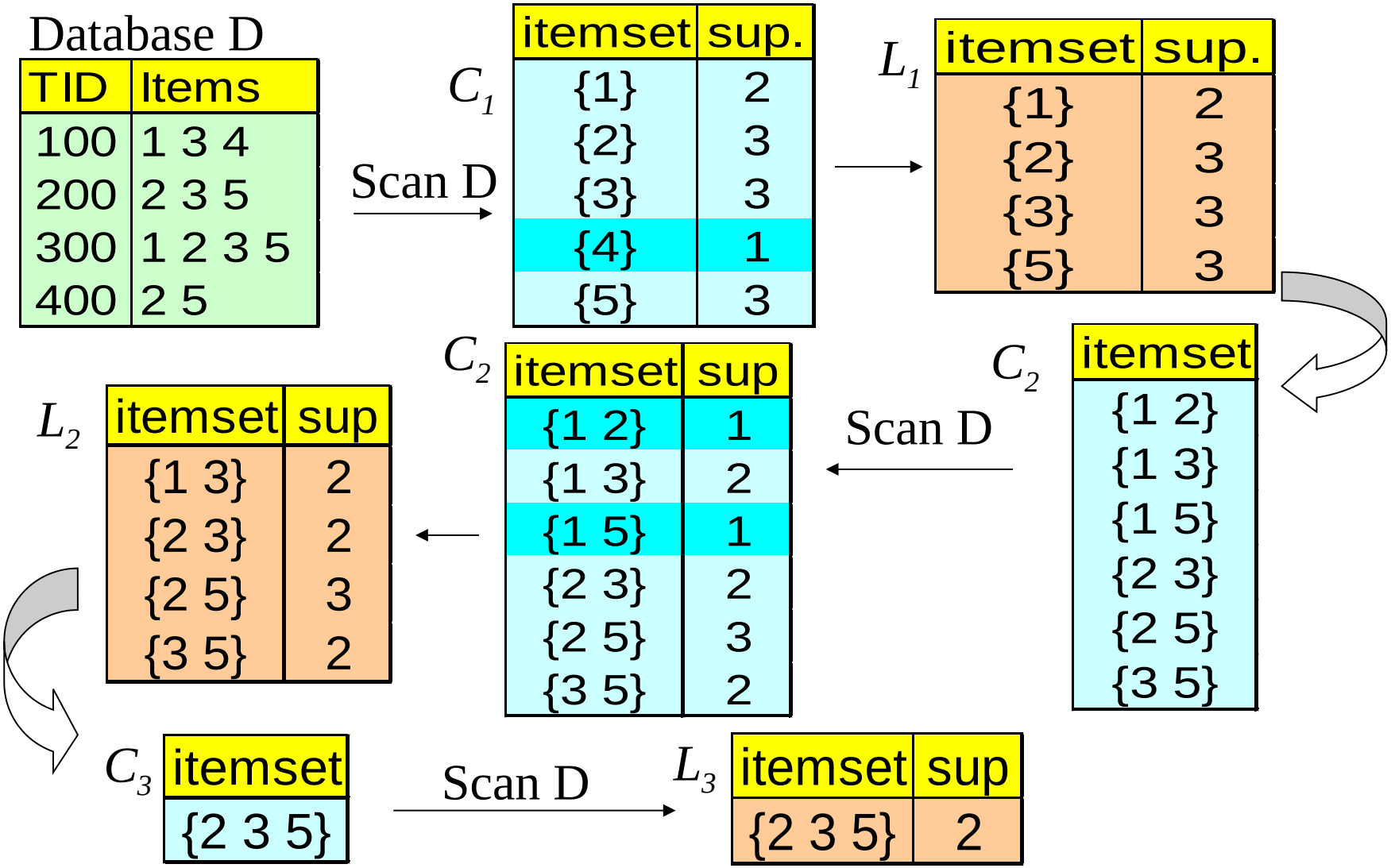
Tính cô đọng

- **Tính cô đọng:**
  - Cho A_1, là tập mục bảo đảm một ràng buộc cô đọng C, thì mọi S bảo đảm C là dựa trên A_1 , chẳng hạn, S chứa một tập con thuộc A_1
  - Tư tưởng: Bỏ qua xem xét toàn bộ CSDL giao dịch, có chăng một tập mục *S bảo đảm ràng buộc* C có thể được xác định dựa theo việc chọn các mục
  - *min*(*S.Price*) v là cô đọng
  - *sum*(*S.Price*) v không cô đọng
- **Tối ưu hóa**: Nếu C là cô đọng có thể đẩy đếm trước

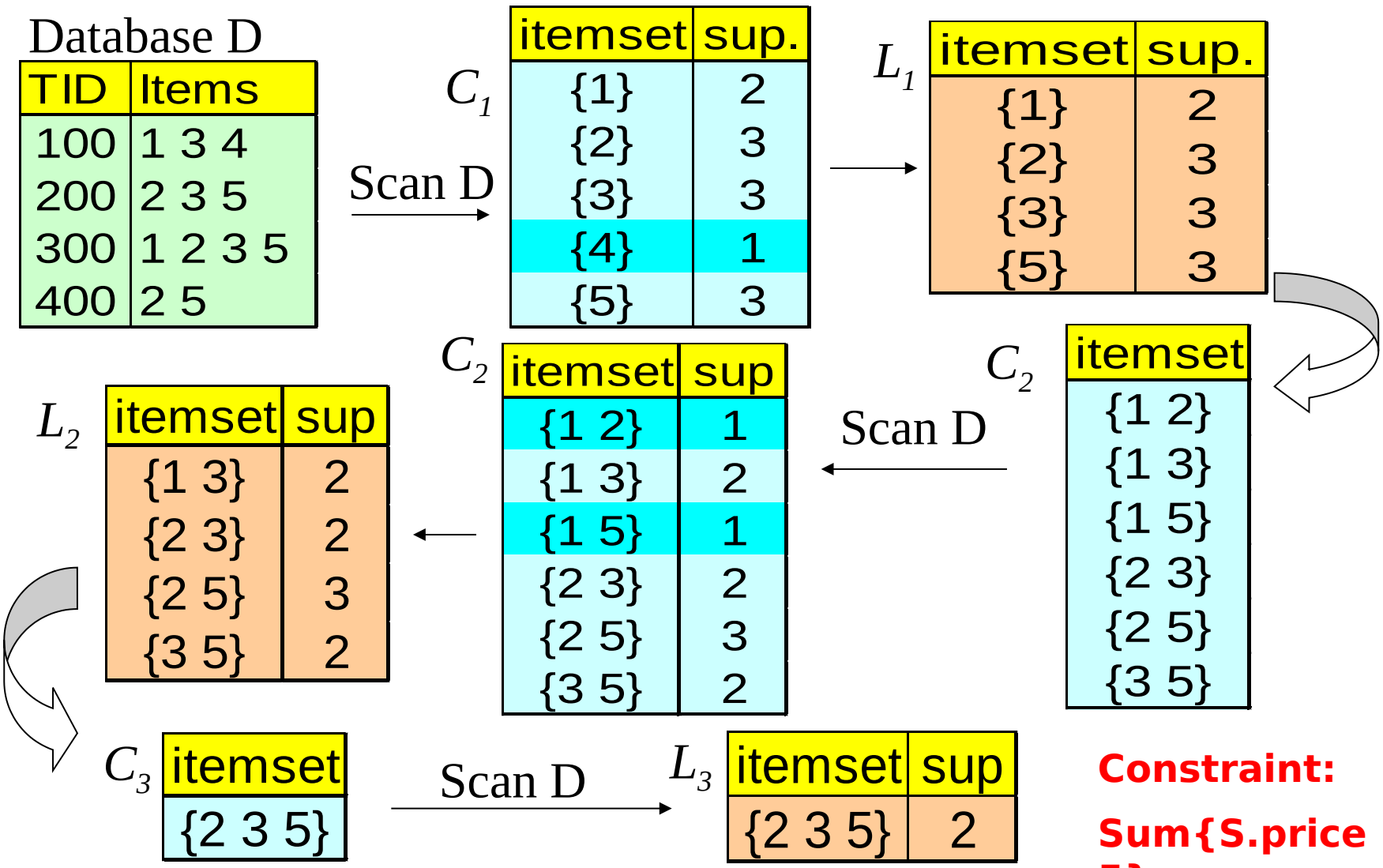
Ràng buộc cô đọng

Ràng buộc	Cô đọng
v S	yes
S V	yes
S V	yes
min(S) v	yes
min(S) v	yes
max(S) v	yes
max(S) v	yes
count(S) v	weakly
count(S) v	weakly
sum(S) v (a S, a 0)	no
sum(S) v (a S, a 0)	no
range(S) v	no
range(S) v	no
avg(S) v, { , , }	no
support(S)	no
support(S)	no

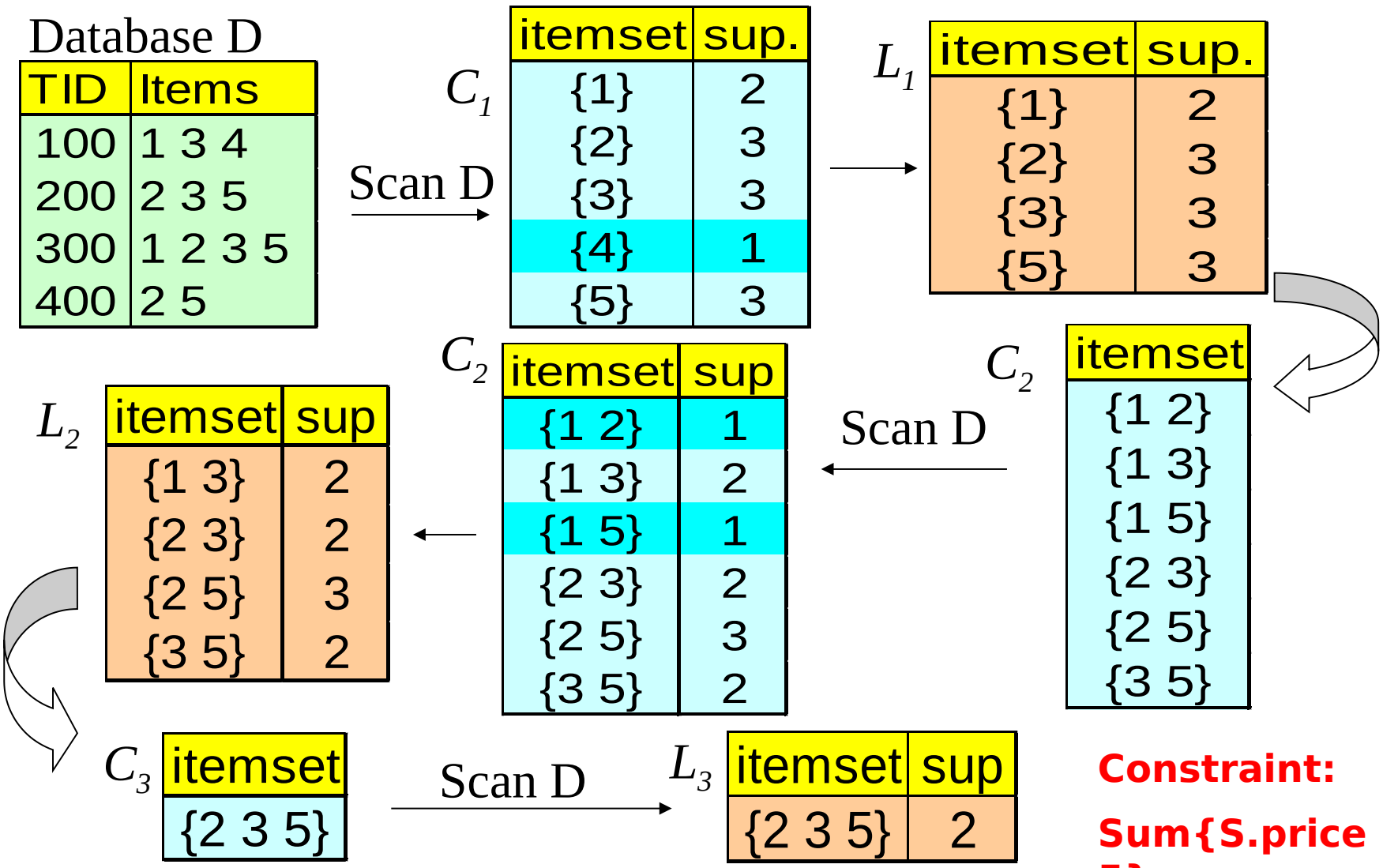
Thuật toán Apriori— Ví dụ



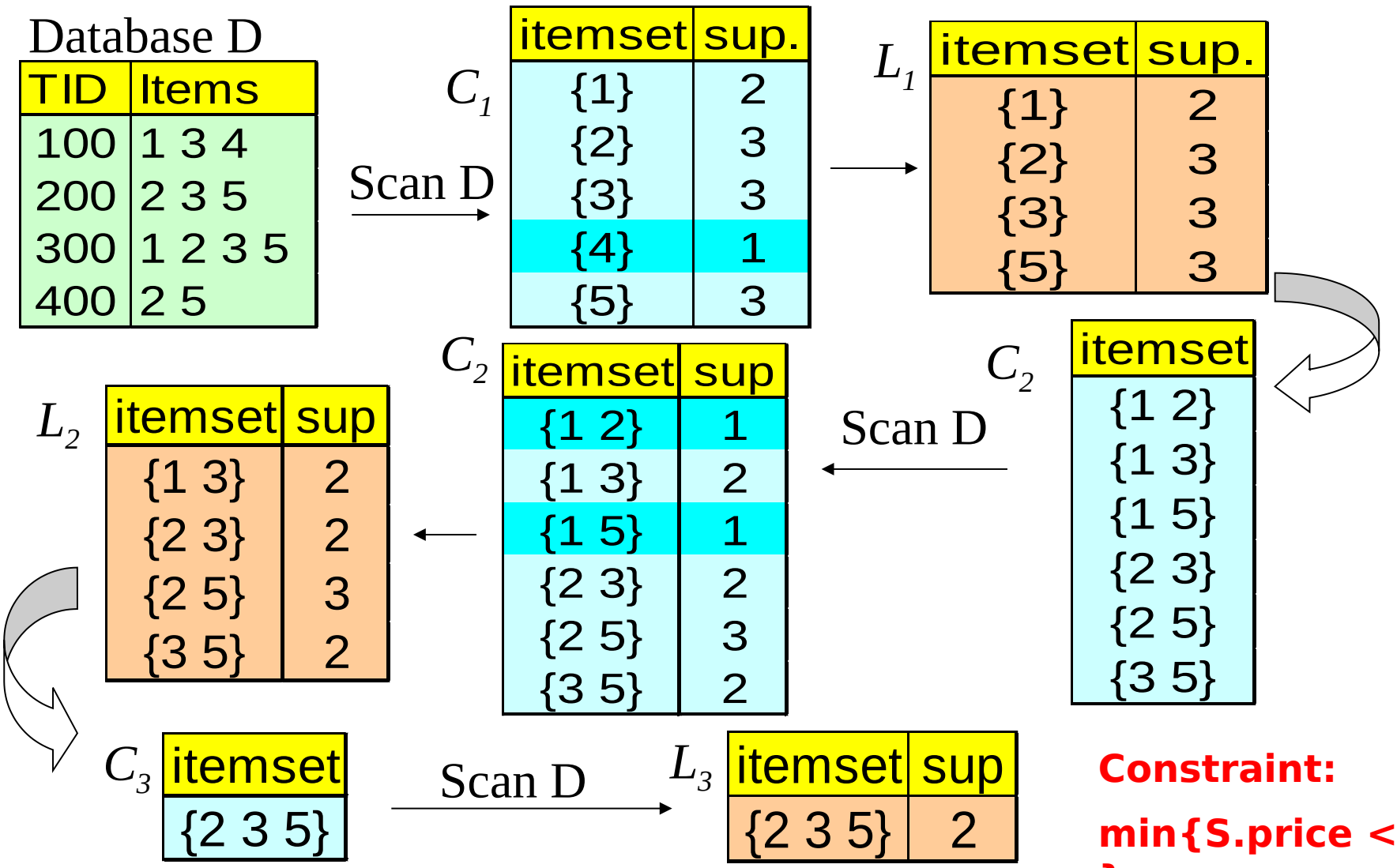
buộc


Database D
TID Items
100 1 3 4
200 2 3 5
300 1 2 3 5
400 2 5
Scan D
C_1
itemset sup.
{1} 2
{2} 3
{3} 3
{4} 1
{5} 3
L_1
itemset sup.
{1} 2
{2} 3
{3} 3
{5} 3
C_2
itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}
Scan D
C_2
itemset sup
{1 2} 1
{1 3} 2
{1 5} 1
{2 3} 2
{2 5} 3
{3 5} 2
L_2
itemset sup
{1 3} 2
{2 3} 2
{2 5} 3
{3 5} 2
C_3
itemset
{2 3 5}
Scan D
L_3
itemset sup
{2 3 5} 2
Constraint:
Sum{S.price < 5}

Apriori tăng cường: Duyệt CSDL chỉ D xuống đáy



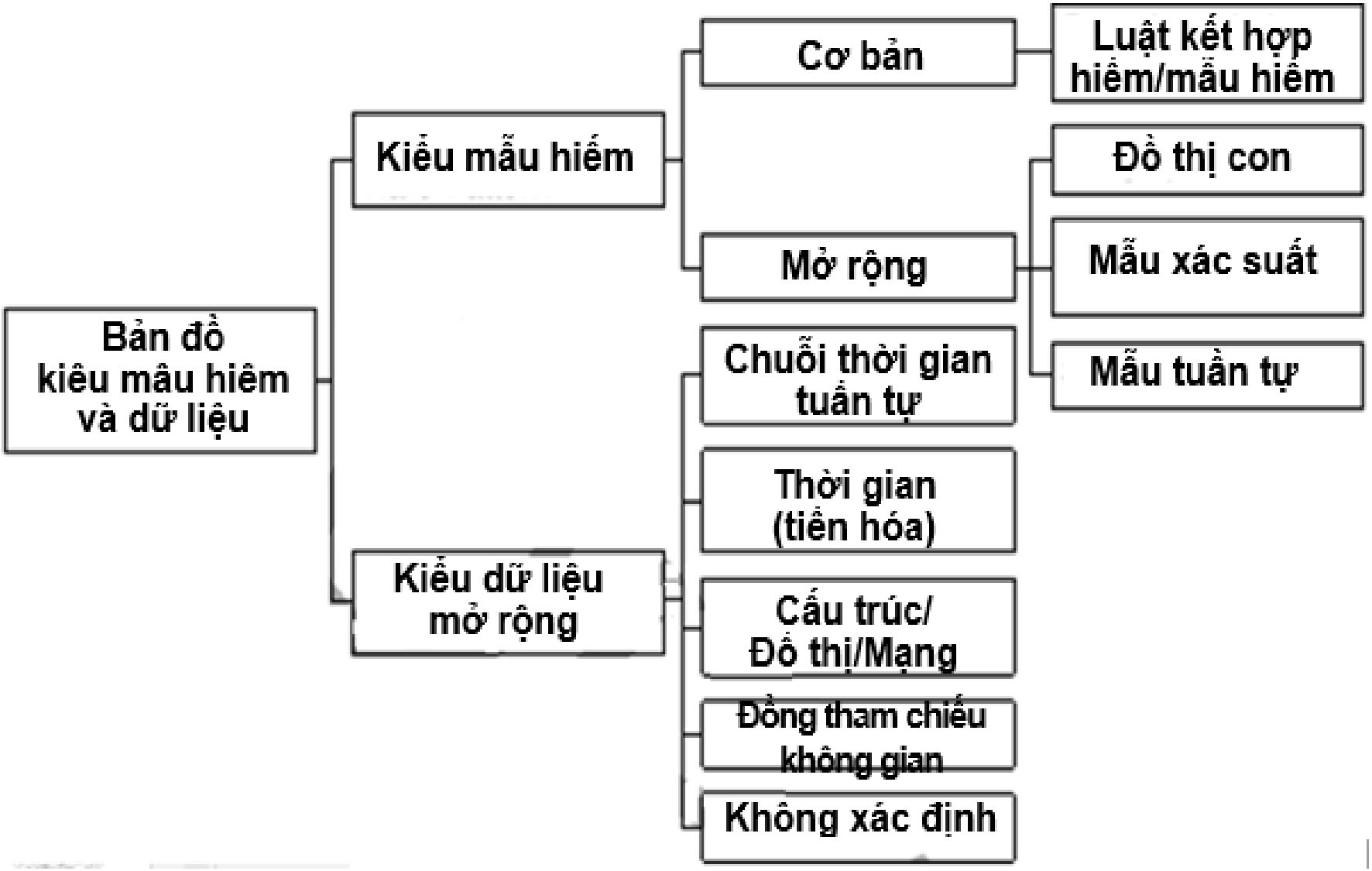
Apriori tăng buộc: Đẩy RB chống D-D xuống đáy



Luật kết hợp hiếm và luật kết hợp âm

- Luật kết hợp hiếm hàm ý chỉ các LKH không xảy ra thường xuyên trong CSDL.
- Ví dụ
  - “máy pha cà phê” “máy xay cà phê” (0.8%, 80%). **[Koh05] Koh Y. S., Rountree N. (2005). Finding Sporadic Rules Using Apriori-Inverse. *Proc. of PAKDD2005*, pp. 97-106.**
  - “ăn chay” “bệnh tim mạch”. **[Szathmary10] Szathmary L., Valtchev P., and Napoli A. (2010). Generating Rare Association Rules Using Minimal Rare Itemsets Family. *International Journal of Software and Informatics*, Vol. 4 (3), pp. 219-238.**
  - "thuốc hạ lipid trong máu Cerivastatin" "tác động xấu khi điều trị“. **[Szathmary10]**
- Luật kết hợp âm hàm ý chỉ các LKH mà các mục là xung khắc nhau trong CSDL “nếu A thì không B”

Luật hiếm: Phân loại



[Koh16] Yun Sing Koh, Sri Devi Ravana. *Unsupervised Rare Pattern Mining: A Survey*. TKDD 10(4): 45 (2016)

Khai phá luật kết hợp hiếm

- Hai hướng tiếp cận chính phát hiện luật hiếm:
  - Sử dụng ràng buộc
  - Sử dụng ranh giới
- Hạn chế của cách tiếp cận hiện tại:
  - Sinh mọi tập không phổ biến chi phí cao.
  - Thực hiện trên CSDL tác vụ.

Luật kết hợp hiếm sporadic tuyệt đối

- Luật hiếm Sporadic tuyệt đối (Koh và csự - 2005):
  - Luật kết h $\begin{cases} \text{conf}(X \rightarrow Y) \geq \textit{minConf}, \\ \sup(X \cup Y) < \text{maxSup}, \\ \forall x \in X \cup Y, \sup(x) < \max \text{Sup}. \end{cases}$:ho:
- Thuật toán tìm các tập Sporadic tuyệt đối: Apriori-Inverse
- Hạn chế:
  - Thuật toán có hiệu quả ở mức trung bình so với các thuật toán khác.
  - Chỉ được tìm trên các CSDL tác vụ.

 Cần phát triển thuật toán phát hiện luật Sporadic tuyệt đối hiệu quả hơn, và phát hiện luật này cả trên CSDL định lượng

Luật kết hợp Sporadic tuyệt đối hai ngưỡng

- Mục đích nghiên cứu:
  - Phát triển thuật toán phát hiện luật Sporadic tuyệt đối hiệu quả hơn.
  - Đề xuất mở rộng bài toán: tìm các luật A B sao cho:

$$\begin{cases} \text{conf}(A \rightarrow B) \geq \textit{minConf}, \\ \text{minSup} \leq \text{sup}(\text{A} \cup \text{B}) < \text{maxSup}, \\ \forall x \in A \cup B,\ \text{sup}(x) < \text{max Sup}. \end{cases}$$

- **Đóng góp chính:**
  - Bài toán phát hiện LKH tuyệt đối 2 ngưỡng là tổng quát hơn.
  - Thuật toán được phát triển theo cách tiếp cận thuật toán CHARM: Chỉ tìm các tập Sporadic tuyệt đối đóng 2 ngưỡng.

CSDL tuần tự và Phân tích mẫu tuần tự

Năm	*<2008*	*2008*	*2009*	*2010*	*2011*	*2012*	*Tất cả*
Tại tiêu đề	193	18	18	20	20	8	277
Mọi nơi	10000	980	1040	1080	1012	788	14900

Phần mềm phân tích chuỗi thời gian EidoSearch: Trợ giúp đánh dấu mẫu dữ liệu hấp dẫn và EidoSearch đi tìm mọi mẫu tương tự từ quá khứ và hiện tại, phân tích kết quả tìm kiếm này, và chỉ ra xu hướng gì sẽ xảy ra.

Gait-CAD Matlab toolbox: trực quan hóa và phân tích chuỗi thời gian, bao gồm phân lớp, hồi quy, và phân cụm. Giấy phép GNU-GPL.

Miningco: chương trình mã nguồn mở tự động tìm ra mẫu và quan hệ trong weblogs và các bộ dữ liệu khác.

SAS Enterprise Miner

XAffinity (TM): xác định mối quan hệ thân hoặc mẫu trong giao dịch và dòng dữ liệu nháy phím

http://www.kdnuggets.com/software/sequence.html

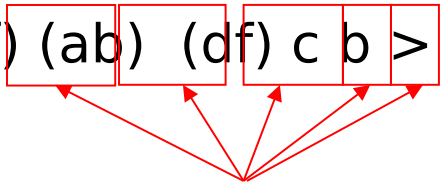
CSDL TT và PT MTT (2)

- ***CSDL giao dịch, CSDL chuỗi thời gian <> CSDL tuần tự***
- Mấu PB <> mấu TT (PB)
- Ứng dụng của KP Mấu TT
  - Tuần tự mua của khách hàng:
    - *Đầu tiên mua máy tính, sau đó CD-ROM, và sau đó là máy ảnh số, trong vòng 3 tháng.*
  - Phẫu thuật y tế, thảm họa tự nhiên (động đất…), quá trình KH và kỹ nghệ, chứng khoán và thị trường….
  - Mẫu gọi điện thoại, dòng click tại Weblogs
  - Dãy DNA và cấu trúc gene

Khái niệm KP mẫu TT

- Cho một tập các dãy, tìm tập đầy đủ các dãy con phổ biến

dãy TT : < (ef) (ab) (df) c b >



CSDL dãy TT

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Một phần tử chứa một tập mục.
Tập mục trong một phần tử là không thứ t
, và viết chúng theo ABC.

<a(bc)dc> là *dãy con* của
<a(abc)(ac)d(cf)>

Cho độ hỗ trợ *min_sup* =2, <(ab)c> là mẫu tuần tự *sequential pattern*