

Skill Based Subject – III: DATA MINING AND WAREHOUSING - 18BIT55S

UNIT I: Introduction: Data Mining Applications – Data Mining Techniques – The Future of Data Mining – Data Mining Software. Association Rule Mining: Introduction – Basics – The Task and a Naïve Algorithm – The Apriori Algorithm – Improving the Efficiency of the Apriori Algorithm – Mining Frequent patterns without Candidate Generation (FP-Growth) – Performance Evaluation of Algorithms.

TEXT BOOK

G.K Gupta, “Introduction to Data Mining with Case Studies”, Prentice Hall of India(Pvt) Ltd, India, 2008.

Prepared by : Mrs. G. Shashikala, Assistant Professor, PG Department of Information Technology

What is Data Mining?

- Data Mining is defined as extracting information from huge sets of data. In other words, data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications: • Market Analysis • Fraud Detection • Customer Retention • Production Control • Science Exploration

- Data mining (knowledge discovery from data)
 - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
- Alternative name
 - Knowledge discovery in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.

What is Data Warehousing?

- A **Data Warehousing** (DW) is the process for collecting and managing data from varied sources to provide meaningful business insights. A Data warehouse is typically used to connect and analyze business data from heterogeneous sources. The data warehouse is the core of the BI system which is built for data analysis and reporting.

Why Data Mining?—Potential Applications

- Data analysis and decision support
 - Market analysis and management
 - Target marketing, customer relationship management (CRM), market basket analysis, market segmentation
 - Risk analysis and management
 - Forecasting, customer retention, quality control, competitive analysis
 - Fraud detection and detection of unusual patterns (outliers)



- Other Applications

- Text mining (news group, email, documents) and Web mining
- Stream data mining
- Bioinformatics and bio-data analysis

Data Mining vs. KDD

- *Knowledge Discovery in Databases (KDD)*: process of finding useful information and patterns in data.
- *Data Mining*: Use of algorithms to extract the information and patterns derived by the KDD process.

Why data mining now ?

- Growth in OLTP data
- Growth in data due to cards and mobile phones
- Growth in data due to the web
- Growth in data due to other sources
- Growth in data storage capacity
- Decline in the cost of processing
- Competitive environment
- Availability of software

The Data Mining Process

- Requirement analysis
- Data selection and collection
- Cleaning and preparing data
- Data mining exploration and validation
- Implementing, evaluating and monitoring
- Results validation

Other Methodologies

- SPSS – assess, access, analyse, act and automate
- SAS – sample, explore, modify, model and assess
- CRISP-DM (Cross-Industry Standard Process for Data Mining) – business understanding, data understanding, data preparation , modelling, evaluation, and deployment

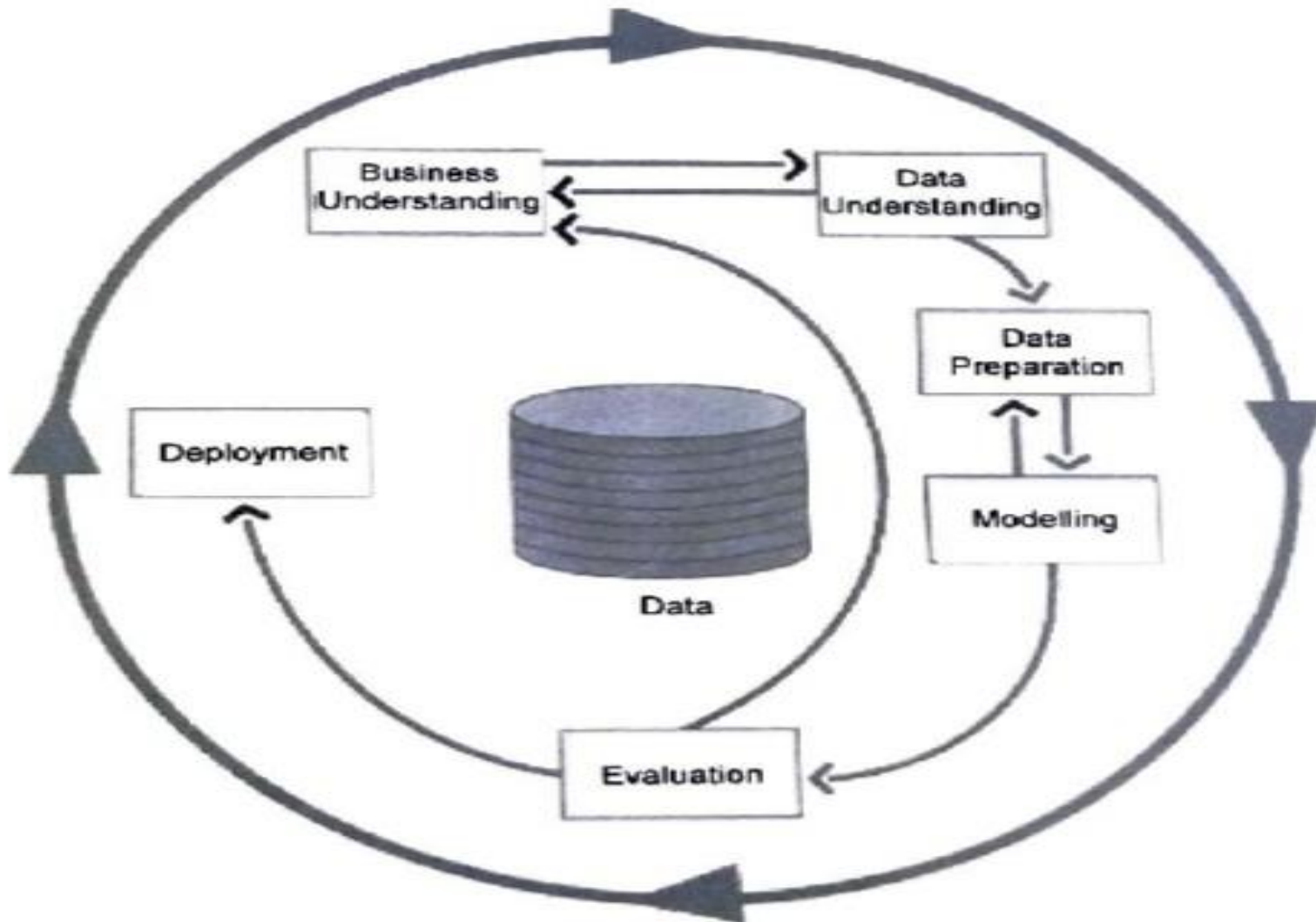


Figure 1.2 CRISP data mining process model.
(Source: <http://www.crisp-dm.org/Process/index.htm>)

Data mining Applications

- Prediction and description
- Relationship marketing
- Customer profiling
- Outliers identification and detecting fraud
- Customer segmentation
- Web site design and promotion

Data mining Techniques

- Association rules mining or market basket analysis
- Supervised classification
- Cluster analysis
- Web data mining
- Search engines
- Data warehousing and OLAP

- Association rules mining or market basket analysis
 - It is a technique that analyses a set of transactions i.e. items purchased by one customer
 - The aim is to identify which items are purchased together frequently so that they may be grouped together on store shelves
 - The term lift is used to measure the power of association between items that are purchased together
 - This has applications in marketing, medicine, electronic commerce etc.
 - Apriori algorithm is used to find associations



- Supervised classification

- This is used when classes are already known and some training data is available


- This is useful in predicting the class to which an object or individual is likely to belong

- The technique used is decision tree method



- Cluster analysis


- This is useful when the classes in data are not already known and the training data is not available
- Used to find groups that are very different from each other in a collection of data
- Widely used method is K-means algorithm – Here the user specifies both the number of clusters and the starting seed value
- Other algorithm used is hierarchical method

- 
- Web data mining
 - 500 million people are using the web everyday
 - Web mining may be divided into
 - Web content mining
 - Web structure mining
 - Web usage mining



- Search engines

- These are huge databases of Web pages as well as software packages for indexing and retrieving the pages asked by users
- Search engine databases of Web pages are built and updated by web crawlers
- During a search, a user is searching only the database compiled by the search engine and not the entire web
- Assigning ranks to a web page is a challenging problem in a search engine

- 
- Data warehousing and OLAP
 - **The data warehouse** works as a central repository where information is coming from one or more **data** sources
 - OLAP tools are decision support tools that are often built on top of a data warehouse or another database

The Future of Data Mining

- In the process of data mining most of the time is spent in data extraction, data cleaning and data manipulation and it is expected that technologies like data warehousing will grow
- Nearly 40% of all collected data contains errors. So it is likely to give more importance for building warehouses using data cleaning and extraction
- Techniques that are able to compare current data with an earlier set of data will gain importance in the future
- Other techniques that will receive more attention in the future are text and web-content mining, bioinformatics and multimedia data mining.

Data mining software

- A list of software packages are
- Angoss Software – Angoss has data mining software called KnowledgeSTUDIO. This software includes facilities for classification, cluster analysis and prediction. Angoss has another package called KnowledgeSEEKER that supports decision tree classification.
- CART and MARS This software from Salford Systems include CART decision trees, MARS predictive modelling, automated regression, TreeNet classification, and regression, data access, preparation, cleaning and reporting modules, RandomForests predictive modelling, clustering and anomaly detection.

Clementine – This software from SPSS provides association rules, classification, cluster analysis, factor analysis, forecasting, prediction and sequence discovery.

Data Miner Software Kit – It is a collection of data mining tools offered with the book “Predictive data mining : A practical guide “

DBMiner Technologies – This provides techniques for association rules, classification and cluster analysis. It interfaces with SQL server.

Enterprise Miner – This package from SAS provides a user-friendly, icon-based GUI front end using their process model called SEMMA (Sample, Explore, Modify, Model, Assess).

GhostMiner – This software includes data preprocessing, feature selection, k-nearest neighbors, neural nets, decision tree, SVM, PCA clustering and visualization

Intelligent Miner – This package from IBM includes association rules, classification, cluster analysis, prediction, sequential patterns and time series, and text mining

JDA Intellect – This provides for association rules, classification, cluster analysis and prediction.

Mantas – This focuses on detecting and analysing suspicious behaviour in financial markets

MCubiX from Diagnos – It is a data mining toolbox including decision tree, neural networks, association rules and visualization.

MineSet – This specializes in visualization

MiningMart – This focuses on data cleaning and provides a graphical tool for data preprocessing. It provides mining on relational databases and supports KDD process.

Oracle – Oracle 10g has data mining facilities embedded in it. Oracle also has a product called Darwin, which includes cluster analysis, prediction, classification and association rules.

Weka 3 – A collection of machine learning algorithms for solving data mining problems. Witten and Frank has developed this package.

Association Rules Mining

- Analysing a large database of supermarket transactions with the aim of finding association rules is called association rules mining or market basket analysis.
- Other applications for association rules mining are in marketing, customer segmentation, medicine, electronic commerce, classification, clustering, Web mining, bioinformatics and finance
- Basics

Assuming that a shop sells the products

Bread

Juice Milk

Biscuits

Cheese Coffee

Tea

Newspaper

Sugar

Each row in the table gives the set of items that one customer bought

Table 2.1 Transactions for a simple example

<i>Transaction ID</i>	<i>Items</i>
10	Bread, Cheese, Newspaper
20	Bread, Cheese, Juice
30	Bread, Milk
40	Cheese, Juice, Milk, Coffee
50	Sugar, Tea, Coffee, Biscuits, Newspaper
60	Sugar, Tea, Coffee, Biscuits, Milk, Juice, Newspaper
70	Bread, Cheese
80	Bread, Cheese, Juice, Coffee
90	Bread, Milk
100	Sugar, Tea, Coffee, Bread, Milk, Juice, Newspaper

keeper wants to si

The shop keeper wants to know which products are sold together frequently - A sale on one item will increase the sales of another

Let the number of items in the shop is n

The items are represented by $I \{i_1, i_2, \dots, i_n\}$

Let there be N transactions denoted as $T \{t_1, t_2, \dots, t_N\}$

Let each transaction of m items be $\{i_1, i_2, \dots, i_m\}$ with $m \leq n$

Here quantities of items bought are not considered

Association rules are written as $X \rightarrow Y$ meaning that whenever X appears, Y also tends to appear.

X and Y may be single item or sets of items

X is referred as the rule's antecedent and Y as the consequent

$X \rightarrow Y$ is a probabilistic relationship

Suppose X and Y appear together in only 10% of the transactions but whenever X appears there is an 80% chance that Y also appears.

The 10% presence of X and Y together is called the support (or prevalence) of the rule and the 80% chance is called the confidence (or predictability) of rule.

If the total number of transactions is N,

$$\text{Support}(X) = (\text{Number of times } X \text{ appears})/N = P(X)$$

$$\text{Support}(XY) = (\text{Number of times } X \text{ and } Y \text{ appear together})/N = P(X \cap Y)$$

Confidence for $X \rightarrow Y$ is defined as the ratio for the support for X and Y together to the support for X

Therefore if X appears much more frequently than X and Y appear together, the confidence will be low.

$$\begin{aligned} \text{Confidence for } (X \rightarrow Y) &= \text{Support}(XY)/\text{Support}(X) \\ &= P(X \cap Y)/P(X) = P(Y|X) \end{aligned}$$

$P(Y|X)$ is the probability of Y once X has taken place, also called the conditional probability of Y

Sometimes the term lift is used to measure the power of associations between items that are purchased together

Lift must be above 1.0 for the association to be of interest

Lift may be defined as $P(Y|X)/P(Y)$

The task and the Naïve Algorithm

- Given a large set of transactions, we seek a procedure to discover all association rules which have at least $p\%$ support and $q\%$ confidence
- A Naïve Algorithm
- This is a naïve brute force algorithm
- Here considering four items for sale (Bread, Cheese, Juice, Milk)
- And only four transactions

Table 2.2 Transactions for Example 2.1

<i>Transaction ID</i>	<i>Items</i>
100	Bread, Cheese
200	Bread, Cheese, Juice
300	Bread, Milk
400	Cheese, Juice, Milk

action "database" in Table 2.2 are given in Table 2.3.

Table 2.3 The list of all itemsets and their frequencies

<i>Itemsets</i>	<i>Frequency</i>
Bread	3
Cheese	3
Juice	2
Milk	2
(Bread, Cheese)	2
(Bread, Juice)	1
(Bread, Milk)	1
(Cheese, Juice)	2
(Cheese, Milk)	1
(Juice, Milk)	1
(Bread, Cheese, Juice)	1
(Bread, Cheese, Milk)	0
(Bread, Juice, Milk)	0
(Cheese, Juice, Milk)	1
(Bread, Cheese, Juice, Milk)	0

We are interested in finding association rules with a minimum support of 50% and a minimum confidence of 75%

Given the required minimum support of 50%, we find the itemsets that occur in at least two transactions. Such itemsets are called frequent

All four items are frequent

The frequency goes down when we consider 2-itemsets, 3-itemsets, 4-itemsets

Table 2.4 The set of all frequent itemsets

<i>Itemsets</i>	<i>Frequency</i>
Bread	3
Cheese	3
Juice	2
Milk	2
Bread, Cheese	2
Cheese, Juice	2

- Every 2-itemset can lead to two rules.
- $A \rightarrow B$
- $B \rightarrow A$ if both satisfy the required confidence
- Confidence of $A \rightarrow B$ is given by the support for A and B together divided by the support for A

ore have four possible rules and their confidence as follows:

Bread \rightarrow Cheese with confidence of $2/3 = 67\%$

Cheese \rightarrow Bread with confidence of $2/3 = 67\%$

Cheese \rightarrow Juice with confidence of $2/3 = 67\%$

Juice \rightarrow Cheese with confidence of 100%

he last rule Juice \rightarrow Cheese has a confidence of 100%

- In the above example, only the last rule Juice → Cheese has confidence above 75%
- Rules that have more than the user-specified minimum confidence are called **confident**

Improved Naïve Algorithm

In this case we count only the combinations that actually occur. i.e., itemsets with zero frequency are not considered

Table 2.5 All possible combinations with nonzero frequencies

<i>Transaction ID</i>	<i>Items</i>	<i>Combinations</i>
100	Bread, Cheese	{Bread, Cheese}
200	Bread, Cheese, Juice	{Bread, Cheese}, {Bread, Juice}, {Cheese, Juice}, {Bread, Cheese, Juice}
300	Bread, Milk	{Bread, Milk}
400	Cheese, Juice, Milk	{Cheese, Juice}, {Cheese, Milk}, {Juice, Milk}, {Cheese, Juice, Milk}

- In this case only the following combinations has to be considered.

Table 2.6 Frequencies of all itemsets with nonzero frequencies

<i>Itemsets</i>	<i>Frequency</i>
Bread	3
Cheese	3
Juice	2
Milk	2
(Bread, Cheese)	2
(Bread, Juice)	1
(Bread, Milk)	1
(Cheese, Juice)	2
(Cheese, Milk)	1
(Juice, Milk)	1
(Bread, Cheese, Juice)	1
(Cheese, Juice, Milk)	1

The Apriori Algorithm

- This algorithm was proposed in 1994
- This algorithm has two parts
- In the first part, the itemsets that exceed the minimum support are found. --- Such itemsets are called frequent itemsets
- In the second part, the association rules that meet the minimum confidence requirement are found from the frequent itemsets.

First Part – Frequent Itemsets

For a given set of transactions

Step 1 : Find all frequent items that have support above $p\%$. Let these frequent items be L_1

Step 2 : Build potential sets of k items from L_{k-1} , such that each pair has the first $k-2$ items in common. Now the $k-2$ common items and the one remaining item from each of the two item sets are combined to form a k -itemset. The set of such potentially frequent k itemsets is the candidate set C_k . (For $k=2$, C_2 is built from L_1) . This step is called Apriori-gen.

Step 3 : Scan all transactions and find all k -itemsets in C_k that are frequent. The frequent set so obtained is L_k . (For $k=2$, C_2 is the set of candidate pairs, The frequent pairs are L_2)

Terminate when no further frequent itemsets are found. Otherwise continue with Step 2.

Some of the issues are :

1. Computing L_1 : This is computed only once, and the count for each item stored in memory is used
2. Apriori-gen function : This takes L_{k-1} and generates all candidate k -itemsets. In computing C_3 from L_2 , the itemsets are sorted in their lexicographic order .
If an itemset in C_3 is (a,b,c) then L_2 must have items (a,b) and (a,c)
When L_{i-1} have the same first $i-2$ items, we can combine them to produce a candidate itemset for C_i
3. Pruning : Once C_i is produced, we can prune some of the candidate itemsets by checking that all subsets of every itemset are frequent. For ex., if we derived $\{a,b,c\}$ from $\{a,b\}$ and $\{a,c\}$, then we check that $\{b,c\}$ is also in L_2 . If not, $\{a,b,c\}$ may be removed from C_3 .
4. Apriori subset function : C_k are stored as a hash tree
5. Transactions Storage
6. Computing L_2 : When C_2 is available, the frequent candidate pair is identified.

Second Part – Finding the Rules

- To find the association rules from the frequent itemsets, we take a large frequent itemset , p , and find each nonempty subset a .
- The rule $a \rightarrow (p-a)$ is possible if it satisfies the confidence
- Since confidence is given by $\text{support}(p)/\text{support}(a)$,
- If for some a , the rule $a \rightarrow (p-a)$ does not have minimum confidence, then all rules like $b \rightarrow (p-b)$ where b is a subset of a , will also not have the confidence, since $\text{support}(b)$ cannot be smaller than $\text{support}(a)$

- When the rule $a \rightarrow (p-a)$ does not have minimum confidence, then smaller subsets need not be checked.
- For ex., if $ABC \rightarrow D$ does not have minimum confidence then $AB \rightarrow CD$ or $A \rightarrow BCD$ also will not have confidence

- Another method is :

- Consider rules like $(p-a) \rightarrow a$

If this rule has minimum confidence the all rules $(p-b) \rightarrow b$

Will also have minimum confidence if b is a subset of a , since $(p-b)$ has more items than $(p-a)$, given that b is smaller than a and cannot have support higher than that of $(p-a)$

Implementation Issue – Transaction Storage

- Let there be 6 items. Let them be {A,B,C,D,E}
- Let there be 8 transactions with transaction IDs {10, 20, 30, 40, 50, 60, 70, 80}
- These transactions can be represented in three different ways.

Table 2.7 A simple representation of transactions as an item list

<i>Transaction ID</i>	<i>Items</i>
10	A, B, D
20	D, E, F
30	A, F
40	B, C, D
50	E, F
60	D, E, F
70	C, D, F
80	A, C, D, F

Table 2.8 Representing transactions as a binary item list

<i>TID</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
10	1	1	0	1	0	0
20	0	0	0	1	1	1
30	1	0	0	0	0	1
40	0	1	1	1	0	0
50	0	0	0	0	1	1
60	0	0	0	1	1	1
70	0	0	1	1	0	1
80	1	0	1	1	0	1

Table 2.9 Representing transactions as binary columns

<i>Item</i>	<i>TID</i>							
	10	20	30	40	50	60	70	80
<i>A</i>	1	0	1	0	0	0	0	1
<i>B</i>	1	0	0	1	0	0	0	0
<i>C</i>	0	0	0	1	0	0	1	1
<i>D</i>	1	1	0	1	0	1	1	1
<i>E</i>	0	1	0	0	1	1	0	0
<i>F</i>	0	1	1	0	1	1	1	1

A Simple Apriori Example

Table 2.10 Transactions for Example 2.2

<i>Transaction ID</i>	<i>Items</i>
100	Bread, Cheese, Eggs, Juice
200	Bread, Cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	Cheese, Juice, Milk

Table 2.11 Frequent items L_1 for Example 2.2

<i>Item</i>	<i>Frequency</i>
Bread	4
Cheese	3
Juice	4
Milk	3

Table 2.12 Candidate item pairs C_2 for Example 2.2

<i>Item pairs</i>	<i>Frequency</i>
(Bread, Cheese)	2
(Bread, Juice)	3
(Bread, Milk)	2
(Cheese, Juice)	3
(Cheese, Milk)	1
(Juice, Milk)	2

From the above table, we have only two frequent item pairs which are {Bread, Juice} and {Cheese, Juice}

This is L_2 .

From this, a candidate 3-itemset cannot be obtained since we do not have two 2-itemsets that have the same first item

The rules derived from the two frequent 2-itemsets are

Bread \rightarrow Juice

Juice \rightarrow Bread

Cheese \rightarrow Juice

Juice \rightarrow Cheese

- The confidence of the above four rules are $\frac{3}{4} = 75\%$, $\frac{3}{4} = 75\%$, $\frac{3}{3} = 100\%$, $\frac{3}{4} = 75\%$ respectively. Since all of them have a minimum 75%, they all qualify.

A Larger Apriori Example

- Consider a store having 16 items for sale
- Consider 25 transactions.
- In this example, we want to find the association rules that satisfy 25% support and 70% confidence

Table 2.13 The list of grocery items for Example 2.3

<i>Item number</i>	<i>Item name</i>
1	Biscuits
2	Bread
3	Cereal
4	Cheese
5	Chocolate
6	Coffee
7	Donuts
8	Eggs
9	Juice
10	Milk
11	Newspaper
12	Pastry
13	Rolls
14	Sugar
15	Tea
16	Yogurt

Transaction Storage

- The below table shows 25 transactions

howing
t to use item numbers.

Table 2.14 Transaction data for Example 2.3

<i>TID</i>	<i>Items</i>
1	Biscuits, Bread, Cheese, Coffee, Yogurt
2	Bread, Cereal, Cheese, Coffee
3	Cheese, Chocolate, Donuts, Juice, Milk
4	Bread, Cheese, Coffee, Cereal, Juice
5	Bread, Cereal, Chocolate, Donuts, Juice
6	Milk, Tea
7	Biscuits, Bread, Cheese, Coffee, Milk
8	Eggs, Milk, Tea
9	Bread, Cereal, Cheese, Chocolate, Coffee
10	Bread, Cereal, Chocolate, Donuts, Juice

(Contd.)

Table 2.14 Transaction data for Example 2.3 (Contd.)

<i>TID</i>	<i>Items</i>
11	Bread, Cheese, Juice
12	Bread, Cheese, Coffee, Donuts, Juice
13	Biscuits, Bread, Cereal
14	Cereal, Cheese, Chocolate, Donuts, Juice
15	Chocolate, Coffee
16	Donuts
17	Donuts, Eggs, Juice
18	Biscuits, Bread, Cheese, Coffee
19	Bread, Cereal, Chocolate, Donuts, Juice
20	Cheese, Chocolate, Donuts, Juice
21	Milk, Tea, Yogurt
22	Bread, Cereal, Cheese, Coffee
23	Chocolate, Donuts, Juice, Milk, Newspaper
24	Newspaper, Pastry, Rolls
25	Rolls, Sugar, Tea

Computing L1

- To find the frequent set L1, we count the number of times each of the 16 items has been sold

Table 2.15 Frequency count for all items

<i>Item no.</i>	<i>Item name</i>	<i>Frequency</i>
1	Biscuits	4
2	Bread	13
3	Cereal	10
4	Cheese	11
5	Chocolate	9
6	Coffee	9
7	Donuts	10
8	Eggs	2
9	Juice	11
10	Milk	6
11	Newspaper	2
12	Pastry	1
13	Rolls	2
14	Sugar	1
15	Tea	4
16	Yogurt	2

- The items that have the necessary support (25% support in 25 transactions) must occur in at least seven transactions.

Table 2.16 The frequent 1-itemset or L_1

<i>Item</i>	<i>Frequency</i>
Bread	13
Cereal	10
Cheese	11
Chocolate	9
Coffee	9
Donuts	10
Juice	11

the number of frequent pairs is 21. The number of frequent pairs is written as $L_1 X L_1$ where X is a special concatenation operation. These are the only ones that could be frequent with the number of frequent pairs is 21.

Table 2.17 The 21 candidate 2-itemsets or C_2

-
- {Bread, Cereal}
 - {Bread, Cheese}
 - {Bread, Chocolate}
 - {Bread, Coffee}
 - {Bread, Donuts}
 - {Bread, Juice}
 - {Cereal, Cheese}
 - {Cereal, Chocolate}
 - {Cereal, Coffee}
 - {Cereal, Donuts}
 - {Cereal, Juice}
 - {Cheese, Chocolate}
 - {Cheese, Coffee}
 - {Cheese, Donuts}
 - {Cheese, Juice}
 - {Chocolate, Coffee}
 - {Chocolate, Donuts}
 - {Chocolate, Juice}
 - {Coffee, Donuts}
 - {Coffee, Juice}
 - {Donuts, Juice}
-

Table 2.18 Frequency count of candidate 2-itemsets

<i>Candidate 2-itemset</i>	<i>Frequency</i>
{Bread, Cereal}	9
{Bread, Cheese}	8
{Bread, Chocolate}	4
{Bread, Coffee}	8
{Bread, Donuts}	4
{Bread, Juice}	6
{Cereal, Cheese}	5
{Cereal, Chocolate}	4
{Cereal, Coffee}	5
{Cereal, Donuts}	4
{Cereal, Juice}	6
{Cheese, Chocolate}	4
{Cheese, Coffee}	9
{Cheese, Donuts}	3
{Cheese, Juice}	4
{Chocolate, Coffee}	1
{Chocolate, Donuts}	7
{Chocolate, Juice}	7
{Coffee, Donuts}	1
{Coffee, Juice}	2
{Donuts, Juice}	9

Table 2.19 The frequent 2-itemsets or L_2

<i>Frequent 2-itemset</i>	<i>Frequency</i>
{Bread, Cereal}	9
{Bread, Cheese}	8
{Bread, Coffee}	8
{Cheese, Coffee}	9
{Chocolate, Donuts}	7
{Chocolate, Juice}	7
{Donuts, Juice}	9

Zoom Out

frequencies.

Table 2.20 Candidate 3-itemsets or C_3 and their frequencies

<i>Candidate 3-itemset</i>	<i>Frequency</i>
{Bread, Cereal, Cheese}	4
{Bread, Cereal, Coffee}	4
{Bread, Cheese, Coffee}	8
{Chocolate, Donuts, Juice}	7

presents the frequent 3-itemsets or L_3 . Note that only one member is present in L_3 . That 2-itemset is {Bread, Cereal}.

Table 2.21 The frequent 3-itemsets or L_3

<i>Frequent 3-itemset</i>	<i>Frequency</i>
{Bread, Cheese, Coffee}	8
{Chocolate, Donuts, Juice}	7

f frequent itemset generation

Finding the Rules

- We know that

$$\text{Confidence}(A \rightarrow B) = P(B|A) = P(A \cap B) / P(A)$$

We write {Bread, Cheese, Coffee} as {B,C,D}

And obtain a rule $B \rightarrow CD$ which may be written as $B \rightarrow C$
and $B \rightarrow D$

A 3-itemset will result in 6 possible rules: 3 rules with one item on the left side and 3 rules with one item on the right side

i.e., $B \rightarrow CD$

$C \rightarrow BD,$

$D \rightarrow BC$

three rules with one item on right side are:

$$CD \rightarrow B$$

$$BD \rightarrow C$$

$$BC \rightarrow D$$

The support for all these rules is 8. We need to compute (Table 2.22) the frequencies of the right side itemsets to compute confidence.

Table 2.22 Confidence of association rules from {Bread, Cheese, Coffee}

<i>Rule</i>	<i>Support of BCD</i>	<i>Frequency of LHS</i>	<i>Confidence</i>
$B \rightarrow CD$	8	13	0.61
$C \rightarrow BD$	8	11	0.72
$D \rightarrow BC$	8	9	0.89
$CD \rightarrow B$	8	9	0.89
$BD \rightarrow C$	8	8	1.0
$BC \rightarrow D$	8	8	1.0

Confidence of all these rules, except the first one, is higher than 70% and so five association rules are generated. Similarly, looking at the other 3-itemset {Chocolate, Donuts, Juice} and writing down all possible rules, we find values as given in Table 2.23.

Table 2.23 Confidence of association rules from {Chocolate, Donuts, Juice}

<i>Rule</i>	<i>Support</i>	<i>Frequency of LHS</i>	<i>Confidence</i>
$N \rightarrow MP$	7	9	0.78
$M \rightarrow NP$	7	10	0.70
$P \rightarrow NM$	7	11	0.64
$MP \rightarrow N$	7	9	0.78
$NP \rightarrow M$	7	7	1.0
$NM \rightarrow P$	7	7	1.0

Confidence of all these rules, except the first one, is higher than 70% and so five

all the rules is now given in Table 2.24

Table 2.24 All association rules for Example 2.3

Cheese \rightarrow Bread
Cheese \rightarrow Coffee
Coffee \rightarrow Bread
Coffee \rightarrow Cheese
Cheese, Coffee \rightarrow Bread
Bread, Coffee \rightarrow Cheese
Bread, Cheese \rightarrow Coffee
Chocolate \rightarrow Donuts
Chocolate \rightarrow Juice
Donuts \rightarrow Chocolate
Donuts \rightarrow Juice
Donuts, Juice \rightarrow Chocolate
Chocolate, Juice \rightarrow Donuts
Chocolate, Donuts \rightarrow Juice
Bread \rightarrow Cereal
Cereal \rightarrow Bread

Comments about the Apriori Algorithm and further Terminology

- In the above examples, we required support of 25% and 50%. In real time transactions, many items will not qualify. So practical examples deals with much smaller support, even 1% or lower.
- A and B are related events, they are not independent. So we cannot use probability theory for independent events.
- Closed and Maximal itemsets : A frequent closed itemset is a frequent itemset X such that there exists no superset of X with the same support count as X .

- A frequent itemset Y is maximal if it is not a proper subset of any other frequent itemset. Therefore, a maximal itemset is a closed itemset, but a closed itemset is not necessarily a maximal itemset.

For ex., $\{B,D\}$ and $\{B,C,D\}$ had the same support of 8

While $\{C,D\}$ had a support of 9. Therefore $\{C,D\}$ is a closed itemset but not maximal.

On the other hand, $\{\text{Bread, Cereal}\}$ was frequent, but no superset of the two itemsets is frequent. So, this is both maximal and closed.

- Frequent maximal itemsets : The frequent maximal itemsets uniquely determine all frequent itemsets. Therefore, the aim of any association rule algorithm is to find all maximal frequent itemsets.

Example 2.4—Closed and Maximal Itemsets

We consider an example to further demonstrate the concepts of closed frequent itemsets and maximal frequent itemsets.

Consider the transaction database in Table 2.25 with minimum support required to be 40%.

Table 2.25 A transaction database to illustrate closed and maximal itemsets

100	Bread, Cheese, Juice
200	Bread, Cheese, Juice, Milk
300	Cheese, Juice, Egg
400	Bread, Juice, Milk, Egg
500	Milk, Egg

All the frequent itemsets from the transaction database in Table 2.25 are given in Table 2.26.

Table 2.26 Frequent itemsets for the database in Table 2.25

Itemset	Support	Closed?	Maximal?	Both?
{Bread}	3	No	No	No
{Cheese}	3	No	No	No
{Juice}	4	Yes	No	No
{Milk}	3	Yes	No	No
{Egg}	3	Yes	No	No
{Bread, Cheese}	2	No	No	No
{Bread, Juice}	3	Yes	No	No
{Bread, Milk}	2	No	No	No
{Cheese, Juice}	3	Yes	No	No
{Juice, Milk}	2	No	No	No
{Juice, Egg}	2	Yes	Yes	Yes
{Milk, Egg}	2	Yes	Yes	Yes
{Bread, Cheese, Juice}	2	Yes	Yes	Yes
{Bread, Juice, Milk}	2	Yes	Yes	Yes

... the closed and maximal itemsets among them have been identified

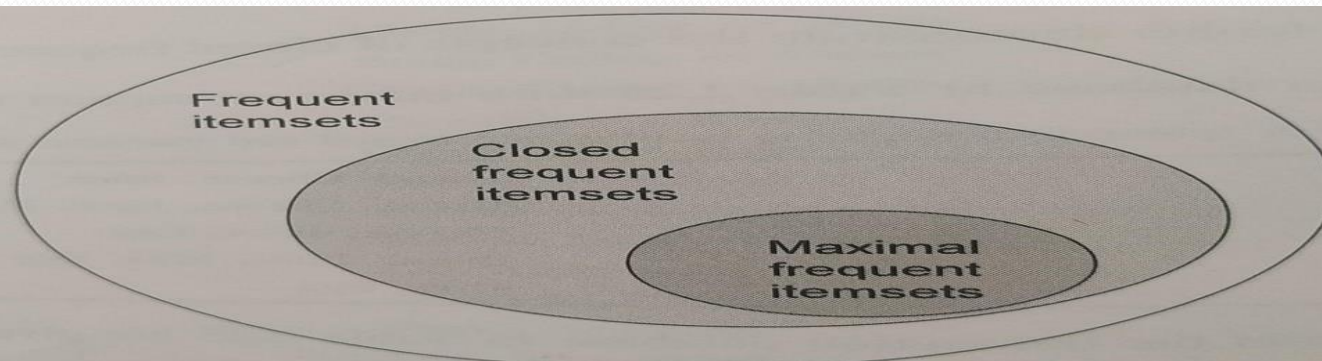
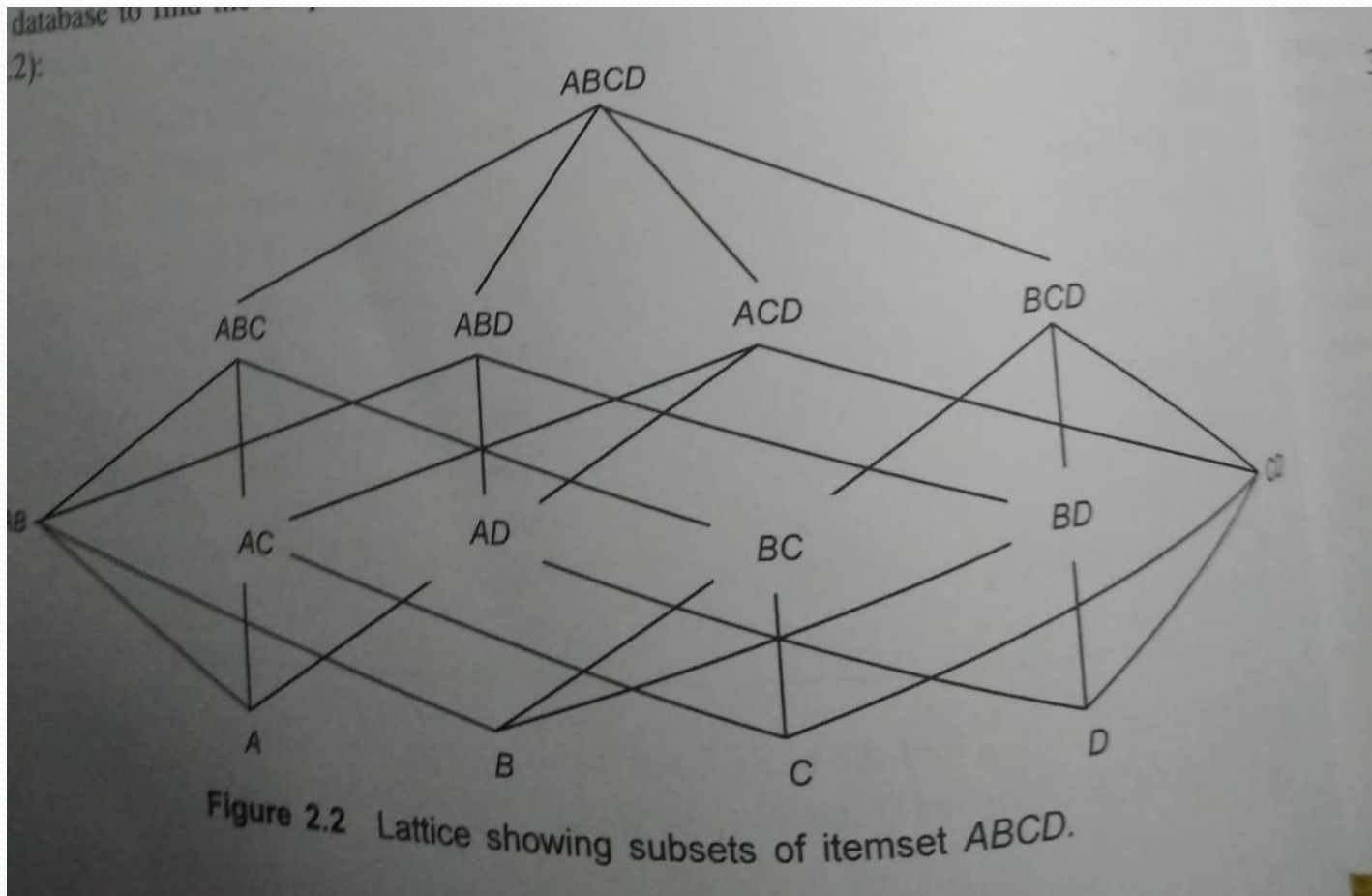


Figure 2.1. Frequent, Closed, and Maximal Itemsets

Lattice showing subsets of itemset ABCD



Improving the efficiency of the Apriori Algorithm

- The no. of candidate sets may be huge. The larger the candidate set, the higher the processing cost for scanning the transaction database to find the frequent itemsets. But the no. of 3-itemsets will be smaller
- The Apriori algorithm requires many scans of the database. If n is the length of the longest itemset, then $(n+1)$ scans are required.
- Many trivial rules are derived and it can be difficult to extract the most interesting rules from all the rules derived.

- Some rules can be inexplicable and very fine grained. For ex., toothbrush was the most frequently sold item on Thursday morning
- Redundant rules are generated for ex., if $A \rightarrow B$ is a rule then $AC \rightarrow B$ is redundant
- The no. of items in each transaction is small when compared to total items. This algorithm works better with sparsity. Some applications produce dense data (i.e., many items in each transaction) which may also have many frequently occurring items. Apriori is likely to be inefficient for such applications.



Techniques for improving the performance of the a priori algorithm are :

1 reduce the number of candidate itemsets

2 reduce the number of transactions

3 reduce the number of comparisons

4 generate candidates efficiently

Mining frequent patterns without candidate generation

Mining the FP-tree for frequent items

The FP growth algorithm does not generate the candidates, it only tests. In contrast, the apriori algorithm generates the candidate itemsets and then tests.

The motivation for the FP tree method are

- 1 only the frequent items are needed to find the association rules
- 2 if the frequent items are stored in a compact structure, then the original transaction database need not be used repeatedly
- 3 If multiple transactions share a set of frequent items, the shared sets may be merged with the number of occurrences registered as count

Generating FP trees

The algorithm works as follows:

- 1 Scan the transaction database once to find all the frequent items and their support
- 2 Sort the frequent items in descending order of their support
- 3 Initially start creating the FP tree with the root “null”
- 4 Get the first transaction from the transaction database. Remove all non frequent items and list the remaining items according to the order in the sorted frequent items
- 5 Use the transaction to construct the first branch of the tree with each node corresponding to a frequent item and showing that item’s frequency, which is 1 for the first transaction
- 6 Get the next transaction from the transaction database. Remove all non frequent items and list the remaining items according to the order in the sorted frequent items
- 7 Insert the transaction in the tree using any common prefix that may appear. Increase the item counts
- 8 Continue with step 6 until all transactions in the database are processed.

6. Get the next transaction from the transaction database. Remove all non-frequent items from the list the remaining items according to the order in the sorted frequent items.
7. Insert the transaction in the tree using any common prefix that may appear. Increase counts.
8. Continue with Step 6 until all transactions in the database are processed.

We now illustrate the process of building FP-trees by means of an example.

Example 2.7—FP-Tree

We again use the transaction database of Example 2.2 as shown in Table 2.36. The minimum support required is again 50% and confidence is 75%.

Table 2.36 Transaction database for Example 2.7

Transaction ID	Items
100	Bread, Cheese, Eggs, Juice
200	Bread, Cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	Cheese, Juice, Milk

The frequent items sorted by their frequency are shown in Table 2.37.

Table 2.37 Frequent items for database in Table 2.36

Item	Frequency
Bread	4
Juice	4
Cheese	3
Milk	3

Now we remove the items that are not frequent from the transactions and order the items according to their frequency as above (Table 2.38).

Table 2.38 Database after removing the non-frequent items and reordering

Transaction ID	Items
100	Bread, Juice, Cheese
200	Bread, Juice, Cheese
300	Bread, Milk
400	Bread, Juice, Milk
500	Juice, Cheese, Milk

In a large transaction database, Table 2.37 will be much smaller than Table 2.36.

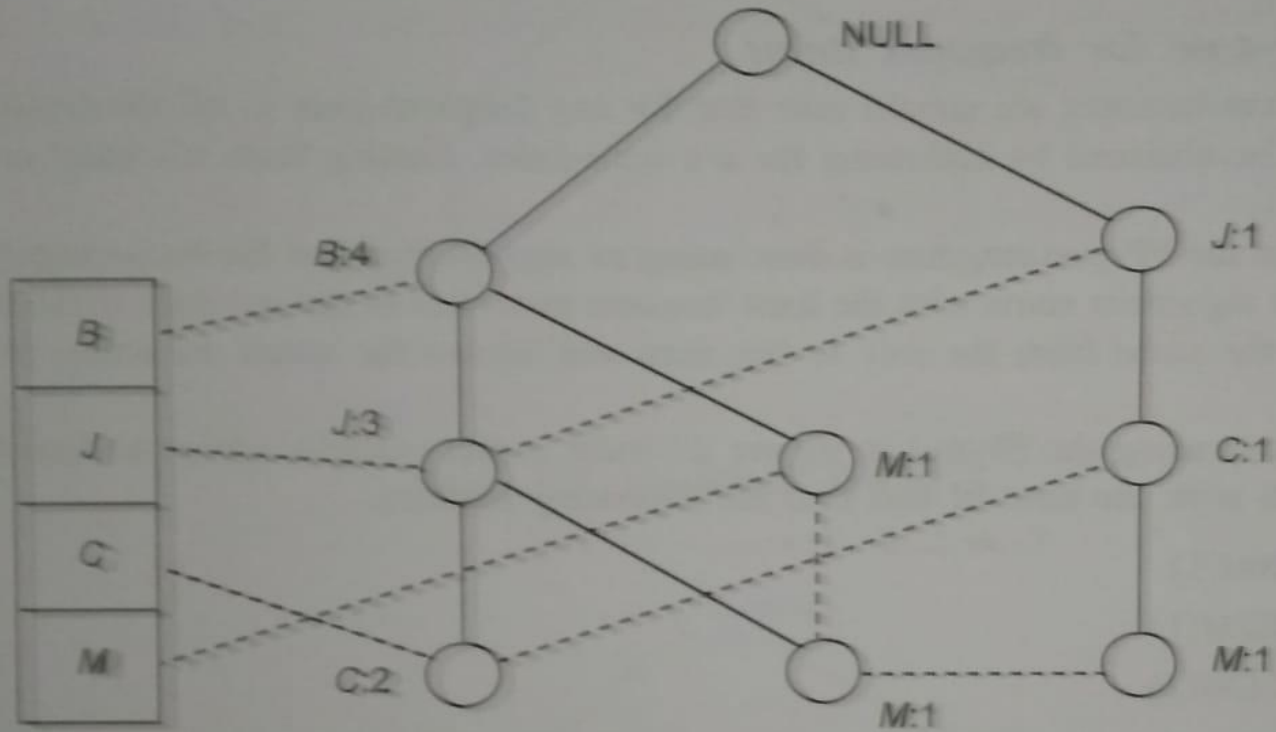


Figure 2.3 FP-tree for Example 2.7.

Mining the FP-tree for frequent items

- For any frequent item a , all the frequent itemsets containing a can be obtained by following the a 's node links, starting from a 's head in the FP-tree header
- The mining on the FP-tree structure is done using the frequent pattern growth algorithm (FP-growth)
 - This algorithm starts with the least frequent (last item) item in the header table. Then it finds all the paths from the root to this item and adjusts the count according to the item's support count.

- We start with the item M and find the following patterns:

BM(1)

BJM(1)

JCM(1)

There are no frequent itemsets in this list since no item appears 3 or more times

- Next we look at C and find the following :

- BJC(2)

- JC(1)

- These two patterns give us a frequent itemset JC(3)

- Looking at J, the next frequent item in the table, we obtain

BJ(3)

J(1)

Again we obtain a frequent itemset BJ(3)

The process above may be represented by the conditional trees for M, C and J

may be represented
ely.

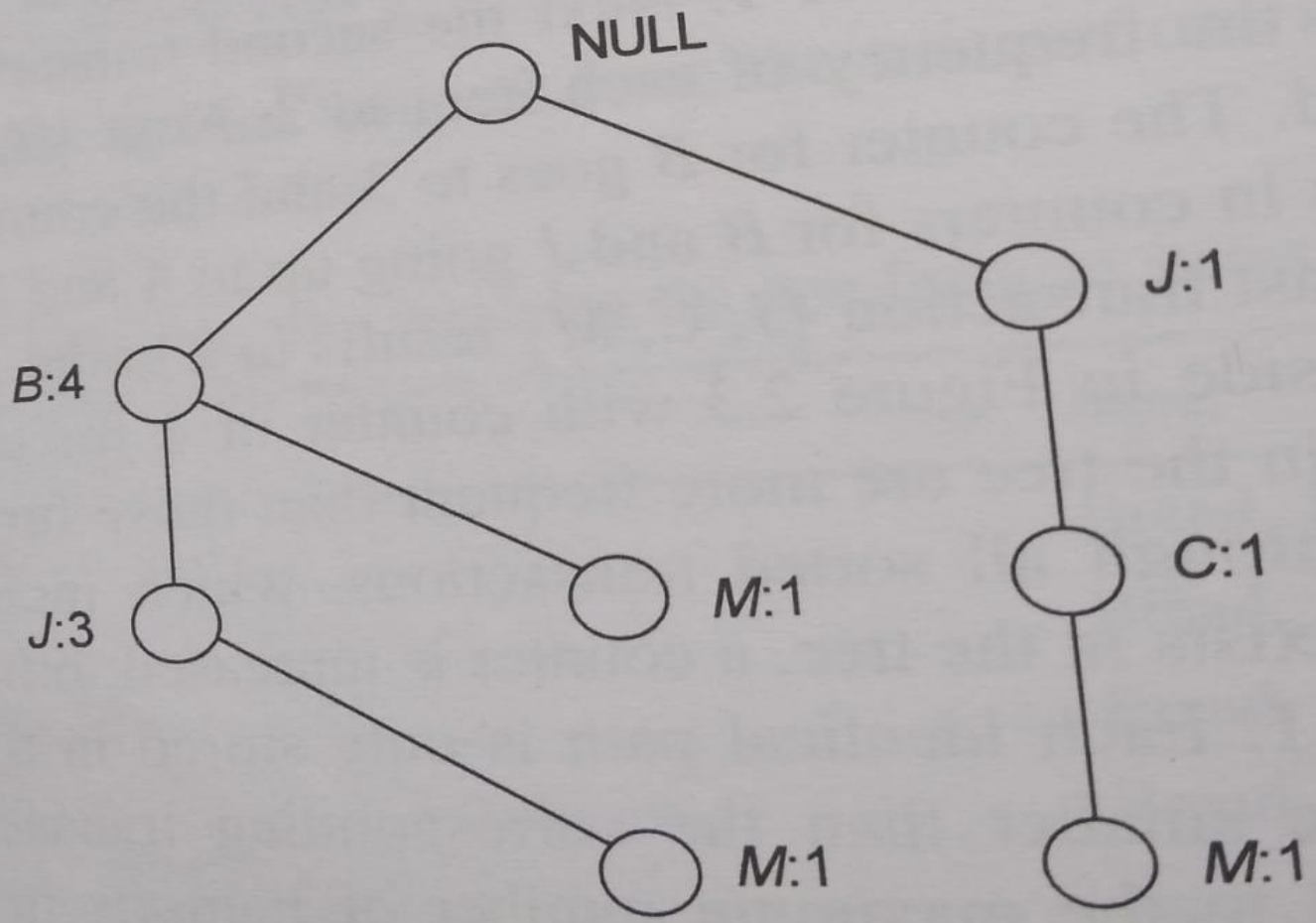


Figure 2.4 Conditional tree for *M*.

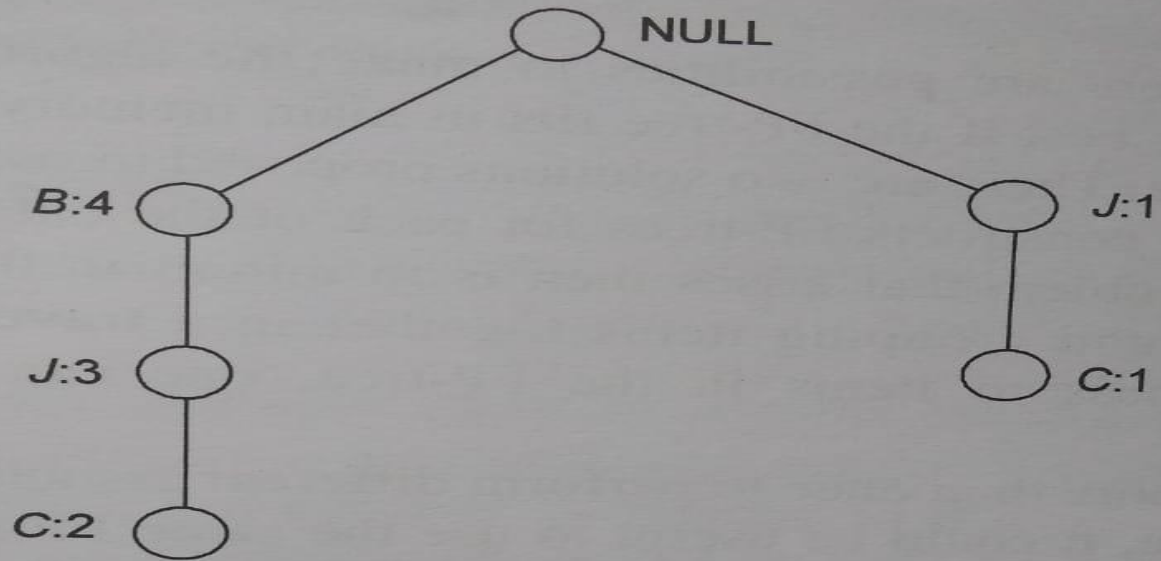


Figure 2.5 Conditional tree for C.

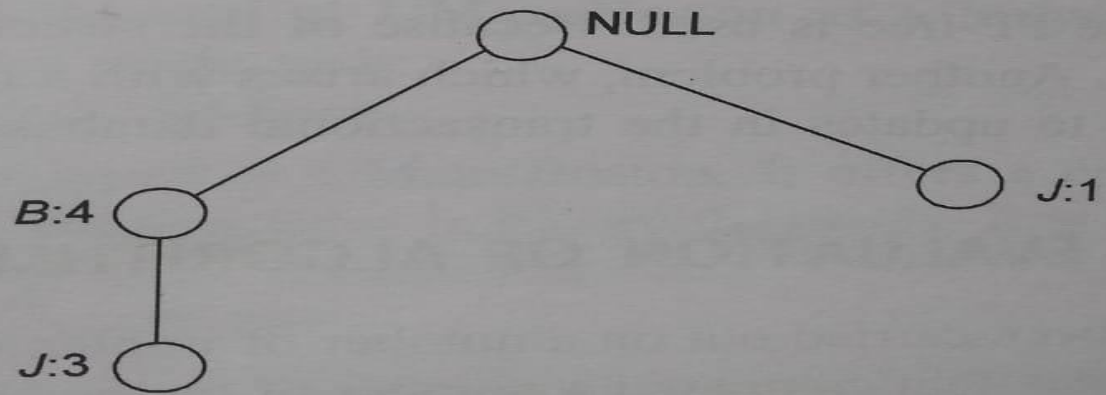


Figure 2.6 Conditional tree for J.

- The algorithm finds the conditional pattern base for each item
- The conditional pattern base of an item consists of all patterns leading to the specific item
- Then the conditional frequent pattern tree is constructed from the conditional pattern base where only the frequent items are included.
- This conditional FP-tree also contains a header table.
- Then the algorithm is called again with this new conditional FP-tree. This recursion is ended when the tree is empty .

Advantages of the FP-tree approach

- It avoids scanning the database more than twice to find the support counts.
- It completely eliminates the candidate set generation
- This algorithm is better than Apriori algorithm when the transaction database is huge and the support count is low.
- FP-growth uses a more efficient structure to mine patterns when the database grows.
- Because of the structure of the FP-tree, it is easy to have different support counts

Performance Evaluation of Algorithms

- Comparing Apriori, CHARM, FP-growth methods,
- The FP-growth method was better than the best implementation of the Apriori algorithm
 - CHARM was better than Apriori. In some cases, CHARM was better than FP-growth method
 - Apriori was generally better than other algorithms if the support required was high
 - At very low support, the number of frequent items became large, and none of the algorithms were able to handle large frequent sets gracefully.