Mining Frequent Patterns, Associations, and Correlations: Basic Concepts and Methods

Basic Concepts

- Frequent pattern mining searches for recurring relationships in a given data set.

- the basic concepts of frequent pattern mining for the discovery of interesting associations and correlations between itemset in transactional and relational databases

Market Basket Analysis

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- 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.

Support and Confidence

- **Support**: how often a given rule appears in the database being mined
- **Confidence**: the number of times a given rule turns out to be true in practice

computer ⇒ antivirus software [support = 2%,confidence = 60%]

- Rule support and confidence are two measures of rule interestingness.
- A support of 2% for Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A confidence of 60% means that 60% of the customers who purchased

Frequent Itemsets, Closed Itemsets, and Association Rules

- Let I = {I1, I2,..., Im} be an itemset. Let D, the task-relevant data, be a set of database transactions where each transaction T is a nonempty itemset such that $T \subseteq I$.
- Each transaction is associated with an identifier, called a TID. Let A be a set of items. A transaction T is said to contain A if $A \subseteq T$.
- An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, A not equal to ∅, B not equal to ∅, and $A \cap B = \varphi$.
- The rule $A \Rightarrow B$ holds in the transaction set D with **support** s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of sets A and B say, or, both A and B). This is taken to be the probability, $P(A \cup B)$.
- The rule $A \Rightarrow B$ has **confidence** c in the transaction set D, where c is

Frequent Itemsets, Closed Itemsets, and Association Rules

$$\text{Support}(A \Rightarrow B) = P(A \cup B)$$

$$\text{Confidence}(A \Rightarrow B) = P(B|A)$$

- Rules that satisfy both a minimum support threshold (min sup) and a minimum confidence threshold (min conf) are called strong.
- By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

- $\text{confidence}(A \Rightarrow B) = P(B|A)$

$$= \text{support}(A \cup B)/\text{support}(A)$$

$$= \text{support count}(A \cup B)/ \text{support count}(A)$$

In general, association rule mining can be viewed as a two-step process:

transaction ID	items
1	{A,C,D}
2	{B,C,E}
3	{A,B,C,E}
4	{B,E}
5	{A,B,C,E}

- Let minsup= 2/5
- Total Itemsets = 31
- Frequent Itemsets = 15
- Infrequent Itemsets = 16

- $2^5 - 1$

Itemset	Support	F/I
A	3/5	F
B	4/5	F
C	4/5	F
D	1/5	I
E	4/5	F
AB	2/5	F
AC	3/5	F
AD	1/5	I
AE	2/5	F
BC	3/5	F
BD	0/5	I
BE	4/5	F
CD	1/5	I
CE	3/5	F
DE	0/5	I
ABC	2/5	F

Itemset	Support	F/I
ABD	0/5	I
ABE	2/5	F
ACD	1/5	I
ACE	2/5	F
ADE	0/5	I
BCD	0/5	I
BCE	3/5	F
BDE	0/5	I
CDE	0/5	I
ABCD	0/5	I
ABCE	2/5	F
ABDE	0/5	I
ACDE	0/5	I
BCDE	0/5	I
ABCDE	0/5	I

transaction ID	items
1	{A,C,D}
2	{B,C,E}
3	{A,B,C,E}
4	{B,E}
5	{A,B,C,E}

- Let minsup= 2/5
- Apriori Principle:
- If an Itemset is Infrequent then all its Supersets are Infrequent
- If an Itemset is frequent then all its Supersets are Frequent

Itemset	Support	F/I
A	3/5	F
B	4/5	F
C	4/5	F
D	1/5	I
E	4/5	F
AB	2/5	F
AC	3/5	F
AE	2/5	F
BC	3/5	F
BE	4/5	F
CE	3/5	F
ABC	2/5	F
ABE	2/5	F
ACE	2/5	F
BCE	3/5	F
ABCE	2/5	F

Closed Frequent Itemset

An itemset X is closed Frequent Itemset only if

- X is Frequent
- No immediate superset of X has same support as X

Itemset	Support	F/I	C/NC
A	3/5	F	NC
B	4/5	F	NC
C	4/5	F	C
D	1/5	I	NC
E	4/5	F	NC
AB	2/5	F	NC
AC	3/5	F	C
AE	2/5	F	NC
BC	3/5	F	NC
BE	4/5	F	C
CE	3/5	F	NC
ABC	2/5	F	NC
ABE	2/5	F	NC
ACE	2/5	F	NC
BCE	3/5	F	C
ABCE	2/5	F	C

Maximal Frequent Itemset

- An itemset X is maximal frequent itemset only if
  - X is Frequent
  - No immediate superset of X if frequent

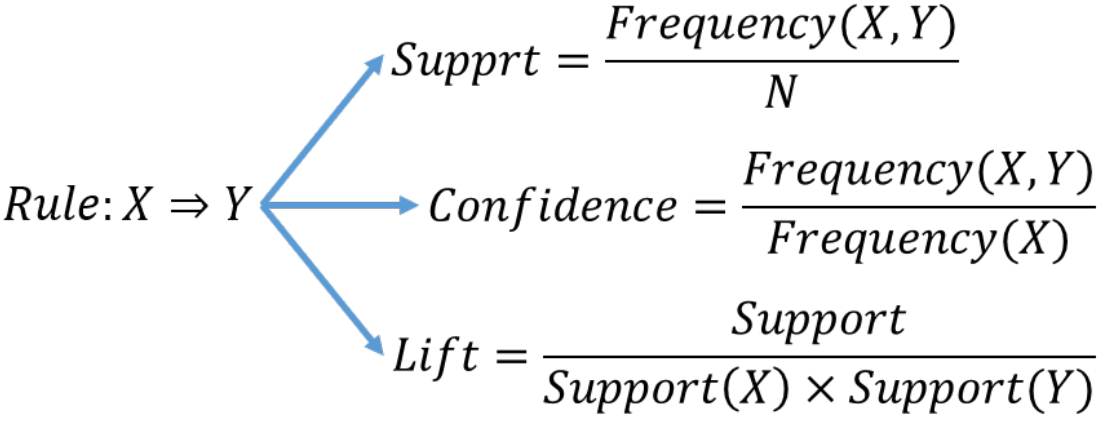
Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

- Let minsup=2/4

- $\sup(\{1,3\}) \rightarrow \frac{2}{4}$
- $\sup(\{1,2,3\}) \rightarrow \frac{1}{4}$ $\sup(\{1,3,4\}) \rightarrow \frac{1}{4}$ $\sup(\{1,3,5\}) \rightarrow \frac{1}{4}$
- $\sup(\{2,3,5\}) \rightarrow \frac{2}{4}$
- $\sup(\{1,2,3,5\}) \rightarrow \frac{1}{4}$ $\sup(\{2,3,4,5\}) \rightarrow \frac{1}{4}$

Support, Confidence and Lift



Association Rule Mining-Apriori Algorithm

Consider the following transactions.
Apply the association rule mining to get the association rules with min support of 2 and confidence of 50%

TID	List of Items IDs
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

Association Rule Mining - Subset Creation

- Frequent 3-Item Set = I => {1, 2, 3} and {1, 2, 5}
- Min_Support = 2 = 2/9 = 22.22% and Min_Confidence = 50%

- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}
  - {{1}, {2}, {5}, {1, 2}, {1, 5}, {2, 5}}

- How to form Association Rule…?
  - For every non-empty subset S of I, the association rule is,
    - **S → (I-S)**
    - **If support(I) / support(S) >= min_confidence**

Association Rule Mining - Subset Creation

- Frequent 3-Item Set = I => {1, 2, 3}
- Non-Empty subset are
  - {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}

- Rule 1: {1} → {2, 3} {S= 22.22 %, C=33.34%}
  - Support = 2/9 = 22.22%
  - Confidence = Support (1, 2, 3)/Support(1) = $\frac{2/9}{6/9}$ = 2/6 = 33.34% < 50%
  - Invalid Rule

- Rule 2: {2} → {1, 3} {S= 22.22 %, C=28.57%}
  - Support = 2/9 = 22.22 %
  - Confidence = Support (1, 2, 3)/Support(2) = 2/7 = 28.57% < 50%
  - Invalid Rule

Advantages and Disadvantages

- Advantages
  - Easy to understand algorithm
  - Join and Prune steps are easy to implement on large itemsets in large databases
- Disadvantages
  - It requires high computation if the itemsets are very large and the minimum support is kept very low.
  - The entire database needs to be scanned.

Methods To Improve Apriori Efficiency

- **Hash-Based Technique:** This method uses a hash-based structure called a hash table for generating the k-itemsets and its corresponding count. It uses a hash function for generating the table.

- **Transaction Reduction:** This method reduces the number of transactions scanning in iterations. The transactions which do not contain frequent items are marked or removed.

- **Partitioning:** This method requires only two database scans to mine the frequent itemsets. It says that for any itemset to be potentially frequent in the database, it should be frequent in at least one of the partitions of the database.

TID	List of 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

Min. Support Count=3

C1

Itemset	Support Count
I1	6
I2	7
I3	6
I4	2
I5	2

Hash Function

Itemset	Count	Hash Function
I1, I2	4	[1*10+2] mod 7=**5**
I1, I3	4	[1*10+3] mod 7=**6**
I1, I4	1	[1*10+4] mod 7=**0**
I1, I5	2	[1*10+5] mod 7=**1**
I2, I3	4	[2*10+3] mod 7=**2**
I2, I4	2	[2*10+4] mod 7=**3**
I2, I5	2	[2*10+5] mod 7=**4**
I3, I4	0	--
I3, I5	1	[3*10+5] mod 7=**0**
I4, I5	0	--

Order of Items I1=1, I2=2, I3=3, I4=4, I5=5

H(x, y)=((Order of First)* 10+(Order of Second))mod 7

Hash Table Structure to generate L2

Bucket address	0	1	2	3	4	5	6
Bucket Count	2	2	4	2	2	4	4
Bucket Contents	{I1-I4}-1 {I3-I5}-1	{I1-I5}-2	{I2-I3}-4	{I2-I4}-2	{I2-I5}-2	{I1-I2}-4	{I1-I3}-4
L2	No	No	Yes	No	No	Yes	Yes

Advantages:

- Reduce the number of scans
- Remove the large candidates that cause high Input/output cost

b) Transaction Reduction

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

Trans.	Items
T1	I1, I2, I5
T2	I2, I3, I4
T3	I3, I4
T4	I1, I2, I3, I4

	I1	I2	I3	I4	I5
T1	1	1	0	0	1
T2	0	1	1	1	0
T3	0	0	1	1	0
T4	1	1	1	1	0

	I1	I2	I2	I4	I5
T1	1	1	0	0	1
T2	0	1	1	1	0
T3	0	0	1	1	0
T4	1	1	1	1	0

Minimum Support Count=2

	I1	I2	I2	I4
T1	1	1	0	0
T2	0	1	1	1
T3	0	0	1	1
T4	1	1	1	1

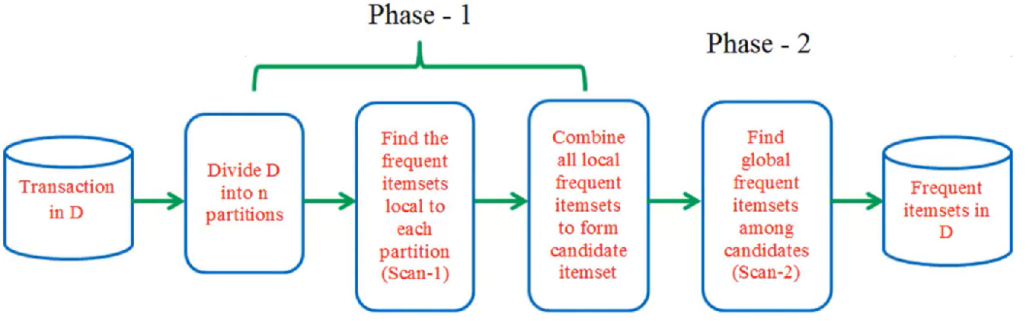
Trans.	Items
T1	I1, I2, I5
T2	I2, I3, I4
T3	I3, I4
T4	I1, I2, I3, I4

	I1,I2	I1,I3	I1,I4	I2,I3	I2,I4	I3,I4
T1	1	0	0	0	0	0
T2	0	0	0	1	1	1
T3	0	0	0	0	0	1
T4	1	1	1	1	1	1

	I1,I2	I2,I3	I2,I4	I3,I4
T2	0	1	1	1
T4	1	1	1	1

	I2,I3, I4
T2	1
T4	1

Partitioning



Partitioning

Transaction	Itemset
T1	I1,I5
T2	I2,I4
T3	I4,I5
T4	I2,I3
T5	I5
T6	I2,I3,I4

Database is divides into 3 Partitions

Partitioning method example

Transaction	Itemset
T1	I1,I5
T2	I2,I4
T3	I4,I5
T4	I2,I3
T5	I5
T6	I2,I3,I4

- Minimum support =20%
- I.e.support count = 2 (20*6/100=1.2=2)
- We partition the database in 3 parts. So each partition will be having 2 transactions and minimum support count=1
- (20*2/100=0.4=1)

Trans.	Itemset	First Scan	Second Scan	Shortlisted
		Support =20% Min. sup=1	Support =20% Min. sup=2	
T1	I1, I5	I1-1, I2-1, I4-1, I5-1	I1-1, I2-3	I2-3, I3-2
T2	I2, I4	{I1,I5}-1 {I2, I4}-1	I3-2. I4-3	I4-3 ,I5-3
T3	I4, I5	I2-1, I3-1, I4-1, I5-1	I5-3, {I1, I5}-1	{I2, I4}-2
T4	I2, I3	{I4,I5}-1,{I2, I3}-1	{I2, I4}-2, {I4, I5}-1	{I2, I3}-2
T5	I5	I2-1, I3-1,I4-1, I5-1	{I2, I3}-2, {I3,I4}-1	
T6	I2, I3, I4	{I2, I3}-1, {I2,I4}-1 {I3, I4}-1 {I2, I3, I4}-1	{I2, I3, I4}-1	

Minsup >= 3

Tid	Items
10	1,2,5
20	2,4
30	2,3
40	1,2,4
50	1,3
60	2,3
70	1,3
80	1,2,3,5
90	1,2,3

First Scan Minsup >= 1

Tid	Items	Non Empty subsets
10	1,2,5	{1}, {2}, {3}, {4}, {5}, {1,2}, {1,5}, {2,3}, {2,4}, {2,5}, {1,2,5}
20	2,4	
30	2,3	

Tid	Items	Non Empty subsets
40	1,2,4	{1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {1,2,4}
50	1,3	
60	2,3	

Tid	Items	Non Empty subsets
70	1,3	{1}, {2}, {3}, {5}, {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}, {1,2,3}, {1,2,5}, {1,3, 5}, {1,2,3,5}
80	1,2,3,5	
90	1,2,3	

Second Scan

{1} – 3
{2} – 3
{3} – 3
{4} – 2
{5} – 2
{1,2} – 3
{1,3} – 2
{1,4} – 1
{1,5} – 2
{2,3} – 3
{2,4} – 2
{2,5} – 2
{3,5} – 1
{1,2,5} – 2
{1,2,4} – 1
{1,2,3} – 1
{1,3,5} – 1
{1,2,3,5} – 1

Minsup >= 3

{1} – 3
{2} – 3
{3} – 3
{1,2} – 3
{2,3} – 3

DIC -**Dynamic Itemset Counting**

- Alternative to Apriori Itemset Generation

- Itemsets are dynamically added and deleted as transactions are read
- Relies on the fact that for an itemset to be frequent, all of its subsets must also be frequent, so we only examine those itemsets whose subsets are all frequent

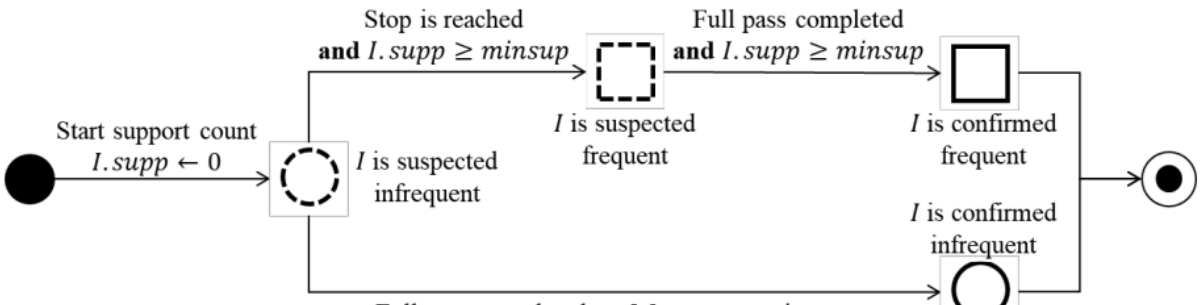


Figure 1. Lifecycle of an itemset in the DIC algorithm.

Itemsets are marked in four different ways as they are counted:

- **Solid box:** ☐ confirmed frequent itemset - an itemset we have finished counting and exceeds the support threshold *minsupp*
- **Solid circle:** ◯ confirmed infrequent itemset - we have finished counting and it is below *minsupp*
- **Dashed box:** ⬚ suspected frequent itemset - an itemset we are still counting that exceeds *minsupp*
- **Dashed circle:** ◌ suspected infrequent itemset - an itemset we are still counting that is below *minsupp*

Itemset lattices

- **Itemset lattices:** An itemset lattice contains all of the possible itemsets for a transaction database. Each itemset in the lattice points to all of its supersets. When represented graphically, a itemset lattice can help us to understand the concepts behind the DIC algorithm.
- Example: *minsupp* = 25% and M = 2.

TID	*A*	*B*	*C*
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

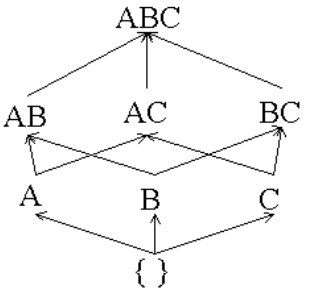
Transaction Database

DIC Algorithm:

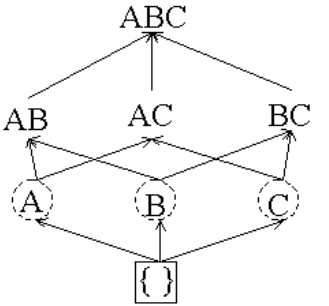
- Mark the empty itemset with a solid square. Mark all the 1-itemsets with dashed circles. Leave all other itemsets unmarked.
- While any dashed itemsets remain:
  - Read *M* transactions (if we reach the end of the transaction file, continue from the beginning). For each transaction, increment the respective counters for the itemsets that appear in the transaction and are marked with dashes.
  - If a dashed circle's count exceeds *minsupp*, turn it into a dashed square. If any immediate superset of it has all of its subsets as solid or dashed squares, add a new counter for it and make it a dashed circle.
  - Once a dashed itemset has been counted through all the transactions, make it solid and stop counting it.

DIC - transactions

- **Itemset lattice for the above transaction database:**



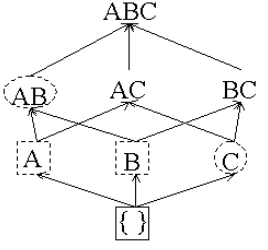
- **Itemset lattice before any transactions are read:**



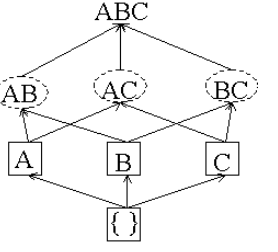
- Counters: A = 0, B = 0, C = 0
  Empty itemset is marked with a solid box.
  All 1-itemsets are marked with dashed circles.

DIC - transactions

- **After M transactions are read:**

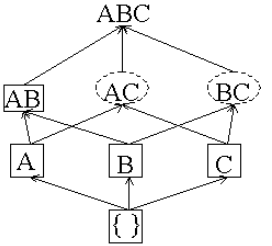


- **After 2M transactions are read**



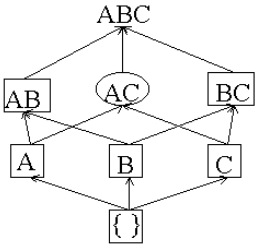
DIC - transactions

- **After 3M transactions read:**



- Counters: A = 2, B = 2, C = 1, AB = 1, AC = 0, BC = 0

- **After 4M transactions read:**



- Counters: A = 2, B = 2, C = 1, AB = 1, AC = 0, BC = 1

Frequent Pattern Growth

- Apriori candidate generate-and-test method signifi-cantly reduces the size of candidate sets, leading to good performance gain. However, itcan suffer from two nontrivial costs:
  - It may still need to generate a huge number of candidate sets. For example, if there are104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107candidate 2-itemsets.
  - It may need to repeatedly scan the whole database and check a large set of candidates bypattern matching.It is costly to go over each transaction in the database to determinethe support of the candidate itemsets.

"Can we design a method that mines the complete set of frequent itemsets without sucha costly candidate generation process?"

Frequent Pattern Growth

- 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.

FP-growth (finding frequent itemsets without candidate generation).

Min support count = 2

Tid	Items
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

- Scan the DB same as Apriori
- Derives the set of frequent items(1-itemsets) and support count

items	freq
I1	6
I2	7
I3	6
I4	2
I5	2

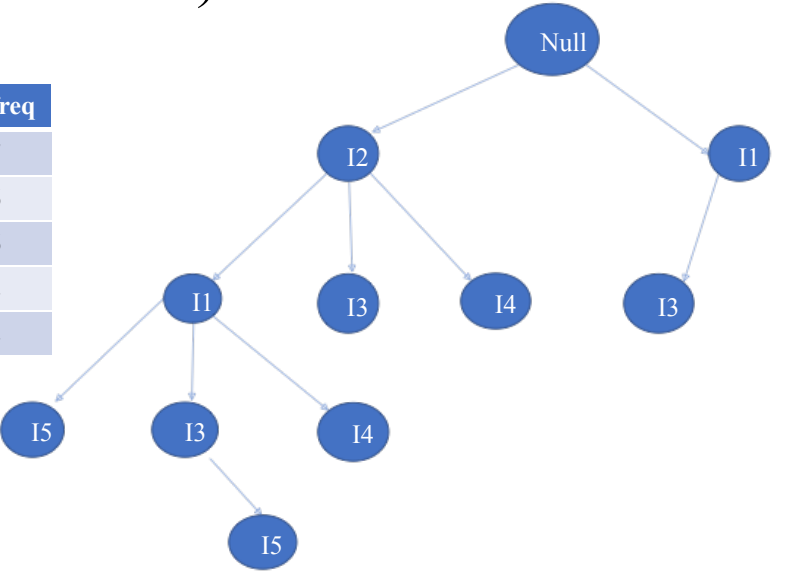
The set of frequent itemset is Sorted in the order of descending support count

items	freq
I2	7
I1	6
I3	6
I4	2
I5	2

FP-growth (finding frequent itemsets without candidate generation).

Tid	Items
T100	I2,I1,I5
T200	I2,I4
T300	I2,I3
T400	I2,I1,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I2,I1,I3,I5
T900	I2,I1,I3

items	freq
I2	7
I1	6
I3	6
I4	2
I5	2



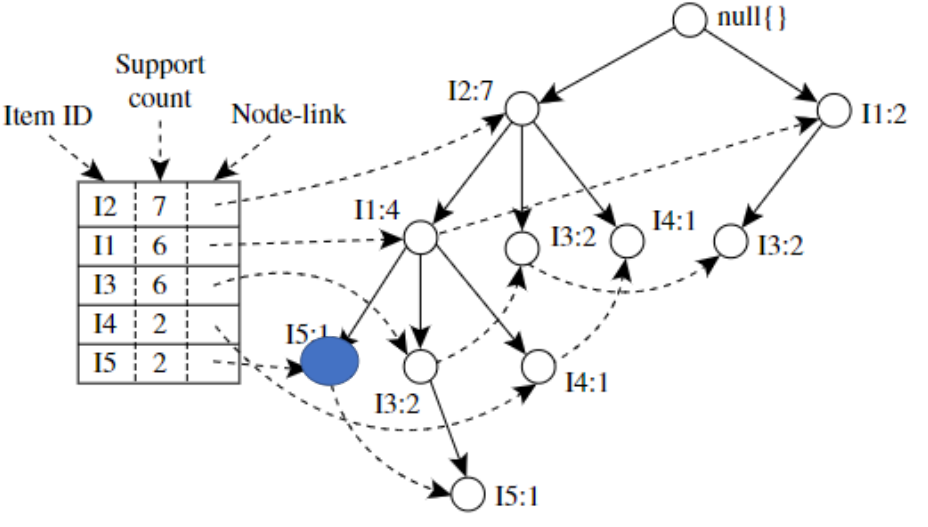


Figure 6.7 An FP-tree registers compressed, frequent pattern information.

Table 6.2 Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

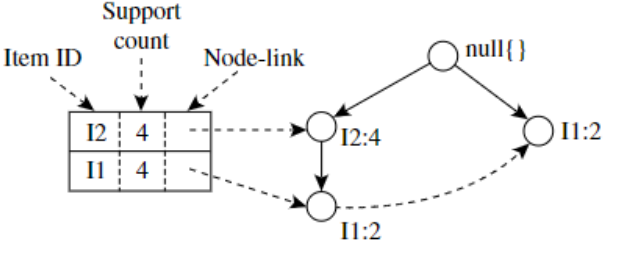


Figure 6.8 The conditional FP-tree associated with the conditional node I3.