| UNITII | MiningFrequent,AssociationsandCorrelations | 10 |
|---|---|---|
| **Mining Frequent, Associations and Correlations:** Basic Concepts, Frequent Itemset Mining Methods: Apriori Algorithm, Finding Frequent Itemsets by Confined Candidate Generation, FP-Growth, Generating Association Rules from Frequent Itemsets, Improving the Efficiency of Apriori, From Association Analysis to Correlation Analysis. | | |

# MiningFrequentPatternsMining

## Frequent Patterns in Data Mining

**ItemSet:**

AnItemsetiscollectionorsetofitems

**Examples:**

{*Computer*, *Printer*, *MSOffice*}is3itemset

{*Milk*,*Bread*}is2itemset Similarly, Set of K items is called k item set

## Frequent patterns

Thesearepatternsthatappearfrequentlyinadataset.Patternsmaybeitemsets,orsub sequences.

**Example:TransactionDatabase(Dataset)**

| TID | Items |
|---|---|
| **T1** | Bread,Coke,Milk |
| **T2** | Popcorn,Bread |
| **T3** | Bread,Egg,Milk. |
| **T4** | Egg,Bread,Coke, Milk |

A set of items, such as **Milk & Bread** that appear together in a transaction data set (Also called as **Frequent Item set**).

**Frequent itemset mining** leadstothediscoveryofassociationsandcorrelationsamong itemsin large transactional (or) relational data sets.

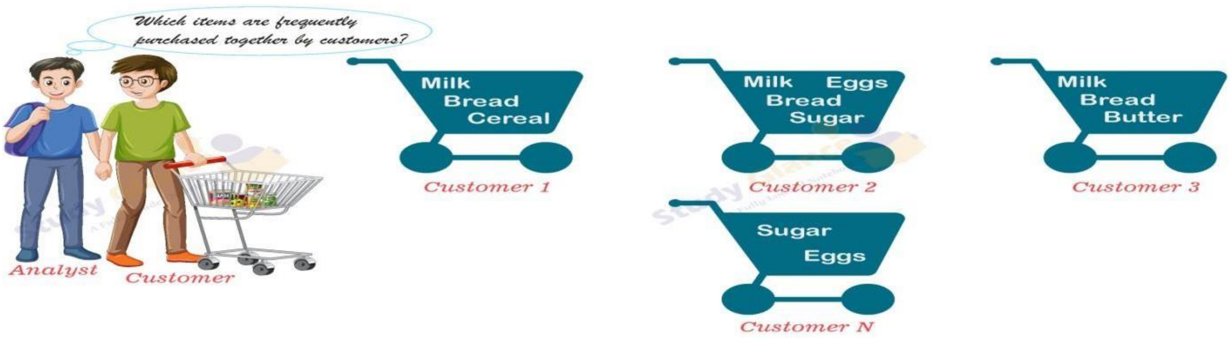
Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data. Moreover, it helps in data **classification, clustering**, and other data mining tasks.

## Associationsandcorrelations

Association rule mining (or) Frequent item set mining finds interesting **associations**and relationships (correlations) in large transactional or relational data sets. This rule shows how frequently an item set occurs in a transaction. A typical example is **Market Based Analysis**.

**MarketBasedAnalysis** is one of the key techniques used by large relations to show associations between items. It allows retailers to identify relationships between the items that people buy together frequently.

Thisprocessanalyzescustomerbuying habitsbyfindingassociationsbetweenthedifferent items that customers place in their **“shopping baskets”**.




The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likelyare theytoalso buybread (and what kind ofbread)onthe same trip to the supermarket? Thisinformation can lead toincreased salesby helping retailersdo **selective marketing** and **plan their shelf space**.
Understanding these buying patterns can help to increase sales in several ways. If there is a pair of items, X and Y, which are frequently bought together:

- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotionaldiscounts couldbeappliedtojust oneout ofthetwoitems.
- AdvertisementsonXcouldbetargetedatbuyerswhopurchaseY.
- XandYcouldbecombinedintoanewproduct,suchas havingYinflavours ofX.

**Association rule:** Ifthere is a pair of items, X and Y, which are frequently bought together then association rule is represented as **X ⇒Y**.

**Forexample**,the informationthatcustomerswho purchase**computers**alsotendto buy **antivirussoftware**at the same time is represented as

**Computer⇒Antivirus_Software**

## Measurestodiscoverinterestingnessofassociationrules

Associationrulesanalysisisatechniquetodiscoverhowitemsareassociatedtoeachother. There are three measure to discover interestingness of association rules. Those are:
**Support:** Thesupportof anitem /item setisthe **numberoftransactions** inwhichtheitem / item set appears, divided by the **total number of transactions**.

**Formula:**

$$\text{Support}\{A\} = \frac{\text{Frequency}\{A\}}{N}$$

or

$$\text{Support}\{A,B\} = \frac{\text{Frequency}\{A,B\}}{N}$$

Where,A,BareitemsandNisthetotalnumber oftransactions.

**Example:**Table-1ExampleTransactions

| TID | Items |
|---|---|
| **T1** | Bread,Coke,Milk |
| **T2** | Popcorn,Bread |
| **T3** | Bread,Egg,Milk. |
| **T4** | Egg,Bread,Coke,Milk |
| **T5** | Egg,Apple |

**Example:** Support of item Coke:

$$\text{Support}\{\text{Coke}\} = \frac{\text{Frequency}\{\text{Coke}\}}{N}$$

$$= 2 / 5 = 0.4 \text{ i.e., } 40\%$$

**Example:** Support of item set Bread, Milk:

$$\text{Support}\{\text{Bread, Milk}\} = \frac{\text{Frequency}\{\text{Bread, Milk}\}}{N}$$

$$= 3 / 5 = 0.6 \text{ i.e., } 60\%$$

**Confidence:**ThissaysthathowlikelyitemBispurchasedwhenitemAispurchased, expressed as {A → B}. The Confidence of items (A and B) is the frequency or numberoftransactionsin whichtheitems(AandB)appear,dividedbythefrequency or number of transactions in which the item (A) appears.

**Formula:**

$$\text{Confidence}\{A \rightarrow B\} = \frac{\text{Frequency}\{A,B\}}{\text{Frequency}\{A\}}$$

or

$$\text{Confidence}\{A \rightarrow B\} = \frac{\text{Support}\{A,B\}}{\text{Support}\{A\}}$$

**Example:** From the Table-1, the confidence of *{Bread → Milk}* is

$$\text{Confidence}\{\text{Bread} \rightarrow \text{Milk}\} = \frac{\text{Frequency}\{\text{Bread, Milk}\}}{\text{Frequency}\{\text{Bread}\}}$$

$$= 3 / 4 = 0.75 \text{ i.e., } 75\%$$

(Or)

The confidence of *{Bread → Milk}* is

Support *{Bread, Milk}* = 3 / 5 = 0.6

Support *{Bread}* = 4 / 5 = 0.8

$$\text{Confidence}\{\text{Bread} \rightarrow \text{Milk}\} = \frac{\text{Support}\{\text{Bread, Milk}\}}{\text{Support}\{\text{Bread}\}}$$

$$= 0.6 / 0.8 = 0.75 \text{ i.e., } 75\%$$

**Lift:** This says that how likely item B is purchased when item A is purchased, expressed as an associationrule {A→ B}. The lift is a measure to predict the performance ofanassociationrule (targeting model).

Ifliftvalue is:

- greaterthan1meansthat itemBislikelytobeboughtifitemAisbought,
- lessthan1means that itemBisunlikelytobeboughtifitemAisbought,
- equalsto1meansthereisnoassociationbetweenitems(AandB).

**Formula:**

$$\text{Lift}\{A \rightarrow B\} = \frac{\text{Support}\{A, B\}}{\text{Support}\{A\} * \text{Support}\{B\}}$$

**Example:** From the Table-1, the lift of {*Bread* → *Milk*} is

Support {Bread, Milk} =3 / 5 = 0.6
Support {Bread} = 4 / 5 = 0.8
Support {Milk} = 3 / 5 = 0.6

$$\text{Lift}\{\text{Bread} \rightarrow \text{Milk}\} = \frac{\text{Support}\{\text{Bread, Milk}\}}{\text{Support}\{\text{Bread}\} * \text{Support}\{\text{Milk}\}}$$

= 0.6 / (0.8 * 0.6) = 0.6 / 0.42 = 1.25

TheLiftvalueisgreaterthan1meansthatitemMilkislikelytobeboughtifitemBreadis bought.

**Example:** TofindSupport,ConfidenceandLiftmeasureson thefollowingtransactionaldata set.

Table-2:ExampleTransactions

| **TID** | **Items** |
|---|---|
| **T1** | Bread,Milk |
| **T2** | Bread,Diaper,Burger,Eggs |
| **T3** | Milk,Diaper,Burger,Coke |
| **T4** | Bread,Milk,Diaper,Burger |
| **T5** | Bread,Milk,Diaper,Coke |

Number oftransactions=5.
**Support:**
**1 – ItemSet:**
Support{Bread}=4/5=0.8=80%
Support{Diaper} = 4/ 5=0.8= 80%
Support{Milk} =4/ 5 =0.8= 80%
Support{Burger}= 3/5= 0.6=60%
Support{Coke}= 2/5= 0.4= 40%
Support{Eggs}= 1/5=0.2= 20%
**2 – ItemSet:**
Support{Bread,Milk}=3 / 5=0.6 =60%
Support{Milk,Diaper} =3 / 5 =0.6=60%
Support{Milk,Burger} = 2/5= 0.4= 40%
Support{Burger,Coke}=1 /5=0.2= 20%
Support{Milk,Eggs} =0/5 =0.0= 0%
**3 – ItemSet:**
Support{Bread,Milk,Diaper} =2 /5 =0.4 =40%
Support{Milk,Diaper,Burger} =2/ 5=0.4=40%

**Confidence:**

$$\text{Confidence}\{\text{Bread} \rightarrow \text{Milk}\} = \frac{\text{Support}\{\text{Bread, Milk}\}}{\text{Support}\{\text{Bread}\}}$$

$= 0.6 / 0.8 = 0.75$ i.e., 75%.

$$\text{Confidence}\{\text{Milk} \rightarrow \text{Diaper}\} = \frac{\text{Support}\{\text{Milk, Diaper}\}}{\text{Support}\{\text{Milk}\}}$$

$= 0.6 / 0.8 = 0.75$ i.e., 75%.

$$\text{Confidence}\{\text{Milk} \rightarrow \text{Burger}\} = \frac{\text{Support}\{\text{Milk, Burger}\}}{\text{Support}\{\text{Milk}\}}$$

$= 0.4 / 0.8 = 0.5$ i.e., 50%.

$$\text{Confidence}\{\text{Burger} \rightarrow \text{Coke}\} = \frac{\text{Support}\{\text{Burger, Coke}\}}{\text{Support}\{\text{Burger}\}}$$

$= 0.2 / 0.6 = 0.33$ i.e., 33.3%.

$$\text{Confidence}\{\text{Bread, Milk} \rightarrow \text{Diaper}\} = \frac{\text{Support}\{\text{Bread, Milk, Diaper}\}}{\text{Support}\{\text{Bread, Milk}\}}$$

$= 0.4 / 0.6 = 0.66$ i.e., 66.6%.

$$\text{Confidence}\{\text{Milk, Diaper} \rightarrow \text{Burger}\} = \frac{\text{Support}\{\text{Milk, Diaper, Burger}\}}{\text{Support}\{\text{Milk, Diaper}\}}$$

$= 0.4 / 0.6 = 0.66$ i.e., 66.6%.

**Lift:**

$$\text{Lift}\{\text{Bread} \rightarrow \text{Milk}\} = \frac{\text{Support}\{\text{Bread, Milk}\}}{\text{Support}\{\text{Bread}\} * \text{Support}\{\text{Milk}\}}$$

$= 0.6 / (0.8 * 0.8) = 0.93$

While lift value less than 1 means that item 'Milk' is unlikely to be bought if item 'Bread' is bought.

$$\text{Lift}\{\text{Milk} \rightarrow \text{Burger}\} = \frac{\text{Support}\{\text{Milk, Burger}\}}{\text{Support}\{\text{Milk}\} * \text{Support}\{\text{Burger}\}}$$

$= 0.4 / (0.8 * 0.6) = 0.83$

While lift value less than 1 means that item 'Burger' is unlikely to be bought if item 'Milk' is bought.

$$\text{Lift}\{\text{Bread, Milk} \rightarrow \text{Diaper}\} = \frac{\text{Support}\{\text{Bread, Milk, Diaper}\}}{\text{Support}\{\text{Bread, Milk}\} * \text{Support}\{\text{Diaper}\}}$$

$= 0.4 / ( 0.6 * 0.8 ) = 0.4 / 0.48 = 0. 83$

While lift value less than 1 means that item 'Diaper' is unlikely to be bought if itemset 'Bread, Milk' is bought.

$$\text{Lift}\{\text{Milk, Diaper} \rightarrow \text{Burger}\} = \frac{\text{Support}\{\text{Milk, Diaper, Burger}\}}{\text{Support}\{\text{Milk, Diaper}\} * \text{Support}\{\text{Burger}\}}$$

$= 0.4 / ( 0.6 * 0.6 ) = 0.4 / 0.36 = 1.11$

While lift value greater than 1 means that item 'Burger' is likely to be bought if itemset 'Milk, Diaper' is bought.

## MiningMethods

The most famous story about association rule mining is the "beer and diaper." Researchers discovered that customers who buy diapers also tend to buy beer. This classic example showsthat there might be many interesting association rules hidden in our daily data.

Association rules help to predict the occurrence of one item based on the occurrences of other items in a set of transactions.

## AssociationrulesExamples

- Peoplewhobuy**bread**willalsobuy **milk;**representedas{**bread→milk**}
- Peoplewhobuy **milk**willalsobuy**eggs;**representedas{**milk→eggs**}
- Peoplewhobuy**bread**willalsobuy**jam;**representedas {**bread→ jam**}

Association Rules discover the relationship between two or more attributes. It is mainly in the form of- If antecedent than consequent. For example, a supermarket sees that there are 200 customersonFridayevening.Outofthe200customers,100boughtchicken,andoutofthe100

customers who bought chicken, 50 have bought Onions. Thus, the association rule would be- If customers buy chicken then buy onion too, with a support of 50/200 = 25% and a confidence of 50/100=50%.
Association rule mining is a technique to identify interesting relations between different items. Association rule mining has to:

- Findallthefrequentitems.
- Generateassociationrulesfromtheabovefrequent itemset.

There are many methods or algorithms to perform Association Rule Mining or Frequent Itemset Mining, those are:

- **Apriorialgorithm**
- **FP-Growthalgorithm**

## Apriorialgorithm

The Apriori algorithm is a classic and powerful tool in data mining used to discover frequent itemsets and generate association rules. Imagine a grocery store database with customer transactions. Apriori can help you find out which items frequently appear together, revealing valuable insights like:

- Customersbuyingbreadoftenbuybutterandmilktoo. (**Frequentitemset**)
- 70%ofpeoplewhopurchasediapersalsobuybabywipes.(**Associationrule**)

### HowApriorialgorithmworks:

- **Bottom-up Approach**: Starts with finding frequent single items, then combines them to find frequent pairs, triplets, and so on.
- **Apriori Property**: If a smaller itemset isn't frequent, none of its larger versions can be either. This "prunes" the search space for efficiency.
- **Support and Confidence**: Two key measures used to define how often an itemset appears and how strong the association between items is.

### LimitationsforApriorialgorithm

- Canbecomputationallyexpensiveforlargedatasets.
- Sensitivetominimumsupportandconfidencethresholds.

## FP-Growthalgorithm

FP-Growthstands forFrequent PatternGrowth, andit'sasmartersiblingoftheApriorialgorithm for mining frequent itemsets in data. But instead of brute force, it uses a clever strategyto avoid generating and testing tons ofcandidate sets, making it much faster and more memory-efficient.

### Here's its secret weapon:

- **Frequent Pattern Tree (FP-Tree)**: This special data structure efficiently stores the frequent itemsets andtheir relationships. Think of it as a compressedand organizedrepresentationof your grocery store database.
- **Pattern Fragment Growth**:Insteadof buildingcandidatesets, FP-Growthfocuses on"growing" smaller frequent patterns (fragments) by adding items at their frequent ends. This avoids the costly generation and scanning of redundant patterns.

### AdvantagesofFP-GrowthoverApriori

- **Fasterforlargedatasets**:Nomorecandidateexplosions, justtargetedpatterngrowth.

- **Lessmemoryrequired**:ThecompactFP-Treeminimizes memoryusage.
- **Moreversatile**:Caneasilymineconditionalfrequentpatternswithoutbuildingnewtrees.

## WhentoChooseFP-Growth

- Ifyou'redealingwithlargedatasetsandwant fasterresults.
- Ifmemorylimitationsareaconcern.
- Ifyouneedtomineconditionalfrequent patterns.

**Remember:** BothApriori andFP-Growthhavetheirstrengthsandweaknesses.Choosingthe right tool depends on your specific data and needs.

# Apriorialgorithm

**Apriori algorithm**was the first algorithmthat was proposed for frequent itemset mining. It was introduced by **R Agarwal** and **R Srikant**.

Nameofthealgorithmis **Apriori** becauseituses**priorknowledge** offrequentitemset properties.

## FrequentItemSet

- Frequent Itemset is an itemset whose support value is **greater than a threshold value**(support).

Apriorialgorithmuses frequent itemsets to generate association rules. To improve the efficiency oflevel-wisegenerationoffrequentitemsets,animportantpropertyisusedcalled **Apriori property** which helps by reducing the search space.

## AprioriProperty

- Allsubsetsofa frequentitemsetmustbefrequent(Aprioriproperty).
- Ifanitemsetisinfrequent, allitssupersetswillbeinfrequent.

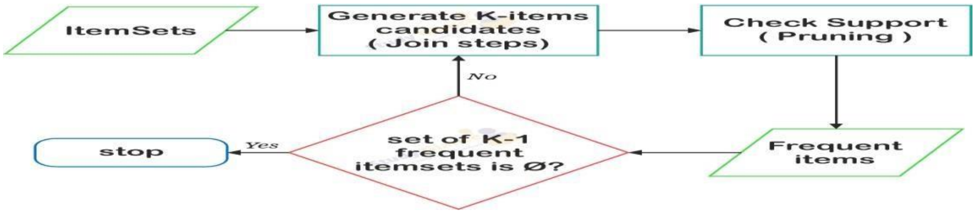
## StepsinApriorialgorithm

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the givendatabase. A**minimum support threshold** is given inthe problemor it is assumed bythe user.
ThestepsfollowedintheAprioriAlgorithmofdataminingare:

**JoinStep:**Thisstep generates(K+1) itemsetfromK-itemsetsbyjoining each itemwith itself.

**Prune Step:** This step scans the count of each item in thedatabase. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

Theabove**join** andthe**prune**steps iterativelyuntilthemostfrequentitemsetsare achieved.



## AprioriAlgorithmExample

Consider the following dataset and find frequent item sets and generate association rules for them. Assume that minimumsupport threshold (s= 50%) and minimumconfident threshold (c = 80%).

| Transaction | Listofitems |
|---|---|
| T1 | I1,I2,I3 |
| T2 | I2,I3,I4 |
| T3 | I4,I5 |
| T4 | I1,I2,I4 |
| T5 | I1,I2,I3,I5 |
| T6 | I1,I2,I3,I4 |

## Solution

**Findingfrequentitemsets:**

Supportthreshold=50%⇒0.5*6=3⇒min_sup= 3

**Step-1:**

(i) Createatable containing support count ofeachitempresent indataset –Called C1 (candidate set).

| Item | Count |
|---|---|
| I1 | 4 |
| I2 | 5 |
| I3 | 4 |
| I4 | 4 |
| I5 | 2 |

**(ii) Prune Step:**Compare candidate set item’s support count with minimum support count. The above table shows that I5 itemdoes not meet min_sup = 3, thus it is removed, only I1, I2, I3, I4 meet min_sup count.

ThisgivesusthefollowingitemsetL1.

| Item | Count |
|---|---|
| I1 | 4 |
| **Item** | **Count** |

| | |
|---|---|
| I2 | 5 |
| I3 | 4 |
| I4 | 4 |

**Step-2:**

(i) **Join step:** Generate candidate set C2 (2-itemset) using L1.And find out the occurrences of2-itemset from the given dataset.

| **Item** | **Count** |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 3 |
| I1,I4 | 2 |
| I2,I3 | 4 |
| I2,I4 | 3 |
| I3,I4 | 2 |

**(ii) Prune Step:** Compare candidate set item's support count with minimum support count. The above table shows that itemsets {I1, I4} and{I3, I4} does not meet min_sup = 3, thus those are removed.

ThisgivesusthefollowingitemsetL2.

| **Item** | **Count** |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 3 |
| I2,I3 | 4 |
| I2,I4 | 3 |

**Step-3:**

**(i) Join step:** Generate candidate set C3 (3-itemset) using L2.And find out the occurrences of 3-itemset from the given dataset.

| Item | Count |
|---|---|
| I1,I2,I3 | 3 |
| I1,I2,I4 | 2 |
| I1,I3,I4 | 1 |
| I2,I3,I4 | 2 |

**(ii) Prune Step:** Compare candidate set item's support count with minimum support count. The above table showsthat itemset {I1, I2, I4},{I1,I3, I4}and{I2, I3, I4}doesnot meet min_sup = 3, thus those are removed. Only the itemset **{I1, I2, I3}** meet min_sup count.

## GenerateAssociationRules:

Thus, we have discovered allthe frequent item-sets. Now we need to generate strong association rules (satisfies the minimum confidence threshold) from frequent item sets. For that we need to calculate confidence of each rule.
ThegivenConfidencethresholdis80%.

Theallpossibleassociationrulesfromthefrequentitemset{I1,I2,I3}are:

***{I1, I2} ⇒ {I3}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I1, I2\}} = (3/\ 4)^*\ 100 = 75\%\ (\textbf{Rejected})$$

***{I1, I3} ⇒ {I2}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I1, I3\}} = (3/\ 3)^*\ 100 = 100\%\ (\textbf{Selected})$$

***{I2, I3} ⇒ {I1}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I2, I3\}} = (3/\ 4)^*\ 100 = 75\%\ (\textbf{Rejected})$$

***{I1} ⇒ {I2, I3}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I1\}} = (3/\ 4)^*\ 100 = 75\%\ (\textbf{Rejected})$$

***{I2} ⇒ {I1, I3}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I2\}} = (3/\ 5)^*\ 100 = 60\%\ (\textbf{Rejected})$$

***{I3} ⇒ {I1, I2}***

$$\text{Confidence} = \frac{\text{support}\ \{I1, I2, I3\}}{\text{support}\ \{I3\}} = (3/\ 4)^*\ 100 = 75\%\ (\textbf{Rejected})$$

Thisshowsthat the associationrule {I1, I3} ⇒{I2}isstrong ifminimumconfidence threshold is 80%.

## Exercise1:AprioriAlgorithm

| TID | Items |
|---|---|
| T1 | I1,I2,I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

Consider the above dataset and find frequentitem sets and generate association rules for them. Assume that Minimum support count is 2 and minimum confident threshold (c = 50%).

## Exercise2:AprioriAlgorithm

| TID | Items |
|---|---|
| T1 | {milk,bread} |
| T2 | {bread,sugar} |
| T3 | {bread,butter} |
| T4 | {milk,bread,sugar} |
| T5 | {milk,bread,butter} |
| T6 | {milk,bread,butter} |
| T7 | {milk,sugar} |
| T8 | {milk,sugar} |
| T9 | {sugar,butter} |
| T10 | {milk,sugar,butter} |
| T11 | {milk,bread,butter} |

Consider the above dataset and find frequentitem sets and generate association rules for them. Assume that Minimum support count is 3 and minimum confident threshold (c = 60%).

## Association Rule Mining:

- Association rule mining is a popular and well researched method for discovering interesting relations between variables in large databases.
- It is intended to identify strong rules discovered in databases using different measures of interestingness.
- Based on the concept of strong rules, Rakesh Agrawaletal.introducedassociation rules.

### ProblemDefinition:

Theproblem ofassociation ruleminingis definedas:

Let$I = \{i_1, i_2, \ldots, i_n\}$beaset of$n$binaryattributescalled*items*.

Let$D = \{t_1, t_2, \ldots, t_m\}$beasetoftransactionscalledthe*database*.

EachtransactioninDhasauniquetransactionID andcontainsasubsetofthe itemsin$I$.

A*rule*isdefinedasanimplicationoftheform$X \Rightarrow Y$ where
$X, Y \subseteq I$and$X \cap Y = \emptyset$.

Thesetsofitems(forshort*itemsets*)$X$and$Y$arecalled*antecedent*(left-hand-sideorLHS)and *consequent*(right-hand-sideorRHS)oftherule respectively.

**Example:**

To illustrate the concepts, we use a small example from the supermarket domain. The set ofitems is $I = \{\mathrm{milk}, \mathrm{bread}, \mathrm{butter}, \mathrm{beer}\}$and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table.

An example rule for the supermarket could be $\{\mathrm{butter}, \mathrm{bread}\} \Rightarrow \{\mathrm{milk}\}$meaning that if butter and bread are bought, customers also buy milk.

Exampledatabasewith 4items and5 transactions

| TransactionID | milk | bread | butter | beer |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 |

## Important concepts of Association Rule Mining:

- The **support** $\mathrm{supp}(X)$ of an itemset $X$ is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset $\{\mathrm{milk}, \mathrm{bread}, \mathrm{butter}\}$ hasasupportof $1/5 = 0.2$ sinceitoccursin20%ofall transactions (1 out of 5 transactions).

- The **confidence**of a rule is defined

  $$\mathrm{conf}(X \Rightarrow Y) = \mathrm{supp}(X \cup Y)/\mathrm{supp}(X).$$

  For example, the rule $\{\mathrm{butter}, \mathrm{bread}\} \Rightarrow \{\mathrm{milk}\}$ has a confidence of $0.2/0.2 = 1.0$ in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter andbread,milkisboughtaswell).Confidencecanbeinterpretedasanestimateofthe probability $P(Y|X)$, the probability of finding the RHS of the rule in transactionsunder the condition that these transactions also contain the LHS.

- The ***lift*** of a rule is defined as

$$\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X) \times \mathrm{supp}(Y)}$$

ortheratiooftheobservedsupporttothatexpectedifXandYwereindependent.The rule $\{\mathrm{milk, bread}\} \Rightarrow \{\mathrm{butter}\}$ has a lift of $\frac{0.2}{0.4 \times 0.4} = 1.25$.

- The **conviction** of a rule is defined as

$$\mathrm{conv}(X \Rightarrow Y) = \frac{1 - \mathrm{supp}(Y)}{1 - \mathrm{conf}(X \Rightarrow Y)}.$$

The rule $\{\mathrm{milk, bread}\} \Rightarrow \{\mathrm{butter}\}$ has aconvictionof $\frac{1 - 0.4}{1 - .5} = 1.2$,

and can be interpreted as the ratio of the expected frequency that X occurs without Y(that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions.

# EfficientFrequentItemsetMiningMethods:

## Finding Frequent Itemsets by Confined Candidate Generation: The Apriori Algorithm

- Apriori is a seminal algorithm proposed byR. Agrawal and R. Srikant in 1994 formining frequent itemsets for Boolean association rules.
- Thenameofthealgorithm is based on thefact that thealgorithm uses *prior knowledge* of frequent itemset properties.
- Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are used to explore (*k*+1)-itemsets.
- First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted $L1$.Next, $L1$ is used to find $L2$, the set of frequent 2-itemsets, which is used to find $L3$, and so on, until no more frequent *k*-itemsets can be found.
- The finding of each $L_k$requires one full scan of the database.
- Atwo-step process isfollowed in Aprioriconsistingof joinandprune action.

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- $D$, a database of transactions;
- $min\_sup$, the minimum support count threshold.

**Output:** $L$, frequent itemsets in $D$.

**Method:**

```
L1 = find_frequent_1-itemsets(D);
for (k = 2; L(k-1) ≠ φ; k++) {
    Ck = apriori_gen(L(k-1));
    for each transaction t ∈ D { // scan D for counts
        Ct = subset(Ck, t); // get the subsets of t that are candidates
        for each candidate c ∈ Ct
            c.count++;
    }
    Lk = {c ∈ Ck | c.count ≥ min_sup}
}
return L = ∪k Lk;

procedure apriori_gen(L(k-1):frequent (k − 1)-itemsets)
for each itemset l1 ∈ L(k-1)
    for each itemset l2 ∈ L(k-1)
        if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k−2] = l2[k−2]) ∧ (l1[k−1] < l2[k−1]) then {
            c = l1 ⋈ l2; // join step: generate candidates
            if has_infrequent_subset(c, L(k-1)) then
                delete c; // prune step: remove unfruitful candidate
            else add c to Ck;
        }
return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
        L(k-1): frequent (k − 1)-itemsets); // use prior knowledge
for each (k − 1)-subset s of c
    if s ∉ L(k-1) then
        return TRUE;
return FALSE;
```

## Example:

| TID | Listof itemIDs |
|---|---|
| T100 | I1,I2,I5 |
| T200 | I2,I4 |
| T300 | I2,I3 |
| T400 | I1,I2,I4 |
| T500 | I1,I3 |
| T600 | I2,I3 |
| T700 | I1,I3 |
| T800 | I1,I2,I3,I5 |
| T900 | I1,I2,I3 |

Thereareninetransactionsinthisdatabase,thatis,|D| =9.

## Steps:

**1.** In the first iteration of the algorithm, each item is a member of the set of candidate1-itemsets, C1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

**2.** Suppose that the minimum support count required is 2, that is, min sup = 2. The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.In our example, all of the candidates in C1 satisfy minimum support.

**3.** To discover the set of frequent 2-itemsets, L2, the algorithm uses the join L1 on L1 to generate a candidate set of 2-itemsets, C2.No candidates are removed fromC2 during the prune step because each subset of the candidates is also frequent.

**4.** Next, the transactions in D arescanned and the support count of each candidate itemsetInC2 is accumulated.

**5.** The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate2-itemsets in C2 having minimum support.

**6.** The generation of the set of candidate 3-itemsets,C3, Fromthejoin step, we first getC3 =L2x L2= ({I1,I2,I3},{I1,I2,I5},{I1,I3,I5},{I2,I3,I4},{I2,I3,I5},{I2,I4,I5}.Based on the Apriori property that all subsets of a frequentitemsetmust also be frequent, we can determine that the four latter candidates cannotpossibly be frequent.

**7.** ThetransactionsinDarescannedinordertodetermineL3,consistingofthosecandidate 3-itemsets in C3 having minimum support.

**8.** ThealgorithmusesL3xL3togenerateacandidatesetof4-itemsets,C4.

Scan $D$ for count of each candidate →

$C_1$

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

Compare candidate support count with minimum support count →

$L_1$

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

Generate $C_2$ candidates from $L_1$ →

$C_2$

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

Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count →

$L_2$

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

Generate $C_3$ candidates from $L_2$ →

$C_3$

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

Scan $D$ for count of each candidate →

$C_3$

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

Compare candidate support count with minimum support count →

$L_3$

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

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

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

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

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of $L_2$. Therefore, keep {I1, I2, I3} in $C_3$.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of $L_2$. Therefore, keep {I1, I2, I5} in $C_3$.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I1, I3, I5} from $C_3$.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I3, I4} from $C_3$.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I3, I5} from $C_3$.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of $L_2$, and so it is not frequent. Therefore, remove {I2, I4, I5} from $C_3$.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

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

## Generating Association Rules from Frequent Itemsets:

Oncethefrequentitemsetsfromtransactionsinadatabase*D*havebeenfound,itis straightforward to generate strong association rules from them.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

The conditional probability is expressed in terms of itemset support count, where $support\_count(A \cup B)$ is the number of transactions containing the itemsets $A \cup B$, and $support\_count(A)$ is the number of transactions containing the itemset $A$. Based on this equation, association rules can be generated as follows:

- For each frequent itemset $l$, generate all nonempty subsets of $l$.
- For every nonempty subset $s$ of $l$, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where $min\_conf$ is the minimum confidence threshold.

**Example:**

**Generating association rules.** Let's try an example based on the transactional data for *AllElectronics* shown in Table 5.1. Suppose the data contain the frequent itemset $l = \{I1, I2, I5\}$. What are the association rules that can be generated from $l$? The nonempty subsets of $l$ are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}. The resulting association rules are as shown below, each listed with its confidence:

$$\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}$$

## From Association Analysis to Correlation Analysis:

- A correlation measure can be used to augment the support-confidence framework for association rules. This leads to *correlation rules* of the form *A*=>*B*[*support*, *confidence*, *correlation*]
- That is, a correlation rule is measured not only by its support and confidence but also by the correlation between itemsets *A* and *B*. There are many different correlation measures from which to choose. In this section, we study various correlation measures to determine which would be good for mining large data sets.

- Liftisasimplecorrelationmeasurethatisgivenasfollows.Theoccurrenceofitemset *A* is independent of the occurrence of itemset *B* if $P(A \cup B)$ = $P(A)P(B)$; otherwise, itemsets *A* and *B* are dependent and correlated as events. This definition can easily be extended to more than two itemsets.

The lift between the occurrence of *A* and *B* can be measured by computing

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- If the lift(A,B)islessthan1,thentheoccurrenceofAisnegativelycorrelatedwiththe occurrence of B.
- Iftheresultingvalueisgreaterthan1,then A and B are positively correlated,meaningthat the occurrence of one implies the occurrence of the other.
- Iftheresultingvalueisequalto1,then A and B are independent and there is no correlation between them.

## Frequent Pattern Growth Algorithm

The two primary drawbacks of the Apriori Algorithm are: 1. At each step, candidate sets have to be built. 2. To build the candidate sets, the algorithm has to repeatedly scan the database. These two properties inevitably make the algorithm slower. To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm. It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure. Consider the following data:-

| Transaction ID | Items |
|---|---|
| T1 | {E, K, M, N, O, Y} |
| T2 | {D, E, K, N, **O**, Y} |
| T3 | {A, E, K, M} |
| T4 | {C, K, M, U, Y} |
| T5 | {C, E, I, K, O, O} |

The above-given data is a hypothetical dataset of transactions with each letter representing an item. The frequency of each individual item is computed:-

| Item | Frequency |
|---|---|
| A | 1 |
| C | 2 |
| D | 1 |
| E | 4 |
| I | 1 |
| K | 5 |
| M | 3 |
| N | 2 |
| 0 | 3 |
| U | 1 |
| Y | 3 |

Let the minimum support be 3. A Frequent Pattern set is built which will contain all the elements whose frequency is greater than or equal to the minimum support. These elements are stored in descending order of their respective frequencies. After insertion of the relevant items, the set L as follows:-

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

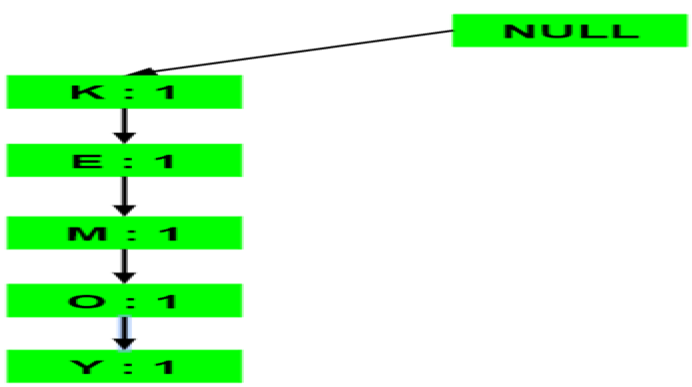
Now, for each transaction, the respective Ordered-Item set is built. It is done by iterating the Frequent Pattern set and checking if the current item is contained in the transaction in question. If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction. The following table is built for all the transactions:

| Transaction ID | Items | Ordered-Item Set |
|---|---|---|
| T1 | {E, K, M, N, O, Y} | {K, E, M, O, Y} |
| T2 | {D, E, K, N, O, Y} | {K, E, O, Y} |
| T3 | { A, E, K, M} | {K, E, M} |
| T4 | {C, K, M, U, Y} | {K, M, Y} |
| T5 | {C, E, I, K, O, O} | {K, E, O} |

Now, all the Ordered-Item sets are inserted into a Trie Data Structure.

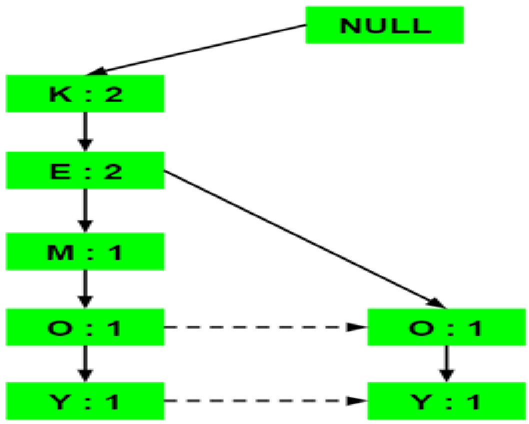
**Inserting the set {K, E, M, O, Y}:**

Here, all the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.



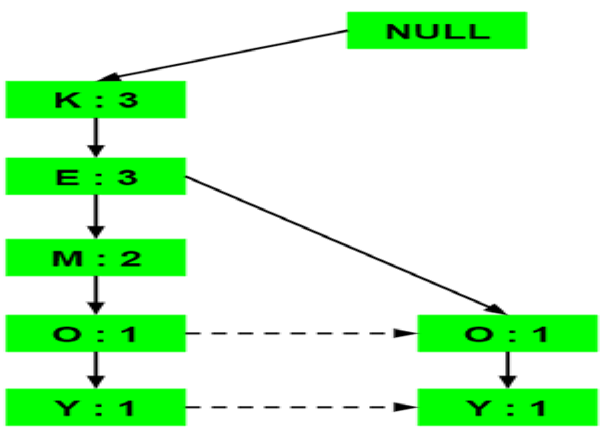

**Inserting the set {K, E, O, Y}:**

Till the insertion of the elements K and E, simply the support count is increased by 1. On inserting O we can see that there is no direct link between E and O, therefore a new node for the item O is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.



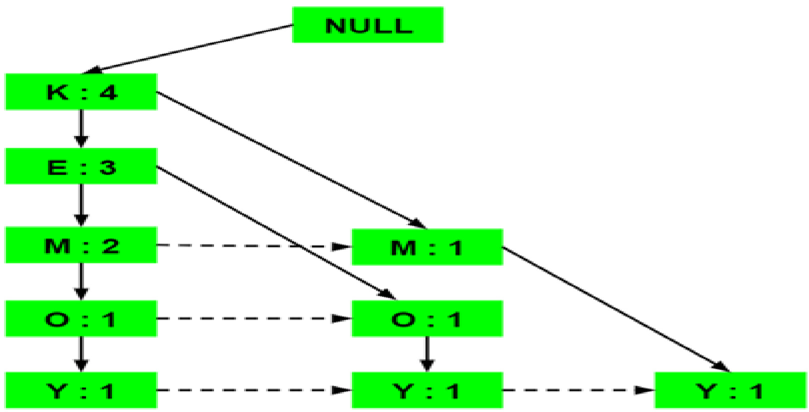

**Inserting the set {K, E, M}:**

Here simply the support count of each element is increased by 1.



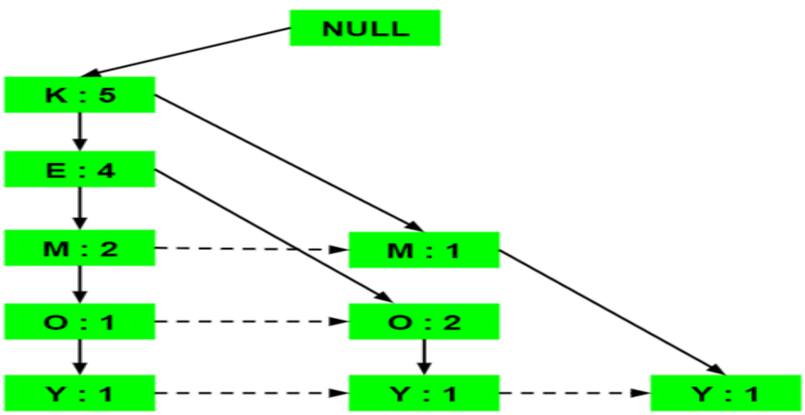

**Inserting the set {K, M, Y}:**

Similar to step b), first the support count of K is increased, then new nodes for M and Y are initialized and linked accordingly.




**Inserting the set {K, E, O}:**

Here simply the support counts of the respective elements are increased. Note that the support count of the new node of item O is increased.




Now, for each item, the Conditional Pattern Base is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies. Now for each item, the Conditional Frequent Pattern Tree is built. It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base. From the Conditional Frequent Pattern tree, the Frequent Pattern rules are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item as given in the below table. For each row, two types of association rules can be inferred for example for the first row which contains the element, the rules K -> Y and Y -> K can be inferred. To determine the valid rule, the confidence of both the rules is calculated and the one with confidence greater than or equal to the minimum confidence value is retained.

## Improving the Efficiency of Apriori:

"How can we further improve the efficiency of Apriori-based mining?" Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the original algorithm. Several of these variations are summarized as follows:

$H_2$

Create hash table $H_2$ using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

| bucket address | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| bucket count | 2 | 2 | 4 | 2 | 2 | 4 | 4 |
| bucket contents | {I1, I4}<br>{I3, I5} | {I1, I5}<br>{I1, I5} | {I2, I3}<br>{I2, I3}<br>{I2, I3}<br>{I2, I3} | {I2, I4}<br>{I2, I4} | {I2, I5}<br>{I2, I5} | {I1, I2}<br>{I1, I2}<br>{I1, I2}<br>{I1, I2} | {I1, I3}<br>{I1, I3}<br>{I1, I3}<br>{I1, I3} |

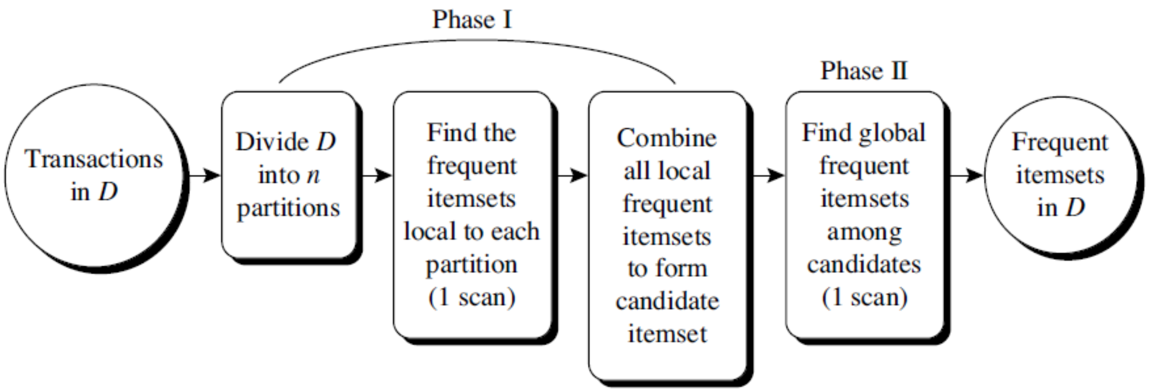
Hash table, $H_2$, for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining $L_1$. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in $C_2$.

**Hash-based technique (hashing itemsets into corresponding buckets):** A hash-based technique can be used to reduce the size of the candidate k-itemsets, Ck, for k > 1. For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L1, we can generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different buckets of a hash table structure, and increase the corresponding bucket counts. A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash-based technique may substantially reduce the number of candidate k-itemsets examined.

**Transaction reduction (reducing the number of transactions scanned in future iterations):**

A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k C1)-itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent database scans for j-itemsets, where j > k, will not need to consider such a transaction.

**Partitioning (partitioning the data to find candidate itemsets):** A partitioning technique can be used that requires just two database scans to mine the frequent itemsets. It consists of two phases. In phase I, the algorithm divides the transactions of D into n nonoverlapping partitions. If the minimum relative support threshold for transactions in D is min sup, then the minimum support count for a partition is min sup _ the number of transactions in that partition. For each partition, all the local frequent itemsets (i.e., the itemsets frequent within the partition) are found. A local frequent itemset may or may not be frequent with respect to the entire database, D. However, any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions.Therefore, all local frequent itemsets are candidate itemsets with respect to D. The collection of frequent itemsets from all partitions forms the global candidate itemsets with respect to D. In phase II, a second scan of D is conducted in which the actual support of each candidate is assessed to determine the global frequent itemsets. Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.




Mining by partitioning the data.

**Sampling (mining on a subset of the given data):** The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency. The S sample size is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S is required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets. To reduce this possibility, we use a lower support threshold than minimum support to find the frequent itemsets local to S (denoted LS). The rest of the database is then used to compute the actual frequencies of each itemset in LS. A mechanism is used to determine whether all the global frequent itemsets are included in LS. If LS actually contains all the frequent itemsets in D, then only one scan of D is required. Otherwise, a second pass can be done to find the frequent itemsets that were missed in the first pass. The sampling approach is especially beneficial when efficiency is of utmost importance such as in computationally intensive applications that must be run frequently.

**Dynamic itemset counting (adding candidate itemsets at different points during a scan):** A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan. The technique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.