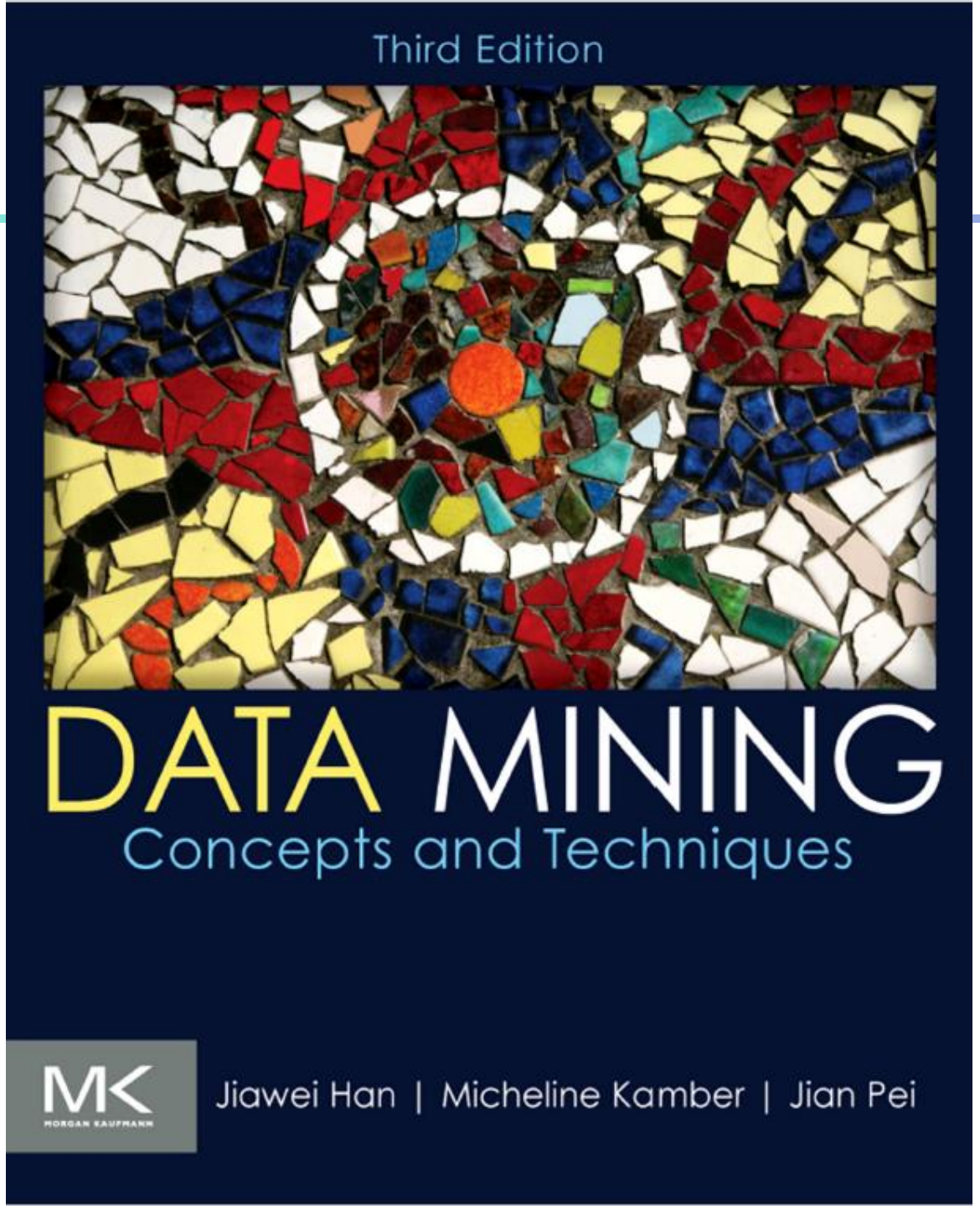
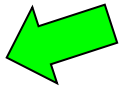
Data Warehousing and Data mining
CSE-463

UNIT 02


Third Edition
DATA MINING
Concepts and Techniques
MK
MORGAN KAUFMANN
Jiawei Han | Micheline Kamber | Jian Pei

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

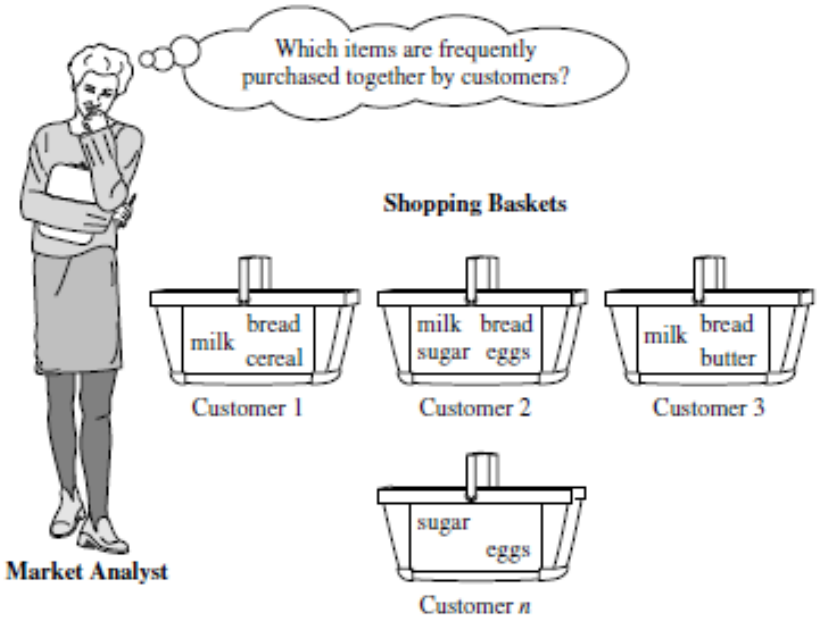
- Basic Concepts 
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

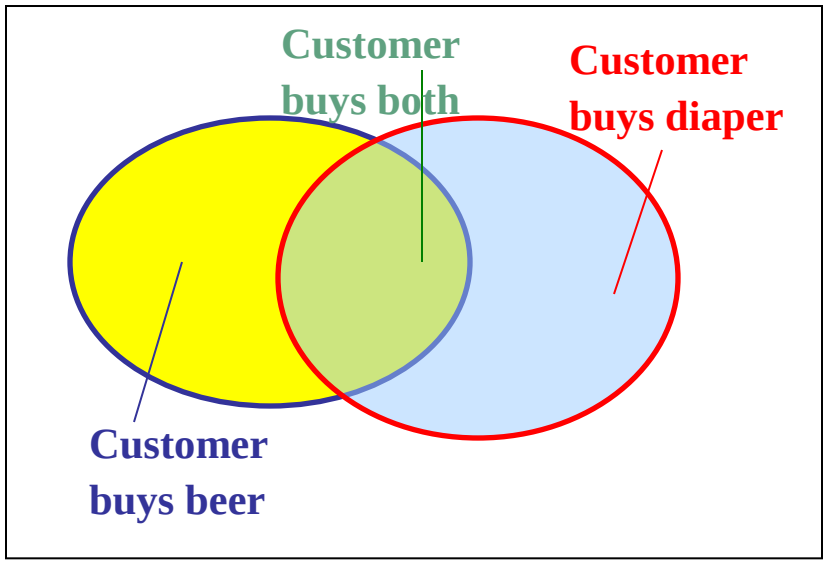
Applications

- Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.



Basic Concepts: Frequent Patterns

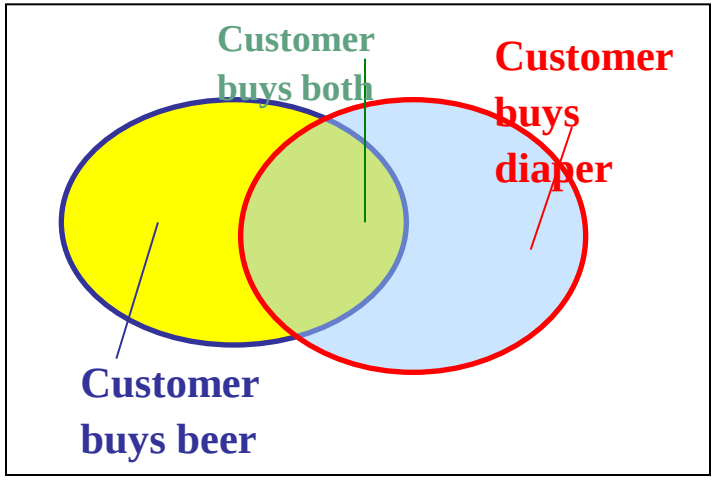
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



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

Basic Concepts: Association Rules

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



- Find all the rules X=> Y with minimum support and confidence
  - support, s, probability that a transaction contains X ∪ Y
  - confidence, c, conditional probability that a transaction having X also contains Y

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence(A \Rightarrow B) = P(B|A).$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

- Association rules: (many more!)
  - Beer=> Diaper (60%, 100%)
  - Diaper=> Beer (60%, 75%)

Association rule mining, a two-step process:

1. **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min sup*.

2. Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ⊃X, *with the same support* as X.
- **Closed set:** {"bread", "milk"} - If you add any other item like "eggs" to this set, the new set {"bread", "milk", "eggs"} will likely have a lower support (occur less frequently in transactions) compared to {"bread", "milk"}.

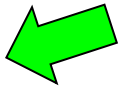
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃X.

- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

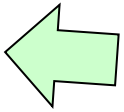
- Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.
- What is the set of closed itemset?
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods 
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach



- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format

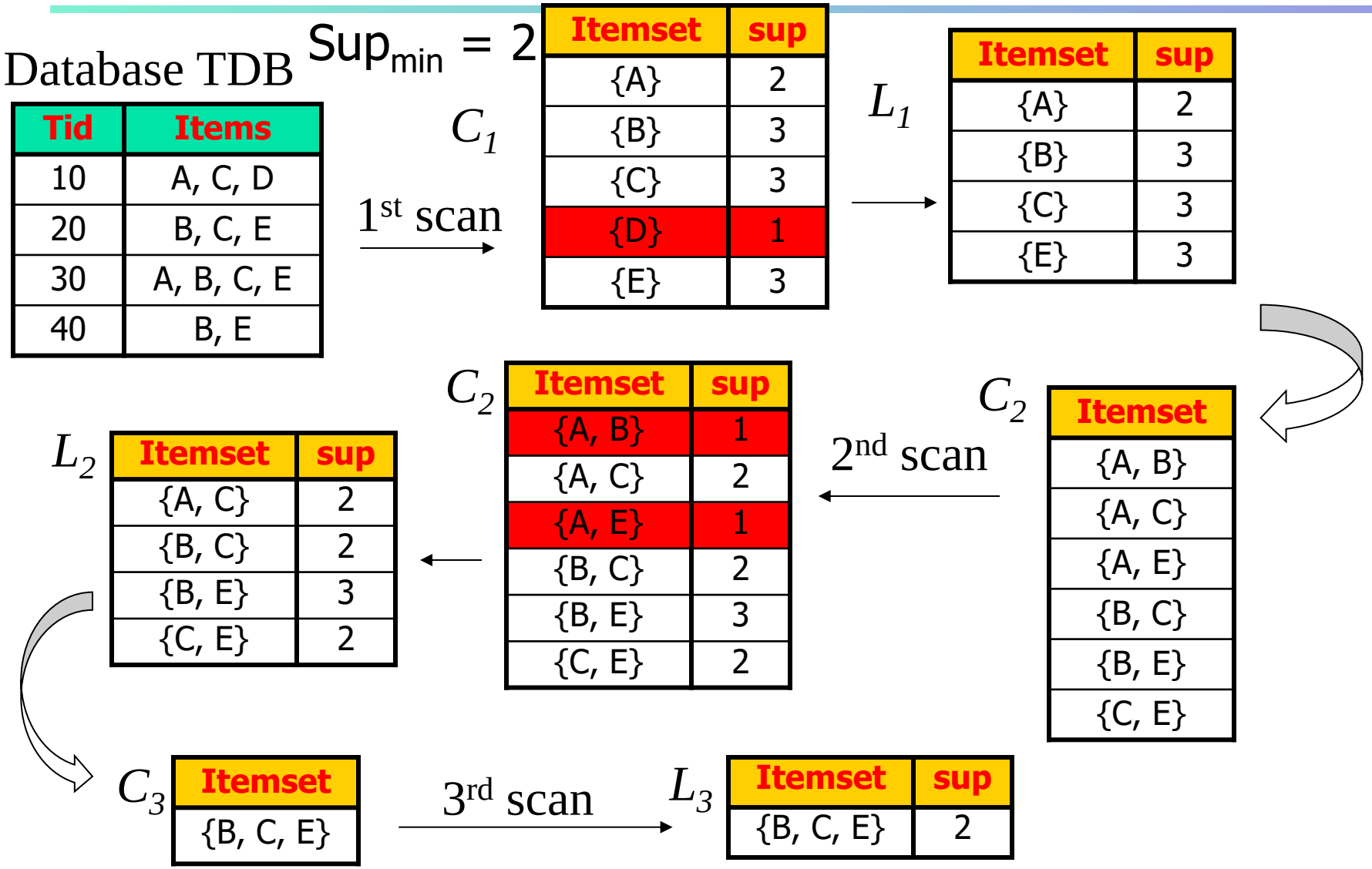
The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

(1) L_1 = find_frequent_1-itemsets(D);
(2) **for** ($k = 2; L_{k-1} \neq \phi; k$++) {
(3) C_k = apriori_gen(L_{k-1});
(4) **for each** transaction $t \in D$ { // scan D for counts
(5) C_t = subset(C_k, t); // get the subsets of t that are candidates
(6) **for each** candidate $c \in C_t$
(7) c.count++;
(8) }
(9) $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10) }
(11) **return** $L = \cup_k L_k$;

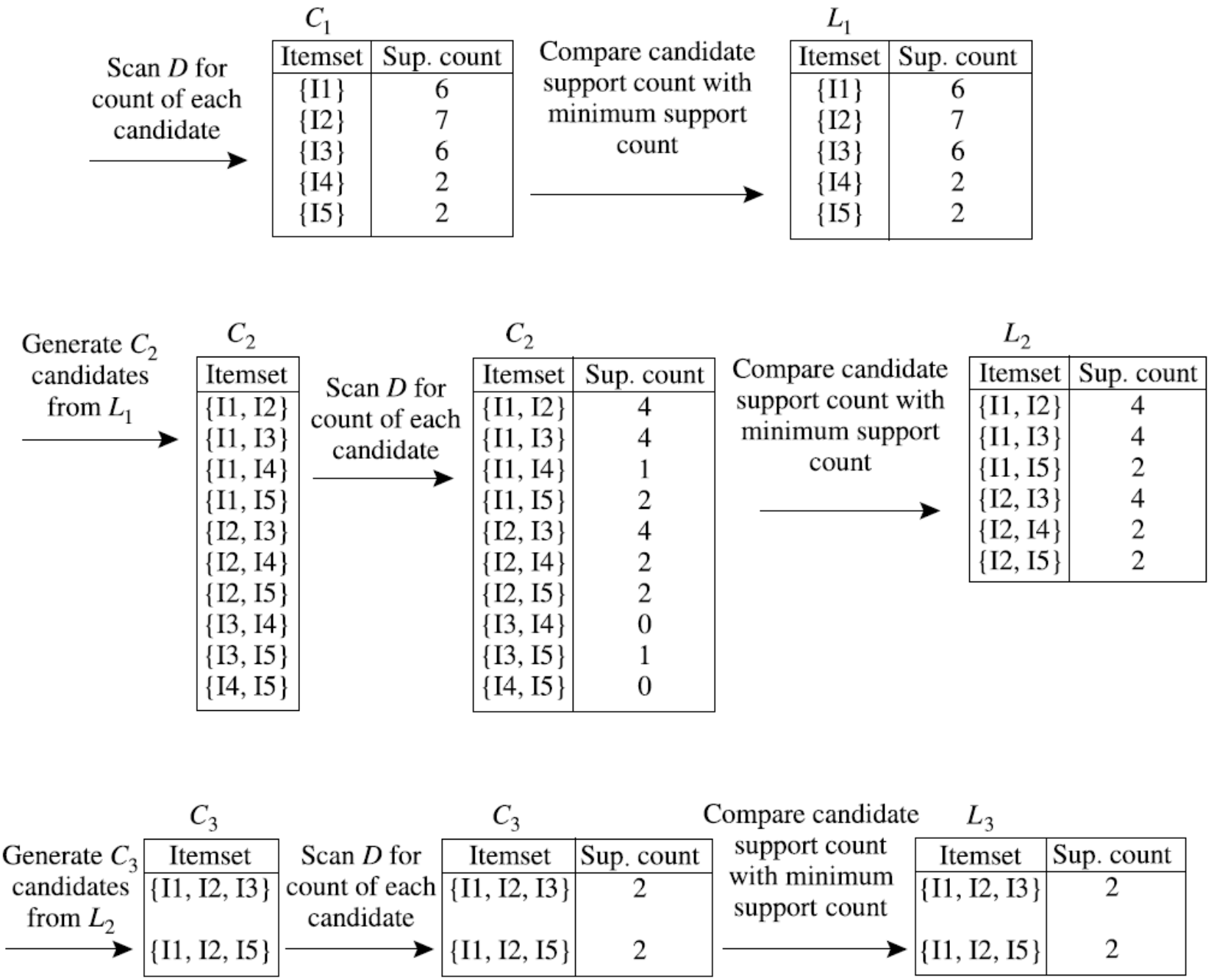
```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
(1)     for each itemset l_1 ∈ L_{k-1}
(2)         for each itemset l_2 ∈ L_{k-1}
(3)             if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2])
                    ∧... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then {
(4)                 c = l_1 ⋈ l_2; // join step: generate candidates
(5)                 if has_infrequent_subset(c, L_{k-1}) then
(6)                     delete c; // prune step: remove unfruitful candidate
(7)                 else add c to C_k;
(8)             }
(9)     return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k-1)-itemsets); // use prior knowledge
(1)     for each (k-1)-subset s of c
(2)         if s ∉ L_{k-1} then
(3)             return TRUE;
(4)     return FALSE;
```

Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in 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


C_1
Itemset | Sup. count
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2
Scan D for count of each candidate
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

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}.$

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Mining Association Rules

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}$$

For each frequent itemset l, generate all nonempty subsets of l.

For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ *min_conf*, where *min_conf* is the minimum confidence threshold.

$\{I1, I2\} \Rightarrow I5$, $confidence = 2/4 = 50\%$
$\{I1, I5\} \Rightarrow I2$, $confidence = 2/2 = 100\%$
$\{I2, I5\} \Rightarrow I1$, $confidence = 2/2 = 100\%$
$I1 \Rightarrow \{I2, I5\}$, $confidence = 2/6 = 33\%$
$I2 \Rightarrow \{I1, I5\}$, $confidence = 2/7 = 29\%$
$I5 \Rightarrow \{I1, I2\}$, $confidence = 2/2 = 100\%$

Scalable Frequent Itemset Mining Methods

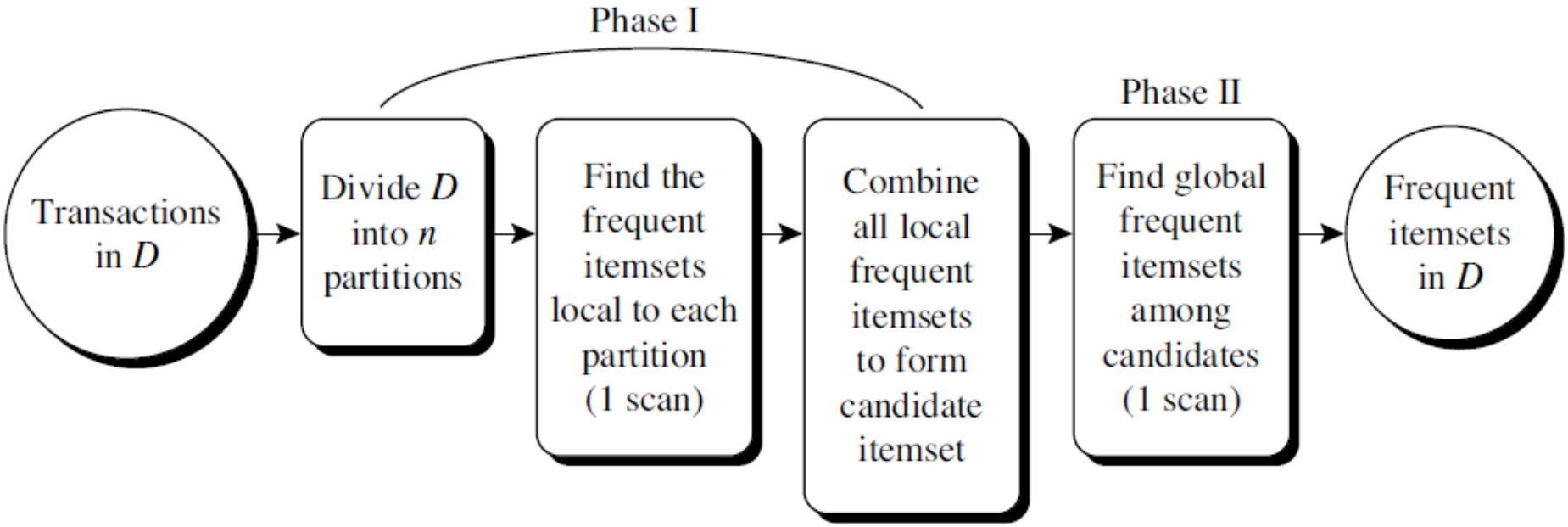
- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns



Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

DHP: Reduce the Number of Candidates

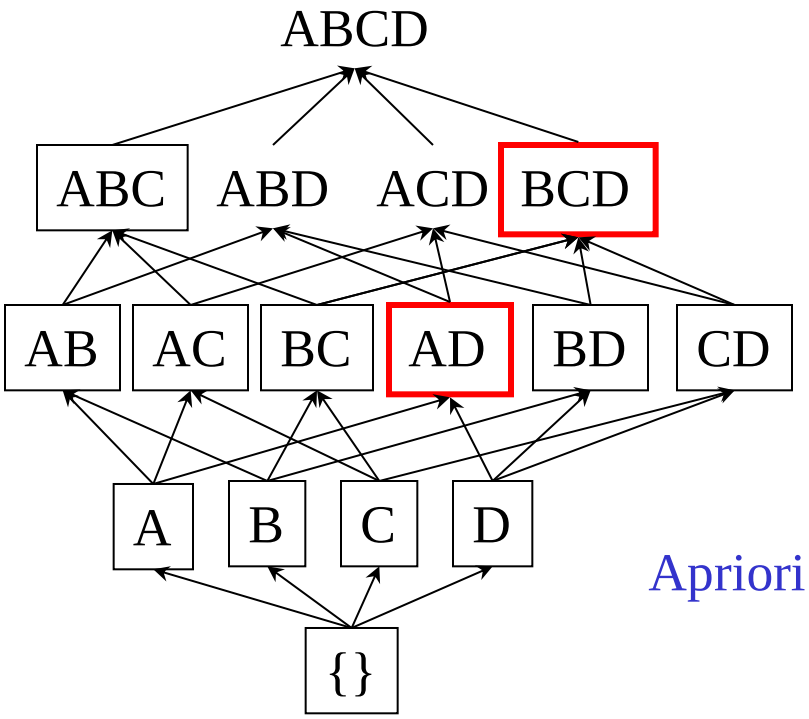
- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries
    - {ab, ad, ae}
    - {bd, be, de}
    - …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
. . .	. . .
102	{yz, qs, wt}

Hash Table

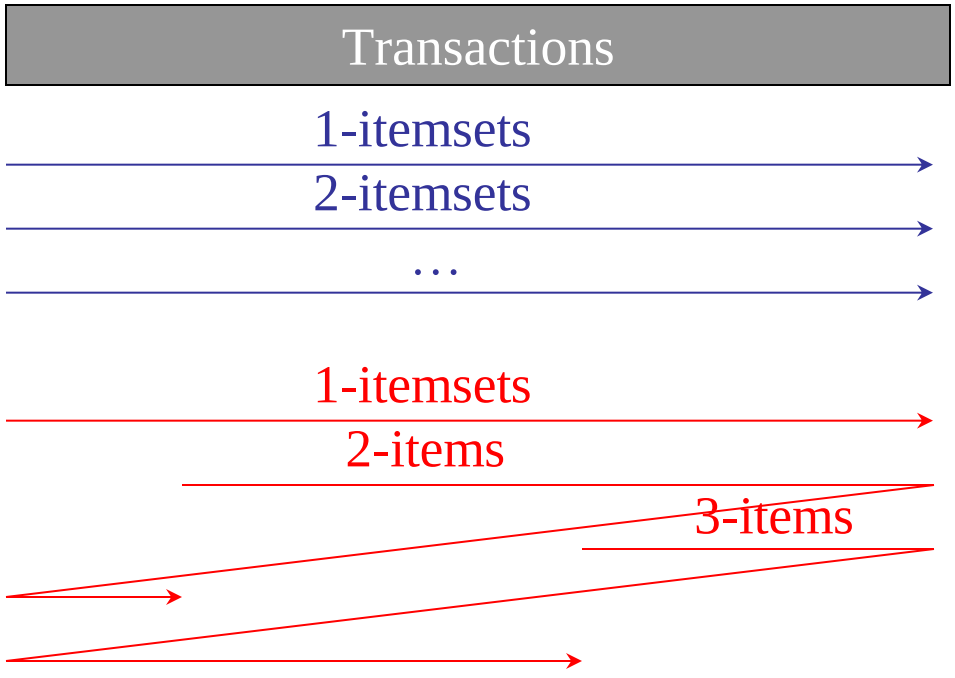
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

DIC: Reduce Number of Scans



Itemset lattice

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



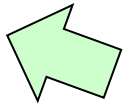
Apriori

DIC

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

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach



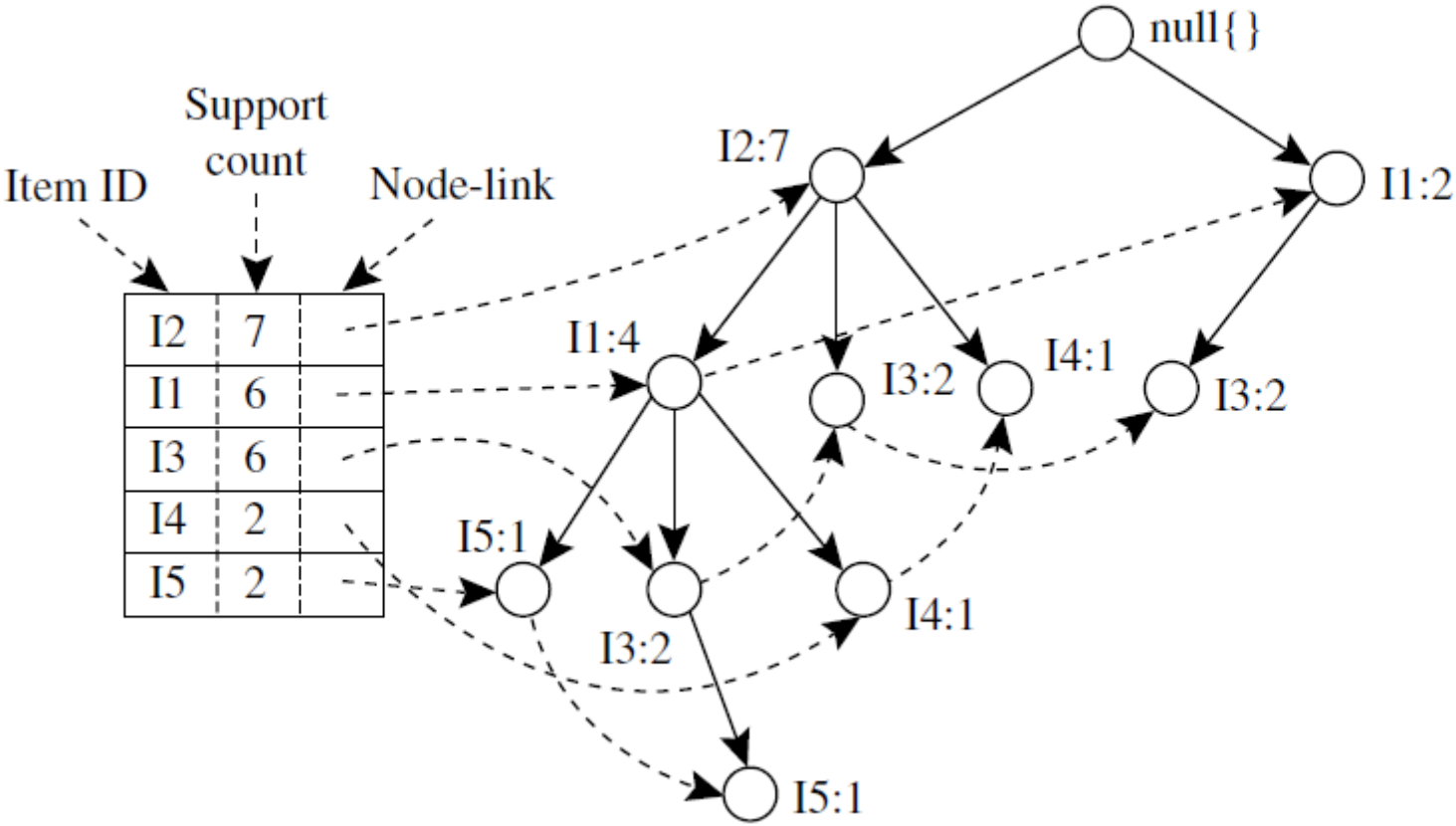
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc □ abcd is a frequent pattern

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

Example: FP-growth Mining



Item	Conditional Pattern Base
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}
I4	{{I2, I1: 1}, {I2: 1}}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}
I1	{{I2: 4}}

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

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}

FP-Tree Construction Algorithm

1. **Scan the dataset once** to find the frequency of each item.
2. **Remove infrequent items** (items with support below the minimum threshold).
3. **Sort the frequent items in descending order of frequency** (most common items first).
4. **Build the FP-Tree**:
    - Start with a root node labeled "**null**".
    - For each transaction:
        - Sort its items according to the frequency order.
        - Insert the sorted items as a **path** in the FP-Tree.
        - If an item already exists in the tree, **increase its count**; otherwise, **create a new node**.
        - Link nodes with the same item name using a **node-link structure** for quick access.

Example: FP-Growth Algorithm

```
procedure FP_growth(Tree, α)
(1)   if Tree contains a single path P then
(2)     for each combination (denoted as β) of the nodes in the path P
(3)       generate pattern β ∪ α with support_count = minimum support count of nodes in β;
(4)   else for each a_i in the header of Tree {
(5)     generate pattern β = a_i ∪ α with support_count = a_i.support_count;
(6)     construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)       call FP_growth(Tree_β, β); }
```

Construct FP-tree from a Transaction Database

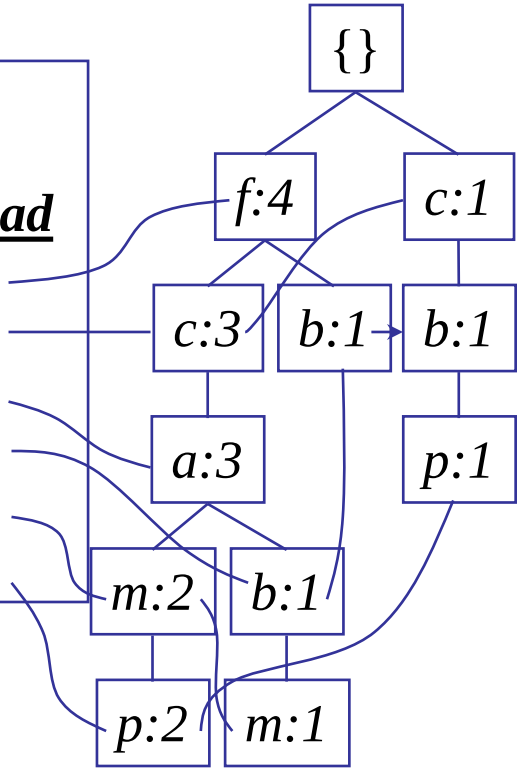
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

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



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

Partition Patterns and Databases

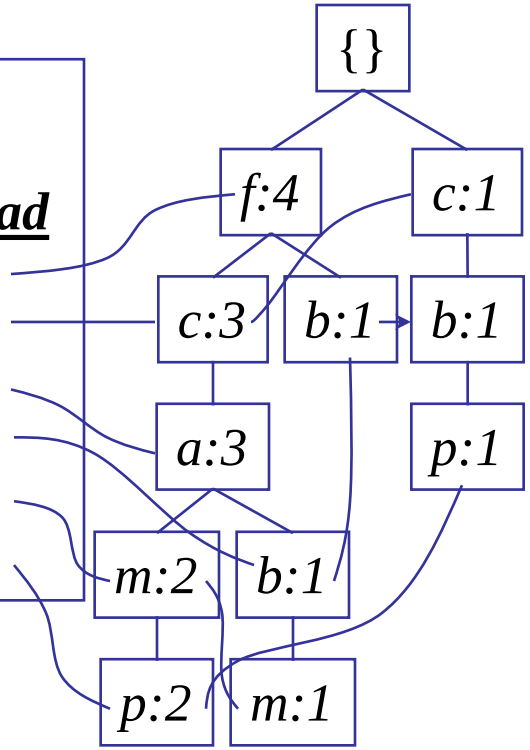
- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

Header Table

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

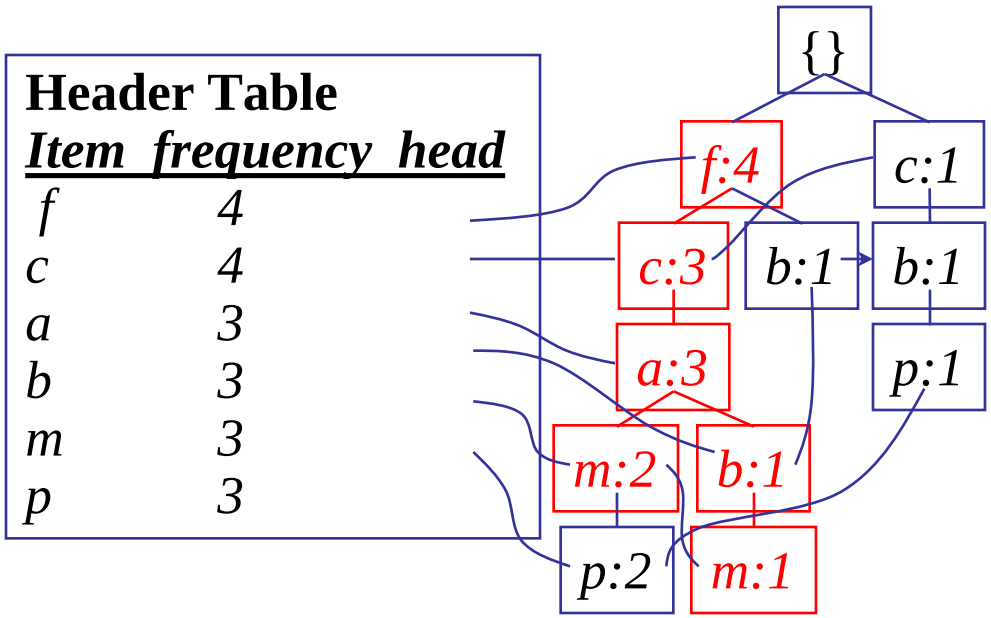


***Conditional* pattern bases**

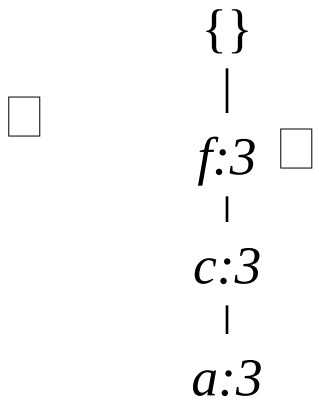
item	***cond. pattern base***
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base



***m-conditional* pattern base:**
fca:2, fcab:1



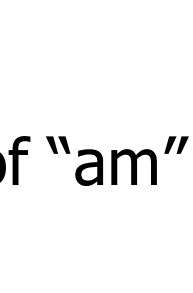
***m-conditional* FP-tree**

All frequent patterns relate to *m*

***m*,**
fm, cm, am,
fcm, fam, cam,
fcam

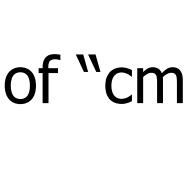
Recursion: Mining Each Conditional FP-tree

Cond. pattern base of "am": (fc:3)

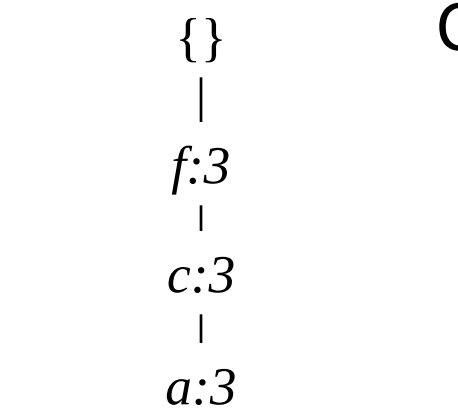


***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)

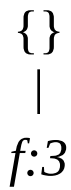


***cm-conditional* FP-tree**



***m-conditional* FP-tree**

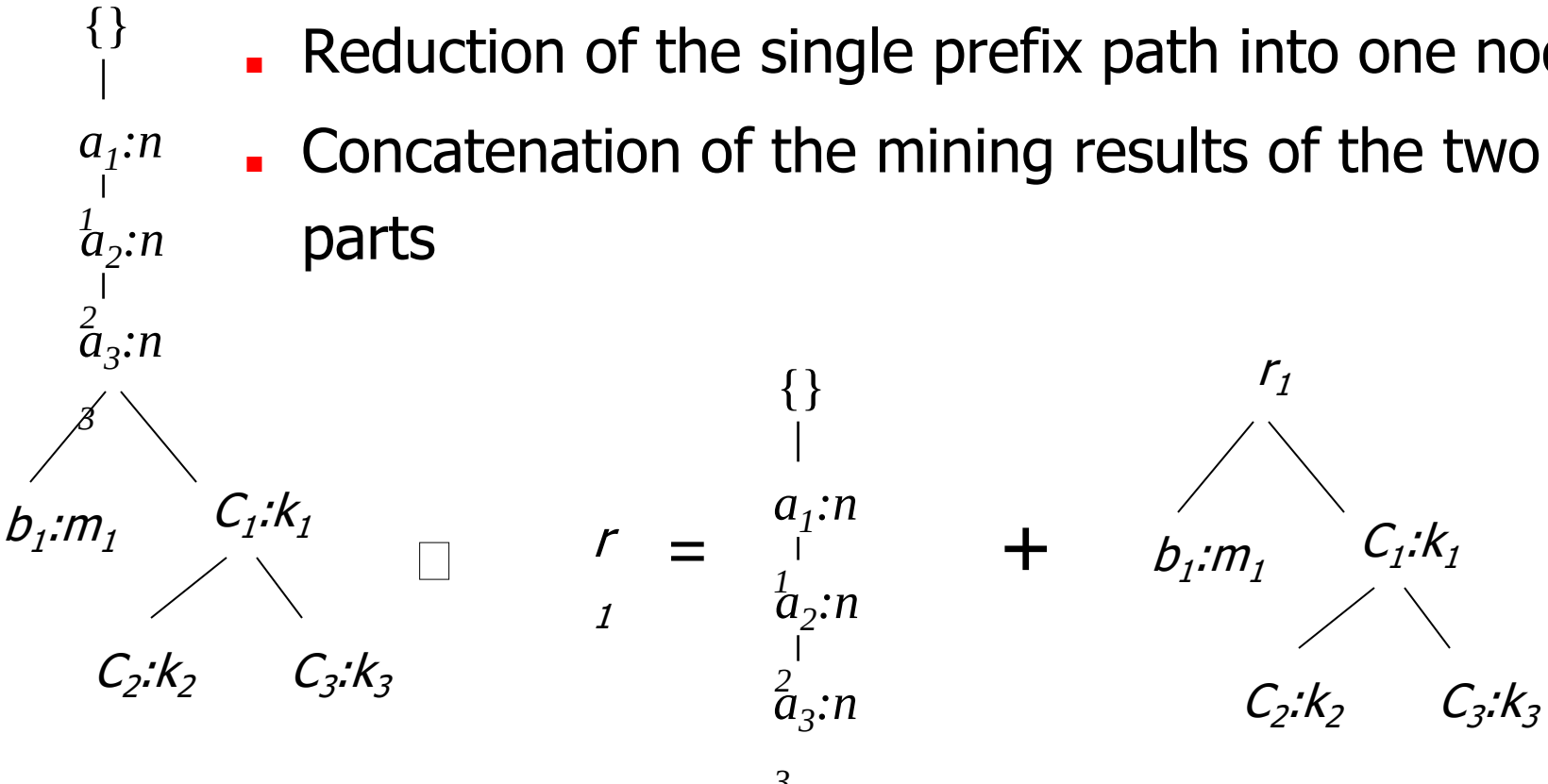
Cond. pattern base of "cam": (f:3)



***cam-conditional* FP-tree**

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

The Frequent Pattern Growth Mining Method

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - 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

Advantages of the Pattern Growth Approach

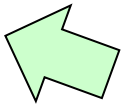
- Divide-and-conquer:
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching
- A good open-source implementation and refinement of FPGrowth
  - FPGrowth+ (Grahne and J. Zhu, FIMI'03)

Practice Question

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

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns



ECLAT: Mining by Exploring Vertical Data Format

- Vertical format: t(AB) = $\{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving frequent patterns based on vertical intersections
  - t(X) = t(Y): X and Y always happen together
  - t(X) $\subset$ t(Y): transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - t(X) = $\{T_1, T_2, T_3\}$, t(XY) = $\{T_1, T_3\}$
  - Diffset (XY, X) = $\{T_2\}$
- Eclat (Zaki et al. @KDD'97)
- Mining Closed patterns using vertical format: CHARM (Zaki & Hsiao@SDM'02)

Example

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

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

2-Itemsets in Vertical Data Format

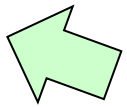
itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns



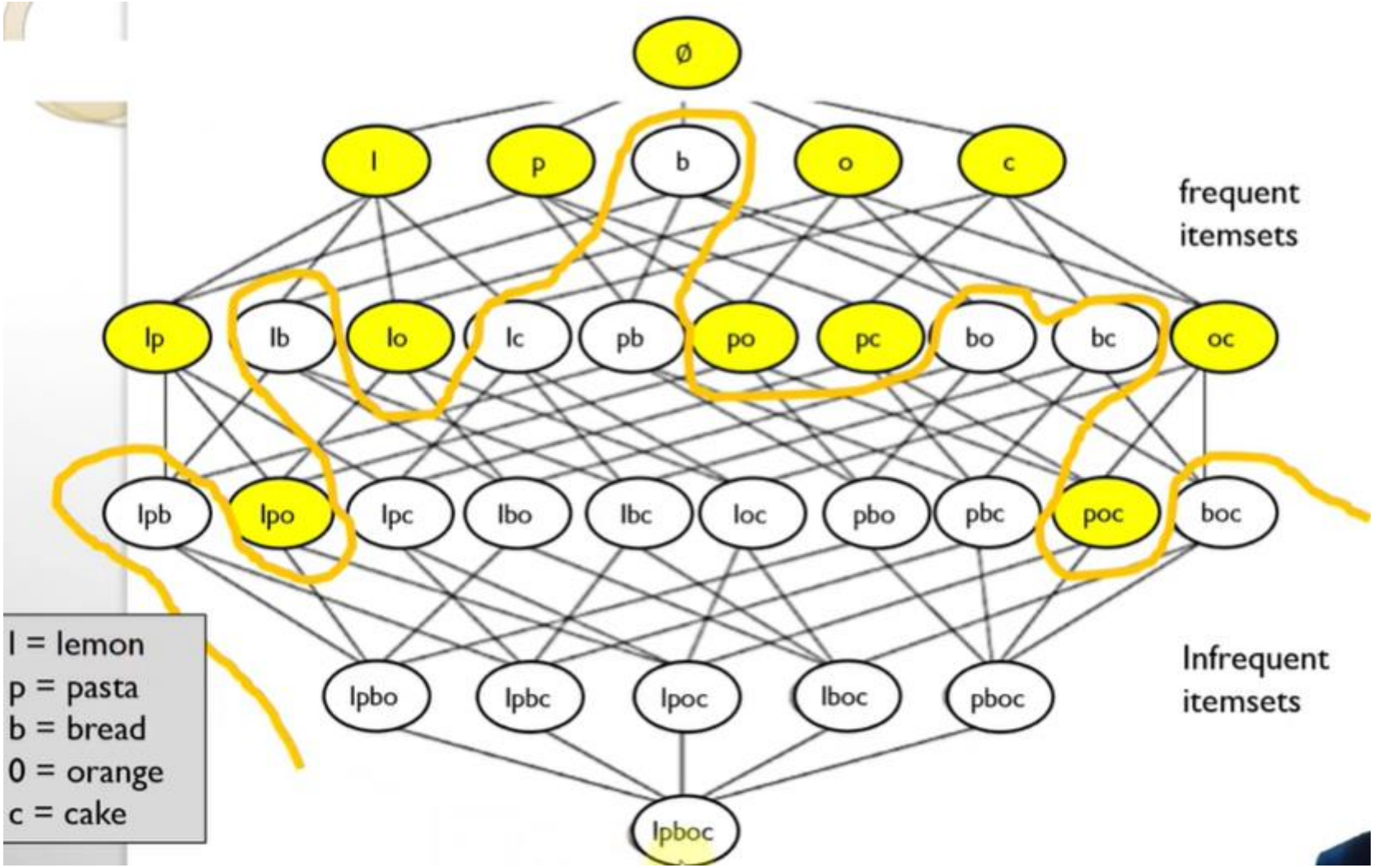
Mining Frequent Closed Patterns

- Frequent itemset mining may generate a huge number of frequent itemsets, especially when the *min sup* threshold is set low or when there exist long patterns in the data set.
- closed frequent itemsets can substantially reduce the number of patterns generated in frequent itemset mining while preserving the complete information.
- Thus, in practice, it is more desirable to mine the set of closed frequent itemsets rather than the set of all frequent itemsets in most cases.

T1	**['pasta', 'lemon', 'bread', 'orange']**
T2	**['pasta', 'lemon']**
T3	**['pasta', 'orange', 'cake']**
T4	**['pasta', 'lemon', 'orange', 'cake']**

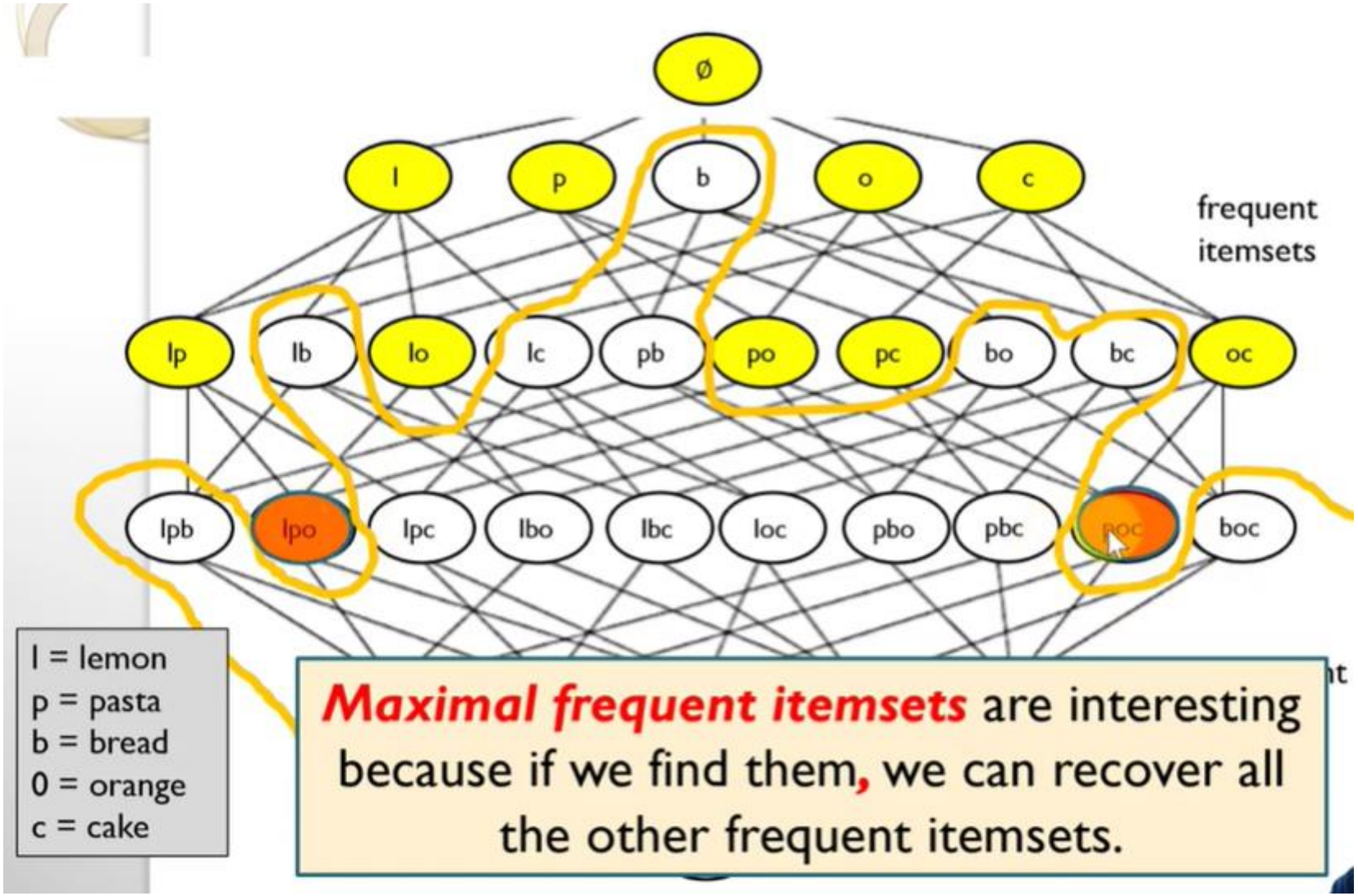
	support	itemsets
0	0.50	(cake)
1	0.75	(lemon)
2	0.75	(orange)
3	1.00	(pasta)
4	0.50	(orange, cake)
5	0.50	(pasta, cake)
6	0.50	(orange, lemon)
7	0.75	(pasta, lemon)
8	0.75	(pasta, orange)
9	0.50	(pasta, orange, cake)
10	0.50	(pasta, orange, lemon)

Frequent Patterns



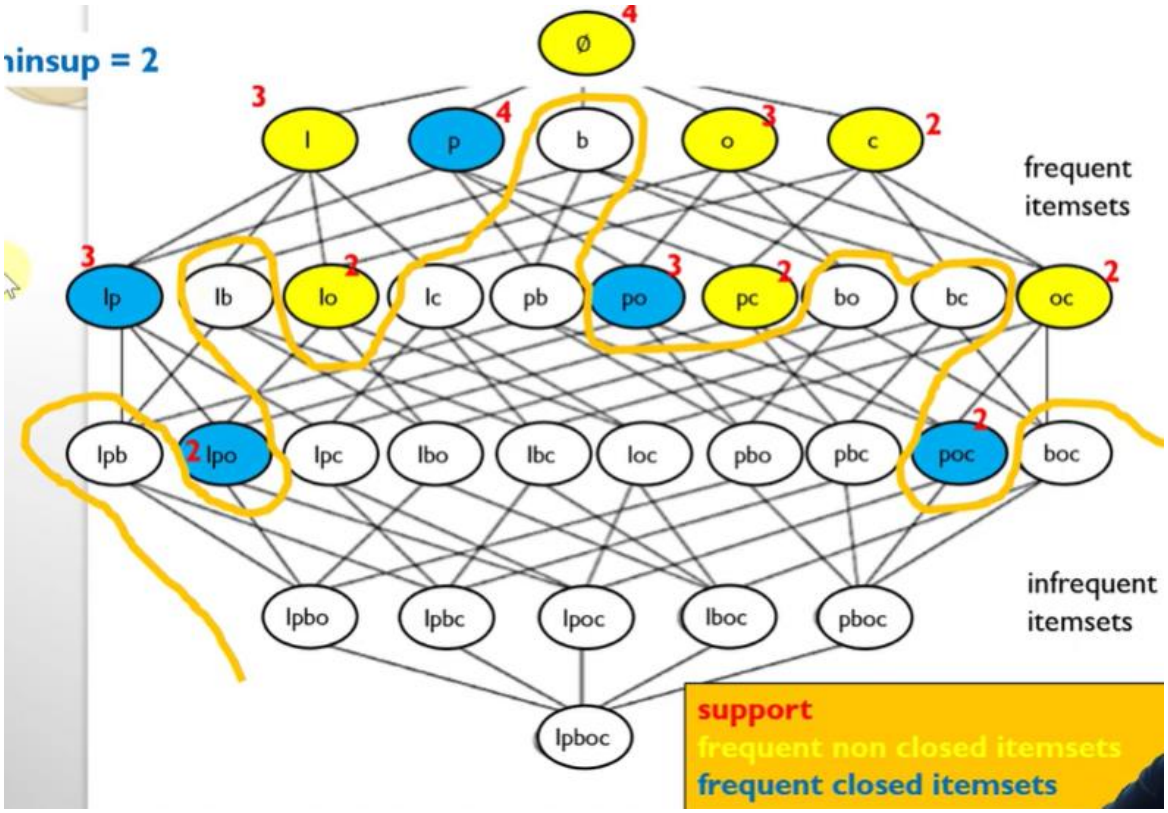
https://www.philippe-fournier-viger.com/COURSES/Pattern_mining/Closed_maximal_itemsets.pdf

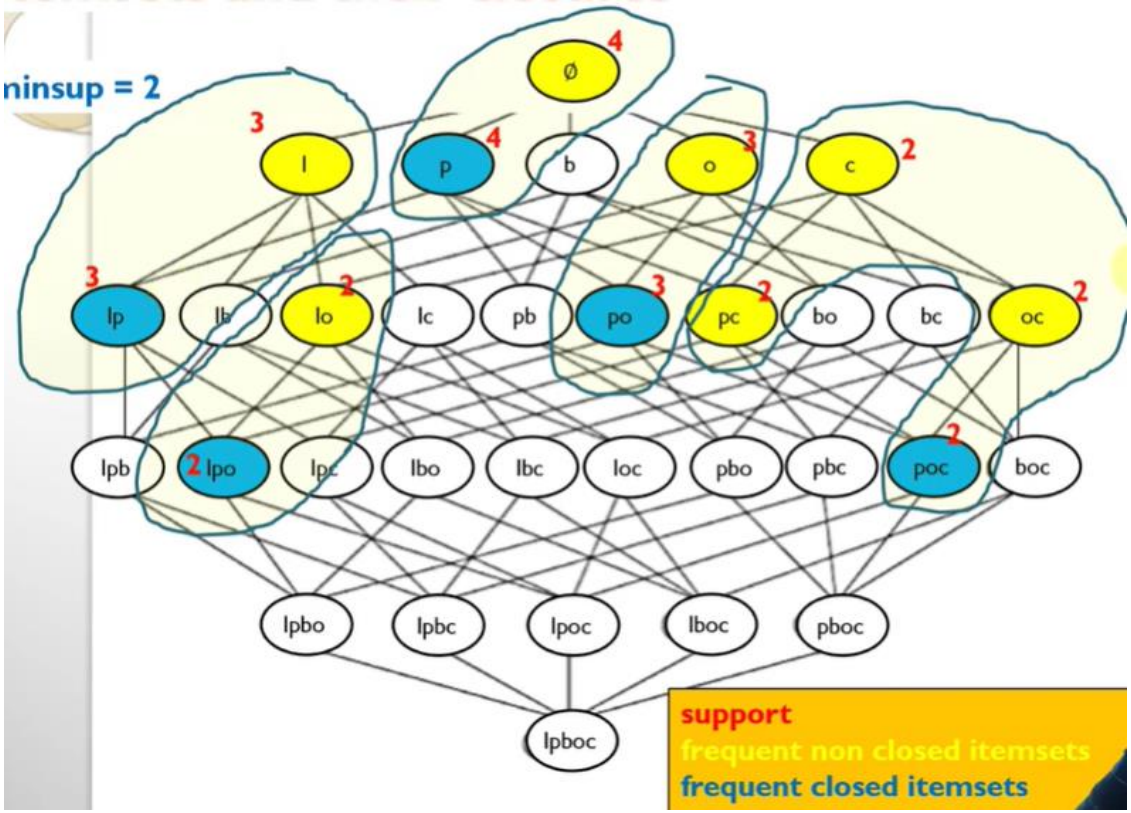
Recover All Frequent Patterns from Maximal frequent Patterns



Recover All Frequent Patterns from Closed Frequent patterns

	support	itemsets
0	0.50	(cake)
1	0.75	(lemon)
2	0.75	(orange)
3	1.00	(pasta)
4	0.50	(orange, cake)
5	0.50	(pasta, cake)
6	0.50	(orange, lemon)
7	0.75	(pasta, lemon)
8	0.75	(pasta, orange)
9	0.50	(pasta, orange, cake)
10	0.50	(pasta, orange, lemon)




minsup = 2
Ø
4
l
3
p
4
b
o
3
c
2
lp
3
lb
lo
2
lc
pb
po
3
pc
2
bo
bc
oc
2
lpb
lpo
2
lpc
lbo
lbc
loc
pbo
pbc
poc
2
boc
lpbo
lpbc
lpoc
lboc
pboc
lpboc
support
frequent non closed itemsets
frequent closed itemsets

A naïve approach

Mine the complete set of frequent itemsets and then remove every frequent itemset that is a proper subset of, and carries the same support as, an existing frequent itemset. However, this is quite costly.

This is prohibitively expensive.

Recommended Approach

Item merging:

Item merging is a concept used in frequent pattern mining. It states that if every transaction containing a frequent itemset X also contains another itemset Y, and there is no transaction that contains a proper superset of Y without X, then the union of X and Y ($X \cup Y$) forms a **frequent closed itemset**. This means we do not need to search for any itemset that contains X but excludes Y.

Example

Given Transactions:

Transaction ID	Items Purchased
T1	{Milk, Bread, Butter}
T2	{Milk, Bread}
T3	{Milk, Bread, Butter}
T4	{Milk, Bread, Butter}
T5	{Milk, Bread}

Frequent Itemsets:

1. **{Milk, Bread}** appears in **all transactions**.
2. **{Butter}** appears **only when {Milk, Bread} is present**.
3. No transaction contains **{Butter}** without **{Milk, Bread}**.

Applying Item Merging

•Since every transaction that contains **{Milk, Bread}** also contains **{Butter}** (in some cases), we **merge them into one closed itemset.**

{Milk, Bread, Butter} is the **frequent closed itemset**.

•There's no need to consider **{Milk, Bread}** alone, since it **always appears with Butter** in the same transactions.

Recommended Approach

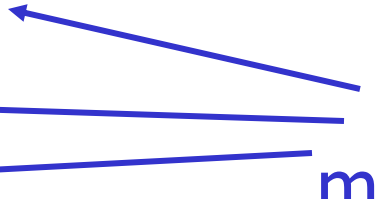
Sub-itemset pruning: If a frequent itemset X is a proper subset of an already found frequent closed itemset Y and support count(X) = support count(Y), then X and all of X's descendants in the set enumeration tree cannot be frequent closed itemsets and thus can be pruned.

Item skipping: In the depth-first mining of closed itemsets, at each level, there will be a prefix itemset X associated with a header table and a projected database. If a local frequent item p has the same support in several header tables at different levels, we can safely prune p from the header tables at higher levels.

MaxMiner: Mining Max-Patterns

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

- 1st scan: find frequent items
  - A, B, C, D, E
- 2nd scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE, DE



Potential max-patterns

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan
- R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD'98*

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'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
- H. Toivonen. Sampling large databases for association rules. VLDB'96
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98

Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing, 2002.
- G. Grahne and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. FIMI'03
- B. Goethals and M. Zaki. An introduction to workshop on frequent itemset mining implementations. *Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03),* Melbourne, FL, Nov. 2003
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03

Ref: Vertical Format and Row Enumeration Methods

- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. DAMI:97.
- M. J. Zaki and C. J. Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.
- C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. KDD’02.
- F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki , CARPENTER: Finding Closed Patterns in Long Biological Datasets. KDD'03.
- H. Liu, J. Han, D. Xin, and Z. Shao, Mining Interesting Patterns from Very High Dimensional Data: A Top-Down Row Enumeration Approach, SDM'06.