

# Chapter 6. Frequent Pattern Mining: FP-Growth

Meng Jiang

CSE 40647/60647 Data Science Fall 2017

Introduction to Data Mining





12 LONGNECK BOTTLES 12 FL. OZ.

# Corona

Extra

SINCE 1925

LA CERVEZA MAS FINA

IMPORTED BEER FROM MEXICO  
CERVECERIA MODELO, NAVA, MEXICO

Pampers Swaddlers

32

32

1

1



# Pattern Discovery: Definition

- What are patterns?
  - Patterns: A set of items, subsequences, or substructures that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets
- Pattern discovery: Uncovering patterns from massive data
- Motivation examples:
  - **What products were often purchased together?**
  - What are the subsequent purchases after buying an iPad?
  - What code segments likely contain copy-and-paste bugs?
  - What word sequences likely form phrases in this corpus?

# Frequent Patterns (Itemsets)

- **Itemset**: A set of one or more items
- **k-itemset**:  $X = \{x_1, \dots, x_k\}$
- **(absolute) support (count)** of X: Frequency or the number of occurrences of an itemset X
- **(relative) support**,  $s$ : The fraction of transactions that contains X (i.e., the **probability** that a transaction contains X)
- An itemset X is **frequent** if the support of X is no less than a *minsup* threshold

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Let *minsup* = 50%

Freq. 1-itemsets:

Beer: 3 (60%); Nuts: 3 (60%)

Diaper: 4 (80%); Eggs: 3 (60%)

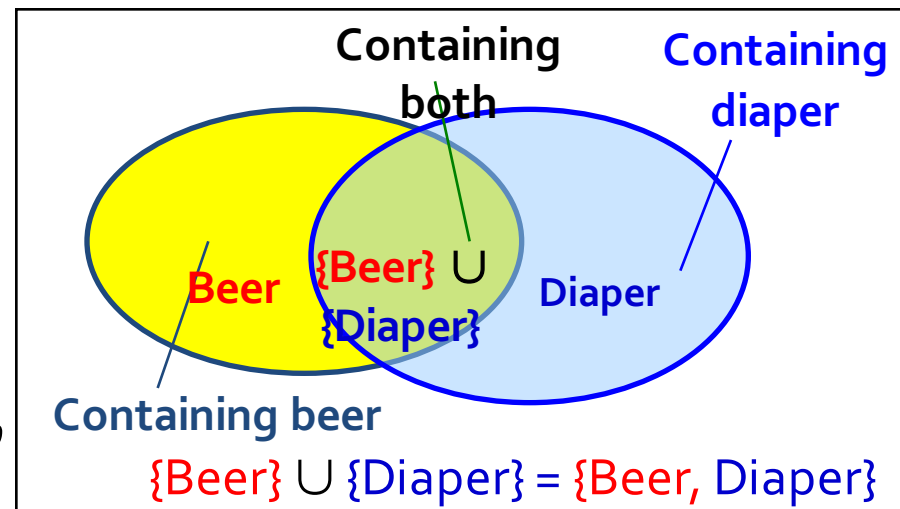
Freq. 2-itemsets:

{Beer, Diaper}: 3 (60%)

# From Frequent Itemsets to Association Rules

- Association rules:  $X \rightarrow Y (s, c)$ 
  - **Support**,  $s$ : The probability that a transaction contains  $X \cup Y$
  - **Confidence**,  $c$ : The conditional probability that a transaction containing  $X$  also contains  $Y$
  - $c = \text{sup}(X \cup Y) / \text{sup}(X)$
- **Association rule mining**: Find **all** of the rules,  $X \rightarrow Y$ , with minimum support and confidence
- Frequent itemsets: Let  $\text{minsup} = 50\%$ 
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets:  $\{\text{Beer}, \text{Diaper}\}$ : 3
- Association rules: Let  $\text{minconf} = 50\%$ 
  - $\text{Beer} \rightarrow \text{Diaper}$  (60%, 100%)
  - $\text{Diaper} \rightarrow \text{Beer}$  (60%, 75%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Note: Itemset:  $X \cup Y$ , a subtle notation!

# Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
  - How many frequent itemsets does the following  $TDB_1$  contain?
    - $TDB_1: T_1: \{a_1, \dots, a_{50}\}; T_2: \{a_1, \dots, a_{100}\}$
    - Assuming (absolute)  $minsup = 1$
    - Let's have a try
- 1-itemsets:  $\{a_1\}: 2, \{a_2\}: 2, \dots, \{a_{50}\}: 2, \{a_{51}\}: 1, \dots, \{a_{100}\}: 1,$
- 2-itemsets:  $\{a_1, a_2\}: 2, \dots, \{a_1, a_{50}\}: 2, \{a_1, a_{51}\}: 1 \dots, \dots, \{a_{99}, a_{100}\}: 1, \dots$
- 99-itemsets:  $\{a_1, a_2, \dots, a_{99}\}: 1, \dots, \{a_2, a_3, \dots, a_{100}\}: 1$
- 100-itemset:  $\{a_1, a_2, \dots, a_{100}\}: 1$
- In total:  $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$  sub-patterns!

A too huge set for any computer to compute or store!

# Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- Solution 1: **Closed patterns**: A pattern (itemset)  $X$  is **closed** if  $X$  is *frequent*, and there exists *no super-pattern*  $Y \supset X$ , **with the same support as  $X$** 
  - Let Transaction DB  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does  $TDB_1$  contain?
    - Two:  $P_1: "\{a_1, \dots, a_{50}\}: 2"$ ;  $P_2: "\{a_1, \dots, a_{100}\}: 1"$
- **Closed pattern** is a **lossless compression** of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say:  $"\{a_2, \dots, a_{40}\}: 2"$ ,  $"\{a_5, a_{51}\}: 1"$

# Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns**: A pattern  $X$  is a **max-pattern** if  $X$  is frequent and there exists no frequent super-pattern  $Y \supset X$ , ~~with the same support as  $X$~~
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$
  - Suppose  $minsup = 1$ . How many max-patterns does  $TDB_1$  contain?
    - One:  $P: \{a_1, \dots, a_{100}\}: 1$
- **Max-pattern** is a **lossy compression**!
  - We only know  $\{a_1, \dots, a_{40}\}$  is frequent
  - But we do not know the real support of  $\{a_1, \dots, a_{40}\}$ , ..., any more!
- Thus in many applications, mining closed-patterns is more desirable than mining max-patterns



# The Downward Closure Property of Frequent Patterns: Apriori

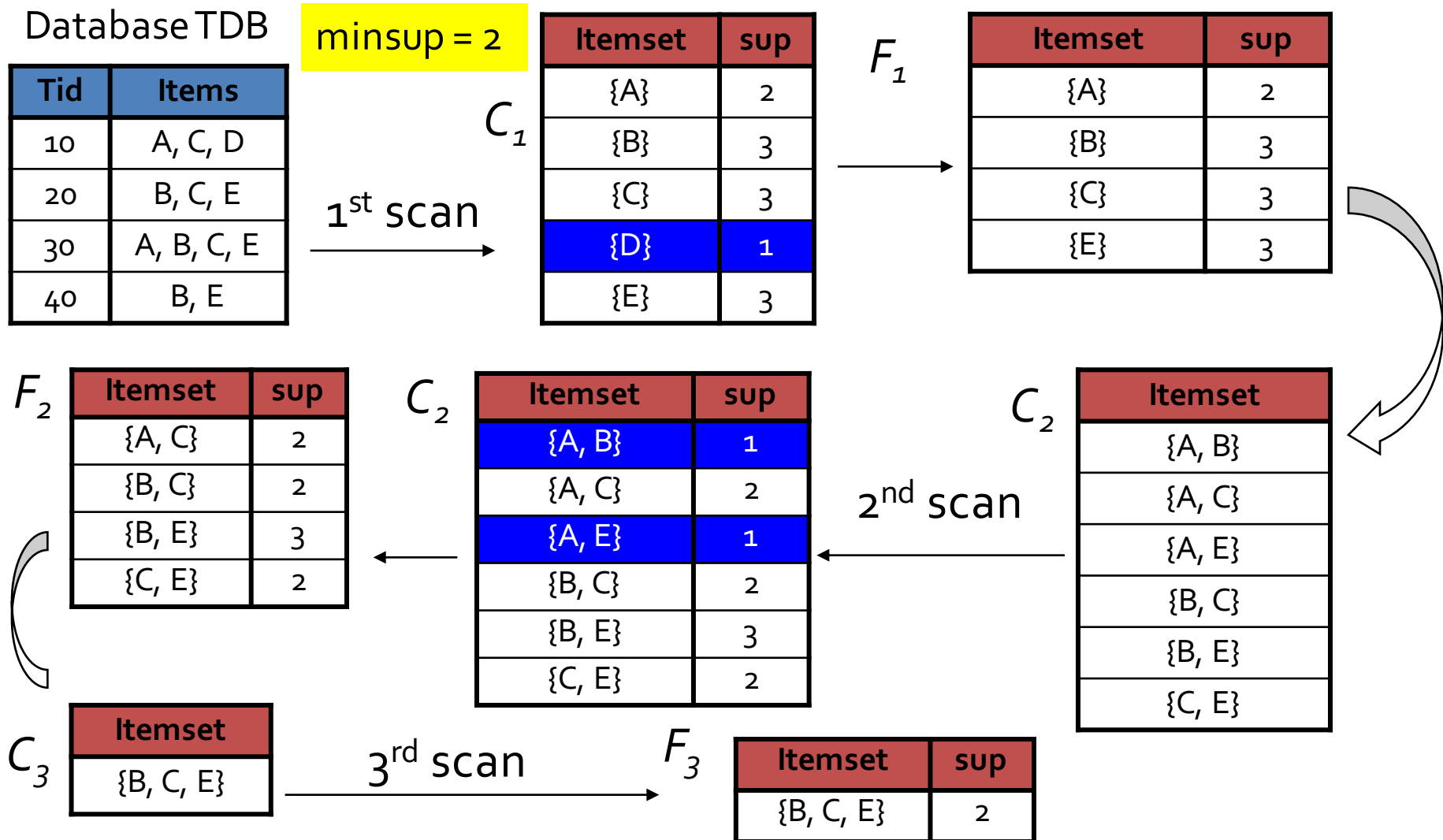
- Observation: From  $TDB_1$ :  $T_1: \{a_1, \dots, a_{50}\}$ ;  $T_2: \{a_1, \dots, a_{100}\}$ 
  - We get a frequent itemset:  $\{a_1, \dots, a_{50}\}$
  - Also, its subsets are all frequent:  $\{a_1\}, \{a_2\}, \dots, \{a_{50}\}, \{a_1, a_2\}, \dots, \{a_1, \dots, a_{49}\}, \dots$
  - There must be some hidden relationships among frequent patterns!
- The **downward closure (also called “Apriori”)** property of frequent patterns
  - If  **$\{\text{beer}, \text{diaper}, \text{nuts}\}$**  is frequent, so is  **$\{\text{beer}, \text{diaper}\}$**
  - Every transaction containing  $\{\text{beer}, \text{diaper}, \text{nuts}\}$  also contains  $\{\text{beer}, \text{diaper}\}$
  - **Apriori: Any subset of a frequent itemset must be frequent**
- Efficient mining methodology
  - If **any subset of an itemset  $S$**  is infrequent, then there is no chance for  $S$  to be frequent—why do we even have to consider  $S$ !?

*A sharp knife for pruning!*

# Apriori: A Candidate Generation & Test Approach

- Outline of Apriori (level-wise, candidate generation and test)
  - Initially, scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-( $k+1$ ) candidate itemsets from length- $k$  frequent itemsets
    - Test the candidates against DB to find **frequent** ( $k+1$ )-itemsets
    - Set  $k := k + 1$
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

# The Apriori Algorithm: An Example



# The Apriori Algorithm (Pseudo-Code)

$C_k$ : Candidate itemset of size  $k$

$F_k$ : Frequent itemset of size  $k$

$K := 1$ ;

$F_k := \{\text{frequent items}\}$ ; // frequent 1-itemset

**While** ( $F_k \neq \emptyset$ ) **do** { // when  $F_k$  is non-empty

$C_{k+1} :=$  candidates generated from  $F_k$ ; // candidate generation

    Derive  $F_{k+1}$  by counting candidates in  $C_{k+1}$  with respect to  $TDB$  at minsup;

$k := k + 1$

}

**return**  $\bigcup_k F_k$  // return  $F_k$  generated at each level



# FPGrowth: Mining Frequent Patterns by Pattern Growth

- Idea: Frequent pattern growth (FPGrowth)
  - Find frequent single items and partition the database based on each such item
  - Recursively grow frequent patterns by doing the above for each partitioned database (also called *conditional database*)
  - To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed
- Mining becomes
  - Recursively construct and mine (conditional) FP-trees
  - Until the resulting FP-tree is empty, or until it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

# Example: Construct FP-tree from a Transactional DB

TID	Items in the Transaction	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}

Answer:

f:4, a:3, c:4, b:3, m:3, p:3;  
 fa:3, fc:3, fm:3, ac:3, am:3,  
 cm:3, cp:3;  
 fcm:3, fam:3, cam:3;  
 fcam:3.

1. Scan DB once, find single item frequent pattern:

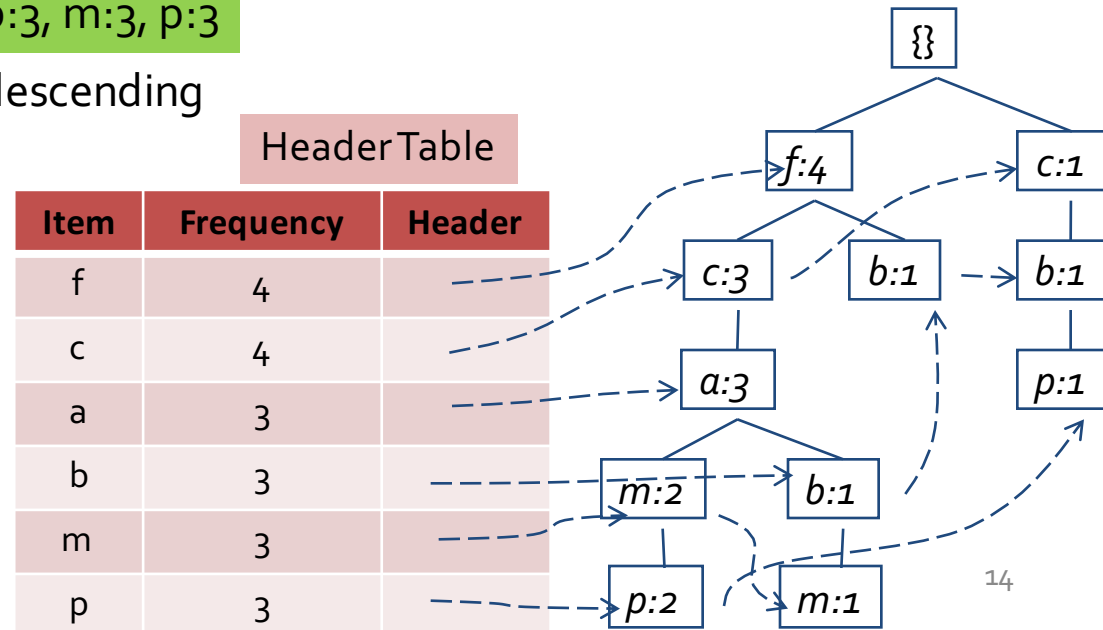
Let min\_support = 3

f:4, a:3, c:4, b:3, m:3, p:3

2. Sort frequent items in frequency descending order, f-list

F-list = f-c-a-b-m-p

3. Scan DB again, construct FP-tree

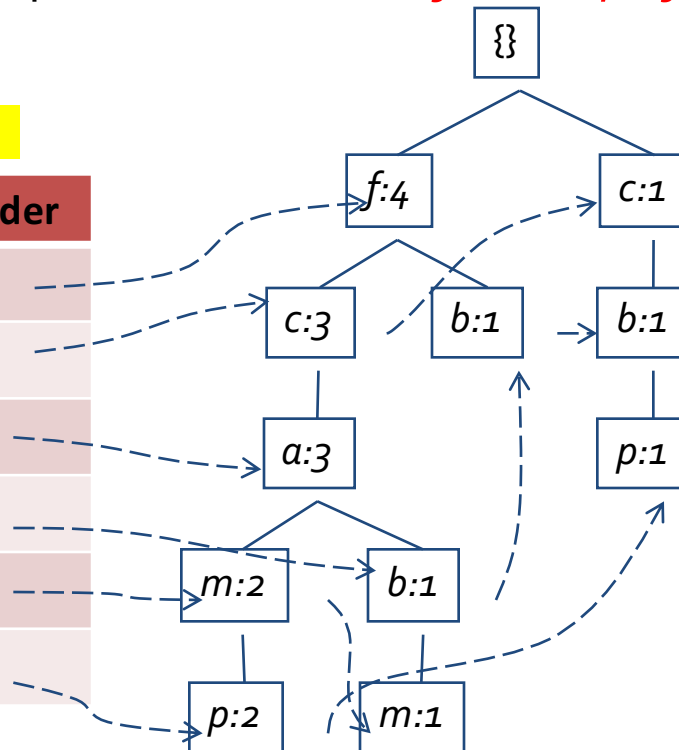


# Divide and Conquer Based on Patterns and Data

- Pattern mining can be partitioned according to current patterns
  - Patterns containing  $p$ :  $p$ 's conditional database:  $fcam:2, cb:1$
  - Patterns having  $m$  but no  $p$ :  $m$ 's conditional database:  $fca:2, fcab:1$
  - .....
- $p$ 's conditional pattern base: *transformed prefix paths* of item  $p$

min\_support = 3

Item	Frequency	Header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



## Conditional pattern bases

Item	Conditional pattern base
c	$f:3$
a	$fc:3$
b	$fca:1, f:1, c:1$
m	$fca:2, fcab:1$
p	$fcam:2, cb:1$

# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

item cond. pattern base

<i>c</i>	<i>f:3</i>
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

min\_support = 3

For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & mine it

*p*-conditional PB: *fcam:2, cb:1* → *c:3*

*m*-conditional PB: *fca:2, fcab:1* → *fca:3*

*b*-conditional PB: *fca:1, f:1, c:1* →  $\phi$

*a*-conditional PB: *fc:3* → *fc:3*

*c*-conditional PB: *f:3* → *f:3*



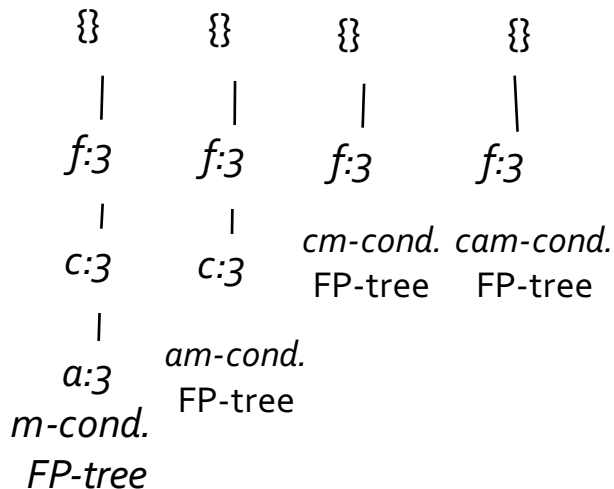
# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

item cond. pattern base

c f:3  
a fc:3  
b fca:1, f:1, c:1  
m fca:2, fcab:1  
p fcam:2, cb:1

min\_support = 3



For each conditional pattern-base

- Mine single-item patterns
- Construct its **cond. FP-tree** & **mine** it

p-conditional PB:  $fcam:2, cb:1 \rightarrow c:3$

m-conditional PB:  $fca:2, fcab:1 \rightarrow fca:3$

b-conditional PB:  $fca:1, f:1, c:1 \rightarrow \phi$

a-conditional PB:  $fc:3 \rightarrow fc:3$

c-conditional PB:  $f:3 \rightarrow f:3$

mine(<f:3, c:3, a:3>|m)

→ (am:3) + mine(<f:3, c:3>|am)

→ (cam:3) + (fam:3) + mine(<f:3>|cam)

→ (fcam:3)

→ (cm:3) + mine(<f:3>|cm)

→ (fcm:3)

→ (fm:3)

# Mine Each Conditional Pattern-Base Recursively

## Conditional pattern bases

### item cond. pattern base

<i>c</i>	<i>f</i> :3
<i>a</i>	<i>fc</i> :3
<i>b</i>	<i>fca</i> :1, <i>f</i> :1, <i>c</i> :1
<i>m</i>	<i>fca</i> :2, <i>fcab</i> :1
<i>p</i>	<i>fcam</i> :2, <i>cb</i> :1

min\_support = 3

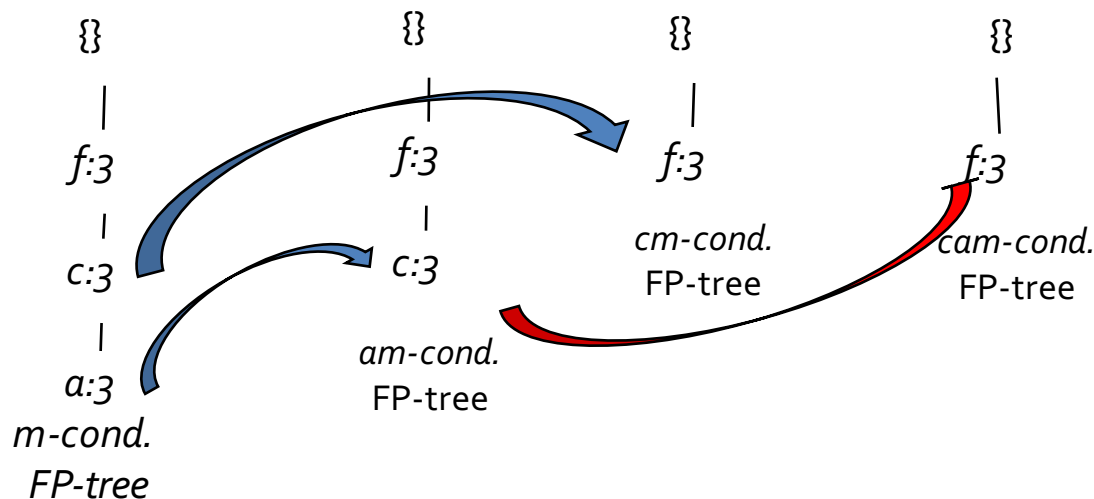
For each conditional pattern-base

- Mine single-item patterns
- Construct its cond. FP-tree & mine it

*p*-conditional PB: *fcam*:2, *cb*:1 → *c*: 3

*m*-conditional PB: *fca*:2, *fcab*:1 → *fca*: 3

*b*-conditional PB: *fca*:1, *f*:1, *c*:1 →  $\phi$



Actually, for single branch FP-tree, all frequent patterns can be generated in one shot

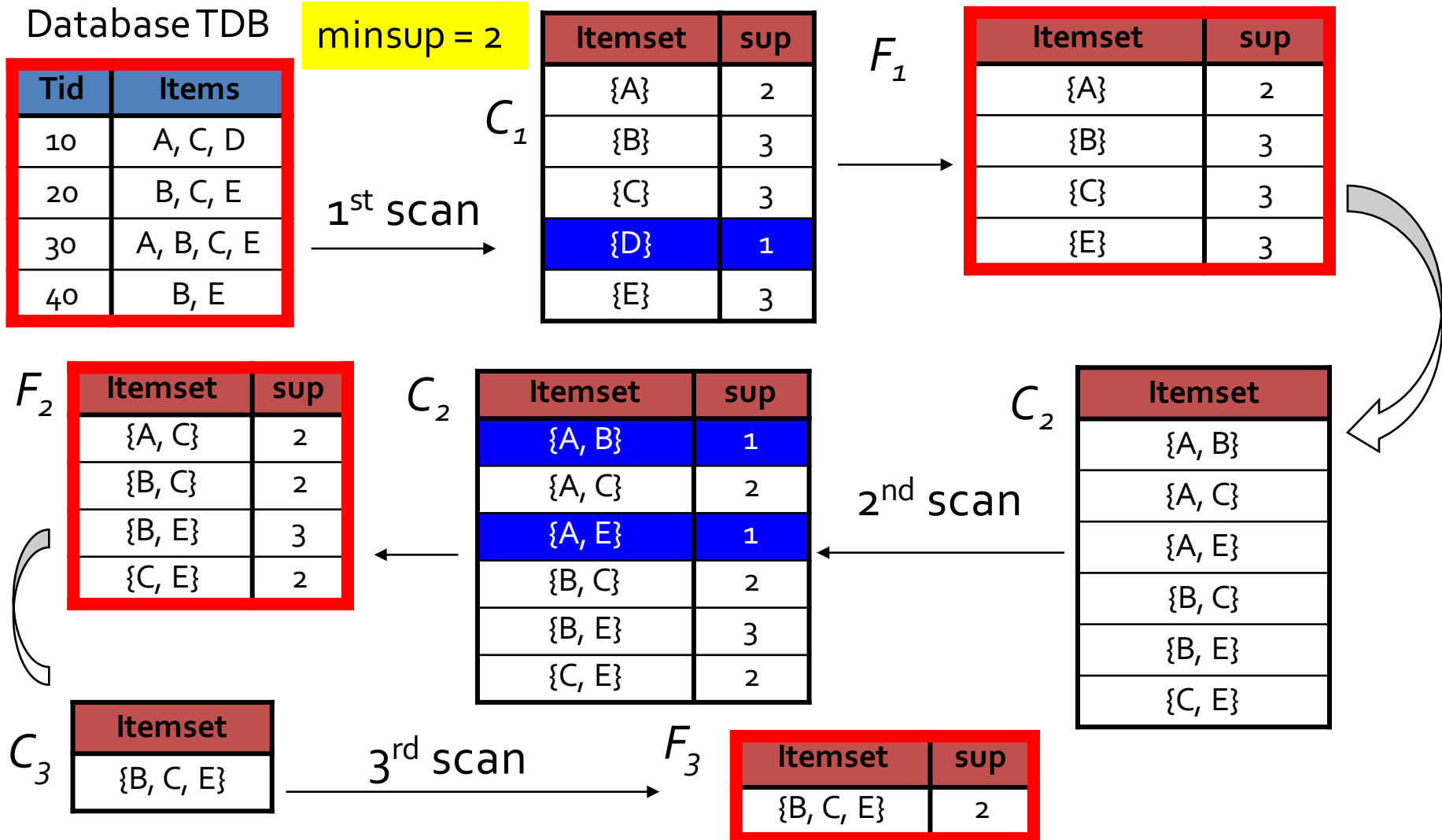
***m*: 3**

***fm*: 3, *cm*: 3, *am*: 3**

***fcm*: 3, *fam*: 3, *cam*: 3**

***fcam*: 3**

# Try FP-Growth?



# Try WikiBooks' Example

- [https://en.wikibooks.org/wiki/Data\\_Mining\\_Algorithms\\_In\\_R/Frequent\\_Pattern\\_Mining/The\\_FP-Growth\\_Algorithm](https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm)



# References

- R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", in Proc. of SIGMOD'93
- R. J. Bayardo, "Efficiently mining long patterns from databases", in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules", in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases", VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules", SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating association rule mining with relational database systems: Alternatives and implications", SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "Parallel algorithm for discovery of association rules", Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", SIGMOD'00

# References (cont.)

- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014
- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010