

Chapter Four

Association Rule

Mining

Association Rule Mining

Association rule mining:

- Finding frequent **patterns, associations, correlations, or causal structures** among sets of **items or objects** in transaction databases, relational databases, and other information repositories.
- **Frequent pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs **frequently in a data set**.
- For example:-
 - 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?

Association Rule Mining

- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
- Classical application: **market basket** data analysis, which aims to discover how items are purchased by customers in a supermarket
- E.g., Cheese $\rightarrow$ Wine [**support** = 10%, **confidence** = 80%] meaning that 10% of the customers buy cheese and wine together, and 80% of customers buying cheese also buy wine.

Association Rule Mining

Basic Concepts of Association Rules

- Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items.
- Let $T = \{t_1, t_2, \dots, t_n\}$ be a set of transactions where each transaction t_i is a set of items such that $t_i \subseteq I$.
- An **association rule** is an implication of the form:
- $X \rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$

Association Rule Mining

- Association rule mining **market basket** analysis example
- I – set of all **items** sold in a store
- E.g., $i_1 = \text{Beef}$, $i_2 = \text{Chicken}$, $i_3 = \text{Cheese}$,
- T – set of **transactions**
- The content of a customers basket
- E.g., $t_1: \text{Beef, Chicken, Milk}$; $t_2: \text{Beef, Cheese}$; $t_3: \text{Cheese, Wine}$; $t_4: \dots$
- An **association rule** might be
- $\text{Beef, Chicken} \rightarrow \text{Milk}$, where $\{\text{Beef, Chicken}\}$ is X and $\{\text{Milk}\}$ is Y



Association Rules Mining

Rules can be **weak** or **strong**

- ✓ The strength of a rule is measured by its **support** and **confidence**
- ✓ The **support** of a rule $X \rightarrow Y$, is the percentage of transactions in T that contains X and Y .
- ✓ **Support** shows the frequency of the patterns in the rule; it is the percentage of transactions that contain both **X** and **Y**, i.e. $\text{Support} = \text{Probability}(X \text{ and } Y)$

$$\text{Support} = \frac{\text{\# of transaction involving } X \text{ and } Y}{\text{Total \# of transactions}}$$

Association Rules Mining

- **Confidence** is conditional probability that a transaction having X also contain Y .
- The confidence of a rule $X \rightarrow Y$, is the percentage of transactions in table T containing X , that contain $X \cup Y$
- Let X and Y be two sets of items.
- The confidence $\text{conf}(X \cup Y)$ of the rule $X \cup Y$ is the conditional probability of $X \cup Y$ occurring in a transaction, given that the transaction contains X .
- **Confidence** is number of transactions involving X and Y divided by total number of transactions that have X .

$$\text{Confidence} = \frac{\text{\# of transaction involving } X \text{ and } Y}{\text{Total \# of transaction that have } X}$$

Association Rules Mining

- How do we interpret support and confidence?
- ✓ If support is too **low**, the rule may just occur due to chance
- ✓ Acting on a rule with **low** support may not be profitable since it covers too few cases
- ✓ If confidence is too **low**, we cannot reliably predict Y from X
- ✓ Objective of mining association rules is to **discover all associated rules** in **T** that have support and confidence greater than a **minimum threshold** (minsup, minconf).

Mining Association Rules—An Example

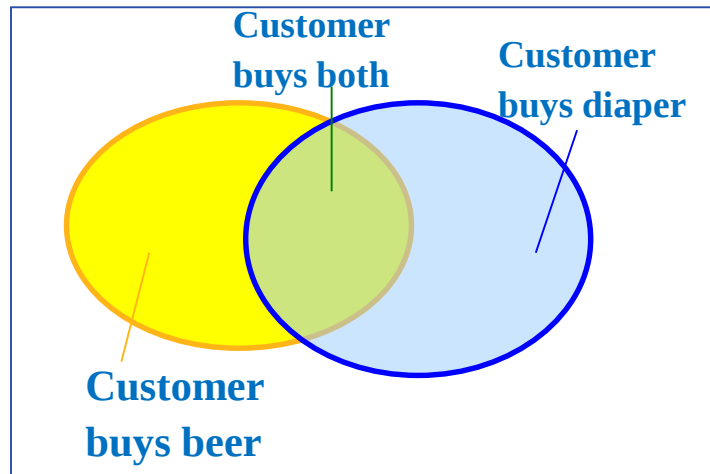
- Finding rules based on support and confidence
- ✓ Let minsup = 30% and minconf = 80%
- ✓ Chicken, Clothes → Milk is valid,
- ✓ [sup = 3/7 (42.84%), conf = 3/3 (100%)]

Transaction ID	Items
T1	Beef, Chicken, Milk
T2	Beef, Cheese
T3	Cheese, Boots
T4	Beef, Chicken, Cheese
T5	Beef, Chicken, Clothes, Cheese, Milk
T6	Clothes, Chicken, Milk
T7	Chicken, Milk, Clothes

- ✓ Clothes → Milk, Chicken is also valid, and there are more...

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 $X \rightarrow Y$ with minimum support and confidence
- Let $minsup = 50\%$, $minconf = 50\%$
- Association rules:
 - $Beer \rightarrow Diaper ?$
 - $Diaper \rightarrow Beer ?$
 - $Beer \rightarrow coffee?$
 - $eggs \rightarrow milk?$

Frequent Itemset Mining Methods

➤ **Apriori Algorithm:** Finding Frequent Itemsets by Confined Candidate Generation

- Problem: the generation of candidate itemsets is expensive

➤ **FPGrowth Algorithm:** A Frequent Pattern-Growth Approach

- ✓ Mining Frequent Patterns without explicit Candidate generation, rather it generates frequent itemsets
- ✓ Uses the Apriori Pruning Principle to generate frequent itemsets

Basic Terminology

- *k-itemset*: a set of k items. E.g.
 - {beer, cheese, eggs} is a 3-itemset
 - {cheese} is a 1-itemset
 - {honey, ice-cream} is a 2-itemset
- *Large itemset*: doesn't mean an itemset with many items.
 - ✓ It means one whose support is at least minimum support.
- L_k : the set of all large k -itemsets in the DB.
- C_k : a set of *candidate* large k -itemsets.

The Apriori Algorithm: Basics

- The Apriori Algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules.
- **Key Concepts :**
 - ✓ **Frequent Itemsets:** The sets of item support denoted by L_i for i^{th} Itemset
 - ✓ **Apriori Property:** Any subset of frequent itemset must be frequent
 - ✓ **Join Operation:** To find L_k , a set of candidate k -itemsets is generated by joining L_{k-1} with itself

Mining Frequent Itemsets

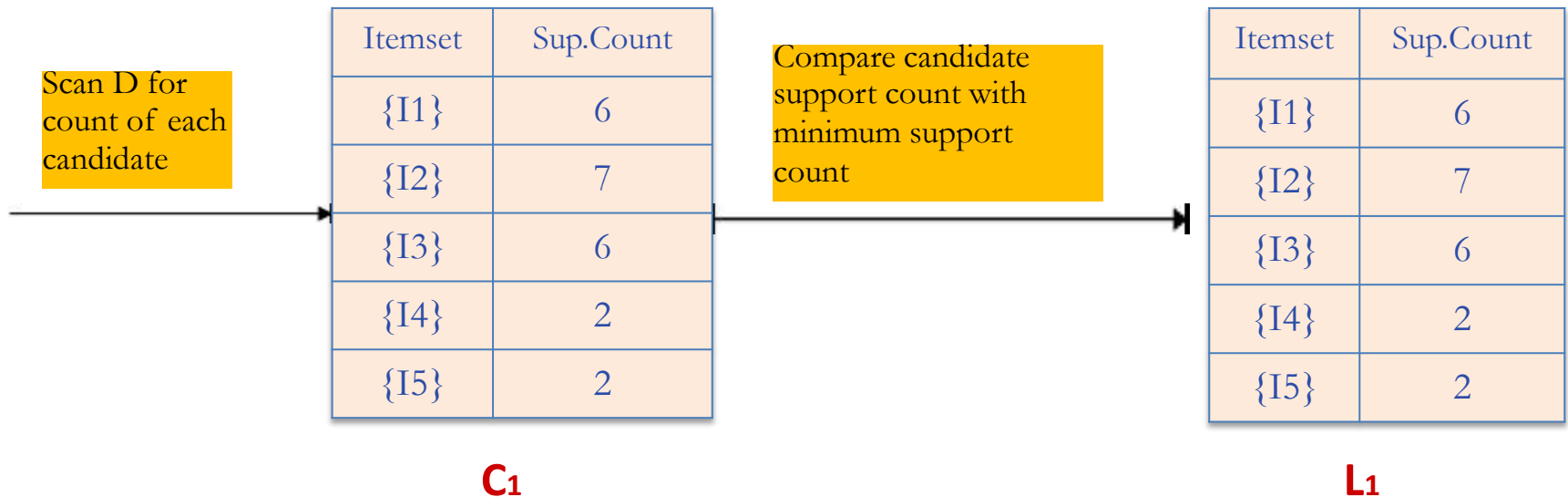
- Find the *frequent itemsets*: the sets of items that have minimum support
 - ✓ A subset of a frequent itemset must also be a frequent itemset
 - i.e., if $\{AB\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$ should be a frequent itemset
 - ✓ Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)
- Use the frequent itemsets to generate association rules.

The Apriori Algorithm: Example

TID	ListofItems
T100	I1,I2,I5
T100	I2,I4
T100	I2,I3
T100	I1,I2,I4
T100	I1,I3
T100	I2,I3
T100	I1,I3
T100	I1,I2,I3,I5
T100	I1,I2,I3

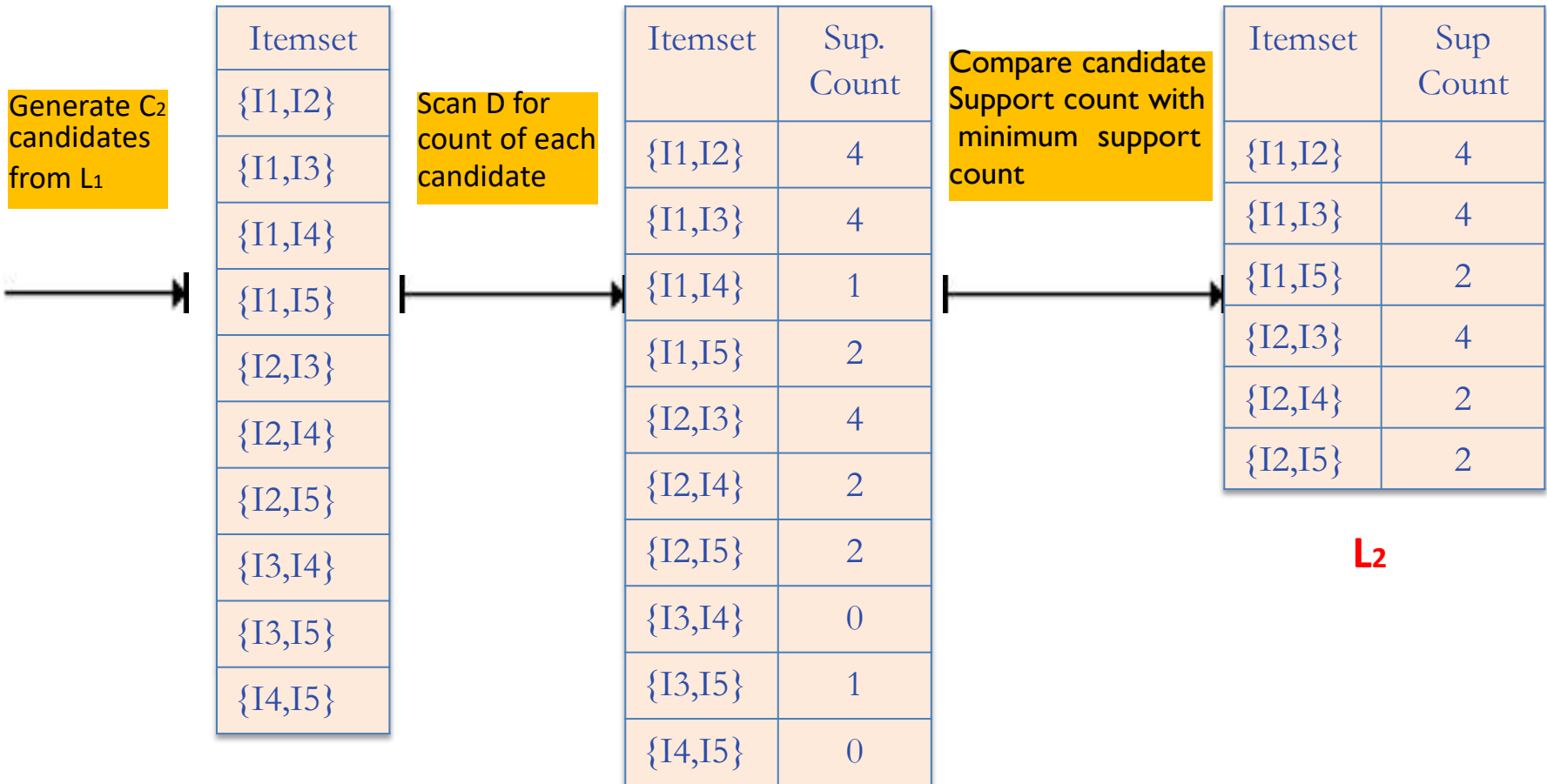
- Consider a database, D ,consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. $\text{min_sup} = 2/9 = 22\%$)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern



- The set of frequent 1-itemsets, **L₁**, consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2: Generating 2-itemset Frequent Pattern



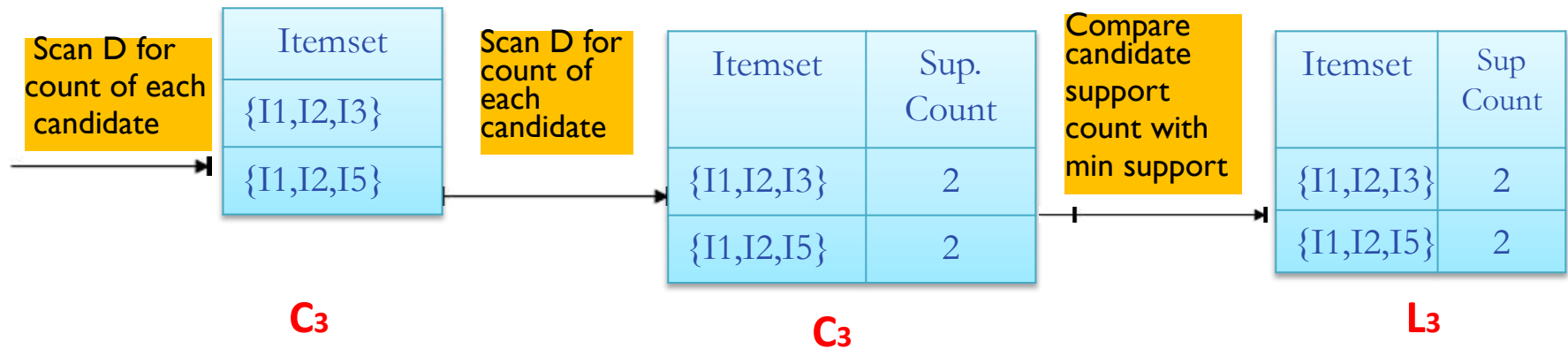
Step 2

C_2

Step 2: Generating 2-itemset Frequent Pattern

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses $L_1 \text{Join } L_1$ to generate a candidate set of 2-itemsets, C_2 .
- Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support
- **Note:** We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern



- The generation of the set of candidate 3-itemsets, C_3 , involves use of the **Apriori Property**
- In order to find C_3 , we compute **$L_2 \text{ Join } L_2$** .
- $C_3 = L_2 \text{ Join } L_2 = \{ \{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\} \}$** .
- Now, **Join step** is complete and **Prune step** will be used to reduce the size of C_3 .
- Prune step** helps to avoid heavy computation due to large C_k

Step 3: Generating 3-itemset Frequent Pattern

- Based on the **Apriori property** that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example , lets take **{I1, I2, I3}**. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, We will keep {I1, I2, I3} in C3.
- Lets take another example of **{I2, I3, I5}** which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L2 and hence it is not frequent **violating**
- **Apriori Property**. Thus We will have to remove {I2, I3, I5} from C3.
- Therefore, $C3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after checking for all members of **result of Join operation** for **Pruning**.
- Now, the transactions in D are scanned in order to determine **L3**, consisting of those candidates 3-itemsets in C3 having minimum support

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses *L3Join* L_3 to generate a candidate set of 4-itemsets, C_4 .
- Although the join results in $\{\{I1, I2, I3, I5\}\}$, this itemset is pruned since its subset $\{\{I2, I3, I5\}\}$ is not frequent.
- Thus, $C_4 = \emptyset$, and algorithm terminates, having found all of the frequent items.
- This completes our Apriori Algorithm.
- What is next?
- These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

- **Procedure:**

- ✓ For each frequent itemset " I ", generate all nonempty subsets of I .
- ✓ We had $L = \{\{I1\}, \{I2\}, \{I3\}, \{I4\}, \{I5\}, \{I1,I2\}, \{I1,I3\}, \{I1,I5\}, \{I2,I3\}, \{I2,I4\}, \{I2,I5\}, \{I1,I2,I3\}, \{I1,I2,I5\}\}$. Lets take $I = \{I1,I2,I5\}$.

- **Back To Example:**

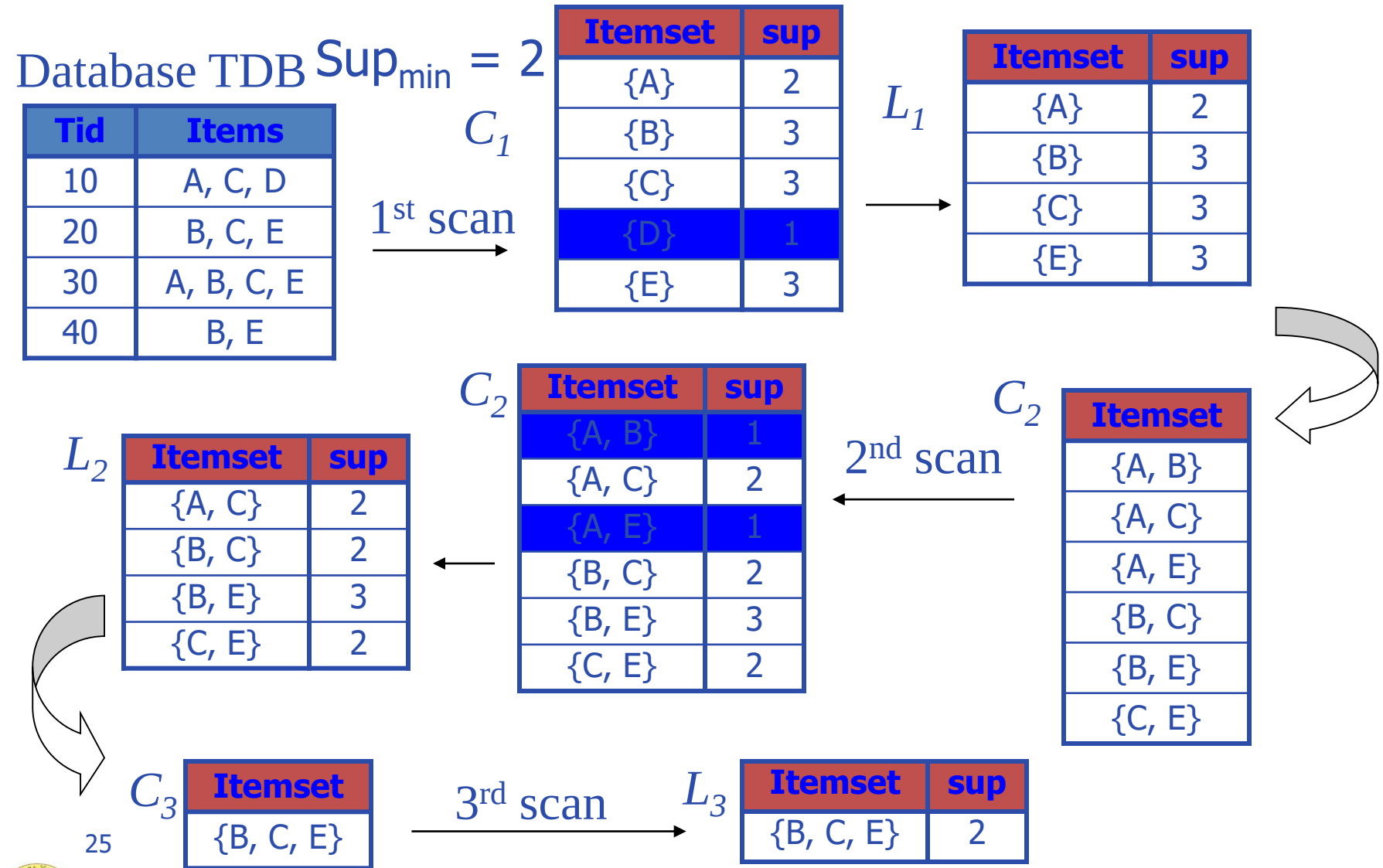
- ✓ Its all nonempty subsets are $\{I1,I2\}, \{I1,I5\}, \{I2,I5\}, \{I1\}, \{I2\}, \{I5\}$.
- ✓ Let **minimum confidence threshold** is , say 70%.
- ✓ The resulting association rules are shown below, each listed with its confidence.
- ✓ $I1 \wedge I2 \rightarrow I5$ Confidence = $\{I1,I2,I5\}/\{I1,I2\} = 2/4 = 50\%$ **R1 is Rejected.**
- ✓ $I1 \wedge I5 \rightarrow I2$ Confidence = $\{I1,I2,I5\}/\{I1,I5\} = 2/2 = 100\%$ **R2 is Selected.**
- ✓ $I2 \wedge I5 \rightarrow I1$ Confidence = $\{I2,I5,I1\}/\{I2,I5\} = 2/2 = 100\%$ **R3 is Selected.**
- ✓ $I1 \rightarrow I2 \wedge I5$ Confidence = $\{I1,I2,I5\}/\{I1\} = 2/6 = 33\%$ **R4 is Rejected.**
- ✓ $I2 \rightarrow I1 \wedge I5$ Confidence = $\{I1,I2,I5\}/\{I2\} = 2/7 = 29\%$ **R5 is Rejected.**
- ✓ $I5 \rightarrow I1 \wedge I2$ Confidence = $\{I1,I2,I5\}/\{I5\} = 2/2 = 100\%$ **R6 is Accepted.**
- ✓ In this way, We have found three strong association rules.

The Apriori Algorithm- Exercise

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

- Generating 1-itemset ?
- Generating 2-itemset?
- Generating 3-itemset ?

The Apriori Algorithm—An Example



Methods to Improve Apriori's Efficiency

- **Hash-based itemset counting:** A k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- **Transaction reduction:** A transaction that does not contain any frequent k -itemset is useless in subsequent scans.
- **Partitioning:** Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
- **Sampling:** mining on a subset of given data, lower support threshold a method to determine the completeness.
- **Dynamic itemset counting:** add new candidate itemsets only when all of their subsets are estimated to be frequent.

Bottlenecks of the Apriori approach

- The Apriori algorithm reduces the size of candidate frequent itemsets by using “Apriori property.”
 - However, it still requires two nontrivial computationally expensive processes.
- It requires as many database scans as the size of the largest frequent itemsets. In order to find frequent k -itemsets, *the Apriori algorithm needs to scan database k times.*
 - Breadth-first (i.e., level-wise) search
 - Candidate generation and test the frequency of true appearance of the itemsets
 - It may generate a huge number of candidate sets that will be discarded later in the test stage.

Mining Frequent Patterns Without Candidate Generation

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

FP-Growth Method : An Example

Items Bought
F,A,C,D,G,I,M,P
A,B,C,F,L,M,O
B,F,H,J,O
B,C,K,S,P
A,F,C,E,L,P,M,N

- Consider a database, D , consisting of 5 transactions.
- Suppose min. support count required is 3 (i.e. $\text{min_sup} = 3/5 = 60\%$)
- The first scan of database is same as Apriori, which derives the set of 1-itemsets & their support counts.
- The set of frequent items is sorted in the order of descending support count.
- The resulting set is denoted as
- $L = \{F:4, C:4, A:3, B:3, M:3, P:3\}$

FP-Growth Method: Construction of FP-Tree

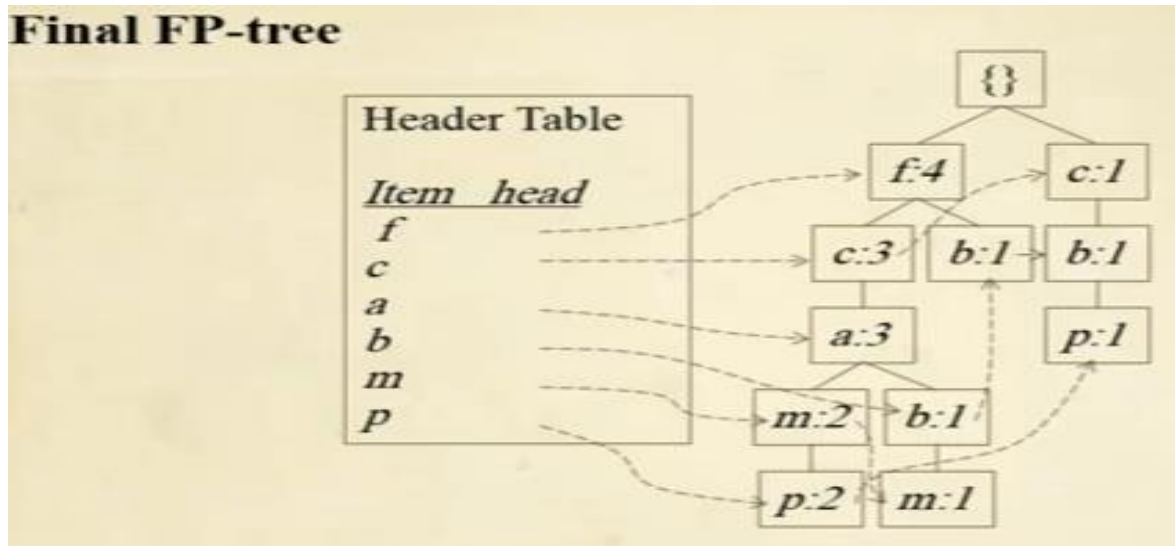
- First, create the **root of the tree**, labeled with “**null**”.
- Scan the database D a second time. (First time we scanned it to create **1-itemset** and then L).
- The items in each transaction are processed in **L** order (i.e. sorted order).
- A branch is created for each transaction with **items** having their **support count** separated by colon.
- Whenever the **same node** is encountered in another transaction, we just increment the **support count** of the common node or Prefix.
- To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of **node-links**. Now, The problem of mining frequent patterns in database is transformed to that of mining the FP-Tree.

Construct FP-tree from a Transaction Database

- Assume min-support = 3 and min-confidence = 80%

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
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}

Final FP-tree



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

Mining the FP-Tree by Creating Conditional (sub)pattern bases

- Steps:

1. Start from each frequent **length-1 pattern** (as an initial suffix pattern).
2. Construct its **conditional pattern base** which consists of the set of prefix paths in the FP-Tree **co-occurring** with suffix pattern.
3. Then, Construct its conditional FP-Tree & perform mining on such a tree.
4. The **pattern growth** is achieved by concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-Tree.
5. The union of all frequent patterns (generated by step 4) gives the **required frequent itemset**.

Conditional Pattern Bases & Conditional FP-Tree

Item	Conditional Pattern base	Conditional FP-Tree	Frequent Pattern Generated
p	{(fcam:2), (cb:1)}	{c:3}	<p,c:3>
m	{(fca:2), (fcab:1)}	{fca:3}	<mf:3><mc:3><ma:3><mfca:3>
b	{(fca:1), (f:1), (c:1)}	empty	empty
a	{(fc:3)}	{fc:3}	<af:3><ac:3><afc:3>
c	{(f:3)}	{f:3}	<cf:3>
f	empty	empty	empty

- ✓ Check the validity of the following rule with confidence level at least 80% and 60% support ?
- ✓ m $\rightarrow$ fc ?
 - ✓ m $\rightarrow$ ca ?
 - ✓ m $\rightarrow$ fca?

Rule Generated From FP-Growth Tree

Rule	Support	Confidence
$c \rightarrow p$	3(60%)	75%
$fca \rightarrow m$	3(60%)	100%
$ca \rightarrow m$	3(60%)	100%
$f \rightarrow a$	3(60%)	75%
$c \rightarrow a$	3(60%)	75%
$f \rightarrow c$	3(60%)	100%
$c \rightarrow f$	3(60%)	?

- According to the given support and confidence rule
 $fca \rightarrow m$,
 $ca \rightarrow m$, $f \rightarrow c$ are valid.

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 information: 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)

Why Is Frequent Pattern Growth Fast?

- Our performance study shows
 - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
 - No candidate generation, no candidate test
 - Use compact data structure
 - Eliminate repeated database scan
 - Basic operation is counting and FP-tree building

Mining Multilevel Association Rules From Transactional Databases

- Why Multiple-Level Association Rules?
- Sometimes, at primitive data level, data does not show any significant pattern. But there are useful information hiding behind.
- The goal of Multiple-Level Association Analysis is to find the hidden information in or between levels of abstraction
- Requirements in Multiple-Level Association Analysis

Two general requirements to do multiple-level association rule mining:

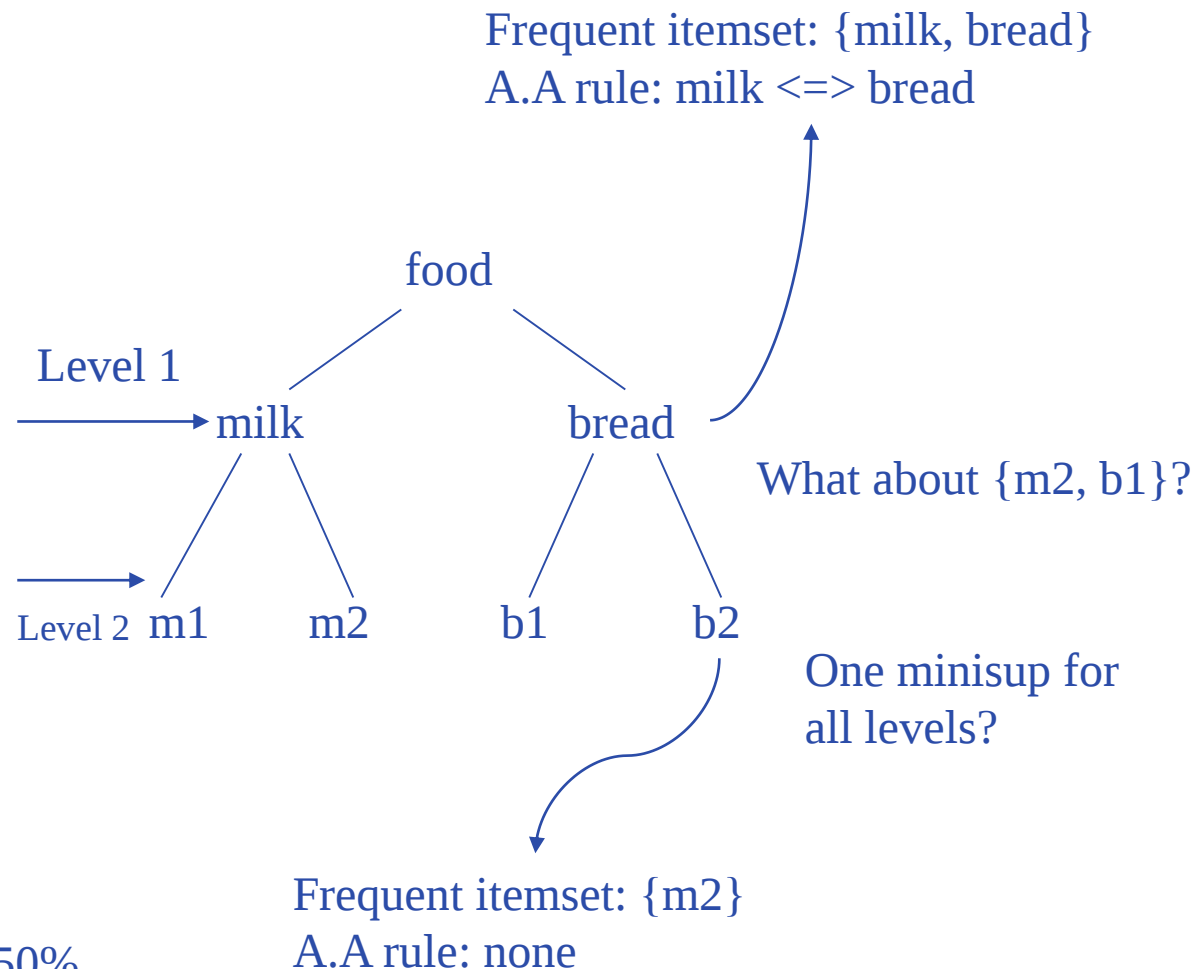
- 1) Provide data at multiple levels of abstraction. (a common practice now)
- 2) Find efficient methods for multiple-level rule mining. (our focus)

Multilevel Association Rule Example

TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}
T4	{milk, bread}
T5	{milk}

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}
T4	{m2, b1}
T5	{m2}

minisup = 50% miniconf = 50%



Multi-Dimensional Association: Concepts

- Single-dimensional rules:
 $\text{buys}(X, \text{"milk"}) \Rightarrow \text{buys}(X, \text{"bread"})$
- Multi-dimensional rules: ○ 2 dimensions or predicates
 - Inter-dimension association rules (*no repeated predicates*)
 $\text{age}(X, \text{"19-25"}) \wedge \text{occupation}(X, \text{"student"}) \Rightarrow \text{buys}(X, \text{"coke"})$
 - hybrid-dimension association 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, also called **nominal**.
- Quantitative Attributes
 - numeric, implicit ordering among values

Techniques for Mining MD Associations

- Search for frequent k -predicate set:
 - Example: {age, occupation, buys} is a 3-predicate set.
 - Techniques can be categorized by how age are treated.
- 1. Using static discretization of quantitative attributes
 - Quantitative attributes are statically discretized by using predefined concept hierarchies.
- 2. Quantitative association rules
 - Quantitative attributes are dynamically discretized into “bins” based on the distribution of the data.
- 3. Distance-based association rules
 - This is a dynamic discretization process that considers the distance between data points.