Khai thác các mẫu tuần tự phổ biến mà không cần phát sinh các tập ứng viên

Nội dung báo cáo

1. **Giới thiệu khai khoáng mẫu tuần tự**
2. **Cách tiếp cận Apriori**
3. **Thiết kế cây và xây dựng cây FP (Frequent Pattern Tree)**
4. **Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP**
5. **Đánh giá các kết quả thực nghiệm**
6. **Các vấn đề đang còn thảo luận**

Giới thiệu khai khoáng các mẫu tuần tự

- Từ một tập dữ liệu, chúng ta tìm các mẫu có chiều dài là 1, 2, 3, … thỏa min_support

có 60% khách hàng mua sản phẩm a: mẫu có chiều dài 1

có 70% khách hàng mua cùng lúc sản phẩm a và b: mẫu có chiều dài 2

Nội dung báo cáo

1. Giới thiệu khai khoáng mẫu tuần tự
2. **Cách tiếp cận Apriori**
3. Thiết kế cây và xây dựng cây FP
4. Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP
5. Đánh giá các kết quả thực nghiệm
6. Các vấn đề đang còn thảo luận

Cách tiếp cận Apriori

- Thuật toán Apriori:
  - Ý tưởng thuật toán
    - Lặp đi lặp lại qúa trình phát sinh tập các ứng viên có chiều dài k+1 từ các mẫu phổ biến chiều dài k
    - Kiểm tra độ phổ biến của ứng viên thỏa min_support trong CSDL

Cách tiếp cận Apriori (tt)

TID	Các mặt hàng được mua
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

Chọn độ phổ biến cực tiểu là ξ (min_support) = 3 (60%)

Cách tiếp cận Apriori (tt)

- **<u>Bước 1:</u>** Tìm F1 chứa các mẫu có chiều dài là 1 thỏa min_support

  F1={f, c, a, b, m, p}

- **<u>Bước 2:</u>** Quá trình lặp tìm tập ứng viên Ck và từ Ck tìm tập Fk

  Với k=2

  C2 = {<fc>, <fa>, <fb>, <fm>, <fp>, <ca>, <cb>, <cm>, <cp>, <ab>, <am>, <ap>, <bm>, <bp>, <mp>}

  F2= {<fc>, <fa>, <fm>, <ca>, <cm>, <cp>, <am>}

Cách tiếp cận Apriori

Với k=3

C3={<fca>, <fcm>, <fcp>, <fam>, <cam>}

F3={<fca>, <fcm>, <fam>, <cam>}

Với k=4

C4={<fcam>}

F4={<fcam>}

Với k=5

C5=∅ → ngưng

Vậy tập đầy đủ các mẫu phổ biến là: **f, c, a, b, m, p, fc, fa, fm, ca, cm, cp, am, fca, fcm, fam, cam, fcam**

Những hạn chế của thuật toán Apriori

- Hai loại chi phí của thuật toán Apriori:
  - Chi phí phát sinh ứng viên

10^4 mẫu phổ biến có kích thước là 1 → cần phải phát sinh hơn 10^7 mẫu phổ biến có kích thước là 2

Đề nghị xây dựng cây FP (FP-tree)

- để kiểm tra ứng viên thỏa *min_support*

↳ **Chi phí duyệt CSDL lớn**

↳ **Mục tiêu: tránh phát sinh tập ứng viên quá lớn**

Nội dung báo cáo

1. Giới thiệu khai khoáng mẫu tuần tự
2. Cách tiếp cận Apriori
3. **Thiết kế cây và xây dựng cây FP**
4. Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP
5. Đánh giá các kết quả thực nghiệm
6. Các vấn đề đang còn thảo luận

Thuật toán xây dựng cây FP

- **Bước 1:** Duyệt CSDL, lấy ra tập các item phổ biến F và tính độ phổ biến của chúng.

  Sắp xếp các item trong tập F theo thứ tự giảm dần của độ phổ biến, ta được tập kết quả là L.

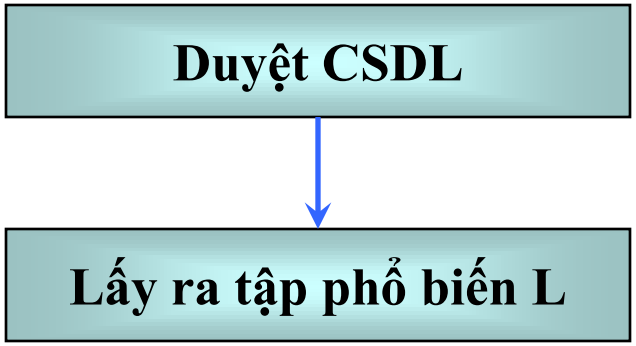
- **Bước 2:** Tạo nút gốc cho cây T, và tên của nút gốc sẽ là Null.

  Sau đó duyệt CSDL lần thứ hai. Ứng với mỗi giao tác trong CSDL thực hiện 2 công việc sau:

  - Chọn các item phổ biến trong các giao tác và sắp xếp chúng theo thứ tự giảm dần độ phổ biến trong tập L
  - Gọi hàm Insert_tree([p|P],T) để đưa các item vào trong cây T

Thuật toán xây dựng cây FP

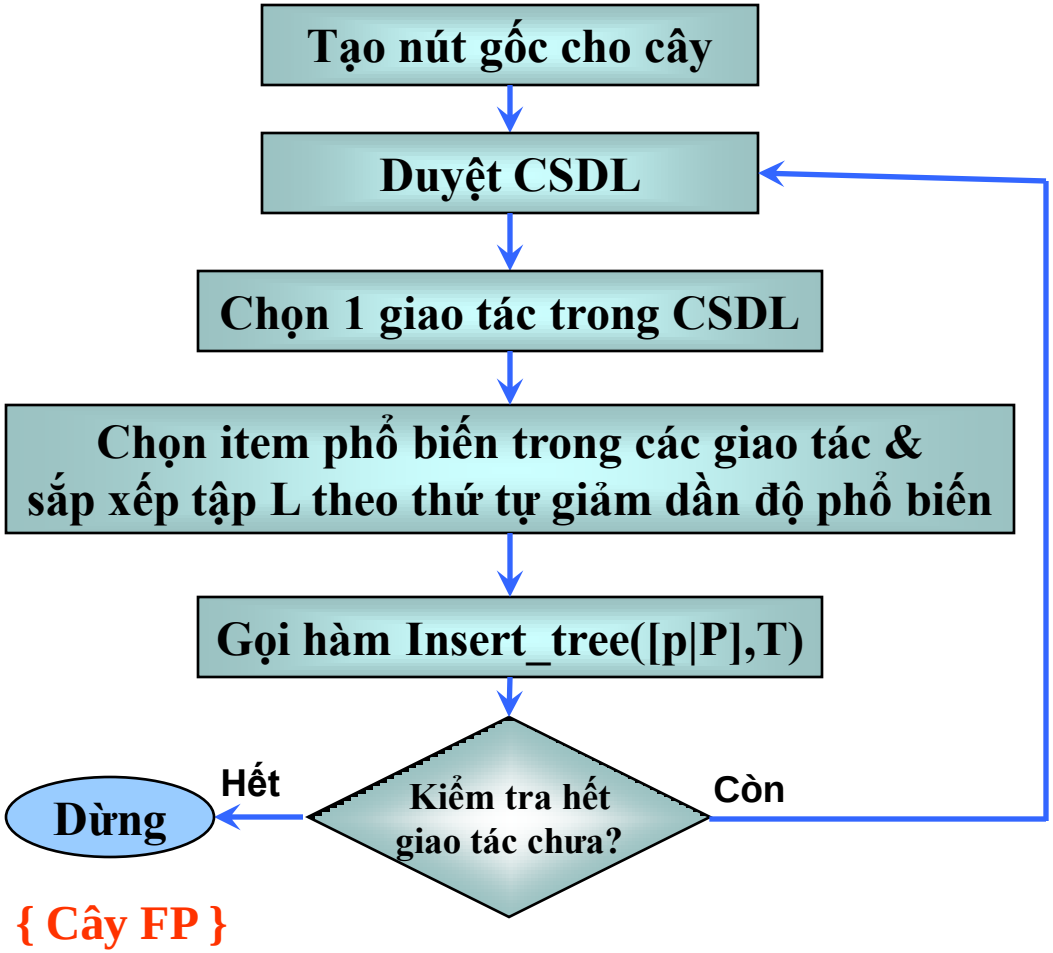
- Bước 1:



L bao gồm các item phổ biến theo thứ tự giảm dần của độ phổ biến

Thuật toán xây dựng cây FP

- Bước 2:



Cây FP - Ví dụ

- Bảng tất cả các item:

TID	Các mặt hàng được mua
100	f, a, c, d, g, i, m,p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

Chọn độ phổ biến cực tiểu là ξ (min_support) = 3 (60%)

Cây FP - Ví dụ (tt)

TID	Các mặt hàng được mua
100	f, a, c, d, g, i, m,p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

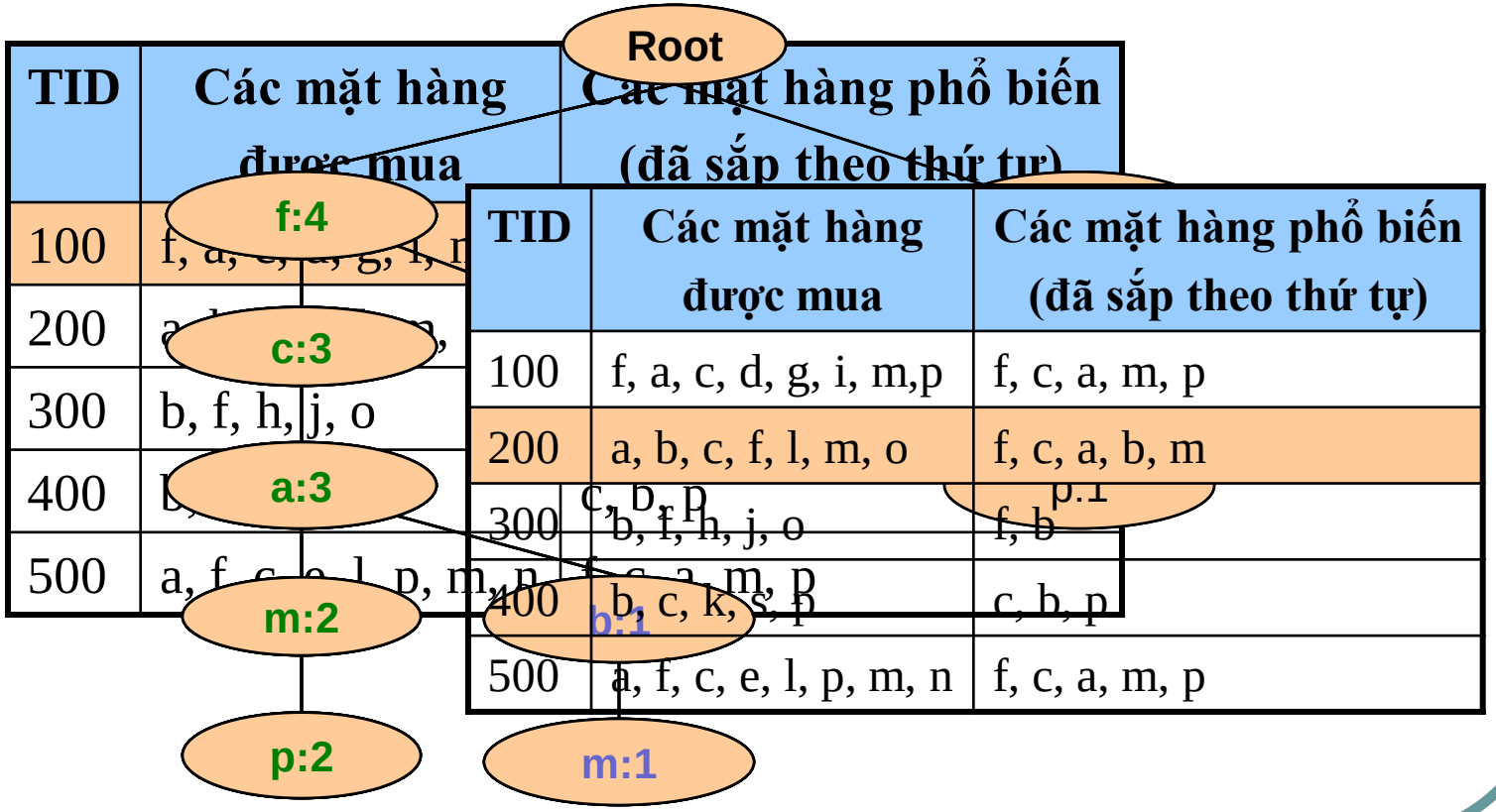
- Ta có một danh sách các mặt hàng phổ biến L là: <(f:4), (c:4), (a:3), (b:3), (m:3), (p:3)>

Các mặt hàng đã được sắp thứ tự giảm dần theo độ phổ biến

Item	a	b	c	d	e	f	g	i	j	l	k	m	n	o	p	s
Supp	3	3	4	1	1	4	1	1	1	2	1	3	1	2	3	1

Cây FP - ví dụ (tt)

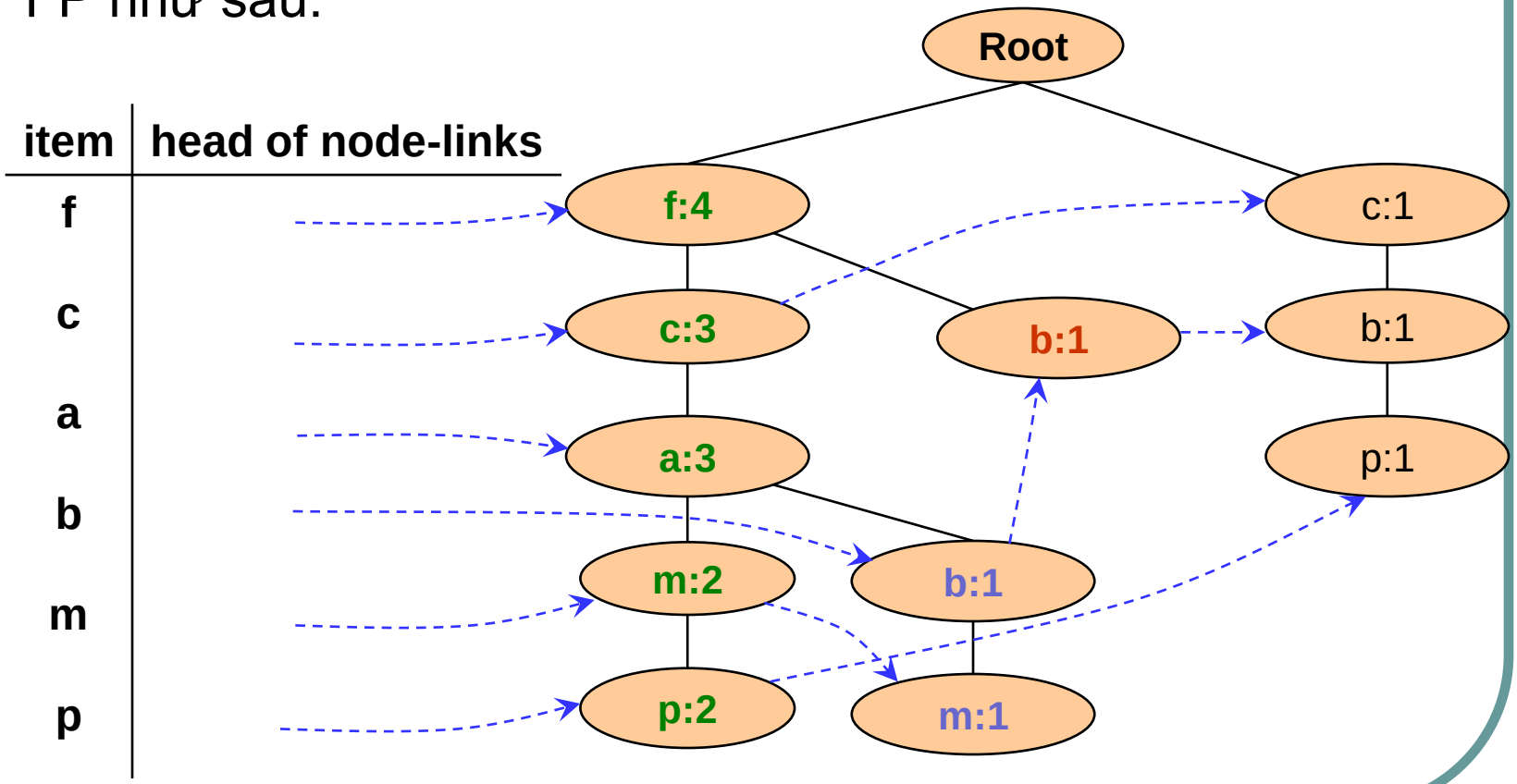
- Từ tập dữ liệu ban đầu, chúng ta có được cây FP như sau:



TID	Các mặt hàng được mua	Các mặt hàng phổ biến (đã sắp theo thứ tự)
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	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

Cây FP - ví dụ (tt)

- Từ tập dữ liệu ban đầu, ta xây dựng header table của cây FP như sau:



Phân tích chi phí thuật toán tạo cây FP

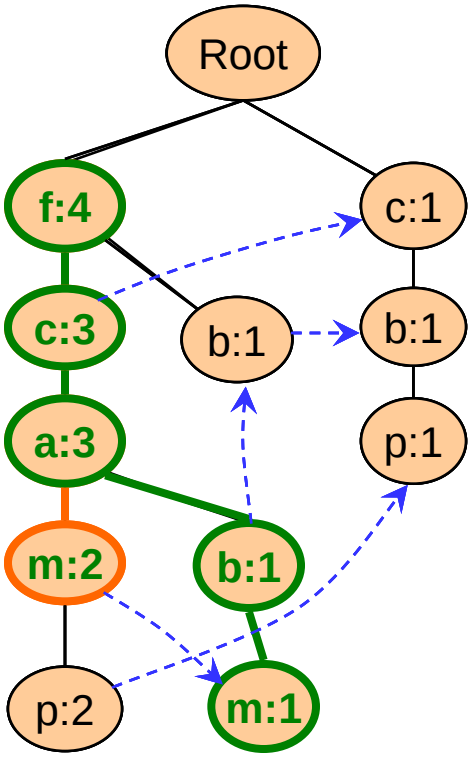
- Ứng với thuật toán trên thì chúng ta cần chính xác là 2 lần quét qua tất cả các giao tác của CSDL
- Chi phí đưa một giao tác Trans vào trong cây là O(|Trans|)

  với |Trans| là số lần xuất hiện của các item trong giao tác Trans này.

Nội dung báo cáo

1. Giới thiệu khai khoáng mẫu tuần tự
2. Cách tiếp cận Apriori
3. Thiết kế cây và xây dựng cây FP
4. **Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP**
5. Đánh giá các kết quả thực nghiệm
6. Các vấn đề đang còn thảo luận

Định nghĩa



- Cơ sở điều kiện của nút “m”:

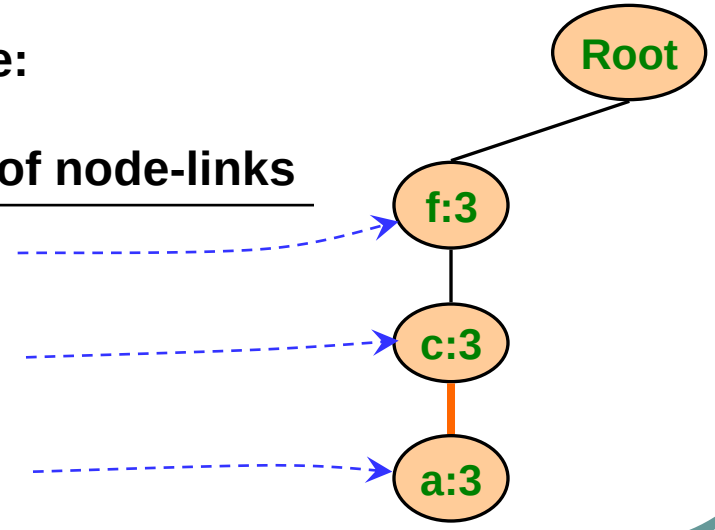
  (f:2, c:2, a:2)

  (f:1, c:1, a:1, b:1)

- Cây điều kiện FP của “m”:

Header table:

item	head of node-links
f	
c	
a	



Thuật toán khai khoáng các mẫu phổ biến sử dụng cây FP

Procedure **FP-growth**(*Tree*, *α*)

{

(1) **Nếu** *Tree* có chứa một đường đi đơn P

(2) **Thì với mỗi** cách kết hợp γ của các nút trong đường đi P **thực hiện**

(3) phát sinh tập mẫu $\gamma U \alpha$, support = min(support của các nút trong γ);

(4) **ngược lại ứng với mỗi** a_i trong thành phần của *Tree* **thực hiện** {

(5) phát sinh tập mẫu $\beta = a_i U \alpha$ với độ phổ biến

support = a_i.support;

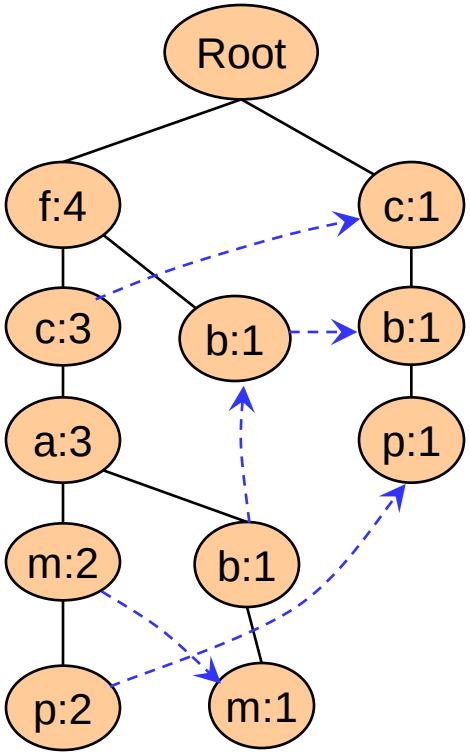
(6) xây dựng cơ sở điều kiện cho β và sau đó xây dựng cây FP $Tree_\beta$ theo điều kiện của β;

(7) **Nếu** $Tree_\beta \neq \varnothing$

(8) **thì** gọi lại hàm FP-growth($Tree_\beta$, β) }

}

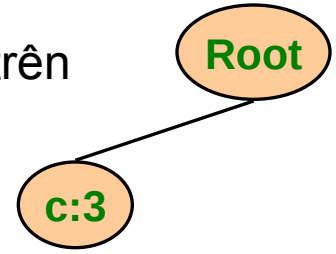
Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP (tt)



Call FP-Growth(Tree, null)

Đối với nút "p"

- β="p" U null = "p", xuất kết quả p:3
- Cơ sở điều kiện là:
  (f:2, c:2, a:2, m:2)
  (c:1, b:1)
- Cây FP với điều kiện trên {(c:3)}|p

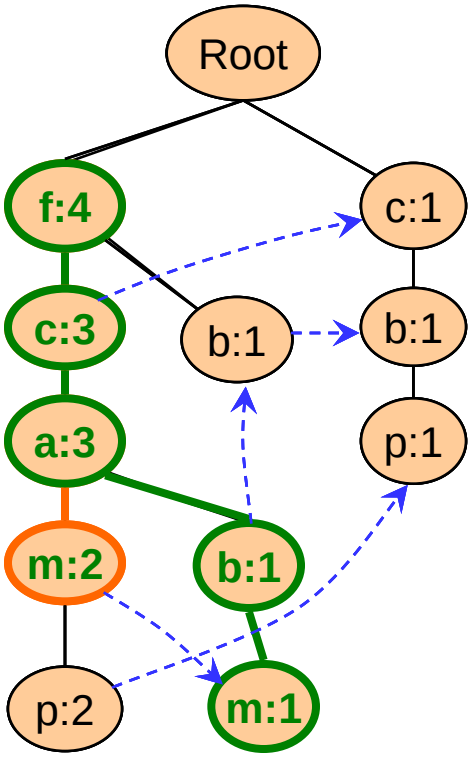


- Xuất kết quả là: cp:3

Vậy nút p có các mẫu tuần tự phổ biến là: p:3, cp:3

Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP (tt)

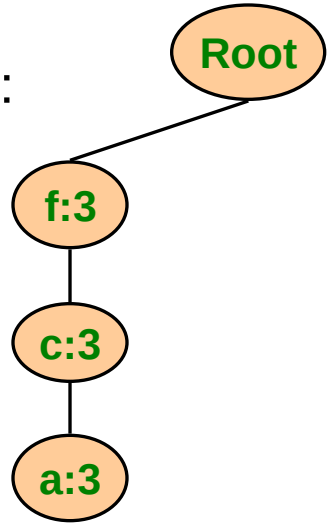
Đối với nút "m"



- β="m" U null = "m", Xuất kết quả m:3
- Cơ sở điều kiện của nút "m":

  (f:2, c:2, a:2)

  (f:1, c:1, a:1, b:1)



- Cây điều kiện FP của "m":
- Gọi FP-Growth($Tree_m$, "m")
- Vì $Tree_m$ có chứa đường đi đơn

Nên nút m có các mẫu tuần tự phổ biến là: {(m:3), (am:3), (cm:3), (fm:3), (cam:3), (fam:3), (fcm:3), (fcam:3)}

Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP (tt)

- Bảng kết quả của tất cả các item:

Item	Cơ sở điều kiện	Cây điều kiện FP
p	{(f:2, c:2, a:2, m:2), (c:1, b:1)}	{(c:3}|p
m	{(f:2, c:2, a:2), (f:1, c:1, a:1, b:1)}	{(f:3, c:3, a:3)}|m
b	{(f:1, c:1, a:1), (f:1), (c:1)}	∅
a	{(f:3, c:3)}	{(f:3, c:3)}|a
c	{(f:3)}	∅
f	∅	∅

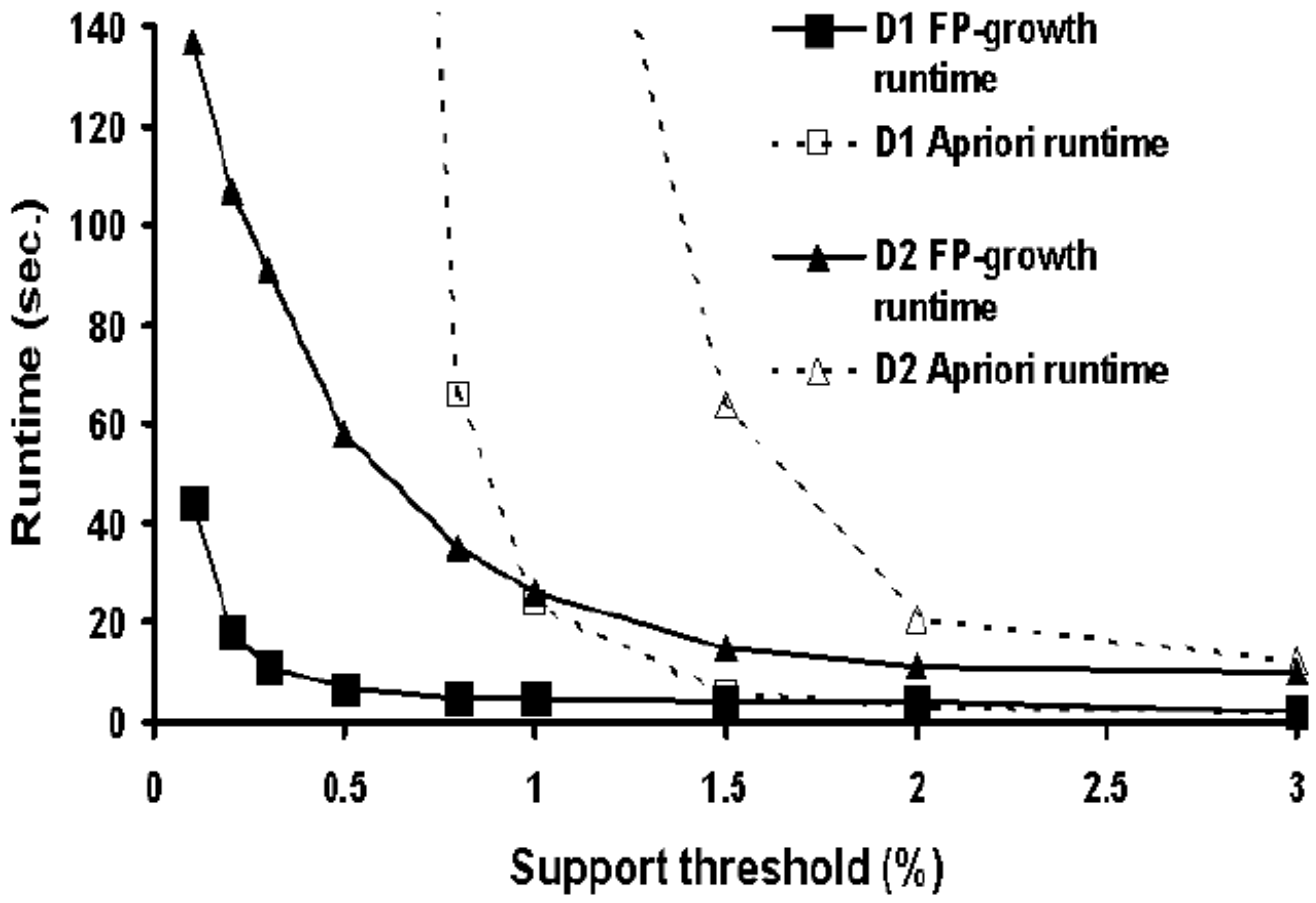
Nội dung báo cáo

1. Giới thiệu khai khoáng mẫu tuần tự
2. Cách tiếp cận Apriori
3. Thiết kế cây và xây dựng cây FP
4. Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP
5. **Đánh giá các kết quả thực nghiệm**
6. Các vấn đề đang còn thảo luận

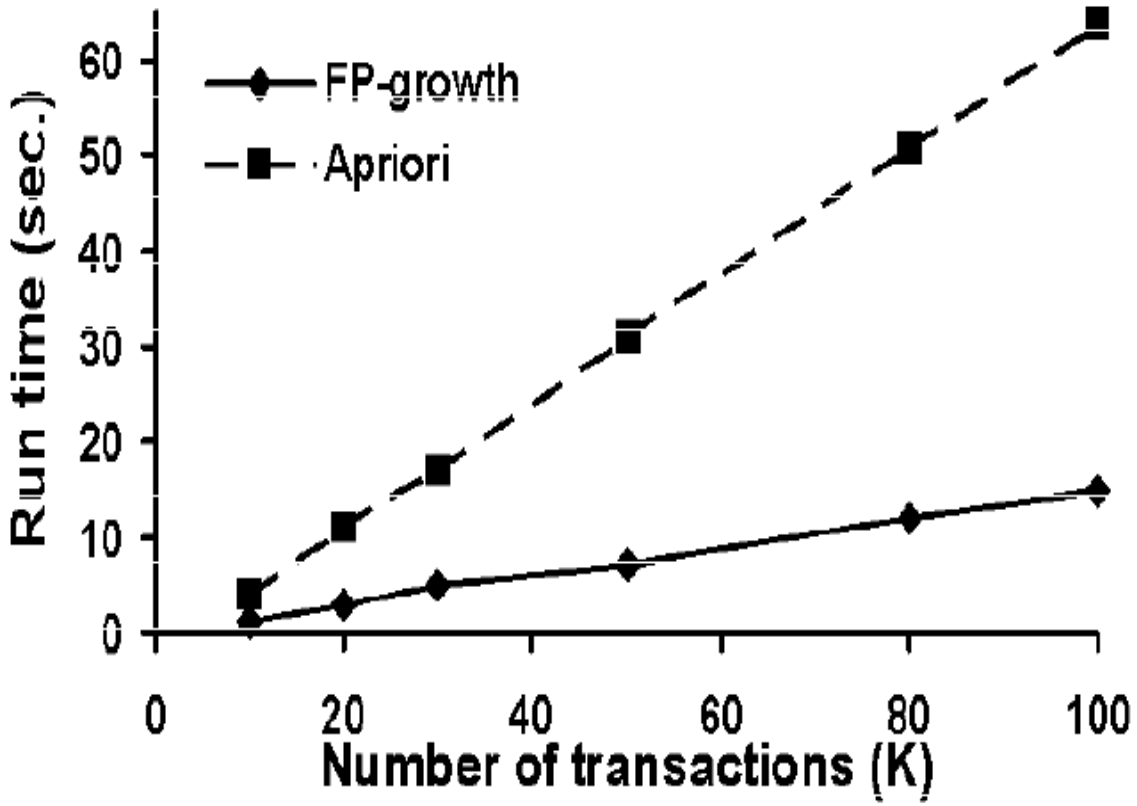
Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP (tt)

- Hiệu quả hơn so với Apriori.
- Phân chia và kiểm soát quá trình xử lý.
- Sử dụng cây FP để biểu diễn các mẫu phổ biến thì dữ liệu giảm rất đáng kể so với cách biểu diễn trong CSDL.

So sánh FP-growth và Apriori



So sánh FP-growth và Apriori



Nội dung báo cáo

1. Giới thiệu khai khoáng mẫu tuần tự
2. Cách tiếp cận Apriori
3. Thiết kế cây và xây dựng cây FP
4. Khai khoáng các mẫu phổ biến bằng cách sử dụng cây FP
5. Đánh giá các kết quả thực nghiệm
6. **Các vấn đề đang còn thảo luận**

Các vấn đề đang còn đang thảo luận

- Vấn đề xây dựng cây FP cho các projected database.
- Vấn đề tổ chức lưu trữ cây FP trên đĩa.
- Vấn đề cập nhật lại cây khi cây tăng trưởng về mặt kích thước.

Vấn đề xây dựng cây FP cho projected database

- Không thể xây dựng cây FP trong bộ nhớ chính khi CSDL là lớn.
- Đầu tiên phân chia CSDL vào trong các projected database và sau đó xây dựng một cây FP và khai thác cây này trong mỗi projected database.

Vấn đề tổ chức lưu trữ cây FP trên đĩa

- Lưu trữ cây FP trong các đĩa cứng.
  Sử dụng cấu trúc B+Tree.

Vấn đề cập nhật lại cây khi cây tăng trưởng về mặt kích thước

- Các thông tin bị mất.
- Việc tái xây dựng lại cây có thể xảy ra.

Tài liệu tham khảo

- **[1] Jiawei Han, Jian Pei, and Yiwen Yin (2000). Mining Frequent Patterns without Candidate Generation. The Natural Sciences and Engineering Research Council of Canada.**
- **[2] H. Huang, X. Wu, and R. Relue (2002). Association analysis with one scan of databases. In IEEE International Conference on Data Mining, pages 629-636.**
- **[3] J. Liu, Y. Pan, K. Wang, and J. Han (2002). Mining frequent item sets by oppotunistic projection. In Eight ACMSIGKDD Internationa Conf. on Knowledge Discovery and Data Mining, pages 229-238, Edmonton, Alberta.**
- **[4] F. Frahne, L. Lakshmanan, and X.Wang (2000). Efficient mining of constrained correlated sets. In ICDE'00.**
- **[5] R. Agrawal and R.Srikant (1995). Mining sequential patterns. In ICDE'95 pp. 3-14.**
- **[6] R. J. Bayardo (1998). Efficiently mining long patterns from databases. In SIGMOD'98 pp. 85-93.**
- **[7] J. Han, J. Pei, and Y. Yin (1999). Mining partial periodicity using frequent pattern trees. In CS Tech. Rep. 99-10, Simon Fraser University.**