ADITYA
ENGINEERING COLLEGE (A)

UNIT IV

Association

ADITYA ENGINEERING COLLEGE (A)

Contents

- Problem Definition
- Frequent Item Set generation
- Rule Generation
- Compact Representation of frequent item sets
- FP-Growth Algorithm.

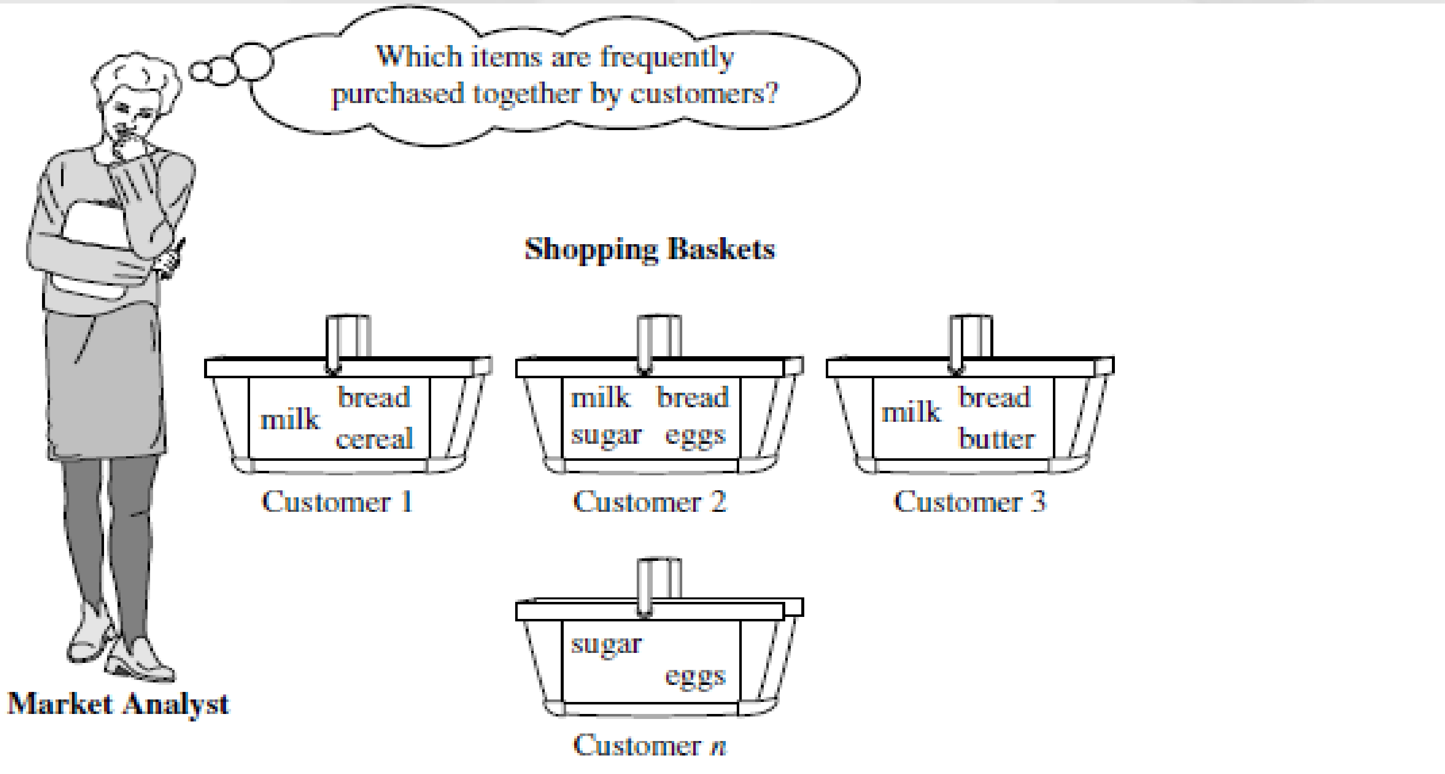
Association: Problem Definition

Market Basket Analysis: A Motivating Example:

This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.

The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.

Association: Problem Definition



Market basket analysis.

ADITYA ENGINEERING COLLEGE (A)

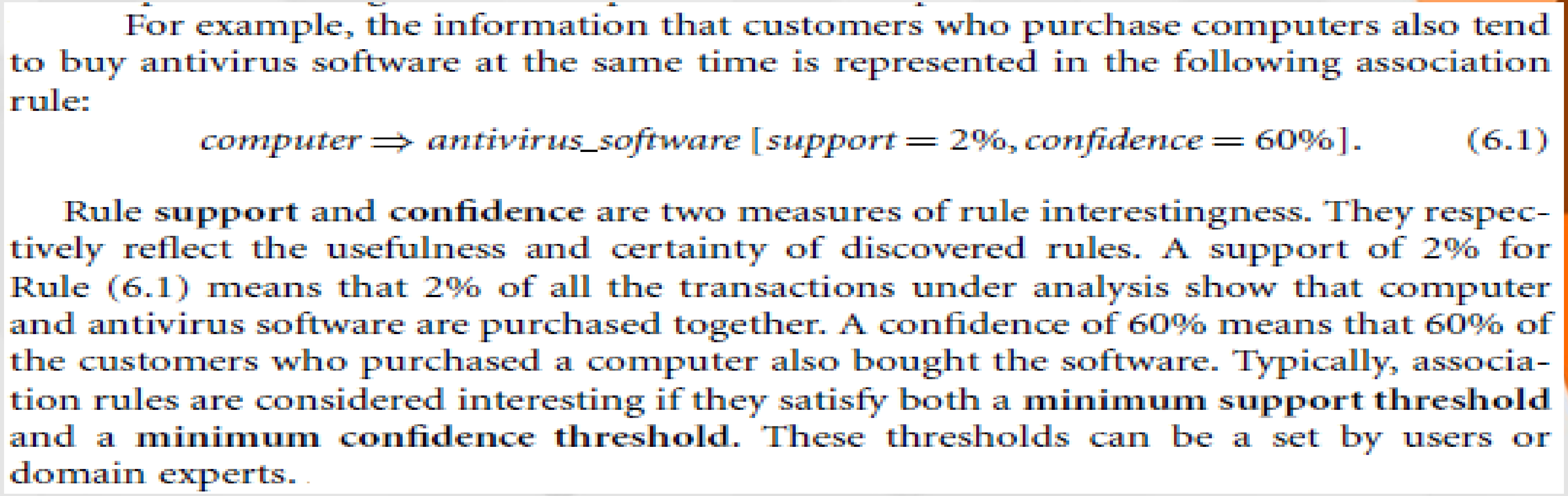
Example

- For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

Example

Suppose, as a marketing manager of *AllElectronics*, *you would like to* determine which items are frequently purchased together within the same transactions.

An example of such a rule, mined from the *AllElectronics transactional database.*



For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in the following association rule:

$$computer \Rightarrow antivirus\_software\ [support = 2\%, confidence = 60\%]. \qquad (6.1)$$

Rule **support** and **confidence** are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for Rule (6.1) means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**. These thresholds can be a set by users or domain experts.

ADITYA ENGINEERING COLLEGE (A)

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Sugar, Flour, Eggs
3	Milk, Sugar, Flour, Coke
4	Bread, Milk, Sugar, Flour
5	Bread, Milk, Sugar, Coke

Example of Association Rules

{Sugar} $\rightarrow$ {Flour},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Flour, Bread} $\rightarrow$ {Milk},

Implication means co-occurrence, not causality!

ADITYA ENGINEERING COLLEGE (A)

Definition: Frequent Itemset

TID	Items
1	Bread, Milk
2	Bread, Sugar, Flour, Eggs
3	Milk, Sugar, Flour, Coke
4	Bread, Milk, Sugar, Flour
5	Bread, Milk, Sugar, Coke

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Sugar}
  - k-itemset
    - An itemset that contains k items

    E.g., {Milk, Bread, Sugar} – 3 itemset

    The null set is an itemset that doesnot contain any items.
- **Support count (σ)**
  - Frequency of occurrence of an itemset(No. of transactions that contain a particular itemset.)
  - E.g. σ({Milk, Bread, Sugar}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Sugar}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

ADITYA ENGINEERING COLLEGE (A)

TID	*Items*
1	Bread, Milk
2	Bread, Sugar, Flour, Eggs
3	Milk, Sugar, Flour, Coke
4	Bread, Milk, Sugar, Flour
5	Bread, Milk, Sugar, Coke

- **Association Rule:** refer to the probability of customer purchasing one product when he purchases some other product.
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Sugar} → {Flour}
- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y

    Support, s (X → Y) = $\frac{\sigma(X \cup Y)}{N}$

    Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

    Confidence, c (X → Y) = $\frac{\sigma(X \cup Y)}{\sigma(X)}$

Example:

$$\{\text{Milk}, \text{Sugar}\} \Rightarrow \{\text{Flour}\}$$

$$s = \frac{\sigma(\text{Milk}, \text{Sugar}, \text{Flour})}{N} = \frac{2}{5} = 0.4$$

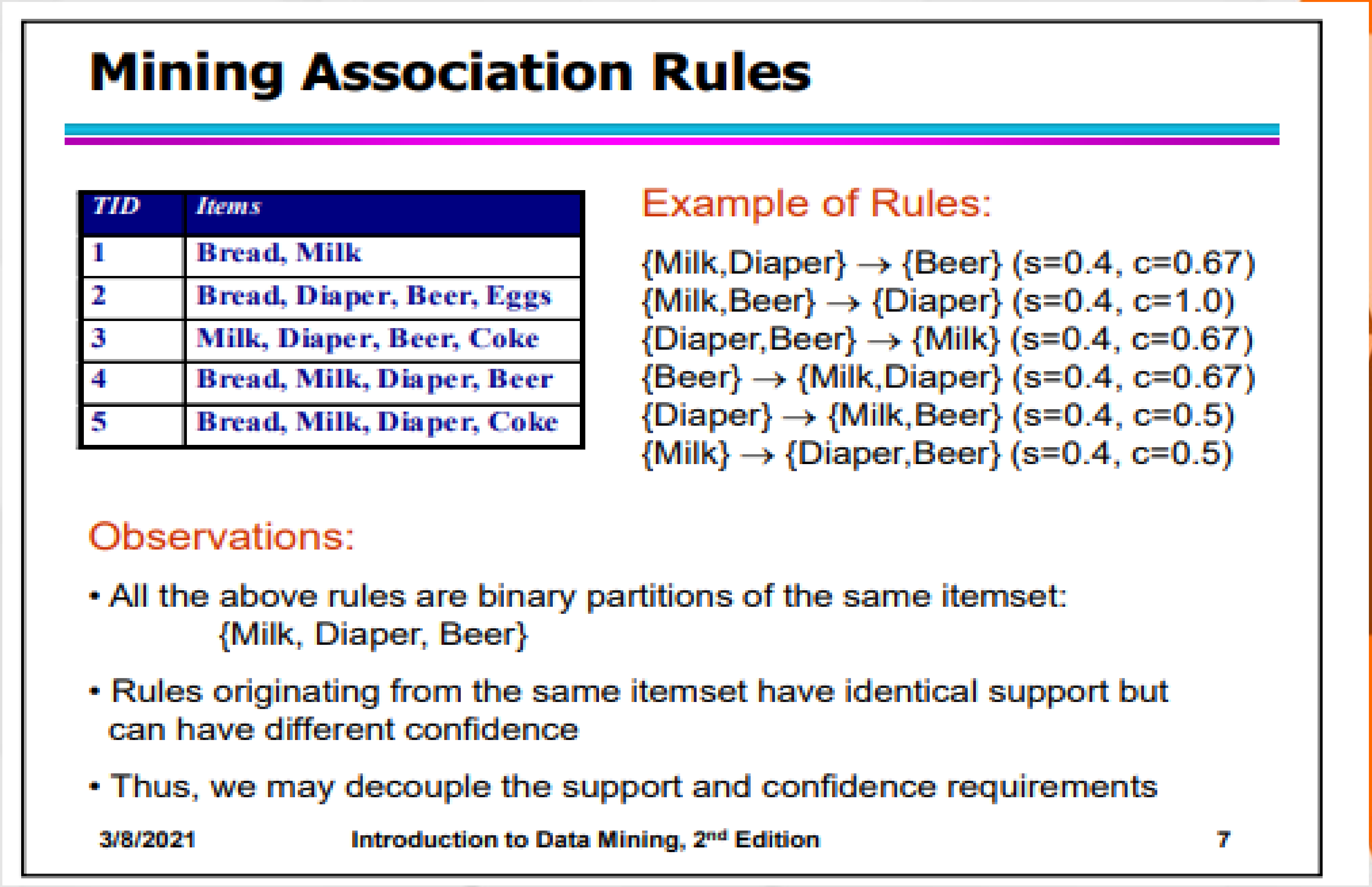
$$c = \frac{\sigma(\text{Milk}, \text{Sugar}, \text{Flour})}{\sigma(\text{Milk}, \text{Sugar})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  $\Rightarrow$ Computationally prohibitive!



Mining Association Rules

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Association Rule Mining

- The problem of mining association rules can be reduced to that of mining frequent itemsets.

- In general, association rule mining can be viewed as 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, **minsup**.
     – Generate all itemsets whose **support ≥ minsup**
  2. **Generate strong association rules from the frequent itemsets**: By definition, these rules must satisfy minimum support and minimum confidence.
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

– Frequent itemset generation is still computationally expensive

Association Rules - Example

Transactions
A,B,D
A,B,C,D
A
A,B,C
B,C
B

minsup = 0.5 minconf=0.7

- Find frequent itemsets and association rules satisfying minsup and minconf.

Frequent Itemsets:

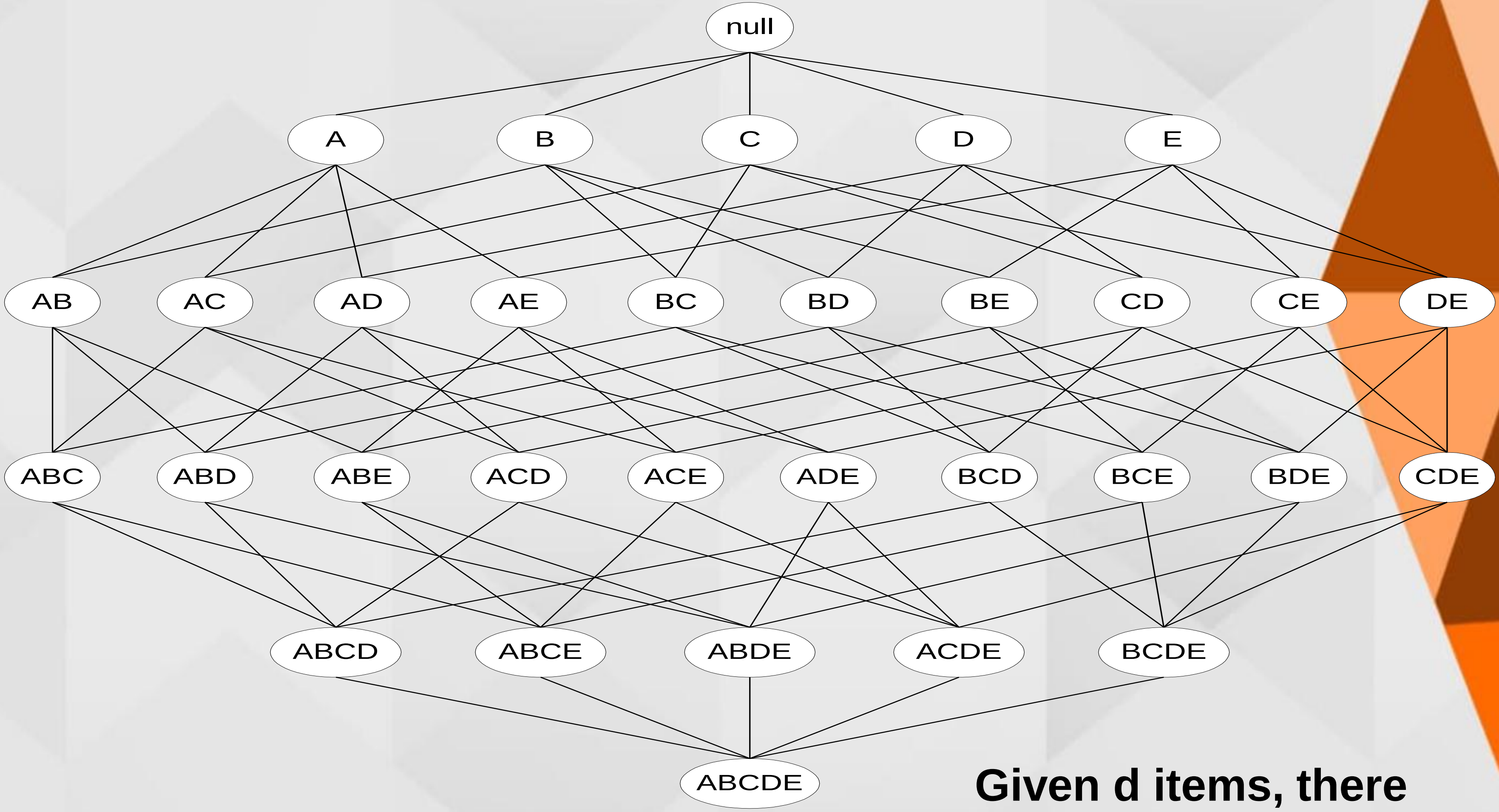
1-itemsets:	**{A}**	**support({A}) = 4/6**
	{B}	**support({B}) = 5/6**
	{C}	**support({C}) = 3/6**
2-itemsets:	**{A,B}**	**support({A,B}) = 3/6**
	{B,C}	**support({B,C}) = 3/6**

Association Rules:

$A \rightarrow B$	**conf($A \rightarrow B$) = 3/4**
$C \rightarrow B$	**conf($C \rightarrow B$) = 3/3**

ADITYA ENGINEERING COLLEGE (A)

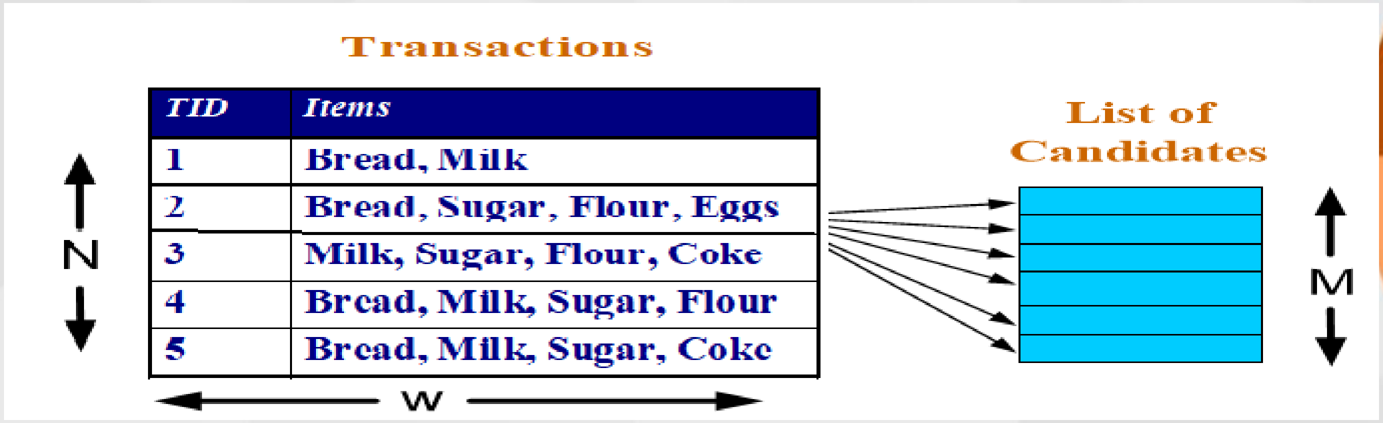
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

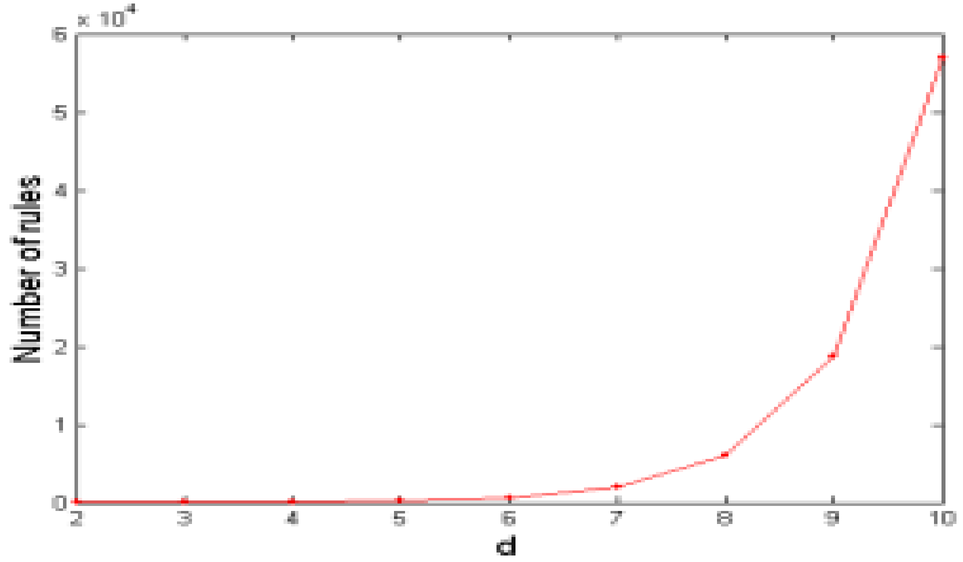


TID	Items
1	Bread, Milk
2	Bread, Sugar, Flour, Eggs
3	Milk, Sugar, Flour, Coke
4	Bread, Milk, Sugar, Flour
5	Bread, Milk, Sugar, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!
  - N- No. of transactions
  - M-No. of candidate itemsets
  - w- Max. transaction width

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:

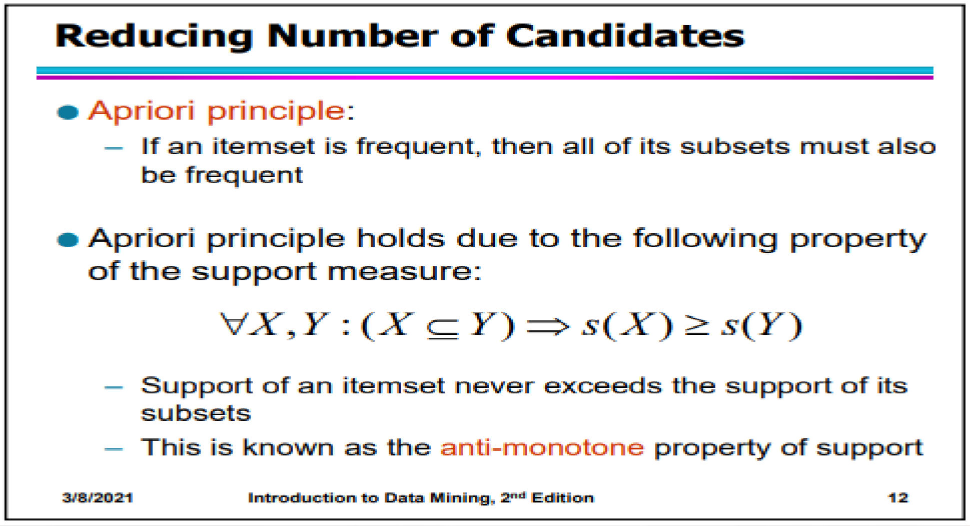


$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

To reduce the computational complexity of frequent itemset generation:

- Reduce the number of candidates (M)--- Apriori principle
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction



Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

3/8/2021 Introduction to Data Mining, 2nd Edition 12

Mining Single Dimensional, Boolean Association rules fro transactional data bases

Methods:

- Apriori Algorithm
- FP growth algorithm

Apriori Algorithm

Apriori Algorithm: Finding Frequent Itemsets using Candidate Generation

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - **Candidate Generation:** Generate length (k+1) candidate itemsets from length k frequent itemsets
    - **Candidate Pruning:** Prune candidate itemsets containing subsets of length k that are infrequent
    - **Support Counting:** Count the support of each candidate by scanning the DB
    - **Candidate Elimination:** Eliminate candidates that are infrequent, leaving only those that are frequent

Apriori principle:

If an itemset is frequent, then all of its subsets must also be frequent

Working of Apriori Property:

- A two-step process is followed, consisting of **join** and **prune** actions.
- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

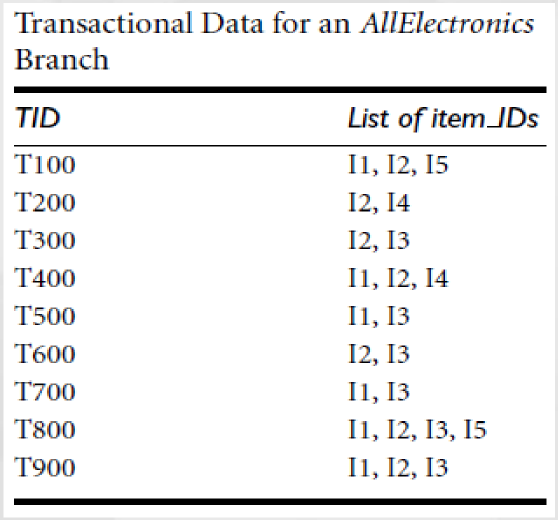
```
insert into $C_k$
select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}$ p, $L_{k-1}$ q
where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
```

- Step 2: pruning

```
forall itemsets c in $C_k$ do
    forall (k-1)-subsets s of c do
        if (s is not in $L_{k-1}$) then delete c from $C_k$
```

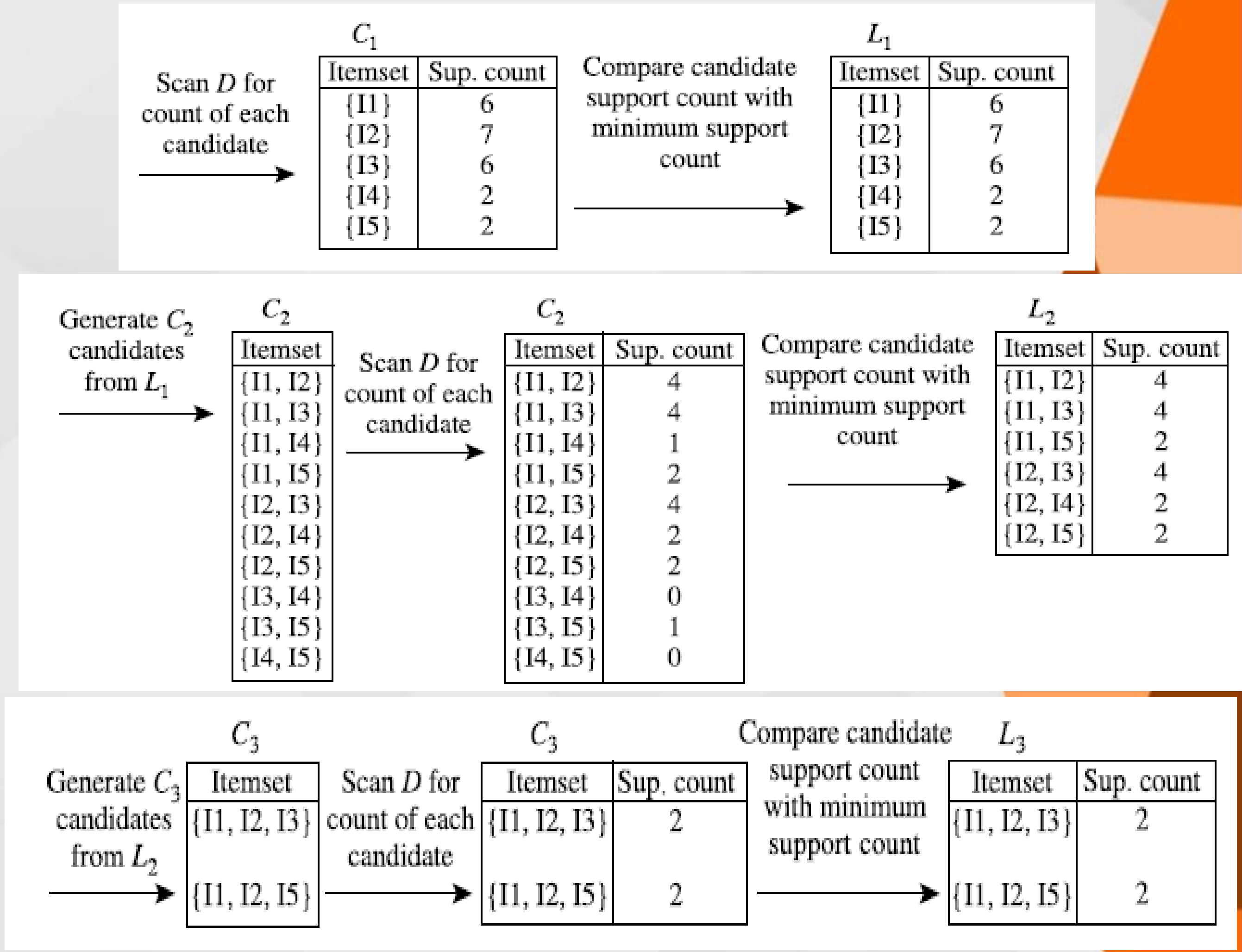
ADITYA ENGINEERING COLLEGE (A)

Example



Transactional Data for an *AllElectronics* Branch

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



Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_2 candidates from L_1
Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_3 candidates from L_2
Scan D for count of each candidate
Compare candidate support count with minimum support count

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

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

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, itemset {I1, I2, I3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

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

The Apriori Algorithm: Example

TID	List of Items
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. 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

Scan D for count of each candidate →

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count →

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

- In the first iteration of the algorithm, each item is a member of the set of candidate.
- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.

Step 2: Generating 2-itemset Frequent Pattern

Generate C_2 candidates from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

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

C_2

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

To discover the set of frequent 2-itemsets, L_2 , the algorithm uses $L_1 \bowtie L_1$ 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

Scan D for count of each candidate

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

C_3

Compare candidate support count with min support count

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

- 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 $L_2 \bowtie L_2$.

Step 3: Generating 3-itemset Frequent Pattern [Cont.]

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

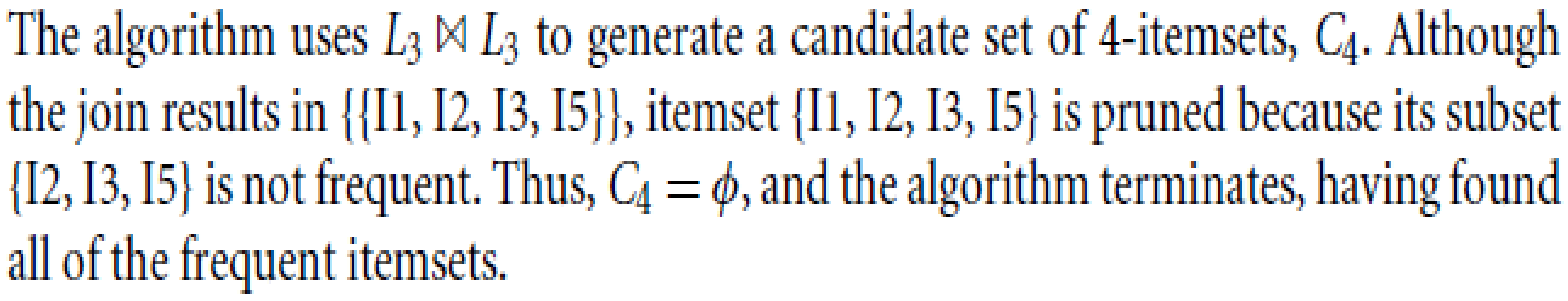
(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- 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.

Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

Step 4: Generating 4-itemset Frequent Pattern



The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, itemset {I1, I2, I3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

Step 5: Generating Association Rules from Frequent Itemsets

- Procedure:
  - For each frequent itemset "*l*", generate all nonempty subsets of ***l.***
  - For every nonempty subset *s* of *l*, output the rule **"s → (l-s)"** if **support_count(l) / support_count(s) >= min_conf** where min_conf is minimum confidence threshold.

- Back To Example:

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 *l* = {I1,I2,I5}.
  - Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

- Let minimum confidence threshold is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.
  - R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - R3: I2 ^ I5 → I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

- R4: I1 → I2 ^ I5
    - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
    - R4 is Rejected.
- R5: I2 → I1 ^ I5
    - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
    - R5 is Rejected.
- R6: I5 → I1 ^ I2
    - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
    - R6 is Selected.

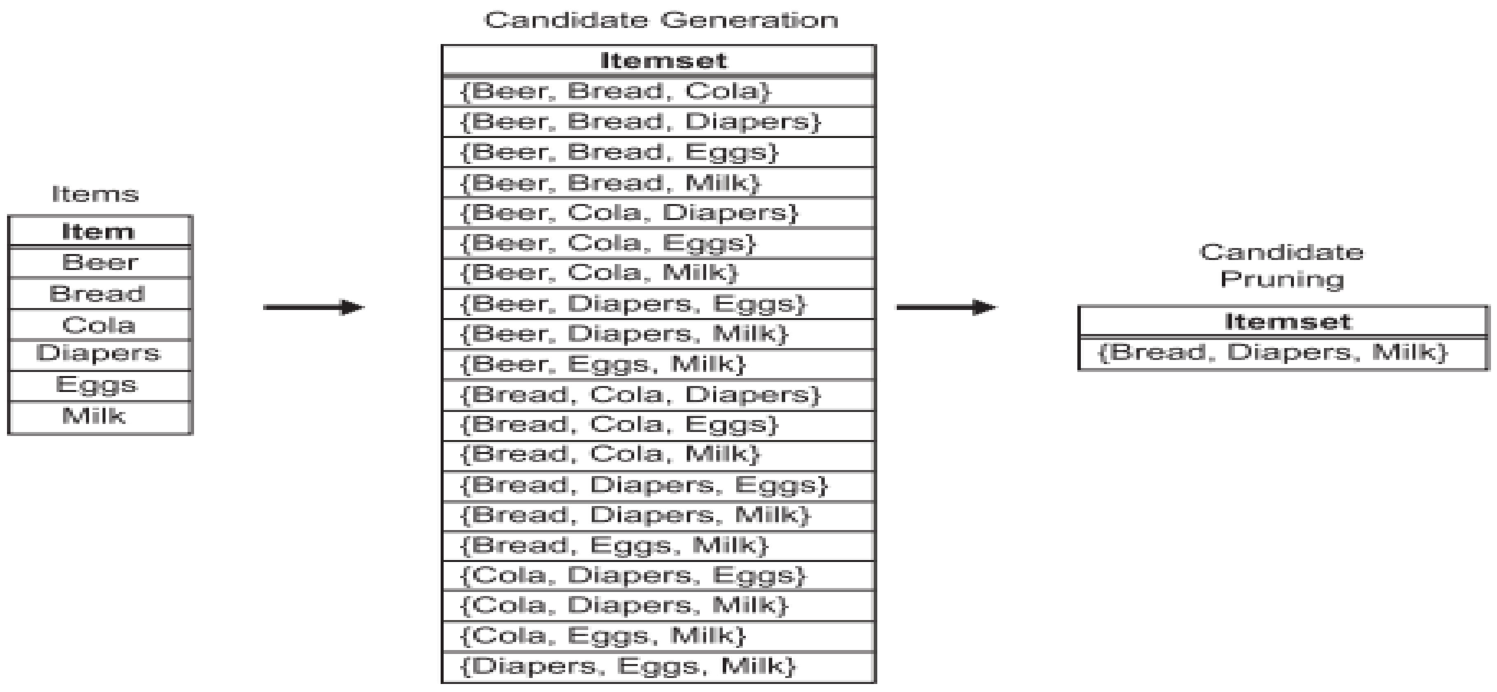
    In this way, We have found three strong association rules.

Candidate Generation Procedures

- **Brute-force method**
- **$F_{k-1} \times F_1$ Method**
- **$F_{k-1} \times F_{k-1}$ Method**

Candidate Generation: Brute-force method

The brute-force method considers every k-itemset as a potential candidate and then applies the candidate pruning step to remove any unnecessary candidates whose subsets are infrequent



Items

Item
Beer
Bread
Cola
Diapers
Eggs
Milk

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Eggs}
{Beer, Bread, Milk}
{Beer, Cola, Diapers}
{Beer, Cola, Eggs}
{Beer, Cola, Milk}
{Beer, Diapers, Eggs}
{Beer, Diapers, Milk}
{Beer, Eggs, Milk}
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Milk}
{Bread, Eggs, Milk}
{Cola, Diapers, Eggs}
{Cola, Diapers, Milk}
{Cola, Eggs, Milk}
{Diapers, Eggs, Milk}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Figure 5.6. A brute-force method for generating candidate 3-itemsets.

Candidate Generation: $F_{k-1} \times F_1$

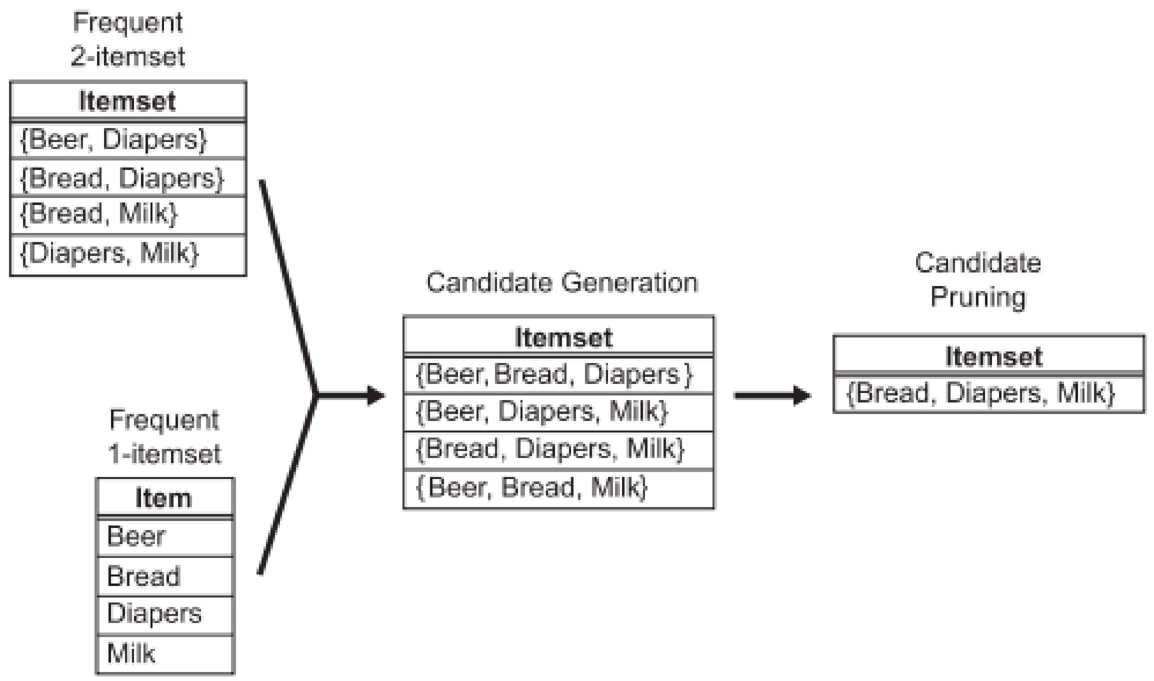


Figure 5.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Apriori Algorithm:
Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE

  - Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

- L_4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets **generated**

Apriori Algorithm:
Candidate Pruning

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- L_4 = {ABCD,ABCE,ABDE} is the set of **candidate 4-itemsets generated**

- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent

- After **candidate pruning**: L_4 = {ABCD}

Candidate Generation: Fk-1 x Fk-1 Method

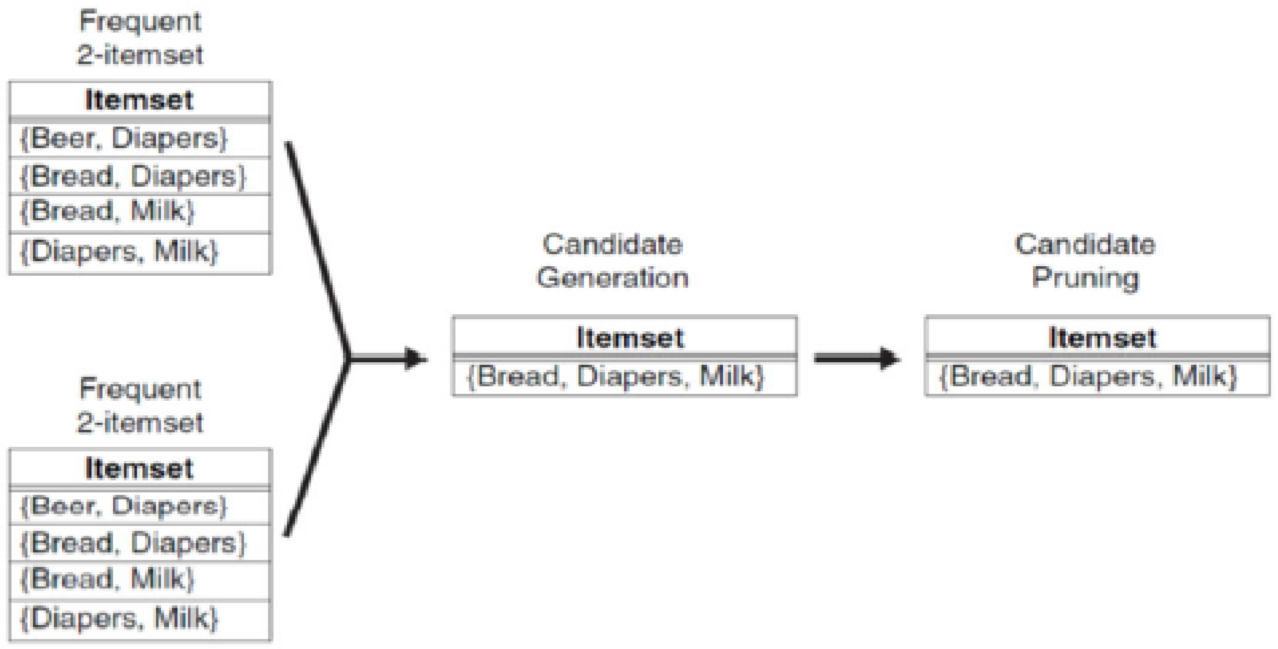


Figure 5.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

Rule Generation in Apriori Algorithm

- Given a frequent itemset **L**, find all non-empty subsets $\mathbf{f} \subset \mathbf{L}$ such that **candidate rule** $\mathbf{f} \rightarrow \mathbf{L} - \mathbf{f}$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

$ABC \rightarrow D$	$ABD \rightarrow C$	$ACD \rightarrow B$	$BCD \rightarrow A$
$D \rightarrow ABC$	$C \rightarrow ABD$	$B \rightarrow ACD$	$A \rightarrow BCD$,
$AB \rightarrow CD$	$AC \rightarrow BD$	$AD \rightarrow BC$	
$CD \rightarrow AB$	$BD \rightarrow AC$	$BC \rightarrow AD$	

- If $|L| = k$, then there are $2^k - 2$ candidate association rules
  - (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation in Apriori Algorithm

- How to efficiently generate rules from frequent itemsets?

- In general, confidence does not have an anti-monotone property

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

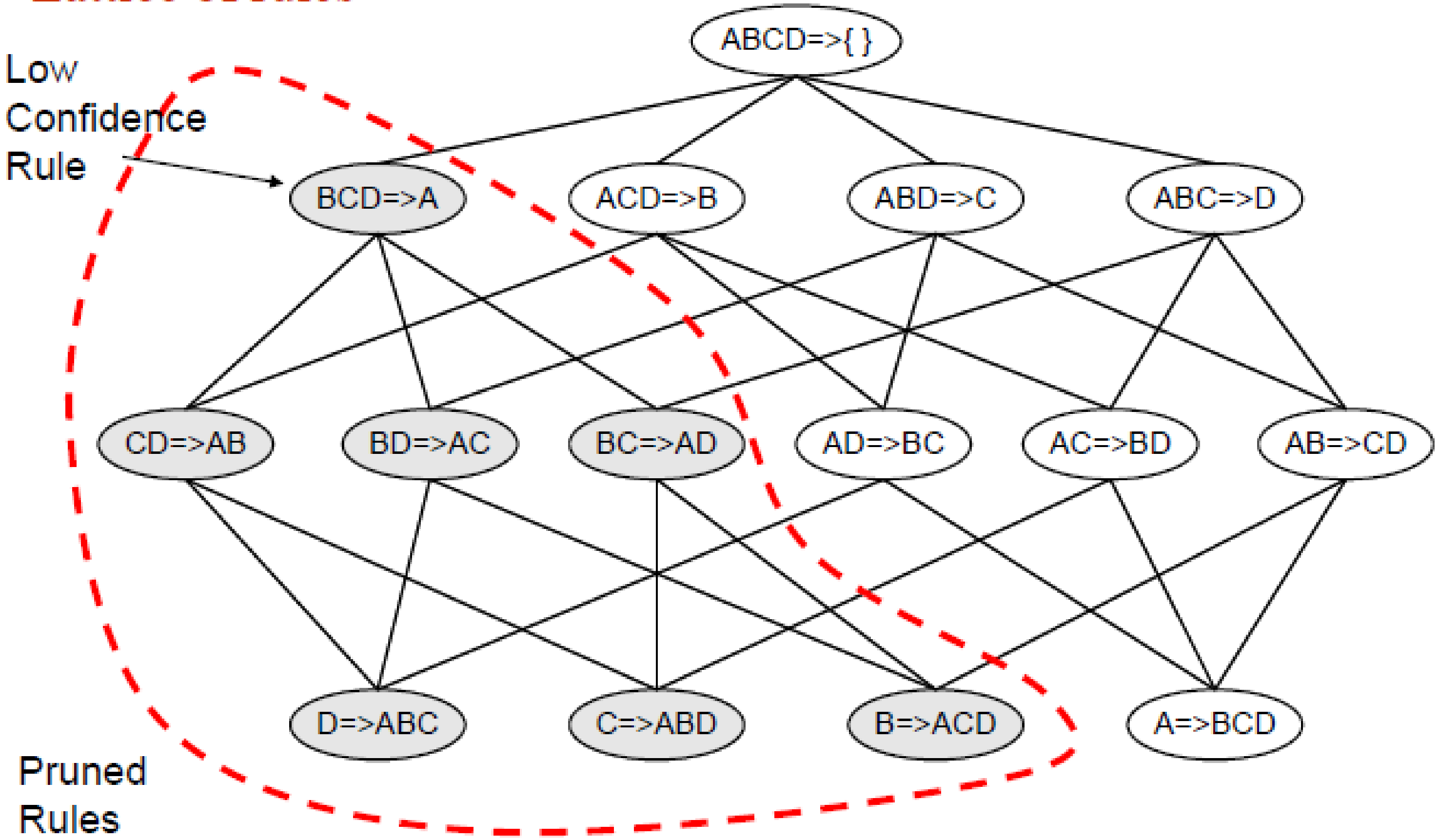
- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:

    $$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

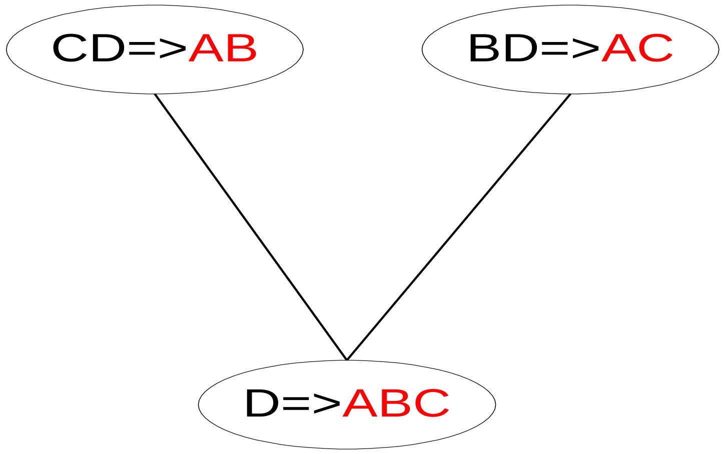
Rule Generation in Apriori Algorithm

Lattice of rules



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC
- Prune rule D=>ABC if its subset AD=>BC does not have high confidence



Compact Representation of Frequent Itemset

- What happens when you have a large market basket data with over a hundred items?
- The number of frequent itemsets grows exponentially and this in turn creates an issue with storage and it is for this purpose that alternative representations have been derived which reduce the initial set but can be used to generate all other frequent itemsets.
- The Maximal and Closed Frequent Itemsets are two such representations that are subsets of the larger frequent itemset.

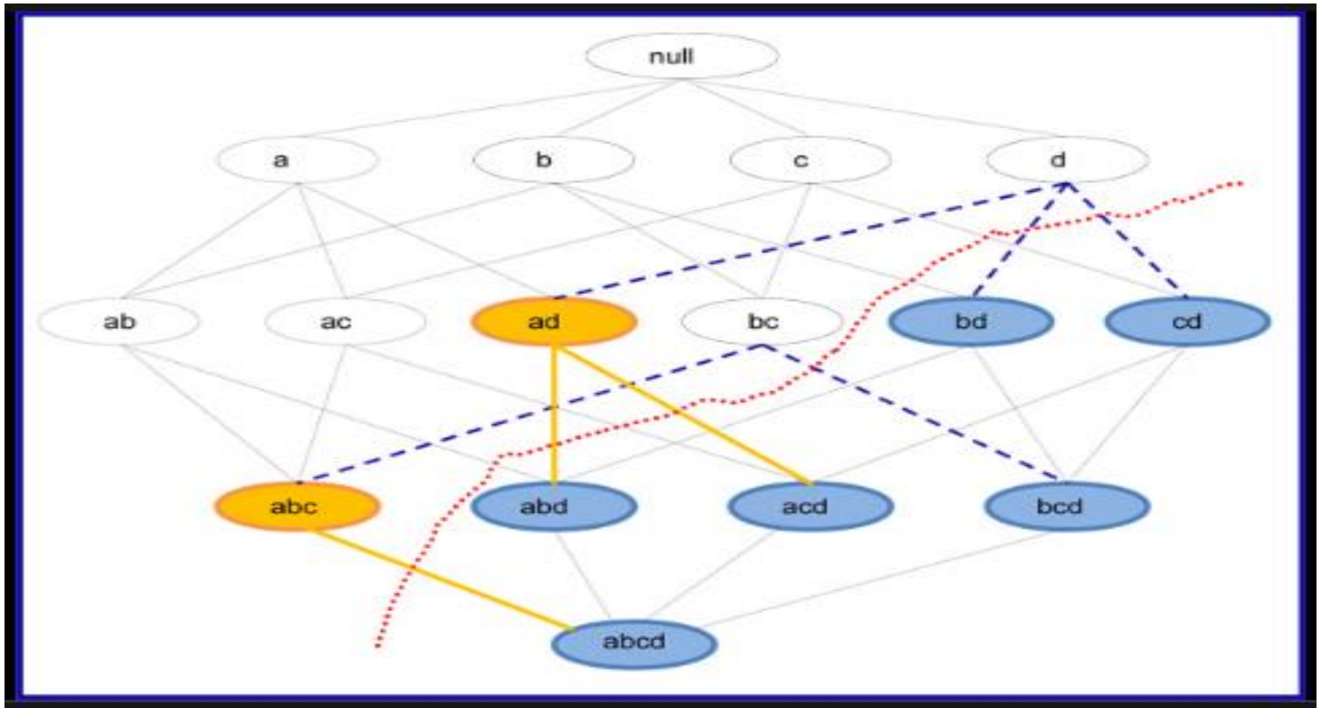
Maximal Frequent Itemset

Definition

- It is a frequent itemset for which none of its immediate supersets are frequent.

Identification

- Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets.
- Identify all of its immediate supersets.
- If none of the immediate supersets are frequent, the itemset is maximal frequent.



Closed Frequent Itemset

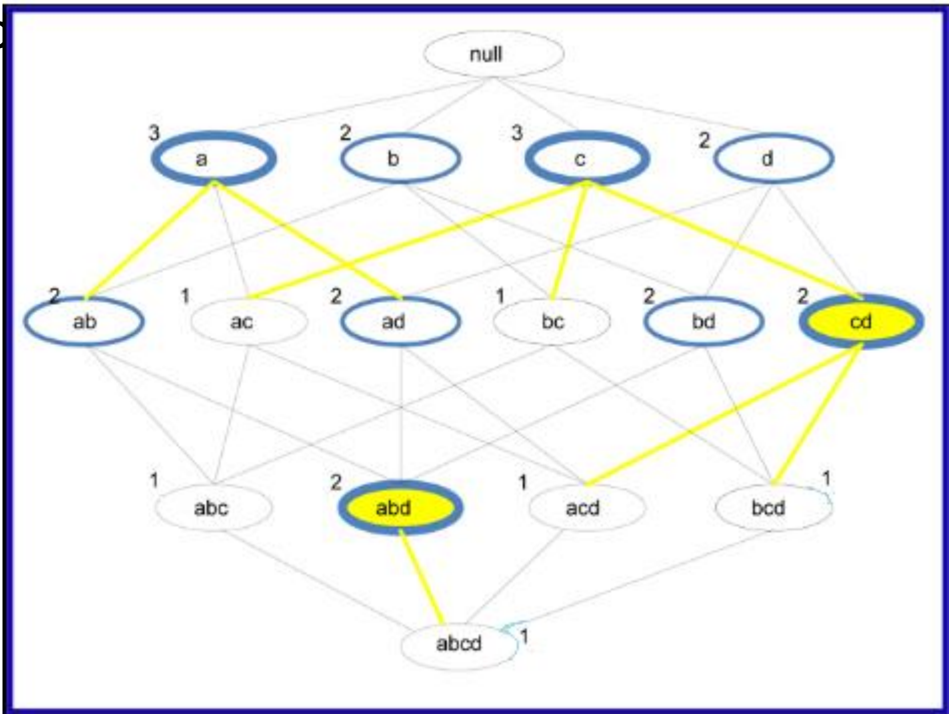
Definition:

- An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset
- It is a frequent itemset that is both closed and its support is greater than or equal to minsup..

Identification

- First identify all frequent itemsets.
- Then from this group find those that are closed by checking to see if there exists a superset that has the same supp
  is, the itemset is disqualified, but if non

Blue- frequent itemsets
thick blue- closed frequent itemsets
thick blue and have the yellow fill-
maximal frequent itemsets



FP-growth

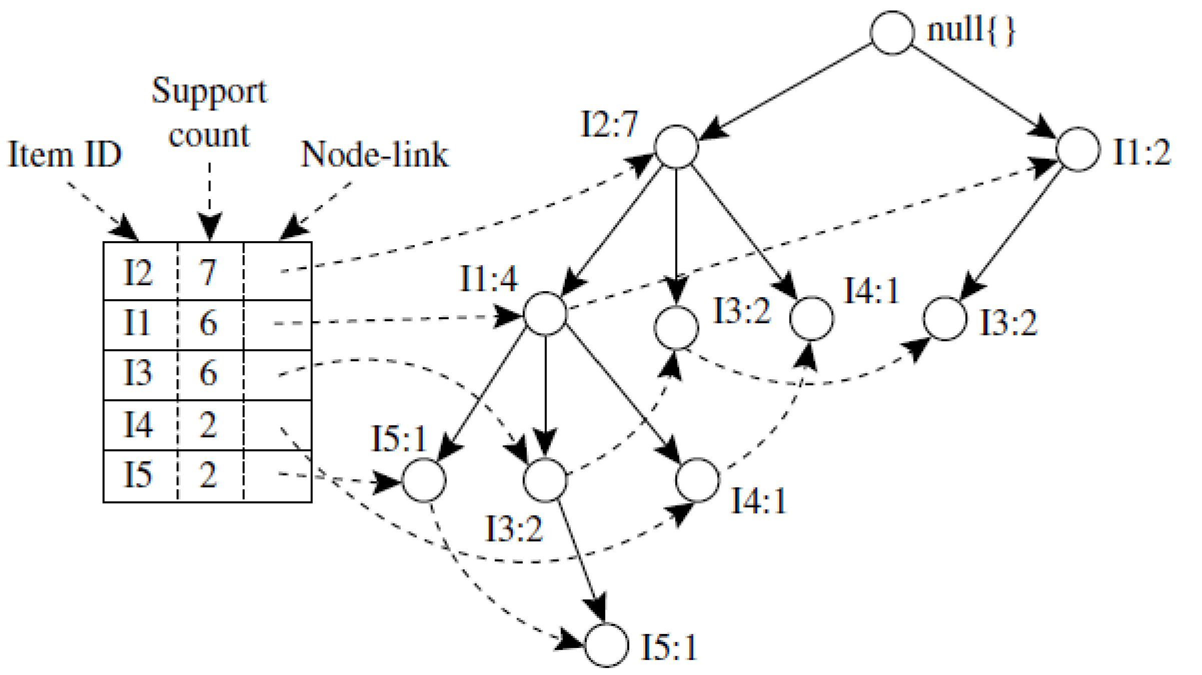
- **FP-growth (finding frequent itemsets without candidate generation).**

- **FP-growth** employs a *divide-and-conquer* strategy as follows.
- First, the database is compressed into a **FP-tree(frequent pattern tree)**.
- FP- tree is then divided into a set of projected databases called *conditional databases*, each associated with one frequent item.
- Every Conditional database is mined separately so as to generate frequent patterns.

Transactional Data for an *AllElectronics* Branch

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

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or *list* is denoted by *L*. Thus, we have $L = \{\{I2: 7\}, \{I1: 6\}, \{I3: 6\}, \{I4: 2\}, \{I5: 2\}\}$.



An FP-tree registers compressed, frequent pattern information.

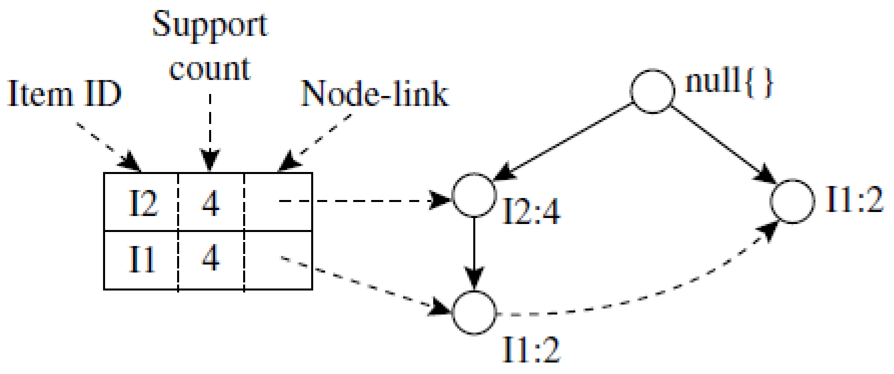
Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

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}

I5 occurs in two FP-tree branches . The paths formed by these branches are ⟨I2, I1, I5: 1⟩ and ⟨I2, I1, I3, I5: 1⟩. Therefore, considering I5 as a suffix, its corresponding two prefix paths are ⟨I2, I1: 1⟩ and ⟨I2, I1, I3: 1⟩, which form its conditional pattern base. Using this conditional pattern base as a transaction database, we build an I5-conditional FP-tree, which contains only a single path, ⟨I2: 2, I1: 2⟩; I3 is not included because its support count of 1 is less than the minimum support count. The single path generates all the combinations of frequent patterns: {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}.

For I4, its two prefix paths form the conditional pattern base, {{I2 I1: 1}, {I2: 1}}, which generates a single-node conditional FP-tree, ⟨I2: 2⟩, and derives one frequent pattern, {I2, I4: 2}.

Similar to the preceding analysis, I3's conditional pattern base is {{I2, I1: 2}, {I2: 2}, {I1: 2}}. Its conditional FP-tree has two branches, ⟨I2: 4, I1: 2⟩ and ⟨I1: 2⟩, as shown in Figure below which generates the set of patterns {{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}}. Finally, I1's conditional pattern base is {{I2: 4}}, with an FP-tree that contains only one node, ⟨I2: 4⟩, which generates one frequent pattern, {I2, I1: 4}.



The conditional FP-tree associated with the conditional node I3.