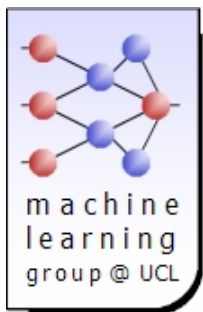


# Association rules



Marco Saerens (UCL), with  
Christine Decaestecker (ULB)



Université catholique de Louvain

Université partenaire de l'Académie universitaire 'Louvain'



# Slides references

- Many slides and figures have been adapted from the slides associated to the following books:
  - Alpaydin (2004), Introduction to machine learning. The MIT Press.
  - Duda, Hart & Stork (2000), Pattern classification, John Wiley & Sons.
  - Han & Kamber (2006), Data mining: concepts and techniques, 2nd ed. Morgan Kaufmann.
  - Tan, Steinbach & Kumar (2006), Introduction to data mining. Addison Wesley.
- As well as from Wikipedia, the free online encyclopedia
- I really thank these authors for their contribution to machine learning and data mining teaching

# Frequent pattern analysis





# What is frequent pattern analysis?

- **Frequent pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- **Motivation:** Finding **inherent regularities** in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - 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.

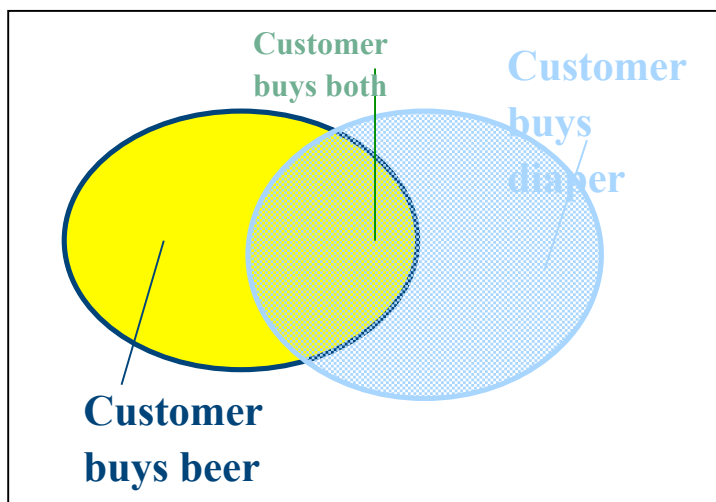


# Why is frequent pattern analysis important ?

- Discloses intrinsic and important properties of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering

# Basic concepts

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



- Itemset  $X = \{x_1, \dots, x_p\}$
- Find all the rules  $X \rightarrow Y$  with minimum support and confidence
  - Support,  $s$ , a priori probability that a transaction contains  $X \cup Y$
  - Confidence,  $c$ , conditional probability that a transaction having  $X$  also contains  $Y$ ,  $P(Y|X)$

Let  $sup_{min} = 50\%$ ,  $conf_{min} = 50\%$

Freq. Pat.:  $\{A:3, B:3, D:4, E:3, AD:3\}$

Association rules:

$A \rightarrow D$  (60%, 100%)

$D \rightarrow A$  (60%, 75%)



# Closed patterns and max-patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g.,  $\{a_1, \dots, a_{100}\}$  contains  $2^{100} - 1 = \text{about } 1.27 \cdot 10^{30}$  sub-patterns!
- Solution: Mine **closed** patterns and **max-patterns** instead
- An itemset  $X$  is **closed**
  - If  $X$  is frequent and there exists no **super-pattern**  $Y \supset X$ , with at least the same support as  $X$
- An itemset  $X$  is a **max-pattern**
  - If  $X$  is frequent and there exists no frequent **super-pattern**  $Y \supset X$



# Closed patterns and max-patterns

## ■ Example:

- $DB = \{ \langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle \}$
- $Min\_sup\_count = 1.$

## ■ What is the set of closed itemset?

- $\langle a_1, \dots, a_{100} \rangle: 1$
- $\langle a_1, \dots, a_{50} \rangle: 2$

## ■ What is the set of max-pattern?

- $\langle a_1, \dots, a_{100} \rangle: 1$

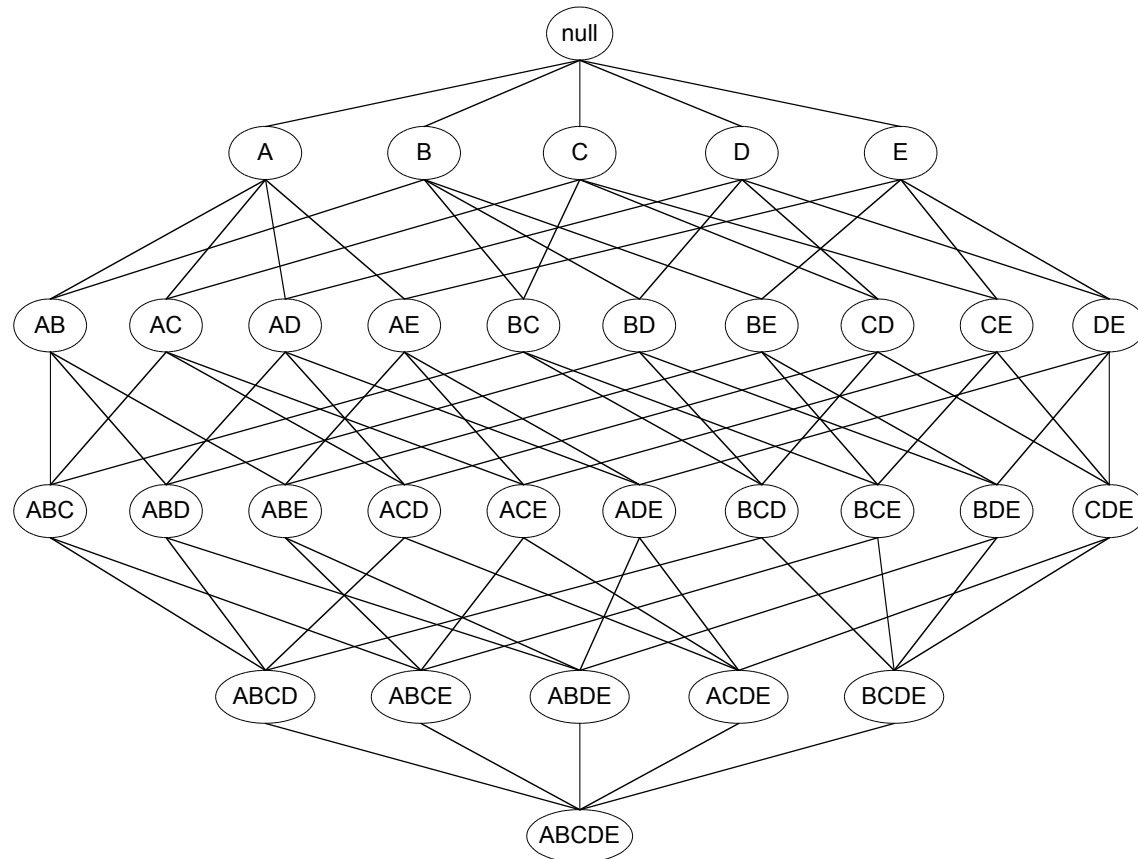


# Scalable methods for mining frequent patterns

- 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)
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin)
  - Vertical data format approach (Charm—Zaki & Hsiao)

# Scalable methods for mining frequent patterns

- Patterns form a **lattice**



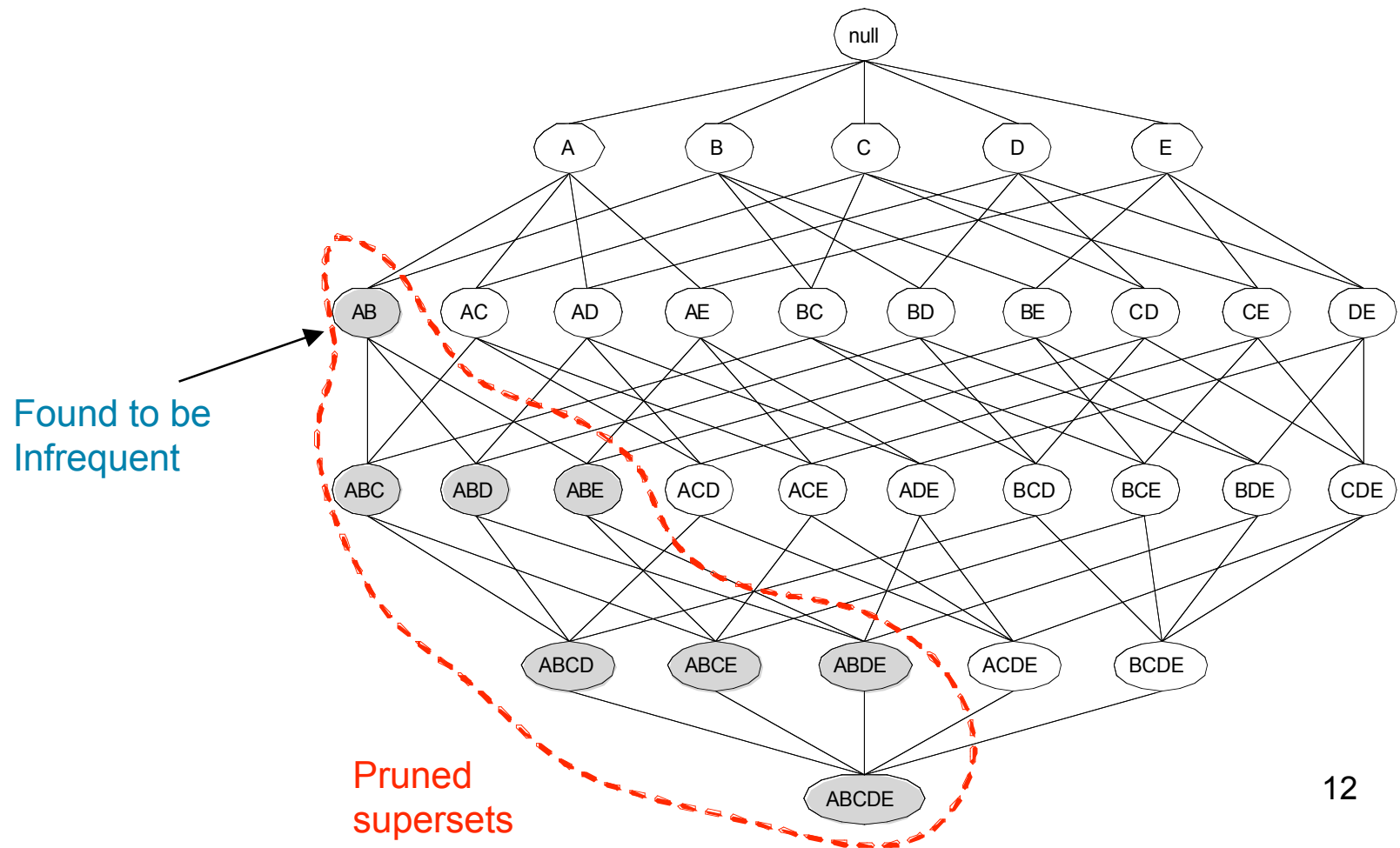


# Apriori: a candidate generation-and-test approach

- **Apriori pruning principle:** If there is any itemset which is infrequent, its superset should not be generated/tested!
- **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

# Apriori: a candidate generation-and-test approach

- This allows to **prune** the lattice



# The Apriori algorithm: an example

$Sup_{min} = 2$

Database

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

1<sup>st</sup> scan

$C_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

$L_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

$L_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

$C_2$

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

2<sup>nd</sup> scan

$C_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

$C_3$

Itemset
{B, C, E}

3<sup>rd</sup> scan

$L_3$

Itemset	sup
{B, C, E}	2



# The Apriori algorithm

## ■ Pseudo-code:

$C_k$ : Candidate itemset of size  $k$

$L_k$ : Frequent itemset of size  $k$

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **do begin**

$C_{k+1} =$  candidates generated from  $L_k$ ;

**for each** transaction  $t$  in database **do**

Increment the count of all candidates in  $C_{k+1}$   
that are contained in  $t$ ;

$L_{k+1} =$  candidates in  $C_{k+1} \geq \text{min\_support}$ ;

**end**

**return**  $\cup_k L_k$ ;



# Important details of Apriori

- How to generate candidates?
  - Step 1: self-joining  $L_k$
  - Step 2: pruning
- How to count supports of candidates?
- 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  $L_3$
  - $C_4 = \{abcd\}$

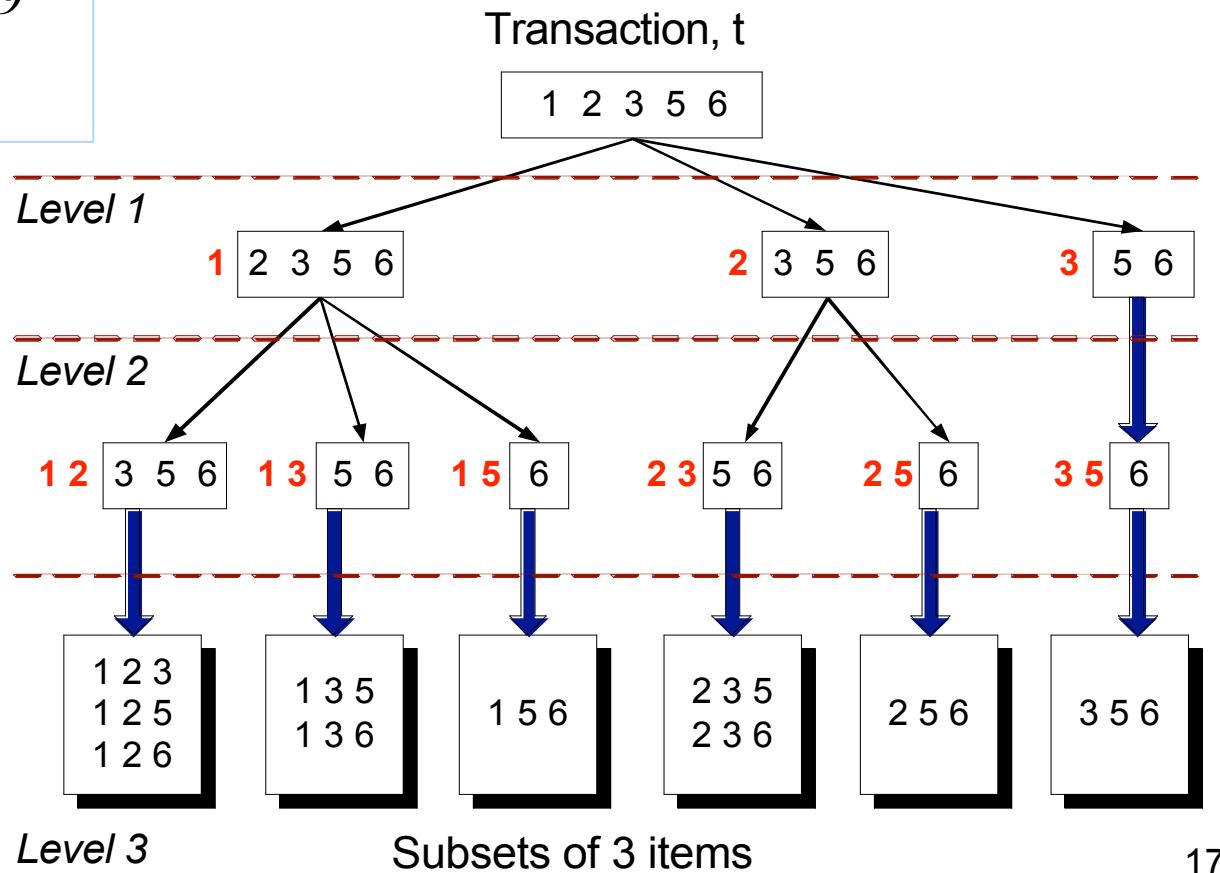
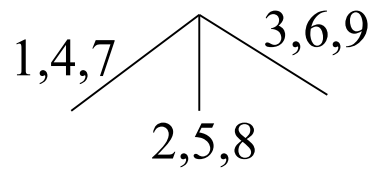


# How to count supports of candidates?

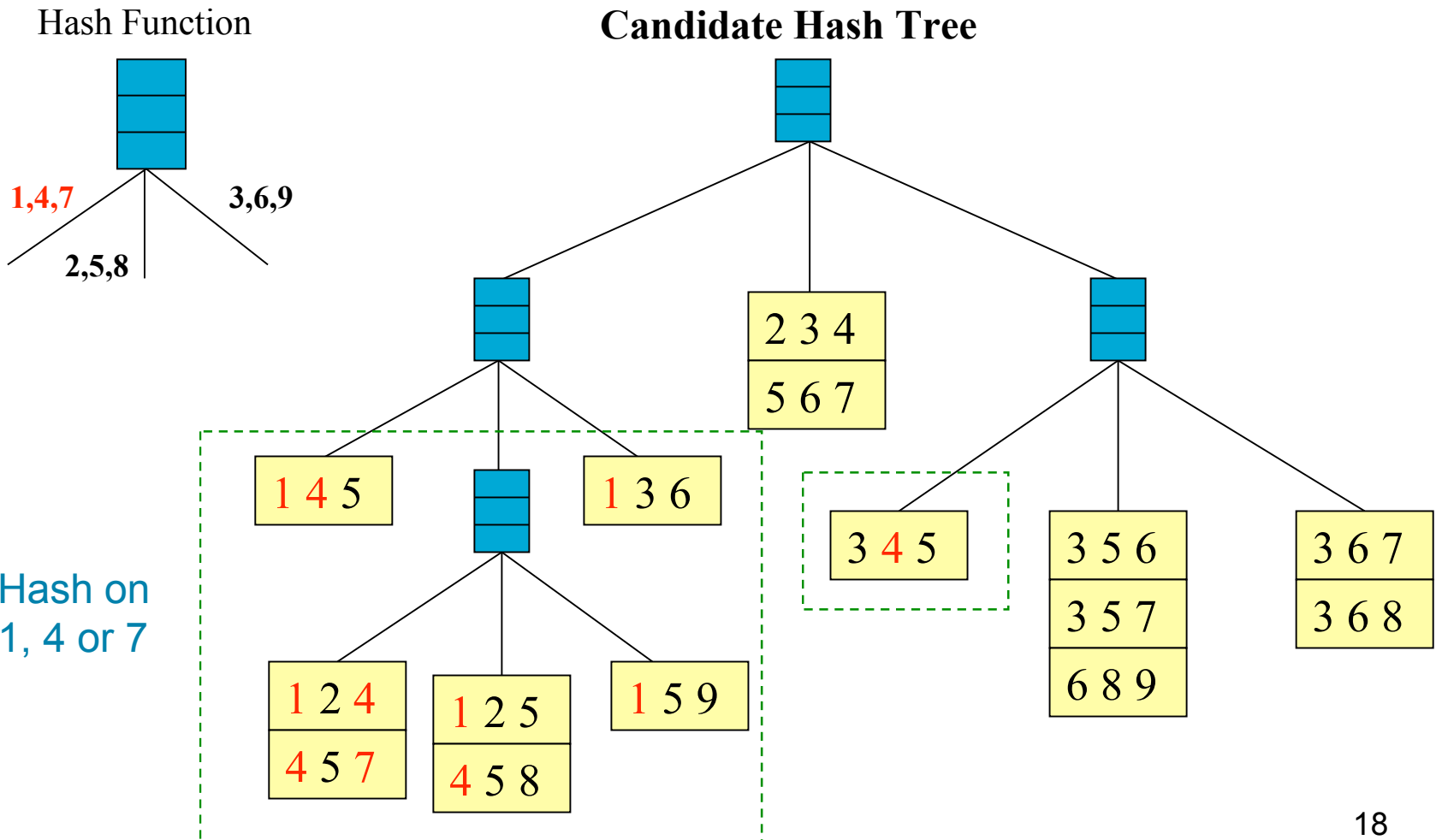
- Why counting supports of candidates is a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a **hash-tree**
  - **Leaf node** of hash-tree contains a list of itemsets and counts
  - **Interior node** contains a hash table
  - **Subset function**: finds all the candidates contained in a transaction

# Example: counting supports of candidates

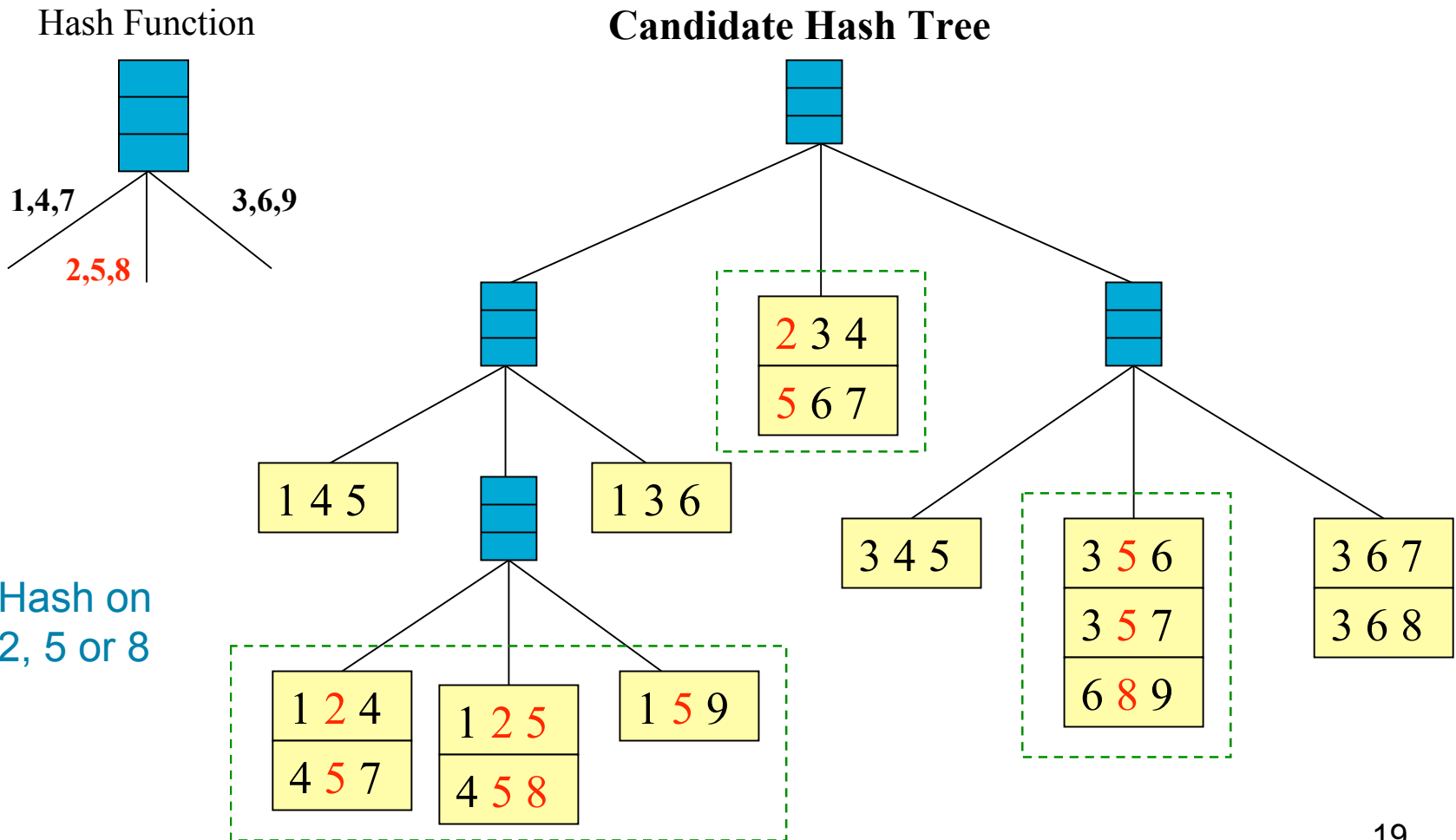
Hash function



# Hash tree

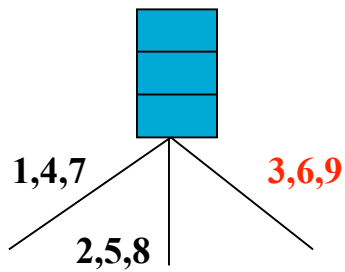


# Hash tree

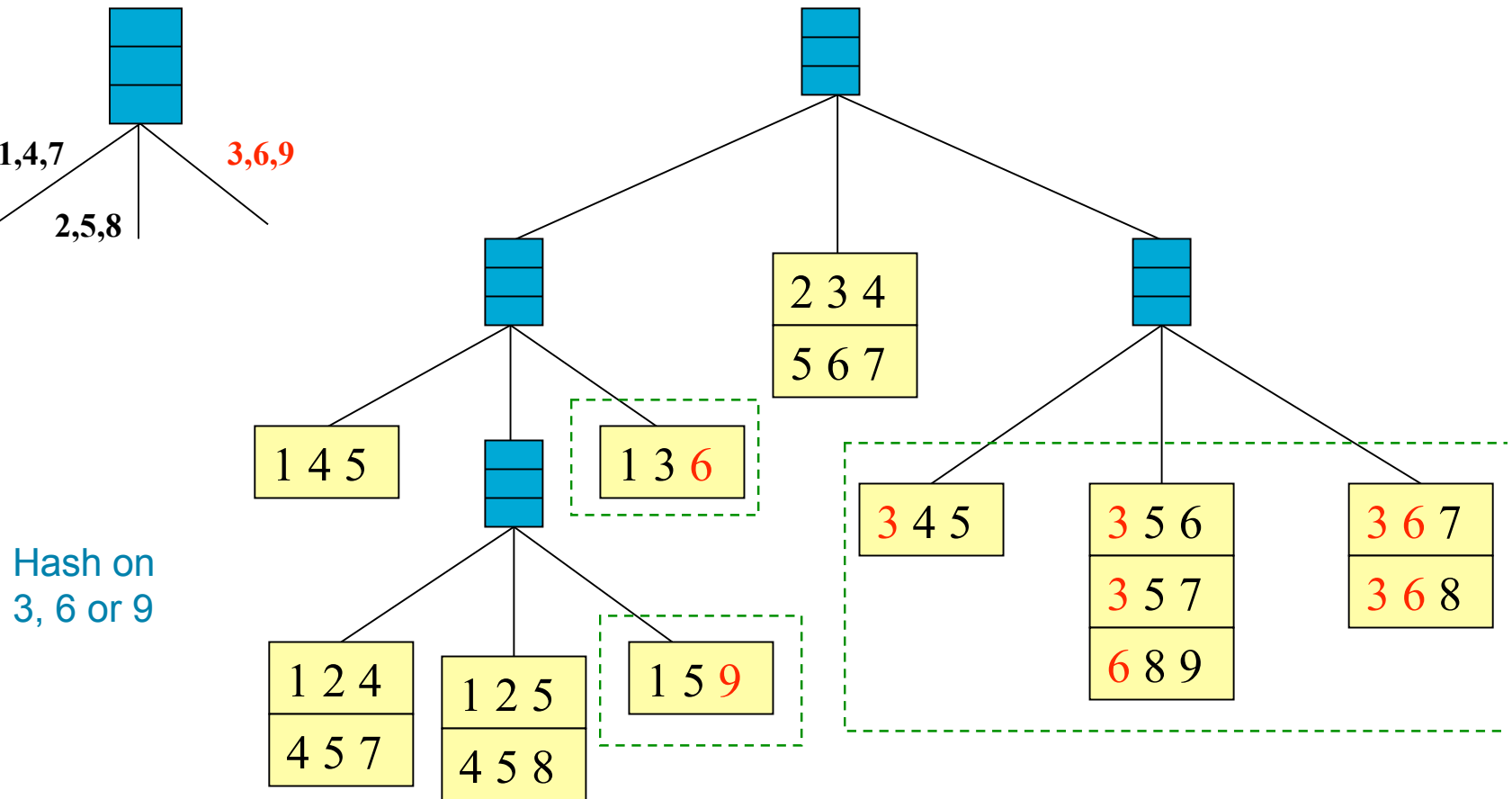


# Hash tree

Hash Function

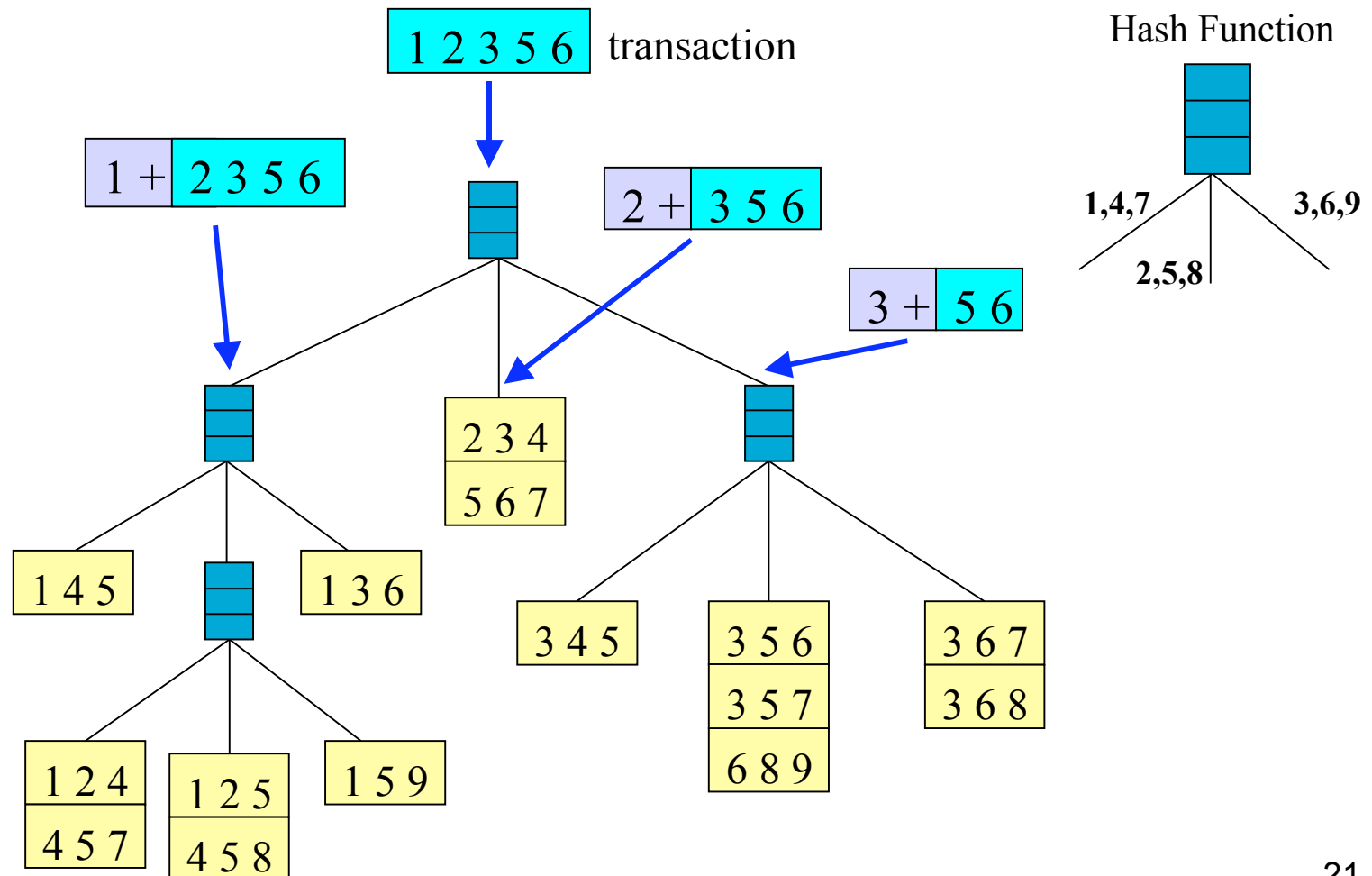


Candidate Hash Tree

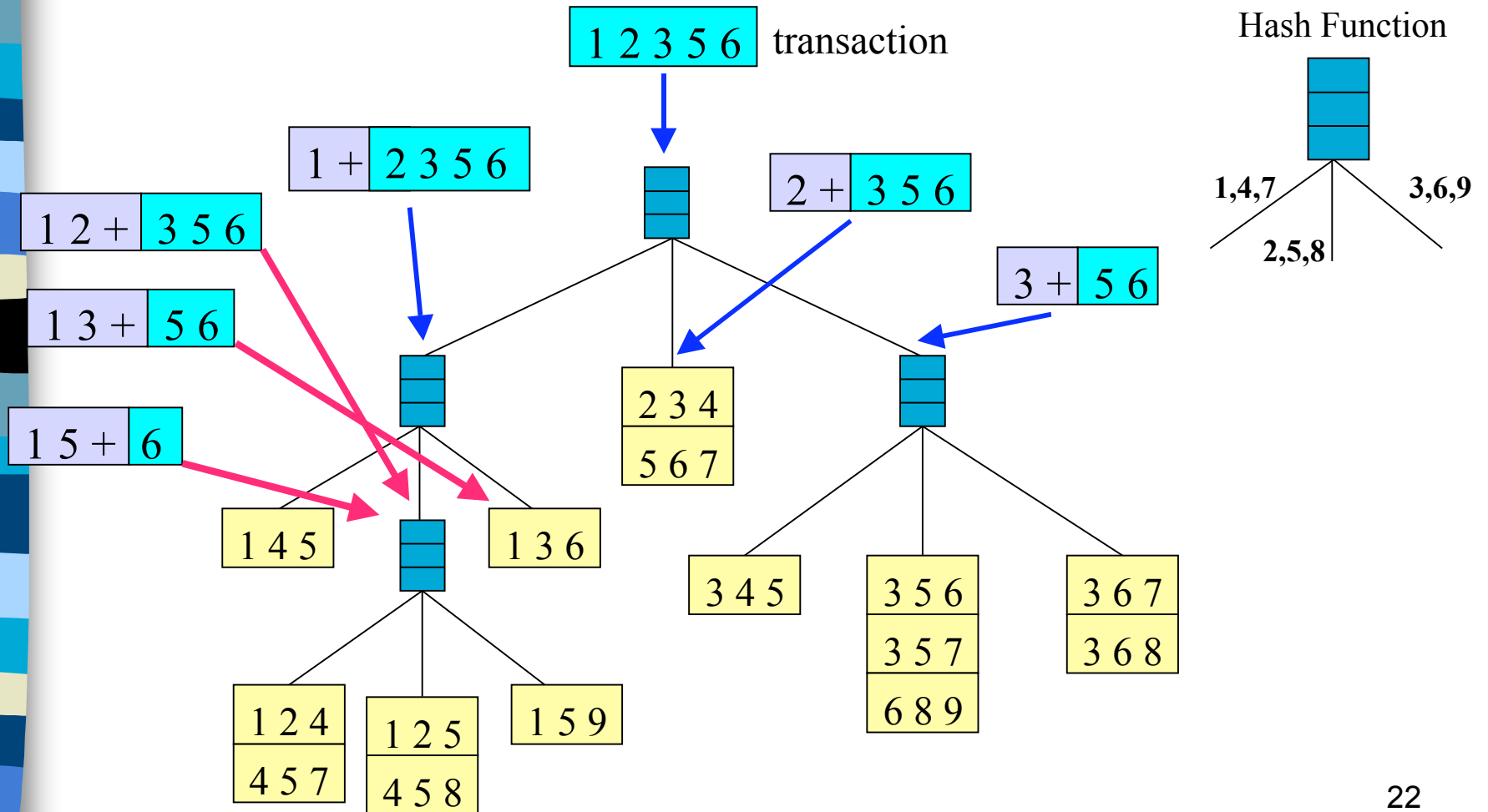


Hash on  
3, 6 or 9

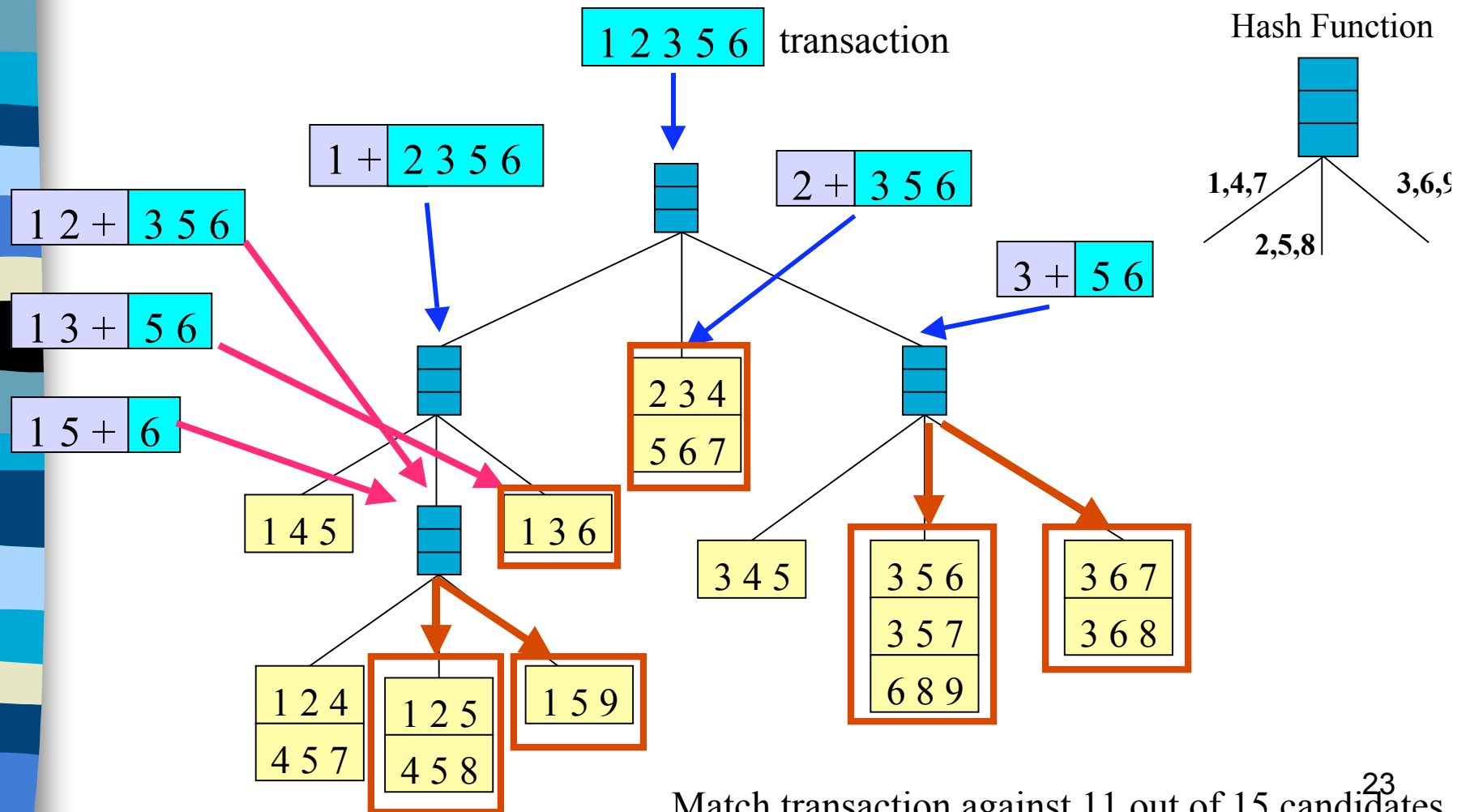
# Subset operation using hash tree



# Subset operation using hash tree



# Subset operation using hash tree



Match transaction against 11 out of 15 candidates



# Challenges of frequent pattern mining

## ■ 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



# Bottleneck of frequent-pattern mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?



# Mining frequent patterns without candidate generation

- Grow long patterns from short ones using local frequent items
  - “abc” is a frequent pattern
  - Get all transactions having “abc”: DB|abc
  - “d” is a local frequent item in DB|abc → abcd is a frequent pattern

# Construct FP-tree from a transaction database

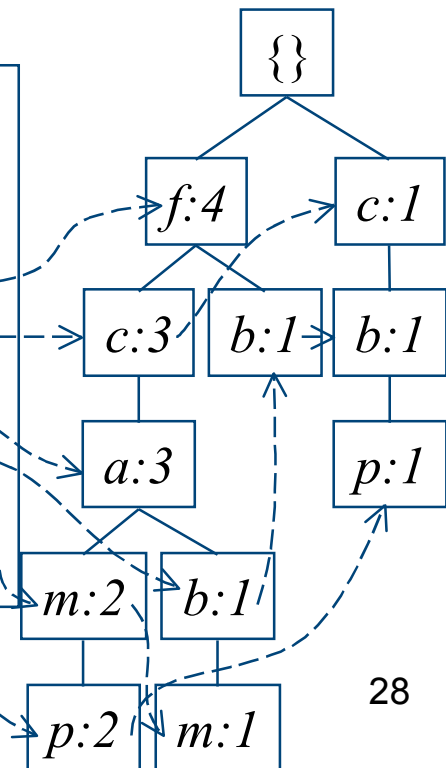
<u>TID</u>	<u>Items bought</u>	<u>(ordered) frequent items</u>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

*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	
<u>Item frequency head</u>	
f	4
c	4
a	3
b	3
m	3
p	3

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





# Benefits of the FP-tree structure

- This is called a **suffix tree** in computer sciences
- **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)



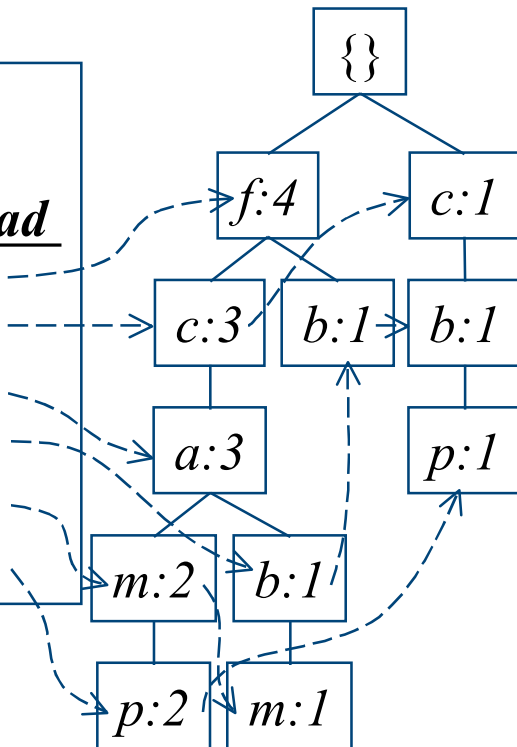
# Partition Patterns and Databases

- 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-redundancy

# 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		
<i>Item</i>	<i>frequency</i>	<i>head</i>
<i>f</i>	4	
<i>c</i>	4	
<i>a</i>	3	
<i>b</i>	3	
<i>m</i>	3	
<i>p</i>	3	

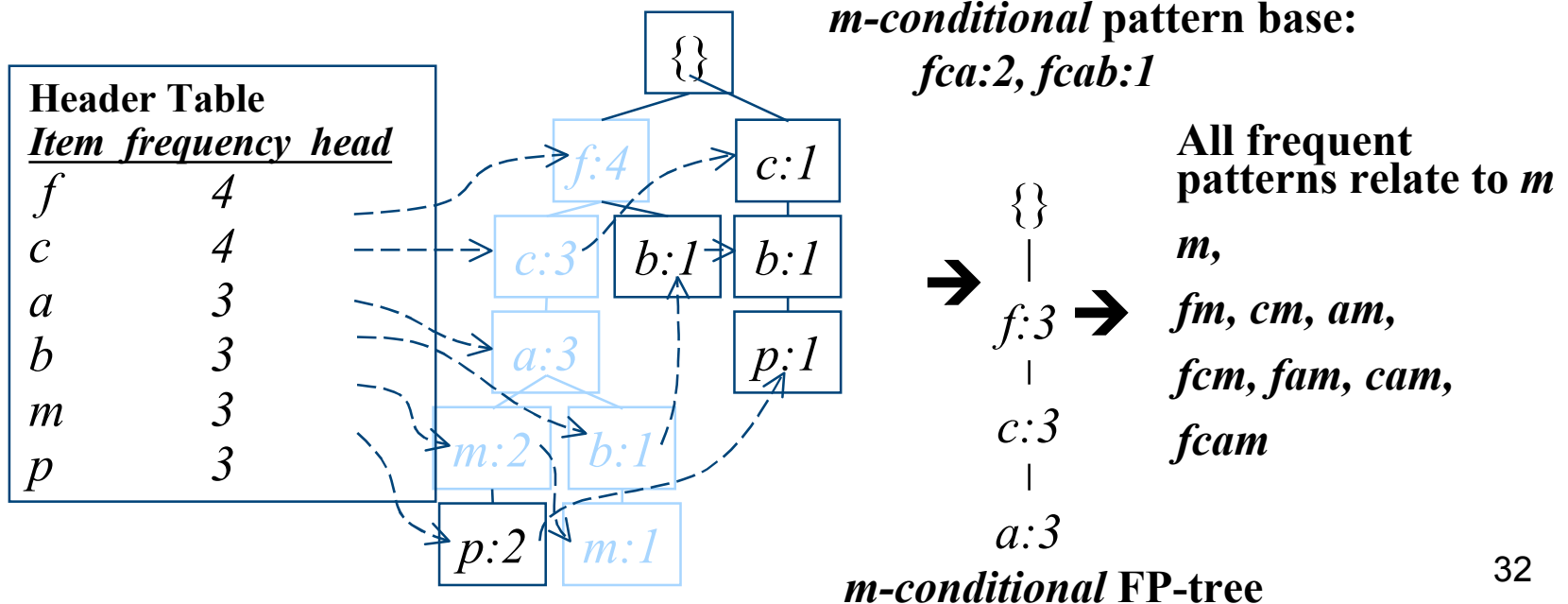


*Conditional pattern bases*

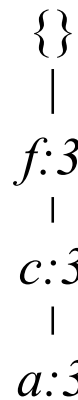
<i>item</i>	<i>cond. pattern base</i>
<i>c</i>	<i>f:3</i>
<i>a</i>	<i>fc:3</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>
<i>m</i>	<i>fca:2, fcab:1</i>
<i>p</i>	<i>fcam:2, cb:1</i>

# 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

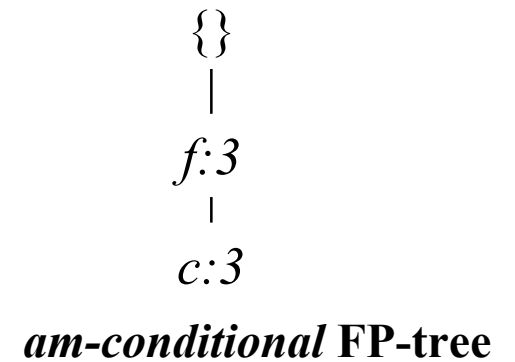


# Recursion: mining each conditional FP-tree

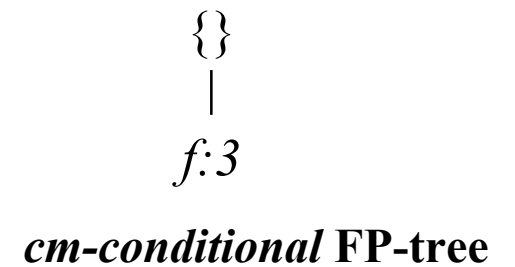


*m*-conditional FP-tree

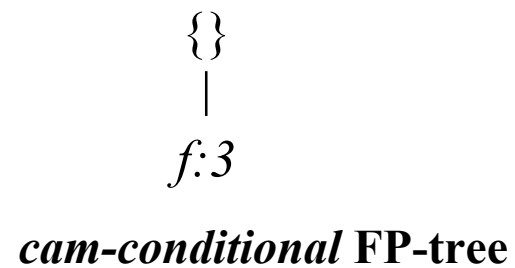
Cond. pattern base of “am”: (fc:3)



Cond. pattern base of “cm”: (f:3)



Cond. pattern base of “cam”: (f:3)

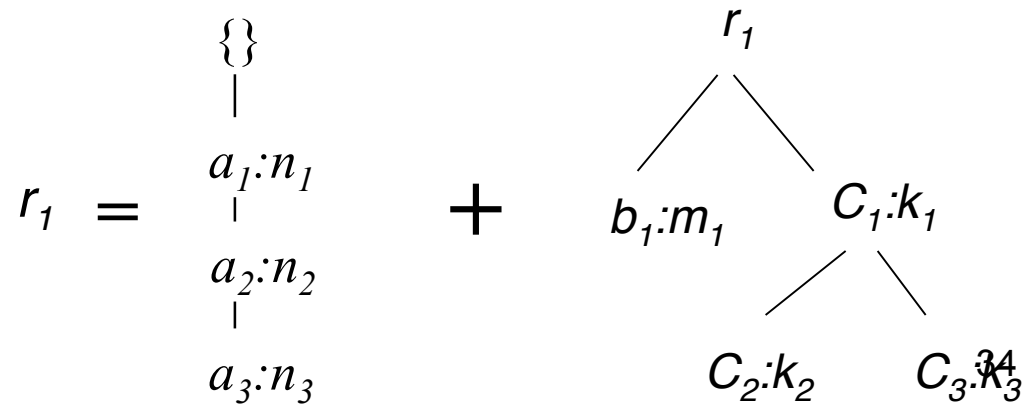
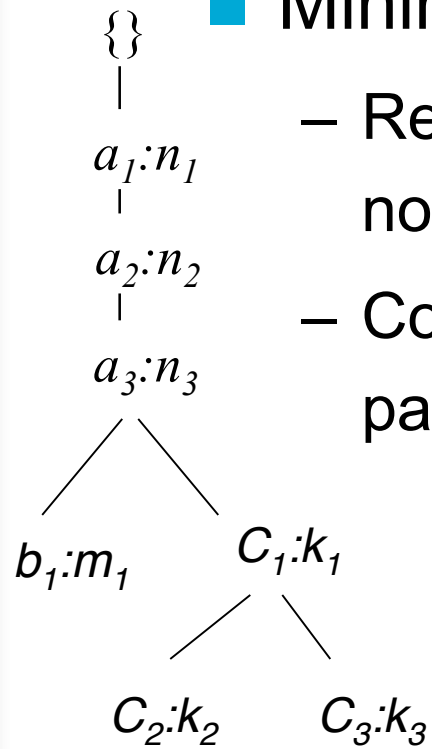


# A special case: single prefix path in FP-tree

- 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





# Mining frequent patterns with FP-trees

- 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

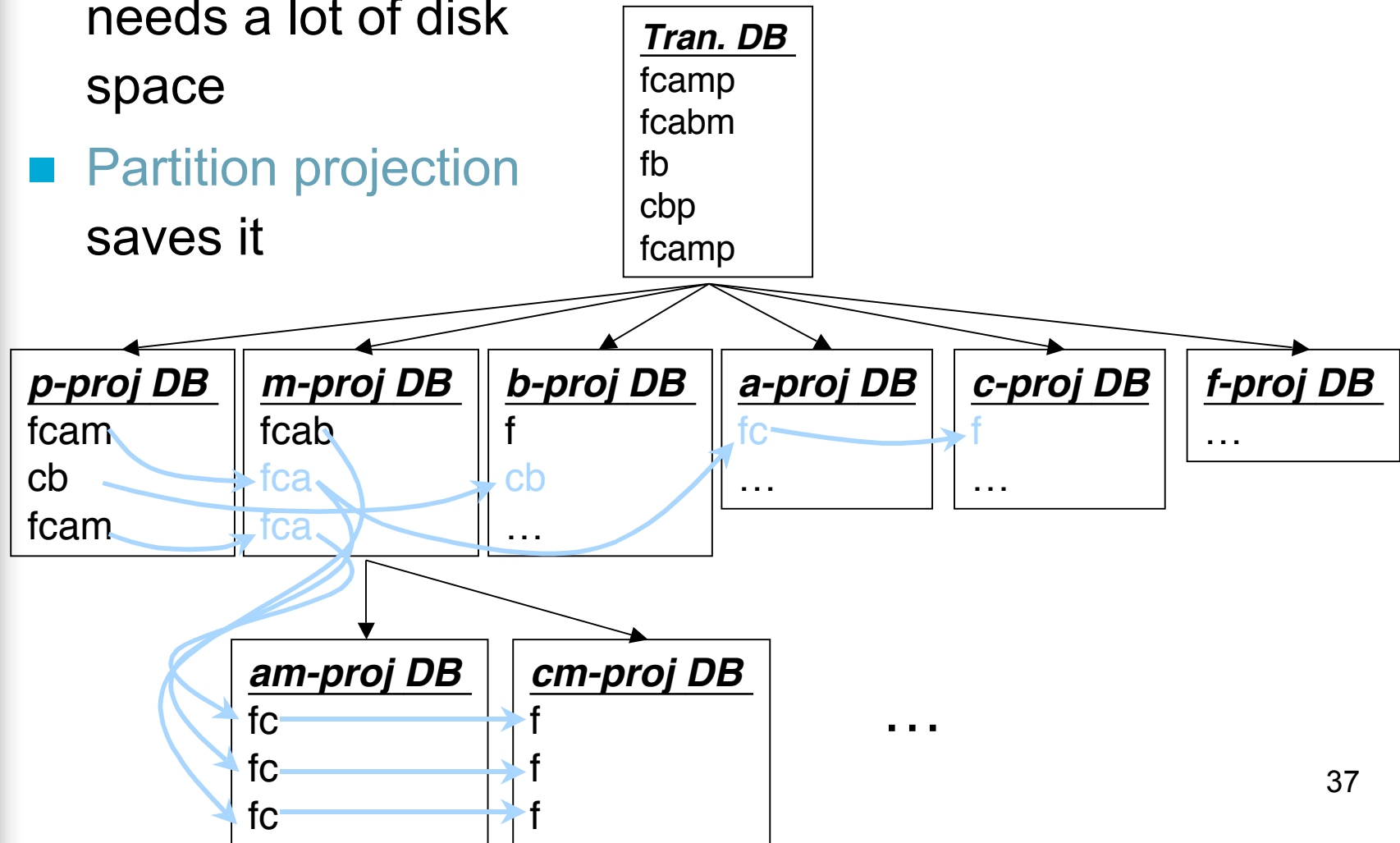


# Scaling FP-growth by DB projection

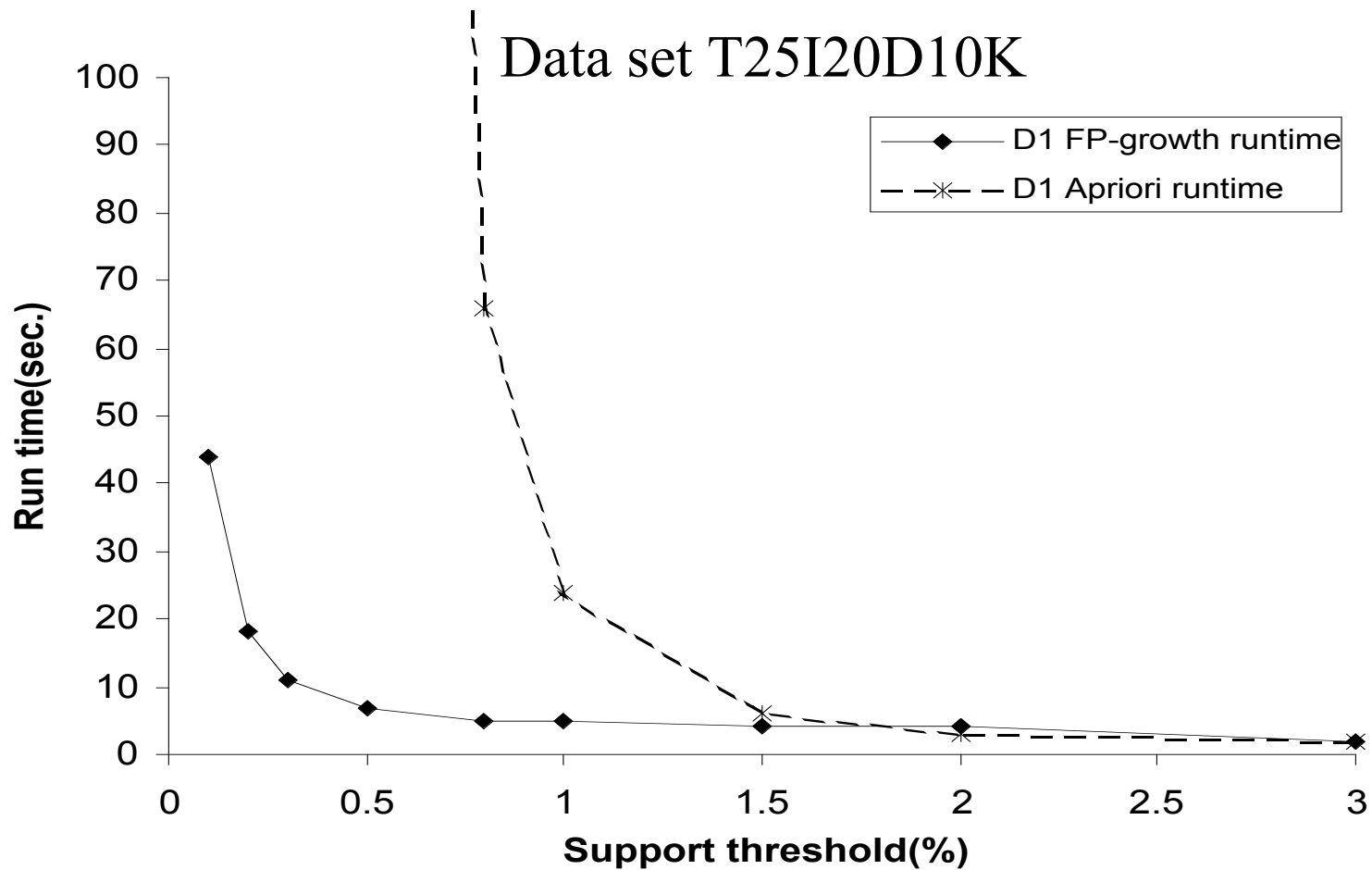
- FP-tree cannot fit in memory?—DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. partition projection techniques
  - Parallel projection is space costly

# Partition-based projection

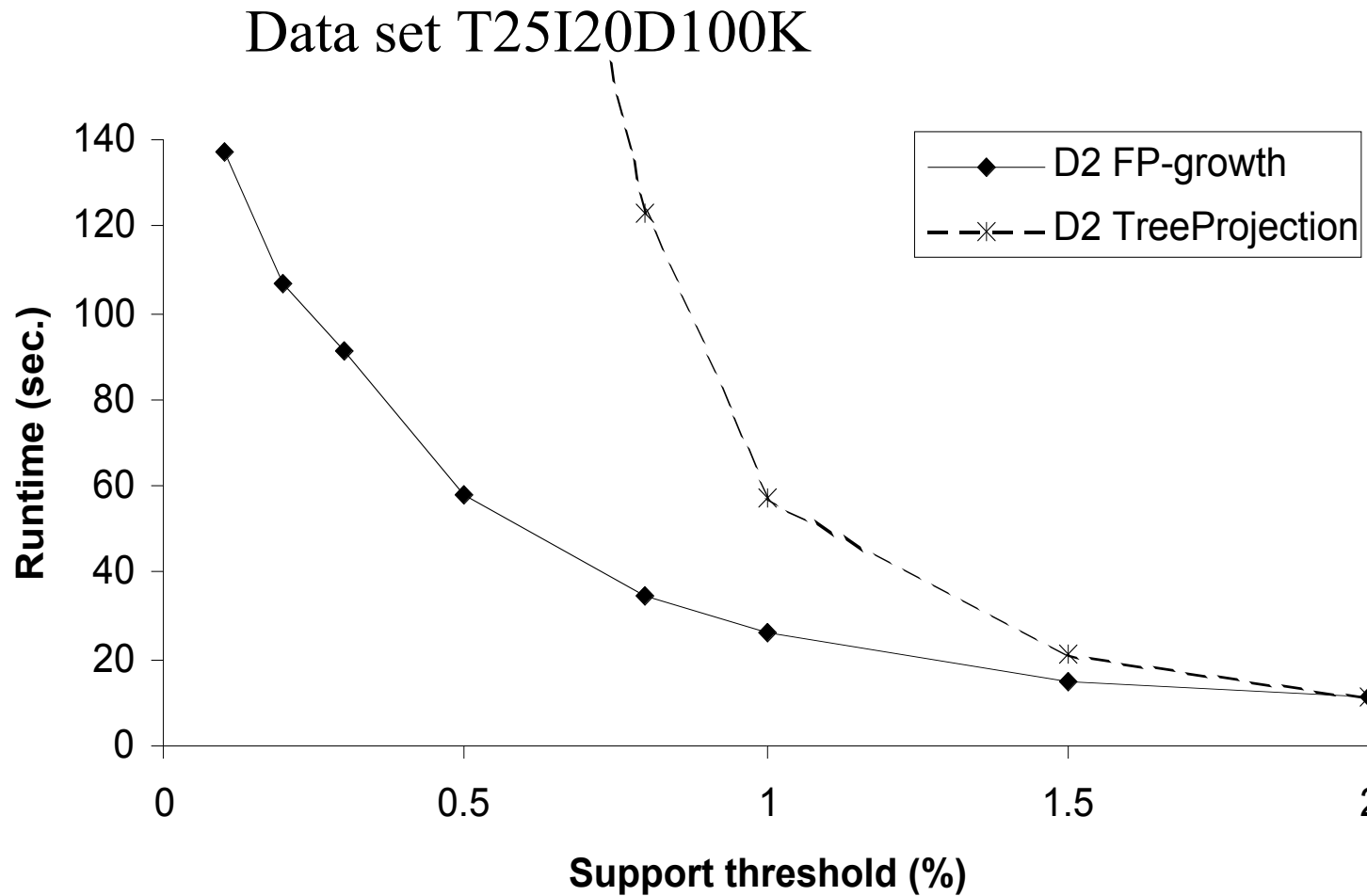
- Parallel projection needs a lot of disk space
- Partition projection saves it



# FP-Growth vs. Apriori: scalability with the support threshold



# FP-Growth vs. tree-projection: scalability with the support threshold

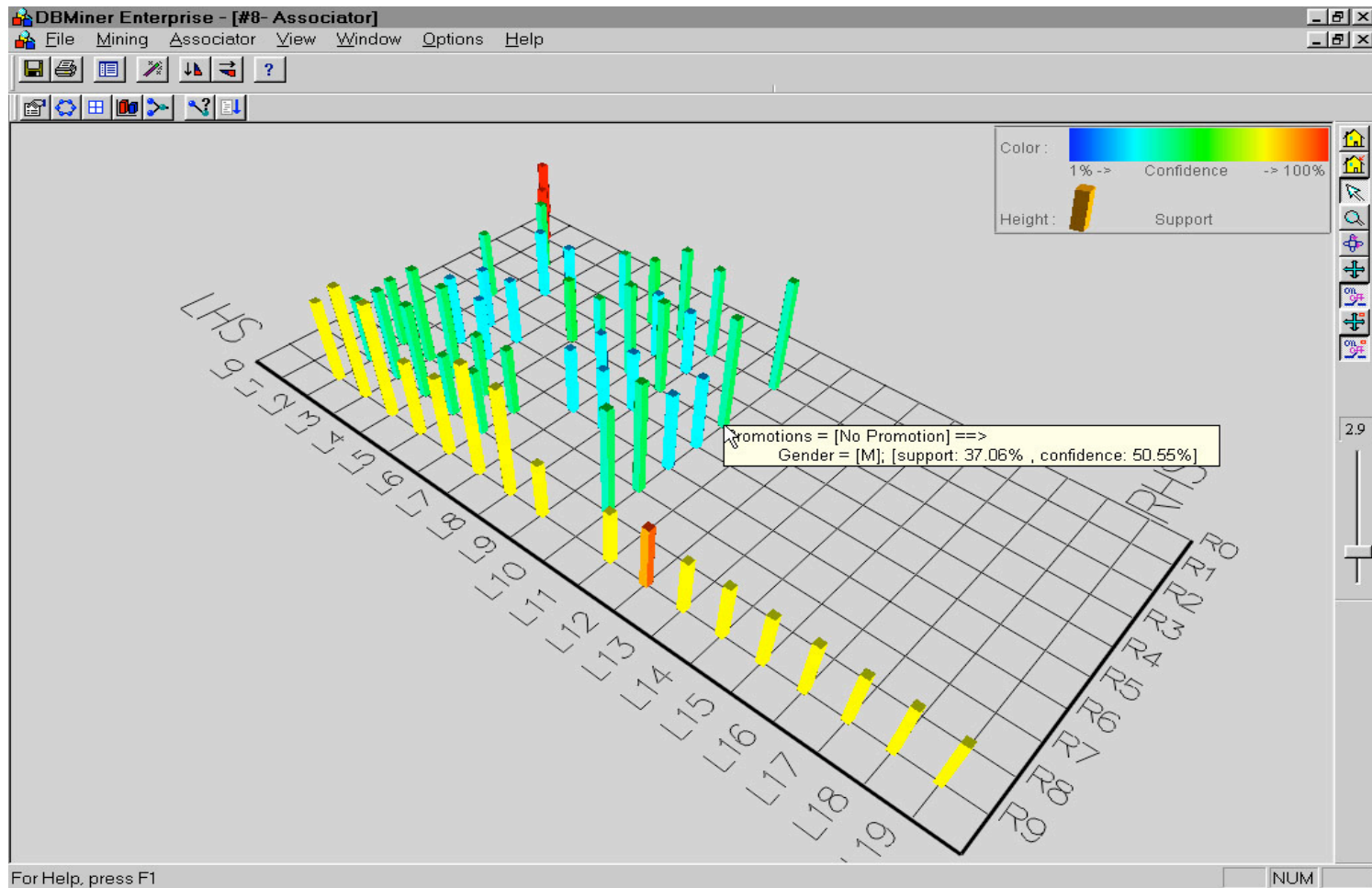




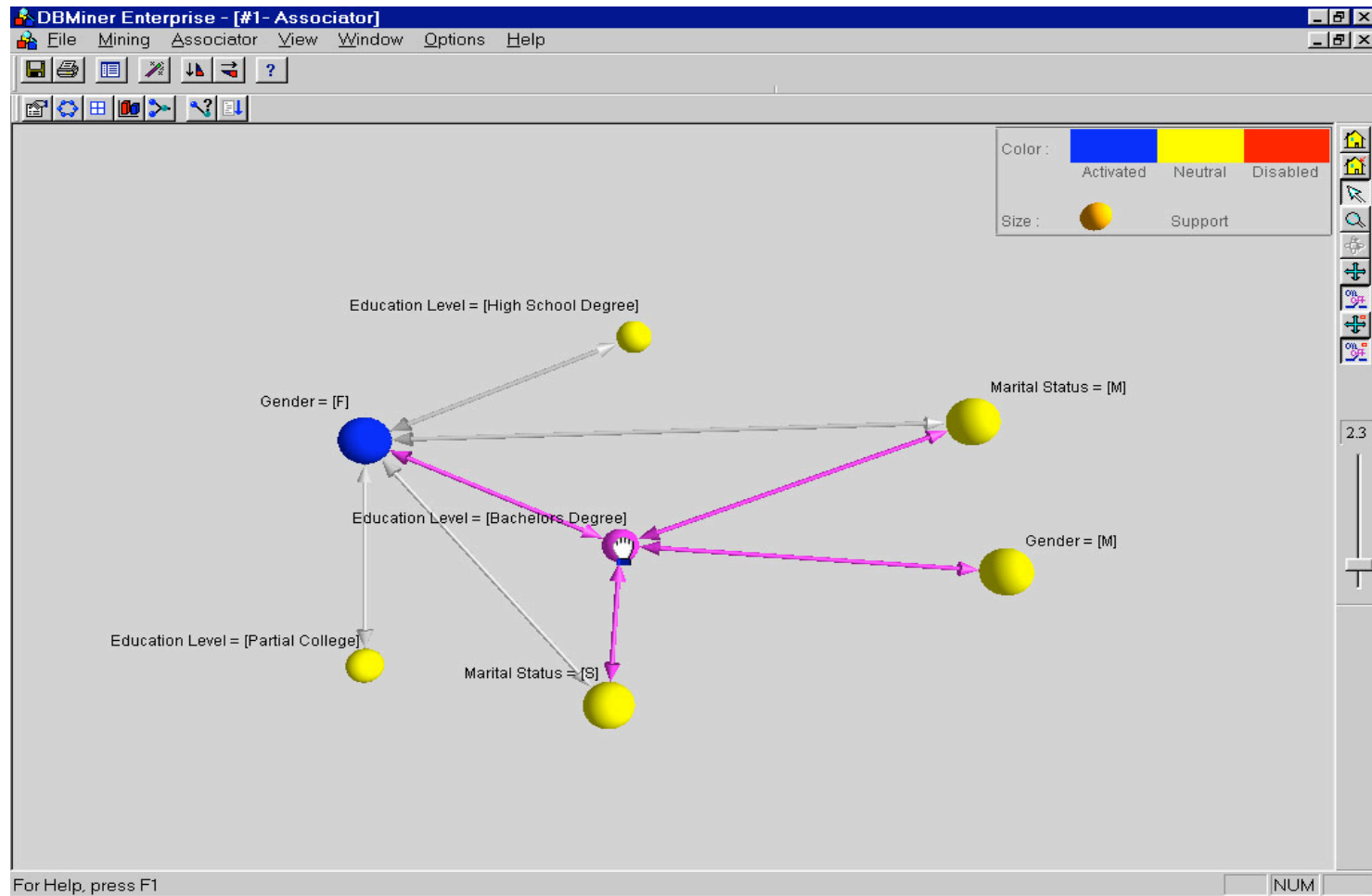
# Why is FP-Growth highly performant?

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads 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

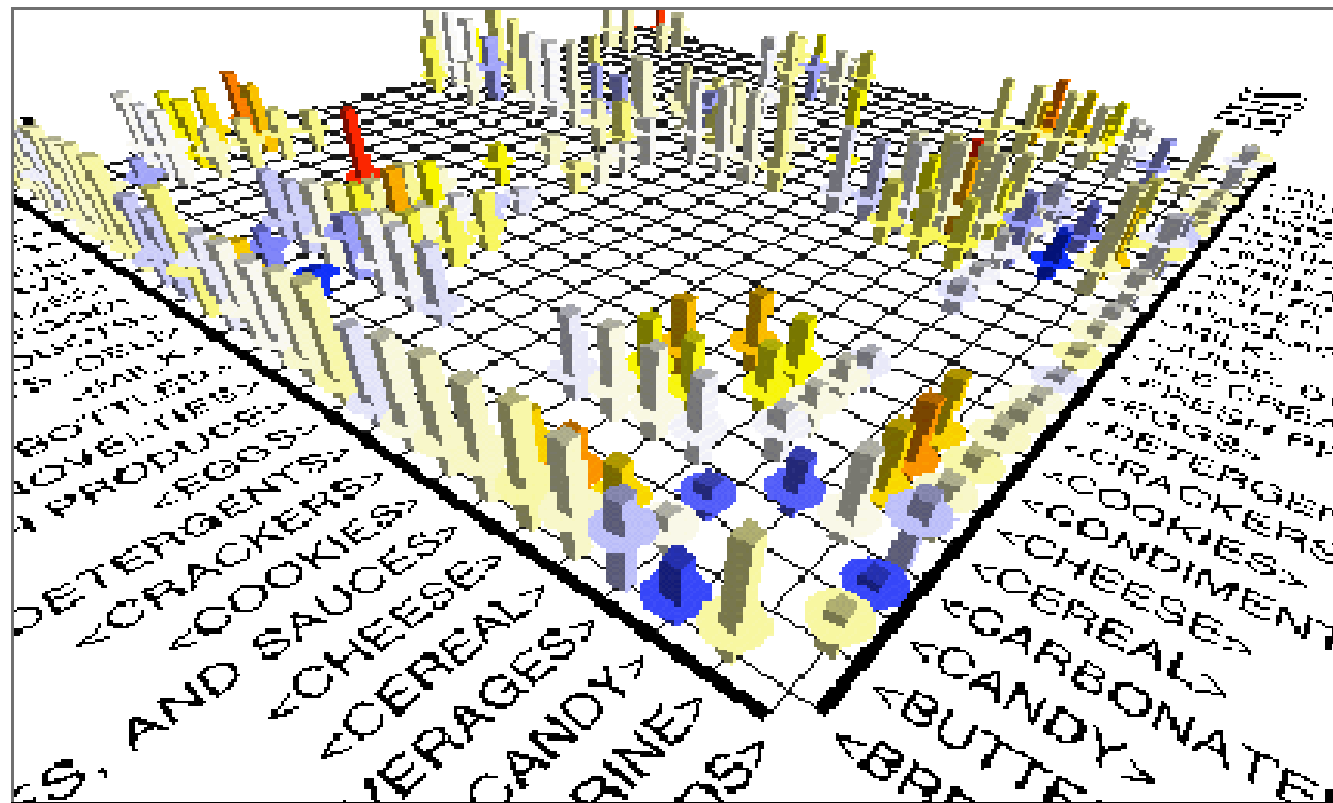
# Visualization of association rules: plane graph



# Visualization of association rules: rule graph



# Visualization of association rules (SGI/MineSet 3.0)



# Mining multiple-level association rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support

uniform support

Level 1  
min\_sup = 5%

Level 2  
min\_sup = 5%

Milk  
[support = 10%]

2% Milk  
[support = 6%]

Skim Milk  
[support = 4%]

reduced support

Level 1  
min\_sup = 5%

Level 2  
min\_sup = 3%



## Multi-level association: redundancy filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk  $\Rightarrow$  wheat bread [support = 8%, confidence = 70%]
  - 2% milk  $\Rightarrow$  wheat bread [support = 2%, confidence = 72%]

We say the first rule is an ancestor of the second rule.

- A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.



# Mining multi-dimensional association

- Single-dimensional rules:

$\text{buys}(X, \text{"milk"}) \Rightarrow \text{buys}(X, \text{"bread"})$

- Multi-dimensional rules:  $\geq 2$  dimensions or predicates

- Inter-dimension assoc. rules (*no repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{occupation}(X, \text{"student"}) \Rightarrow \text{buys}(X, \text{"coke"})$

- hybrid-dimension assoc. rules (*repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{buys}(X, \text{"popcorn"}) \Rightarrow \text{buys}(X, \text{"coke"})$

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach



# Interestingness measure: correlations

- *play basketball*  $\Rightarrow$  *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%
- *play basketball*  $\Rightarrow$  *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: **lift**

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



# Interestingness measure: correlations

## ■ Example

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

$$lift(B, C) = \frac{2000 / 5000}{3000 / 5000 * 3750 / 5000} = 0.89$$

$$lift(B, \neg C) = \frac{1000 / 5000}{3000 / 5000 * 1250 / 5000} = 1.33$$

# Are *lift* and $\chi^2$ good measures of correlation?

- “Buy walnuts  $\Rightarrow$  buy milk [1%, 80%]” is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- Many interestingness measures

$$all\_conf = \frac{sup(X)}{\max\_item\_sup(X)}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	$\sim$ m, c	c
No Coffee	m, $\sim$ c	$\sim$ m, $\sim$ c	$\sim$ c
Sum(col.)	m	$\sim$ m	$\Sigma$

$$coh = \frac{sup(X)}{|universe(X)|}$$

DB	m, c	$\sim$ m, c	m $\sim$ c	$\sim$ m $\sim$ c	lift	all-conf	coh	$\chi^2$
A1	1000	100	100	10,000	9.26	0.91	0.83	9055
A2	100	1000	1000	100,000	8.44	0.09	0.05	670
A3	1000	100	10000	100,000	9.18	0.09	0.09	840
A4	1000	1000	1000	1000	1	0.5	0.33	0

# Which measures should be used?

symbol	measure	range	formula
$\phi$	$\phi$ -coefficient	-1...1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
$Q$	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},\bar{B}) + P(A,\bar{B})P(\bar{A},B)}$
$Y$	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
$k$	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
$PS$	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A, B) - P(A)P(B)$
$F$	Certainty factor	-1 ... 1	$\max(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)})$
$AV$	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
$K$	Klogsen's Q	-0.33 ... 0.38	$\frac{\sqrt{P(A, B)} \max(P(B A) - P(B), P(A B) - P(A))}{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}$
$g$	Goodman-kruskal's	0 ... 1	$\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{2 - \max_j P(A_j) - \max_k P(B_k)}$
$M$	Mutual Information	0 ... 1	$\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i) \log P(A_i) \log P(A_i), -\sum_i P(B_i) \log P(B_i) \log P(B_i))}$
$J$	J-Measure	0 ... 1	$\max(P(A, B) \log(\frac{P(B A)}{P(B)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{B} \bar{A})}{P(\bar{B})}), P(A, B) \log(\frac{P(A B)}{P(A)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{A} \bar{B})}{P(\bar{A})})$
$G$	Gini index	0 ... 1	$\max(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A}[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2, P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B}[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2)$
$s$	support	0 ... 1	$P(A, B)$
$c$	confidence	0 ... 1	$\max(P(B A), P(A B))$
$L$	Laplace	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
$IS$	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
$\gamma$	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
$\alpha$	all_confidence	0 ... 1	$\frac{P(A,B)}{\max(P(A), P(B))}$
$o$	odds ratio	0 ... $\infty$	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(A,\bar{B})}$
$V$	Conviction	0.5 ... $\infty$	$\max(\frac{P(A)P(\bar{B})}{P(\bar{A}\bar{B})}, \frac{P(B)P(\bar{A})}{P(\bar{B}\bar{A})})$
$\lambda$	lift	0 ... $\infty$	$\frac{P(A,B)}{P(A)P(B)}$
$S$	Collective strength	0 ... $\infty$	$\frac{P(A,B)+P(\bar{A}\bar{B})}{P(A)P(B)+P(\bar{A})P(\bar{B})} \times \frac{1-P(A)P(B)-P(\bar{A})P(\bar{B})}{1-P(A,B)-P(\bar{A}\bar{B})}$
$\chi^2$	$\chi^2$	0 ... $\infty$	$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$