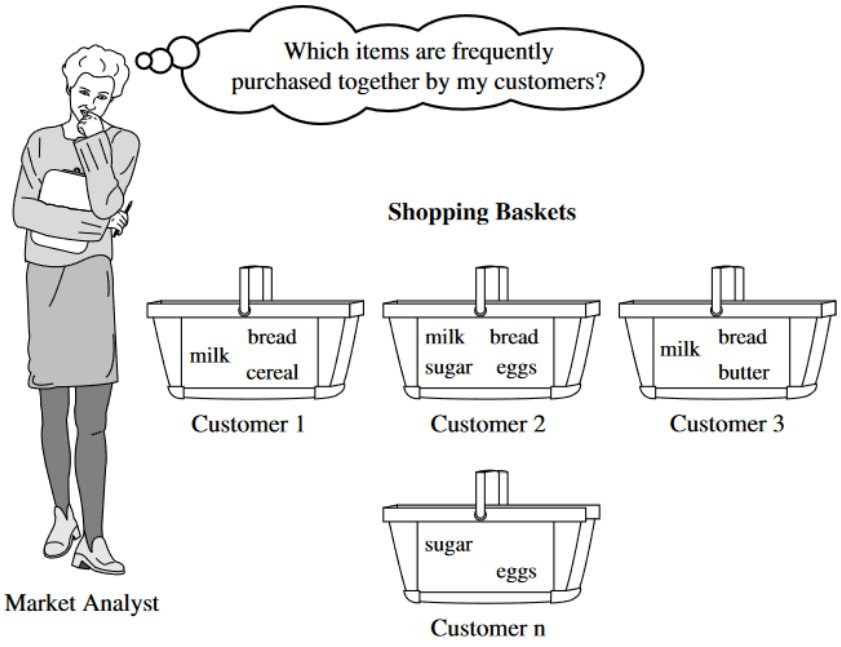
Frequent patterns

Frequent patterns are patterns (such as itemsets, subsequences, or substructures) that appear in a data set frequently. For example:

- A set of items, such as milk and bread, that appear frequently together in a transaction data set is a frequent itemset.
- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a frequent sequential pattern.
- A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a frequent structured pattern.

Finding such 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** as well. Thus, frequent pattern mining has become an important data mining task and a focused theme in data mining research.

Market basket analysis (Frequent Itemset Mining)



Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets. With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.

The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes, such as catalog design, cross-marketing, and customer shopping behavior analysis.

Frequent itemsets, closed itemsets, association rules

Let,

- $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items
- D be the set of task-relevant database
- $T_i = \{I_a, I_b, \ldots I_n\}: T \subseteq I$ are the set of transactions where each transaction is associated with unique identifier called TID.
- **A** be a set of items then a transaction T is said to contain A iff $A \subseteq T$.

k-itemset | k-Candidate itemset:

An itemset that contains k items

Frequency, Support Count or Count:

Frequency of an itemset is the number of transactions in D that contain the itemset

Association Rule:

An association rule is an implication of the form

$$A \Rightarrow B, \text{ where } A \subset I, B \subset I, \text{ and } A \cap B = \varphi.$$

Support ($A \Rightarrow B$) | Relative Support:

is the percentage of transactions in D that contain both A and B.

$$support(A \Rightarrow B) = P(A \cup B) = support(A \cup B)$$

Confidence ($A \Rightarrow B$):

is the percentage of transactions in D containing A that also contains B

$$confidence(A \Rightarrow B) = P\left(\frac{B}{A}\right) = \frac{support(A \cup B)}{support(A)}$$

Frequent Itemset:

If the relative support of an itemset I satisfies a pre-specified minimum support threshold (min_sup), then I is a frequent itemset. The set of frequent k-itemsets is denoted by L_k.

Closed Itemsets:

An itemset is closed if **none of its immediate supersets have same support count** same as Itemset.

Maximal Itemset:

An itemset is maximal frequent if none of its supersets are frequent.

For example:

TID	Items
1	{A, C, D}
2	{B, C, D}
3	{A, B, C, D}
4	{B, D}
5	{A, B, C, D}

Suppose minimum support count is 3

1-candidate itemset (C_1):

TID	**Support count**
{A}	3
{B}	4
{C}	4
{D}	5

1- Frequent itemsets (L_1):{A}, {B}, {C}, {D}

2-candidate itemset (C_2):

TID	**Support count**
{A, B}	2
{A, C}	3

{A, D}	3
{B, C}	3
{B, D}	4
{C, D}	4

2-Frequent itemsets (L_2):{A,C}, {A,D}, {B,C},{B,D}, {C,D}

3-candidate itemset (C_2):

TID	Support count
{A, C, D}	3
{B, C, D}	3

3-Frequent itemsets (L_3):{A, C, D}, {B, C,D}

Types of association rule (Single dimensional, multidimensional, multilevel, quantitative)

Single Dimensional:

If the items or attributes in an association rule reference only one dimension, then it is a single-dimensional association rule.

For example:

buys(X, "computer”) => buys(X, “antivirus software")

Multi-dimensional:

If a rule references two or more dimensions, such as the dimensions age, income and buys, then it is a multidimensional association rule.

For example:

age(X, “30...39") ^ income(X, “42K...48K”) => buys(X, “high resolution TV”)

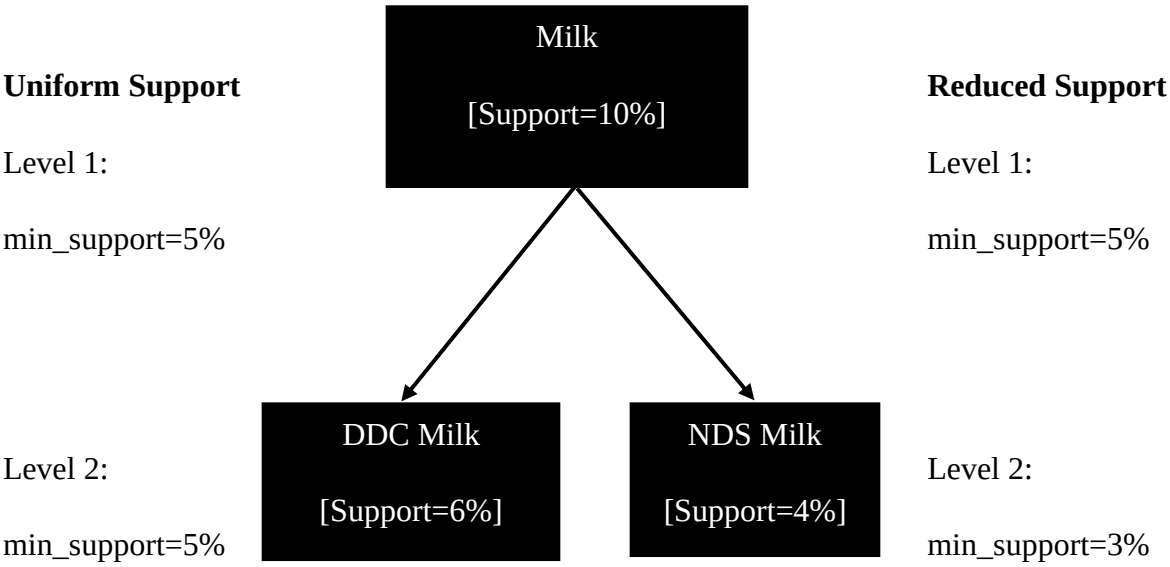
Multilevel association:

Association rules generated from mining data at multiple levels of abstraction are called multiple-level or multilevel association rules. This can be done in two ways:

– Using **uniform minimum support** for all levels

– Using **reduced minimum support** at lower levels

For example:



Quantitative Association:

Quantitative association is a type of multi-dimensional association rule where **at least one attribute** (left or right) of quantitative association rules **is a numeric attribute**.

For example:

age(X, “30...39") ^ income(X, “42K...48K”) => buys(X, “high resolution TV”)

It is unlike general association rules, where both left and right sides of the rule should be categorical (nominal or discrete) attributes.

Finding frequent itemset (Apriori algorithm, FP growth)

Apriori algorithm:

Apriori algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules is based on the fact that uses **prior knowledge of frequent itemset properties**, as we shall see following. 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 L_1**. Next, **L_1 is used to find L_2,** the set of frequent 2-itemsets, which is used to find L_3, 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**.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used.

Apriori Property:

All nonempty subsets of a frequent itemset must also be frequent

Apriori Algorithm is hence carried out in two steps:

1. **Join Step:** To find L_k, a set of candidate k-itemsets C_k is generated by joining L_{k-1} with itself.
2. **Prune Step:** C_k is a superset of L_k. To reduce the size of C_k, the Apriori property is used as follows. Any (k − 1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k − 1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be **removed or pruned** from C_k.

Pseudo-code:

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

C_1={1-itemset}

L_1 = {frequent items};

for (k = 1; L_k !=∅; k++) do begin

C_{k+1} = candidates generated from L_k;

for each transaction t in database do

increment the count of all candidates in C_{k+1} that are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

End

return $\cup_k L_k$;

Example:

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

min_sup=2 and min_confidence=3/4=75%

After First Scan, we obtain 1-candidate itemset C_1 as:

Itemset	Support_Count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Since {D} does not satisfy minimum support we can neglect it and 1-frequent itemset L_1 is:

Itemset	Support_Count
{A}	2
{B}	3
{C}	3
{E}	3

Similarly, we can generate C_2

Itemset	Support_Count
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

Generating L_2

Itemset	Support_Count
{A,C}	2
{B,C}	2
{B,E}	3
{C,E}	2

Computing C_3 using L_2

Itemset	Support_Count
{B, C, E}	2

Generating 3-frequent itemset L_3 from 3-candidate itemsets C_3

Itemset	Support_Count
{B, C, E}	2

Generating Association Rules from frequent itemset

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

Now, the association rules we can obtain are:

Association Rules	Confidence
B ∧ C=>E	$= \frac{support(B \cup C \cup E)}{support(B \cup C)} = \frac{2}{2} = 100\%$
B ∧ E=>C	$= \frac{support(B \cup C \cup E)}{support(B \cup E)} = \frac{2}{3} = 66.67\%$
C ∧ E=>B	100%
B=> C ∧ E	66.67%
C=> B∧E	66.67%
E=> B∧C	66.67%

Since min_confidence is 75% the valid association rules are:

B ∧ C=>E and

C ∧ E=>B

Limitation of Apriori algorithm:

- Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge.
- Apriori needs multiple scans of the database to check the support of each itemset generated and this leads to high costs.

These can be eliminated using the FP growth algorithm.

Improving the Efficiency of Apriori Algorithm:

- **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, from the candidate 1-itemsets in C1, we can generate all of 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.

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

H_2

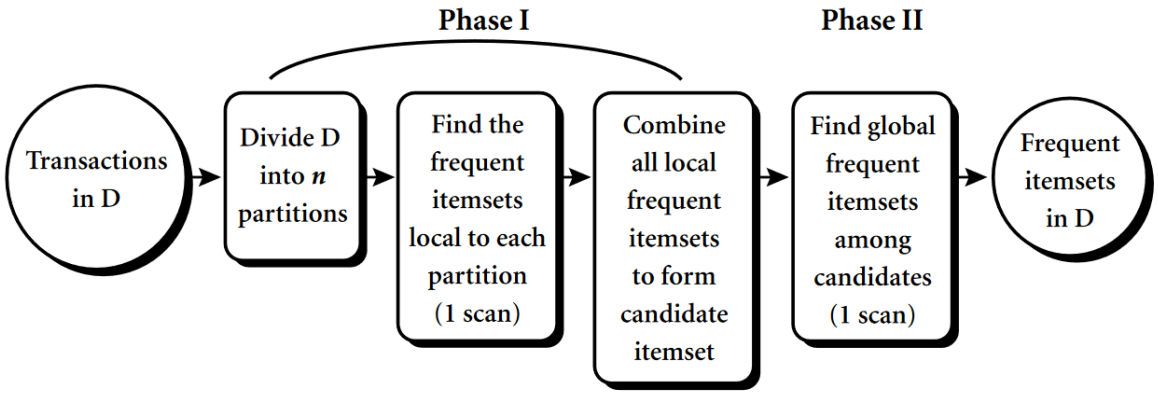
bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

- **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 + 1)-itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent scans of the database for j-itemsets, where j > k, will not require it

- **Partitioning (partitioning the data to find candidate itemsets):**

Partitioning technique requires just two database scans and is carried out in two phases as:



- **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.

- **Dynamic itemset counting (adding candidate itemsets at different points during a scan):**

Here, 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 is dynamic in that it **estimates the support of all of the itemsets that have been counted so far**, **adding new candidate itemsets if all of their subsets are estimated to be frequent.** The resulting algorithm requires **fewer** database **scans than Apriori**.

FP-Growth Algorithm (Frequent Pattern Growth) or Mining Frequent Itemsets without Candidate Generation :

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a **frequent pattern tree or FP tree**.

This tree structure will maintain the association between the itemsets. The database is fragmented using one frequent item. This fragmented part is called “pattern fragment”. The itemsets of these fragmented patterns are analyzed. Thus, with this method, the search for frequent itemsets is reduced comparatively.

Steps carried out for FP-growth Algorithm:

1. The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.
2. The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.
3. The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.
4. The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.

This means that the common itemset is linked to the new node of another itemset in this transaction.

5. Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.
6. The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern

length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base.

Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

7. Construct a Conditional FP Tree, which is formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.
8. Frequent Patterns are generated from the Conditional FP Tree.

Example:

Transaction	**List of items**
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

Min_support=50%=0.5*6=3,

Confidence=60%

Count the Itemset in all transactions

Item	Frequency
I1	4
I2	5
I3	4
I4	4
I5	2

Sorting the itemset that satisfies min_support in descending order:

Item	Frequency
I2	5
I1	4
I3	4
I4	4

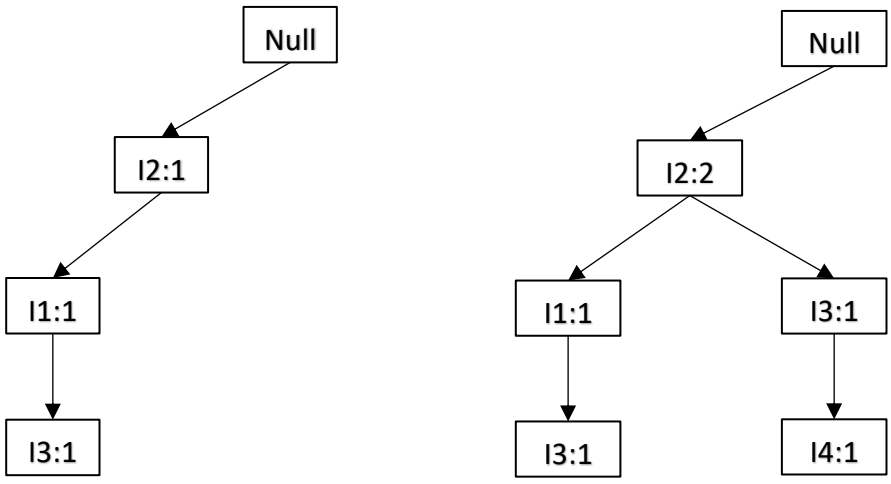
Creating Ordered Items sets for each transaction

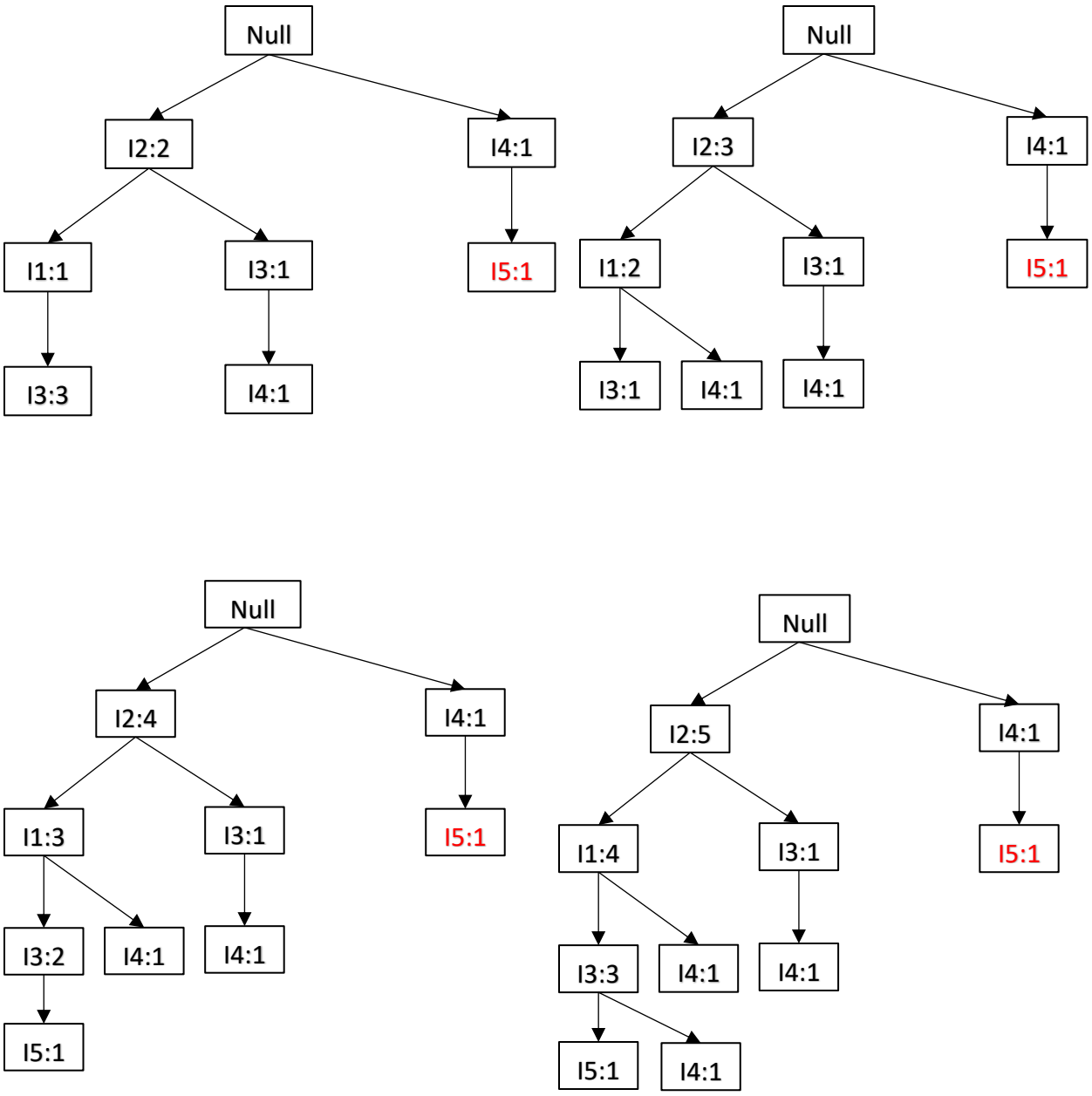
Transaction	List of items	Ordered item sets (Only the items satisfying minimum support are considered)
T1	I1, I2, I3	I2, I1, I3
T2	I2, I3, I4	I2, I3, I4
T3	I4, I5	I4
T4	I1, I2, I4	I2, I1, I4
T5	I1, I2, I3, I5	I2, I1, I3
T6	I1, I2, I3, I4	I2, I1, I3, I4

Steps for Building FP-tree:

1. Considering the root node null.

2. The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child to root, I1 is linked to I2 and I3 is linked to I1.
3. T2: I2, I3, I4 contains I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share I2 node as common as it is already used in T1.
4. Increment the count of I2 by 1 and I3 is linked as a child to I2, I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.
5. T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.
6. T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node, hence it will be incremented by 1. Similarly, I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.
7. T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.
8. T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.





Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I4	{I2,I1:1},{{I2,I1,I3:1},{I2,I3:1}	{I2:3, I1:2,I3:2}	{I2,I4:3},{I1,I4:2},{I3,I4:2},{I2,I3, I4:2},{I2,I1,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

Generating association rules from frequent itemset

Discussed above…

Limitation and improving Apriori

Discussed above…

From Association Mining to Correlation Analysis

Most association rule mining algorithms employ a support-confidence framework. Often, many interesting rules can be found using low support thresholds. Although minimum support and confidence thresholds help weed out or exclude the exploration.

> **Strong Rules Are Not Necessarily Interesting: Example**
>
> Suppose we are interested in analyzing transactions at AllElectronics with respect to the purchase of computer games and videos. Let game refer to the transactions containing computer games, and video refer to those containing videos. Of the 10,000 transactions analyzed, the data show that 6,000 of the customer transactions included computer games, while 7,500 included videos, and 4,000 included both computer games and videos. Suppose that a data mining program for discovering association rules is run on the data, using a minimum support of, say, 30% and a minimum confidence of 60%. The following association rule is discovered:
>
> $$\boldsymbol{buys}(\boldsymbol{X}, \text{“}\boldsymbol{computer\ games}\text{”}) \Rightarrow \boldsymbol{buys}(\boldsymbol{X}, \text{“}\boldsymbol{videos}\text{”})[\boldsymbol{support} = \mathbf{40}\%, \boldsymbol{confidence} = \mathbf{66}\%]$$
>
> This rule is misleading because the probability of purchasing videos is 75%, which is even larger than 66%. In fact, computer games and videos are negatively associated because the purchase of one of these items actually decreases the likelihood of purchasing the other. Without fully understanding this phenomenon, we could easily make unwise business decisions based on Rule

As we have seen above, the support and confidence measures are insufficient at filtering out uninteresting association rules. To tackle this weakness, a correlation measure can be used to augment the support-confidence framework for association rules. This leads to correlation rules of the form

$$\boldsymbol{A} \Rightarrow \boldsymbol{B}\ [\boldsymbol{support}, \boldsymbol{confidence}, \boldsymbol{correlation}]$$

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.

We can use lift for calculating the correlation.

Lift

Lift is a simple correlation measure that is given as follows. The occurrence of itemset A is independent of the occurrence of itemset B if P(A ∪ 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

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

If the value is less than 1, then the occurrence of A is negatively correlated with the occurrence of B. If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.