DATA MINING
&
DATA WAREHOUSING

Module III

- Association Rule Mining
  - ● What is AR
  - ● Methods to discover AR
  - ● Apriori algo
  - ● Partition algo
  - ● Pincer seaarch algo
  - ● FPtree growth algo
  - ● Incremental algo
  - ● Border algo
  - ● Generalized ARs

FP-GROWTH Algorithm

for

Frequent Pattern Generation

What is

FP Tree Growth Algorithm

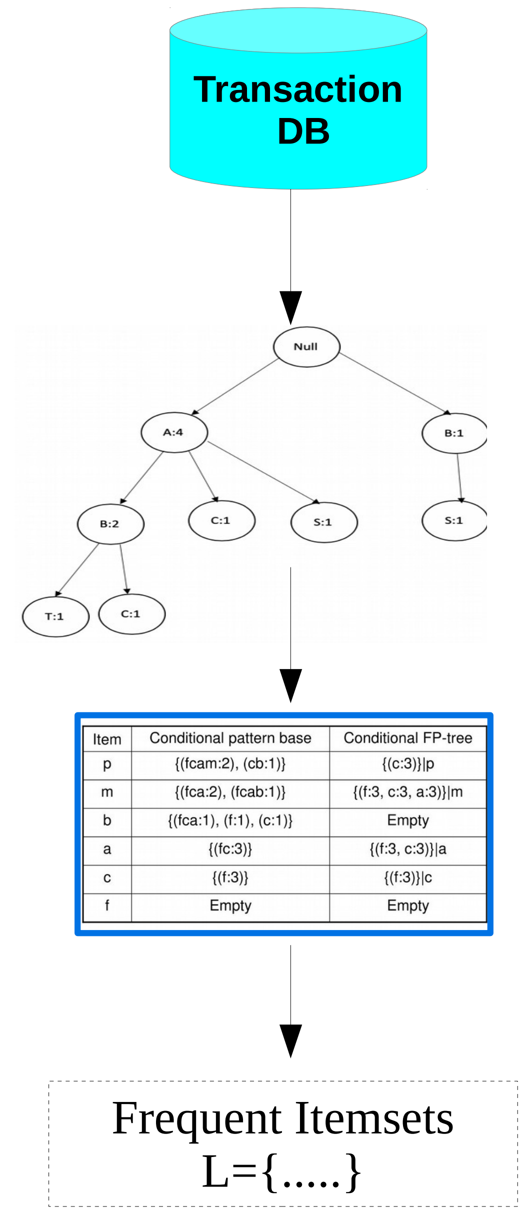
- **FP tree algorithm, which is used to identify frequent patterns in the area of Data Mining**
- **This algorithm avoids the generation of large number of candidate sets**
- **Prposed by Han et al**
- **Idea – to maintain FP-tree of the DB**

- FP-Tree
  - Extended prefix-tree structure
  - That stores the crucial and quantitative info about frequent sets
- Tree nodes are frequent itemsets
  - More frequently occurring nodes are having better chances of sharing nodes than the less frequently occurring ones

Overview of FP-Growth

Overview of FP-Growth

- **Compress** a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly compacted, but complete for frequent pattern mining
  - avoid costly repeated database scans
- Develop an efficient, FP-tree-based frequent **pattern mining method** (FP-growth)
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only.



Item	Conditional pattern base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

- **Two phases**
  - **Phase I**
    - **Construction of FP Tree**
  - **Phase II**
    - **Mine the FP Tree to generate Frequent Patterns**

Construction of FP-Tree

Example

STEP 1

Finding Frequency of Single Items :

Transaction DB

TID	*Items bought*
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*}

min_sup = 3

All Items and Frequency

Items	frequency
a	3
b	3
c	4
d	1
e	1
f	4
g	1
h	1
i	1
j	1
k	1
l	2
m	3
n	1
o	2
p	3

Frequent Items

Items	frequency
f	4
c	4
a	3
b	3
m	3
p	3

After the First Scan of Database

STEP 2

Scan the DB for the second time, order frequent items each transaction

Transaction DB

TID	*Items bought*
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	$\{a, f, c, e, l, p, m, n\}$

TID	Items
100	{f,c,a,m,p}
200	{f,c,a,b,m}
300	{f,b}
400	{c,b,p}
500	{f,c,a,m,p}

order the frequent items in each transaction and remove the infrequent items

STEP 3

From the reorderd transaction DB construct the FP Tree – Create Header Table

Create the Header Table

Reordered Transaction

TID	Items
100	{f,c,a,m,p}
200	{f,c,a,b,m}
300	{f,b}
400	{c,b,p}
500	{f,c,a,m,p}

Item	Pointer to Header Node
f	NULL
c	NULL
a	NULL
b	NULL
m	NULL
p	NULL

STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Root Node

{ }

Header Table

Item	Pointer to Header Node
f	NULL
c	NULL
a	NULL
b	NULL
m	NULL
p	NULL

STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Root Node

{ }

Header Table

Item	Pointer to Header Node
f	NULL
c	NULL
a	NULL
b	NULL
m	NULL
p	NULL

T#1 {f, c, a, m, p}

f	1	NULL

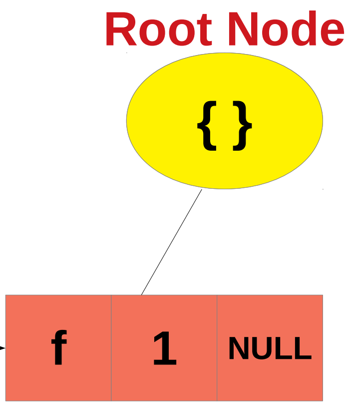
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	NULL
a	NULL
b	NULL
m	NULL
p	NULL

T#1 {f, c, a, m, p}



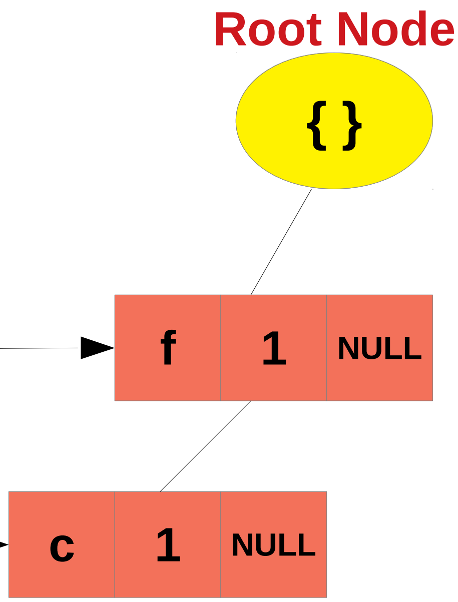
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	NULL
b	NULL
m	NULL
p	NULL

T#1 {f, c, a, m, p}

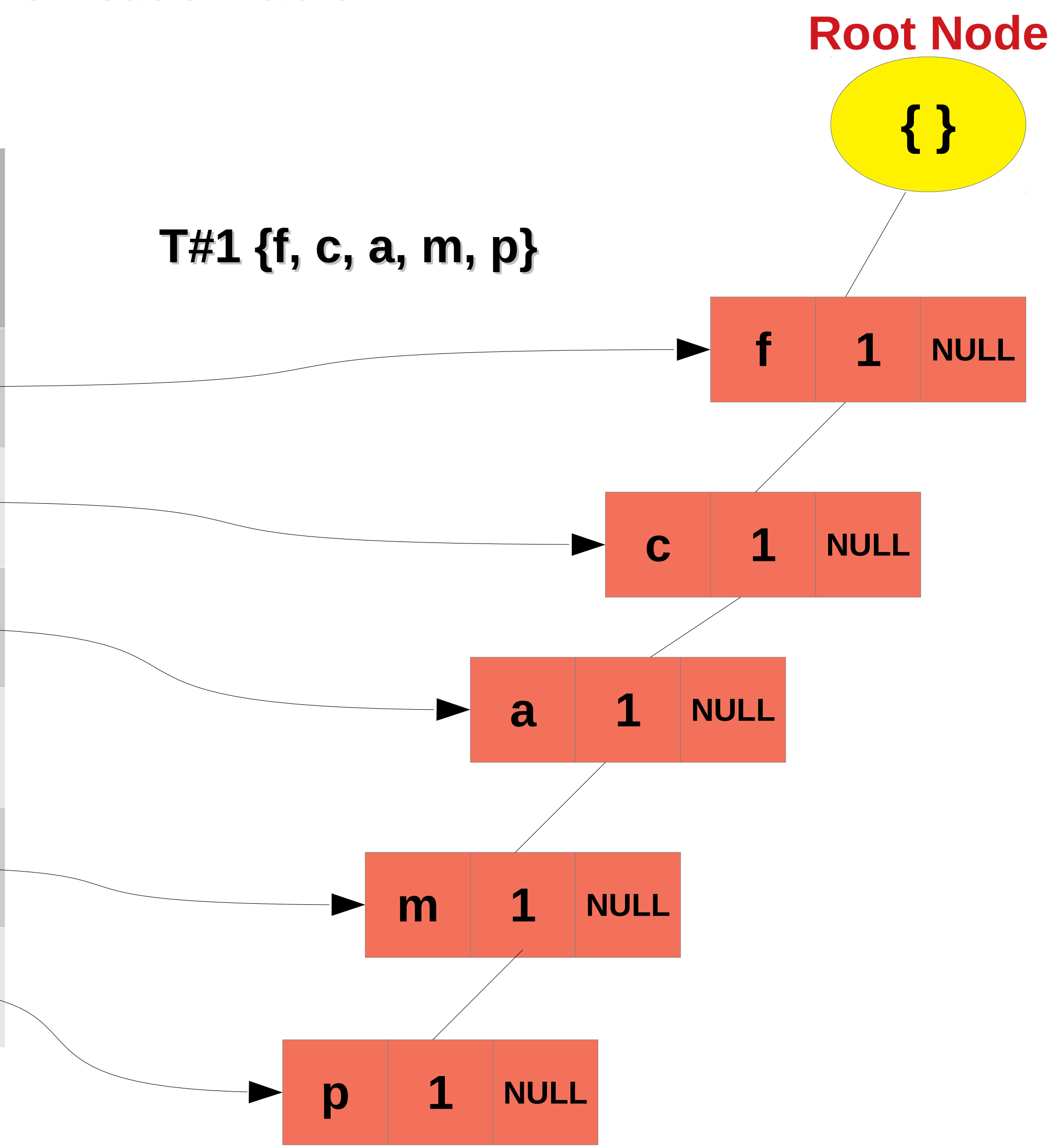


STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	NULL
m	
p	



Success is the sum of small efforts repeated day in and day out

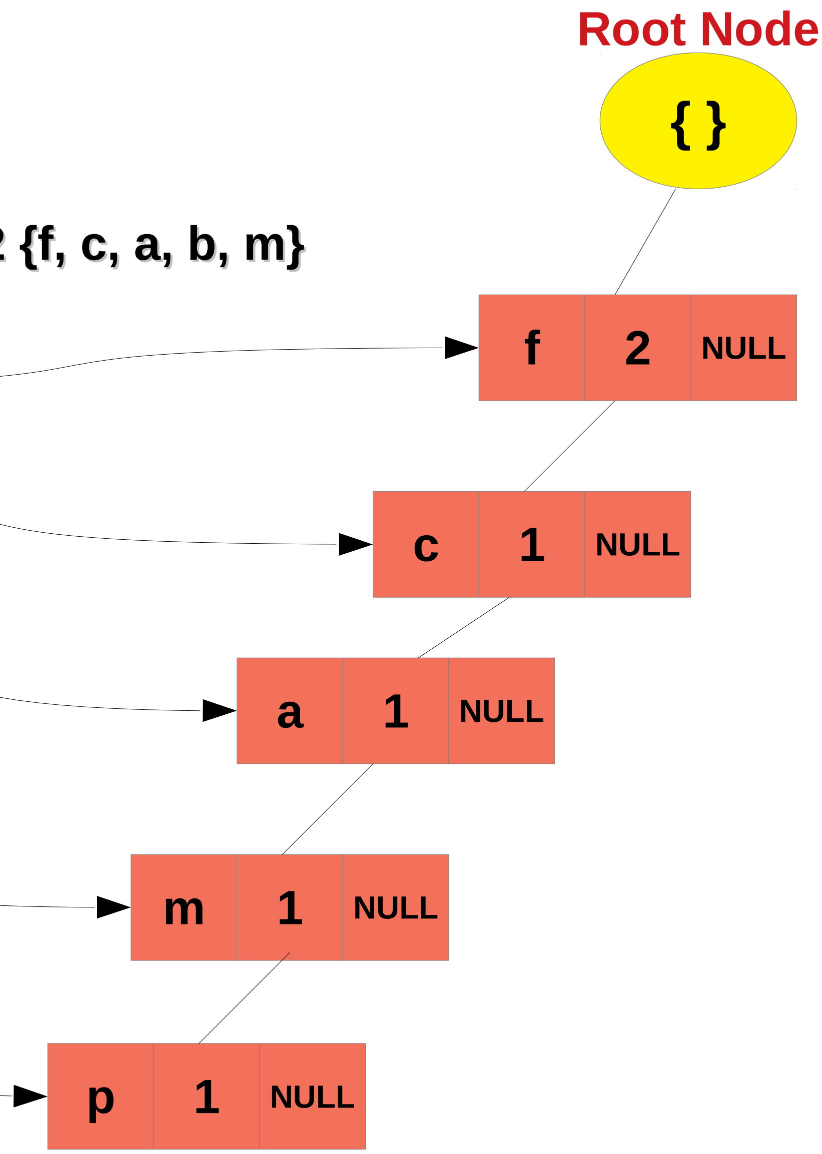
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	NULL
m	
p	

T#2 {f, c, a, b, m}



Success is the sum of small efforts repeated day in and day out

STEP 4

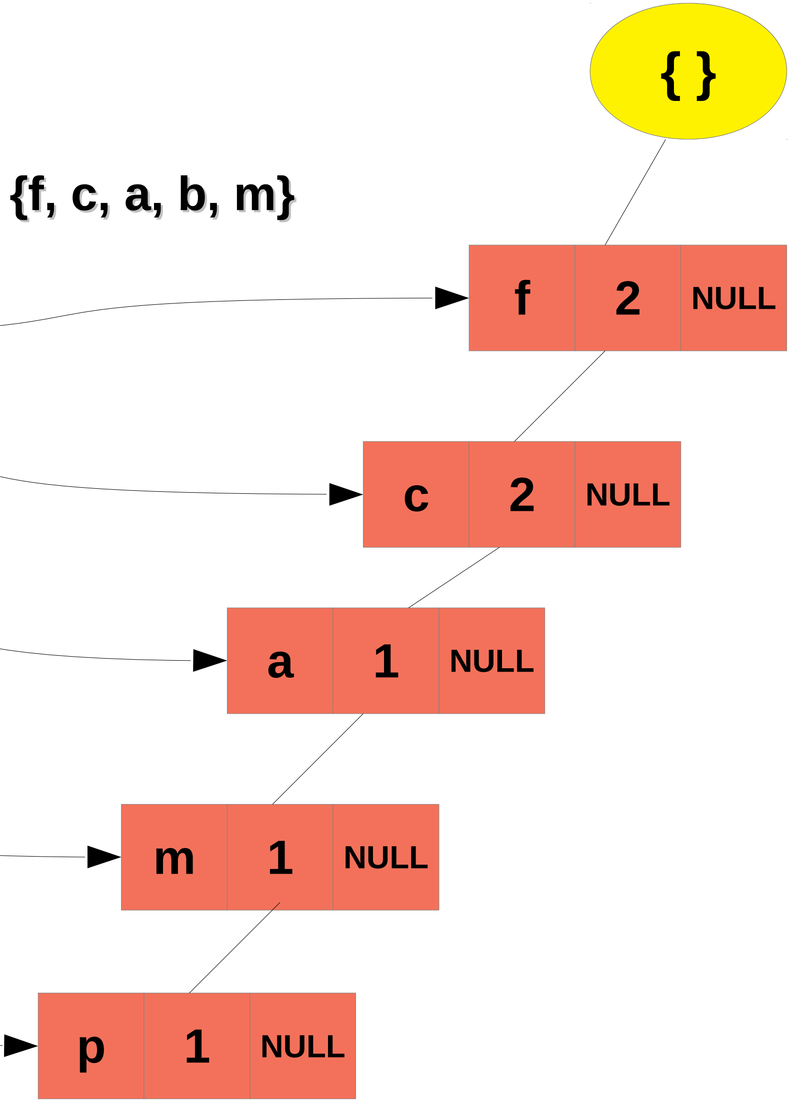
Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	NULL
m	
p	

T#2 {f, c, a, b, m}

Root Node



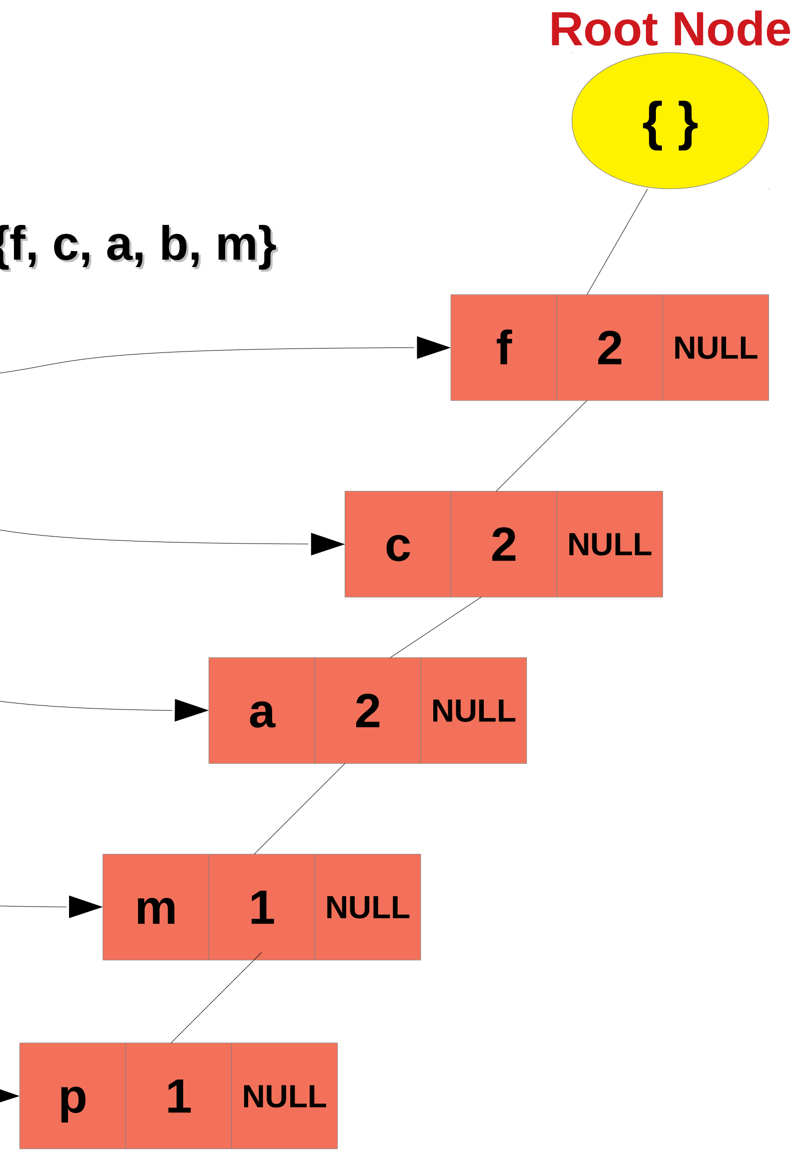
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	NULL
m	
p	

T#2 {f, c, a, b, m}

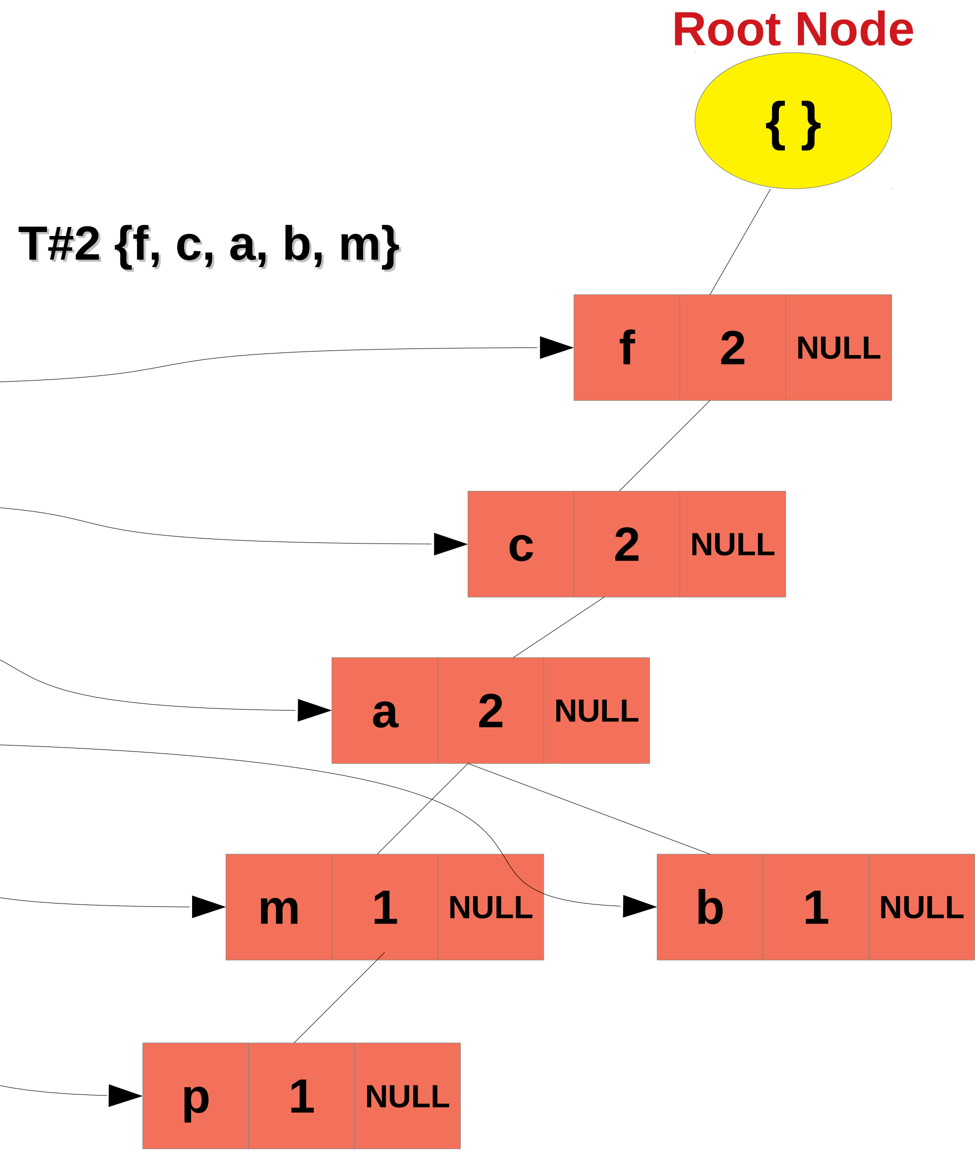


STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	

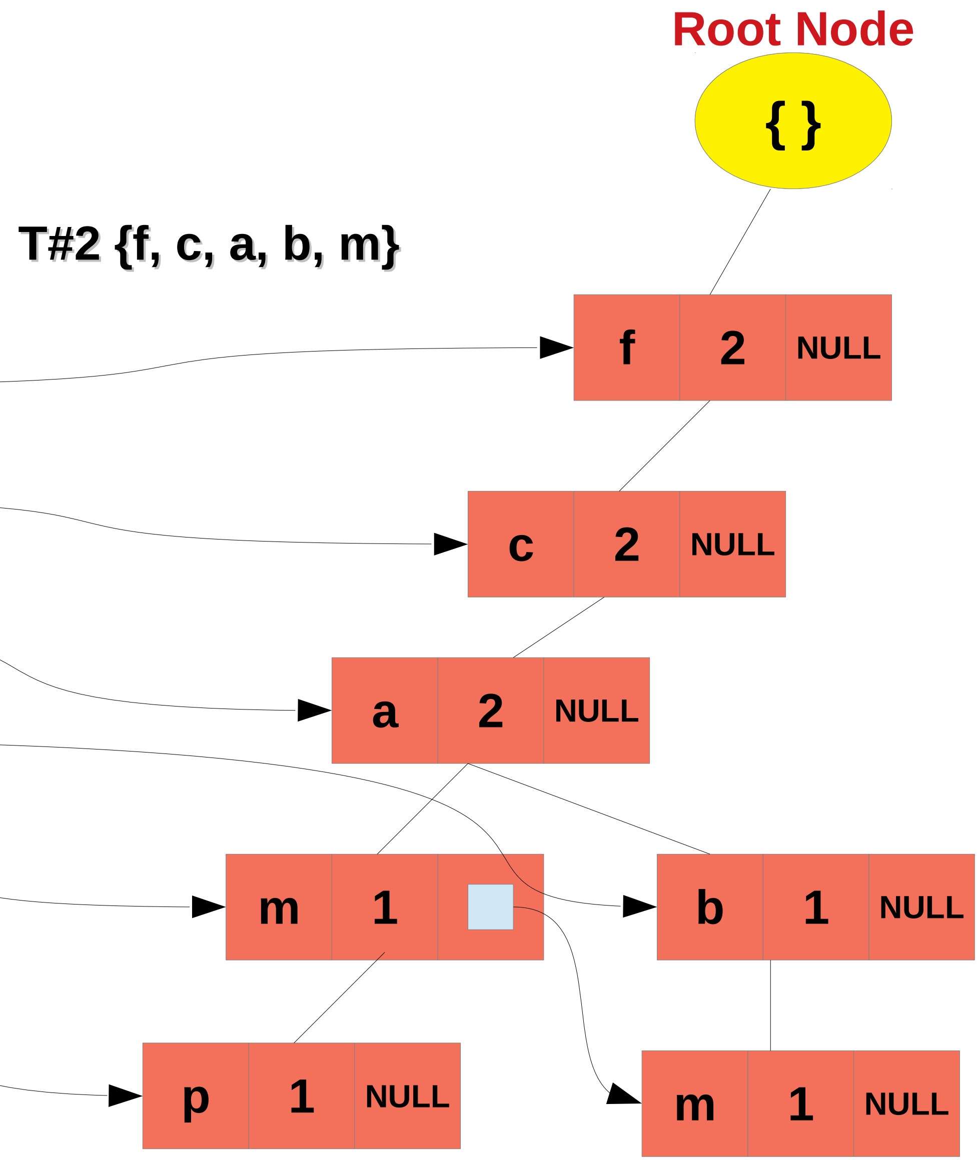


STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	



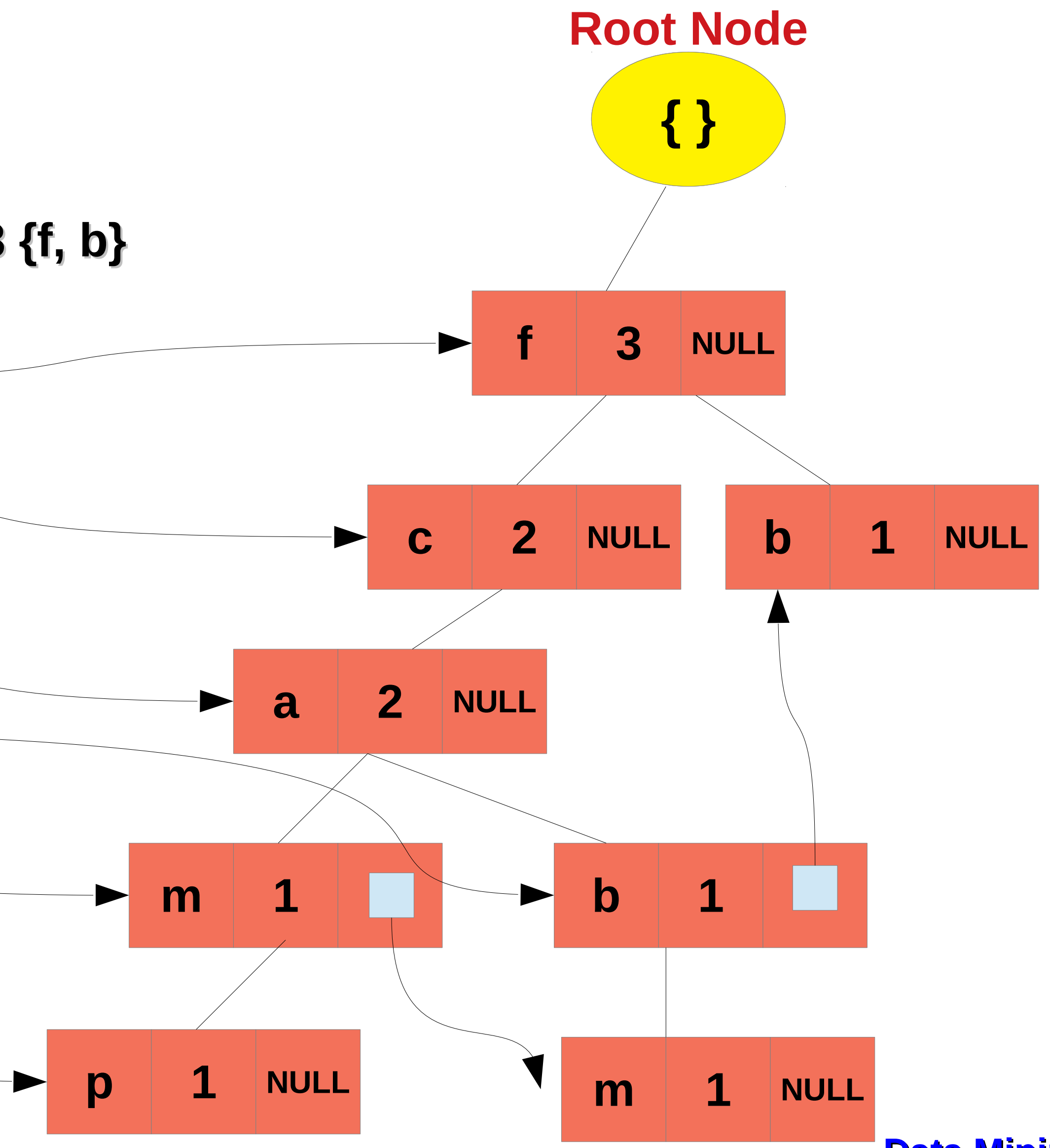
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	

T#3 {f, b}



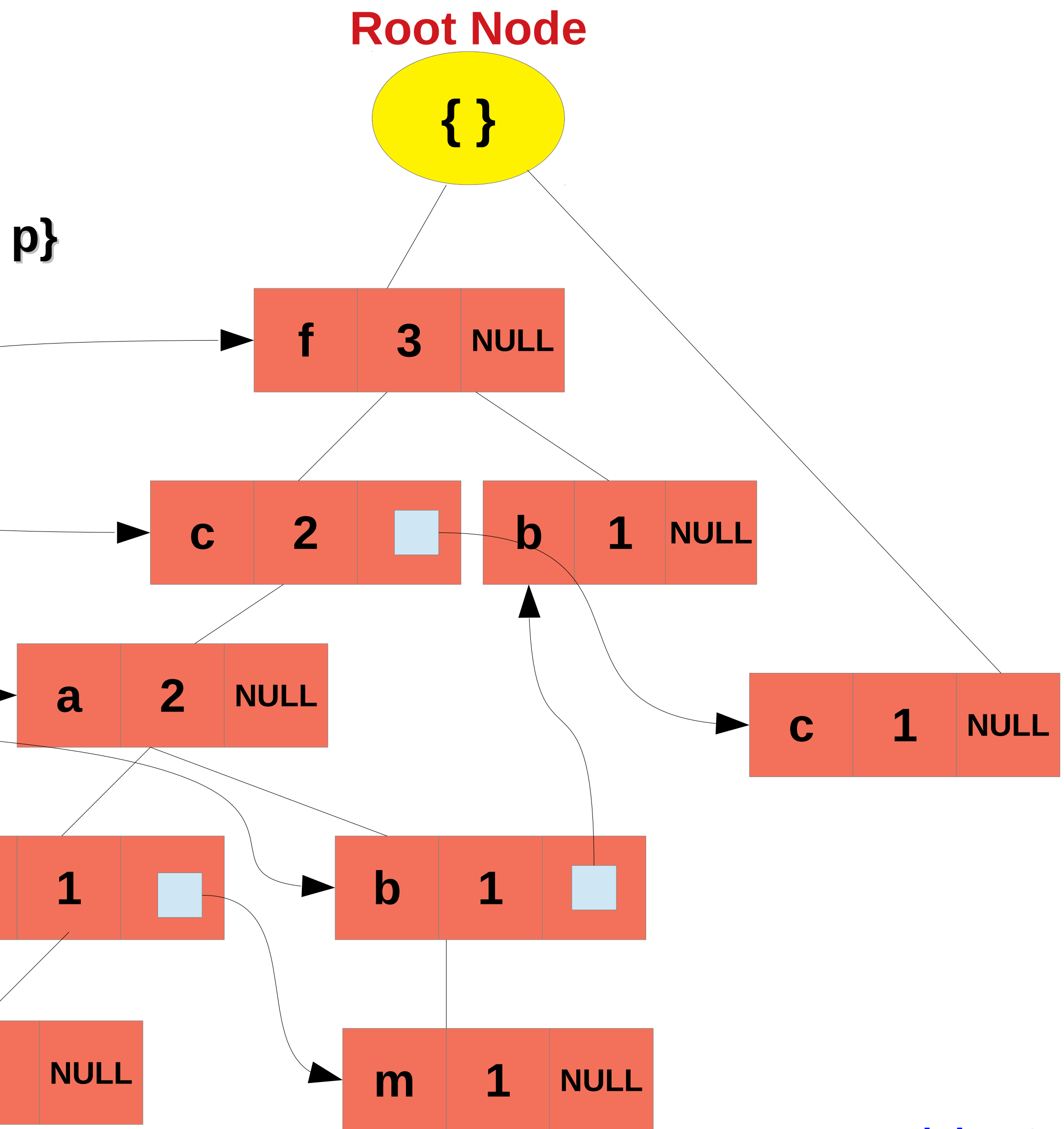
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	

T#4 {c, b, p}

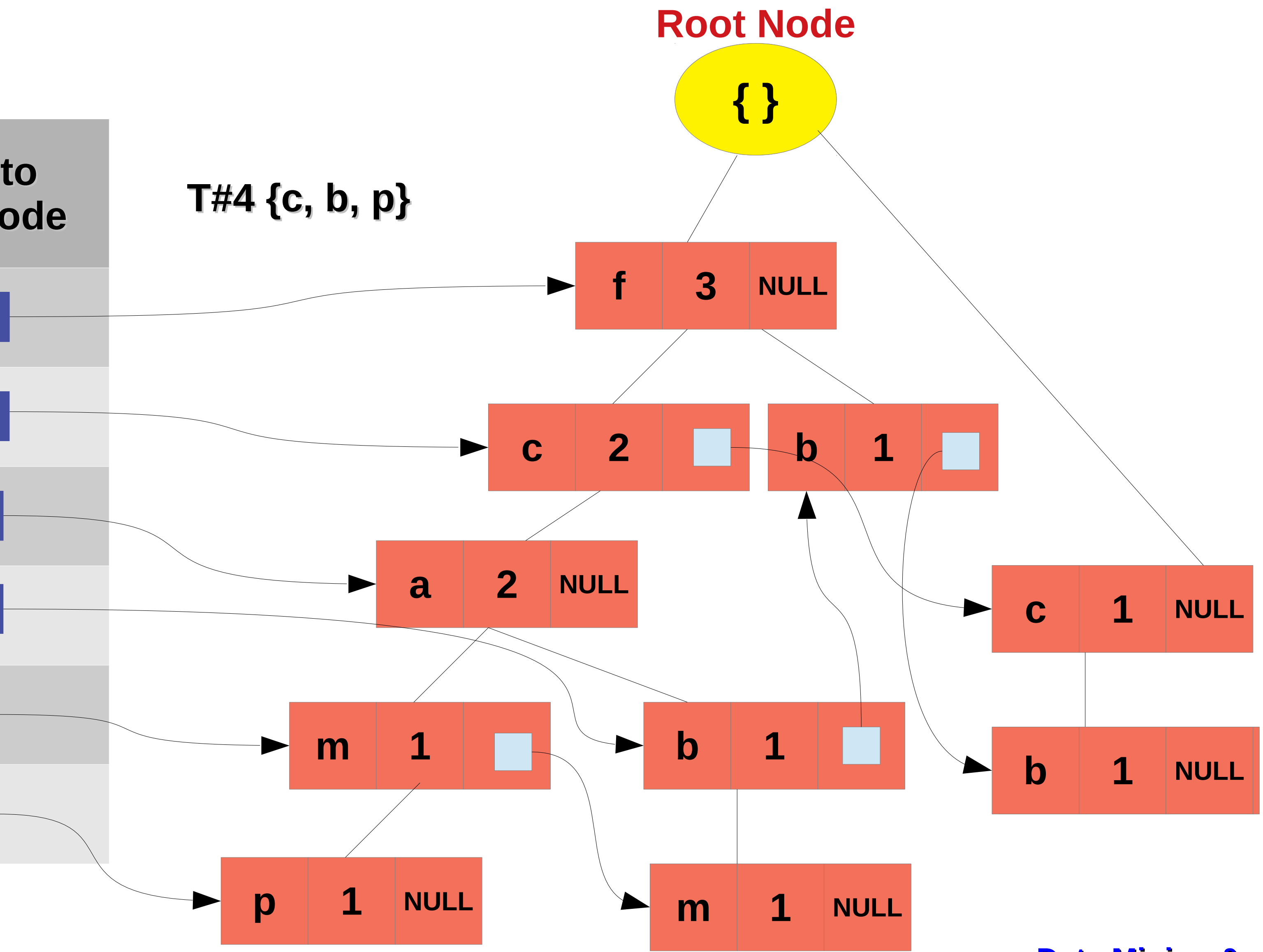


STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	



Success is the sum of small efforts repeated day in and day out

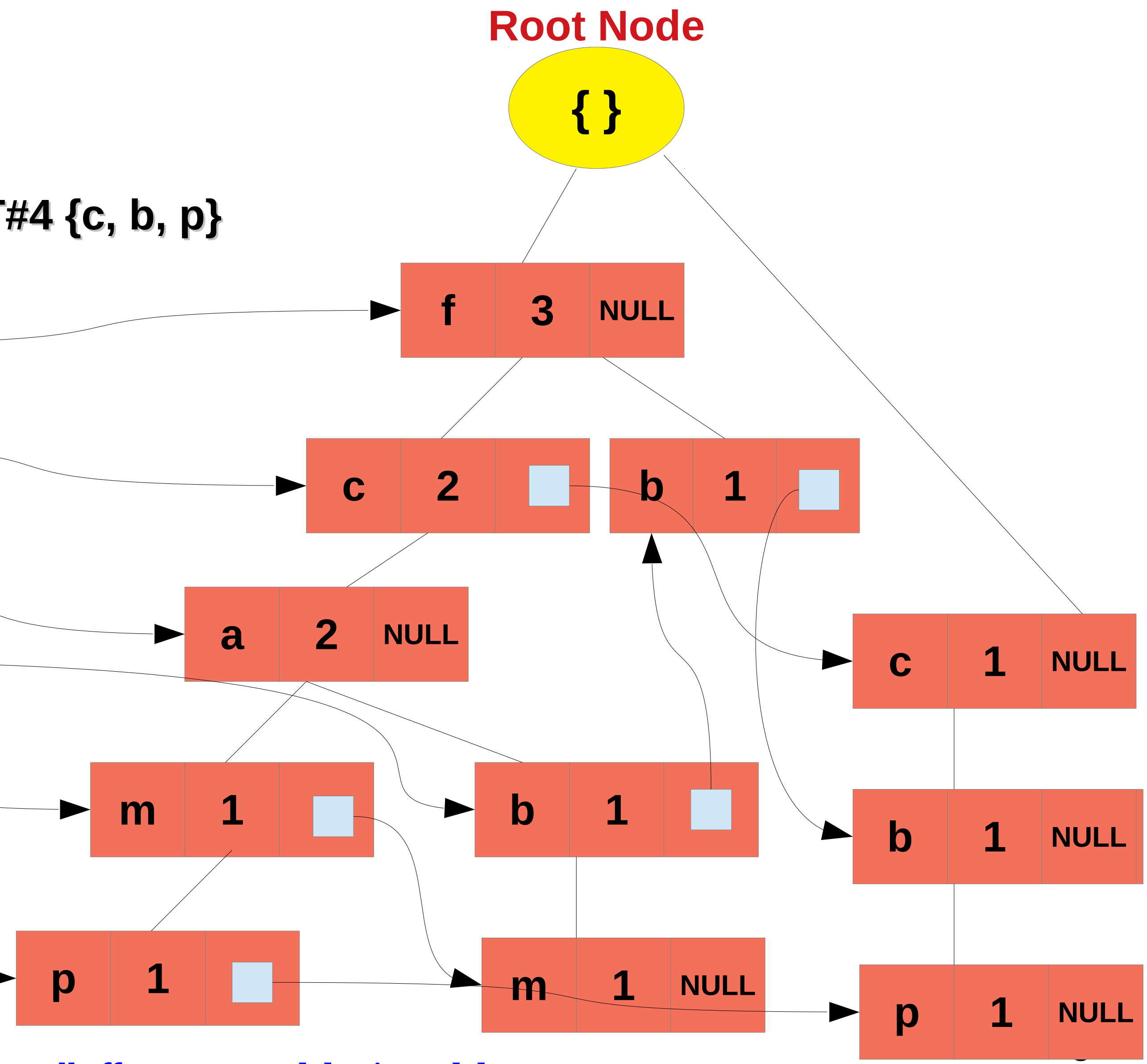
STEP 4

Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	

T#4 {c, b, p}



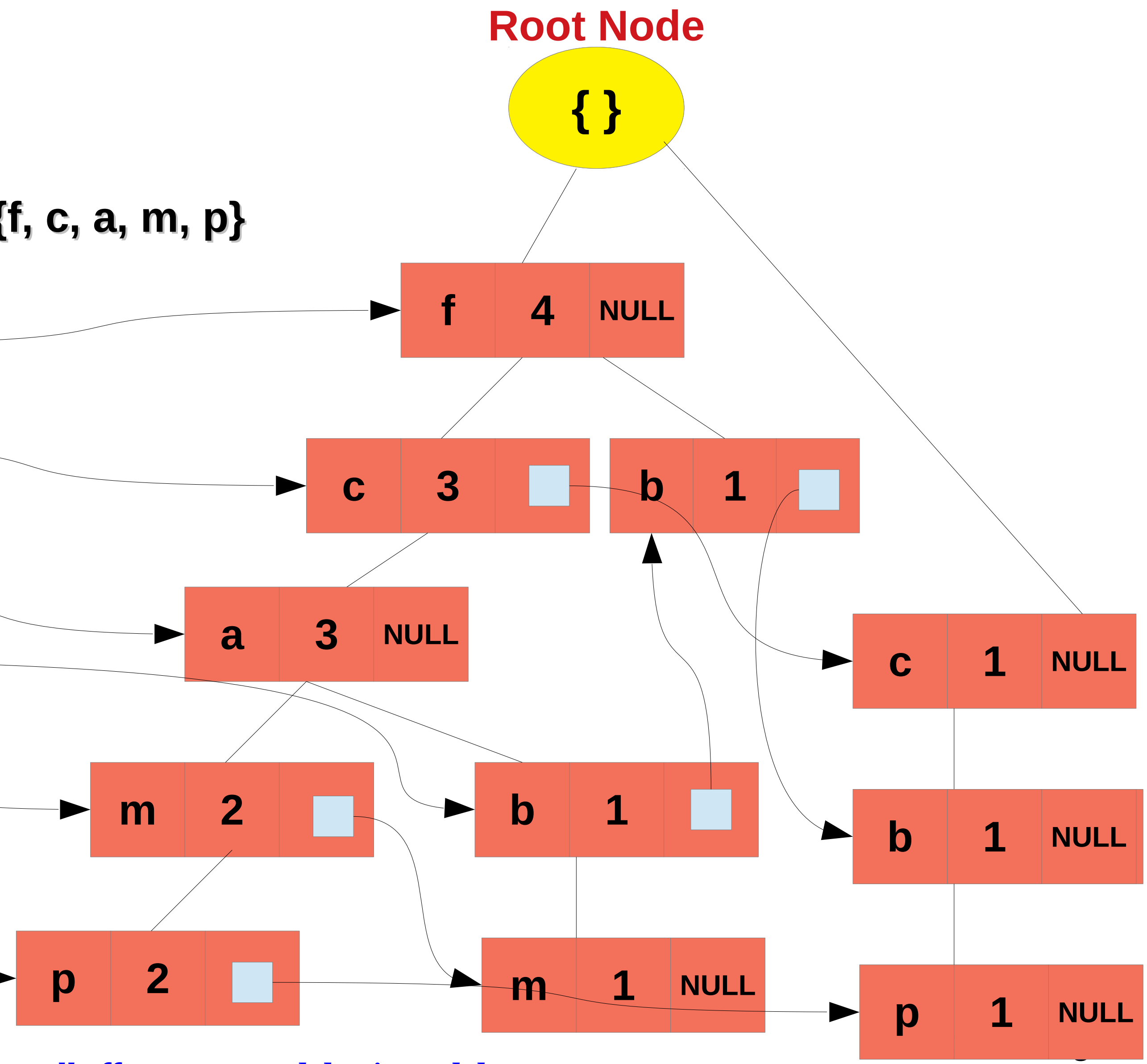
STEP 4

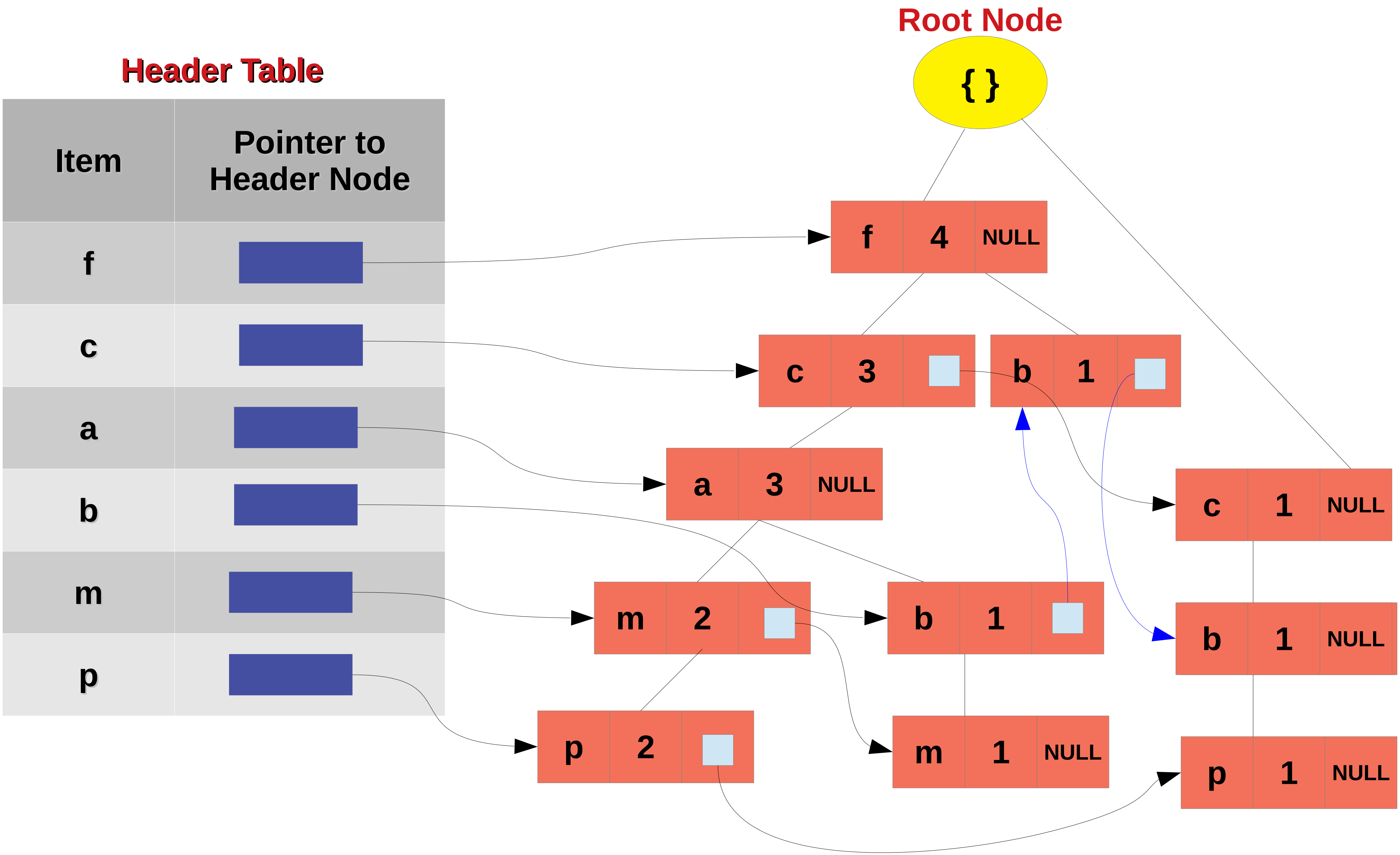
Read each transaction and start creating the nodes of the FP tree with the Header Table

Header Table

Item	Pointer to Header Node
f	
c	
a	
b	
m	
p	

T#5 {f, c, a, m, p}



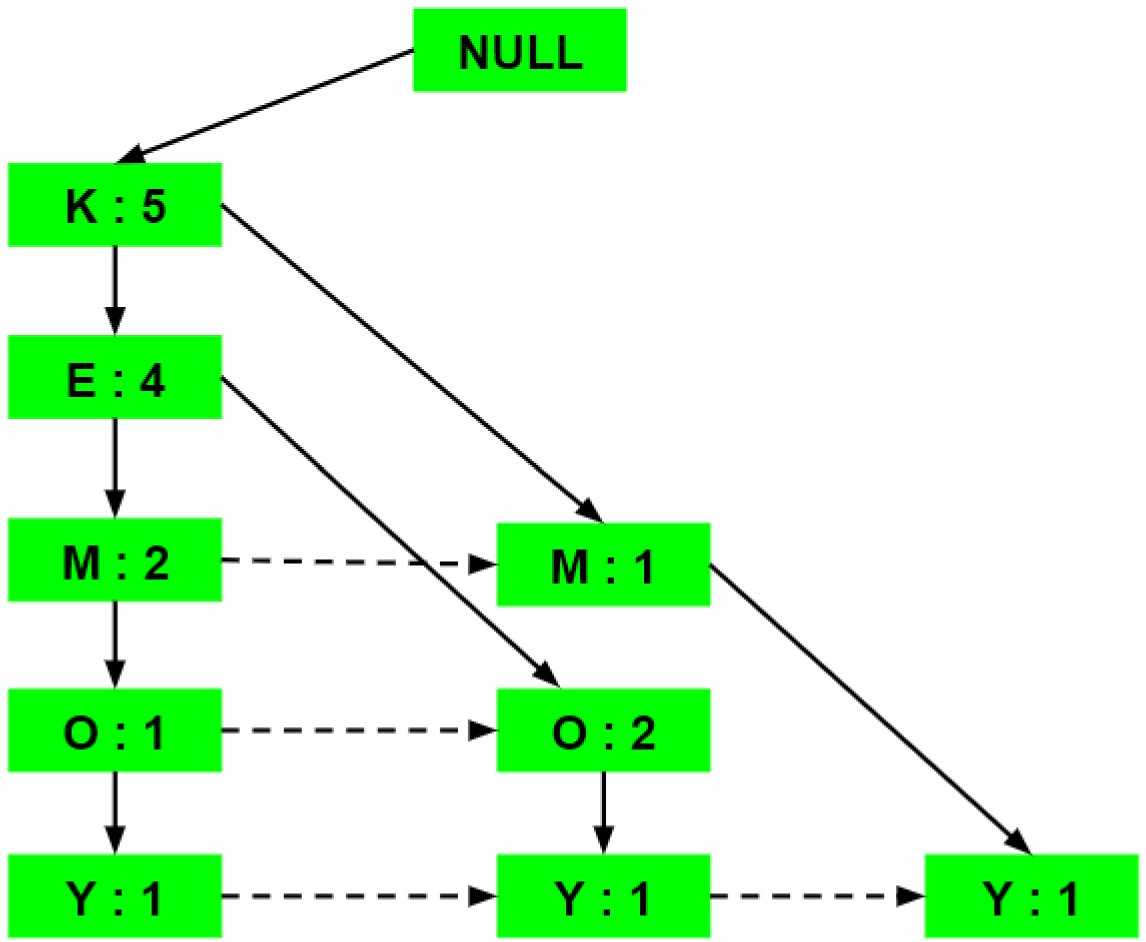


Header Table
Item
Pointer to Header Node
f
c
a
b
m
p
Root Node
{ }
f 4 NULL
c 3
b 1
a 3 NULL
c 1 NULL
m 2
b 1
b 1 NULL
p 2
m 1 NULL
p 1 NULL

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Home Work : Construct the FP tree for the given DB

Minimum support = 3


NULL
K : 5
E : 4
M : 2
M : 1
O : 1
O : 2
Y : 1
Y : 1
Y : 1

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

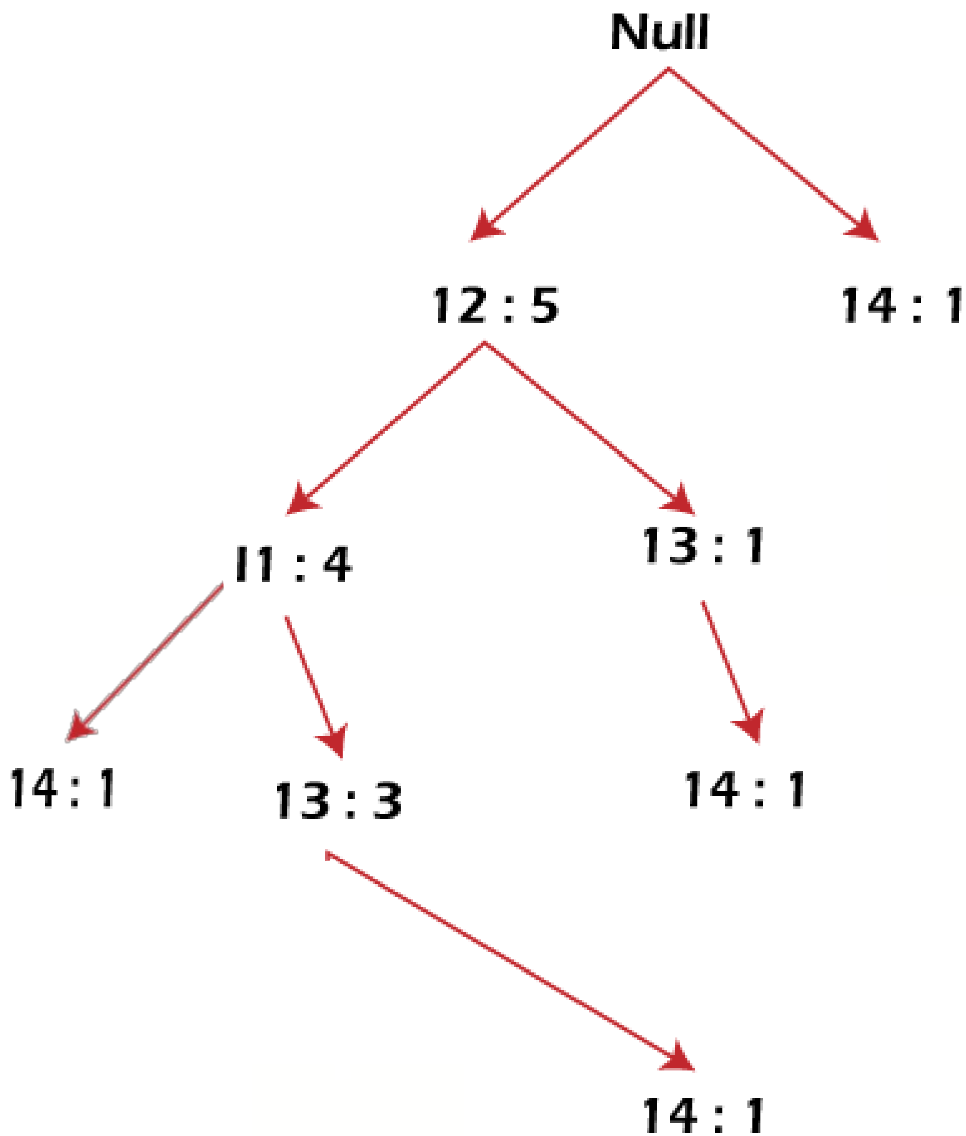
Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

Table 1:

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution: Support threshold=50% => 0.5*6= 3 => min_sup=3


Null
I2 : 5
I4 : 1
I1 : 4
I3 : 1
I4 : 1
I3 : 3
I4 : 1
I4 : 1

FP-Tree construction algo

- Construct_Tree([p|P], T)
  - If T has a child N, where N.item = p
    - Then increment N.count by one
  - else create new node N with N.count = 1
    - Link it up from the header table
  - If P is nonempty call Construct_Tree([p|P], N)
- p is each item in transaction P
- Construct_tree is called with (transaction P, root node)

FP Growth Algorithm

- Two phases
  - Phase I
    - Construction of FP Tree
  - Phase II
    - Mine the FP Tree to generate Frequent Patterns

Phase 2

Mine the FP Tree

Mine the FP tree and conditional FP trees

Mining Frequent Patterns Using FP tree

- **General idea (divide and conquer)**
  - **Recursively grow frequent patterns using the FP tree:**
    - **For each frequent item, construct its conditional pattern base, and then its conditional FP tree;**
    - **Repeat the process on each newly created conditional FP tree until**
      - **the resulting FP tree is empty,**
      - **or it contains only one path**
        - **(single path will generate all the combinations of its sub paths, each of which is a frequent pattern)**

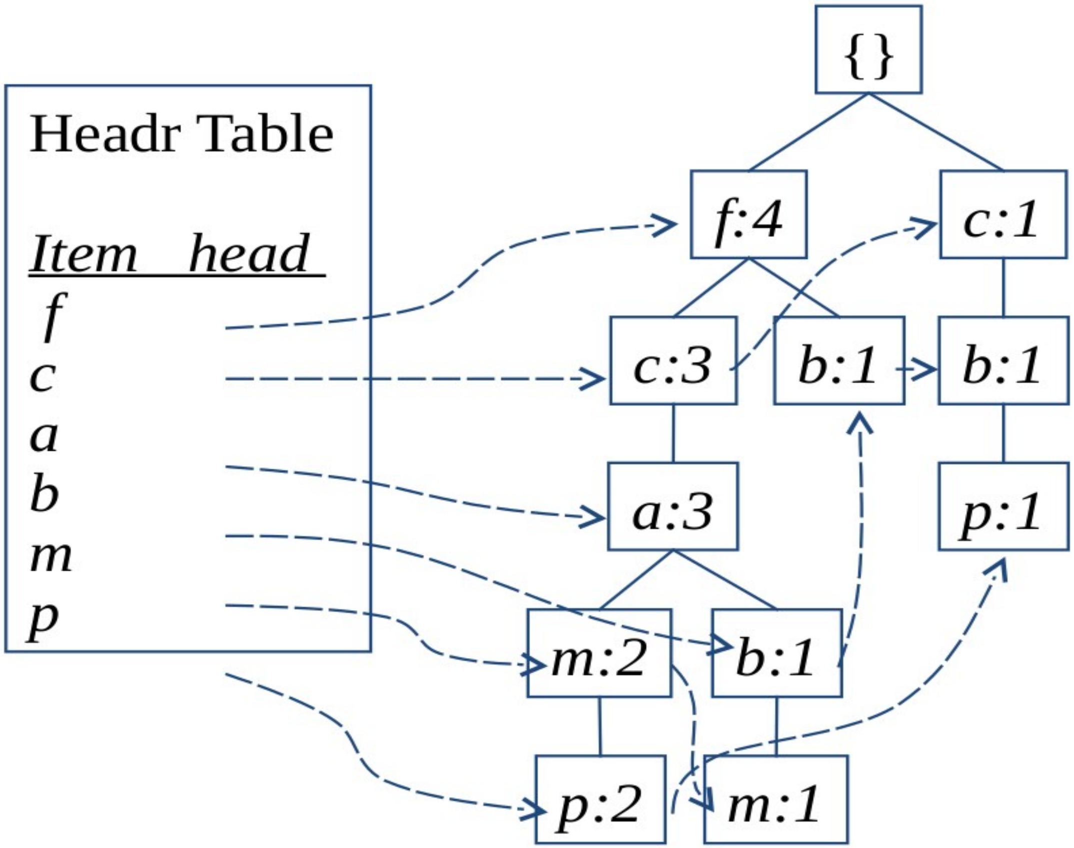
Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node.

- **Step 1:**
  - **Divide the main FP tree into conditional FP trees**
    - **Starting from each frequent 1-pattern, we create conditional pattern bases with the set of prefixes in the FP tree.**
    - **Then, we use those pattern bases to construct conditional FP trees**

- **Step 2:**
  - **Mine each conditional FP trees recursively**
    - **The frequent patterns are generated from the conditional FP Trees.**
    - **One conditional FP tree is created for one frequent pattern.**

Example

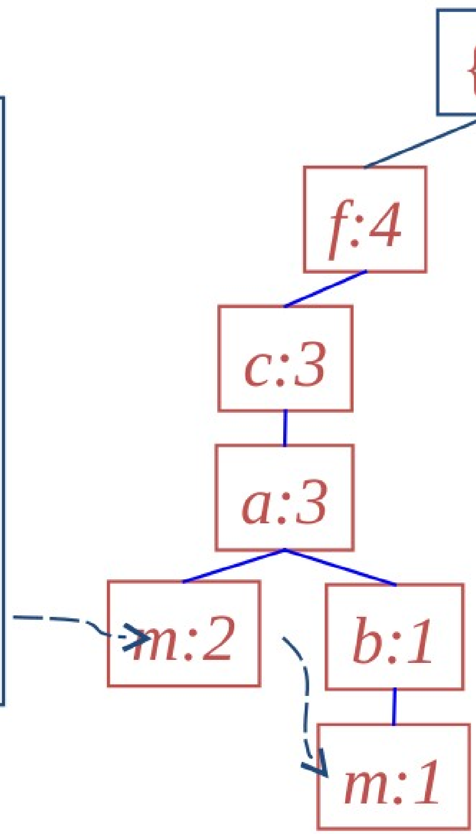
- **Therefore to mine the FP tree………….**



- **Find the *conditional pattern base***
  - **the lowest node is considered**
    - **The lowest node represents the frequency pattern of length 1.**
  - **From this, traverse the prefix path in the FP Tree.**
  - **This path or paths are called a *conditional pattern base.***

Header Table

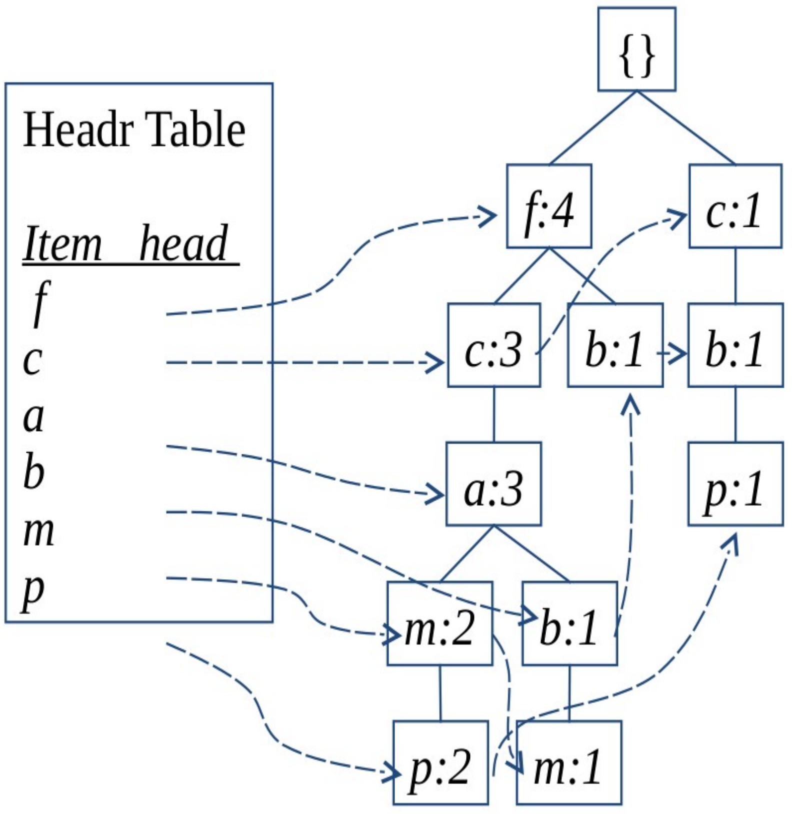
Item	*head*
f	4
c	4
a	3
b	3
m	3
p	3



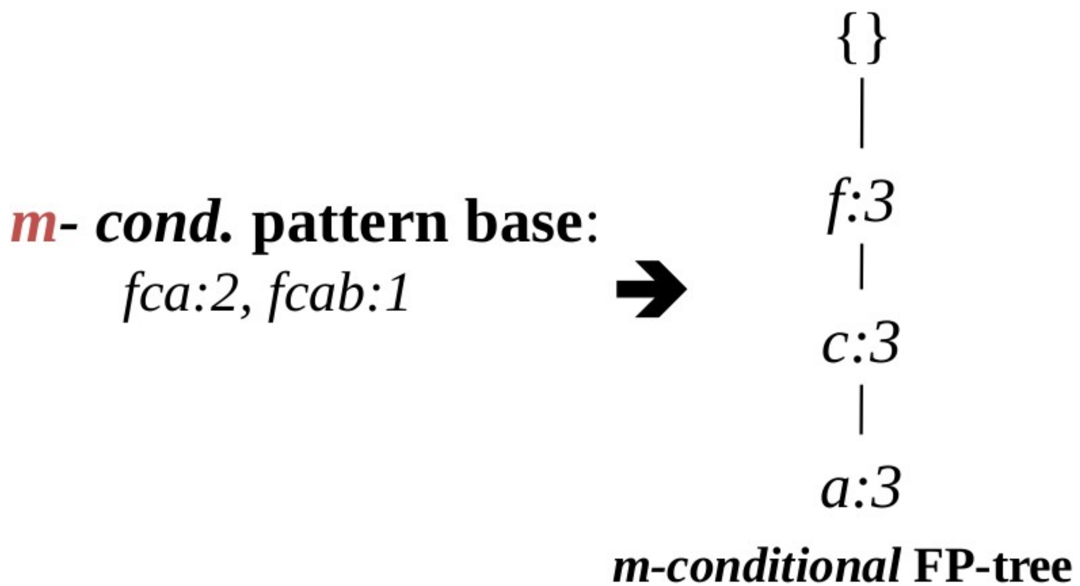
➔ ***m*- *cond.* pattern base:**

fca:2, fcab:1

Item	Conditional pattern base
p	{(fcam:2), (cb:1)}
m	{(fca:2), (fcab:1)}
b	{(fca:1), (f:1), (c:1)}
a	{(fc:3)}
c	{(f:3)}
f	Empty



- Construct *Conditional FP Tree* from the pattern bases
  - Construct a Conditional FP Tree, which is formed by a count of itemsets in the path.
  - The itemsets meeting the threshold support are considered in the Conditional FP Tree.



Item	Conditional pattern base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

- **Generate *Frequent Patterns* from the Conditional FP Trees**
  - **Frequent Patterns are *generated* from the Conditional FP Tree.**
  - **Recursively mine the conditional FP-tree**

Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P. The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P

{}
|
f:3
|
c:3
|
a:3

m-conditional FP-tree

➔

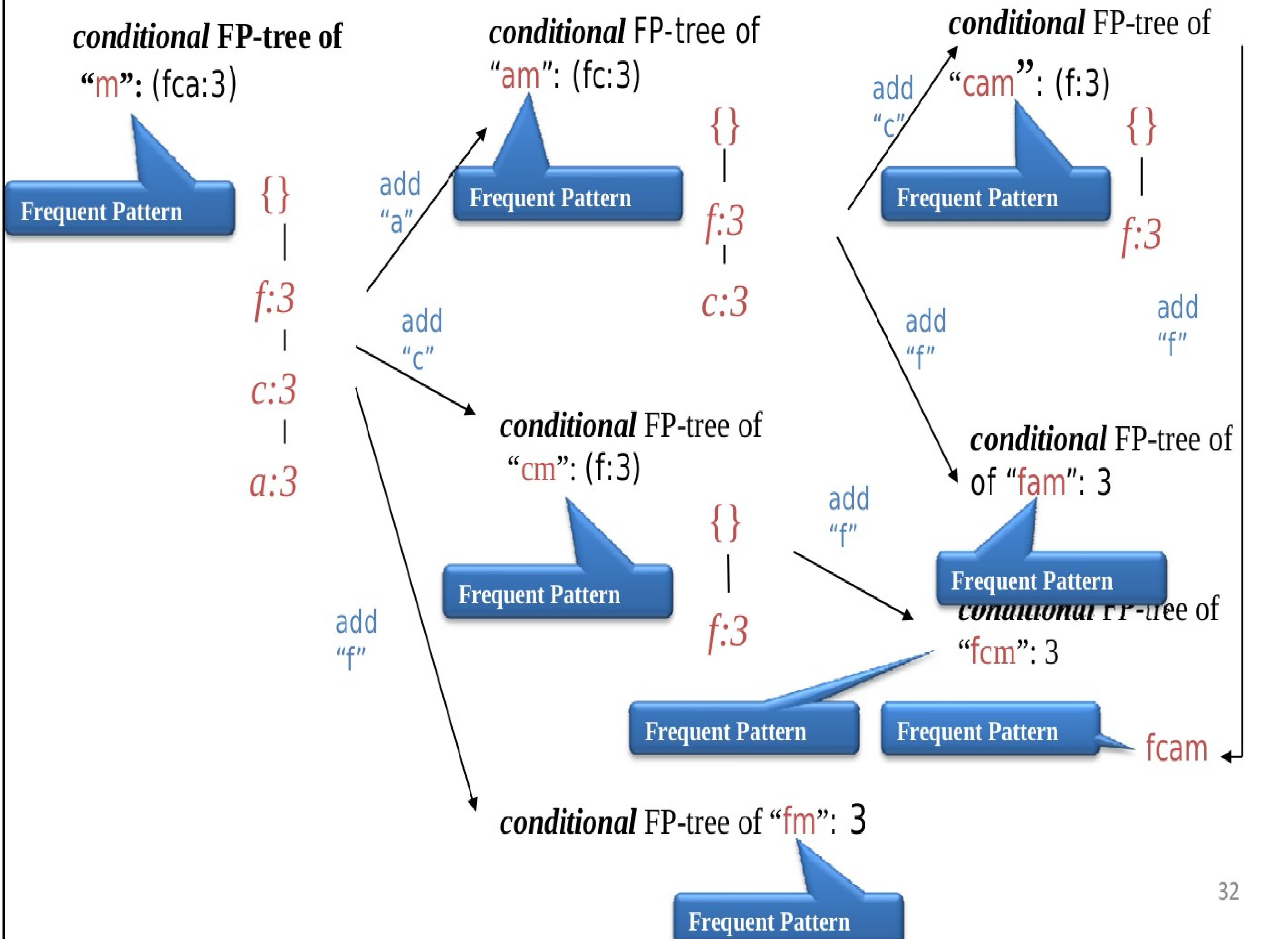
All frequent patterns concerning *m*: combination of {f, c, a} and *m*

m,

fm, *cm*, *am*,

fcm, *fam*, *cam*,

fcam

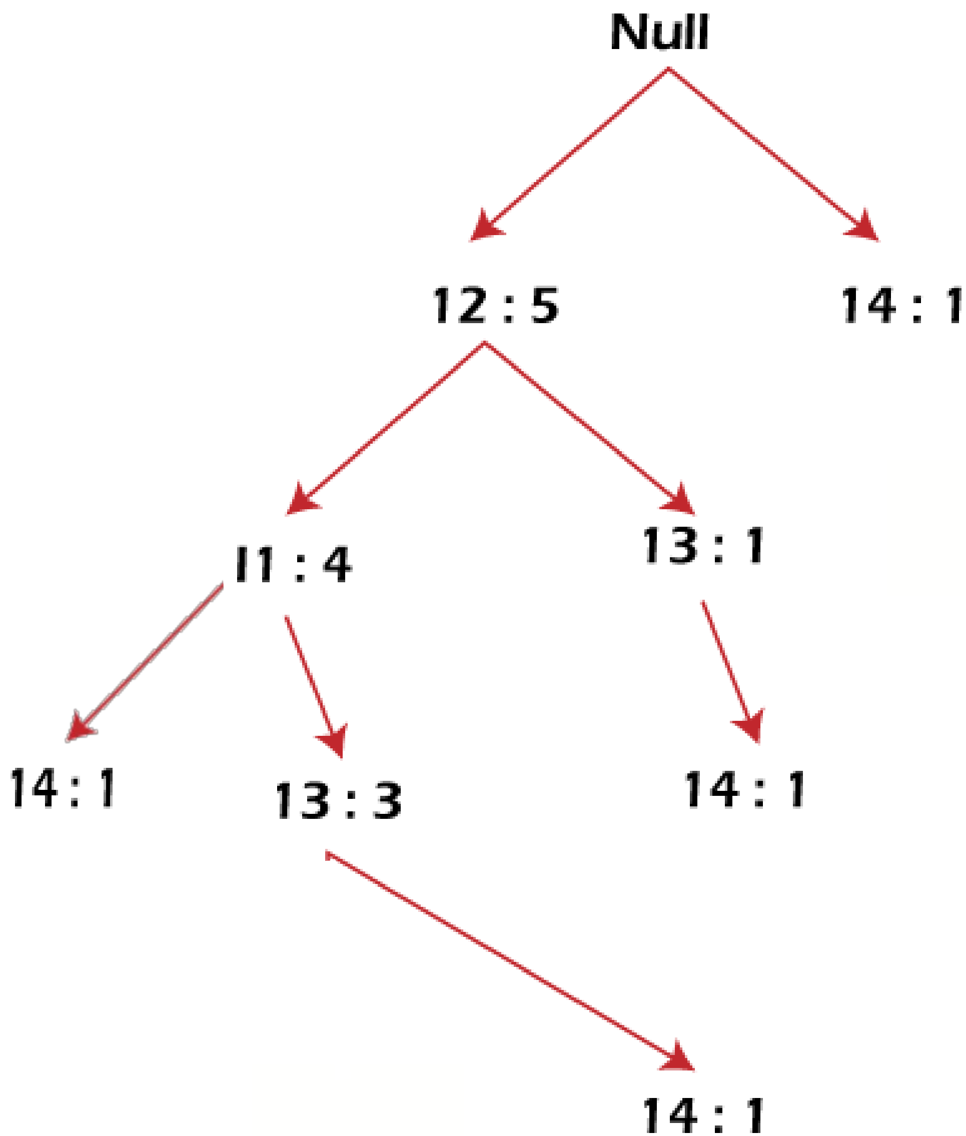


conditional FP-tree of "m": (fca:3)
Frequent Pattern
{}
f:3
c:3
a:3
add "a"
conditional FP-tree of "am": (fc:3)
Frequent Pattern
{}
f:3
c:3
add "c"
conditional FP-tree of "cam": (f:3)
Frequent Pattern
{}
f:3
add "f"
add "c"
conditional FP-tree of "cm": (f:3)
Frequent Pattern
{}
f:3
add "f"
conditional FP-tree of of "fam": 3
Frequent Pattern
add "f"
conditional FP-tree of "fcm": 3
Frequent Pattern
Frequent Pattern
fcam
add "f"
conditional FP-tree of "fm": 3
Frequent Pattern

Table 1:

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution: Support threshold=50% => 0.5*6= 3 => min_sup=3


Null
I2 : 5
I4 : 1
I1 : 4
I3 : 1
I4 : 1
I3 : 3
I4 : 1
I4 : 1

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

Scan D for count of each candidate →

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate →

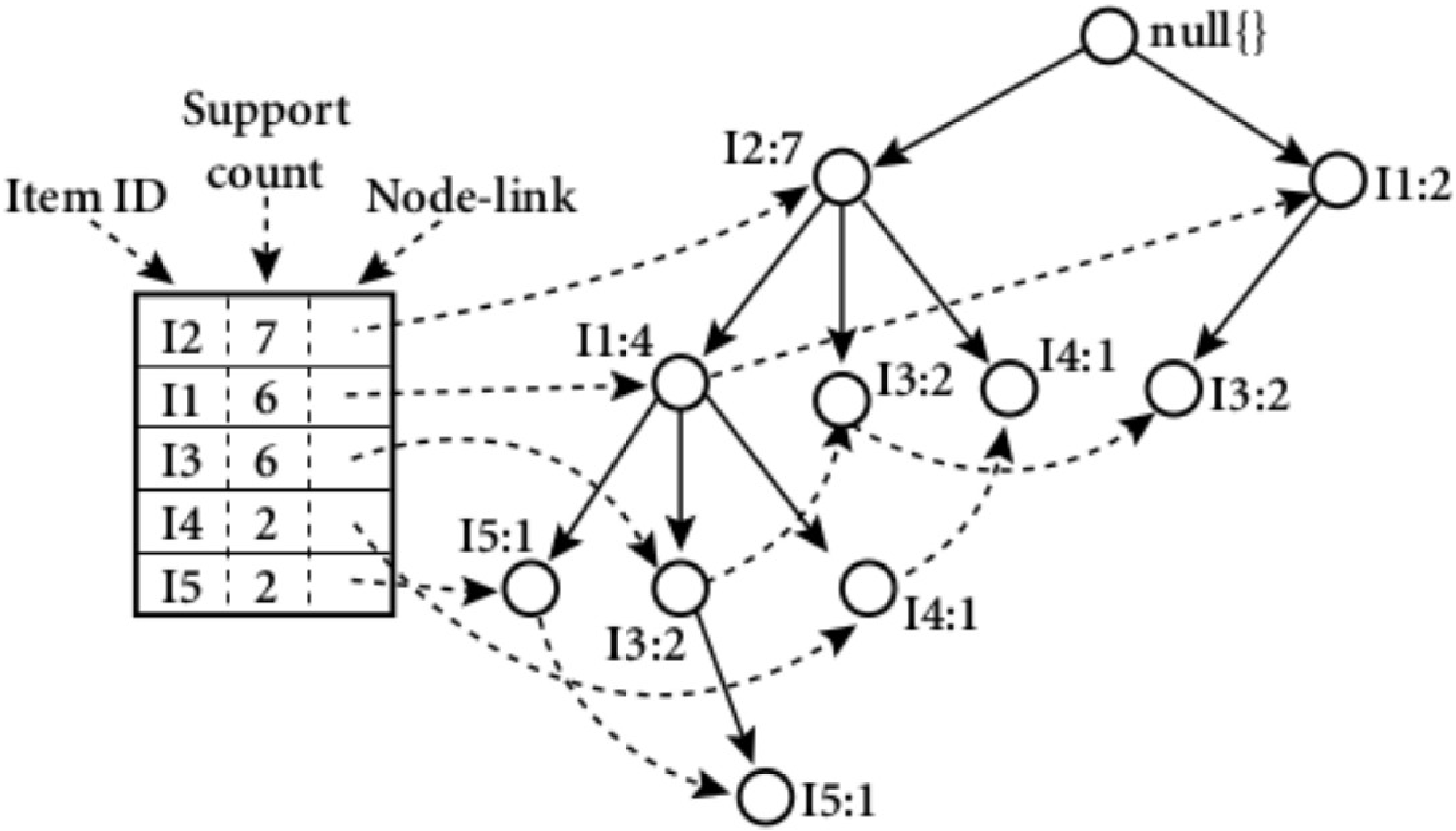
C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

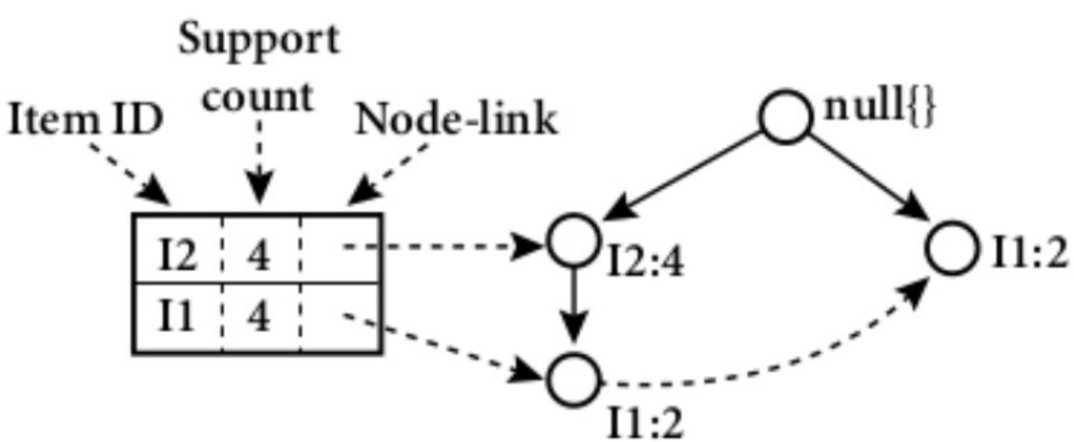
L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2


null{}
Support
count
Item ID
Node-link
I2 7
I1 6
I3 6
I4 2
I5 2
I2:7
I1:2
I1:4
I3:2
I4:1
I3:2
I5:1
I3:2
I4:1
I5:1

Mining the FP-tree by creating conditional (sub-)pattern bases.

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}



The conditional FP-tree associated with the conditional node I3.

- **Advantages**
  - **1. Faster than apriori algorithm**
  - **2. No candidate generation**
  - **3. Only two passes over dataset**
- **Disadvantages**
  - **1. FP tree may not fit in memory**
  - **2. FP tree is expensive to build**