# UNIT 2: ASSOCIATION ANALYSIS: BASIC CONCEPTS AND ALGORITHMS

Date No.

## Association Analysis

* Useful for discovering interesting relationships in large dataset
* is used to learn about the purchasing behaviour of customers
* Applications are marketing promotions, inventory management & customer relationship management bioinformatics, medical diagnosis, web mining and scientific data analysis

In the analysis of earth science data, for example; the association patterns may reveal interesting connections among the ocean, land and atmospheric process.

## Issues in Association Analysis

* Discovering patterns from a large transaction dataset can be computationally expensive
* Some of the discovered patterns may be potentially spurious because they may happen simply by chance

## Basic Terminologies

### Binary representation

Market basket data can be represented in a binary format as shown below

| TID | Bread | Milk | Diapers | Beer | Egg | Cola |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 0 | 0 | 1 |

* Here each row corresponds to a transaction
* Each column corresponds to an item.

## Itemset

* A collection of zero or more items is termed as an itemset.

* A set of items that appear together in a transaction

**K- itemset** : If a itemset contains K items, it is called K- itemset

Eg: 3- itemset is {Beer, Diapers, Milk}
null (empty) itemset is { }

## Support Count $\sigma(x)$

Refers to the number of transactions that contain a particular itemset

Eg: $\sigma$(Bread) = 4
$\sigma$(Eggs) = 1
$\sigma$({Bread, Milk}) = 3
$\sigma$({Diapers, Beer}) = 3

take eg ① from next page for reference

$$\sigma(x) = |\{t_i \mid x \subseteq t_i, t_i \in T\}|$$

$x \rightarrow$ itemset
$t_i \rightarrow$ transaction i
$T \rightarrow$ set of transactions
$|\,| \rightarrow$ denote the number of elements in a set

## Association Rule

An Association Rule is an implication expression of the form $x \rightarrow y$, where $x$ & $y$ are disjoint itemsets
i.e., $x \cap y = \phi$
i.e., a rule which implies people who buy $x$ also buy $y$.

## Support $S(x, y)$

Support of an implication $x \rightarrow y$ is the no. of transactions having $\{x, y\}$ purchased together divided by total no. of transactions.

$$\text{Support } S(x \rightarrow y) = \frac{\text{Support count } (x \cup y)}{\text{total no. of transaction}}$$

$$= \frac{\sigma(x \cup y)}{N}$$

* Support determines how often a rule is applicable to a given dataset.
* Fraction of transactions that contain both $x$ & $y$

## Confidence

Confidence determines how (often) (a rule) frequently items in $y$ appear in transactions that contain $x$.

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

Example: ①

| TID | Items |
|---|---|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diaper, Beer} |
| 5 | {Bread, Milk, Diaper, Cola} |

For all the above transaction, find

i) Support Count {Bread, Milk} = 3

ii) Support Count {Diaper, Cola} = 2

iii) Support Count {Bread} = 4

Consider the rule {Milk, diapers} → {Beer}
Find the support & Confidence

$$\text{Support} = \frac{\sigma(\{\text{Milk, Diapers, Beer}\})}{5} = \frac{2}{5} = 0.4$$

$$\text{Confidence} = \frac{\sigma(\{\text{Milk, Diapers, Beer}\})}{\sigma(\{\text{Milk, Diapers}\})} = \frac{2}{3} = 0.667$$

* The support above shows only 40% of the customers purchased the three items together.

* The Confidence above shows that 67% of the customers who purchased Milk & Diapers also purchased Beer.

## Why use Support and Confidence ?

* Support is an important measure because a rule that has very low support may occur simply by chance

* A low support rule is uninteresting from a business perspective, as it may not be profitable to promote items that customers seldom buy together

* Support can be used to eliminate uninteresting rules.

* Confidence provides an estimate of the conditional probability of Y given X.

## minsup and minConf

minsup is support threshold
minConf is Confidence threshold.

## Association Rule Discovery

Given a set of transactions T, find all the rules

having support ≥ minsup & Confidence ≥ minconf.

## Brute Force Approach Association Rule Discovery

* Computes the support and Confidence for every possible rule
* Expensive as it is exponential
* Not used

## Another method of association rule discovery

The rule discovery is divided into two major tasks

### 1- Frequent Itemset Generation

objective is to find all the itemsets that satisfy the minsup threshold. These itemsets are called frequent itemsets

### 2. Rule Generation

objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules

Frequent itemset generation is computationally expensive.

## Frequent Itemset Generation

Let $I = \{a, b, c, d, e\}$

Since the number of items in $I$ is 5, the no. of itemsets is $2^5 - 1$ excluding the null set

i.e., for $k$ items the possible number of itemsets are $2^k - 1$ excluding the null set

This can be represented by/using a Lattice structure.

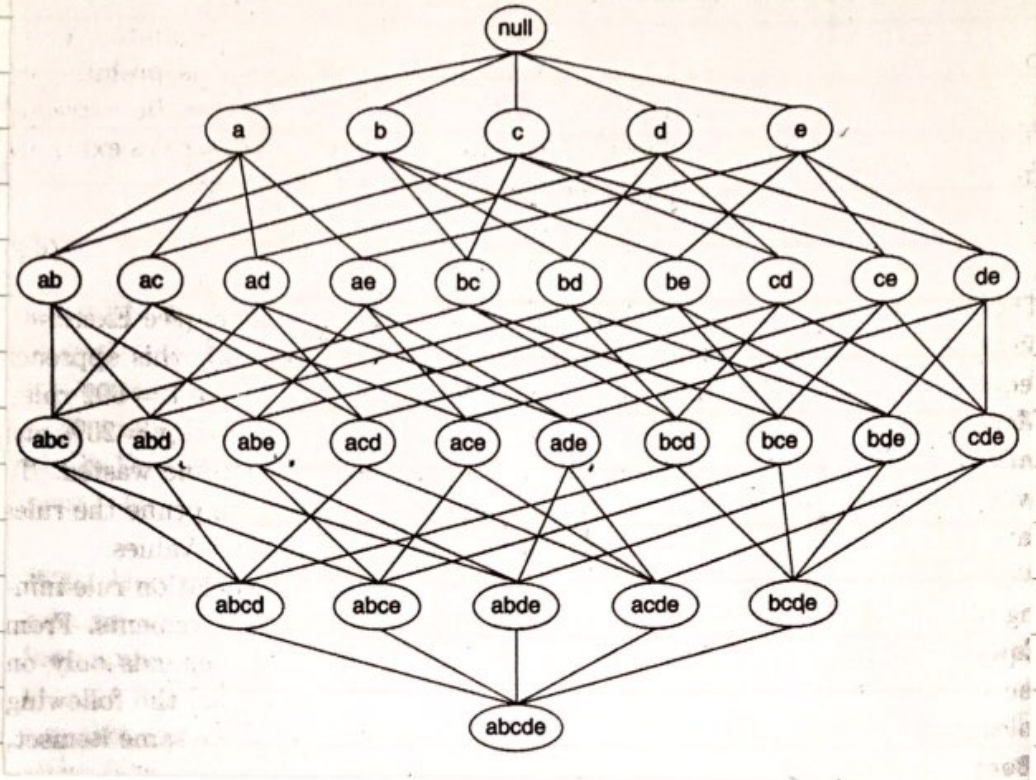



fig: Itemset Lattice

## Brute Force Method of finding frequency itemset

Determine the support count for every candidate itemset in the Lattice structure.

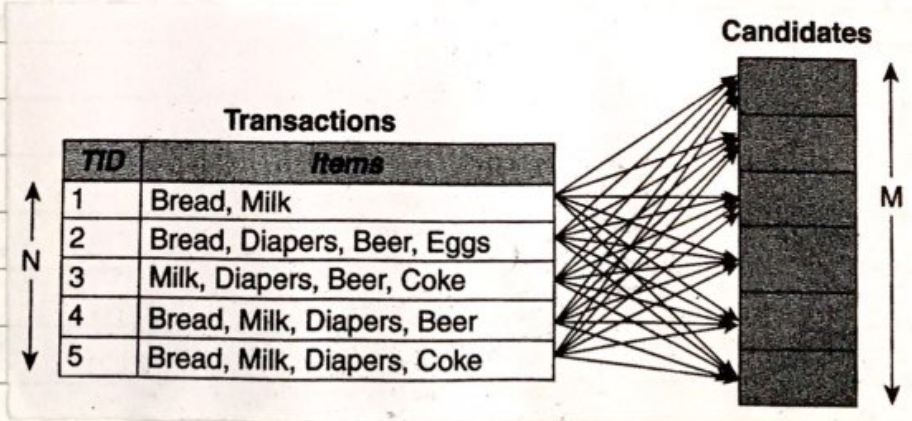



fig: Counting the support of candidate itemsets

$M = 2^k - 1$

This approach is very expensive as it requires $O(NMw)$ comparisions,

where,

$N \rightarrow$ no. of transactions

$M \rightarrow 2^k - 1$ , no. of candidate itemsets
$w \rightarrow$ Maximum transaction width

* The computational complexity of frequent itemset generation can be reduced by
  i) Reducing the no. of candidate itemsets (M)
     - Apriori principle is used
  ii) Reducing the no. of comparisions

## The Apriori Principle

- If an itemset is frequent, then all of its subsets must also be frequent
- Suppose {c, d, e} is a frequent itemset, then {c, d}, {c, e}, {d, e}, {c}, {d}, {e} must also be frequent

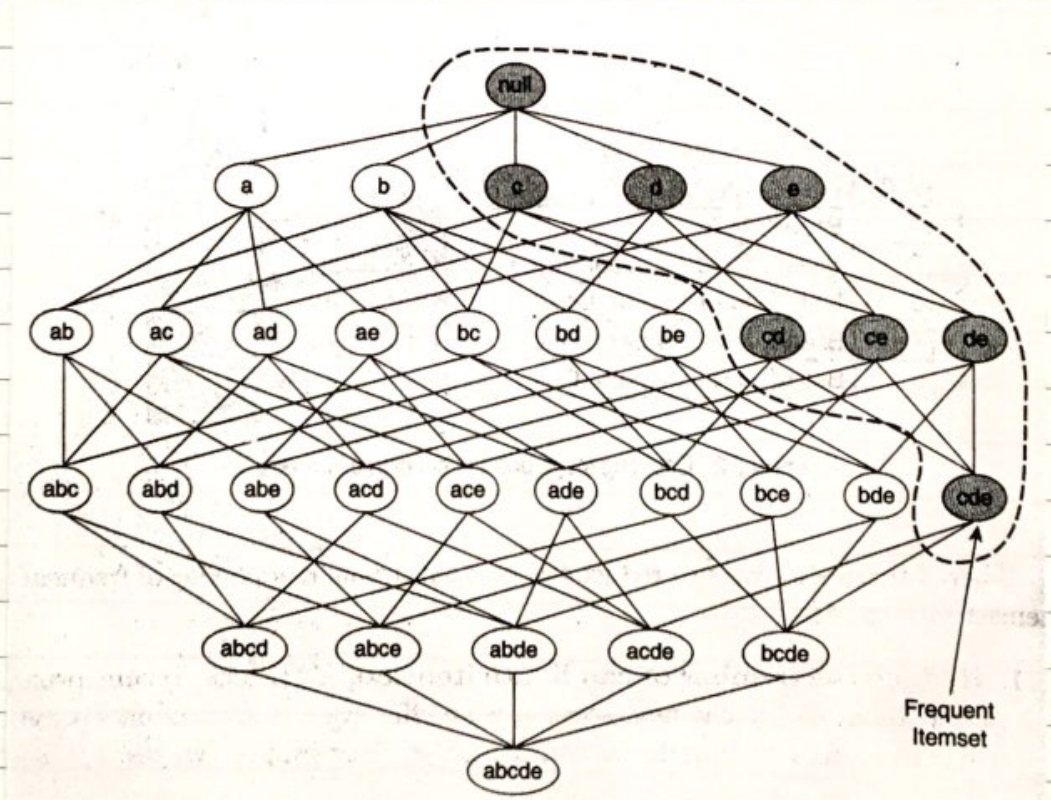



fig: An illustration of the Apriori principle. If {c, d, e} is frequent, then all subsets of this itemset are frequent

- Conversly, if an itemset {a, b} is infrequent, then all of its supersets are infrequent too.

## Pruning:

Trim (a tree, shrub or bush) by cutting away dead or overgrown branches or stems.

The entire subgraph containing the supersets of {a, b} can be pruned immediately once {a, b} is found to be infrequent.

This strategy of trimming the exponential search based on the support measure is known as **support-based pruning**

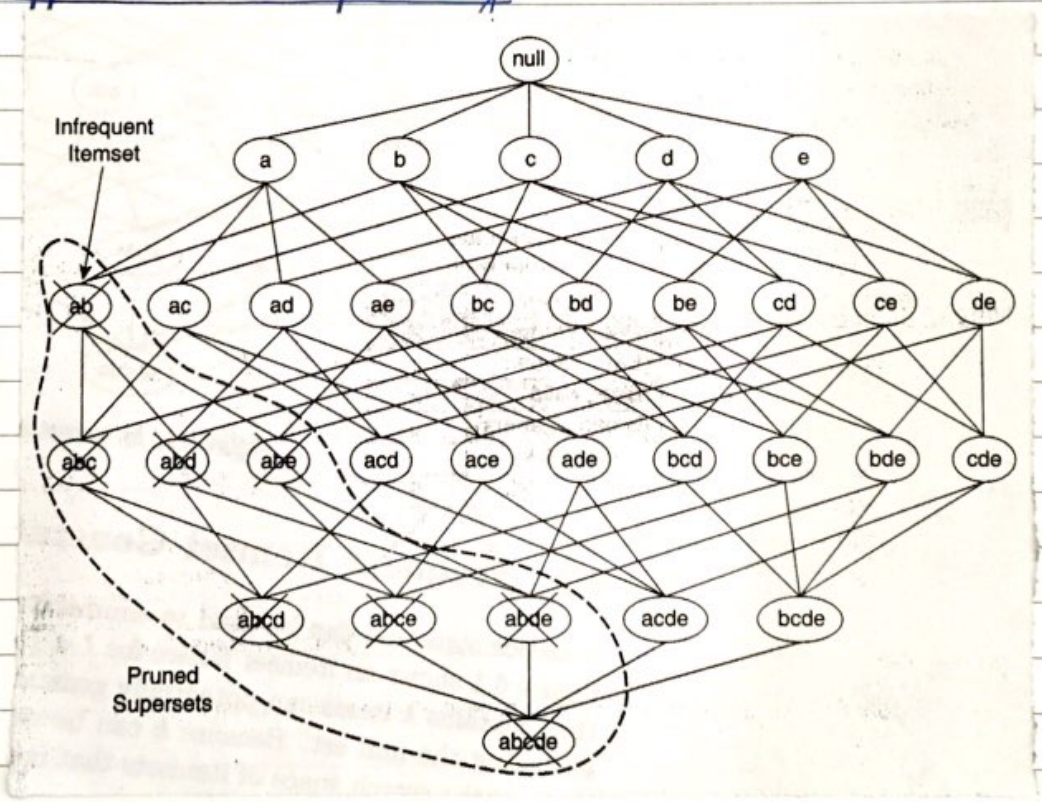



fig: An illustration of Support-Based Pruning of {a, b} is infrequent, then all supersets of {a, b} are infrequent.

## Frequent itemset generation in the Apriori Algorithm

For the given market basket transactions, generate frequent itemsets

| TID | Items |
|---|---|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diapers, Beer} |
| 5 | {Bread, Milk, Diapers, Cola} |

**Step 1:** Generate Candidate 1- itemsets

| Item | Count |
|---|---|
| Beer | 3 |
| Bread | 4 |
| Cola | 2 |
| Diapers | 4 |
| Milk | 3 |
| Eggs | 1 |

Assume minsup = 3
i.e; minimum support count = 3

Keep the items whose count is $\geq 3$ & eliminate the others

Cola and Eggs get eliminated as the support count is 2 less than 3.

After Elimination we have,
Frequent 1- itemsets

| Item | Count |
|---|---|
| Beer | 3 |
| Bread | 4 |
| Diapers | 4 |
| Milk | 3 |

Now,
From the 1- itemset, generate Candidate 2- itemset

Candidate 2-itemsets

| Itemset | Count |
|---|---|
| {Beer, Bread} | 2 |
| {Beer, Diapers} | 3 |
| {Beer, Milk} | 2 |
| {Bread, Diapers} | 3 |
| {Bread, Milk} | 3 |
| {Diapers, Milk} | 3 |

After Elimination we get Frequent 2-itemset

| Itemset | Count |
|---|---|
| {Beer, Diapers} | 3 |
| {Bread, Diapers} | 3 |
| {Bread, Milk} | 3 |
| {Diapers, Milk} | 3 |

Generate Candidate 3-itemsets

| Itemset | Count |
|---|---|
| {Beer, Diapers, Bread} | 2 |
| {Beer, Diapers, Milk} | 2 |
| {Bread, Diapers, Milk} | 2 |

The process stops here, as we have no candidate 3-itemsets, where support count is 3 or greater.

Algorithm: Frequent itemset generation of the Apriori Algorithm

1: $k = 1$

2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N * minsup \}$
// Find all frequent 1-itemsets

3: repeat

4: $k = k + 1$

5: $C_k = apriori\_gen(F_{k-1})$
// Generate candidate itemsets

$F_k$ = frequent itemset of size k
$C_k$ = candidate itemset of size k
T = Total transactions



6: for each transaction $t \in T$ do
7: $C_t$ = subset $(C_k, t)$
// identify all candidates that belong to t
8: for each candidate itemset $C \in C_t$ do
9: $\sigma(C) = \sigma(C) + 1$
// increment support count
10: end for
11: end for
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N * minsup \}$
// extract the frequent K-itemsets
13: until $F_k = \phi$
14: Result = $\cup F_k$

Explanation:
- Let K = 1
- Generate frequent itemset of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (K+1) candidate itemsets from length K frequent itemsets
  - Prune candidate itemsets containing subsets of length K that are infrequent
  - Count the support of each candidate by scanning the database
  - Eliminate candidates that are infrequent, leaving only those are frequent

## Candidate Generation and Pruning

* The apriori_gen function shown in step 5 of the algorithm generates candidate K-itemsets by performing the following two operations:

1) Candidate Generation:
This operation generates new candidate K-itemsets based on the frequent (K-1) - itemsets found in the previous iteration.

2) Candidate Pruning:

This operation eliminates some of the candidate K-itemsets using support-based pruning strategy

* To illustrate Candidate Pruning, Consider a candidate K-itemset, $X = \{i_1, i_2, \ldots i_k\}$. The algorithm must determine whether all of its proper subsets $X - \{i_j\}$ $\forall j = 1, 2, \ldots, k$ are frequent. If one of them is infrequent, then X is immediately pruned.

Complexity of this operation is $O(k)$

## Requirements for an effective candidate generation procedure

1. It should avoid generating too many unnecessary candidates
2. It must ensure that the candidates set is complete i.e., no frequent itemsets are left out
3. It should not generate the same candidate itemset more than once

Several Candidate generation procedures are

1. Brute Force method
2. $F_{K-1} \times F_1$ method
3. $F_{K-1} \times F_{K-1}$ method – used by apriori_gen function

## Computational Complexity

Computational Complexity of Apriori Algorithm can be affected by the following factors.

1. Support threshold
   - lowering the support threshold results in more frequent itemsets

- this increases computational complexity because more candidate itemsets must be generated and counted.

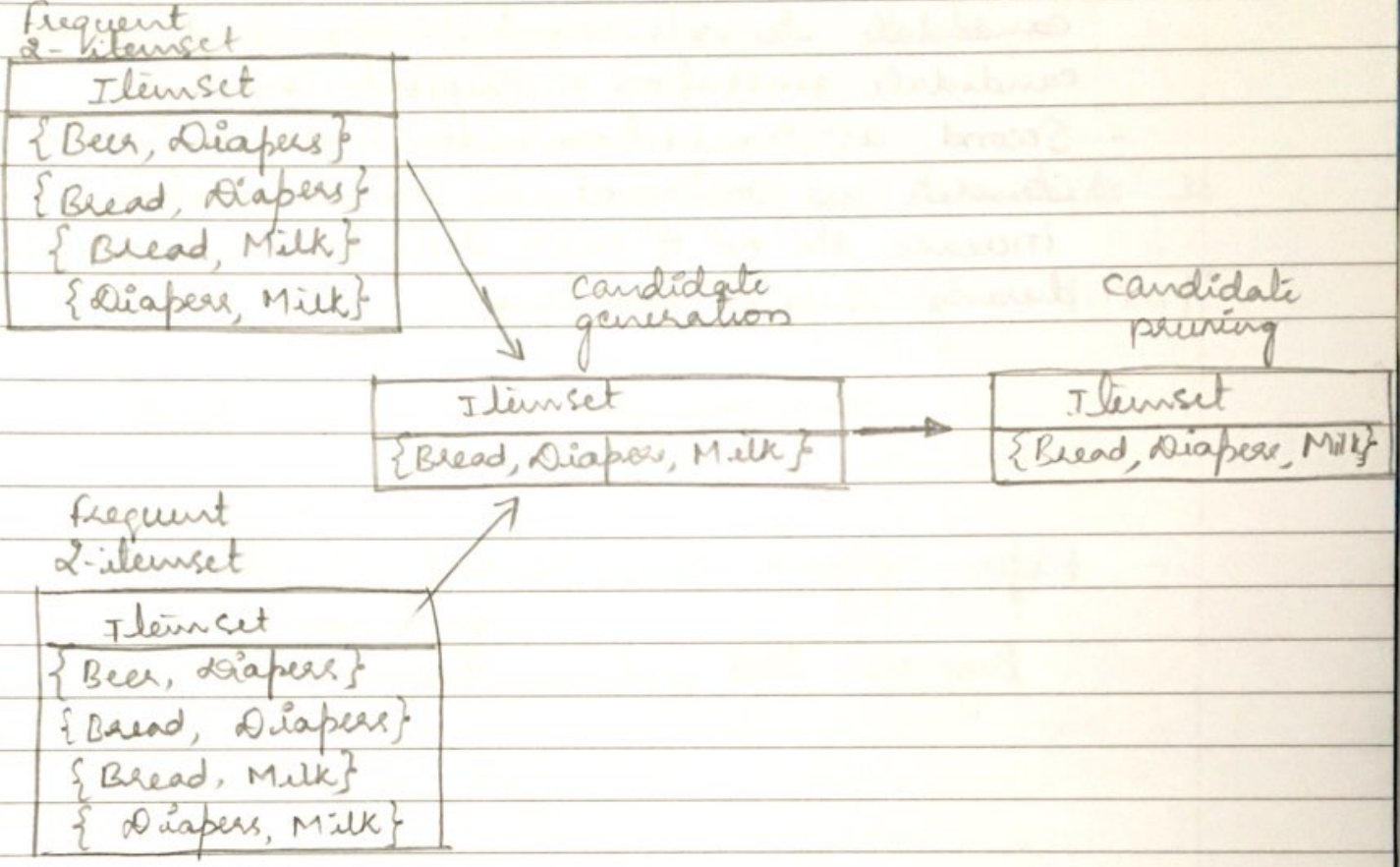



fig: Generating and pruning candidate k-Itemsets by merging pairs of frequent (k-1) itemsets.

2. Number of items (Dimensionality)
   - As no. of items increases, more space will be needed to store the support count of items
   - This results in more computation and I/O cost

3. Number of Transactions
   Larger the no. of transactions, run time of Apriori algorithm increases.

4. Average Transaction Width

- First, max. size of frequent items increase as average transaction increases. As a result, more candidate itemsets must be examined during candidate generation & support counting.
- Second, as transaction width increases, more itemsets are contained in the transaction. This will increase the no. of hash tree traversals performed during support counting.

Refer figure 6.13(a) and 6.13(b)

Page No. 347

5. Generation of frequent 1-itemset :
- For each transaction, we need to update the support count for every item present in the transaction.
- This operation requires $O(NW)$ time.

6. Candidate Generation
- Overall cost of merging frequent itemsets is

$$\sum_{K=2}^{\omega} (K-2)\,|C_K| < \text{cost of merging} < \sum_{K=2}^{\omega} (K-2)\,|F_{K-1}|^2$$

- Candidate pruning step requires

$$O\left(\sum_{K=2}^{\omega} K(K-2)\,|C_K|\right) \text{ time}$$

7. Support counting
cost for support counting is

$$O\left(N \sum_{K} \binom{\omega}{K} \alpha_K\right)$$

# FP - Growth Algorithm (Frequent Pattern Growth)

* Uses separate approach when compared to Apriori
* Encodes the data set using a compact data structure called FP-tree and extracts frequent itemsets directly from this structure

## FP-tree Representation

* An FP-tree is a compressed representation of the input data
* It is constructed by reading the data set one transaction at a time and mapping each transaction onto a path in the FP-tree
* If the different transactions have the same items, then the paths may overlap & this leads to compressed FP-tree.
* Each node of tree represents the item & its support count

## Steps to construct FP-tree

1. Determine the support count of all the items
2. Find the frequent 1-itemset using min_sup and discard the infrequent 1-itemsets
3. Arrange the obtained frequent items in the descending order of support count in each transaction.

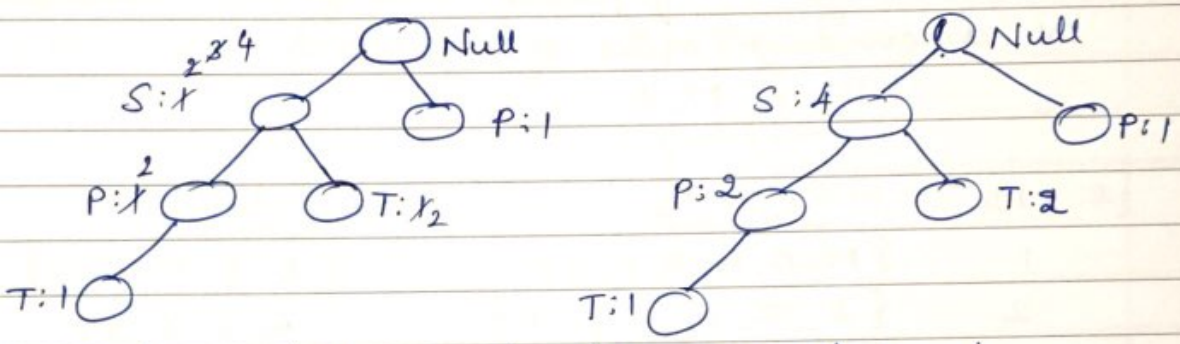
## Example:

(1)

| TID | Items |
|---|---|
| 1 | {S, P, T} |
| 2 | {P} |
| 3 | {S, T} |
| 4 | {S, T} |
| 5 | {S, P} |

| Items | Support Count |
|---|---|
| S | 4 |
| T | 3 |
| P | 3 |

Construct FP-tree




T: is eliminated coz of less support count

| Items | Conditional pattern base | Conditional FP-tree |
|---|---|---|
| P | {S:2} → {S:2} | <S, P:2> ✗ |
| S | | |
| T | {S,P:1} {S:2} → {S:3} | <S, T:3> ✓ |

(2) Example

| TID | Itemset |
|---|---|
| 1 | {M, O, N, K, E, Y} |
| 2 | {D, O, N, K, E, Y} |
| 3 | {M, A, K, E} |
| 4 | {M, U, C, K, Y} |
| 5 | {C, O, K, I, E} |

| Item | Sup-count | |
|---|---|---|
| M | 3 | |
| O | 3 | |
| N | 2 | ✗ |
| K | 5 | |
| E | 4 | |
| Y | 3 | |

| Item | Sup-count | |
|---|---|---|
| D | 1 | ✗ |
| A | 1 | ✗ |
| U | 1 | ✗ |
| C | 2 | ✗ |
| I | 1 | ✗ |

Date No.

By assuming min Sup = 3.
The frequent 1-itemset could be as shown in the table above ie.,

M, O, K, E, Y

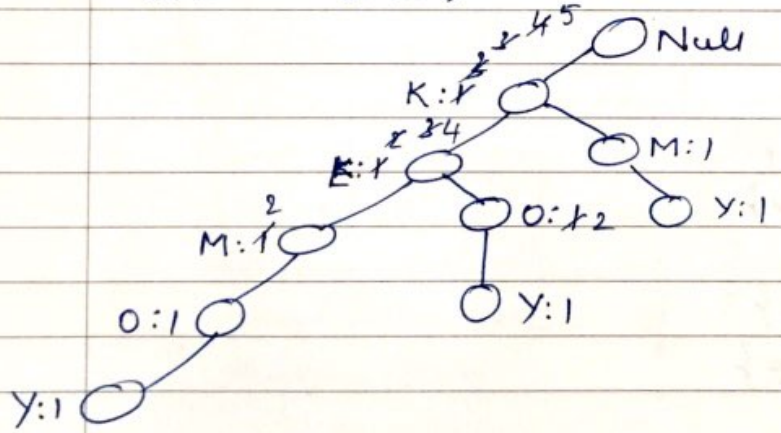
Lets arrange the frequent 1-itemsets according to the descending order of sup-count

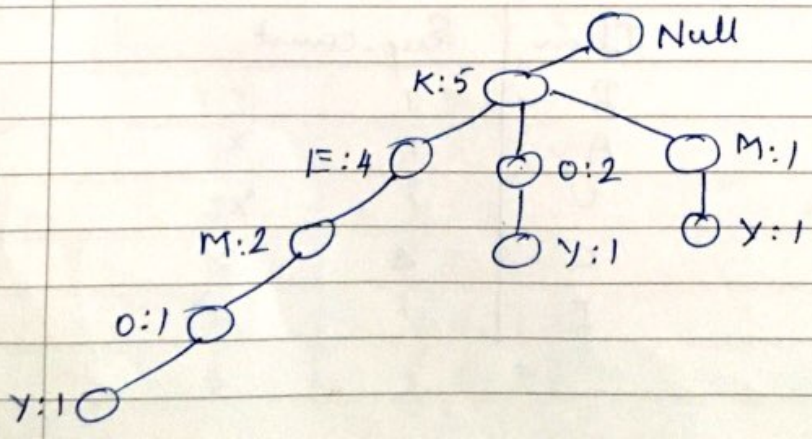
K, E, M, O, Y

| TID | Current itemsets | ordered itemsets |
|---|---|---|
| 1 | {M, O, N, K, E, Y} | {K, E, M, O, Y} |
| 2 | {D, O, N, K, E, Y} | {K, E, O, Y} |
| 3 | {M, A, K, E} | {K, E, M} |
| 4 | {M, U, C, K, Y} | {K, M, Y} |
| 5 | {C, O, K, I, E} | {K, E, O} |

## FP-tree Construction

Read the first transaction and create the nodes of the FP-tree,



Date No.

Using FP-tree, now write the items in the ascending order of frequency (occurence)

i.e; Y, O, M, E, K(highest)

Find Conditional pattern base i.e, to reach a item, path used. For eg: to reach y there are 3 paths KEMO, KEO & KM with y's frequency

| Items | Conditional pattern base | Conditional FP-tree |
|---|---|---|
| Y | {KEMO:1}, {KEO:1}, {KM:1} | {K:3} {KM:2} {KE:2} |
| O | {KEM:1}, {KE:2} | {KE:3} |
| M | {KE:2}, {K:1} | {K:3} |
| E | {K:4} | {K:4} |
| K | | |

## frequent pattern Generated

<K, Y:3>

<K, O:3> <E, O:3> <K, E, O:3>

<K, M:3>

<K, E:4>

Conditional FP-tree is obtained by finding only the common elements in the conditional pattern base for each item & adding the frequency.

## Compact Representation of Frequent Itemsets

* The number of frequent itemsets produced from a transaction dataset can be very large
* It is useful to identify a small representative set of itemsets from which all other frequent itemsets can be derived
* Two such representations are maximal and closed frequent itemsets

## Maximal Frequent Itemsets

A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent

Maximal frequent itemsets effectively provide a compact representation of frequent itemsets.

In other words, they form the smallest set of itemsets from which all frequent itemsets can be derived.

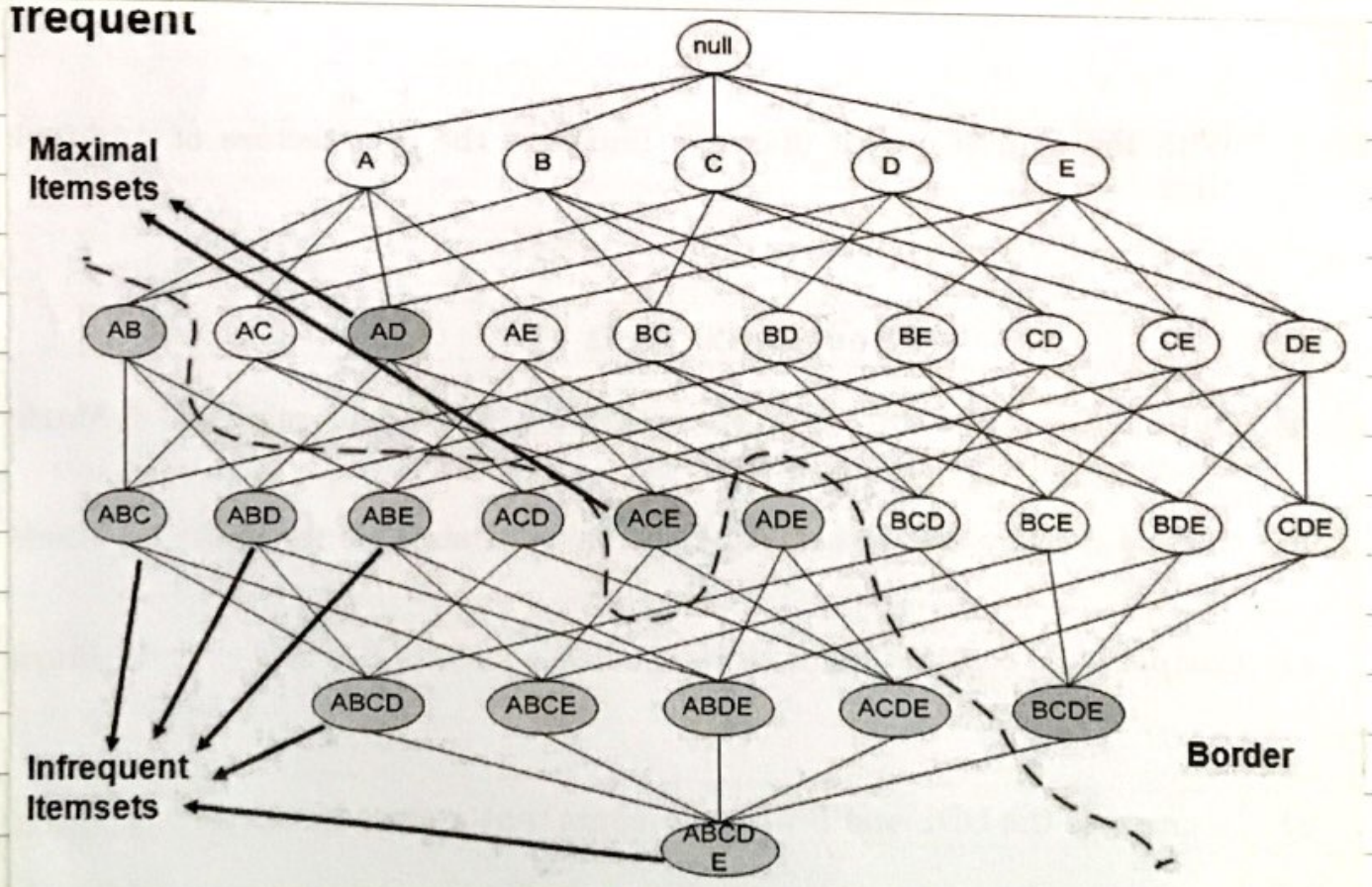



fig: Maximal frequent itemset

The frequent itemsets can be divided into 2 groups

* Frequent itemsets that begin with item a and that may contain items c, d, or e. This group includes items such as {a}, {a,c}, {a,d}, {a,e} and {a,c,e}
* Frequent itemsets that begin with items b, c, d or e. This group includes itemsets such as {b}, {b,c}, {c,d} {b,c,d,e} etc

## Closed Frequent Itemsets

* A frequent itemset X is closed if none of its immediate supersets has exactly the same support count as X.
* Closed frequent itemsets are useful for removing some of the redundant association rules
* All maximal frequent itemsets are closed, as none of the maximal frequent itemsets can have the same support count as their immediate supersets

| TID | Items |
|---|---|
| 1 | ABC |
| 2 | ABCD |
| 3 | BCE |
| 4 | ACDE |
| 5 | DE |

minSup = 40%

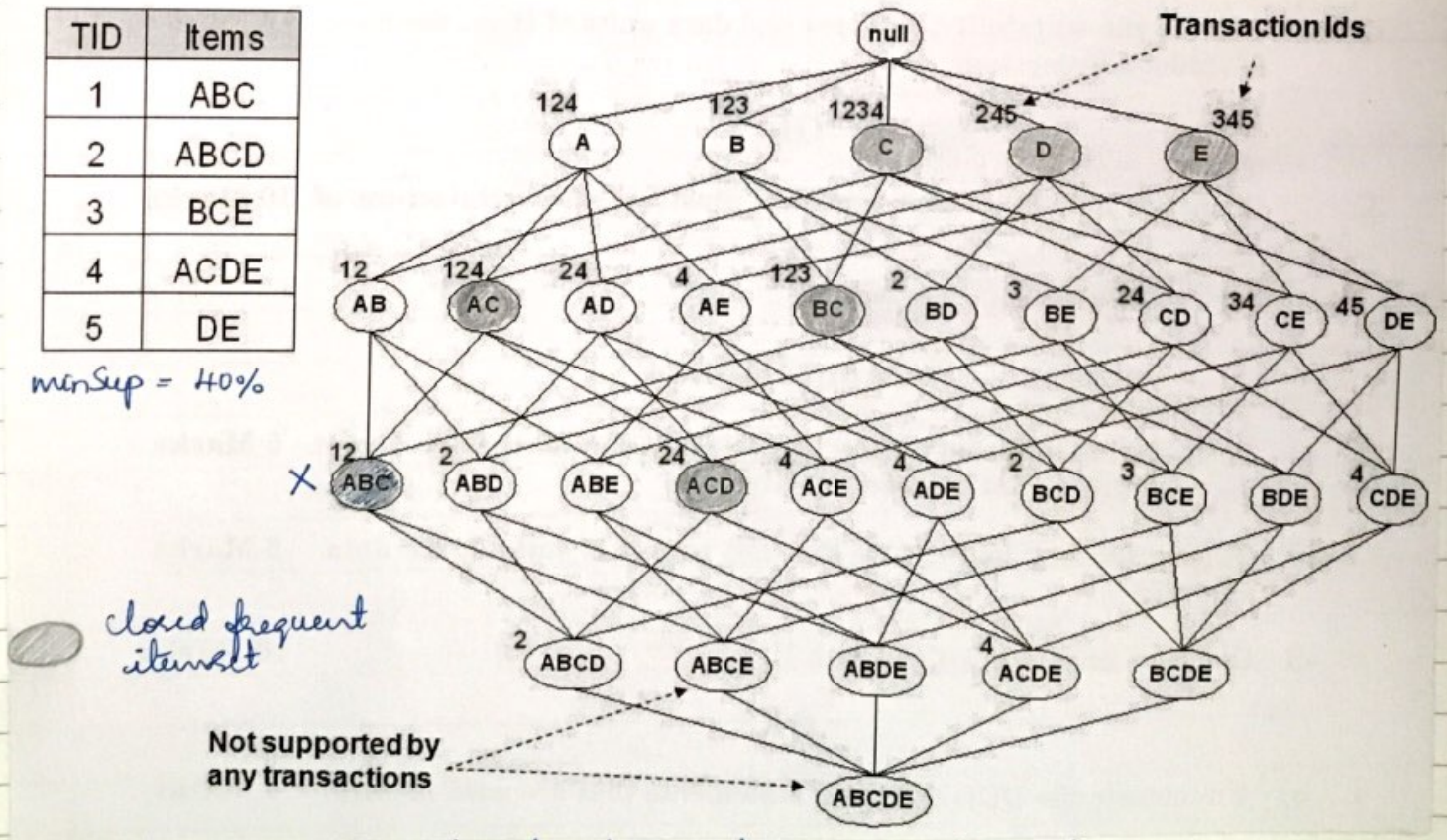



fig: An example of closed frequent itemsets (with min. support count equal to 40%)

MATRIKAS

Date No.

## Rule Generation

* Each frequent k-itemset Y can produce upto $2^k - 2$ association rules, ignoring rules that have empty antecedents or consequents ($\phi \rightarrow Y$ or $Y \rightarrow \phi$)

* An association rule can be extracted by partitioning the itemset Y into two non-empty subsets, X & Y−X, such that $X \rightarrow Y - X$ satisfies the confidence threshold

Eg:- Let $X = \{1, 2, 3\}$ be a frequent itemset, then the possible subsets include
$\{\}, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}$ and the rules in the order would be

$\{\} \rightarrow \{1,2,3\} \rightarrow$ not possible
$\{1\} \rightarrow \{2,3\}$
$\{2\} \rightarrow \{1,3\}$
$\{3\} \rightarrow \{1,2\}$
$\{1,2\} \rightarrow \{3\}$
$\{1,3\} \rightarrow \{2\}$
$\{2,3\} \rightarrow \{1\}$
$\{1,2,3\} \rightarrow \{\} \rightarrow$ not possible

Hence only 6 combinations

## Confidence-Based Pruning

* Confidence does not have any monotone property
* If a rule $X \rightarrow Y - X$ does not satisfy the confidence threshold, then any rule $X' \rightarrow Y - X'$ where, $X'$ is a subset of $X$ must not satisfy the confidence threshold as well

Date No.

## Rule Generation in Apriori Algorithm

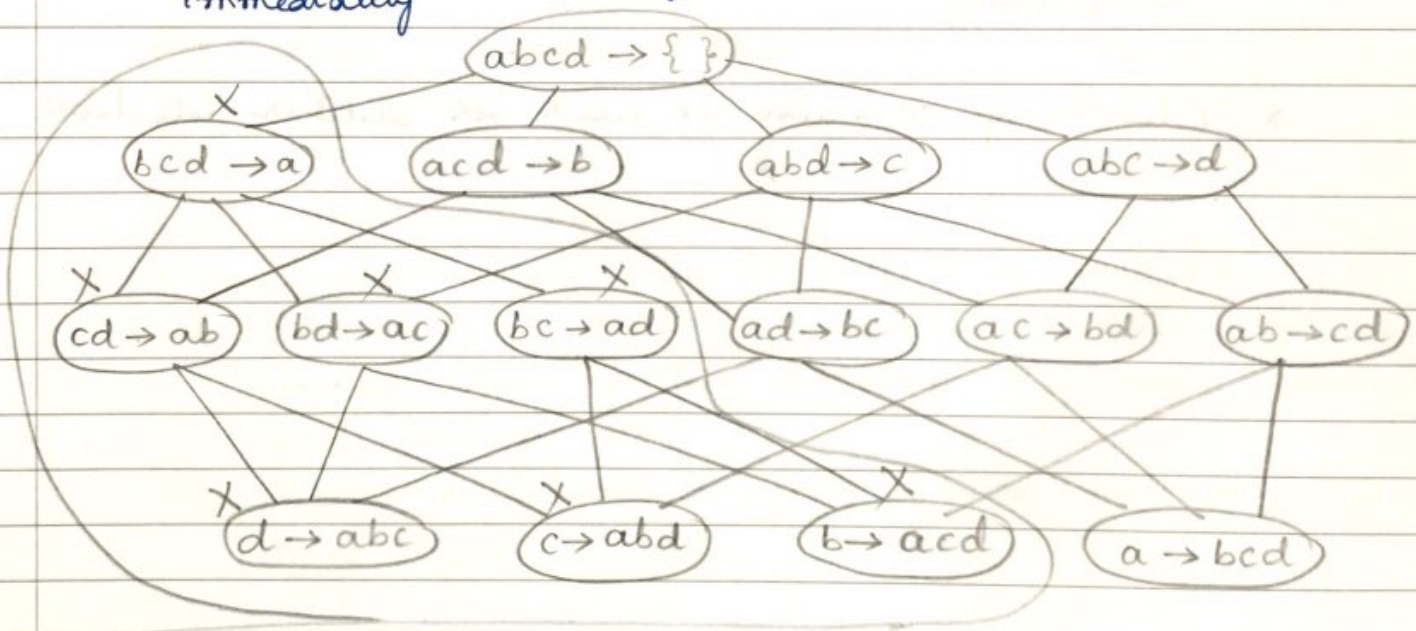
Uses a level-wise approach for generating association rule

* Initially all the high-confidence rules that have only one item in the rule consequent are extracted. These rules are then used to generate new candidate rules

Eg:- if {acd} → {b} & {abd} → {c} are high confidence rules, then the candidate rule {ad} → {bc} is generated by merging the consequents of both rules

Antecedent → A → B → Consequent

* Any node in the lattice has low confidence, the entire subgraph spanned by the node can be pruned immediately




Since {bcd} → {a} has less confidence, all its subsets could be pruned, hence all crossed ones (subsets of the rule {bcd} → {a}) are pruned

## What is Cluster Analysis ?

* Cluster Analysis or simply clustering is the process of Partitioning a set of data objects into subsets

* Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters

* The set of clusters resulting from a cluster analysis can be referred to as clustering

* Clustering is also called data segmentation in some application because clustering partitions large datasets into groups according to their similarity

* clustering can also be used for outlier detection.