Lesson B-3

Frequent Pattern Mining

-FP-Tree Algorithm, Mining Multilevel Association Rules, Mining Multidimensional Association Rules, Mining Rare Patterns, Mining Negative Patterns

FP-Growth Algorithm

- The Apriori Algorithm has a major shortfall
  - required multiple scans of the databases (datasets) to check the support count of each item and itemsets
- FP-growth is an improved version of the Apriori Algorithm
  - scans the database twice and used a tree structure (FP-tree) to store all the information

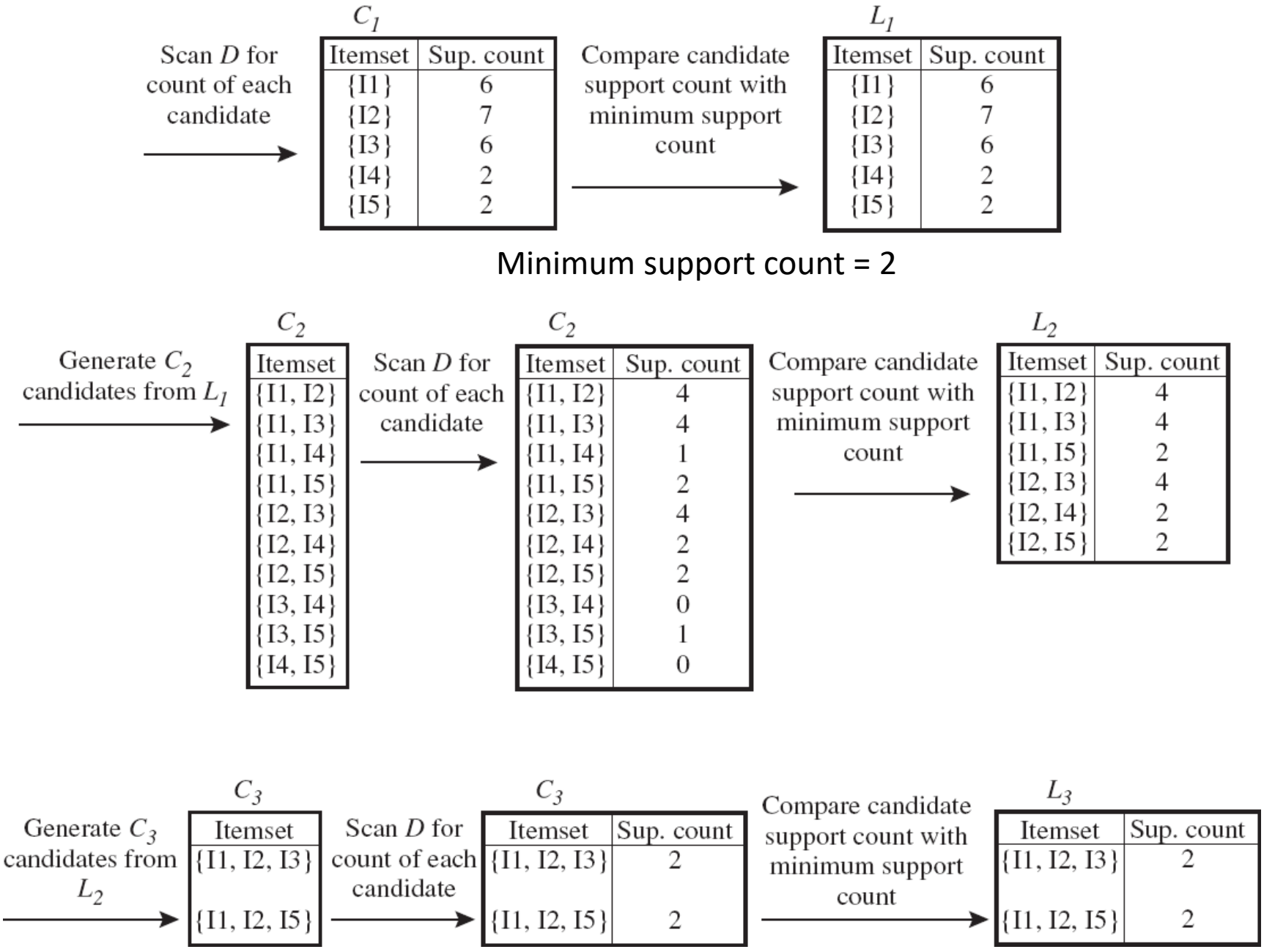
FP-Tree Algorithm

- Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- Step 2: Construct the FP-tree from the above data
- Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- Step 4: Determine the frequent patterns.

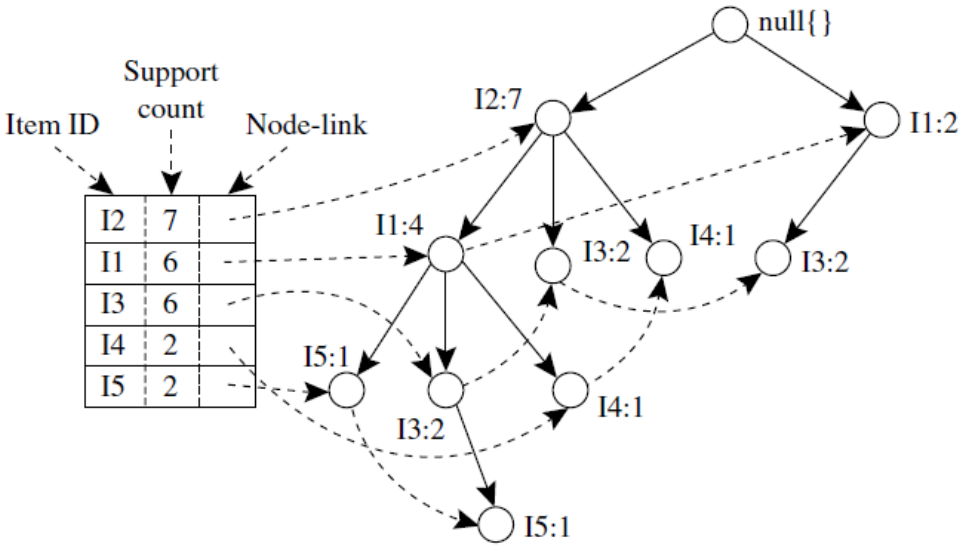
Transactional Database

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

Transactional data for an AllElectronics branch.



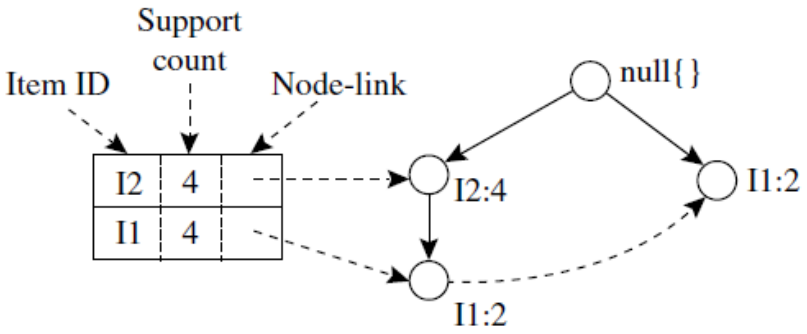
Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.



An FP-tree registers compressed, frequent pattern information.

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.

Algorithm: **FP_growth.** Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

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

Output: The complete set of frequent patterns.

Method:

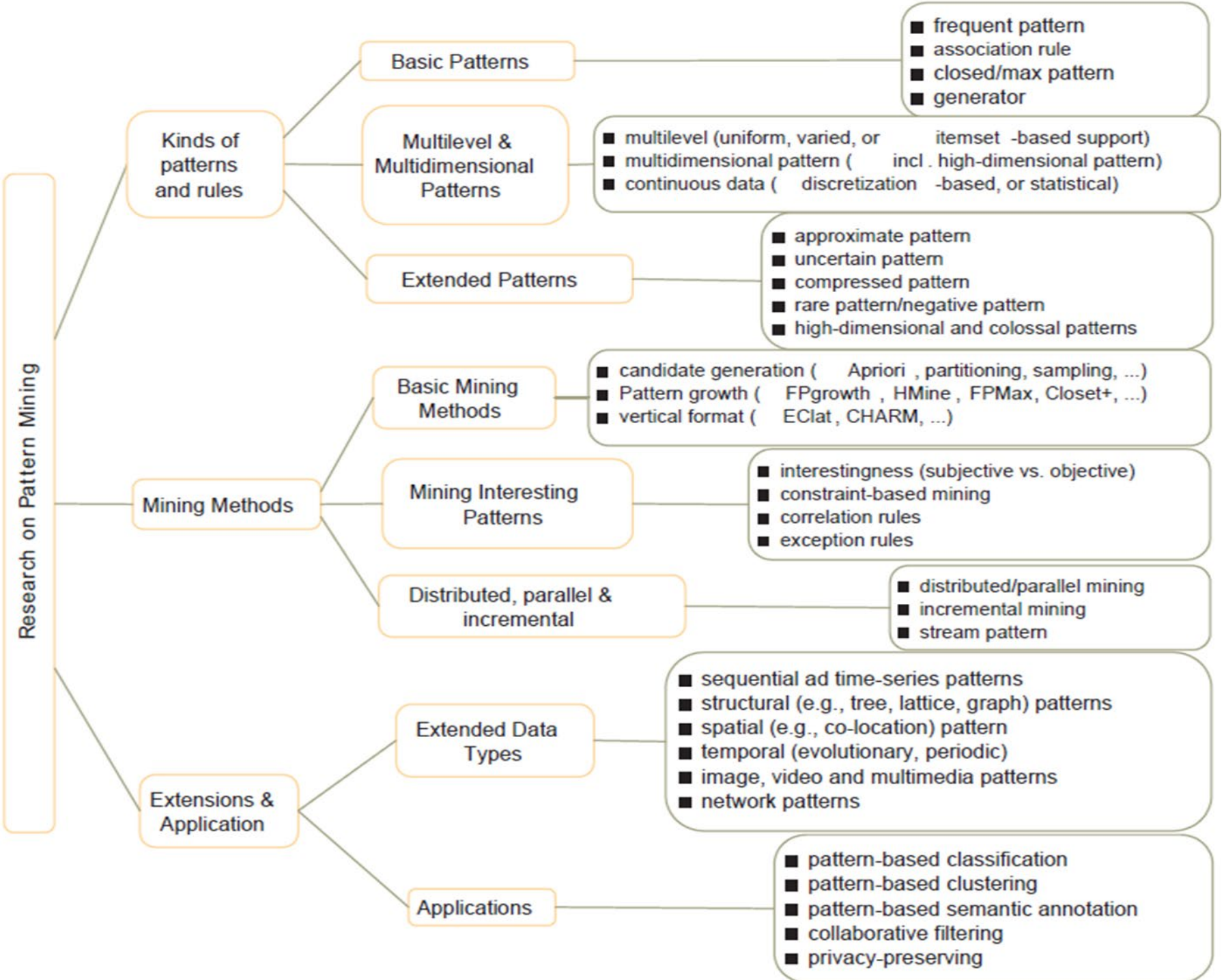
1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following.
   Select and sort the frequent items in *Trans* according to the order of L. Let the sorted frequent item list in *Trans* be $[p|P]$, where p is the first element and P is the remaining list. Call insert_tree($[p|P]$, T), which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.
2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

```
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); }
```

FP-growth algorithm for discovering frequent itemsets without candidate generation.

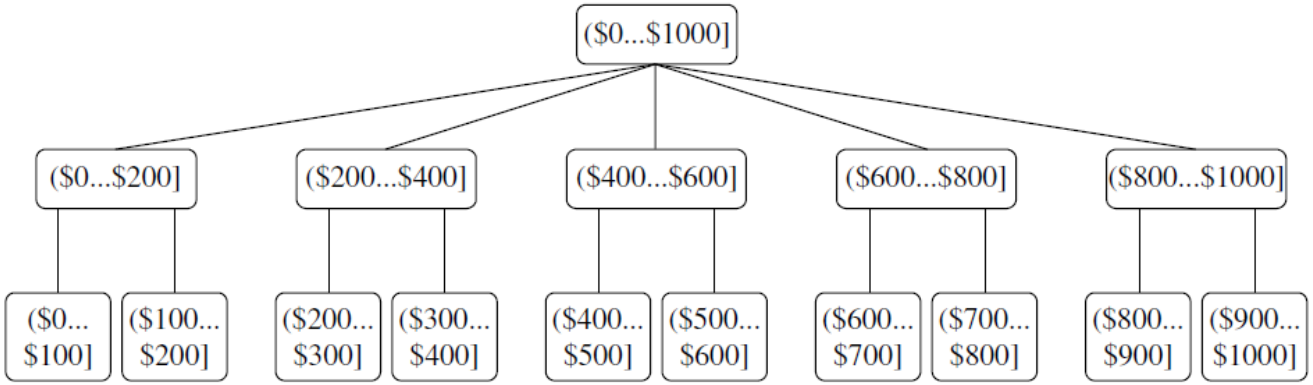
Benefits of the FP-tree

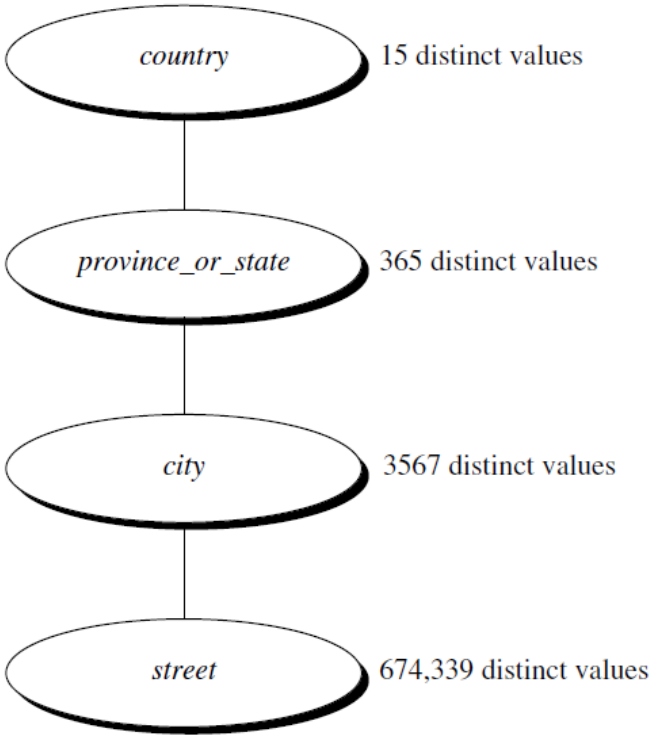
- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database


Research on Pattern Mining
Kinds of patterns and rules
Basic Patterns
frequent pattern
association rule
closed/max pattern
generator
Multilevel & Multidimensional Patterns
multilevel (uniform, varied, or itemset -based support)
multidimensional pattern (incl . high-dimensional pattern)
continuous data (discretization -based, or statistical)
Extended Patterns
approximate pattern
uncertain pattern
compressed pattern
rare pattern/negative pattern
high-dimensional and colossal patterns
Mining Methods
Basic Mining Methods
candidate generation (Apriori , partitioning, sampling, ...)
Pattern growth (FPgrowth , HMine , FPMax, Closet+, ...)
vertical format (EClat , CHARM, ...)
Mining Interesting Patterns
interestingness (subjective vs. objective)
constraint-based mining
correlation rules
exception rules
Distributed, parallel & incremental
distributed/parallel mining
incremental mining
stream pattern
Extensions & Application
Extended Data Types
sequential ad time-series patterns
structural (e.g., tree, lattice, graph) patterns
spatial (e.g., co-location) pattern
temporal (evolutionary, periodic)
image, video and multimedia patterns
network patterns
Applications
pattern-based classification
pattern-based clustering
pattern-based semantic annotation
collaborative filtering
privacy-preserving

Concept Hierarchy Generation

- Concept Hierarchy Generation for Numeric Data
  - A concept hierarchy for a given numerical attribute defines a discretization of the attribute
  - Recursively reduce the data by collecting and replacing low level concepts by higher level concepts
- Concept Hierarchy Generation for Categorical Data
  - Specification of an ordering of attributes explicitly at the schema level by users or experts
    - street < city < state < country
  - Automatic Concept Hierarchy Generation
    - Based on the number of distinct values per attributes


($0...$1000]
($0...$200]
($200...$400]
($400...$600]
($600...$800]
($800...$1000]
($0... $100]
($100... $200]
($200... $300]
($300... $400]
($400... $500]
($500... $600]
($600... $700]
($700... $800]
($800... $900]
($900... $1000]

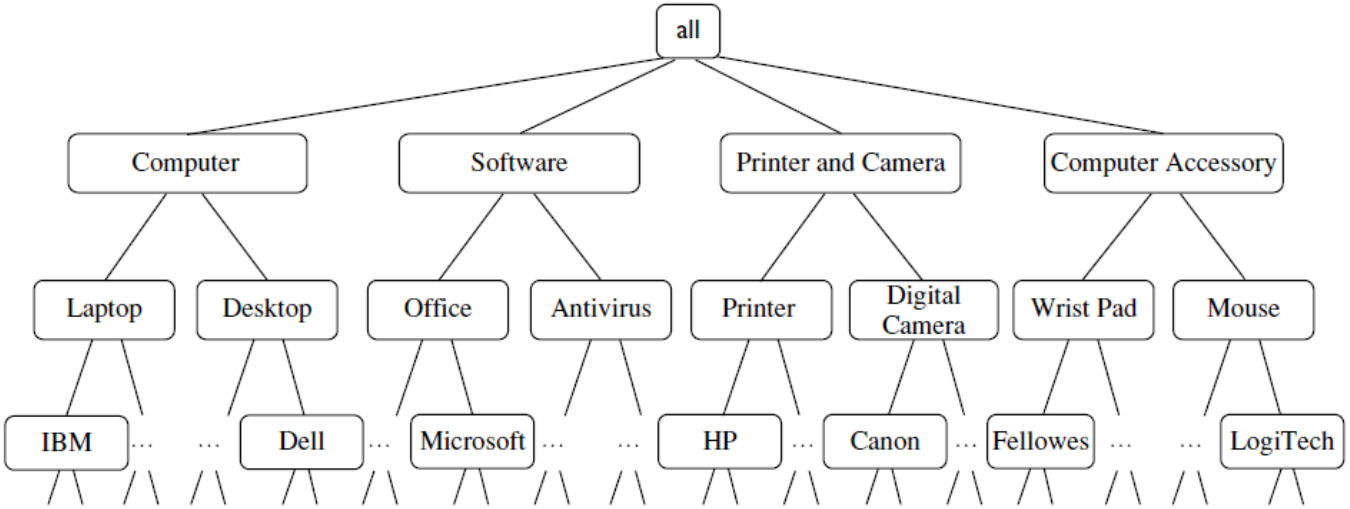

country
15 distinct values
province_or_state
365 distinct values
city
3567 distinct values
street
674,339 distinct values

Mining Multilevel Association Rules

- Data mining systems should provide capabilities for mining association rules at multiple levels of abstraction
- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support

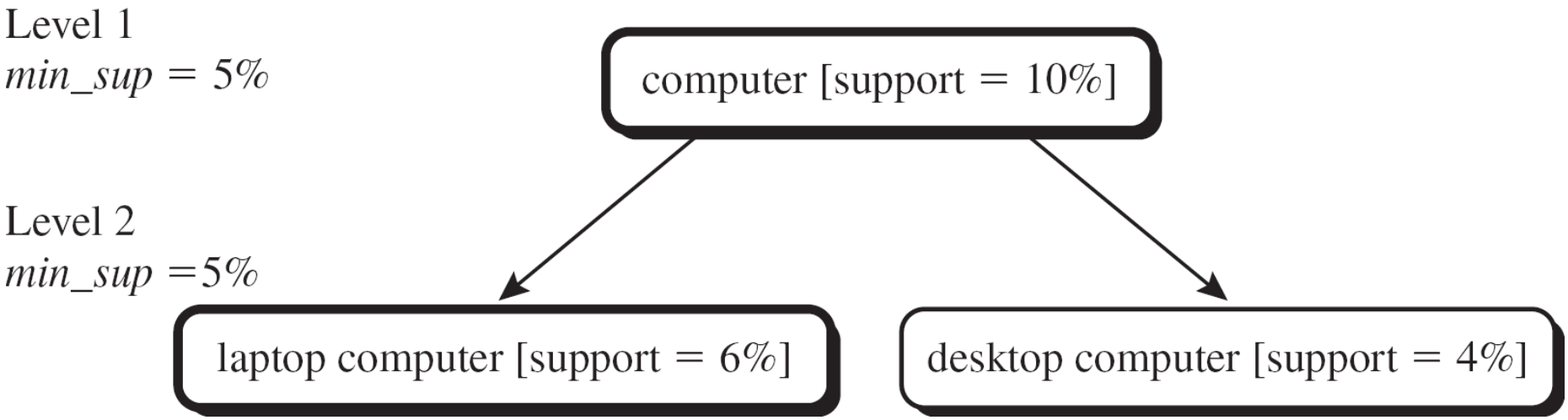
Concept Hierarchy for Computer Items

TID	Items Purchased
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…

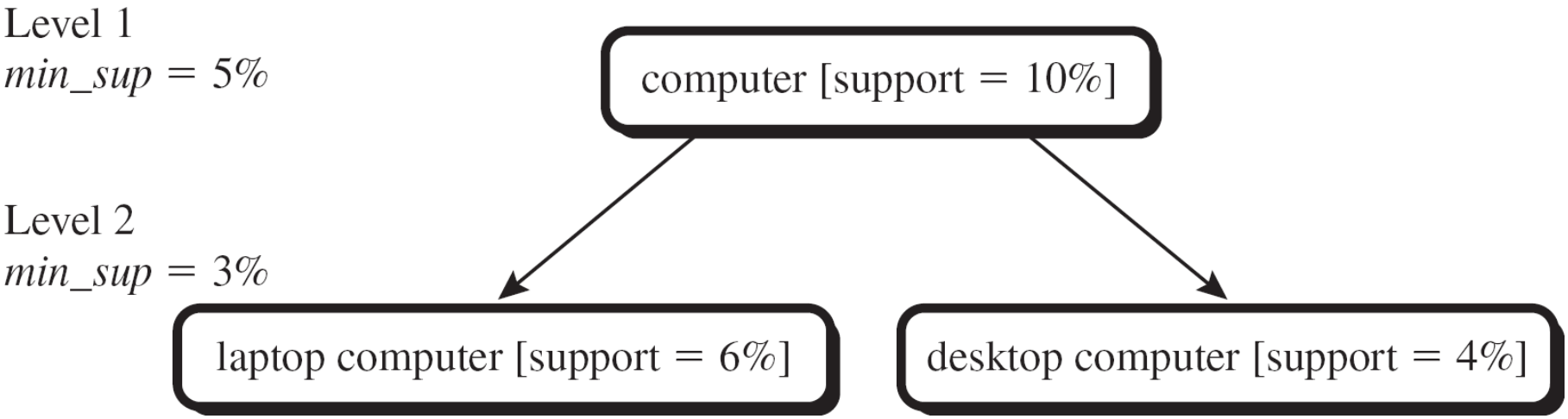


Mining Multilevel Association Rules

- For each level, any algorithm for discovering frequent itemsets may be used, such as Apriori or FP-tree
  - Using uniform minimum support for all levels (referred to as uniform support)
  - Using reduced minimum support at lower levels (referred to as reduced support)
  - Using item or group-based minimum support (referred to as group_based support)



Using uniform minimum support for all levels (referred to as **uniform support**)



Using reduced minimum support at lower levels (referred to as **reduced support**):

Flexible Min-Support Thresholds

- Using item or group-based minimum support (referred to as group-based support):
  - For example, a user could set up the minimum support thresholds based on product price or on items of interest
    - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …

Mining Multilevel Association Rules

- The generation of many redundant rules across multiple levels of abstractions due to the ancestor relationships among items
- A rule is *redundant* if its support is close to the "expected" value, based on the rule's ancestor
  - *buys(X, "laptop computer") ⇒ buys(X, "HP printer")*

    [support = 8%, confidence = 70%]
  - *buys(X, "IBM laptop computer") ⇒ buys(X, "HP printer")*

    [support = 2%, confidence = 70%]

Mining Multidimensional Association Rules

- Single-dimensional rules:

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

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    $age(X,\text{”}19\text{-}25\text{”}) \wedge occupation(X,\text{“}student\text{”}) \Rightarrow buys(X, \text{“}coke\text{”})$
  - hybrid-dimension assoc. rules (*repeated predicates*)

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

Mining Rare Patterns

- In jewelry sales data, sales of diamond watches are rare; however, patterns involving the selling of diamond watches could be interesting.

Mining Negative Patterns

- In supermarket data, if we find that customers frequently buy Coca-Cola Classic or Diet Coke but not both, then buying Coca-Cola Classic and buying Diet Coke together is considered a negative (correlated) pattern