9

Association Mining

Chapter Objectives

- ✓ To comprehend the concept of association mining and its applications
- ✓ To understand the role of support, confidence and lift
- ✓ To comprehend the Naïve algorithm, Its limitations and improvements
- ✓ To learn about approaches for transaction database storage
- ✓ To understand and demonstrate the use of the apriori algorithm and the direct hashing and pruning algorithm
- ✓ To use dynamic Itemset counting to identify association rules
- ✓ To use FP growth for mining frequent patterns without candidate generation

9.1 Introduction to Association Rule Mining

Association rule mining often known as 'market basket' analysis is very effective technique to find the association of sale of item X with item Y. In simple words, market basket analysis consists of examining the items in the baskets of shoppers checking out at a market to see what types of items 'go together' as illustrated in Figure 9.1.

It would be useful to know, when people make a trip to the store, what kind of items do they tend to buy during that same shopping trip? For example, as shown in Figure 9.1, a database of customer transactions (i.e., shopping baskets) is given where each transaction consists of a set of items (i.e., products purchased during a visit). Association rule mining is used to identify groups of items which are frequently purchased together (customers' purchasing behavior). For example, 'IF one buys bread and milk, THEN he/she also buys eggs with high probability.' This information is useful for the store manager for better planning of stocking items in the store to improve its sale and efficiency.

Let us suppose that the store manager, receives customer complaints about heavy rush in his store and its consequent slow working. He may then decide to place associated items such as bread and

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

milk together, so that customers can buy the items easier and faster than if these were at a distance. It also improves the sale of each product. In another scenario, let us suppose his store is new and the store manager wishes to display all its range of products to prospective customers. He may then decide to put the associated items, i.e., bread and milk, at opposite ends of the store and would place other new items and items being promoted in between them, thereby ensuring that many customers, on the way from bread to milk, would take notice of these and some would end up buying them.

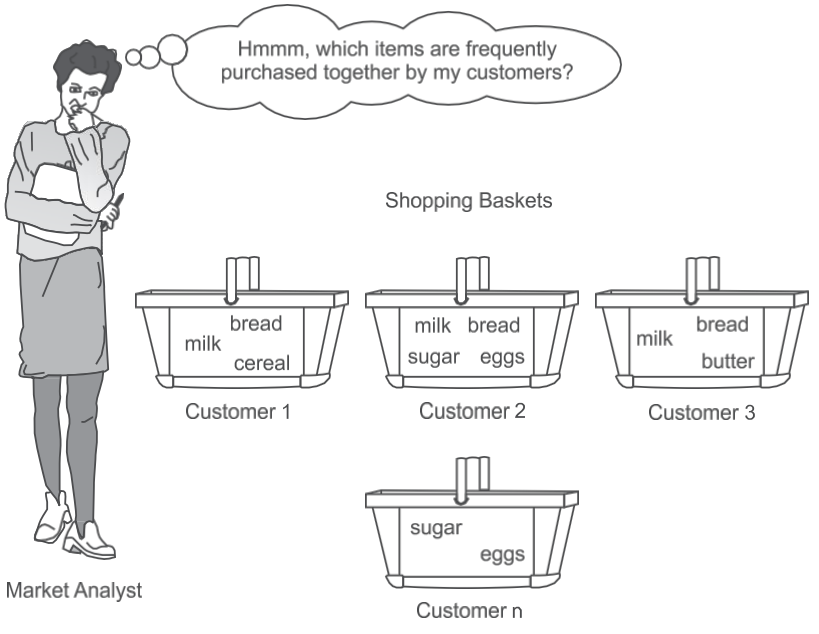


Figure 9.1 Need for association mining

Sometimes analysis of sale records leads to very interesting and unexpected results. In one very popular case study by Walmart USA, they had identified that people buying diapers often also bought beer. By putting the beer next to the diapers in their stores, it was found that the sales of each skyrocketed as depicted in Figures 9.2, 9.3 and 9.4.

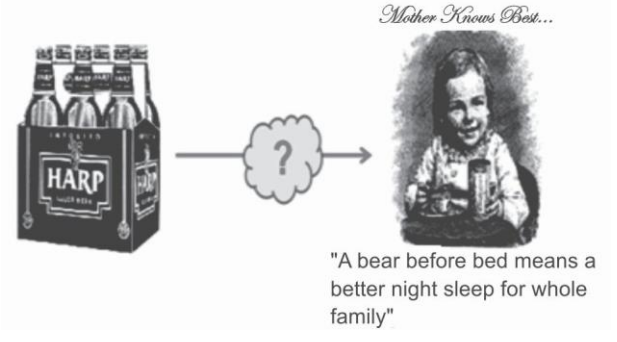


Figure 9.2 Association of sale of beer and diapers

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

One may make multiple conclusions about the association of bear and diapers. It may be funny to assume that 'A beer before bed means a better night's sleep for the whole family.' But serious analysis could lead to 'Young couples prepare for the weekend by stocking up on diapers for the infants and beer for dad'.

Figure 9.3 Association of sale of beer and diapers

Nevertheless, this story aptly illustrates the market basket concept of things 'going together', which is the basis of the data mining association function.

Figure 9.4 Association of sale of beer and diapers

Some other interesting findings of association mining may be trivial, for example, 'Customers who purchase maintenance agreements are very likely to purchase large appliances.' Or it may be unexplicable and unexpected, as in 'When a new restaurant opens, one of the most sold items is coffee'. This may be due to the reason that coffee is the cheapest item in the menu. So, to get the

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

feel of the restaurant, people only order coffee. This type of finding suggests to the manager to increase the cost of coffee. It happens to be the reason why in many good restaurants, the price of coffee is alarming high.

Often, while conducting market basket analysis or association rule mining, the only source of data available on customers is the bills of purchase, which tells what items go together in a shopping cart, and can also show what items 'go together' at certain times of the day, days of the week, seasons of the year, credit versus cash versus check payment, geographical locations of stores, and so on. Discovering associations among these attributes can lead to fact-based marketing strategies for things like store f loor plans, special discounts, coupon offerings, product clustering, catalog design, identifying items that need to be put in combo packs and can be used to compare stores as shown in Figure 9.5.

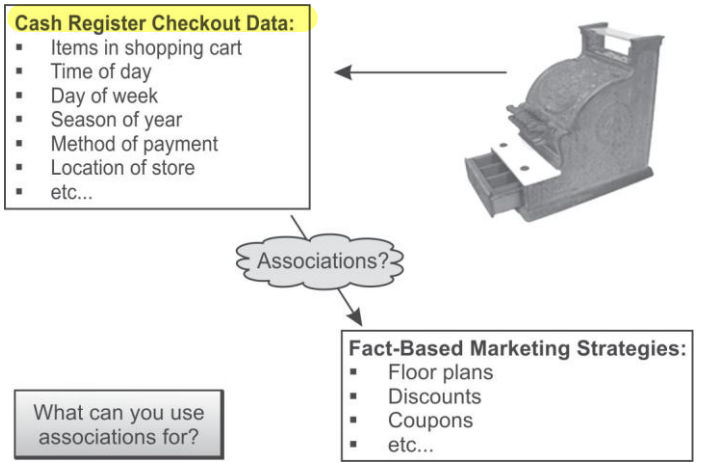


Figure 9.5 Association and customer purchase bills

Nowadays, recommendations given by online stores like Amazon and Flipkart also make use of association mining to recommend products related with your purchase and the system offers the list of products that others often buy together with the product you have just purchased. Besides the examples from market basket analysis given above, association rules are used today in many application areas such as intrusion detection, Web usage mining, bioinformatics and continuous production. Programmers use association rules to build programs capable of machine learning.

This is commonly known as market basket data analysis.

Association rule mining can also be used in applications like marketing, customer segmentation, web mining, medicine, adaptive learning, finance and bioinformatics, etc.

9.2 Defining Association Rule Mining

Association rule mining can be defined as identification of frequent patterns, correlations, associations, or causal structures among sets of objects or items in transactional databases, relational databases, and other information repositories.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Association rules are generally ***if/then*** statements that help in discovering relationships between seemingly unrelated data in a relational database or other information repository. For example, 'If a customer buys a dozen eggs, he is 80% likely to also purchase milk.'

An association rule consists of two parts, i.e., an antecedent (if) and a consequent (then). An antecedent is an object or item found in the data while a consequent is an object or item found in the combination with the antecedent.

Association rules are often written as X → Y meaning that whenever X appears Y also tends to appear. X and Y may be single items or sets of items. Here, X is referred to as the rule's antecedent and Y as its consequent.

For example, the rule found in the sales data of a supermarket could specify that if a customer buys onions and potatoes together, he or she will also like to buy burgers. This rule will be represented as onions, potatoes → burger.

The concept of association rules was formulated by two Indian scientists Dr Rakesh Agrawal and Dr R. Srikant. The detail about their brief profile is given in Section 9.7.1.

9.3 Representations of Items for Association Mining

Let us assume that the number of items in the shop stocks is n. In Table 9.1, there are 6 items in stock, namely, bread, milk, diapers, beer, eggs, cola, thus n = 6 for this shop.

The item list is represented by I and its items are represented by {i1,i2,…in}.

The number of transactions are represented by N transactions, i.e., N = 5 for the shop data as given in Table 9.1.

Table 9.1 Sale database

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}

Each transaction is denoted by T {t1, t2, ...,tN} each with a unique identifier (TID) and each transaction consists of a subset of items (possibly a small subset) purchased by one customer.

Let each transaction of m items be {i1,i2, …, im}, where m <= n [number of items in a transaction should be less than or equal to total items in the shop]. Typically, transactions differ in the number of items.

As shown in Table 9.1, the first transaction is represented as T1 and it has two items, i.e., m = 2, with i1 = Bread and i2 = Milk. Similarly transaction T4, has four items, having m = 4, with i1 = Bread, i2 = Milk, i3 = Diapers and i4 = Beer.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Our task will be to find association relationships, given a large number of transactions, such that items that tend to occur together are identified. It is important to note that association rules mining does not take into account the quantities of items bought.

9.4 The Metrics to Evaluate the Strength of Association Rules

The metrics to judge the strength and accuracy of the rule are as follows:

- Support
- Confidence
- Lift

9.4.1 Support

Let N is the total number of transactions. Support of X is represented as the number of times X appears in the database divided by N, while the support for X and Y together is represented as the number of times they appear together divided by N as given below.

Support(X) = (Number of times X appears) / N = P(X)

Support(XY) = (Number of times X and Y appear together) / N = P(X ∩ Y)

Thus, Support of X is the probability of X while the support of XY is the probability of X ∩ Y.

Table 9.1, has been reproduced as Table 9.2 to calculate the supports for each item as given below:

Table 9.2 Sale database

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}

Support(Bread) = Number of times Bread appears / total number of translations = 4/5 = P(Bread)

Support(Milk) = Number of times Milk appears / total number of translations = 4/5 = P(Milk)

Support(Diapers) = Number of times Diapers appears / total number of translations = 4/5 = P(Diapers)

Support(Beer) = Number of times Beer appears / total number of translations = 3/5 = P(Beer)

Support(Eggs) = Number of times Eggs appears / total number of translations = 1/5 = P(Eggs)

Support(Cola) = Number of times Cola appears / total number of translations = 2/5 = P(Cola)

Support(Bread, Milk) = Number of times Bread, Milk appear together / total number of translations = 3/5 = P(Bread ∩ Milk)

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Support(Diapers, Beer) = Number of times Diapers, Beer appears together / total number of translations = 3/5 = P(Diapers ∩ Beer)

A high level of support indicates that the rule is frequent enough for the business to take interest in it.

Support is very important metric because if a rule has low support then it may be the case that the rule occurs by chance and it will not be logical to promote items that customers seldom buy together. But if a rule has high support then that association becomes very important and if implemented properly will result in increase in revenue, efficiency and customer satisfaction.

9.4.2 Confidence

To understand the concept of confidence, let us suppose that support for X→Y is 80%, then it means that X→Y is very frequent and there are 80% chances that X and Y will appear together in a transaction. This would be of interest to the sales manager.

Let us suppose we have another pairs of items (A and B) and support for A→B is 50%.

Of course it is not as frequent as X→Y, but if this was higher, such as whenever A appears there is 90% chance that B also appears, then of course it would be of great interest.

Thus, not only the probability that A and B appear together matters, but also the conditional probability of B when A has already occurred plays a significant role. This conditional probability that B will follow when A has already been occurred is considered during determining the confidence of the rule.

Thus, Support and Confidence are important metrics to judge the quality of the association mining rule.

A high level of confidence shows that the rule is true often enough to justify a decision based on it.

Confidence for X→Y is defined as the ratio of the support for X and Y together to the support for X (which is same as the conditional probability of Y when X has already been occurred). Therefore if X appears much more frequently than X and Y appearing together, the confidence will be low.

Confidence of (X→Y) = Support(XY) / Support(X) = P(X ∩ Y) / P(X) = P(Y|X)

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

Let us consider Table 9.3 and Ttable 9.4 to further understand the concept of support and confidence.

Table 9.3 Example of the support measure

TID	*Items*	*Support = Occurence / Total Support*
1	ABC	Total Support = 5 Support {AB} = 2/5 = 40% Support {BC} = 3/5 = 60% Support {ABC} = 1/5 = 20%
2	ABD	
3	BC	
4	AC	
5	BCD	

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.4 Example of the confidence measure

TID	Items	Given X ⇒ Y Confidence = Occurence {X and Y}/ Occurence of (X)
1	ABC	Confidence {A ⇒ B} = 2/3 = 66% Confidence {B ⇒ C} = 3/4 = 75% Confidence {AB ⇒ C} = 1/2 = 50%
2	ABD	
3	BC	
4	AC	
5	BCD	

Let us consider the database given in Table 9.5 for further understanding support and confidence.

Table 9.5 Database for identification of association rules

Antecedent	*Consequent*
A	0
A	0
A	1
A	0
B	1
B	0
B	1

There are two rules derived from the association of these combinations:

Rule 1: A implies 0, i.e., A→0

Rule 2: B implies 1, i.e., B→1

The support for Rule 1 is the Number of times A and 0 appear together / Total Number of transactions.

Thus, Support of Rule 1 = 3/7

The Support for Rule 2 is the Number of times B and 1 appear together / Total Number of transactions.

Support of Rule 2 = 2/7

The Confidence for Rule 1 is Support of (A,0) / Support A, i.e.,

(Number of times A and 0 appear together / Total number of items) DIVIDED BY

(Number of times A appears / Total number of items)

Here, total number of items get cancelled.

Thus, the Confidence for Rule 1 is the Number of times A and 0 appear together / Number of times A appears.

So the Confidence for Rule 1 is 3/4

Similarly, the Confidence for Rule 2 is 2/3

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Going back to our general example involving 'X' (the Antecedent) and 'Y' (the Consequence), the Confidence of the rule does not depend on the frequency of 'Y' appearing. But in order to identify the strength of the rule it is important to consider the frequency of X and Y; X only; and Y only. In simple words, it is important to consider the whole dataset.

TheLift of the rule however considers the whole dataset, by taking into account the probability of Y also, in deciding the strength of the rule. So, Lift is the third important metric to check the strength or power of the rule.

9.4.3 Lift

It is very important to consider the frequency of Y or probability of Y for the effectiveness of the association mining rule X→Y.

As already explained, Confidence is the conditional probability of Y when X has already occurred. It is very important to consider how frequent Y is to guage the effectiveness of the confidence.

Thus, Lift is the ratio of conditional probability of Y when X is given to the unconditional probability of Y in the dataset. In simple words, it is Confidence of X→Y divided by the probability of Y.

1. Lift = P(Y|X) / P(Y)

   Or

2. Lift = Confidence of (X→Y) / P(Y)

   Or

3. Lift = (P(X ∩ Y) / P(X))/P(Y)

Thus, lift can be computed by dividing the confidence by the unconditional probability of consequent Y.

Let us suppose that Coke is a very common sales item in a store and that it usually appears in most of the transactions. Let us suppose that we have a rule of Candle→Coke which has a support of 20% and has a confidence of 90%. It is very logical to think that if coke is very popular and it appears in 95% of transactions, then obviously it also appears quite often with the candle as well. So, the rule for association of candle and coke will not be all that useful. But if we find that Candle→ Matchbox also has a support of 20% and a confidence of 90% then it is logical to suppose that the frequency of matchbox sales is very little as compared to the sale of coke. And the rule suggests that when we make a sale of candles, 90% chance indicates that a matchbox will also be sold in the same transaction. It is more effective and logical to conclude that when we sell a candle then we also sell a coke (coke is popular and will appear with every item not just with candle). As support and confidence are unable to handle this case, it is handled by the lift of the rule.

In this case, the probability of Y is very low in case of Candle→ Matchbox (Here, Y is matchbox) and will be very high in case of Candle→Coke (Here, Y is coke).

One can note that the low probability of Y, makes the X→Y rule more effective as compared to high probability of Y. Lift takes the note of this, and is defined as follows.

Lift = P(Y|X) / P(Y)

Or

Lift = Confidence of (X→Y) / P(Y)

Or

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Lift = (P(X ∩ Y) / P(X)) / P(Y)

Thus, the high probability of Y, i.e., P(Y) makes lift less effective and its low value makes it more effective.

Let us again consider the data given in Table 9.5, which has been reproduced in Table 9.6 for quick reference to calculate the lift of Rule 1 and Rule 2 as discussed earlier.

Table 9.6 Dataset

Antecedent	*Consequent*
A	0
A	0
A	1
A	0
B	1
B	0
B	1

$A \to 0 = \frac{P(0|A)}{P(0)}$ Rule ①

$= \frac{P(A \cap 0)\ ^{3/7}}{P(A)\ ^{4/7}} \Big/ P(0)\ ^{4/7}$

$= \frac{3/4}{4/7}$

$= 1.3125$ #

$B \to 1 = \frac{P(1|B)}{P(1)}$

$= \frac{P(1 \cap B)^{2/7}}{P(B)^{3/7}} \Big/ P(1)^{3/7}$

$= \frac{2/3}{3/7}$

$= 1.55$ #

Lift for Rule1, i.e., A→0 = P (0 | A) / P(0) = (P (A ∩ 0) / P(A))/ P(0) = (3/4) / (4/7)=1.3125

Lift for Rule2, i.e., B→1= P (1|B) / P(1) = (P (B ∩ 1) / P (B))/ P(1) = (2/3) / (3/7) =1.55

As discussed earlier, the confidence for Rule 1 is 3/4=0.75 and confidence for Rule 2 is 0.66.

It should be observed that although the Rule 1 has higher confidence as compared to Rule 2, but it has lower lift as compared to Rule 2.

Naturally, it would appear that Rule1 is more valuable because of having higher confidence; it appears more accurate (better supported). But, the accuracy of the rule can be misleading if it is independent of the dataset. Lift, as a metric is important because it considers both the confidence of the rule and the overall dataset.

In order to understand the importance of lift let us consider modified dataset given in Table 9.7.

In this modified database another five records have been added to the existing database (the five at the bottom). In these five records 0 appears four times, i.e., in the modified database 0 becomes more frequent; so, its probability has been increased, while the probability of 1 has been reduced.

Thus for first rule, i.e., A→0 the lift of the rule has been reduced in the modified database while it has same value of confidence as calculated below.

$Sup(A,0) = \frac{3}{12}$

Support of A→0 = 3/12

Confidence of A→0 = Support(A,0) / Support(A) = 3/4 = 0.75

$Con = \frac{Sup(A,0)}{Sup(A)} = \frac{3}{12} \div \frac{4}{12} = \frac{3}{12} \times \frac{12}{4} = \frac{3}{4}$

Lift = Confidence (A,0) / Support(0) = (3/4) / (8/12) = 36/32 = 1.125

Here, the lift of rule 1 has been reduced from 1.312 in the original dataset to 1.125 in the modified dataset, so the strength of rule 1 has been decreased in the modified dataset.

For Rule 2, i.e., B→1, the lift rule has been improved in the modified database while it has same value of confidence as calculated below.

$lift = \frac{con}{Sup(0)} = \frac{3}{4} \div \frac{8}{12} = \frac{3}{4} \times \frac{12}{8} = \frac{9}{8} = 1.125$

$Sup(B,1) = \frac{2}{12}$

Support of B→1 = 2/12

Confidence of B→1 = Support(B,1) / Support(B) = 2/3 = 0.66

$Con = \frac{Sup(B,1)}{Sup(B)} = \frac{2}{12} \div \frac{8}{12} = \frac{2}{3}$

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010

www.EBooksWorld.ir

Lift = Confidence / Support(1) = (2/3) / (4/12) = 2

Here, the lift of Rule 2 has been increased from 1.555 in the original dataset to 2 in the modified dataset, so the strength of Rule 2 has been improved in the modified dataset. Thus, lift considers both the confidence of the rule and the overall dataset.

Table 9.7 Modified dataset

Antecedent	*Consequent*	
A	0	
A	0	
A	1	
A	0	
B	1	
B	0	
B	1	
C	0	Five new records inserted into the database
D	0	
C	0	
C	1	
E	0	

Another representation of association rules

Sometimes, the representation of association rules also includes support and confidence as shown in Figure 9.6.

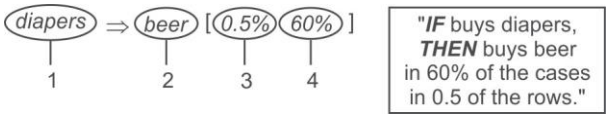


Figure 9.6 Representation of association rules

diapers ⇒ beer [0.5%, 60%]

Here,

1. Antecedent, left-hand side (LHS), body
2. Consequent, right-hand side (RHS), head
3. Support, frequency
4. Confidence, strength

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

9.5 The Naïve Algorithm for Finding Association Rules

To understand the Naïve Algorithm, let us consider the sales record of a grocery store. For simplicity, only four sale transactions of it are considered which deal with only four items for sale (Bread, Cornflakes, Jam and Milk) as shown in Table 9.8 and a naïve brute force algorithm is used to perform the task to identify the association of items in this sale record.

Let us find the association rules having minimum 'support' of 50% and minimum 'confidence' of 75%.

Table 9.8 Sale record of grocery store

Transaction ID	*Items*
100	Bread, Cornflakes
101	Bread, Cornflakes, Jam
102	Bread, Milk
103	Cornflakes, Jam, Milk

9.5.1 Working of the Naïve algorithm

In order to find association rules, the first step of the naïve algorithm is to list all the combinations of the items that are in stock and then identify the frequent combinations or combinations having frequency more than or equal to a specified support limit. Using this approach, the association rules that have the 'confidence' more than the threshold limit are identified.

Here, we have four items in stock, thus there are 2^4 possible combinations that we can create by considering null as one combination and if we ignore null then the following 15 combinations are possible as given in Table 9.9 along with their frequencies of occurrence in the transaction database.

Table 9.9 List of all itemsets and their frequencies

Itemsets	*Frequency*
Bread	**3**
Cornflakes	**3**
Jam	**2**
Milk	**2**
(Bread, Cornflakes)	**2**
(Bread, Jam)	1
(Bread, Milk)	1
(Cornflakes, Jam)	**2**

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Itemsets	Frequency
(Cornflakes, Milk)	1
(Jam, Milk)	1
(Bread, Cornflakes, Jam)	1
(Bread, Cornflakes, Milk)	0
(Bread, Jam, Milk)	0
(Cornflakes, Jam, Milk)	1
(Bread, Cornflakes, Jam, Milk)	0

Here, the minimum required support is 50% and we have to identify the frequent itemsets that appear in at least two transactions. The list of frequencies shows that all four items Bread, Cornflakes, Jam and Milk are frequent. It has been observed that the frequency goes down when we look at 2-itemsets and only two 2-itemsets such as (Bread, Cornflakes) and (Cornflakes, Jam) are frequent. All the frequent itemsets has been shown in bold in Table 9.8. There are no three or four itemsets that are frequent. The frequent itemsets are provided in Table 9.10.

Table 9.10 The set of all frequent items

Itemsets	Frequency
Bread	3
Cornflakes	3
Jam	2
Milk	2
(Bread, Cornflakes)	2
(Cornflakes, Jam)	2

As we are interested in association rules that can only occur with item pairs, thus individual frequent items Bread, Cornflakes, Jam and Milk are ignored, and item pairs (Bread, Cornflakes) and (Cornflakes, Jam) are considered for association rule mining.

Now, the association rules for the two 2-itemsets (Bread, Cornflakes) and (Cornflakes, Jam) are determined with a required confidence of 75%.

It is important to note that every 2-itemset (A, B) can lead to two rules A→B and B→A if both satisfy the required confidence. As stated earlier, confidence of A→B is given by dividing the support for A and B together, by the support for A.

Therefore, we have four possible rules which are given as follows along with their confidence:

Bread→Cornflakes

Confidence = Support of (Bread, Cornflakes) / Support of (Bread) = 2/3 = 67%

Cornflakes→Bread

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Confidence = Support of (Cornflakes, Bread) / Support of (Cornflakes) = 2/3 = 67%
Cornflakes→Jam
Confidence = Support of (Cornflakes, Jam) / Support of (Cornflakes) = 2/3 = 67%
Jam→Cornflakes
Confidence = Support of (Jam, Cornflakes) / Support of (Jam) = 2/2 = 100%

Therefore, only the last rule Jam→Cornflakes has more than the minimum required confidence, *i.e.,* 75% and it qualifies. The rules having more than the user-specified minimum confidence are known as confident.

9.5.2 Limitations of the Naïve algorithm

We can observe that in Table 9.7, we had a small number of items. However, the number of all possible combinations for which we have to calculate the frequency grows enormously as the number of items increase in the dataset. When dealing with 4 itemsets, it produces 15 different combinations in Table 9.8 and if we include a null combination (a combination with no items) as well, we would obtain 16. If there are six items it would have produced 64 combinations and if the number of items were 10 then we would have a table with 1024 combinations.

This simple algorithm works well with four items but if the number of items is say 100, the number of combinations is much larger, in billions. The number of combinations becomes about a million with 20 items since the number of combinations is 2^n with n items. The naive algorithm can be improved to deal more effectively with larger datasets.

9.5.3 Improved Naïve algorithm to deal with larger datasets

Rather than counting all the possible item combinations in the stock it will be better to focus only on the items that are sold in transactions. Because, we may have hundreds of items in stock but we are concerned with finding associations only for the items that are being sold together. Thus, it is natural to to focus only on the items that are sold in transactions instead of all the items available in stock.

We can look at each transaction and count only the combinations that actually occur; with this approach we do not count itemsets with zero frequency. And that will be an improvement.

As an example, Table 9.11 lists all the actual combinations occurring within the transactions given in Table 9.8.

Table 9.11 All possible combinations with nonzero frequencies

Transaction ID	*Items*	*Combinations*
100	Bread, Cornflakes	(Bread, Cornflakes)
200	Bread, Cornflakes, Jam	(Bread, Cornflakes), (Bread, Jam), (Cornflakes, Jam), (Bread, Cornflakes, Jam)
300	Bread, Milk	(Bread, Milk)
400	Cornflakes, Jam, Milk	(Cornflakes, Jam), (Cornflakes, Milk), (Jam, Milk), (Cornflakes, Jam, Milk)

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

We therefore only need to look at the following combinations and their frequencies are given in Table 9.12.

Table 9.12 Frequencies of all itemsets with nonzero frequencies

Itemsets	*Frequency*
Bread	3
Cornflakes	3
Jam	2
Milk	2
(Bread, Cornflakes)	2
(Bread, Jam)	1
(Cornflakes, Jam)	2
(Bread, Cornflakes, Jam)	1
(Bread, Milk)	1
(Cornflakes, Milk)	1
(Jam, Milk)	1
(Cornflakes, Jam, Milk)	1

We can now proceed as before. This would work better since the list of item combinations is significantly reduced from 15 (as given in Table 9.8) to 12 (as given in Table 9.11) and this reduction is likely to be much larger for bigger problems. Regardless of the extent of the reduction, this list will also become very large for, say, 1000 items since it includes all the nonzero itemsets that exist in the transactions database whether they have minimum support or not. Therefore, the naïve algorithm or any improvement of it is not suitable for large problems. We will discuss a number of algorithms for more efficiently solving the association rule problem starting with the classical ***Apriori algorithm***.

Before starting with the Apriori algorithm, it is important to discuss different ways to store our transaction database in computer memory because, it will have an impact on performance if we are processing a large number of transactions. In the next section, implementation issues of transition storage have been discussed.

9.6 Approaches for Transaction Database Storage

It is important to understand different ways to store a dataset of transactions before processing it using algorithms, as storage is an important issue for its performance.

There are three ways to store datasets of transactions. These are as follows.

- Simple Storage
- Horizontal Storage
- Vertical Storage

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Let us suppose the number of items be five; to demonstrate the different options. Let them be {I1, I2, I3, I4, I5}. Let there be only seven transactions with transaction IDs {T1, T2, T3, T4, T5, T6, T7}. This set of seven transactions with five items can be stored using three different methods given as follows.

9.6.1 Simple transaction storage

The first representation is commonly used and known as Simple Transaction Storage. Each row of the table shows the transaction ID and the purchased items as given in Table 9.13.

Table 9.13 A simple representation of transactions as an item list

Transaction ID	*Items*
T1	I1, I2, I4
T2	I4, I5
T3	I1, I3
T4	I2, I4, I5
T5	I4, I5
T6	I2, I3, I5
T7	I1, I3, I4

9.6.2 Horizontal storage

In this representation, each row is still a transaction, but columns have been created for each item. In the cell, 1 is filled against the item that occurs in a transaction and 0 against the rest as shown in Table 9.14.

Table 9.14 Horizontal storage representation

TID	*I1*	*I2*	*I3*	*I4*	*I5*
T1	1	1	0	1	0
T2	0	0	0	1	1
T3	1	0	1	0	0
T4	0	1	0	1	1
T5	0	0	0	1	1
T6	0	1	1	0	1
T7	1	0	1	1	0

The advantage of this storage system is that we can count the frequency of each item by counting the '1'sin the given column. As we can easily calculate the frequency of item I1 as 3 and item I4 as

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

5. We can also easily calculate the frequency of item pairs by counting the 1's that result after an 'AND' operation on corresponding columns. For example, the frequency of item pairs I1 and I2 is 1 (By AND operation on column of I1 and I2 we will get only one 1, i.e., T1).

9.6.3 Vertical representation

In this representation as given in Table 9.15, the transaction list is turned around. Rather than using each row to represent a transaction of the items purchased, each row now represents an item and it indicates transactions in which the item appears. The columns now represent the transactions. This representation is also called a TID-list since for each item it provides a list of TIDs (Transition Ids).

Table 9.15 Vertical storage representation

Item	*TID*						
	T1	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*
I1	1	0	1	0	0	0	1
I2	1	0	0	1	0	1	0
I3	0	0	1	0	0	1	1
I4	1	1	0	1	1	0	1
I5	0	1	0	1	1	1	0

A vertical representation also facilitates counting of items by counting the number of 1s in each row and in case of 2-itemsets, where you want to find out the frequency of occurrences of 2 items together in a transaction, you have to refer to the intersection of 2 rows corresponding to the items, and inspect one column at a time.

Example: If you want to find out frequency of item pairs (I1,I2) just look at rows I1 and I2 for each column if values of both I1 and I2 is 1 then this means I1 and I2 are occurring together in a transaction hence write their count as 1, move further in some way for each transaction. Finally you will get Table 9.16.

From the table it can be seen that the count for item pair {I1,I2}is 1; similarly the count of item pair {I1, I4} is 2. (Table 9.16)

Table 9.16 Frequency of item pairs

	T1	T2	T3	T4	T5	T6	T7
I1, I2	1	0	0	0	0	0	0
I1, I4	1	0	0	0	0	0	1

The vertical representation is not storage efficient in case of very large number of transactions. Also, if an algorithm needs data in the vertical representation then there would be a cost in transforming data from the horizontal representation to the vertical one.

With this we can move on to the concept of the Apriori algorithm.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

9.7 The Apriori Algorithm

The Apriori algorithm was developed by two Indians Rakesh Agrawal and Ramakrishnan Srikant in 1994, to mine frequent itemsets for identifying association rules.

It should be a matter of pride and motivation to have two Indians as inventors of association mining. Their work is the base for commonly used recommendation systems for modern applications such as product recommendations for online purchases and video recommendation and so on.

A brief profile of these two scientists, based on Wikipedia, has been presented below to motivate readers and in appreciation of their efforts.

9.7.1 About the inventors of Apriori

Rakesh Agrawal is the President and Founder of the Data Insights Laboratories. He is also the President of the Professor Ram Kumar Memorial Foundation.

Rakesh is an innovator and thought leader driven by the desire to make the world better through scientific breakthroughs and by building practical working systems. He is the recipient of the ACM-SIGKDD Inaugural Innovation Award, ACM-SIGMOD Edgar F. Codd Innovations Award, VLDB 10-Yr Most Influential Paper Award. Scientific American has included him in its first list of 50 top scientists and technologists.

Until recently, Rakesh was a Microsoft Technical Fellow and headed the Search Labs in Microsoft Research. Prior to joining Microsoft in March 2006, Rakesh was an IBM Fellow and led the Quest group at the IBM Almaden Research Center. Earlier, he was with the Bell Laboratories, Murray Hill from 1983 to 1989. He also worked for three years at a leading Indian, namely Bharat Heavy Electricals Ltd. He received the M.S. and Ph.D. degrees in Computer Science from the University of Wisconsin-Madison in 1983. He also holds a B.E. degree in Electronics and Communication Engineering from IIT-Roorkee, and a two-year Post Graduate Diploma in Industrial Engineering from the National Institute of Industrial Engineering (NITIE), Bombay. Both IIT-Roorkee and NITIE have decorated him with their distinguished alumni awards.

Rakesh has been granted 83 patents. He has published more than 200 research papers, many of them considered seminal. He has written the 1st as well as the 2nd highest cited of all papers in the fields of databases and data mining (18th and 26th most cited across all computer science). Wikipedia lists one of his papers as one of the most influential database papers. His papers have been cited more than 80,000 times, with more than 25 of them receiving more than 500 citations each and three of them receiving 5,000 citations each (Google Scholar). He is the most cited author in the field of database systems and the 26th most cited author across all of Computer Science (Citeseer). His research has been featured in N.B.C., New York Times, and several other media channels.

It is rare that a researcher's work creates not only a product, but a whole new industry. IBM's data mining product, Intelligent Miner, grew straight out of Rakesh's research. IBM's introduction of Intelligent Miner and associated services created a new category of software and services. His research has been incorporated into many other commercial products, including DB2 Mining Extender, DB2 OLAP Server, WebSphere Commerce Server, and Microsoft Bing Search engine, as well as many research prototypes and applications.

To know more, please visit http://rakeshagrawal.org/

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Ramakrishnan Srikant is a Distinguished Research Scientist at Google. His research interests include data mining, online advertising, and user modeling.

He previously managed the Privacy Research group, part of the Intelligent Information Systems Research department at IBM's Almaden Research Center.

Srikant has published more than 30 research papers that have been extensively cited. He has been granted 25 patents. Dr Srikant was a key architect for the IBM Intelligent Miner, and contributed the association rules and sequential patterns modules to the product.

Srikant received the ACM SIGKDD Innovation Award (2006) for his seminal work on mining association rules and privacy preserving data mining, the 2002 ACM Grace Murray Hopper Award for his work on mining association rules, the VLDB 2004 Ten Year Best Paper Award, and the ICDE 2008 Influential Paper Award. He was named an IBM Research Division Master Inventor in 1999. He has received 2 Outstanding Technical Achievement Awards for his contributions to the Intelligent Miner, and an Outstanding Innovation Award for his work on Privacy in Data Systems.

Srikant received his M.S. and Ph.D. from the University of Wisconsin, Madison and his B. Tech. from the Indian Institute of Technology, Madras.

To know more, please visit http://www.rsrikant.com/index.html

9.7.2 Working of the Apriori algorithm

The Apriori algorithm has been named so on the basis that it uses prior knowledge of frequent itemset properties. This algorithm consists of two phases. In the first phase, the frequent itemsets, i.e., the itemsets that exceed the minimum required support are identified. In the second phase, the association rules meeting the minimum required confidence are identified from the frequent itemsets. The second phase is comparatively straight forward – therefore, the major focus of research in this field is to improve the first phase.

To understand the Apriori algorithm, let us follow the learn by example approach. We will first apply this algorithm on a simple database to illustrate its working and then we will go into detail to learn the concept more deeply.

Example: Let us consider an example of only five transactions and six items. We want to identify association rules with 50% support and 75% confidence with the Apriori algorithm. The transactions are given in Table 9.17.

Table 9.17 Transactions database

Transaction ID	Items
T1	Bread, Cornflakes, Eggs, Jam
T2	Bread, Cornflakes, Jam
T3	Bread, Milk, Tea
T4	Bread, Jam, Milk
T5	Cornflakes, Jam, Milk

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

For 50% support each frequent item must appear in at least three transactions. The first phase of the Apriori algorithm will be to find frequent itemsets.

Phase 1: Identification of frequent itemsets

It starts with the identification of candidate 1 itemsets, represented by C1 (one itemsets that may be frequent) and it is always the items in the stock which the store deals with, Here, we have six items, so C1 will be as shown below.

Table 9.18 Candidate one itemsets C1

Item
Bread
Cornflakes
Eggs
Milk
Jam
Tea

From Candidate 1 itemsets, frequent one-itemsets are represented by L1 and found by calculating the count of each candidate item and selecting only those counts which are equal to or more than the threshold limit of support, i.e., 3 in this case. Thus, frequent single itemsets, L1, will be as shown in Table 9.19.

Table 9.19 Frequent items L1

Item	*Frequency*
Bread	4
Cornflakes	3
Milk	3
Jam	4

Then, the candidate 2-itemsets or C2 are determined per the process illustrated below:
C2 = L1 JOIN L1.
The joining of L1 with itself has been illustrated below.

Bread	JOIN	Bread
Cornflakes		Cornflakes
Milk		Jam
Jam		Milk

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

In the Join operation the first step will be to perform a Cartesian product, i.e., to make all possible pairs between L1 and L1 as shown below.

Bread, Bread
Bread, Cornflakes
Bread, Milk
Bread, Jam
Cornflakes, Bread
Cornflakes, Cornflakes
Cornflakes, Milk
Cornflakes, Jam
Milk, Bread
Milk, Cornflakes
Milk, Milk
Milk, Jam
Jam, Bread
Jam, Cornflakes
Jam, Milk
Jam, Jam

Here, we have 16 possible pairs. In association mining, we are interested in item pairs, so the pairs having same itemsets (like Bread, Bread) need to be considered for removal from the list. Similarly, if we already have an item pair of Milk, Jam then there is no need to list item pair Jam, Milk so it will also be removed from the list.

Thus, to create C2 the following steps are applied:

- Perform Cartesian product of L1 with itself
- Some item pairs may have identical items. Keep just one. Remove the extras.
- Select only those item pairs in which items are in lexical order (so that if we have Milk, Jam then Jam, Milk should not appear).

Thus, the final C2 will be as shown below:

Bread, Cornflakes
Bread, Milk
Bread, Jam
Cornflakes, Milk
Cornflakes, Jam
Milk, Jam

The frequency of each of these item pairs will be found as shown in the table given below.

Table 9.20 Candidate item pairs C2

Item pairs	*Frequency*
(Bread, Cornflakes)	2
(Bread, Milk)	2
(Bread, Jam)	3

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Item pairs	*Frequency*
(Cornflakes, Milk)	1
(Cornflakes, Jam)	3
(Milk, Jam)	2

We therefore have only two frequent item pairs in L2 as shown in Table 9.21.

Table 9.21 Frequent two item pairs L2

Item pairs	*Frequency*
(Bread, Jam)	3
(Cornflakes, Jam)	3

Generation of C3 from L2

C3 is generated from L2 by carrying out a JOIN operation over L2 as shown below.

C3 = L2 JOIN L2

It will involve the same steps as performed for C2, but it has one important pre-requisite for Join, i.e., two items are joinable if their first item is common. In a generalized case:

Ck = Lk-1 JOIN Lk-1

And they are joinable if their first k-2 items are the same. So, in case of C3, the first item should be the same in L2, while in case of C2 there is no requirement of first item similarity because k-2 in the C2 case is 0.

From {Bread, Jam} and {Cornflakes, Jam} two frequent 2-itemsets, we do not obtain a candidate 3-itemset since we do not have two 2-itemsets that have the same first item. This completes the first phase of the Apriori algorithm.

Phase 2: Generation of rules

The two frequent 2-itemsets given above lead to the following possible rules.

Bread→Jam

Jam→Bread

Cornflakes→Jam

Jam→Cornflakes

The confidence of these rules is obtained by dividing the support for both items in the rule by the support for the item on the left-hand side of the rule. The confidence of the four rules therefore are 3/4 = 75%, 3/4 = 75%, 3/3 = 100%, and 3/4 = 75% respectively. Since all of them have a minimum 75% confidence, they all qualify.

In order to generalize the Apriori algorithm, let us define the representation of itemsets.

Representation of Itemsets

To generate C_k, let us first define L_{k-1}.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Let *l*1 and *l*2 be itemsets in list L_{k-1} and the notation *li*[*j*] refers to the *j*th item in *li*.

To generate C2 from L1, C2 is represented as C_k and L1 as L_{k-1}. Let us consider L1 as shown in Table 9.22, the description of each item list has been given below.

Table 9.22 L1 for generation of C2 having only one element in each list

	L1
1	represented as item list l1 and its item, i.e., 1 is represented as l1[1].
2	represented as item list l2 and its item, i.e., 2 is represented as l2[1].
3	represented as item list l3 and its item, i.e., 3 is represented as l3[1].
5	represented as item list l4 and its item, i.e., 5 is represented as l4[1].

Let us consider L2 given in Table 9.23.

Table 9.23 L2 for generation of C3 (i.e., K=3) having two elements in each list

	L2
1,2	represented as item list l1 and its item, 1 is represented as l1[1] or l1[k-2] and 2 is represented as l1[2] or l1[k-1]
4,5	represented as item list l2 and its item, 4 is represented as l2[1] or l2[k-2] and 5 is represented as l2[2] or l2[k-1]
2,6	represented as item list l3 and its item, 2 is represented as l3[1] or l3[k-2] and 6 is represented as l3[2] or l3[k-1]

Let us consider L3 given in Table 9.24.

Table 9.24 L3 for generation of C4 (i.e., K = 4) having three elements in each list

	L3
1,2,3	represented as item list l1 and its item, 1 is represented as l1[1] or l1[k-3] and 2 is represented as l1[2] or l1[k-2] and 3 is represented as l1[3] or l1[k-1]
1,2,5	represented as item list l2 and its item, 1 is represented as l2[1] or l2[k-3] and 2 is represented as l2[2] or l2[k-2] and 5 is represented as l2[3] or l2[k-1]
2,3,6	represented as item list l3 and its item, 2 is represented as l3[1] or l3[k-3] and 3 is represented as l3[2] or l3[k-2] and 6 is represented as l1[3] or l1[k-1]

Defining Phase 1: Identification of frequent itemsets

The Apriori algorithm uses an iterative approach and is also known as a level-wise search where k-itemsets are used to search (k+1) itemsets. First, the database is scanned to find the set of frequent 1-itemsets to accumulate the count for each item denoted as C1, known as candidate 1-itemset, and gathering those items satisfying minimum required support denoted as L1, frequent 1-itemset.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Next, L1 is used to find C2, candidate 2-itemsets and collecting those items that satisfy minimum support denoted as L2, frequent 2-itemsets; which is used to find C3, candidate 3-itemsets; which is used further to identify L3, frequent 3-itemsets and so on, until no more frequent k-itemsets can be found. This process has been illustrated in Figure 9.7. The finding of each L_k requires one full scan of the database.

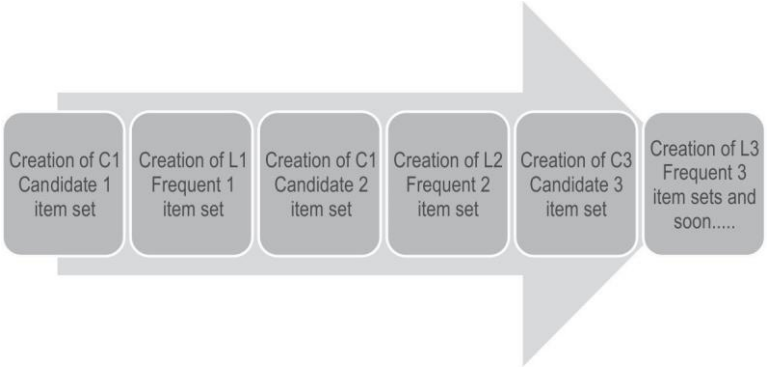


Figure 9.7 Process for identification of frequent itemsets

To proceed further, we have to generate C2 based on L1. To understand the generation of candidate itemsets, it is important to understand the representation of items in the item lists.

Step 3: Form candidate 2 itemset pair, i.e., C2.

C2 is created by joining L1 with itself by using following joining rule.

The join step

To find C_k, a set of candidate k-itemsets denoted as C_k is generated by joining L_{k-1} with itself. Let $l1$ and $l2$ be itemsets in L_{k-1}, the notation $li[j]$ refers to the jth item in li, e.g., $l1[k\text{-}2]$ refers to the second last item in $l1$. By convention, the prior assumption is that items within a transaction or itemset are sorted in lexicographic order. For the (k-1) itemset, li, this means that the items are sorted such that $li[1] < li[2] < : : : < li[k\text{-}1]$.

The join, L_{k-1} on L_{k-1}, is performed, where members of L_{k-1} are joinable if their first (k-2) items are in common.

That is, members $l1$ and $l2$ of L_{k-1} are joined if $(l1[1] = l2[1]) \wedge (l1[2] = l2[2]) \wedge : : : \wedge (l1[k\text{-}2] = l2[k\text{-}2]) \wedge (l1[k\text{-}1] < l2[k\text{-}1])$. The condition $l1[k\text{-}1] < l2[k\text{-}1]$ simply ensures that no duplicates are generated.

The resulting itemset formed by joining $l1$ and $l2$ is $l1[1], l1[2], : : : , l1[k\text{-}2], l1[k\text{-}1], l2[k\text{-}1]$.

Thus, there are three important points that make items of L_{k-1} joinable and these are as follows.

Point 1: The members of L_{k-1} are joinable if their first (k-2) items are in common

It means for C2 = L1 JOIN L1, with k = 2, there is no requirement for having the first common data item, because k-2 = 0.

For C3 = L2 JOIN L2, with k = 3, the first k-2, i.e., first 1 item should be the same in the itemset.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

For C4 =L3 JOIN L3, having k =4, the first k-2, i.e., first 2 items should be the same in the itemset.

For example, Let us verify whether L1 given in Table 9.25 is joinable or not, to create C2.

Table 9.25 L1

L1
1
2
3

C2 =L1 JOIN L1

*There is no requirement for first common item because k-2 =0 in this case and all itemsets, i.e., 1, 2 and 3 are joinable.

Now, let us verify whether L2 given in Table 9.26 is joinable or not to create C3.

Table 9.26 L2

L2	
1,2	represented as item list l1 and its item, i.e.,1 is represented as l1[1] or l1[k-2] and 2 is represented as l1[2] or l1[k-1]
1,3	represented as item list l2 and its item, i.e.,1 is represented as l2[1] or l2[k-2] and 3 is represented as l2[2] or l2[k-1]
2,5	represented as item list l3 and its item, 2 is represented as l3[1] or l3[k-2] and 5 is represented as l3[2] or l3[k-1]

C3 =L2 JOIN L2 [Here, k =3]

Itemsets are joinable only if their first k-2 itemsets, i.e., first 1 item is common. Thus, itemsets 1, 2 and 1, 3 are joinable as their first element is 1 in both the itemsets.

Now, let us identify that L3 given in Table 9.27 is joinable or not to create C4.

Table 9.27 L3

L3	
1,2,3	represented as item list l1 and its item, i.e.,1 is represented as l1[1] or l1[k-3] and 2 is represented as l1[2] or l1[k-2] and 3 is represented as l1[3] or l1[k-1]
1,2,5	represented as item list l2 and its item, i.e.,1 is represented as l2[1] or l2[k-3] and 2 is represented as l2[2] or l2[k-2] and 5 is represented as l2[5] or l2[k-1]
1,3,5	represented as item list l3 and its item, 1 is represented as l3[1] or l3[k-3] and 3 is represented as l3[2] or l3[k-2] and 5 is represented as l1[3] or l1[k-1]

C4 =L3 JOIN L3 [Here, k =4]

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Itemsets are joinable only if their first k-2 itemsets, i.e., first 2 items are common in this case. Thus, itemsets 1, 2, 3 and 1, 2, 5 are joinable as their first two elements, i.e., 1 and 2 are common in both the itemsets.

Point 2

In joinable item lists l1(k-1) should be less than l2(k-1), i.e., l1(k-1)<l2(k-1): this rule ensures that items in the itemset should be in lexicographic order and there is no repetition.

To understand this condition, let us find C2 for given L1 in Table 9.28.

Table 9.28 L1

L1
1
2
3
5

As discussed earlier, there is no requirement for the first common item, because k-2 = 0 in this case and all itemsets, i.e., 1, 2, 3 and 5 are joinable.

Thus, C2 = L1 JOIN L1 (the base for join is the Cartesian product of two itemsets) as illustrated in Table 9.29.

Table 9.29 Generation of C2

1	JOIN	1
2		2
3		3
5		5

Possible pairs for 1 are 11*, 12, 13&15

*11 will not appear in the final list because, here l1(1) = l2(1) and according to the joining rule l1(1) should be less than l2(1) [1 is not less than 1].

Thus, selected pairs for 1 are 12, 13 & 15

Possible pairs for 2 are 21*, 22*, 23 & 25

21 and 22 will not be selected because, l1(1) is not less than l2(1) [2 is not less than 1 and 2 is not less than 2].

Thus, selected pairs for 2 are 23 & 25

Possible pairs for 3 are 31*, 32*, 33* & 35

31, 32 and 33 will not be selected because , l1(1) is not less than l2(1) [3 is not less than 1; 3 is not less than 2; and 3 is not less than 3].

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Thus, the selected pair for 3 is 35
Possible pairs for 5 are 51*, 52*, 53* & 55*
*No item will be selected because, l1(1) in not less than l2(1) [5 is not less than 1,2,3 and 5].
Thus, C2 will be as shown in Table 9.30.

Table 9.30 Generated C2

C2
12
13
15
23
25
35

Point 3

The resulting itemset formed by joining *l*1 and *l*2 is *l*1[1], *l*1[2], : : : , *l*1[*k*-2], *l*1[*k*-1], *l*2[*k*-1].

It means that, resulting itemset will contain all the items of the first item list, i.e., from item 1 to k-1 and last item of the second item list, i.e., k-1 item of second list, because other k-2 items of second list are already there, as they are common with first list.

To illustrate this point, let us identify C3 for given L2 in Table 9.31.

Table 9.31 L2

L2
1,2
1,3
2,3

Here, itemsets 1, 2 (first list) and 1, 3 (second list) are joinable. The final item list for C3 will be all items of first list and last item of second list.

Thus, C3 will be as shown in Table 9.32.

Table 9.32 C3

C3
1, 2, 3

Let us identify C4 for the given L3 in Table 9.33.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.33 L3

L3
1, 2, 4
1, 2, 5
1, 2, 7
1, 3, 6

Here, itemsets (1, 2, 4), (1, 2, 5) and (1, 2, 7) are joinable. The resultant C4 will be as shown in Table 9.34.

Table 9.34 C4

C4	
1, 2, 4, 5	Considering (1, 2, 4) as first list and (1, 2, 5) secondlist
1, 2, 4, 7	Considering (1, 2, 4) as first list and (1, 2, 7) secondlist
1, 2, 5, 7	Considering (1, 2, 5) as first list and (1, 2, 7) secondlist

Example

Let us understand the process for identification of frequent itemsets for the given transaction database of Table 9.35. The threshold value of support is 50% and confidence is 70%,

Table 9.35 Transaction database

T1	1,3,4
T2	2,3,5
T3	1,2,3,5
T4	2,5

Step 1

Create C1 candidate 1-itemsets. It is all the items in the transaction database as shown in Figure 9.11. Figure 9.8 also contains the frequency of each item.

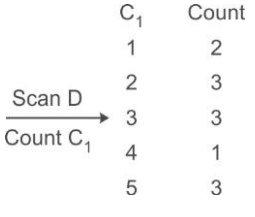


Figure 9.8 C1, candidate 1-itemset and their count

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Step 2

Create frequent 1-itemset, i.e., list of 1-itemsets whose frequency is more than the threshold value of support, i.e., 2 as shown in Figure 9.9.

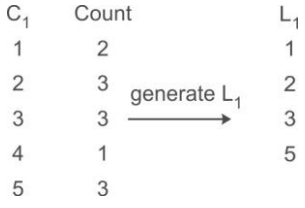


Figure 9.9 L1, frequent 1-itemset

Candidate two itemsets, C2 with its frequency count as given in Table 9.36.

Table 9.36 Generation of C2

C2	*Count*
1, 2	1
1, 3	2
1, 5	1
2, 3	2
2, 5	3
3, 5	2

L2 is created by selecting those candidate pairs having support of 2 or more as shown in Table 9.37.

Table 9.37 Generation L2

L2
1, 3
2, 3
2, 5
3, 5

From the given L2, the next step will be to generate C3 by using L2 JOIN L2
Thus, C3 will be as shown in Table 9.38.

Table 9.38 Generation of C3

C3
2, 3, 5

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

The frequency of candidate three itemset 2, 3, 5 is 2, so C3 is qualified as L3. Thus, frequent three itemset is 2, 3, 5 for given dataset.

This is the final frequent item list and it completes the first phase of the Aproiri algorithm to find the frequent itemsets.

Phase 2: Generating association rules from frequent itemsets

To understand the process of generation of association rules from the frequent itemset, let us consider frequent itemset *l*.

- For each frequent itemset *l* generates all non-empty subsets of *l*.
- For every non-empty subset *s* of *l*, output the rule $s \rightarrow l\text{-}s$ and if the confidence of the rule is more than the threshold value of the confidence, then, this rule will be selected as the final association rule.

Let us generate the rules for the frequent itemset (2, 3, 5) represented as *l*. Here, non-empty subsets are {{2},{3},{5},{(2,3)}, {(2, 5)},{(3,5)}.

For every non-empty subset, the rule will be generated as follows.

$2 \rightarrow 3, 5$ [Here, (2) is *s* and (3, 5) is *l-s*]

$3 \rightarrow 2, 5$

$5 \rightarrow 2, 3$

$2, 3 \rightarrow 5$

$2, 5 \rightarrow 3$

$3, 5 \rightarrow 2$

The next step will be to calculate the confidence for each rule as shown below.

$2 \rightarrow 3, 5$; Confidence $= S(2 \cap 3 \cap 5) / S(2) = 2/3 = 0.67$

$3 \rightarrow 2, 5$; Confidence $= S(2 \cap 3 \cap 5) / S(3) = 2/3 = 0.67$

$5 \rightarrow 2, 3$; Confidence $= S(2 \cap 3 \cap 5) / S(5) = 2/3 = 0.67$

$2, 3 \rightarrow 5$; Confidence $= S(2 \cap 3 \cap 5) / S(2 \cap 3) = 2/2 = 1.0$

$2, 5 \rightarrow 3$; Confidence $= S(2 \cap 3 \cap 5) / S(2 \cap 5) = 2/3 = 0.67$

$3, 5 \rightarrow 2$; Confidence $= S(2 \cap 3 \cap 5) / S(3 \cap 5) = 2/2 = 1.0$

Here, the minimum threshold for confidence is 70%, thus selected association rules are as follows.

$2, 3 \rightarrow 5$;

$3, 5 \rightarrow 2$;

The possible rules from $2, 3 \rightarrow 5$

It is intuitive to create two new rules as $2 \rightarrow 5$ and $3 \rightarrow 5$ from the given rule. Since $2, 3 \rightarrow 5$ has confidence more than the threshold limit, in this case, it is not implicit or guaranteed that $2 \rightarrow 3$ and $3 \rightarrow 5$ will have confidence more than the threshold limit as there is no correlation between denominator and quotient of both the rules as shown below.

Confidence of $2, 3 \rightarrow 5 = S(2 \cap 3 \cap 5) / S(2 \cap 3)$

Confidence of $2 \rightarrow 5 = S(2 \cap 5) / S(2)$

Same is the case for $3 \rightarrow 5$ as shown below.

Confidence of $3 \rightarrow 5 = S(3 \cap 5) / S(3)$

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Thus, the given high confidence rule 2, 3→5 does not implicitly state that 2→5 and 3→5 will have its confidence more than the threshold limit. Thus, there is a need to calculate the confidence of these rules as shown in Table 9.39.

Similarly, the given high confidence rule 3, 5→2 does not implicitly state that 3→2 and 5→2 will have its confidence more than the threshold limit. Thus, again there is a need to calculate the confidence of these rules as shown in Table 9.39.

Table 9.39 Calculation of confidence

Rule	*Confidence*
2, 3→5	1.0
2→5	S(2∩5) / S(2) = 3/3 = 1.0
3→5	S(2∩5) / S(3) = 3/3 = 1.0
3, 5→2	1.0
3→2	S(3∩2) / S(3) = 2/3 = 0.67
5→2	S(5∩2) / S(5) = 3/3 = 1.0

Thus, final rules having confidence more than the threshold limit, i.e., 75% are as follows.

Selected Association rules are
2, 3→5
2→5
3→5
3, 5→2
5→2

Apriori property

Apriori property states that all nonempty subsets of a frequent itemset must also be frequent.

It means all the subsets of the candidate itemset should be frequent otherwise that itemset will be removed from the candidate itemset. This step is called pruning of the candidate itemset and it will help to reduce the search space and to improve the performance of the algorithm because now the frequency of fewer item pairs need to be computed.

Lattice structure of frequent itemsets

The Apriori property that defines an itemset can be frequent only if all the subsets of the itemset are frequent. This can be illustrated by a lattice structure. Let us suppose there are four items A, B, C and D in the dataset. Then, the four-itemset ABCD will be frequent only if three itemsets

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

ABC, ABD, ACD and BCD are frequent. These three itemsets are frequent only if their subsets, i.e., AB, AC, BC, BD, AD and CD are frequent. And these two itemsets are frequent only if their subsets, i.e., A, B, C and D items are also frequent as shown in Figure 9.10.

From the lattice structure we can conclude that the Apriori algorithm works bottom up one level at a time, i.e., it first computes frequent 1-itemsets, the candidate 2-itemsets and then frequent 2-temsets and so on. The number of scans of the transaction database is equal to the maximum number of items in the candidate itemsets.

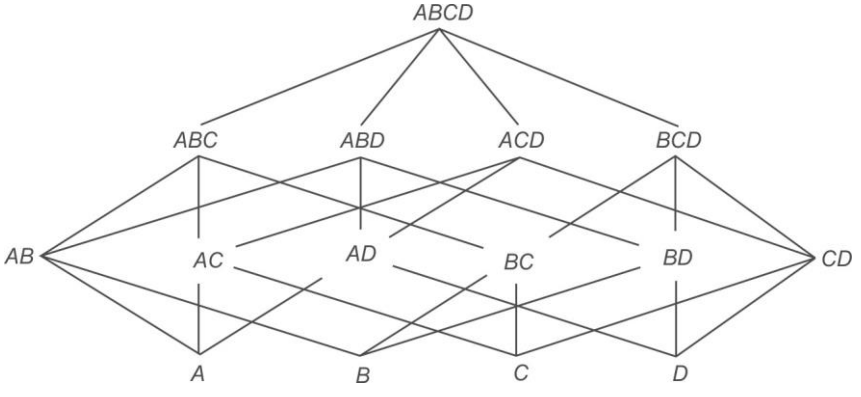


Figure 9.10 Lattice structure of frequent itemsets

For example, Apriori property states that for a frequent three itemset (1, 2, 3) all of its non empty subsets, i.e., (1, 2), (2, 3) and (1, 3) must be frequent, because, support of a superset is always less than or equal to support of its subset. It means that if (1, 3) is not frequent then (1, 2, 3) will also not be frequent.

This Apriori property also explains the significance of having first k-2 items common in a joining principle.

Let us suppose, we have L2 that is (1, 2) and (2, 3), then, one may suppose that it could produce a set (1,2,3) as C3. But as discussed earlier, (1, 2, 3) should be frequent only if all of its nonempty subsets, i.e., (1, 2), (2, 3) and (1, 3) are frequent. Here, as given in L2 (1, 3) is not frequent. So, itemsets given in L2 are considered as non joinable and the condition of having first k-2 items common is enforced to ensure it.

Now, what would happen if L2 has (1, 2) and (1, 3) but not (2, 3)? Then, according to the joining principle C3 will be generated as (1, 2, 3), but it will be discarded by the Apriori property because one of its nonempty subset, i.e., (2, 3) is not frequent. Thus, the combination of joining principle and apriori property make the whole process complete.

In conclusion, it is not the only joining principle that decides about the candidate itemsets, the joining principle can produce some extra items sets that can be removed further by the pruning or apriori property.

Example: Find association rules for the transaction data given in Table 9.40 for having support 15% and confidence 70%.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.40 Transaction database for identification of association rules

TID	*List of Items*
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I4

The frequency count for each item, C1 is given in Table 9.41.

Table 9.41 C1

C1	*Count*
I1	6
I2	7
I3	5
I4	3
I5	2

All the items have frequency more than the specified support limit, thus all 1-item candidate sets given as C1 qualify as 1-item frequent itemset L1. The next step will be to generate C2.

C2 = L1 JOIN L1

The process of creation of C2 and L2 is illustrated in Figure 9.11.

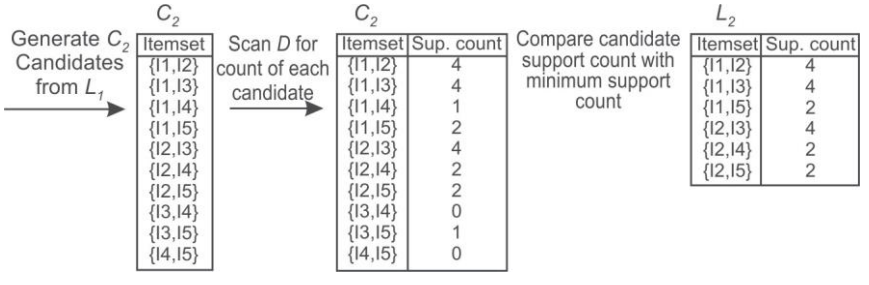


Figure 9.11 Generation of C2 and L2

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

From the given L2, C3 is generated by considering those item pairs whose first k-2 items, i.e. first 1 item is common as shown in Table 9.42.

Table 9.42 Generation of C3

C3
I1, I2, I3
I1, I2, I5
I1, I3, I5
I2, I3, I4
I2, I3, I5
I2, I4, I5

Before finding the frequency of each candidate item pair, the Apriori property will be applied to prune the candidate list.

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? Let us check that as shown in Table 9.43.

Table 9.43 Pruning of candidate itemset C3

I1, I2, I3	The 2-item subsets of I1, I2 and I3 are (I1, I2), (I1, I3), and (I2, I3). All 2-item subsets are members of L2. Therefore, keep I1, I2 and I3 in C3.	Qualify
I1, I2, I5	The 2-item subsets of I1, I2 and I5 are (I1, I2), (I1, I5), and (I2, I5). All 2-item subsets are members of L2. Therefore, keep I1, I2 and I5 in C3.	Qualify
I1, I3, I5	The 2-item subsets of I1, I3 and I5 are (I1, I3), (I1, I5), and (I3, I5). Since, (I3, I5) is not a member of L2, and so it is not frequent. Therefore, remove I1, I3 and I5 from C3.	Does not qualify
I2, I3, I4	The 2-item subsets of I2, I3 and I4 are (I2, I3), (I2, I4), and (I3, I4). Since, (I3, I4) is not a member of L2, and so it is not frequent. Therefore, remove I2, I3 and I4 from C3.	Does not qualify
I2, I3, I5	The 2-item subsets of I2, I3 and I5 are (I2, I3), (I2, I5), and (I3, I5). Since, (I3, I5) is not a member of L2, so it is not frequent. Therefore, remove I2, I3 and I5 from C3.	Does not qualify
I2, I4, I5	The 2-item subsets of I2, I4 and I5 are (I2, I4), (I2, I5), and (I4, I5). Since, (I4, I5) is not a member of L2, it is not frequent. Therefore, remove I2, I4 and I5 from C3.	Does not qualify

Thus pruned C3 will be as shown in Table 9.44.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.44 Pruned C3

Pruned C3
I1, I2, I3
I1, I2, I5

It is important to note that pruning results into improving the efficiency of the algorithm, as in this case instead of finding the count of six three item pairs, there is a need to find the count for only 2 three item pairs (pruned list) which is a significant improvement in the efficiency of the system.

The process of creation of L3 from C3 by identifying those 3 itemsets that has a frequency more than or equal to the given value of threshold value of support, i.e., 2 has been shown below in Figure 9.12.

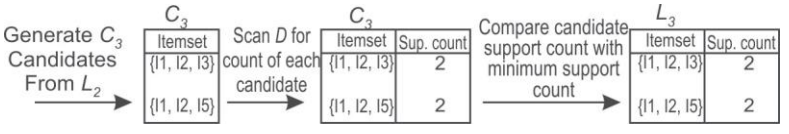


Figure 9.12 Generation of C3 and L3

The next step will be to generate C4 as shown below.

C4 =L3 JOIN L3

C4 will be I1, I2, I3 & I5, because 2 items are common as shown in Table 9.45.

Table 9.45 C4

C4
I1, I2, I3, I5

But this itemset is pruned by the Apriori property because its subset (I2, I3, I5) is not frequent as it is not present in L3. Thus, C4 is null and the algorithm terminates at this point, having found all of frequent itemsets as shown in Table 9.46.

Table 9.46 Pruned C4

C4
NULL

In this case, instead of finding the count of 1 four-item pair, pruning indicates that there is no need to find its count as all its subsets are not frequent.

With this, the first phase of the Apriori algorithm has been completed. Now the next phase to generate association rules from frequent itemsets will be performed.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Phase 2: Generating association rules from frequent itemsets

Let us apply this rule to frequent 3-itemsets (I1, I2, I3) and (I1, I2, I5) found in case of example given in Table 9.40.

For first frequent itemset (I1, I2, I3), non-empty subsets are {{I1},{I2},{I3},{(I1, I2)}, {(I1, I3)},{(I2, I3)}.

For every non-empty set, the rule will be generated as follows:

I1→I2, I3 [Here, (I1) is *s* and I2 and I3 are *l-s*]

I2→I1, I3

I3→I1, I2

I1, I2→I3

I1, I3→I2

I2, I3→I1

The next will be to calculate the confidence for each rule as shown below.

I1→I2, I3; Confidence = S(I1∩I2∩I3) / S(I1) = 2/6 = 0.3

I2→I1, I3; Confidence = S(I1∩I2∩I3) / S(I2) = 2/7 = 0.28

I3→I1, I2; Confidence = S(I1∩I2∩I3) / S(I3) = 2/5 = 0.4

I1, I2→I3; Confidence = S(I1∩I2∩I3) / S(I1∩I2) = 2/4 = 0.5

I1, I3→I2; Confidence = S(I1∩I2∩I3) / S(I1∩I3) = 2/4 = 0.5

I2, I3→I1; Confidence = S(I1∩I2∩I3) / S(I2∩I3) = 2/4 = 0.5

Since, the minimum threshold is 70%, there are no rules that qualify from the frequent itemset {1,2, 3}.

Now, let us apply this rule to second frequent 3-itemset (I1, I2, I5). For this frequent itemset, non-empty subsets are {I1}, {I2}, {I5}, {(I1, I2)}, {(I1, I5)}, {(I2, I5)}.

For every non-empty set the rule will be generated as follows:

I1→I2, I5 (Here, (I1) is *s* and I2 and I5 are *l-s*]

I2→I1, I5

I5→I1, I2

I1, I2→I5

I1, I5→I2

I2, I5→I1

The next step will be to calculate the confidence for each rule as shown below.

I1→I2, I5; Confidence = S(I1∩I2∩I5) / S(I1) = 2/6 = 0.3

I2→I1, I5; Confidence = S(I1∩I2∩I5) / S(I2) = 2/7 = 0.28

I5→I1, I2; Confidence = S(I1∩I2∩I5) / S(I5) = 2/2 = 1

I1, I2→I5; Confidence = S(I1∩I2∩I5) / S(I1∩I2) = 2/4 = 0.5

I1, I5→I2; Confidence = S(I1∩I2∩I5) / S(I1∩I5) = 2/2 = 1

I2, I5→I1; Confidence = S(I1∩I2∩I5) / S(I2∩I5) = 2/2 = 1

Now, there are three rules whose confidence is more than minimum threshold value of 70%, and these rules are as follows:

I5→I1, I2

I1, I5→I2

I2, I5→I1

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

The possible rules from I5→I1, I2

It is intuitive to create two new rules as I5→I1 and I5→I2 from the given rule.

Since, I5→I1, I2 has confidence more than threshold limit, so it is implicit that I5→I1 and I5→I2 will also have their confidence more than the threshold limit as justified below.

Confidence of I5→I1,I2 = S(I5∩I1∩I2) / S(I5)

Confidence of I5→I1 = S(I5∩I1) / S(I5)

Since, Support of subset (I5, I1) will be more than or equal to Support of superset (I5, I1, I2), it is implicit that Confidence of I5→I1 will be more than equal to Confidence of I5→I1, I2.

Similarly, Confidence of I5→I2 = S(I5∩I2) / S(I5) will be more than equal to Confidence of I5→I1,I2.

Thus, I5→I1 and I5→I2 will also have their confidence more than the threshold limit.

Discussing possible rules from I2, I5→I1

It is intuitive to create two new rules as I2→I1 and I5→I1 from the given rule. I2, I5→I1 has confidence more than the threshold limit, but it is not implicit or guaranteed that I2→I1 and I5→I1 will have their confidence more than the threshold limit as explained below.

Confidence of I2, I5→I1 = S(I2∩I5∩I1) / S(I2∩I5)

Confidence of I2→I1 = S(I2∩I1) / S(I2)

Here, there is no correlation between denominator and quotient of both the rules, so it is not implicit that this rule will also have its confidence more than the threshold.

Similarly, the Confidence of I5→I1 = S(I5∩I1) / S(I5)

Again, there is no correlation between denominator and quotient of this rule with I2, I5→I1 rules, so it is not implicit that this rule will also have its confidence more than the threshold.

The given high confidence rule I2, I5→I1 does not implicitly state that I2→I1 and I5→I1 will have their confidence more than the threshold limit. Thus, there is a need to calculate the confidence of these rules.

Similarly, the given high confidence rule I1, I5→I2 does not implicitly state that I1→I2 and I5→I2 will have their confidence more than the threshold limit. Thus, again there is a need to calculate the confidence of these rules.

The process of generation of all association rules from I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules has been illustrated in Table 9.47.

Table 9.47 Generation of association rules

Association Rule	*Discussion*	*Confidence*	*More than threshold limit Or Not*
I5→I1, I2	Already identified	1.0	Yes
I5→I1	Implicit	No need to calculate it will be more than or equal to I5→I1, I2	Yes
I5→I2	Implicit	No need to calculate it will be more than or equal to I5→I1, I2	Yes

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Association Rule	*Discussion*	*Confidence*	*More than threshold limit Or Not*
I2, I5→I1	Already identified	1.0	Yes
I2→I1	Not found earlier, confidence need to be calculated	Confidence of I2→I1 = S(I2∩I1) /S(I1) = 4/6 = 67%	No
I5→I1	Already found from I5→I1, I2 given in row 2	No need to calculate it will be more than or equal to I5→I1, I2	Yes (Already listed)
I1, I5→I2	Already identified	1.0	Yes
I1→I2	Not found earlier, so confidence needs to be calculated	Confidence of I1→I2 = S(I2∩I1) /S(I2) = 4/7 = 57%	No
I5→I2	Already found from I5→I1, I2 given in row 3		Yes (Already listed)

During generation of the final rules, it is important to look at L2 because there may be some frequent 2-itemsets that have not appeared in three itemsets and may be left out. The frequent 2 itemsets generated earlier have been reproduced in Table 9.48 from Figure 9.10.

Table 9.48 Frequent 2-itemsets, i.e., L2

L2	*Discussion*	*New Rules Identified*
I1, I2	This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules discussed in Table 9.47.	NIL
I1, I3	This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I1, I3. I1→I3, Confidence = S(I1∩I3) /S(I1) = 4/6 = 66% I3→I1, Confidence = S(I3∩I1) /S(I3) = 4/5 = 80%	Since, I3→I1 has confidence more than the threshold limit. So, the new rule identified is I3→I1.
I1, I5	This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules.	NIL
I2, I3	This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I2, I3. I2→I3, Confidence = S(I2∩I3) /S(I2) = 4/7 = 57% I3→I2, Confidence = S(I2∩I3) /S(I3) = 4/5 = 80%	I3→I2
I2, I4	This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I2, I4. I2→I4, Confidence = S(I2 I4) /S(I2) = 2/7 = 28% I4→I2, Confidence = S(I4 I2) /S(I4) = 2/3 = 66%	NIL, because the confidence of both the rules is less than threshold limit; so, no new rule has been found.
I2, I5	This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules.	NIL

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Here, we have identified two new rules by considering L2 and these are I3→I1 and I3→I2, both of which has its confidence more than threshold limit of 70%.

The final association rules having support more than 20% and confidence more than 70% are given below.

I5→I1, I2
I2, I5→I1
I1, I5→I2
I5→I1
I5→I2
I3→I1
I3→I2.

Example: Consider a store having 16 items for sale as listed in Table 9.49. Now consider the 25 transactions on the sale of these items as given in Table 9.50. As usual, each row in the table represents one transaction, that is, the items bought by one customer. In this example, we want to find association rules that satisfy the requirement of 25% support and 70% confidence.

Table 9.49 List of grocery items

Item number	Item name
1	Biscuit
2	Bournvita
3	Bread
4	Butter
5	Coffee
6	Cornflakes
7	Chocolate
8	Curd
9	Eggs
10	Jam
11	Juice
12	Milk
13	Rice
14	Soap
15	Sugar
16	Tape

In Table 9.50 showing 25 transactions, we list the names of items of interest but it would be more storage efficient to use item numbers.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.50 Transaction data

TID	*Items*
1	Biscuit, Bournvita, Butter, Cornflakes, Tape
2	Bournvita, Bread, Butter, Cornflakes
3	Butter, Coffee, Chocolate, Eggs, Jam
4	Bournvita, Butter, Cornflakes, Bread, Eggs
5	Bournvita, Bread, Coffee, Chocolate, Eggs
6	Jam, Sugar
7	Biscuit, Bournvita, Butter, Cornflakes, Jam
8	Curd, Jam, Sugar
9	Bournvita, Bread, Butter, Coffee, Cornflakes
10	Bournvita, Bread, Coffee, Chocolate, Eggs
11	Bournvita, Butter, Eggs
12	Bournvita, Butter, Cornflakes, Chocolate, Eggs
13	Biscuit, Bournvita, Bread
14	Bread, Butter, Coffee, Chocolate, Eggs
15	Coffee, Cornflakes
16	Chocolate
17	Chocolate, Curd, Eggs
18	Biscuit, Bournvita, Butter, Cornflakes
19	Bournvita, Bread, Coffee, Chocolate, Eggs
20	Butter, Coffee, Chocolate, Eggs
21	Jam, Sugar, Tape
22	Bournvita, Bread, Butter, Cornflakes
23	Coffee, Chocolate, Eggs, Jam, Juice
24	Juice, Milk, Rice
25	Rice, Soap, Sugar

Computing C1

All 16 items are candidate items for a frequent single-itemset. So, all the items listed in Table 9.49 will act as C1.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Computing L1

To find the frequent single itemset, we have to scan the whole database to count the number of times each of the 16 items has been sold. Scanning transaction database only once, we first set up the list of items and one transaction at a time, update the count of every item that appears in that transaction as shown in Table 9.51.

Table 9.51 Frequency count for all items

Item no.	*Item name*	*Frequency*
1	Biscuit	4
2	Bournvita	13
3	Bread	9
4	Butter	12
5	Coffee	9
6	Cornflakes	9
7	Chocolate	10
8	Curd	2
9	Eggs	11
10	Jam	6
11	Juice	2
12	Milk	1
13	Rice	2
14	Soap	1
15	Sugar	4
16	Tape	2

The items that have the necessary support (25% support in 25 transactions) must occur in at least 7 transactions. The frequent 1-itemset or L1 is now given in Table 9.52.

Table 9.52 The frequent 1-itemset or L1

Item	*Frequency*
Bournvita	13
Bread	9
Butter	12

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Item	*Frequency*
Coffee	9
Cornflakes	9
Chocolate	10
Eggs	11

Computing C2

C2 = L1 JOIN L1

This will produce 21 pairs as listed in Table 9.53. These 21 pairs are the only ones worth investigating since they are the only ones that could be frequent.

Table 9.53 The 21 candidate 2-itemsets or C2

{Bournvita, Bread}
{Bournvita, Butter}
{Bournvita, Coffee}
{Bournvita, Cornflakes}
{Bournvita, Chocolate}
{Bournvita, Eggs}
{Bread, Butter}
{Bread, Coffee}
{Bread, Cornflakes}
{Bread, Chocolate}
{Bread, Eggs}
{Butter, Coffee}
{Butter, Cornflakes}
{Butter, Chocolate}
{Butter, Eggs}
{Coffee, Cornflakes}
{Coffee, Chocolate}
{Coffee, Eggs}
{Cornflakes, Chocolate}
{Cornflakes, Eggs}
{Chocolate, Eggs}

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Computing L2

This step is the most resource intensive step. We scan the transaction database once and look at each transaction and find out which of the candidate pairs occur in that transaction. This then requires scanning the list C2 for every pair of items that appears in the transaction that is being scanned. Table 9.54 presents the number of times each candidate 2-itemset appears in the transaction database example given in Table 9.50.

Table 9.54 Frequency count of candidate 2-itemsets

Candidate 2-itemset	*Frequency*
{Bournvita, Bread}	8
{Bournvita, Butter}	9
{Bournvita, Coffee}	4
{Bournvita, Cornflakes}	8
{Bournvita, Chocolate}	4
{Bournvita, Eggs}	6
{Bread, Butter}	5
{Bread, Coffee}	5
{Bread, Cornflakes}	3
{Bread, Chocolate}	4
{Bread, Eggs}	5
{Butter, Coffee}	4
{Butter, Cornflakes}	8
{Butter, Chocolate}	4
{Butter, Eggs}	6
{Coffee, Cornflakes}	2
{Coffee, Chocolate}	7
{Coffee, Eggs}	7
{Cornflakes, Chocolate}	1
{Cornflakes, Eggs}	2
{Chocolate, Eggs}	9

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

From this list we can find the frequent 2-itemsets or the set L2 by selecting only those item pairs from Table 9.54 given above, having a frequency equal to or more than 7. Therefore the list of frequent-2 itemsets or L2 is given in Table 9.55. Usually L2 is smaller than L1 but in this example they both have seven itemsets and each member of L1 appears as a subset of some member of L2. This is unlikely to be true when the number of items is large.

Table 9.55 The frequent 2-itemsets or L2

Frequent 2-itemset	Frequency
{Bournvita, Bread}	8
{Bournvita, Butter}	9
{Bournvita, Cornflakes}	8
{Butter, Cornflakes}	8
{Coffee, Chocolate}	7
{Coffee, Eggs}	7
{Chocolate, Eggs}	9

Computing C3

C3 = L2 JOIN L2 (First item should be common)

To find the candidate 3-itemsets or C3, we combine the appropriate frequent 2-itemsets from L2 (these must have the same first item) and obtain four such itemsets as given in Table 9.56.

Table 9.56 Candidate 3-itemsets or C3

Candidate 3-itemset
{Bournvita, Bread, Butter}
{Bournvita, Bread, Cornflakes}
{Bournvita, Butter, Cornflakes}
{Coffee, Chocolate, Eggs}

Pruning C3

Before processing further to calculate the frequency of each 3-item pairs, it is important to apply the Apriori property to prune the candidate 3-itemsets as shown in Table 9.57.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.57 Pruning of candidate itemset C3

Candidate Item Pair	*Discussion*	*Qualify or Not*
{Bournvita, Bread, Butter}	Its subsets are (Bournvita, Bread), (Bournvita, Butter) and (Bread, Butter). Out of these subsets (Bread, Butter) is not frequent as it is not in L2 given in Table 9.55. Thus, {Bournvita, Bread, Butter} will be removed from C3.	Not Qualified, so will be pruned
{Bournvita, Bread, Cornflakes}	Its subsets are (Bournvita, Bread), (Bournvita, Cornflakes) and (Bread, Cornflakes). Out of these subsets, (Bread, Cornflakes) is not frequent as it is not in L2. Thus, {Bournvita, Bread, Cornflakes} will be removed from C3.	Not Qualified, so will be pruned
{Bournvita, Butter, Cornflakes}	Its subsets are (Bournvita, Butter), (Bournvita, Cornflakes) and (Butter, Cornflakes). Here all subsets are frequent as they are in L2. Thus, {Bournvita, Butter, Cornflakes} will be retained in C3.	Qualified
{Coffee, Chocolate, Eggs}	Its subsets are (Coffee, Chocolate), (Coffee, Eggs) and (Chocolate, Eggs). Here, all subsets are frequent as they are in L2. Thus, {Coffee, Chocolate, Eggs} will be retained in C3.	Qualified

Thus, pruned C3 will be as shown in Table 9.58.

Table 9.58 Pruned candidate itemset C3

{Bournvita, Butter, Cornflakes}
{Coffee, Chocolate, Eggs}

As observed, this pruning step has reduced the candidate items from 4 to 2 and it improves the performance of the algorithm. Table 9.59 shows candidate 3-itemsets or C3 and their frequencies.

Table 9.59 Candidate 3-itemsets or C3 and their frequencies

Candidate 3-itemset	*Frequency*
{Bournvita, Butter, Cornflakes}	8
{Coffee, Chocolate, Eggs}	7

Computing L3

Table 9.60 now presents the frequent 3-itemsets or L3. Note that only one member of L2 is not a subset of any itemset in L3. That 2-itemset is {Bournvita, Bread}.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.60 The frequent 3-itemsets or L3

Frequent 3-itemset	*Frequency*
{Bournvita, Butter, Cornflakes}	8
{Coffee, Chocolate, Eggs}	7

Computing C4

C4 = L3 JOIN L3 (First two items should be common)

Since, no first two items are common, thus, C4 is null.

This is the end of frequent itemset generation since there cannot be any candidate 4-itemsets.

Finding the rules

The rules obtained from frequent 3-itemsets would always include some rules that we could obtain from L2 embedded in them, but some additional rules will be obtainable from itemsets in L2 which are not subsets of itemsets in L3, for example, {Bournvita, Bread} in this case.

Here, the frequent itemset is {Bournvita, Butter, Cornflakes} whose resulting subsets are as followed:

{Bournvita},{Butter},{Cornf lakes},{Bournvita, Butter},{Butter, Cornf lakes},{Bournvita, Cornflakes} [represented by *s*, and the superset or whole list {Bournvita, Butter, Cornflakes} is represented as *l*, and the rules will be generated in the form of $s \rightarrow l\text{-}s$].

The rules generated from {Bournvita, Butter, Cornflakes} are as follows.

Bournvita→Butter, Cornflakes

Butter→Bournvita, Cornflakes

Cornflakes→Bournvita, Butter

Bournvita, Butter→Cornflakes

Bournvita, Cornflakes→Butter

Butter, Cornflakes→Bournvita

Now, let us calculate the confidence of these rules as shown in Table 9.61.

Table 9.61 Confidence of association rules from {Bournvita, Butter, Cornflakes}

Rule	*Support of Both together*	*Frequency of Antecedent LHS (Refer to Table 9.52 and 9.55)*	*Confidence*	*Whether Confidence is more than threshold limit or not*
Bournvita→Butter, Cornflakes	8	13	8/13 = 0.61	No
Butter→Bournvita, Cornflakes	8	12	8/12 = 0.67	No
Cornflakes→Bournvita, Butter	8	9	8/9 = 0.89	Yes
Bournvita, Butter→Cornflakes	8	9	8/9 = 0.89	Yes
Bournvita, Cornflakes→Butter	8	8	8/8 = 1.0	Yes
Butter, Cornflakes→Bournvita	8	8	8/8 = 1.0	Yes

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

We also have two rules from L2 for a frequent item pair {Bournvita, Bread} let us calculate its confidence as shown in Table 9.62.

Table 9.62 Confidence of association rules from {Bournvita, Bread}

Rule	*Support of Both together*	*Frequency of Antecedent LHS (Refer to Tables 9.52 and 9.55)*	*Confidence*	*Whether Confidence is more than threshold limit or not*
Bournvita →Bread	8	13	8/13 = 0.61	No
Bread→Bournvita	8	9	8/9 = 0.89	Yes

Thus, from this discussion we have found out five rules having confidence more than 70% as shown below in Table 9.63.

Table 9.63 Identified rules from {Bournvita, Butter, Cornflakes} having confidence more than 70%

Cornflakes→Bournvita, Butter
Bournvita, Butter→Cornflakes
Bournvita, Cornflakes→Butter
Butter, Cornflakes→Bournvita
Bread→Bournvita

Now, let us find all possible rules from these identified rules as shown in Table 9.64.

Table 9.64 List of all possible rules from rules given in Table 9.61

Rules	*Discussion*	*Rules having confidence more than the threshold value*
Cornflakes→ Bournvita, Butter	It has confidence 0.89 as discussed in Table 9.61. [Formula = S (Cornflakes, Bournvita, Butter) /S(Cornflakes)]	Cornflakes → Bournvita, Butter
Cornflakes → Bournvita	It is intuitive that this rule will have confidence greater than the threshold limit as its formula is S (Cornflakes, Bournvita) /S(Cornflakes). Since, support of subset, i.e, Cornflakes, Bournvita, should be equal to or more than superset, Cornflakes, Bournvita, Butter so this rule qualifies. (Resulting confidence = 0.89)	Cornflakes → Bournvita

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Rules	*Discussion*	*Rules having confidence more than the threshold value*
Cornflakes→Butter	It is also intuitive that this rule will have confidence greater than the threshold limit as its formula is S (Cornflakes, Butter) / S(Cornflakes). Since, support of subset, i.e, Cornflakes, Butter should be equal to or more than superset, Cornflakes, Bournvita, Butter so this rule qualifies. (Resulting confidence = 0.89)	Cornflakes → Butter
Bournvita, Butter→ Cornflakes	It has confidence 0.89 as shown in Table 9.61. [Formula = S (Bournvita, Butter, Cornflakes) / S(Bournvita, Butter)]	Bournvita, Butter→ Cornflakes
Bournvita → Cornflakes	It is not intuitive that this rule will have confidence more than the threshold limit as its formula is S (Bournvita, Cornflakes) / S(Bournvita). So, let us calculate its confidence. Confidence= S (Bournvita, Cornflakes) / S(Bournvita) = 8/13 = 0.61. This rule has confidence less than threshold limit, so it will be discarded.	Discarded
Butter→ Cornflakes	It is not intuitive that this rule will have confidence more than the threshold limit as its formula is S (Butter, Cornflakes)/S(Butter). So, let us calculate its confidence. Confidence = S (Butter, Cornflakes) / S(Butter) = 8/12 = 0.67 This rule has confidence less than threshold limit, so it will be discarded.	Discarded
Bournvita, Cornflakes→Butter	It has confidence 1.0 as shown in Table 9.61.	Bournvita, Cornflakes → Butter
Bournvita→Butter	It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = S (Bournvita, Butter) / S(Bournvita) = 9/13 = 0.69 This rule has confidence less than threshold limit, so it will be discarded.	Discarded
Cornflakes→Butter	It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = S (Cornflakes, Butter) / S(Cornflakes)= 8/9 = 0.88 This rule has confidence more than threshold limit, so it will be selected.	Cornflakes → Butter

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Rules	*Discussion*	*Rules having confidence more than the threshold value*
Butter, Cornflakes → Bournvita	It has confidence 1.0 as shown in Table 9.61.	Butter, Cornflakes → Bournvita
Butter→ Bournvita	It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = S (Butter, Bournvita) / S(Butter) = 8/12 =0.67 This rule has confidence less than threshold limit, so it will be discarded.	Discarded
Cornflakes → Bournvita	It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = S (Cornflakes, Bournvita) / S(Cornflakes)= 8/9 = 0.89 This rule has confidence more than threshold limit, so it will be selected.	Cornflakes → Bournvita

Similarly, we will examine the other 3-itemset {Coffee, Chocolate, Eggs}.

Table 9.65 Confidence of association rules from {Coffee, Chocolate, Eggs}

Rule	*Support*	*Frequency of LHS (Antecedent)*	*Confidence*
Coffee→Chocolate, Eggs	7	9	0.78
Chocolate→Coffee, Eggs	7	10	0.70
Eggs→Coffee, Chocolate	7	11	0.64
Chocolate, Eggs→Coffee	7	9	0.78
Coffee, Eggs→Chocolate	7	7	1.0
Coffee, Chocolate→Eggs	7	7	1.0

Again, the confidence of all these rules, except the third, is higher than 70%, and so five of them also qualify.

List of all possible rules for three item pairs {Coffee, Chocolate, Eggs} has been explained in Table 9.66, given below.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.66 List of all possible rules from rules given in Table 9.65

Rules	*Discussion*	*Rules having confidence more than the threshold value*
Coffee→Chocolate, Eggs	It has confidence of 0.78 as shown in Table 9.65.	Coffee→Chocolate, Eggs
Coffee→Eggs	It is intuitive from rule Coffee→Chocolate, Eggs	Coffee→Eggs
Chocolate→Coffee, Eggs	It has confidence of 0.70 as shown in Table 9.65.	Chocolate→Coffee, Eggs
Chocolate→Coffee	It is intuitive from rule Chocolate→Coffee, Eggs	Chocolate→Coffee
Chocolate→Eggs	It is intuitive from rule Chocolate→Coffee, Eggs	Chocolate→Eggs
Chocolate, Eggs→Coffee	It has confidence of 0.78 as shown in Table 9.65.	Chocolate, Eggs→Coffee
Chocolate→Coffee	It is non intuitive, so let us calculate its confidence. Confidence = S(Chocolate, Coffee) / S(Coffee) = 7/9 = 0.77 Since, confidence is more than the threshold so this rule will be selected.	Chocolate→Coffee
Eggs→Coffee	It is non intuitive, so let us calculate its confidence. Confidence = S(Eggs, Coffee) / S(Eggs) = 7/11 = 0.64 Since, confidence is less than the threshold so this rule will be discarded.	Discarded
Coffee, Eggs→Chocolate	It has confidence 1.0 as discussed in Table 9.65.	Coffee, Eggs→Chocolate
Coffee→Chocolate	It is non intuitive, so let us calculate its confidence. Confidence = S(Coffee, Chocolate) / S(Coffee) =7/9 = 0.77 Since, confidence is more than the threshold so this rule will be selected.	Coffee→Chocolate
Eggs→Chocolate	It is non intuitive, so let us calculate its confidence. Confidence = S(Eggs, Chocolate) /S(Eggs) = 9/11= 0.81 Since, confidence is more than the threshold so this rule will be selected.	Eggs→Chocolate

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Rules	*Discussion*	*Rules having confidence more than the threshold value*
Coffee, Chocolate→Eggs	It has confidence 1.0 as shown in Table 9.65.	Coffee, Chocolate→Eggs
Coffee→Eggs	This rule is non intuitive, but it is already selected as given in row 2 from rule Coffee→Chocolate, Eggs	Coffee→Eggs
Chocolate→Eggs	This rule is non intuitive, but it is already selected as given in row 5 from rule Chocolate→Coffee, Eggs	Chocolate→Eggs

Thus, all association mining rules for the given database and having confidence of more than 70% are listed in Table 9.67.

Table 9.67 All association rules for the given database

Rule No.	*Rule*	*Confidence*
1	Bread→Bournvita	0.89
2	Cornflakes→ Bournvita, Butter	0.89
3	Cornflakes→ Bournvita	0.89
4	Cornflakes→ Butter	0.89
5	Bournvita, Butter→ Cornflakes	0.89
6	Bournvita, Cornflakes→ Butter	1
7	Cornflakes→ Butter	0.89
8	Butter, Cornflakes→ Bournvita	1
9	Coffee→Chocolate, Eggs	0.78
10	Coffee→Chocolate	0.78
11	Coffee→Eggs	0.78
12	Chocolate→Coffee, Eggs	0.7
13	Chocolate→Coffee	0.7
14	Chocolate→Eggs	0.9
15	Chocolate, Eggs→Coffee	1
16	Coffee, Eggs→Chocolate	1
17	Eggs→Chocolate	0.82
18	Coffee, Chocolate→Eggs	1

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

9.8 Closed and Maximal Itemsets

To further improve the efficiency of the association mining process, frequent itemsets can be divided into 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 as shown in Figure 9.13.

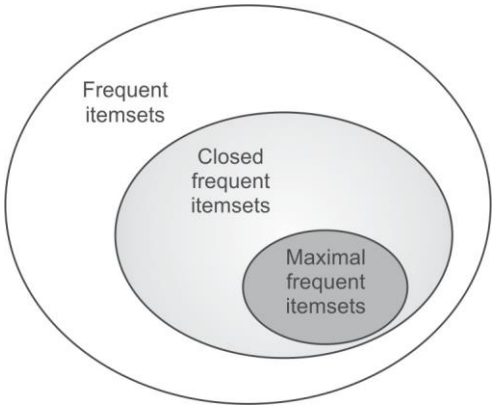


Figure 9.13 Illustration of closed and maximal frequentitemsets

The concept of Closed and maximal itemsets have been explained with an example database.

Example: Consider the transaction database in Table 9.68 with minimum support required to be 40%.

Table 9.68 A transaction database to illustrate closedand maximal itemsets

100	Butter, Curd, Jam
200	Butter, Curd, Jam, Muffin
300	Curd, Jam, Nuts
400	Butter, Jam, Muffin, Nuts
500	Muffin, Nuts

There are a total of 14 frequent itemsets as shown in Table 9.69 and the closed and maximal itemsets for the given database have also been presented there.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.69 Frequent itemsets for the database in Table 9.68

Itemset	*Support*	*Closed?*	*Maximal?*	*Both?*
{Butter}	3	No, there exists a superset of Butter with the same support 3, i.e., {Butter, Jam}	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Curd}	3	No, there exists a superset of Curd with the same support 3, {Curd, Jam}	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Jam}	4	Yes, there exists no superset of Jam with the same support 4	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Muffin}	3	Yes, there exists no superset of Muffin with the same support 3	No, it is a proper subset of another frequent itemset, i.e, {Muffin, Nuts}	No
{Nuts}	3	Yes, there exists no superset of Nuts with the same support 3	No, it is a proper subset of another frequent itemset, i.e, {Muffin, Nuts}	No
{Butter, Curd}	2	No, there exists a superset of Butter, Jam with the same support 2, i.e., {Butter, Curd, Jam}	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Butter, Jam}	3	Yes, there exists no superset of Butter, Jam with the same support 3	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Butter, Muffin}	2	No, there exists a superset of Butter, Muffin with the same support 2, i.e., {Butter, Jam, Muffin}	No, it is a proper subset of another frequent itemset, i.e, {Butter, Jam, Muffin}	No
{Curd, Jam}	3	Yes, there exists no superset of Curd, Jam with the same support 3	No, it is a proper subset of another frequent itemset, i.e, {Butter, Curd, Jam}	No
{Jam, Muffin}	2	No, there exists a superset of Jam, Muffin with the same support 2, i.e., {Butter, Jam, Muffin}	No, it is a proper subset of another frequent itemset, i.e, {Butter, Jam, Muffin}	No
{Jam, Nuts}	2	Yes, there exists no superset of Jam, Nuts with the same support 2	Yes, it is not proper subset of any other frequent itemset	Yes

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Itemset	Support	Closed?	Maximal?	Both?
{Muffin, Nuts}	2	Yes, there exists no superset of Muffin, Nuts with the same support 2	Yes, it is not propersubset of any other frequent itemset	Yes
{Butter, Curd, Jam}	2	Yes, there exists no superset of Butter, Curd, Jam with the same support 2	Yes, it is not propersubset of any other frequent itemset	Yes
{Butter, Jam, Muffin}	2	Yes, there exists no superset of Butter, Jam, Muffin with the same support 2	Yes, it is not propersubset of any other frequent itemset	Yes

One can observe that there are 9 closed frequent itemsets and 4 maximal frequent itemsets as shown in Table 9.69.

It is important to note that closed and maximal frequent itemsets are often smaller than all frequent itemsets and association rules can be generated directly from them. Thus, it is desirable that instead of mining every frequent itemset, an efficient algorithm should mine only maximal frequent itemsets. The association of frequent itemsets, closed frequent itemsets and maximal frequent itemsets has been illustrated in Figure 9.16 for better understanding. From this one can observe that a maximal itemset is a closed itemset, but a closed itemset is not necessarily a maximal itemset.

9.9 The Apriori–TID Algorithm for Generating Association Mining Rules

Apriori–TID uses the Apriori algorithm to generate the candidate and frequent itemsets, but the difference is that it uses the TID or vertical storage approach (discussed in Section 9.6.3) for its implementation.

The working of the Apriori–TID algorithm is given as follows.

1. Scan the whole transaction database to convert it into TID storage and represent it as T1 (i.e., each entry of T1 consists of all items in the transaction along with the corresponding TID).
2. Calculate the frequent 1-itemset L1 using T1.
3. Obtain C2 by applying the joining principle discussed under the Apriori algorithm.
4. Calculate the support for the candidates in C2 with the help of T1 by intersecting corresponding rows and a new TID dataset is created and represent it as T2 containing two item pairs.
5. Generate L2 from C2 by the usual means and then generate C3 from L2, again by using the joining principle.
6. Generate T3 using T2 and C3. This process is repeated until the set of candidate k-itemsets is an empty set.

Example of Apriori–TID: Let us understand this concept with an example database of transactions given in Table 9.70 for finding frequent itemsets with the Apriori–TID algorithm. Here, minimum support is 50% and minimum confidence is 75%.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.70 Transaction database

100	Butter, Curd, Eggs, Jam
200	Butter, Curd, Jam
300	Butter, Muffin, Nuts
400	Butter, Jam, Muffin
500	Curd, Jam, Muffin

Step 1:

Scan the entire transaction database to convert it into TID storage represented as T1 as given in Table 9.71.

Table 9.71 Transaction database T1

	100	*200*	*300*	*400*	*500*
Butter	1	1	1	1	0
Curd	1	1	0	0	1
Eggs	1	0	0	0	0
Jam	1	1	0	1	1
Muffin	0	0	1	1	1
Nut	0	0	1	0	0

Step 2:

Frequent 1-itemset L1 is calculated using T1.

Since, the minimum support is 3, all items have been selected as frequent items having support equal or more than threshold limit of support, i.e., 3 with the help of T1.

Thus, L1 is as shown in Table 9.72.

Table 9.72 L1

Itemset	Support
Butter	4
Curd	3
Jam	4
Muffin	3

Step 3:

C2 is obtained by applying the joining principle, i.e., L1 JOIN L1

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Thus, C2 will be as given in Table 9.73.

Table 9.73 C2

Itemsets
Butter, Curd
Butter, Jam
Butter, Muffin
Curd, Jam
Curd, Muffin
Jam, Muffin

Step 4:

Support for the candidates in C2 is then calculated with the help of T1 by intersecting corresponding rows and a new TID dataset is created represented as T2 containing two item pairs as illustrated in Table 9.74.

Table 9.74 Transaction database T2

	100	200	300	400	500
Butter, Curd	1	1	0	0	0
Butter, Jam	1	1	0	1	0
Butter, Muffin	0	0	1	1	0
Curd, Jam	1	1	0	0	1
Curd, Muffin	0	0	0	0	1
Jam, Muffin	0	0	0	1	1

Thus, support for C2 is given in Table 9.75.

Table 9.75 Support for C2

Itemsets	*Support*
Butter, Curd	2
Butter, Jam	3
Butter, Muffin	2
Curd, Jam	3
Curd, Muffin	1
Jam, Muffin	2

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Step 5:

L2 is then generated from C2 as shown in Table 9.76.

Table 9.76 L2

Itemsets	*Support*
Butter, Jam	3
Curd, Jam	3

Thus, {Butter, Jam} and {Curd, Jam} are the frequent pairs and they generate L2. C3 may also be generated but it is observed that C3 is empty because the first item is not common in the L2 item .pairs. If C3 was not empty then it would be used to identify T3 using the transaction set T2. That would result in a smaller T3 and it might have resulted in the removal of a transaction or two. In the example database, all the records of T2 have been removed and T3 is null.

The generation of association rules from the derived frequent set can be done in the usual way.

Advantages of the Apriori–TID algorithm

The advantage of Apriori–TID algorithm is in the storage structure because it facilitates counting of items by counting the number of 1s in each row and the number of 2-itemsets can be counted by finding the intersection of the two rows.

The other main advantage of the Apriori–TID algorithm is that the size of T_k is generally smaller than the transaction database.

Because the support for each candidate k-itemset is counted using the corresponding T_k, therefore this algorithm runs faster than the basic Apriori algorithm. It is important to note that both Apriori and Apriori–TID use the same candidate generation algorithm, and therefore they count the same itemsets.

9.10 Direct Hashing and Pruning (DHP)

A major shortcoming of the Apriori algorithm is that it cannot prune 2 itemset candidate items. As C2 is derived from L1 by using following formula:

C2 = L1 Join L1.

Since, every subset of C2 is a member of L1, C2 cannot be pruned by the Apriori property, i.e., subset of frequent itemsets should also be frequent. In case of C2, all the subsets are frequent, i.e., member of L1, so it cannot be pruned. However, Apriori property is very helpful in pruning higher order candidate itemsets like C3 or C4.

The DHP algorithm is useful as it overcomes this limitation since it has the capability to prune C2; this is important to improve performance.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Working of the DHP algorithm

This algorithm employs a hash-based technique for reducing the count of candidate itemsets generated in the first pass (i.e., a significantly smaller C2 is generated). The number of itemsets in C2 generated using DHP can be smaller in magnitude; therefore the scan to identify L2 is more efficient.

The steps to be taken to operate DHP are:

1. Find all frequent 1-itemsets and all the candidate 2-itemsets by using the same steps as the Apriori algorithm, i.e., C2 is generated by L1 JOIN L1.
2. Generate all possible 2-itemsets in each transaction and assign a code to each item and itemsets.
3. All the possible 2-itemsets are hashed to a hash table by using the code of each itemset. For hashing, a hash function is applied on each code and a bucket is assigned to each itemset based on the output of the hash function. The count of each bucket in the hash table is increased by one, each time when an itemset is hashed to that bucket. When different itemsets are hashed to the same bucket then collisions can occur. To assign a f lag for each bucket, a bit vector is associated with the hash table. If the bucket count is equal or above the minimum required support count, then accordingly f lag in the bit vector is set to 1, otherwise it is set to 0.
4. In this step C2 is pruned. Each item pair of C2 generated in step 1 is checked for the bit vector of its corresponding bucket to which it was assigned in step 3. If its bit vector is 0, then the item pair will be pruned and removed from C2, while all those item pairs having a bit vector of 1 are retained in C2. This step helps to prune C2 and this pruning of C2 is the major objective of the DHP algorithm.
   It is important to note that, having the corresponding bit vector or bit set does not guarantee that the itemset is frequent; this is due to collisions. The hash table filtering does reduce C2, significantly.
5. Then, from the pruned C2, frequent item pairs, L2 is generated and process continues as in Aproiroi property.
6. In this step, the hash table for the next step is generated. Only those itemsets that are frequent, i.e. in L2, are kept in the database. The pruning not only trims each transaction by removing the unwanted itemsets but also removes transactions that have no itemsets that could be frequent.
7. The algorithm is continued till no more candidate itemsets are identified. The whole transaction database is scanned to identify the support count for each itemset despite using a hash table to identify the frequent itemsets. Now, the database seems relatively smaller because of the pruning. The algorithm identifies the frequent itemsets as earlier by checking against the minimum support when the support count is calculated. The algorithm then produces candidate itemsets similar to what the Apriori algorithm does.

To get a clear idea about the working of DHP algorithm, let us consider some case studies by way of examples. After going through these examples, this algorithm will become clear to you.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Example: Using the DHP algorithm

To understand this concept, let us identify the association rules satisfying 50% support and 75% confidence for the transaction database given in Table 9.77 with the DHP algorithm.

Table 9.77 Transaction database

100	Butter, Curd, Eggs, Jam
200	Butter, Curd, Jam
300	Butter, Muffin, Nuts
400	Butter, Jam, Muffin
500	Curd, Jam, Muffin

Let us first find all frequent 1-itemsets and in this case it will all those items that appear 3 or more times in the database. The L1 is given in Table 9.78.

Table 9.78 Frequent 1-itemset L1

Butter
Curd
Jam
Muffin

Then, generate C2 from L1, i.e., L1 JOIN L1. This is given in Table 9.78. Here, for simplicity we are representing each item by its first alphabet.

Table 9.79 Candidate 2 itemsets C2

BC ('B' for Butter and 'C' for Curd)
BJ
BM
CJ
CM
JM

Since, the size of C2 is 6 and the major advantage of DHP is to prune C2 so hashing is applied on each item pair as discussed below.

To apply hashing, the next step is to generate all possible 2-itemsets for each transaction as shown in Table 9.80.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.80 Possible 2-itemsets for each transaction

100	(Butter, Curd), (Butter, Eggs), (Butter, Jam), (Curd, Eggs), (Curd, Jam), (Eggs, Jam)
200	(Butter, Curd), (Butter, Jam), (Curd,Jam)
300	(Butter, Muffin), (Butter, Nuts), (Muffin,Nuts)
400	(Butter, Jam), (Butter, Muffin), (Jam, Muffin)
500	(Curd, Jam), (Curd, Muffin), (Jam, Muffin)

For a support of 50%, the frequent items are Butter, Curd, Jam, and Muffin as shown in Table 9.81. Now, to hash the possible 2-itemsets, a code is assigned to each item as given in Table 9.82.

Table 9.81 Frequent itemsets for a support of 50%

Bread
Curd
Jam
Muffin

Table 9.82 Code for each item

Code	*Item Name*
1	Butter
2	Curd
3	Eggs
4	Jam
5	Muffin
6	Nuts

For each pair, a numeric value is obtained by representing Butter by 1, Curd by 2 and so on as shown in the above table. Then, each pair is represented by a two-digit number, for example, (Butter, Curd) is represented by 12; (Curd, Jam) is represented by 24 and so on.

The coded value for each item pair is given in Table 9.83.

Table 9.83 Coded representation for each item pair

100	12, 13, 14, 23, 24, 34
200	12, 14, 24
300	15, 16, 56
400	14, 15, 45
500	24, 25, 45

Now, the hash function is applied on coded values so that these items can be assigned to different buckets. Here, we are using modulo 8 number, i.e., dividing the code by 8 and using the remainder as hash function for the given dataset.

Hash function: Modulo 8, i.e., divide the concerned item pair code by 8 and use the remainder as its bucket number.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

The guidelines to decide about hash function have been discussed later. For the moment, let's apply hash function to each item pair and assign the item pairs to each bucket on the basis of modulo 8 as shown in Table 9.60.

Applying the hash function to each item pair in transaction 100, the first item pair is 12, and when modulo 8 hash function is applied on 12, it gives a remainder of 4, so 12 is assigned to bucket 4. The next item pair in transaction 100 is 13, and gives a remainder of 5 after dividing it by 8, so 13 is assigned to bucket 5. Similarly, 14 has a remainder of 6, 23 has a remainder of 7, 24 has the remainder 0 and 34 has a remainder of 2, so these item pairs are assigned corresponding bucket numbers as shown in Table 9.84. This process is repeated for each transaction and every item pair is assigned to its corresponding bucket.

Table 9.84 Assigning item pairs to buckets based on hash function modulo 8

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*Bucket address*
24 24 16 56 24	25	34		12 12	13 45 45	14 14 14	23 15 15	Item pairs
5	1	1	0	2	3	3	3	Support of bucket
1	0	0	0	0	1	1	1	Bit Vector (it refers to a sequence of numbers which are either 0 or 1 and stored contiguously in memory)

After assigning every item pair to its corresponding bucket, the support or count for each bucket has been calculated and if its support is equal to the threshold value then the corresponding bit vector is set 1 as shown in Table 9.84.

Now, C2 is pruned by checking the bit vector of the corresponding bucket for each candidate item pair. And only those item pairs are retained in C2 which have the corresponding bit vectoras 1. The process of pruning of C2 has been illustrated in Table 9.85.

It has been noted that pair BC, i.e, 12 belong to the bucket with support of 2, since its support is less than threshold value of support so its corresponding bit vector is 0 and this item pair has been removed from C2; in other words, C2 has been pruned.

Similarly, this analysis is performed for each item pair of C2 as shown in Table 9.85.

Table 9.85 Pruning of C2

Item Pairs in C2	*Bucket*	*Bit Vector*	*Remarks*	*Pruned C2*
BC, i.e., 12	4	0	Removed from C2	
BJ, i.e., 14	6	1	Will be retained	BJ
BM, i.e., 15	7	1	Will be retained	BM
CJ, i.e., 24	0	1	Will be retained	CJ
CM, i.e., 25	1	0	Removed from C2	
JM, i.e., 45	5	1	Will be retained	JM

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

These candidate pairs are then hashed to the hash table and those candidate pairs are removed that are hashed to locations where the bit vector bit is not set. Table 9.85 illustrates that (B, C) and (C, M) have been eliminated from C2. Now, only four candidate item pairs are left instead of six item pairs in C2 and thus, C2 has been pruned as given in the last column of Table 9.85. After that, analyze the transaction database and modify it to include only these candidate pairs as given in Table 9.86.

Now, from the pruned C2, let us find L2 as shown in Table 9.86.

Table 9.86 Finding L2

Itemset	*Count*
BJ	3
BM	2
CJ	3
JM	2

Thus, L2 has two item pairs, i.e, BJ and CJ. Now, let us use L2 by selecting only those item pairs of the transaction database that qualify by being above the threshold value of support. The method of finding three itemsets is shown in Table 9.87.

Table 9.87 Finding three itemsets

TID	*Original Item groups*	*Selected Item pairs according to L2*	*Three Itemgroupings (sets)*
100	(Butter, Curd), (Butter, Eggs), (Butter, Jam), (Curd, Eggs), (Curd, Jam), (Eggs, Jam)	(Butter, Jam), (Curd, Jam)	NIL
200	(Butter, Curd), (Butter, Jam), (Curd, Jam)	(Butter, Jam), (Curd, Jam)	NIL
300	(Butter, Muffin), (Butter, Nuts), (Muffin, Nuts)	NIL	NIL
400	(Butter, Jam), (Butter, Muffin), (Jam, Muffin)	(Butter, Jam)	NIL
500	(Curd, Jam), (Curd, Muffin), (Jam, Muffin)	(Curd, Jam)	NIL

As shown above in Table 9.87, there are no 3-Item groups, and this complete the process of identifying frequent item pairs .So, in this case, there are two item pairs (Butter, Jam) and (Curd, Jam).We will find out the rules for selected item pairs in the same way as we did in the Apriori algorithm, previously. Proceed as follow:

These item pairs produce four rules as shown below.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Butter → Jam
Jam → Butter
Curd → Jam
Jam → Curd

Now, let us calculate their confidence as shown below.

Confidence of Butter → Jam = S (Butter, Jam) / S(Butter) = 3/4 = 0.75
Jam → Butter = S (Jam, Butter) / S(Jam) = 3/4 = 0.75
Curd → Jam = S (Curd, Jam) / S(Curd) = 3/3 = 1.0
Jam → Curd = S (Jam, Curd) / S(Jam) = 3/4 = 1.0

Based on given threshold value of confidence we can select the qualified rules.

It is important to note that, the DHP algorithm usually performs better than the Apriori algorithm during the initial stages, especially during L2 generation. Added to this improvement is also the feature that trims and prunes the transaction set progressively, thereby improving the overall efficiency of the algorithm. The startup phase of DHP does consume some extra resources when the hash table is constructed but this is outweighed by the advantages gained during the later phases when both the number of candidate itemsets and the transaction set are reduced (pruned better).

Guidelines for selection of the hash function

As one of the major objectives of the DHP algorithm is to reduce the size of C2, it is essential that the hash table is large enough so that the collisions are low, i.e., different item pairs should not be hashed into the same bucket. In case of collisions, the bit vector is not very effective because in this case more than one item pair is contributing to support for the bucket and in ideal case it should be the support of only one item pair that should contribute, so that candidate item pairs are close to frequent item pairs.

If the hash table is too small and the collisions are high then more than one itemset will hash to most of the buckets. It will result into loss of effectiveness of the hash table in reducing the number of candidate itemsets C2. This is what happened in the example above in which we had collisions in three of the eight buckets. If the minimum support count is low, it is even more important that all collisions should be avoided.

If the hash table is too large then calculation will increase as we have to calculate the values for each bucket. Thus, the hash function should be selected in such a manner that it results in the least collisions.

Increasing the efficiency of the DHP algorithm

For efficient working of DHP, it is best to ensure that the hash table resides in the main memory. If the hash table does not fit in the main memory, additional disk I/O costs will apply which may again reduce the efficiency. This algorithm is most efficient when the number of itemsets in L2 is lot smaller than C2. This reduction in database size will have a positive impact on the efficiency of the algorithm.

The efficient pruning of the database, particularly C2, helps DHP to achieve a significantly shorter execution time than the Apriori algorithm.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Some more examples of DHP and Apriori algorithms

Another example will help in comparing Apriori with the DHP algorithm. In this example, we will first apply the Apriori algorithm to identify frequent item pairs; then we will apply the DHP algorithm for the same identification to illustrate differences in the process. Here, we have to find frequent item pairs with support more than 40% for the database given in Table 9.88.

By applying both the algorithms on the same database, comparison of the workings of both algorithms will be easy.

Table 9.88 Transaction database for Apriori and DHP

TID	*Items*
100	A C D
200	B CE
300	A B CE
400	B E

The process to identify frequent item pairs through the Apriori algorithm has been illustrated in Figure 9.14.

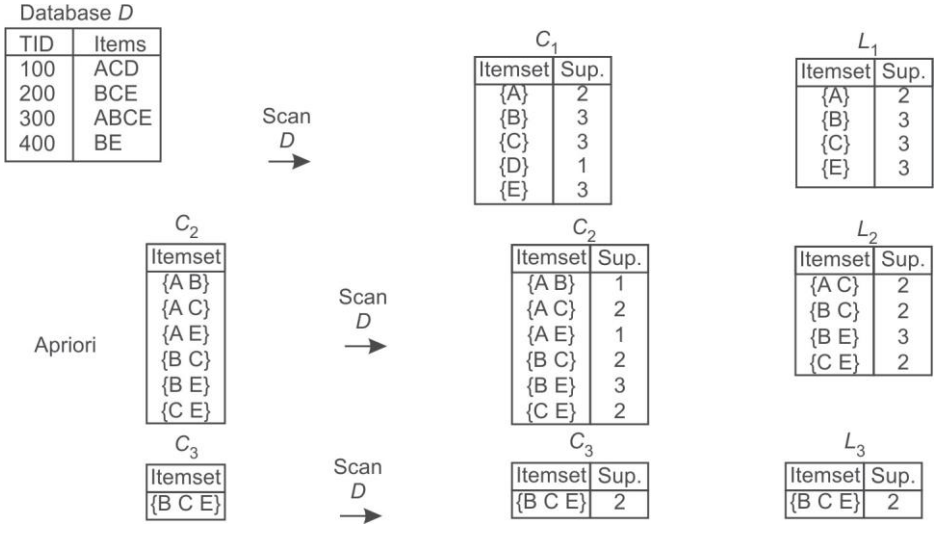


Figure 9.14 Process by the Aprioiri algorithm method

The process to identify frequent item pairs through DHP algorithm has been illustrated in Figure 9.15.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

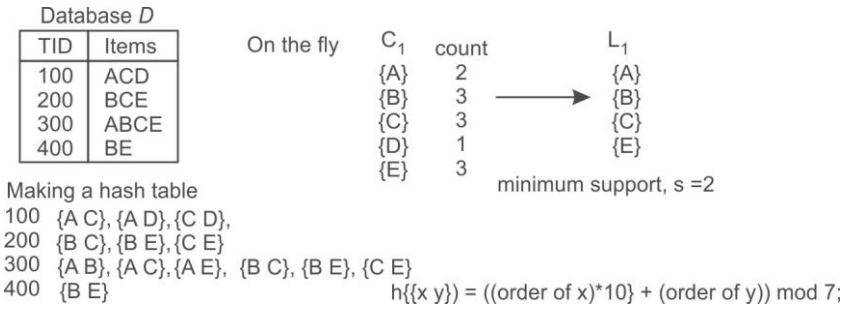


Figure 9.15 Process by the DHP algorithm method

Now, to hash the possible 2-itemsets, the code is assigned to each item. Let us assign code to each item as given in Table 9.89.

Table 9.89 Code for each item

Item	*Code*
A	1
B	2
C	3
D	4
E	5
F	6

A numeric value is attained for each pair by representing A by 1, B by 2 and so on as given in above table. Then each pair is represented by a two-digit number, for example, AC by 13 and AD by 14. The coded value for each item pair is given in Table 9.90.

Table 9.90 Coded representation for each item pair

TID	*Item pairs code*
100	13 (AC), 14(AD), 34 (CD)
200	23 (BC), 25 (BE), 35 (CE)
300	12 (AB), 13 (AC), 15 (AE), 23 (BC), 25 (BE), 35 (CE)
400	25 (BE)

Now, hash function is applied on the coded values so that these items can be assigned to different buckets. Here, we are using modulo 7 number, i.e., dividing by 7 and using the remainder as hash function for the given dataset as given in Table 9.91.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.91 Assigning of item pairs to buckets based on hash function modulo 7

0	1	2	3	4	5	6	Bucket address
14 35 35	15	23 23		25 25 25	12	13 34 13	Item pairs
3	1	2	0	3	1	3	Support of bucket
1	0	1	0	1	0	1	Bit Vector

The process of pruning C2 based on bit vector has been shown in Table 9.92.

Table 9.92 Pruning of C2

Item Pairs in C2	*Bucket*	*Bit Vector*	*Remarks*	*Pruned C2*
AB, i.e., 12	5	0	Removed from C2	
AC, i.e., 13	6	1	Will be retained	AC
AE, i.e., 15	1	0	Removed from C2	
BC, i.e., 23	2	1	Will be retained	BC
BE, i.e., 25	4	1	Will be retained	BE
CE, i.e., 35	0	1	Will be retained	CE

This pruned C2 is used to find L2 as shown in Table 9.93.

Table 9.93 Finding L2

Item pairs	*Count*
AC	2
BC	2
BE	3
CE	2

All these item pairs have support more than threshold limit. This L2 will be used for identifying three item pairs by selecting only those item pairs from Table 9.90 which are having support more than the threshold limit, as shown below in Table 9.94

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.94 Identifying three itemsets

TID	*Original Item Pairs*	*Selected Item Pairs according to L2*	*Three Itemsets*
100	AC, AD, CD	AC	NIL
200	BC, BE, CE	BC, BE, CE	BCE
300	AB, AC, AE, BC, BE, CE	AC, BC, BE, CE	BCE
400	BE	BE	NIL

Here, BCE has a support of 2; thus, the frequent item group is BCE.

In next phase, association rules for three item group BCE can be found by using the same approach as discussed in the Apriori algorithm.

Example: Let us consider, the database given in Table 9.95 for identification of frequent item pairs with the Apriori algorithm and the DHP algorithm for minimum support of 50%.

Table 9.95 Transaction database

TID	*Item Bought*
100	1, 2, 3, 4
200	5, 6, 7, 8, 9
300	6, 8, 9, 10
400	2, 3, 9

The process to identify frequent item pairs through Apriori algorithm has been illustrated in Figure 9.16.

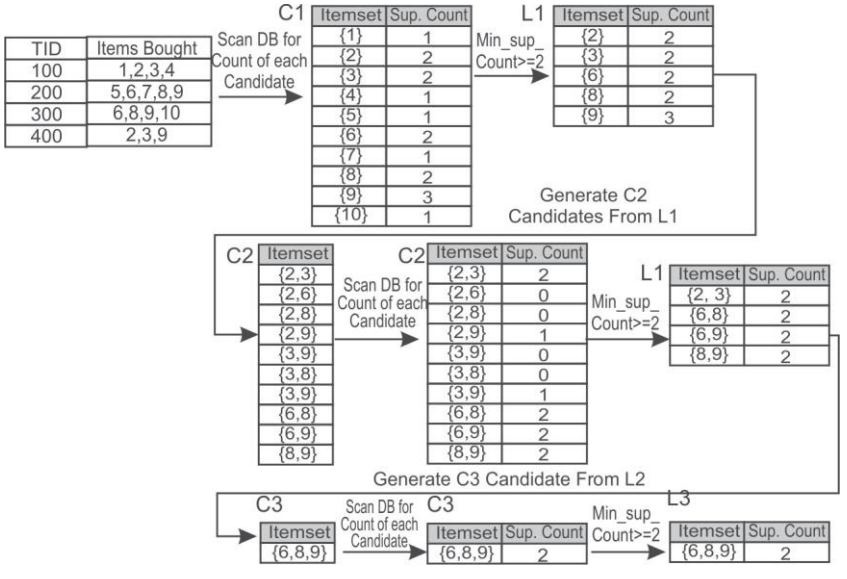


Figure 9.16 Identifying frequent item pairs and groups by Apriori algorithm

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Thus, the process of the Apriori algorithm has produced a frequent three item group {6, 8, 9}.

The process to identify frequent item pairs and groups through the DHP algorithm has been illustrated in Table 9.96.

Table 9.96 Coded item pairs for DHP

TID	*Items bought*	*Item pairs code*
100	1,2,3,4	12,13,14,23,24,34
200	5,6,7,8,9	56,57,58,59,67,68,69,78,79,89
300	6,8,9,10	68,69,610,89,810,910
400	2,3,9	23,29,39

Now, the hash function is applied on coded values so that these items can be assigned to different buckets. Here, we are going to use Modulo 11 number, i.e., dividing by 11 and using the remainder as hash function to assign buckets for the given dataset.

Let's apply hash function to each item pair and assign the item pairs to each bucket on the basis of modulo 11 as shown in Table 9.97.

Table 9.97 Assigning of item pairs to buckets based on hash function modulo 11

0	1	2	3	4	5	6	7	8	9	10	Bucket address
	12 23 34 56 67 89 23	13 24 57 68 68	14 58 69 69	59	610	39	29 810	910			Item pairs
0	7	5	4	1	1	1	2	1	0	0	Support of bucket
0	1	1	1	0	0	0	1	0	0	0	Bit Vector

The pruning of C2 has been illustrated in Table 9.98.

Table 9.98 Pruning of C2

Item Pairs in C2	*Bucket*	*Bit Vector*	*Remarks*	*Pruned C2*
23	1	1	Will be retained	23
26	-	-	Removed from C2	
28	-	-	Removed from C2	
29	7	1	Will be retained	29
36	-	-	Removed from C2	

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Item Pairs in C2	Bucket	Bit Vector	Remarks	Pruned C2
38	-	-	Removed from C2	
39	6	0	Removed from C2	
68	2	1	Will be retained	68
69	3	1	Will be retained	69
89	1	1	Will be retained	89

This pruned C2 is used to find L2 as shown in Table 9.99.

Table 9.99 Finding L2

Item pair	Count
23	2
29	1
68	2
69	2
89	2

Thus, L2 is 23, 68, 69 and 89, which is used to find three item groups as shown in Table 9.100.

Table 9.100 Finding three itemsets

TID	Original Item pairs	Selected Item pairs according to L2	Three Itemsets
100	12,13,14,23,24,34	23	NIL
200	56,57,58,59,67,68,69,78,79,89	68,69,89	689
300	68,69,610,89,810,910	68,69,89	689
400	23,39	23	NIL

Here, 689 has a support of 2 and so the frequent item grouping is 689.

We have found the identical three item grouping by using both the Apriori and the DHP algorithms. Now, the same procedure that was discussed in the section on the Apriori algorithm can be applied to determine rules for the three item group $\{6, 8, 9\}$.

9.11 Dynamic Itemset Counting (DIC)

The major issue with the Apriori algorithm is that it requires multiple scans of the transaction database which is equal to the number of items in the last candidate itemset that was checked for its support.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

For example, to identify candidate 3-itemset groupings, it will require a minimum of 3 scans of the database and to identify candidate 4-itemset groupings, it will require four scans of the database.

The major advantage of the Dynamic Itemset Counting (DIC) algorithm is that it requires only 2 scans of the database to identify frequent item pairs in the whole database. It reduces the number of scans required by not just doing one scan for the frequent 1-itemset and another for the frequent 2-itemset but combining the counting for a number of itemsets as soon as it appears that it might be necessary to count it.

The basic algorithm is as follows.

1. Divide the transaction database into *n* number of partitions.
2. In the first partition of the transaction database, count the 1-itemsets.
3. Count the 1-itemsets at the beginning of the second partition of the transaction database and also start counting the 2-itemsets using the frequent 1-itemsets from the first partition.
4. Count the 1-itemsets and the 2-itemsets at the beginning of the third partition and also start counting the 3-itemsets using results from the first two partitions.
5. Continue this process until the whole database is scanned once. There will be final set of frequent 1-itemsets once you scan the database fully.
6. Go back to the starting of the transaction database and continue counting the 2-itemsets and the 3-itemsets, etc.
7. After the end of the first partition in the second scan of the database, the whole database has been scanned for 2-itemsets. Therefore, we have the final set of frequent 2-itemsets.
8. After the end of the second partition in the second scan of the database, the whole database has been scanned for 3-itemsets. Therefore, we have the final set of frequent 3-itemsets.
9. This process is continued in a similar manner until no frequent k-itemsets are identified.

Illustration on working of the DIC algorithm

Let us consider, a dataset having 100 records which is divided into a number of, say 5 equal partitions. Now, the major assumption in DIC is that the data should be homogeneous throughout the file. This means that all the items should appear in partitions and the percentage of each item in the partition should be same as percentage of each item in whole dataset. In simple words, if there are 5 items A, B, C, D and E in the dataset then each of these items should appear in the each partition and if the percentage of item A in the whole transaction dataset is 60%, then it should be the same in each partition, i.e, if the count of data item A is 50 in whole dataset of 100 transactions, then the item should have the count of 10 in each partition of 20 records i.e. 50% both ways.

On the assumption that data is homogenous throughout in each partition, data is processed for the DIC algorithm.

Here, we have a database of 100 transactions which is divided into five equal partitions of 20 records each. During counting of the 1-itemsets in the first partition of the transaction database, we have identified data items A, B, C and D in the first partition. The count of these items is indicated as C_A, C_B and so on. Count the 1-itemsets at the beginning of the second partition, and also starts counting the 2-itemsets using the frequent 1-itemsets from the first partition, i.e. we also start counting two item pairs, i.e., AB, AC, AD, BC, BD and CD. The working of the DIC algorithm has been illustrated in Table 9.101.

It has been observed the counting of 1-itemsets, i.e., A, B, C and D has been completed after

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010

Table 9.101 Working of the DIC algorithm for the example database

Total dataset of 100 records	Partition-1 20 records	PHASE-1 1-iemsets $A\text{-}C_A$ $B\text{-}C_B$ $C\text{-}C_C$ $D\text{-}C_D$				PHASE-2 2-itemsets $AB\text{-}C_{AB}$ $AC\text{-}C_{AC}$ $AD\text{-}C_{AD}$ $BC\text{-}C_{BC}$ $BD\text{-}C_{BD}$ $CD\text{-}C_{CD}$	PHASE-2 3-itemsets $ABC\text{-}C_{ABC}$ $ABD\text{-}C_{ABD}$ $ACD\text{-}C_{ACD}$ $BCD\text{-}C_{BCD}$	PHASE-2 4-itemsets $ABCD\text{-}C_{ABCD}$	**Count of 2-itemset has been completed after first partition of second pass**
	Partition-2 20 records	1-iemsets $A\text{-}C_A$ $B\text{-}C_B$ $C\text{-}C_C$ $D\text{-}C_D$	2-itemsets $AB\text{-}C_{AB}$ $AC\text{-}C_{AC}$ $AD\text{-}C_{AD}$ $BC\text{-}C_{BC}$ $BD\text{-}C_{BD}$ $CD\text{-}C_{CD}$			PHASE-2 3-itemsets $ABC\text{-}C_{ABC}$ $ABD\text{-}C_{ABD}$ $ACD\text{-}C_{ACD}$ $BCD\text{-}C_{BCD}$	PHASE-2 4-itemsets $ABCD\text{-}C_{ABCD}$	**Count of 3-itemset has been completed after second partition of second pass**	
	Partition-3 20 records	1-iemsets $A\text{-}C_A$ $B\text{-}C_B$ $C\text{-}C_C$ $D\text{-}C_D$	2-itemsets $AB\text{-}C_{AB}$ $AC\text{-}C_{AC}$ $AD\text{-}C_{AD}$ $BC\text{-}C_{BC}$ $BD\text{-}C_{BD}$ $CD\text{-}C_{CD}$	3-itemsets $ABC\text{-}C_{ABC}$ $ABD\text{-}C_{ABD}$ $ACD\text{-}C_{ACD}$ $BCD\text{-}C_{BCD}$		PHASE-2 4-itemsets $ABCD\text{-}C_{ABCD}$	Count of 4-itemsets has been completed after third partition of second pass		

Contd.

www.EBooksWorld.ir

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010

	Partition -4 20 records	1-iemsets A-C_A B-C_B C-C_C D-C_D	2-itemsets AB-C_{AB} AC-C_{AC} AD-C_{AD} BC-C_{BC} BD-C_{BD} CD-C_{CD}	3-itemsets ABC-C_{ABC} ABD-C_{ABD} ACD-C_{ACD} BCD-C_{BCD}	4-itemsets ABCD-C_{ABCD}		
	Partition -5 20 records	1-iemsets A-C_A B-C_B C-C_C D-C_D	2-itemsets AB-C_{AB} AC-C_{AC} AD-C_{AD} BC-C_{BC} BD-C_{BD} CD-C_{CD}	3-itemsets ABC-C_{ABC} ABD-C_{ABD} ACD-C_{ACD} BCD-C_{BCD}	4-itemsets ABCD-C_{ABCD}	Count of 1-itemset has been completed after one pass	

www.EBooksWorld.ir

one pass, the counting of 2-itemset has been completed after the first partition of the second pass, counting of 3-itemsets has been completed after the second partition of the second pass and counting of 4-itemsets has been completed after the third partition of the second pass. It is clear from above illustration that DIC will require only two scans of database while Apriori will require 4 scans for the identification of frequent 4-itemsets.

Major condition to make DIC effective

The DIC algorithm works well when the data throughout the file is relatively homogeneous as it starts counting of the 2-itemset before having a count of final 1-itemset. The algorithm is not able to identify the large itemset till the whole database is scanned if the data distribution is not homogeneous. The major benefit of DIC is that it finishes the counting of the itemset in two scans of the database while Apriori often takes three or more scans.

9.12 Mining Frequent Patterns without Candidate Generation (FP Growth)

As only frequent items are required for identifying the association rules, so it is best to identify the frequent items in the dataset and ignore all others. The FP growth works on the same principle and instead of generating candidate itemsets as in the case of the Aprioiri algorithm, frequent pattern-growth (FP growth) only tests and generates frequent itemsets. Thus, the major difference between the Apriori algorithm and FP growth is that FP growth does not generate candidate itemsets, it only tests, whereas the Apriori algorithm generates candidate itemsets and then tests them.

To improve the performance of the algorithm, it uses the principle to store frequent items in a compact structure which does not require the need to use the original transaction database repeatedly.

The working of the FP growth algorithm is as follows:

1. Like the Apriori algorithm, the transaction database is scanned once to identify all the 1-itemset frequent items and their supports are noted down.
2. Then 1-itemset frequent items are sorted in descending order of their support.
3. After that the FP-tree is created with a 'NULL' root.
4. Then the first transaction is obtained from the transaction database and all the non-frequent items are removed. And the remaining items are listed according to the order in the sorted frequent items.
5. After this, the first branch of the tree is constructed using the transaction with each node corresponding to a frequent item and by setting the frequency of each item as 1 for the first transaction.
6. Then the next transaction is retrieved from the transaction database and all the non-frequent items are removed. And the remaining items are listed according to the order in the sorted frequent items as done in step 4.
7. Now, the transaction is inserted in the tree using any common prefix that may appear and

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

count of items is increased by 1. If no common prefix is found then the branch of the tree is constructed using the transaction; with each node corresponding to a frequent item and by setting the frequency of each item as 1.

8. Step 6 is repeated until all transactions in the database have been processed.

To understand this algorithm let us find frequent item pairs by building FP-trees for the following database.

Example: Let us consider a transaction database shown in Table 9.102. The minimum support required is 50% and confidence is 75%.

Table 9.102 Transaction database

100	Butter, Curd, Eggs, Jam
200	Butter, Curd, Jam
300	Butter, Eggs, Muffin, Nuts
400	Butter, Eggs, Jam, Muffin
500	Curd, Jam, Muffin

According to the algorithm, the first step will be to scan the transaction database to identify all the frequent items in the 1-itemsets and sort them in descending order of their support.

The frequent 1-itemset database sorted on the basis of frequency is shown in Table 9.103.

Table 9.103 Frequency of each item in sorted order

Item	*Frequency*
Butter	4
Jam	4
Curd	3
Eggs	3
Muffin	3

Here, Nuts has been removed as it has a support of 1 only, which is less than the threshold limit of support (which is 3 in our case).

To create the FP-tree, the next step will be to exclude all the items that are not frequent from the transactions and sort the remaining frequent items in descending order of their frequency count. The example transaction database will be processed accordingly. The updated database after eliminating the non-frequent items and reorganising the frequent items on the basis of frequency, is shown in Table 9.104.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.104 Updated database after eliminating the non-frequent items and reorganising it according to support

Transaction ID	*Items*
100	Butter, Jam, Curd, Eggs
200	Butter, Jam, Curd
300	Butter, Eggs, Muffin
400	Butter, Jam, Eggs, Muffin
500	Jam, Curd, Muffin

It is important to note that in case of identical support, items are arranged according to lexicographic or sorted order. Now, we create the FP-tree by using the algorithm discussed above. For simplicity of discussion, we are referencing items with their first letter, i.e, Butter by B, Jam by J and so on.

An FP-tree consists of nodes. Here, each node contains three fields, i.e, an item name, a count, and a node link as given in Figure 9.17. To built the FP-tree let us consider first transaction, i.e., 100 having Butter, Jam, Curd, Eggs as items. Each item node will be created from the root node, i.e., NULL as shown in Figure 9.17. The tree is built by making a root node labeled NULL. A node is made for each frequent item in the first transaction and the count is set to 1. For example, to build the FP-tree of Figure 9.17, the first transaction {B, J, C, E} is inserted in the empty tree with the root node labeled NULL. Each of these items is given a frequency count of 1.

Next, the tree is traversed for the next transaction, i.e., 200. If a path already exists then it will follow the same path and corresponding count of item is increased by 1. If a path does not exist then a new path will be created. Next the second transaction, which is {B, J, C} having all the items common with first, is inserted. It changes the frequency of each item, i.e., {B, J, C} to 2. The FP for first two transactions is shown in Figure 9.18.

Figure 9.17 FP-tree for first transaction, i.e., 100 only

Figure 9.18 FP-tree for the first two transactions

Similarly, third transaction is considered and {B, E, M} is inserted. This requires that nodes for E and M be created. The counter for B goes to 3 and the counter for E and M is set to 1.The FP-tree for first three transactions is shown in Figure 9.19.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

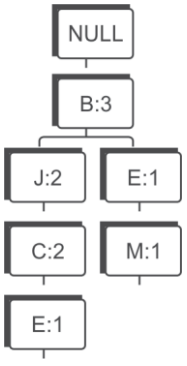


Figure 9.19 FP-tree for first three transactions

The next transaction {B, J, E, M} results in counters for B and J going up to 4 and 3 respectively and a new nodes for E and M have been inserted with count of 1 under node J as shown in Figure 9.20.

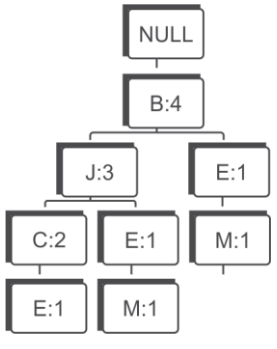


Figure 9.20 FP-tree for first four transactions

The last transaction {J, C, M} results in a brand new branch for the tree which is shown on the right-hand side in Figure 9.21 with a counter of 1 for each item.

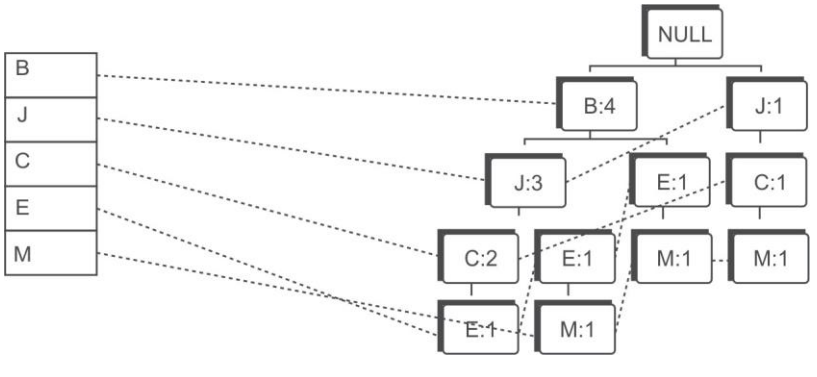


Figure 9.21 The final FP-tree for the example database after five transactions

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

The count tells the number of occurrences the path (constructed from the root node to this node) has in the transaction database. The node link is a link to the next node in the FP-tree containing the same item name or a null pointer if this node is the last one with this name. The FP-tree also consists of a header table with an entry for each itemset and a link to the first item in the tree with the same name. This linking is done to make traversal of the tree more efficient. Nodes with the same item name in the tree are linked via the dotted node-links.

The nodes near the root in the tree are more frequent than those further down the tree. Each identical path is only stored in the FP-tree once, which often makes the size of an FP-tree smaller than the corresponding transactional database. The height of an FP-tree is always equal to the maximum number of itemsets in a transaction excluding the root node. The FP-tree is compact and often orders of magnitude smaller than the transaction database. Once the FP-tree is constructed, the transaction database is not required.

Identification of frequent itemsets from FP-tree

The mining on the FP-tree structure is performed using the Frequent Pattern growth (FP growth) algorithm. This algorithm starts with the least frequent item, *i.e.*, the last item in the header table. Then the algorithm identifies all the paths from the root to this least frequent item and adjusts the count on the basis of the item's support count.

First, look at Figure 9.21 built using the FP-tree to identify the frequent itemsets in the example.

Start with the item M and identify the patterns given as follows:

Identification of patterns for item M

BEM(1)
BJEM(1)
JCM(1)

From this we can identify that, the support for BM is 2 (1 from BEM and 1 from BJEM), but its support is less than the threshold limit of 3, so this item pair will be discarded and we have identified no frequent item pairs by using item M.

Next consider the item E and identify the patterns given as follows:

Identification of patterns for item E

BJCE(1)
BJE(1)
BE(1)

From this we can identify that the support for BE is 3 (1 from BJCE, 1 from BJE and 1 from BE). However there is also a three item pair BJE having a support of 2 (1 from BJCE and 1 BJE), but its support is less than threshold value so will be discarded.

So, we have identified BE has two item pairs having support more than the threshold limit.

Next consider the item C and identify the following patterns:

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Identification of patterns for item C

BJC(2)
JC(1)

From this we can identify that, the support for JC is 3 (2 from BJC and 1 from JC). There is no other item pair having support equal to or more than 3.

So, we have identified JC has two item pairs having supports more than the threshold limit.

Next consider the item J to identify its patterns.

Identification of patterns for item J

BJ(3)
J(1)

From this we can identify that, the support for BJ is 3. There is no other item pair having support equal to or more than 3.

So, we have identified that BJ has two item pairs having support more than the threshold limit.

Since, B is at top of the tree and cannot have any item prior to it to make a pair, so there is no need to follow links for item B. Thus, when the tree only contains one path, the algorithm will find all the possible combinations of the items in the itemset that support the minimum support count. It is therefore important to start with the least frequent item the first time, otherwise the items with the higher frequency will not be considered.

Thus, the final list of frequent item pairs for the given database of transactions is shown in Table 9.105.

Table 9.105 Frequent item pairs for database example given in table

Frequent Item pairs	*Count*
BE	3
JC	3
BJ	3

Finding association rules

To find the association mining rules, the same approach is followed as in Apriori algorithm.

For item pair BE, the possible rules are B→E and E→B. The confidence for each rule will be calculated and if its value is more than given threshold value of confidence then corresponding rule will be selected otherwise it will be discarded. The same approach will be followed for other frequent item pairs JC and BJ to find association rules for the given dataset.

The readers are advised to identify association mining rules that have confidence of more than 80%.

I hope that, you have identified it correctly and answer is E→B and C→J as explained below.

Confidence of (B→E) = S(B∩E) / S(B) = 3/4 =0.75 [support of individual items is given in Table 9.101]

Confidence of (E→B) = S(E∩B) / S(E) = 3/3 =1.0

Confidence of (J→C) = S(J∩C) / S(J) = 3/4 = 0.75

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Confidence of $(C \rightarrow J) = S(C \cap J) / S(C) = 3/3 = 1$
Confidence of $(B \rightarrow J) = S(B \cap J) / S(B) = 3/4 = 0.75$
Confidence of $(J \rightarrow B) = S(J \cap B) / S(J) = 3/4 = 0.75$

Example: Let us consider another database given in Table 9.106, to identify frequent item pairs having support more than 50%.

Table 9.106 Transaction database

TID	Items
100	A C D
200	B CE
300	A B CE
400	B E

The count for each item is given in Table 9.107.

Table 9.107 Count for each data item

Item	Count
A	2
B	3
C	3
D	1
E	3

On the basis of the frequent items given in Table 9.108, those items are selected that have support equal to or more than 2 and these frequent items are sorted on the basis of their frequency as shown in Table 9.108.

Table 9.108 Frequency of each item in sorted order

Item	Count
B	3
C	3
E	3
A	2

Now, the non-frequent items are removed from the transaction database and items are ordered on the basis of their frequency as given in the above table. The database after eliminating the items that are not frequent and reorganising them on the basis of their frequency is shown in Table 9.109.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.109 Modified database after eliminating the non-frequent items and reorganising

TID	Items
100	C A
200	B C E
300	B C E A
400	B E

The process of creating the FP-tree for the database given above has been shown in Figure 9.22 on a step-by-step basis.

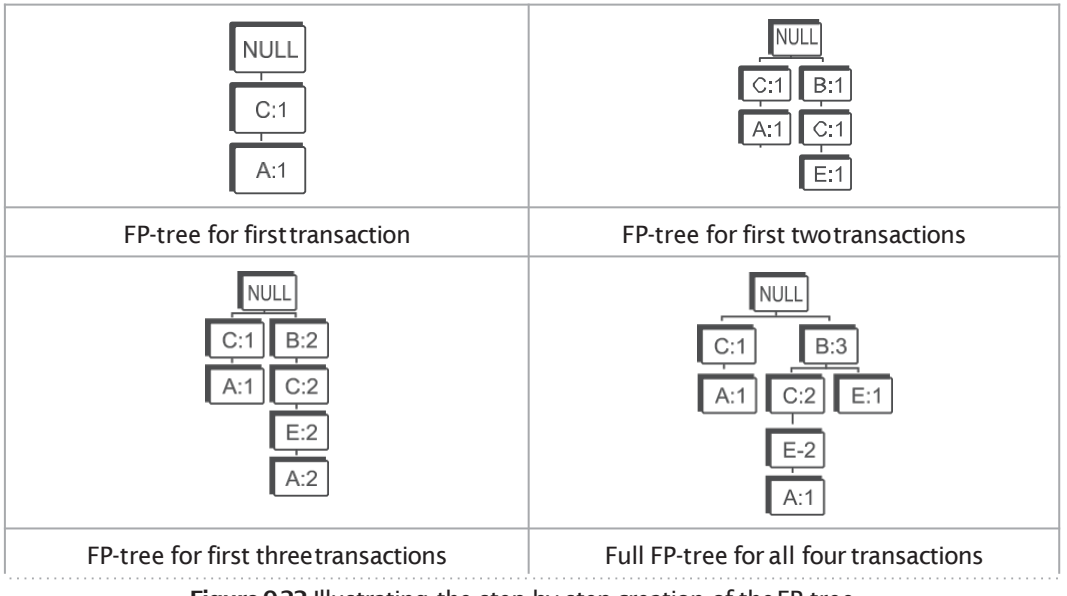


Figure 9.22 Illustrating the step-by-step creation of the FP-tree

Thus, the final FP-tree with nodes will be as shown in Figure 9.23.

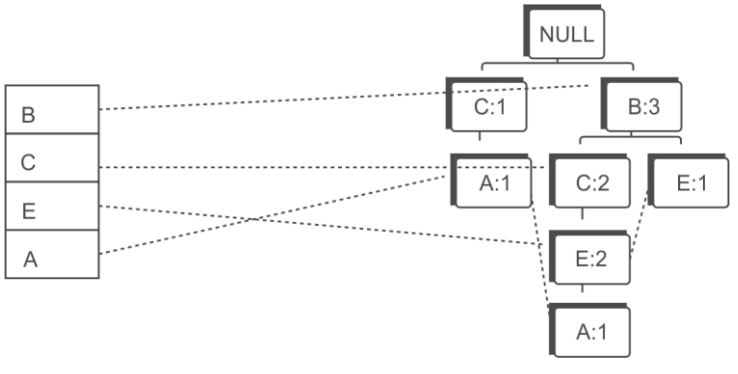


Figure 9.23 Final FP-tree for database given in Table 9.105

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Identification of Frequent Itemsets from the FP-tree

Start with the item A and identify the patterns given as follows:

Identification of patterns for item A

CA(1),
BCEA(1),

The support for CA is 2 (1 from CA and 1 from BCEA) and it is equal to the threshold limit thus CA is identified as a frequent item pair.

Next consider the item E and identify the following patterns:

Identification of patterns for item E

BCE(2)
BE(1)

The support for BCE is 2 and it is equal to the threshold limit thus BCE is identified as a frequent 3-itemset.

Also from E, BE is identified as a frequent 2-item pair, since it has a support of 3 (2 from BCE and 1 from BE).

Thus, we have selected two frequent item pairs from E, i.e., BCE and BE.

Next consider the item C and examine the following patterns:

Identification of patterns for item C

BC(2)

The support for BC is 2 and it is equal to the threshold limit thus BC is identified as a frequent item pair.

Since, B is at top of the tree and cannot have any item prior to it to make a pair, so there is no need to follow links for item B.

The final list of frequent item pairs for the given database of transactions is as shown in Table 9.110.

Table 9.110 Frequent item pairs for the example database

Frequent Item pairs	*Count*
CA	2
BCE	2
BE	2
BC	2

Finding association rules

The identification process of association rules having confidence more than 70% has been shown in Table 9.111.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.111 Calculation of confidence for identification of association rules

CA	C→A = S(C∩A) /S(C) = 2/3 = 0.67 A→C = S(A∩C) /S(A) = 2/2 = 1.0	Selected rule is A→C
BCE	B→CE = S(B∩C∩E) /S(B) = 2/3 = 0.67	Confidence is less then threshold value, so it will be discarded.
	CE→B = S(B∩C∩E) /S(CE) = 2/2 = 1.0 Possible other non-implied rules are: C→E, E→B C→E = S(C∩E) /S(C) = 2/3 = 0.67 E→B = S(E∩B) /S(E) = 2/3 = 0.67	Selected rule is CE→B. Others are discarded
BE	B→E = S(B∩E) /S(B) = 2/3 = 0.67 E→B = S(E∩B) /S(E) = 2/3 = 0.67	Both the rules are discarded.
BC	B→C = S(B∩C) /S(B) = 2/3 = 0.67 C→B = S(C∩B) /S(C) = 2/3 = 0.67	Both the rules are discarded.

Thus, final association rules having confidence more than 70% are CE→B and A→C.

Example: Let us consider another database given in Table 9.112, to identify frequent item pairs having support more than 50%.

Table 9.112 Transaction database

Tr.	Itemsets
T1	{1,2,3,4}
T2	{1,2,4}
T3	{1,2}
T4	{2,3,4}
T5	{2,3}
T6	{3,4}
T7	{2,4}

The count for each item is given in Table 9.113.

Table 9.113 Frequency of each item

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

On the basis of the frequent items given in Table 9.114, those items are selected that have support equal to or more than 2 and these frequent items are sorted on the basis of their frequency as shown in Table 9.114.

Table 9.114 Frequency of each item in sorted order

Item	Support
{2}	6
{4}	5
{3}	4
{1}	3

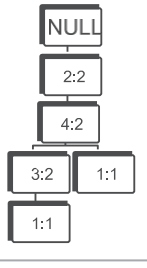
Now, the non-frequent items are removed from the transactions and items are organized according to their frequency as given in the above Table 9.114. Since, all items are frequent, transaction items are reorganized according to their support as shown in Table 9.115.

Table 9.115 Modified database after eliminating the non-frequent items and reorganizing

T1	2 4 3 1
T2	2 4 1
T3	2 1
T4	2 4 3
T5	2 3
T6	4 3
T7	2 4

The process of creating the FP-tree for this database has been shown in Figure 9.24 on a step-by-step basis.

FP-tree for first transaction: {2 4 3 1}



FP for first and second transaction {2 4 1}

Contd.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

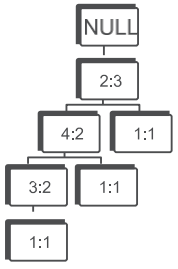
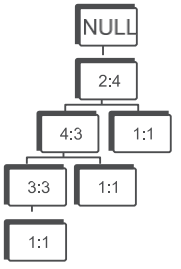
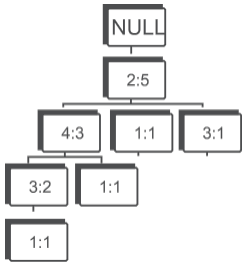
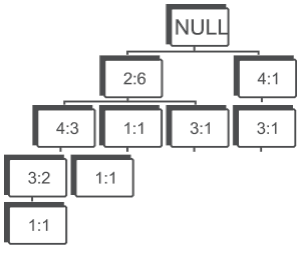
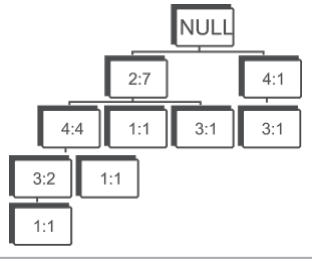
	
FP for first, second and third transaction {2 1}	FP for first, second, third and fourth transaction {2 4 3}
	
FP for first, second, third, fourth and fifth transaction {2 3}	FP for first, second, third, fourth, fifth and sixth transaction {4 3}
	
FPfor first, second, third, fourth, fifth, sixth and seventh transaction {2 4}	

Figure 9.24 Illustrating the step-by-step creation of the FP-tree

Thus, the final FP-tree with nodes will be as shown in Figure 9.25.

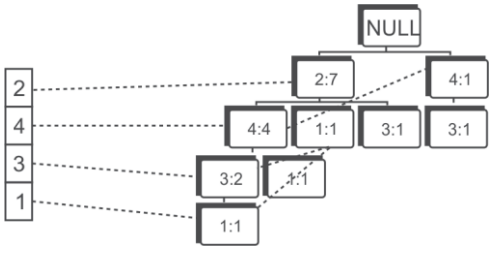


Figure 9.25 FP-tree for the example database

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Identification of frequent itemsets from the FP-tree

Start with the item 1 and identify the patterns given as follows.

Identification of patterns for item 1

2431(1)
241(1)
21 (1)

The support for 21 is 3 (1 from 2431, 1 from 241 and 1 from 21) and it is equal to the threshold limit thus 21 is a identified as frequent item pair.

Next consider the item 3 and identify its patterns.

Identification of patterns for item 3

243(2)
23(1)
43(1)

The support for 43 is 3 (2 from 243 and 1 from 43) and it is equal to the threshold limit thus 43 is identified as a frequent item pair.

The support for 23 is also 3 (2 from 243 and 1 from 23) and it is equal to the threshold limit thus 23 is also identified as a frequent item pair.

Next consider the item 4 and identify its patterns.

Identifications of patterns for item 4

24(4)

The support for 24 is 4 and it is more than threshold limit, thus 24 is identified as a frequent item pair.

The final list of frequent item pairs for given database of transactions is shown in Table 9.116.

Table 9.116 Frequent item pairs for example database

Item	Support
{1,2}	3
{3,4}	3
{2,3}	3
{2,4}	4

It is important to note that the order of data items does not matter in listing frequent item pairs.

Finding association rules

The identification process of association rules having confidence more than 70% has been shown in Table 9.117.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Table 9.117 Association rules for database given in Table 9.76

{1, 2}	1→2 = S(1∩2) / S(1) = 3/3 =1.0 2→1 = S(2∩1) / S(1) = 3/3 = 1.0	Both the rules, i.e., 1→2 and 2→1 are selected
{3, 4}	3→4 = S(3∩4) / S(3) = 3/4 = 0.75 4→3 = S(4∩3) / S(3) = 3/4 = 0.75	Both the rules, i.e., 3→4 and 4→3 are selected
{2, 3}	2→3 = S(2∩3) / S(2) = 3/6 = 0.50 3→2 = S(3∩2) / S(3) = 3/4 = 0.75	3→2 rule has been selected
{2, 4}	2→4 = S(2∩4) / S(2) = 4/6 = 0.67 4→2 = S(4∩2) / S(4) = 4/5 = 0.80	4→2 rule has been selected

Thus, final association rules having confidence more than 70% are 1→2, 2→1, 3→4, 4→3, 3→2 and 4→2.

9.12.1 Advantages of the FP-tree approach

The advantages of the FP-tree algorithm over the Apriori algorithm are as follows:

- The FP-tree algorithm avoids the need to scan the database more than twice to identify the counts of support.
- The FP-tree completely eliminates the requirement of costly candidate generation, which can be expensive in case of the Apriori algorithm for the candidate set C2.
- The FP growth algorithm is better than the Apriori algorithm when the transaction database is huge and the minimum support count is low. Because in case of low minimum support count, there will be more items that will satisfy the support count and hence the size of the candidate sets for Apriori will be large. As the database grows, FP growth uses an efficient structure to mine patterns.

9.12.2 Further improvements of FP growth

In spite of the advantages there are possibilities to make the algorithm even more efficient and scalable. Important observations to further improve the performance of FP growth are as follows:

- It works best if the FP-tree fits in the main memory. However, in large databases this will not always be possible, and then it is better to first divide the database and then constructs FP-trees for each of the databases.
- Use ***B+-tree*** to index the FP-tree for faster access to items in the FP-tree.
- If the same data is used multiple times with different minimum confidence thresholds in order to yield different association rule mining then, it could be useful to use the same FP-tree results. This will save the initial cost of generating the FP-tree and make the algorithm overall more efficient.
- Sometimes the minimum support count will be different in different tasks, so to overcome this problem, the lowest minimum support count is used in the materialization of the FP-tree. When a higher support count is wanted, only the top of the FP-tree is used. Because of the structure of the FP-tree, it is easy to have different support counts.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Remind Me

- Association rule mining can be defined as identification of frequent patterns, correlations, associations, or causal structures among sets of objects or items in transactional databases, relational databases, and other information repositories.
- Association rule mining often known as 'market basket' analysis is a very effective technique to find the association of sale of item X with item Y.
- Association rules are often written as X→Y meaning that whenever X appears Y also tends to appear.
- The metrics to evaluate the strength of association rules are support, confidence and lift.
- Let N is the total number of transactions. Support of X is represented as the number of times X appears in the database divided byN.
- Confidence for X→Y is defined as the ratio of the support for X and Y together to the support for X (which is same as the conditional probability of Y when X has already occurred).
- Lift is the ratio of conditional probability of Y when X is given to the unconditional probability of Y in the dataset.
- Rather than counting all the possible item combinations in the stock, the improved Naïve algorithm focuses only on the items that are sold in transactions.
- There are three ways to store datasets of transactions, i.e., simple storage, horizontal storage and vertical storage.
- The name of the Apriori algorithm is assigned on the basis of the fact that it uses prior knowledge of frequent itemset properties.
- The Apriori algorithm consists of two phases. In the first phase, the frequent itemsets, i.e., the itemsets that exceed the minimum required support are identified. In the second phase, the association rules meeting the minimum required confidence are identified from the frequent itemsets.
- Apriori property states that all nonempty subsets of a frequent itemset must also be frequent.
- Association rules for frequent itemset l generate all non-empty subsets of l and for every non-empty subset s of l, the output rule is s→l-s and if the confidence of the rule is more than the threshold value of the confidence then, this rule will be selected in the final association rules.
- Apriori–TID uses the Apriori algorithm to generate the candidate and frequent itemsets. The only difference is that it uses TID or vertical storage approach for its implementation.
- The DHP algorithm is useful to improve the pruning of C2.This algorithm employs a hash-based technique for reducing the count of candidate itemsets generated in the first pass.

Point Me (Books)

- Han, Jiawei, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam: Elsevier.
- Gupta, G. K. 2014. *Introduction to Data Mining with Case Studies*. Delhi: PHI Learning Pvt.Ltd.
- Mitchell, Tom M. 2017. *Machine Learning*. Chennai: McGraw Hill Education.
- Witten, Ian H., Eibe Frank, Mark A. Hall, and Christopher Pal. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington: Morgan Kaufmann.

Point Me (Video)

- https://www.coursera.org/lecture/text-mining/1-7-word-association-mining-and-analysis-Uufkz

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Connect Me (Internet Resources)

- https://en.wikipedia.org/wiki/Association_rule_learning
- https://www.hackerearth.com/blog/machine-learning/beginners-tutorial-apriori-algorithm-data-mining-r-implementation/

Test Me

Answer these questions based on your learning in this chapter.

1. Apply the Apriori and DHP algorithms to find frequent itemsets for the following dataset, Minimum support is 50%.

TID	*List of Items*
100	I1, I2, I5
200	I2, I4
300	I2, I3
400	I1, I2, I4
500	I1, I3
600	I2, I3
700	I1, I3
800	I1, I2, I3, I5
900	I1, I2, I3

2. Identify C4 after pruning the following frequent three itemsets as L3. Justify your answer.
   {3,4,5}, {3,4,7}, {3,5,6}, {3,5,7}, {3,5,8}, {4,5,6}, {4,5,7}
3. Show that if the rule ABC→ D does not have the minimum confidence required, then the rules like AB→CD, A→BCD and C→ABD will also not have the minimum confidence.
4. Compute the support for itemsets *{e}*, *{b, d}*, and *{b, d, e}* by treating each transaction ID as a market basket.

Customer ID	*Transaction ID*	*Items Bought*
1	0001	{a, d, e}
1	0024	{a, b, c, e}
2	0012	{a, b, d, e}
2	0031	{a, c, d, e}
3	0015	{b, c, e}
3	0022	{b, d, e}
4	0029	{c, d}
4	0040	{a, b, c}
5	0033	{a, d, e}
'5	0038	{a, b, e}

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Also compute the confidence and lift for the association rules
$\{b, d\} \rightarrow \{e\}$ and $\{e\} \rightarrow \{b,d\}$.

5. Consider the following group of frequent 3-itemsets and hence, identify C4 after pruning the following frequent three itemsets as L3. Justify your answer.
   {1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}, {2, 3, 5}, {3, 4, 5}.
6. Apply the Apriori algorithm and the FP-tree growth approach for the following database to identify frequent itemsets with minimum support of 50%. Also find association rules having confidence of more than 75%.

Transaction ID	*Items Bought*
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

7. For an Apriori algorithm the L2 is given below:

L_2	
Itemset	*Sup. count*
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

What will be C3 after pruning? Justify your answer.

8. Apply the DHP algorithm to find association mining rules for the following dataset with minimum support 50% and confidence 75%.

TID	*Items Bought*
100	1, 2, 3, 4
200	5, 6, 7, 8, 9
300	6, 8, 9, 10
400	2, 3, 9

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir

Also apply the FP growth algorithm to find the most frequent items pair(s).

9. What is wrong in the following L3:

   L2
   XY
   XZ

   L3
   XYZ

10. What are rules that can be generated from the three itemset {1,2,3}?
11. How is lift calculated in case of A->B
12. Convert the given transaction database to TID-list representation.

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

13. Frequent 2-itemsets are: {1,2}, {1,3}, {1,5}, {2,3}, {2,4}, {2,5}. What will be a pruned candidate 3-itemset?
14. Given a set of transactions, find rules to predict the occurrence of an item on the basis of the occurrences of other items in the transaction. The value of support is 40% and confidence is 70%.

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

Also apply the FP growth algorithm to find frequent item pairs.

Downloaded from https://www.cambridge.org/core. University of Sussex Library, on 01 May 2019 at 23:42:55, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781108635592.010
www.EBooksWorld.ir