— Module 5 —

Frequent Pattern Mining

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalogue design, Add on sales, store layout,Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns and Association Rules

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*.
- Transaction *t* :
  - *t* a set of items, and $t \subseteq I$.
- Transaction Database *T*: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$.
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Basic Concepts: Frequent Patterns and Association Rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$.
- An association rule is an implication of the form:

  $X \rightarrow Y$, where $X, Y \subset I$, *and* $X \cap Y = \emptyset$
- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset.

Support and Confidence

- Support count: The support count of an itemset *X*, denoted by *X.count or Support_count(X)*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *D* has *n* transactions.
- Then,

Support & Confidence

- Support- Usefulness
- Confidence - Certainty of discovered rules

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B|A).$$

$$support = \frac{(A \cup B).count}{n}$$

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

- In general, association rule mining is a two-step process:

  1. Find all frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min sup*.

  2. Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

Association Rules Ex (cont'd)

Itemset
- A collection of one or more items
  - Example: {Milk, Bread, Biscuit}
- k-itemset
  - An itemset that contains k items

Support count
- Frequency of occurrence of an itemset
- E.g. ({Milk, Bread,Biscuit}) = 2

Support
- Fraction of transactions that contain an itemset
- E.g. s({Milk, Bread, Biscuit}) = 2/5

Frequent Itemset
- An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Biscuit, FruitJuice, Eggs
3	Milk, Biscuit, FruitJuice, Coke
4	Bread, Milk, Biscuit, FruitJuice
5	Bread, Milk, Biscuit, Coke

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- **Brute-force approach**:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Association Rules

- An association rule is an implication of the form:
  $X \rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \emptyset$
- {Milk, Biscuit} ⟶ {FruitJuice}

TID	*Items*
1	**Bread, Milk**
2	**Bread, Biscuit, FruitJuice, Eggs**
3	**Milk, Biscuit, FruitJuice, Coke**
4	**Bread, Milk, Biscuit, FruitJuice**
5	**Bread, Milk, Biscuit, Coke**

$$\{\text{Milk, Biscuit}\} \Rightarrow \text{FruitJuice}$$

$$s = \frac{\sigma(\text{Milk, Biscuit, FruitJuice})}{|T|} = \frac{2}{5} = 0.4$$

- {Milk,Biscuit} ⟶ {FruitJuice} (s

$$c = \frac{\sigma(\text{Milk, Biscuit, FruitJuice})}{\sigma(\text{Milk, Biscuit})} = \frac{2}{3} = 0.67$$

Association Rules

Example of Rules:

{Milk,Biscuit} → {FruitJuice} (s=0.4, c=0.67)

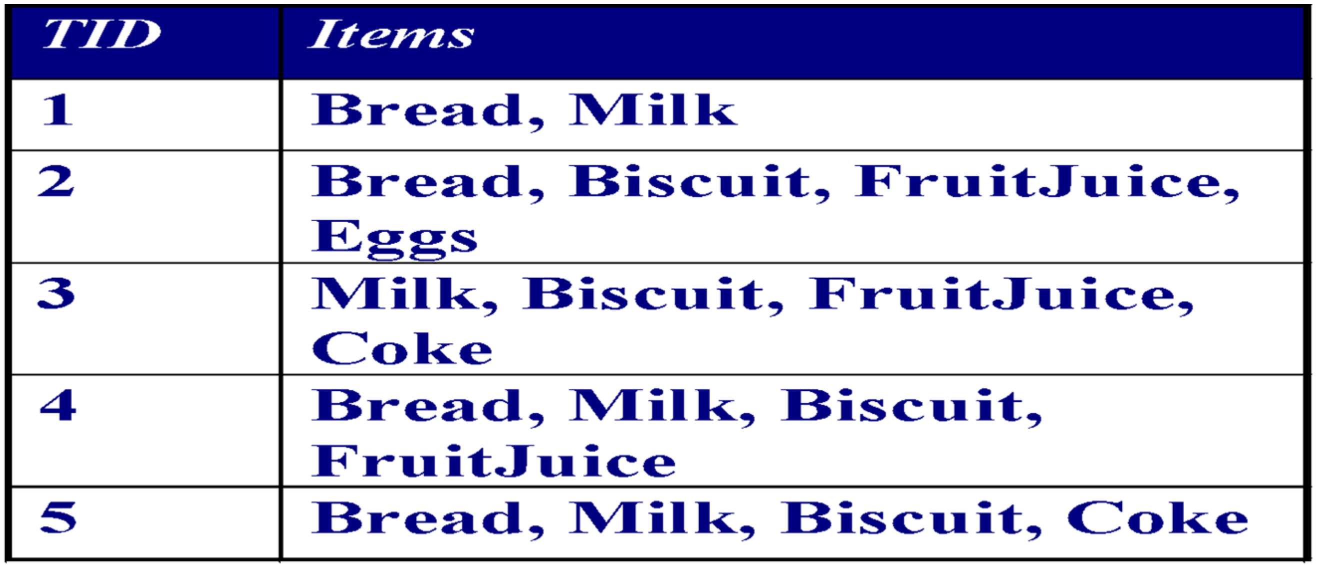
{Milk,FruitJuice} → {Biscuit} (s=0.4, c=1.0)

{Biscuit,FruitJuice} → {Milk} (s=0.4, c=0.67)

{FruitJuice} → {Milk,Biscuit} (s=0.4, c=0.67)

{Biscuit} → {Milk,FruitJuice} (s=0.4, c=0.5)

{Milk} → {Biscuit,FruitJuice} (s=0.4, c=0.5)



TID	*Items*
1	Bread, Milk
2	Bread, Biscuit, FruitJuice, Eggs
3	Milk, Biscuit, FruitJuice, Coke
4	Bread, Milk, Biscuit, FruitJuice
5	Bread, Milk, Biscuit, Coke

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally **expensive**

Apriori

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:
  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Idea of the Apriori Algorithm

- Start with all 1-itemsets
- Go through data and count their support and find all “large” 1-itemsets
- Combine them to form “candidate” 2-itemsets
- Go through data and count their support and find all “large” 2-itemsets
- Combine them to form “candidate” 3-itemsets

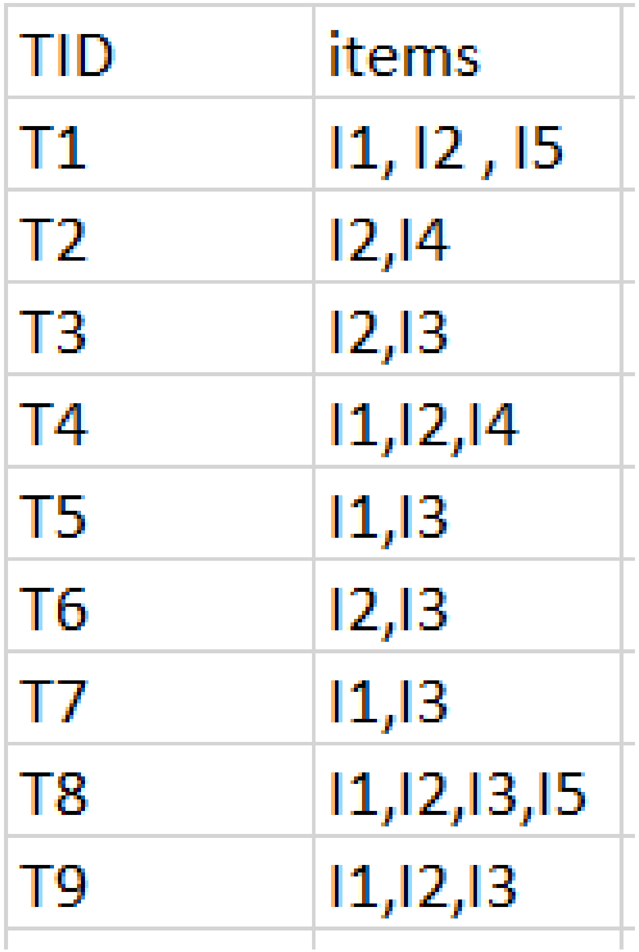
Apriori property:
All subsets of a frequent itemset must be frequent
If a itemset is infrequent all its supersets will be infrequent.

Exercise 1

TID	List of Item 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

- Min sup count =2,
- Min Conf =50%

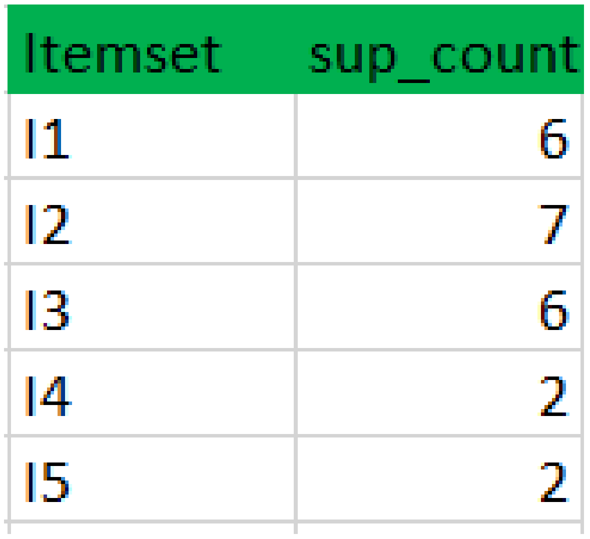
Solution



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

- **Step-1:** K=1
  Create a table containing support count of each item present in dataset – Called **C1(candidate set)**
-

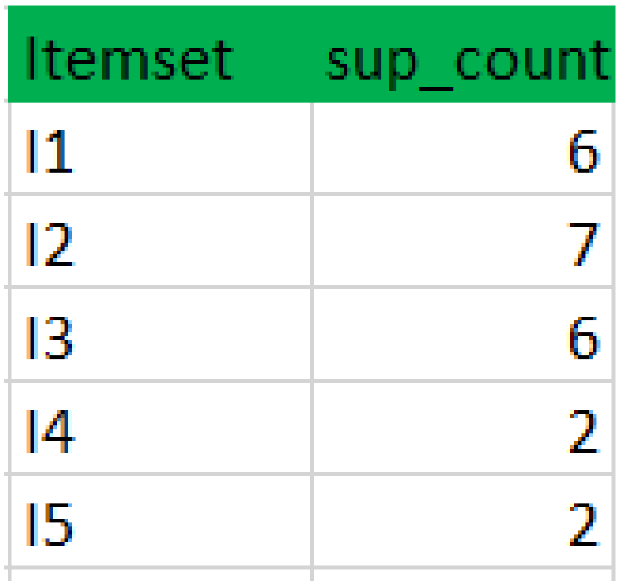
C1



Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

- Compare candidate set item’s support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items) this gives us itemset L1.

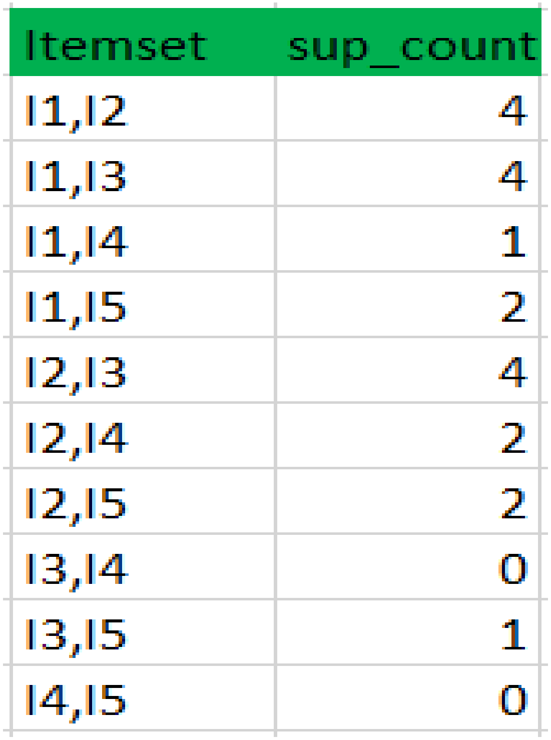
L1



Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

- **Step-2: K=2**
- Generate candidate set C2 using L1 (this is called join step).
- Check all subsets of a itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

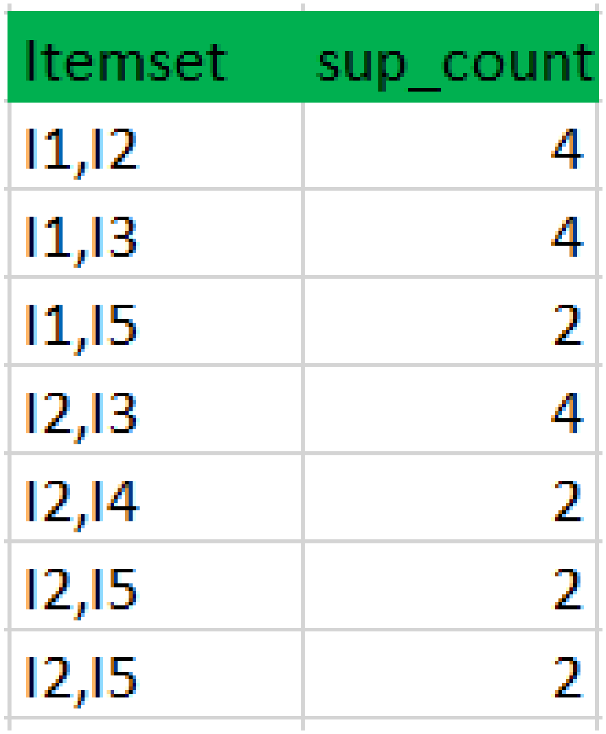
C2



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

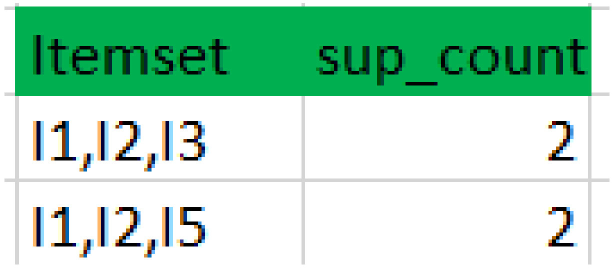
- compare candidate (C2) support count with minimum support count (here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

L2



Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

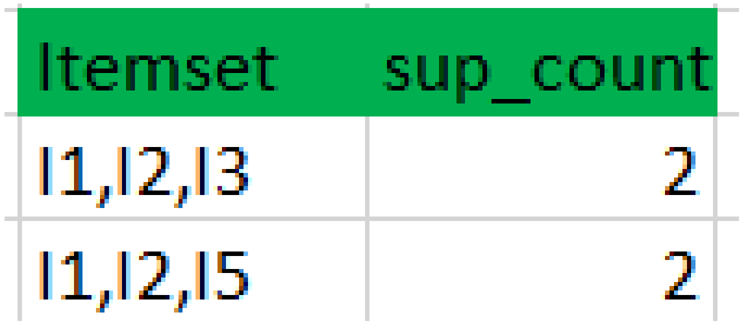
- **Step-3:**
  - –Generate candidate set C3 using L2 (join step)
  - –So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, I5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
  - –Check all subsets of these itemsets are frequent or not and if not remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2}{I2, I3}{I1, I3} which are frequent. For {I2, I3, I4} subset {I3, I4} is not frequent so remove this. Similarly check for every itemset)
  - –find support count of these remaining itemset by searching in dataset.



Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

- Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

L3



Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

- **Step-4:**
  - Generate candidate set C4 using L3 (join step).
  - Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contain {I1, I3, I5} which is not frequent). itemset in C4
  - We stop here because no frequent itemset are found frequent further

Thus we discovered all frequent item-sets now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

Confidence –

A confidence of 50% means that 50% of the customers who purchased a milk and bread also bought the butter.

Confidence(A->B)=Support_count(A∪B)/Support_count(A)

So here By taking example of any frequent itemset we will show rule generation.

Itemset {I1, I2, I3} //from L3 SO rules can be

Itemset {I1, I2, I3}   //from L3 SO rules can be
[I1^I2]=>[I3]
confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%
[I1^I3]=>[I2]
confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%
[I2^I3]=>[I1]
confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%
[I1]=>[I2^I3]
confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%
[I2]=>[I1^I3]
confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%
[I3]=>[I1^I2]
confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%
So if minimum confidence is 50 % first 3 rules can be considered strong association rules.

$\xrightarrow{\text{Scan } D \text{ for count of each candidate}}$

C_1

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

$\xrightarrow{\text{Compare candidate support count with minimum support count}}$

L_1

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

$\xrightarrow{\text{Generate } C_2 \text{ candidates from } L_1}$

C_2

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

$\xrightarrow{\text{Scan } D \text{ for count of each candidate}}$

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

$\xrightarrow{\text{Compare candidate support count with minimum support count}}$

L_2

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

$\xrightarrow{\text{Generate } C_3 \text{ candidates from } L_2}$

C_3

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

$\xrightarrow{\text{Scan } D \text{ for count of each candidate}}$

C_3

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

$\xrightarrow{\text{Compare candidate support count with minimum support count}}$

L_3

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

- *C*3 ={{I1, I2, I3}, {I1, I2, I5}} after pruning.
- 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:
- 50%, then only ,first, second, third, and last rules are output, because these are the only ones generated that are strong.

$I1 \wedge I2 \Rightarrow I5,$	$confidence = 2/4 = 50\%$
$I1 \wedge I5 \Rightarrow I2,$	$confidence = 2/2 = 100\%$
$I2 \wedge I5 \Rightarrow I1,$	$confidence = 2/2 = 100\%$
$I1 \Rightarrow I2 \wedge I5,$	$confidence = 2/6 = 33\%$
$I2 \Rightarrow I1 \wedge I5,$	$confidence = 2/7 = 29\%$
$I5 \Rightarrow I1 \wedge I2,$	$confidence = 2/2 = 100\%$

The Apriori Algorithm—An Example

Sup_{min} = 50%
min_confidence is 60%

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

The Apriori Algorithm—An Example

Support Count = 2

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan

C_1

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

L_1

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

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Association rule generation

- Generating association rules from frequent itemset
  - For each frequent itemset l, generate all nonempty subsets of l
  - For every nonempty subset s of l, output the rule s=> (l-s) if support(l)/support(s)>= min confidence

B^ C=>E conf =2/2 =100%

C^ E=>B conf =2/2 =100%

B^E=>C conf =2/3 =66.67%

B=>C ^E conf =2/3 =66.67%

C=>B ^E conf =2/3 =66.67%

E=>B ^C conf =2/3 =66.67%

The Apriori Algorithm

- Pseudo-code:

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {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 $\cup_k L_k$;
```

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: $L_3 * L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

```
insert into $C_k$
select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}$ $p$, $L_{k-1}$ $q$
where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
```

- Step 2: pruning

```
forall itemsets $c$ in $C_k$ do
        forall $(k-1)$-subsets $s$ of $c$ do
                if ($s$ is not in $L_{k-1}$) then delete $c$ from $C_k$
```

Exercise 2

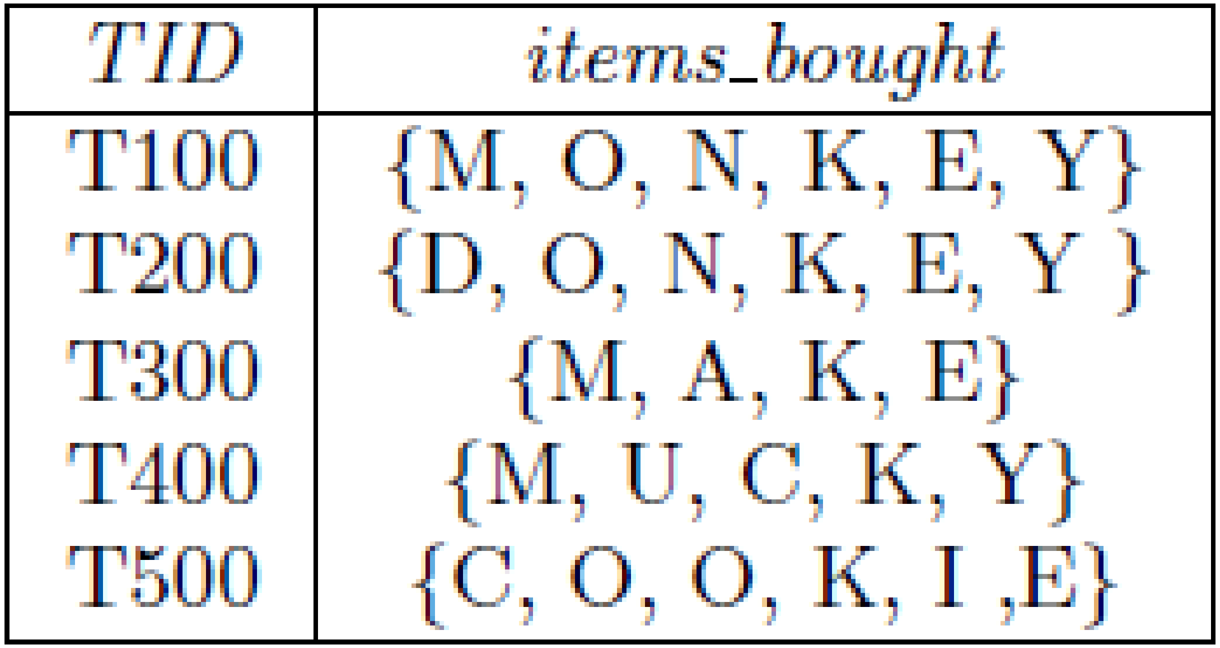
- A database has have 5 transactions. Let min sup = 60% and min conf = 80%.

(a) Find all frequent itemsets using Apriori

(b) List all of the strong association rules (with support s and confidence c)

Exercise 2

$$\forall x \in transaction,\ buys(X, item_1) \land buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$



TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

Ans 2

C1 =

m	3
o	3
n	2
k	5
e	4
y	3
d	1
a	1
u	1
c	2
i	1

L1 =

m	3
o	3
k	5
e	4
y	3

C2 =

mo	1
mk	3
me	2
my	2
ok	3
oe	3
oy	2
ke	4
ky	3
ey	2

L2 =

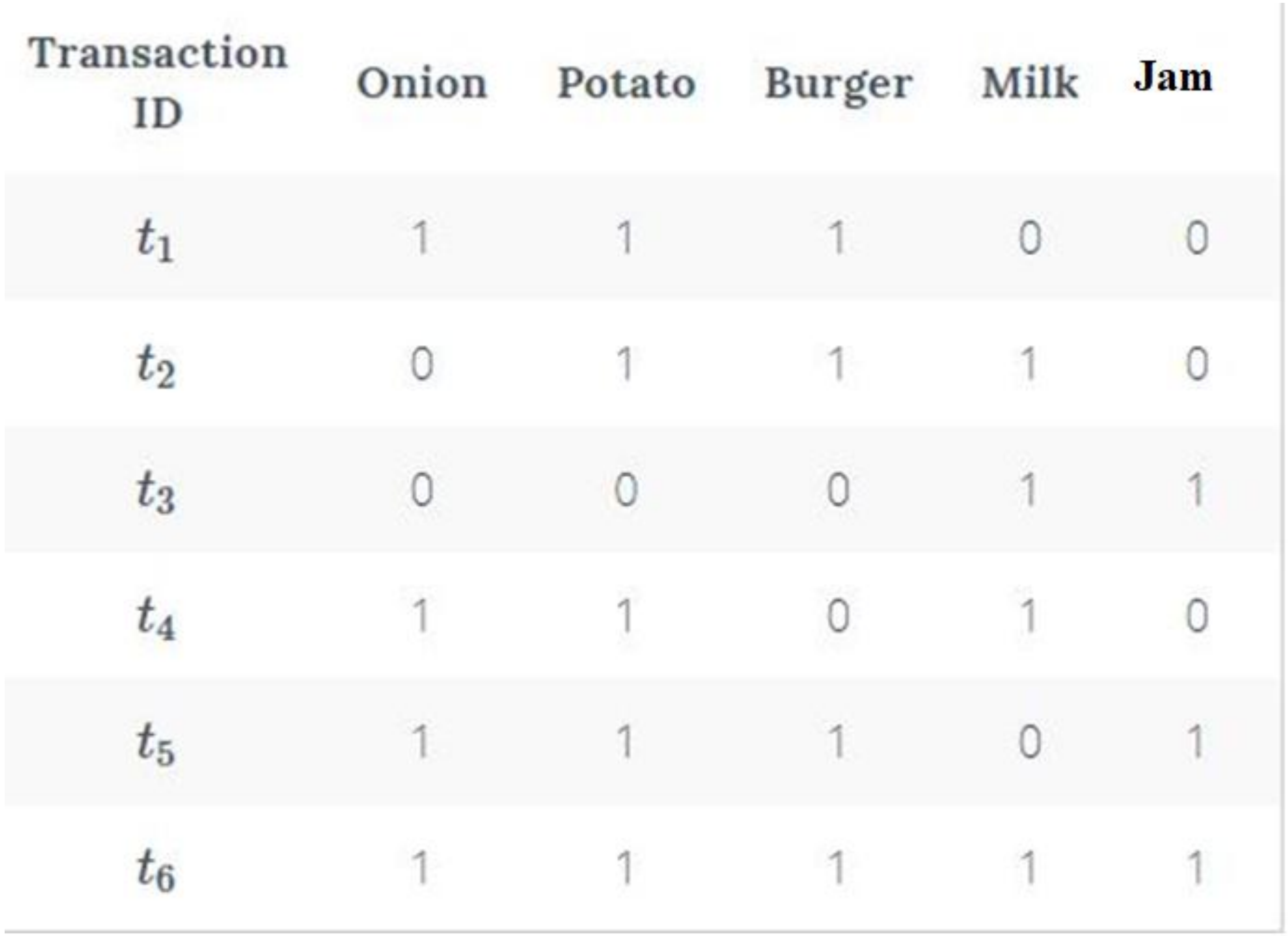
mk	3
ok	3
oe	3
ke	4
ky	3

C3=

oke	3
key	2

L3=

oke	3



Transaction ID	Onion	Potato	Burger	Milk	Jam
t_1	1	1	1	0	0
t_2	0	1	1	1	0
t_3	0	0	0	1	1
t_4	1	1	0	1	0
t_5	1	1	1	0	1
t_6	1	1	1	1	1

- Min support count=4
- Min Conf= 70%

Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $({}_{100}{}^{1}) + ({}_{100}{}^{2}) + \ldots + ({}_{1}{}^{1}{}_{0}{}^{0}{}_{0}{}^{0}) = 2^{100}-1 =$ $1.27*10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Improving the efficiency of Apriori

1. Hashing based technique

The direct hashing and pruning (DHP) algorithm attempts to generate large itemsets efficiently and reduces the transaction database size.

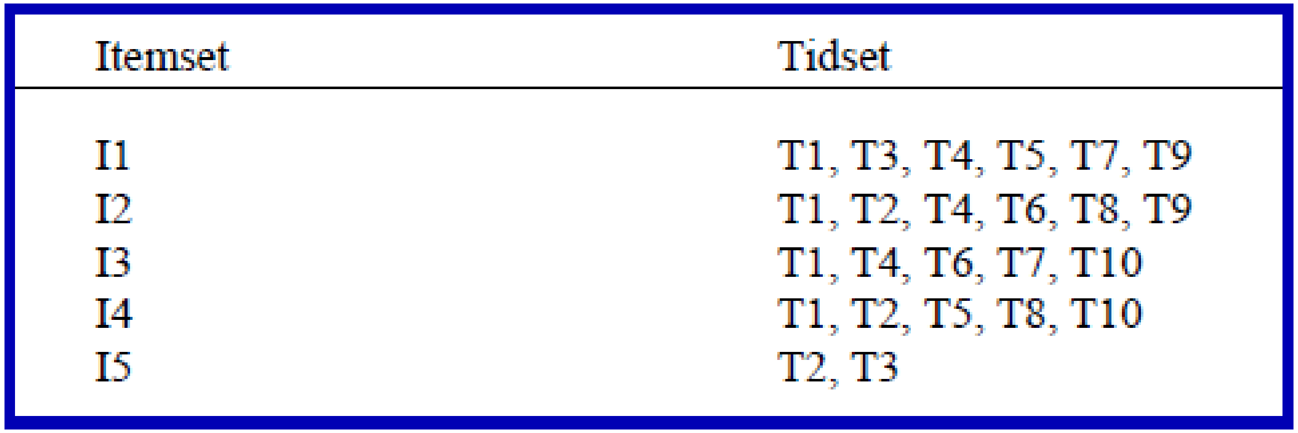
When generating L_1, the algorithm also generates all the 2-itemsets for each transaction, hashes them to a hash table and keeps a count.

1. Hashing based technique

- The items in the transactions are hashed based on the hash function as
- h(k) = (order of item K) mod n.
- The item set in the second level are hashed based on the hash function,
- h(k)=((order of X)*10+order of Y)mod n.
- The itemsets in the next level are hashed based on the hash function,
- h(k) = (((order of X)*100 + (order of Y)*10 + order of Z) mod n).

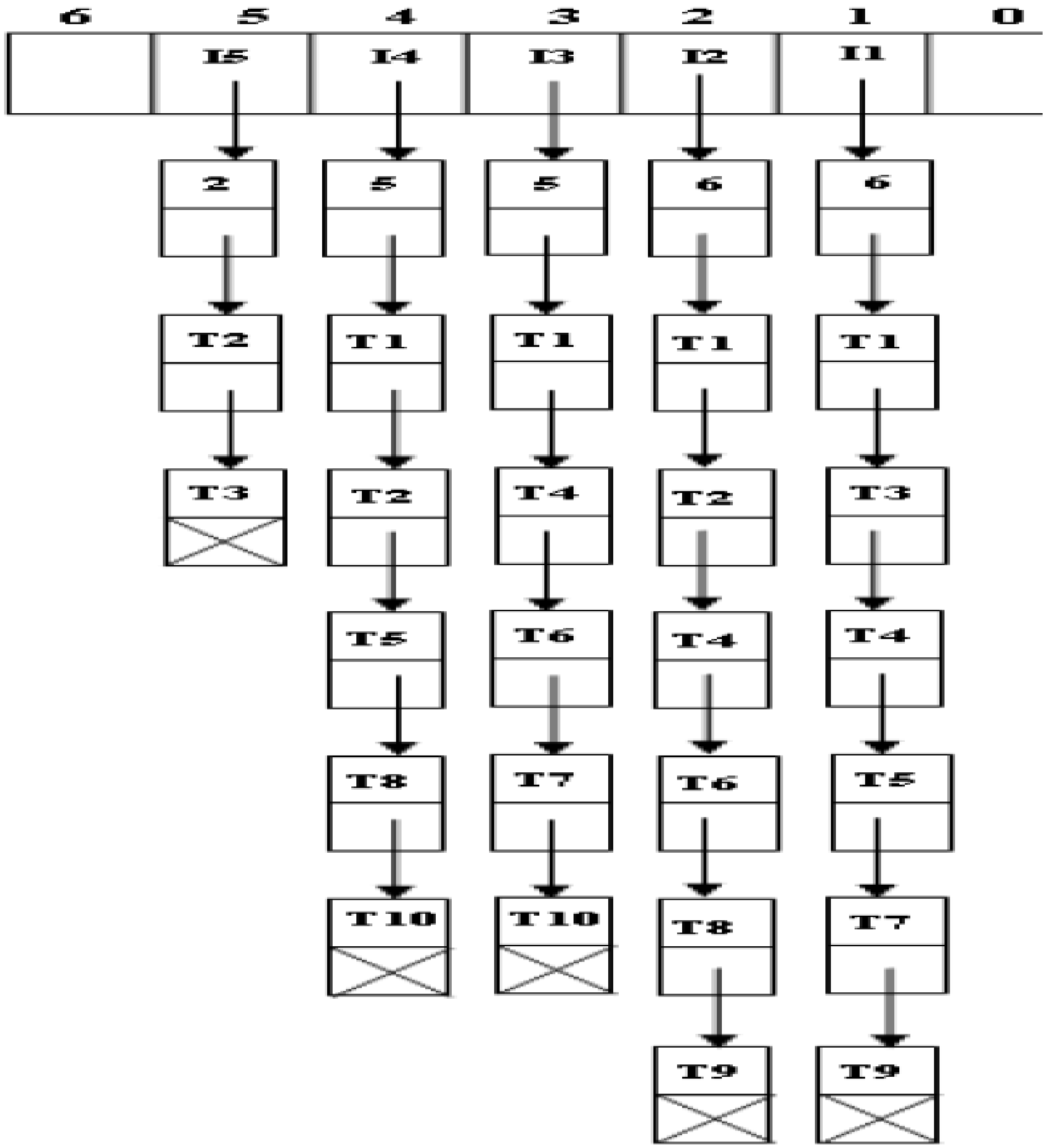
Hashing based technique

- Example
- minimum support count=3



Itemset	Tidset
I1	T1, T3, T4, T5, T7, T9
I2	T1, T2, T4, T6, T8, T9
I3	T1, T4, T6, T7, T10
I4	T1, T2, T5, T8, T10
I5	T2, T3

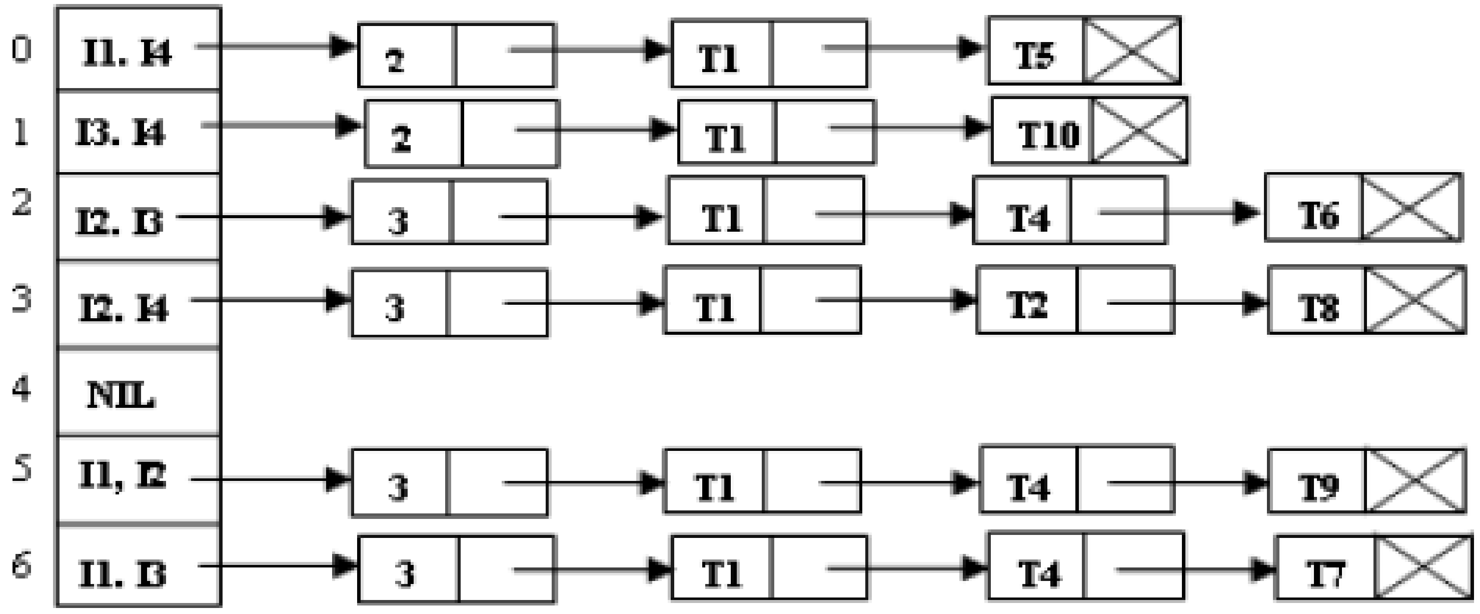
1. Hashing based technique



1. Hashing based technique

- Example

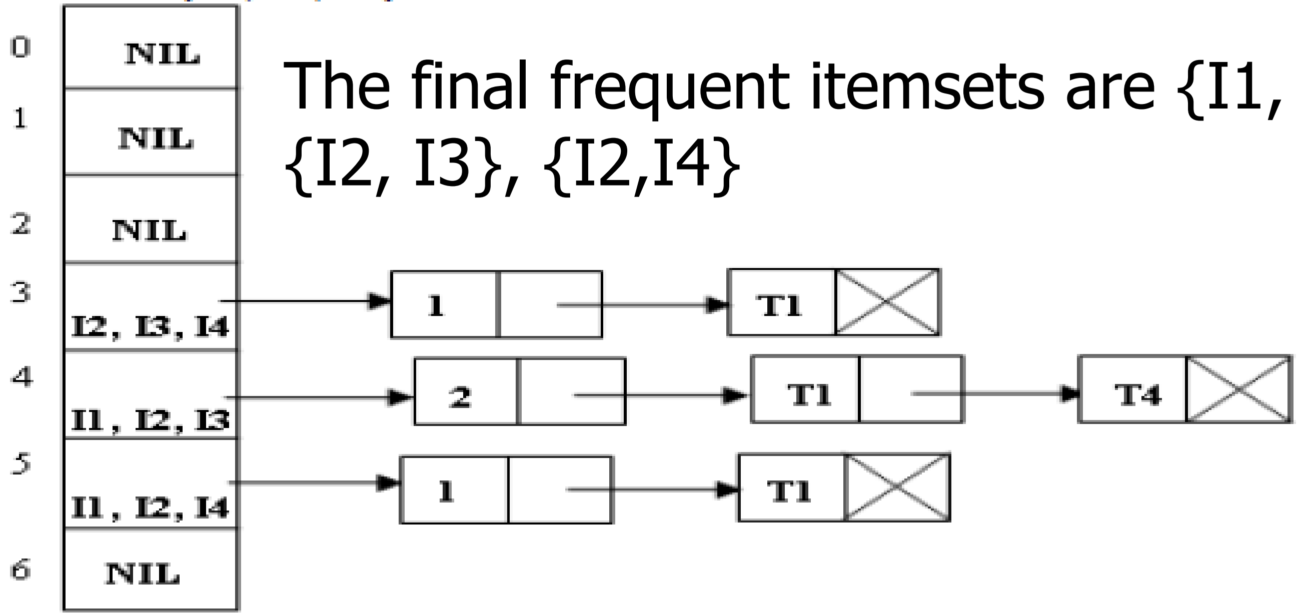
Itemset	Tidset
{I1, I2}	T1, T4, T9
{I1, I3}	T1, T4, T7
{I1, I4}	T1, T5
{I2, I3}	T1, T4, T6
{I2, I4}	T1, T2, T8
{I3, I4}	T1, T10



1. Hashing based technique

- Example

Itemset	Tidset
{I1, I2, I3}	T1, T4
{I1, I2, I4}	T1
{I2, I3, I4}	T1



The final frequent itemsets are {I1, I2},{I1, I3}
{I2, I3}, {I2,I4}

1. Hashing based technique

The major aim of the algorithm is to reduce the size of C_2. It is therefore essential that the hash table is large enough so that collisions are low.

2. 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.

3. Partitioning

The set of transactions may be divided into a number of disjoint subsets. Then each partition is searched for frequent itemsets. These frequent itemsets are called local frequent itemsets.

3. Partitioning

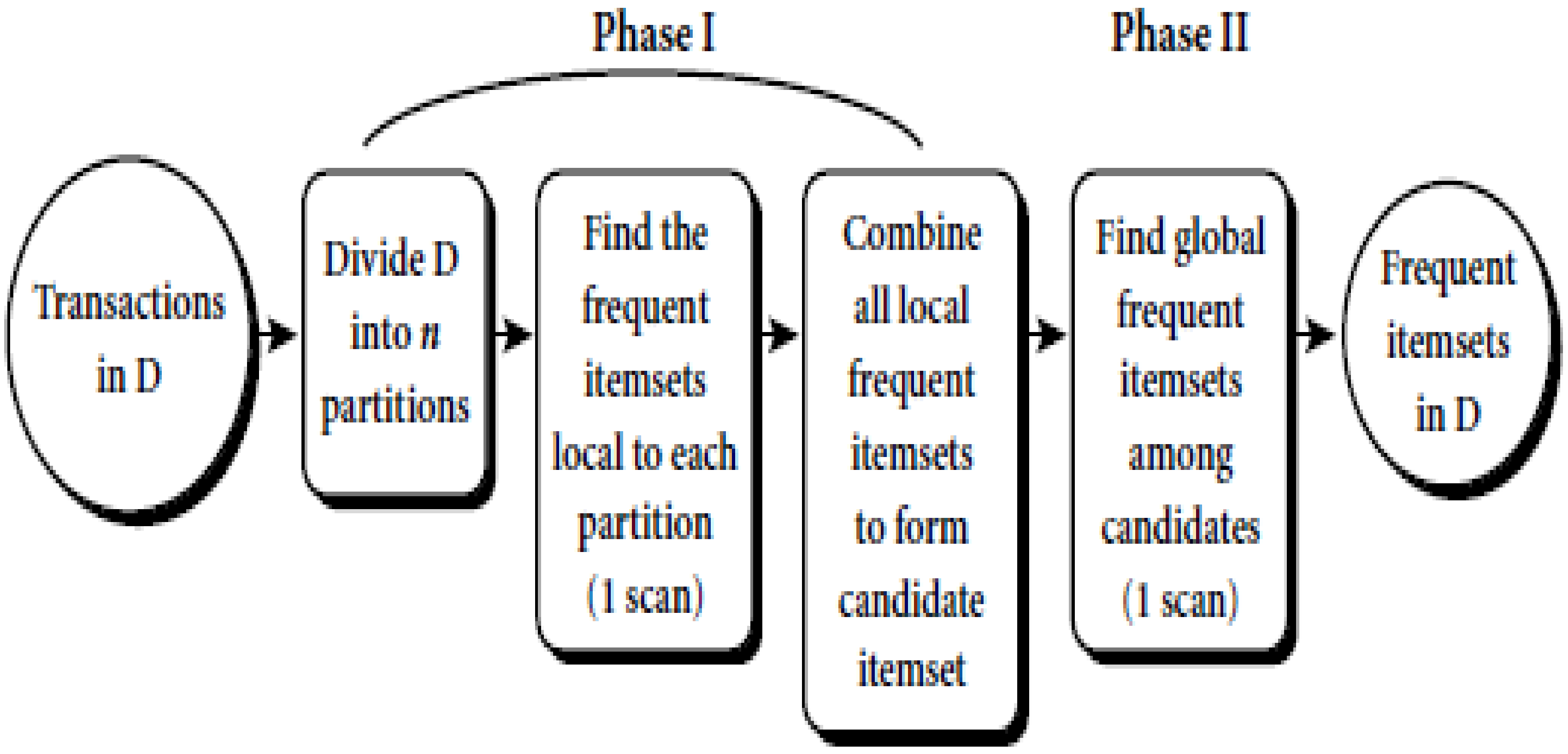
Phase 1

- Divide n transactions into m partitions
- Find the frequent itemsets in each partition
- Combine all local frequent itemsets to form candidate itemsets

Phase 2

Find global frequent itemsets

3. Partitioning


Phase I
Phase II
Transactions in D
Divide D into n partitions
Find the frequent itemsets local to each partition (1 scan)
Combine all local frequent itemsets to form candidate itemset
Find global frequent itemsets among candidates (1 scan)
Frequent itemsets in D

4. 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*.
- The sample size of *S* is such that the search for frequent itemsets in *S* can be done in main memory
- 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.

4. Sampling

- Not guaranteed to be accurate but we sacrifice accuracy for efficiency.
- A lower support threshold may be used for the sample to ensure not missing any frequent datasets.

- The actual frequencies of the sample frequent itemsets are then obtained.

- More than one sample could be used to improve accuracy.

FP Growth

- FP growth improves Apriori to a big extent
- Frequent Item set Mining is possible without candidate generation
- Only “two scan” to the database is needed

BUT HOW?

FP Growth

- Simply a two step procedure
  - Step 1: Build a compact data structure called the FP-tree
    - Build using 2 passes over the data-set.
  - Step 2: Extracts frequent item sets directly from the FP-tree

Construct FP-tree

Two Steps:

1. Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

   e.g., L={f:4, c:4, a:3, b:3, m:3, p:3}

   In the format of (item-name, support)

2. For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.

FP Growth

- Now Lets Consider the following transaction table minimum support count=2,minimum confidence=70%

TID	List of item 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

I1	6
I2	7
I3	6
I4	2
I5	2

First Scan of Database

I2	7
I1	6
I3	6
I4	2
I5	2

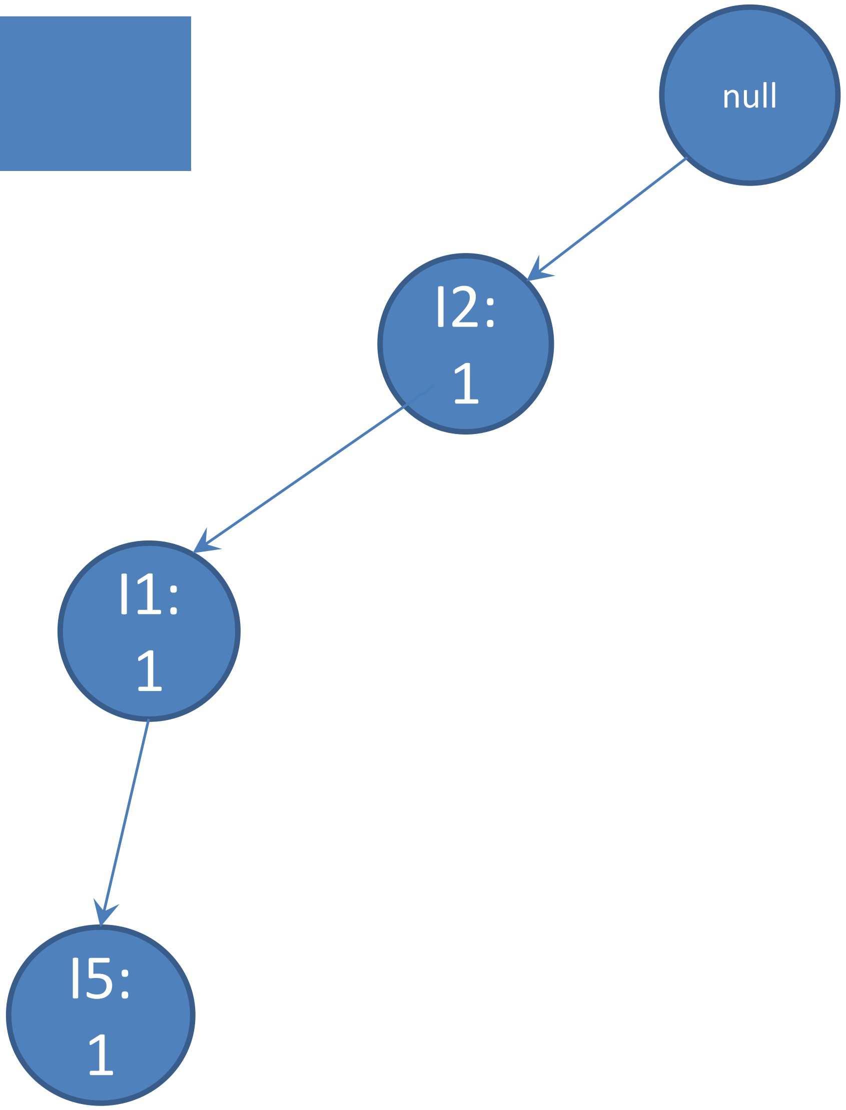
Scan the DB for the second time, order frequent items in each transaction

TID	List of item IDs
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

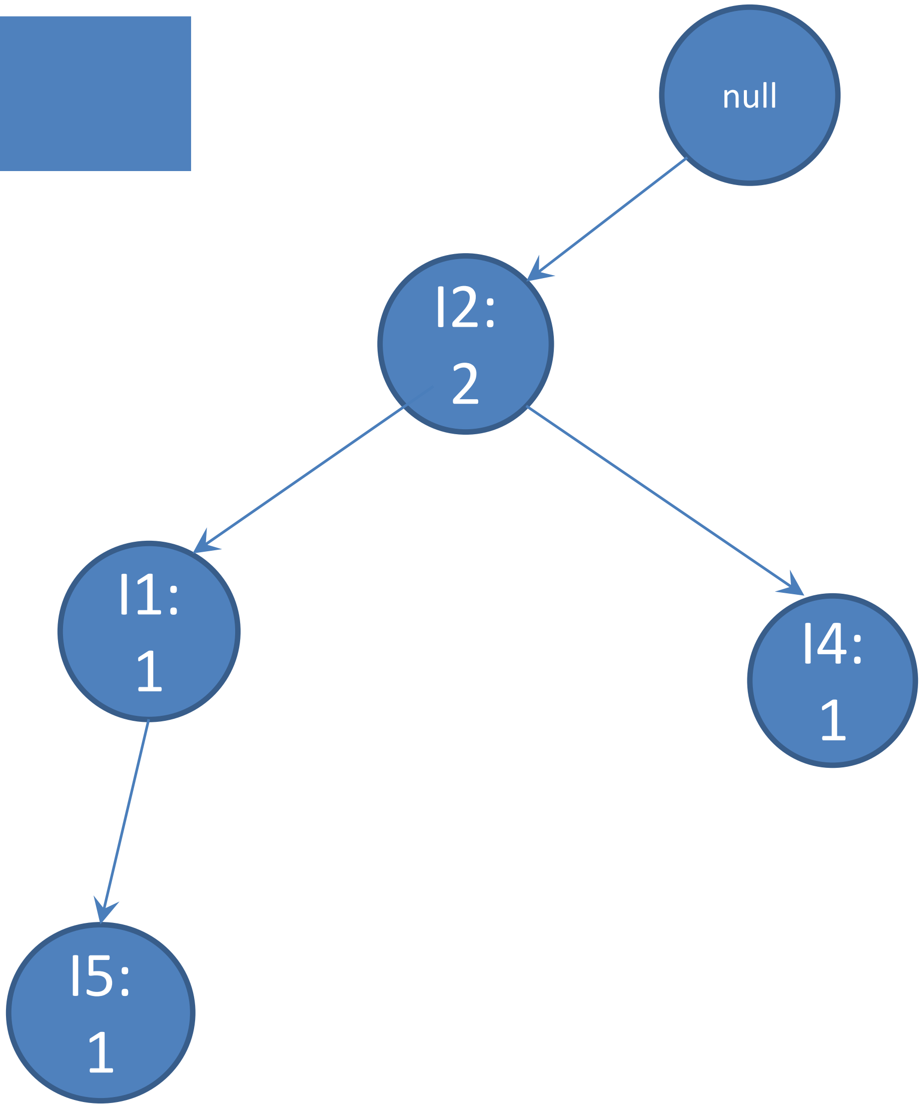
FP Growth

- Now we will build a FP tree of that database
- Item sets are considered in order of their descending value of support count.

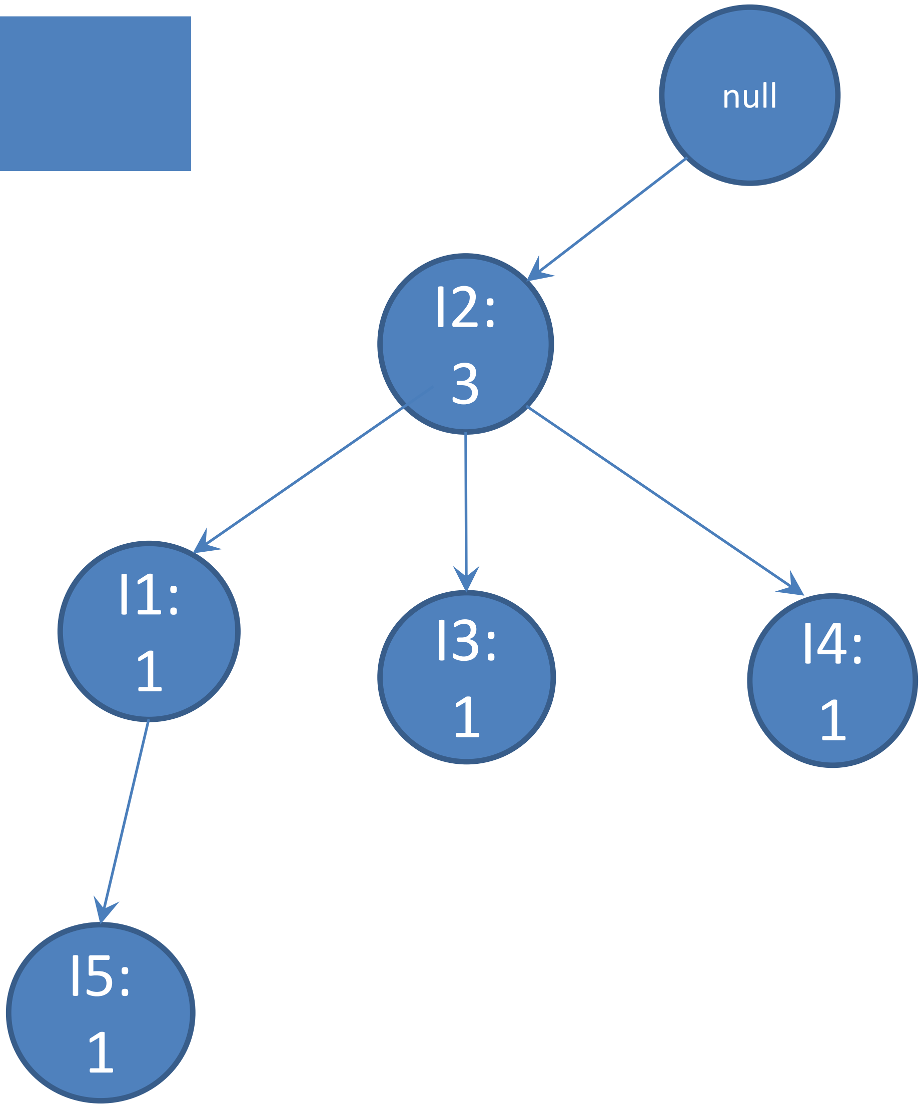
For Transaction:
I2,I1,I5



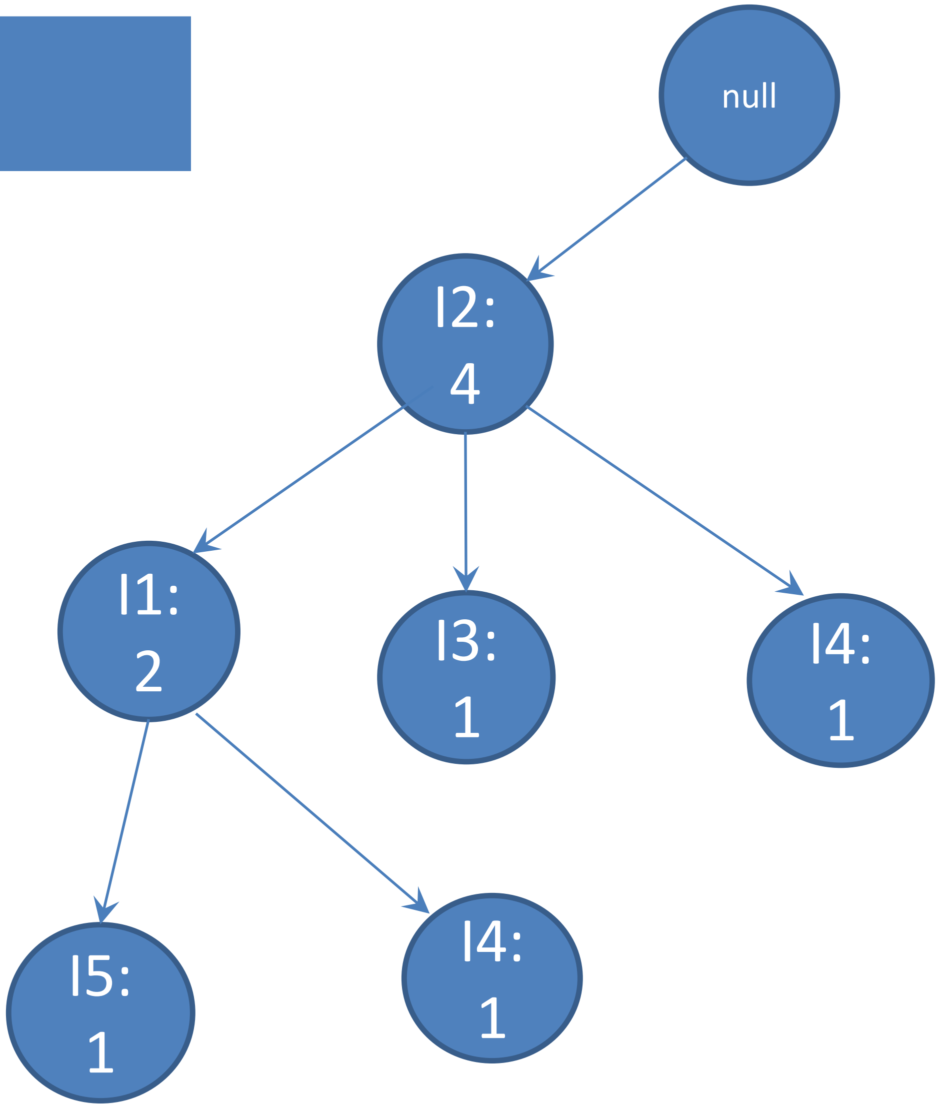
For Transaction:
I2,I4



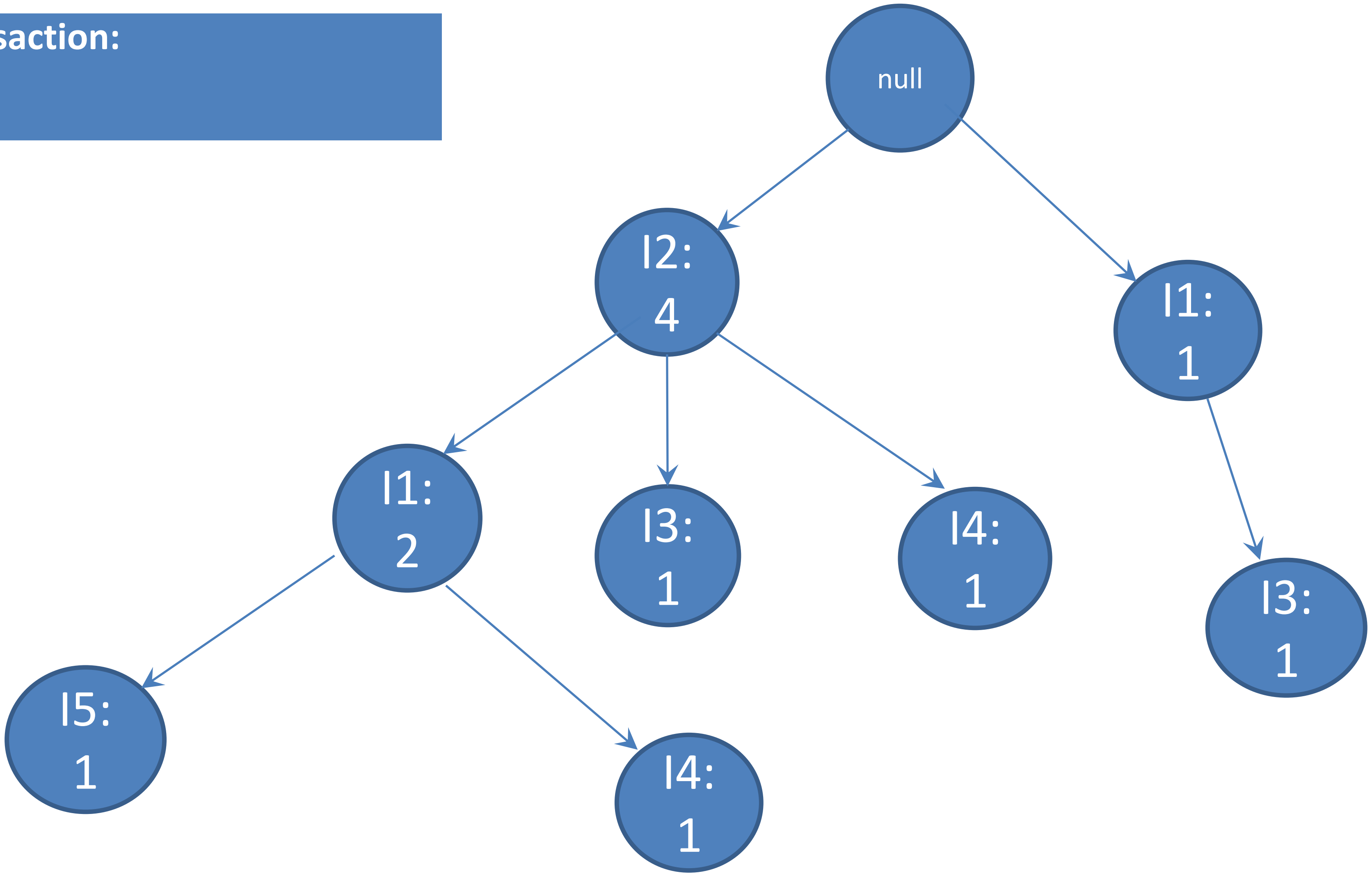
For Transaction:
I2,I3



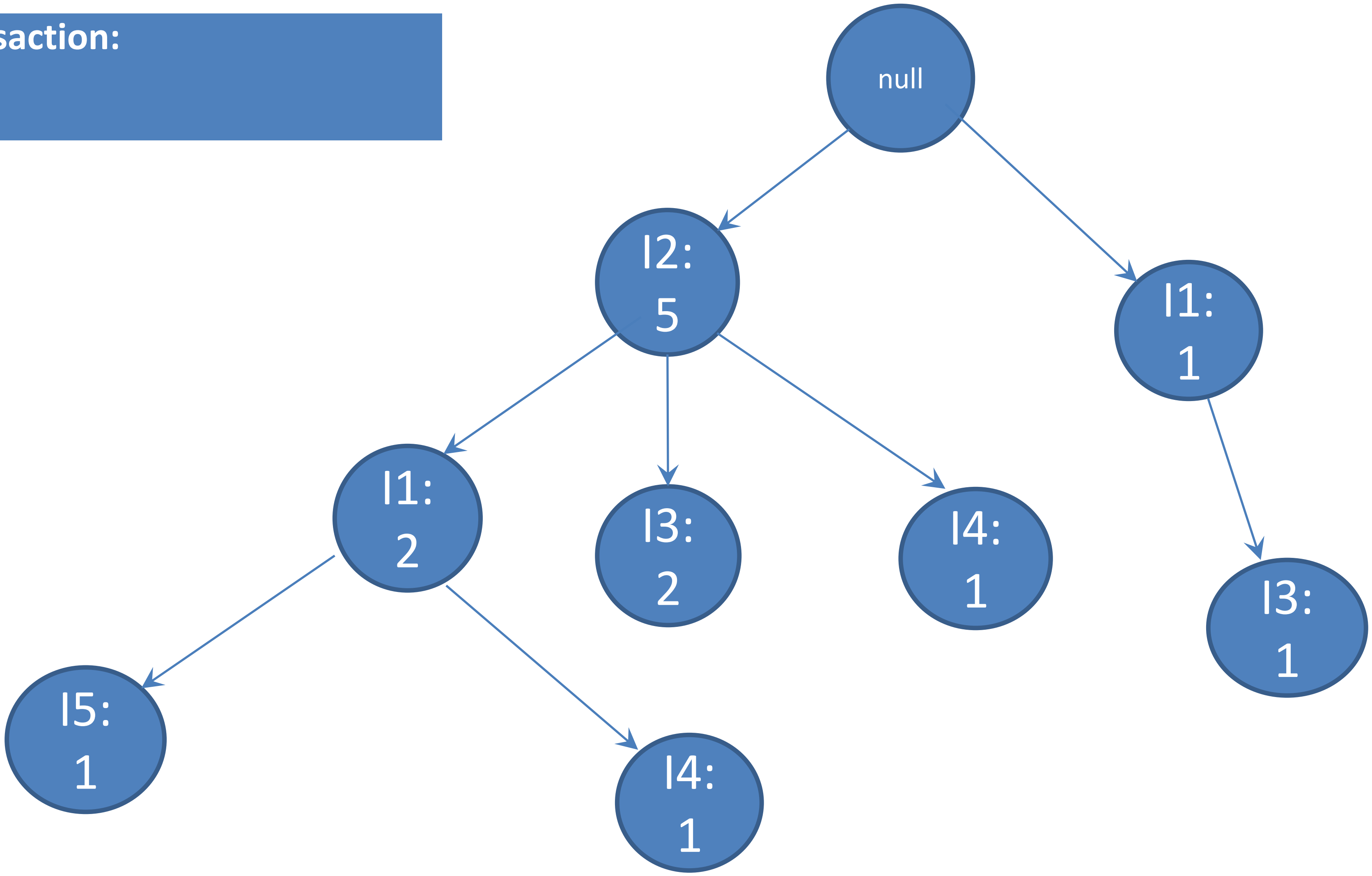
For Transaction:
I2,I1,I4



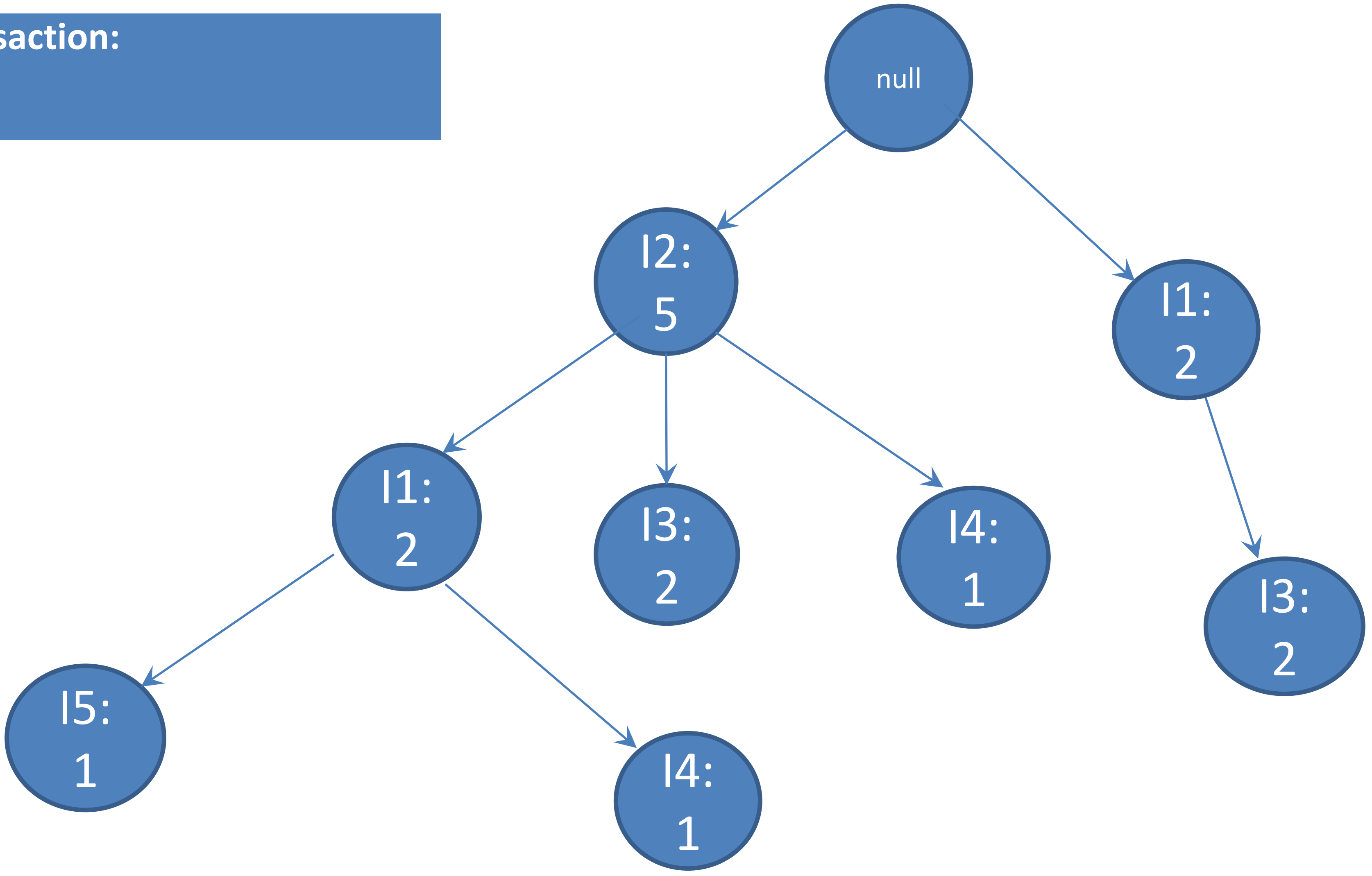
For Transaction:
I1,I3



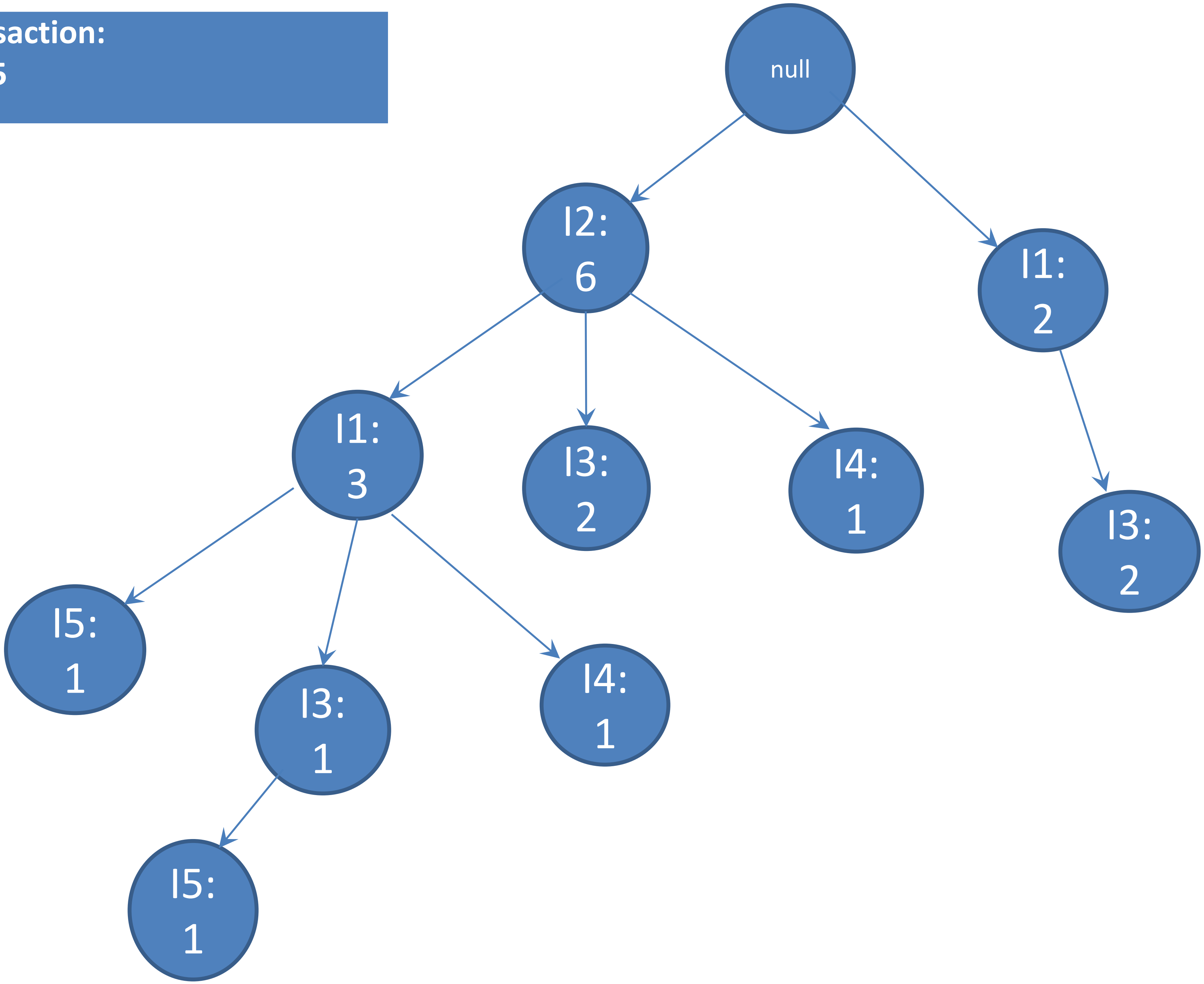
For Transaction:
I2,I3



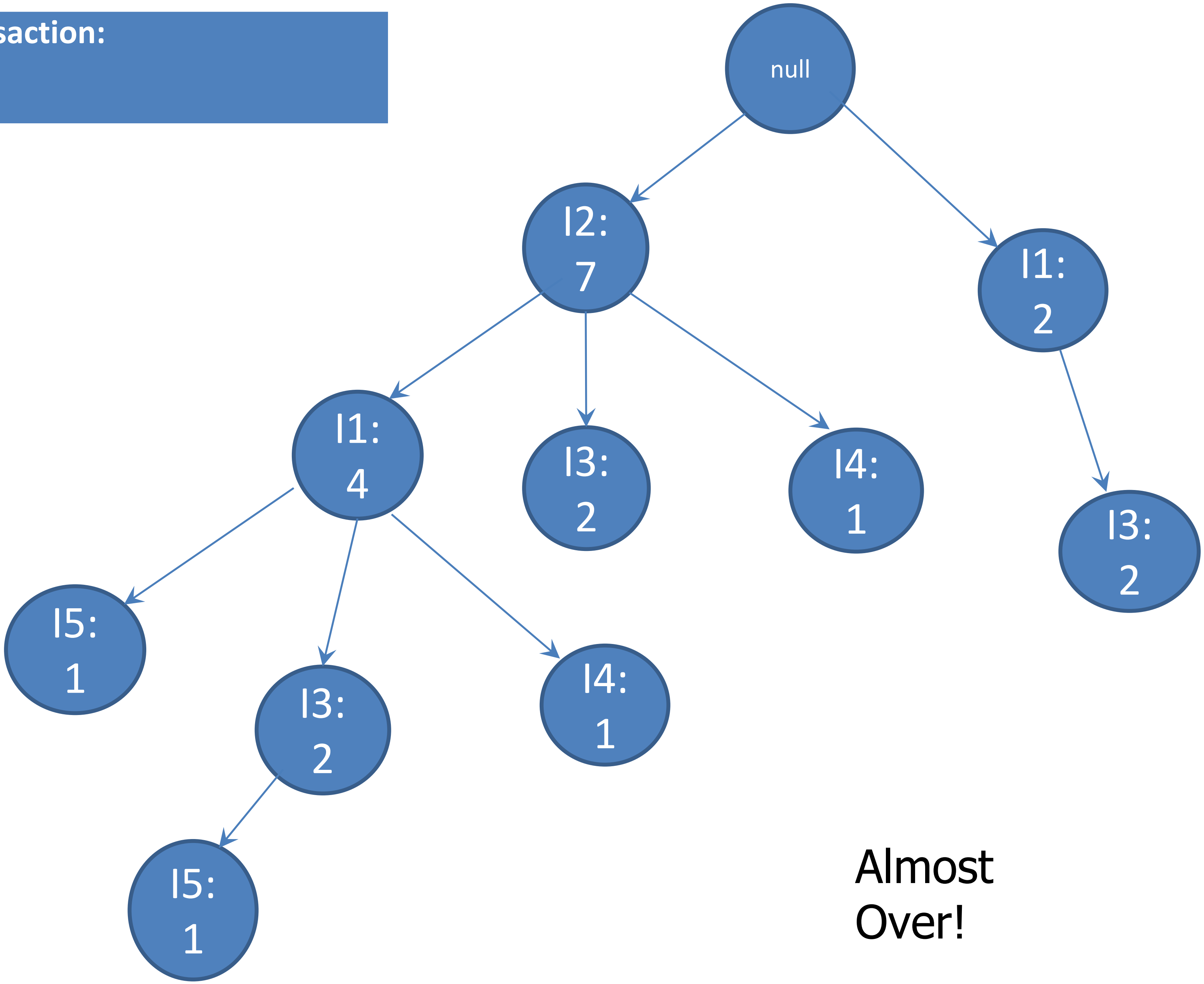
For Transaction:
I1,I3



For Transaction:
I2,I1,I3,I5



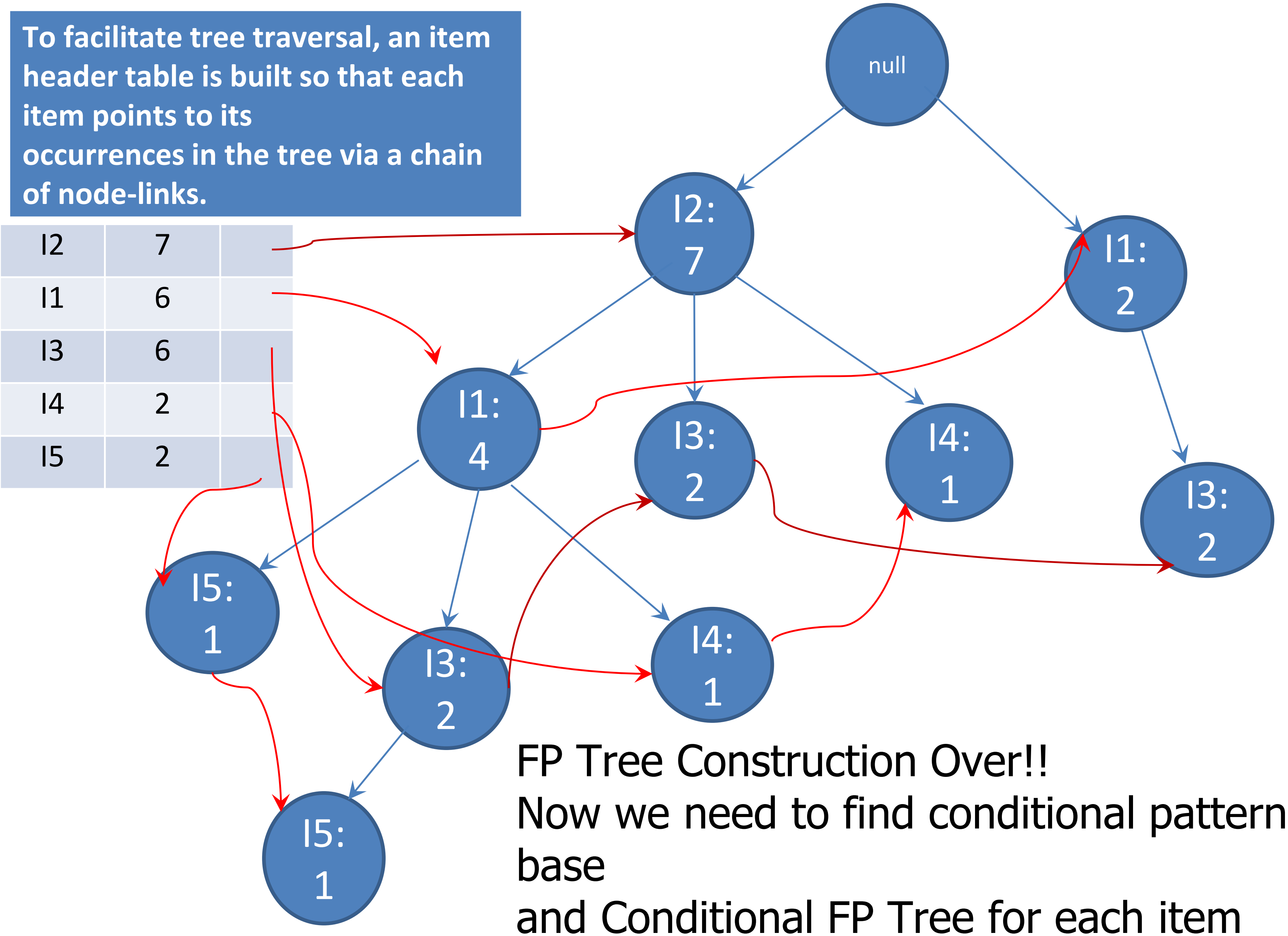
For Transaction:
I2,I1,I3



Almost Over!

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links.

I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



FP Tree Construction Over!!
Now we need to find conditional pattern base
and Conditional FP Tree for each item

FP-Tree Definition

- FP-tree is **a frequent pattern tree** . Formally, FP-tree is a tree structure defined below:

  1. One root labeled as “null", a set of *item prefix sub-trees* as the children of the root, and a *frequent-item header table*.

  2. Each node in *the item prefix sub-trees* has three fields:
     - item-name : register which item this node represents,
     - count, the number of transactions represented by the portion of the path reaching this node,
     - node-link that links to the next node in the FP-tree carrying the same item-name, or null if there is none.

  3. Each entry in the *frequent-item header table* has two fields,
     - item-name, and
     - head of node-link that points to the first node in the FP-tree carrying the item-name.

Construct Conditional Pattern Base

- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**

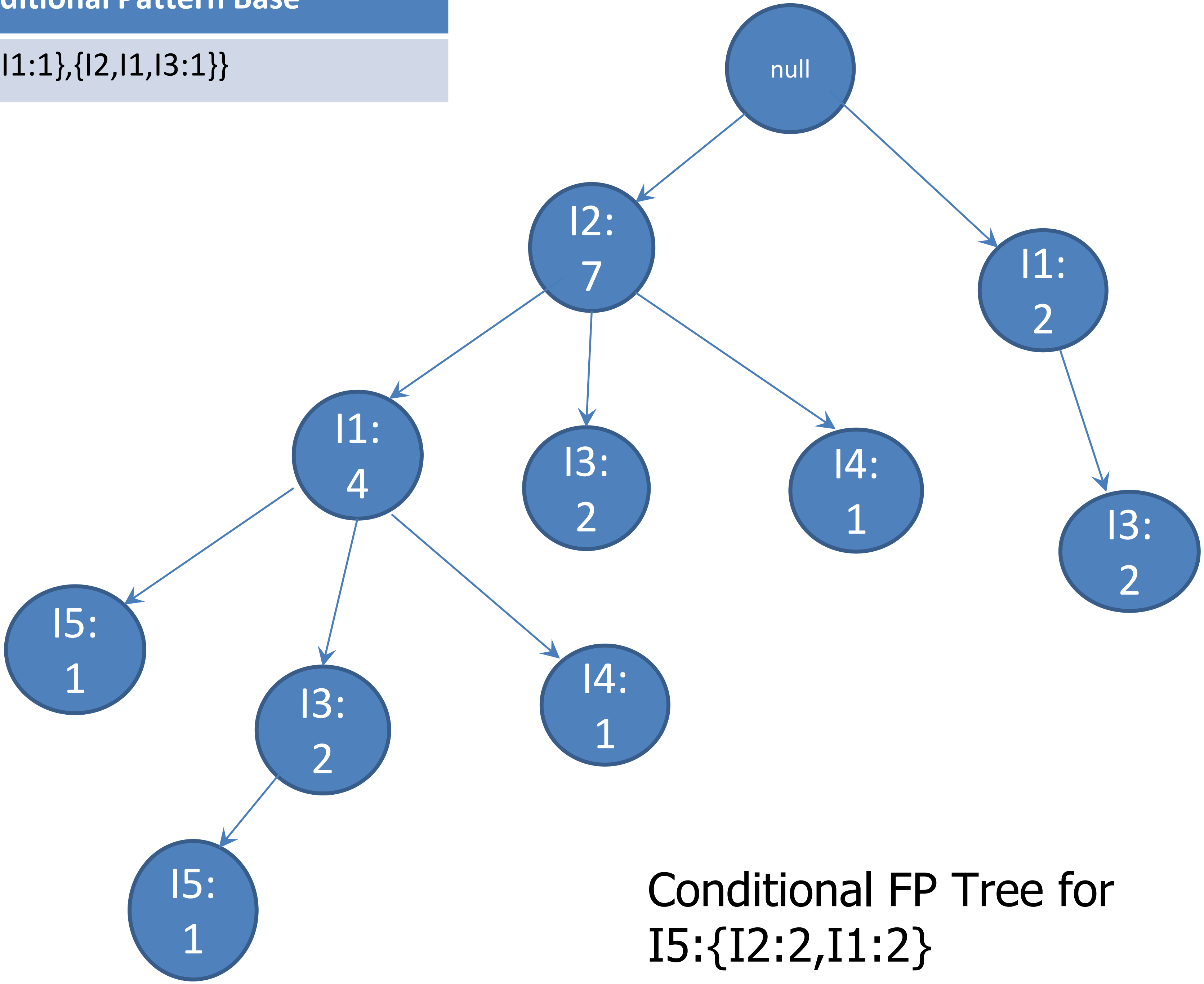
Properties of FP-Tree

- Node-link property
  - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.
- Prefix path property
  - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Construct Conditional FP-tree

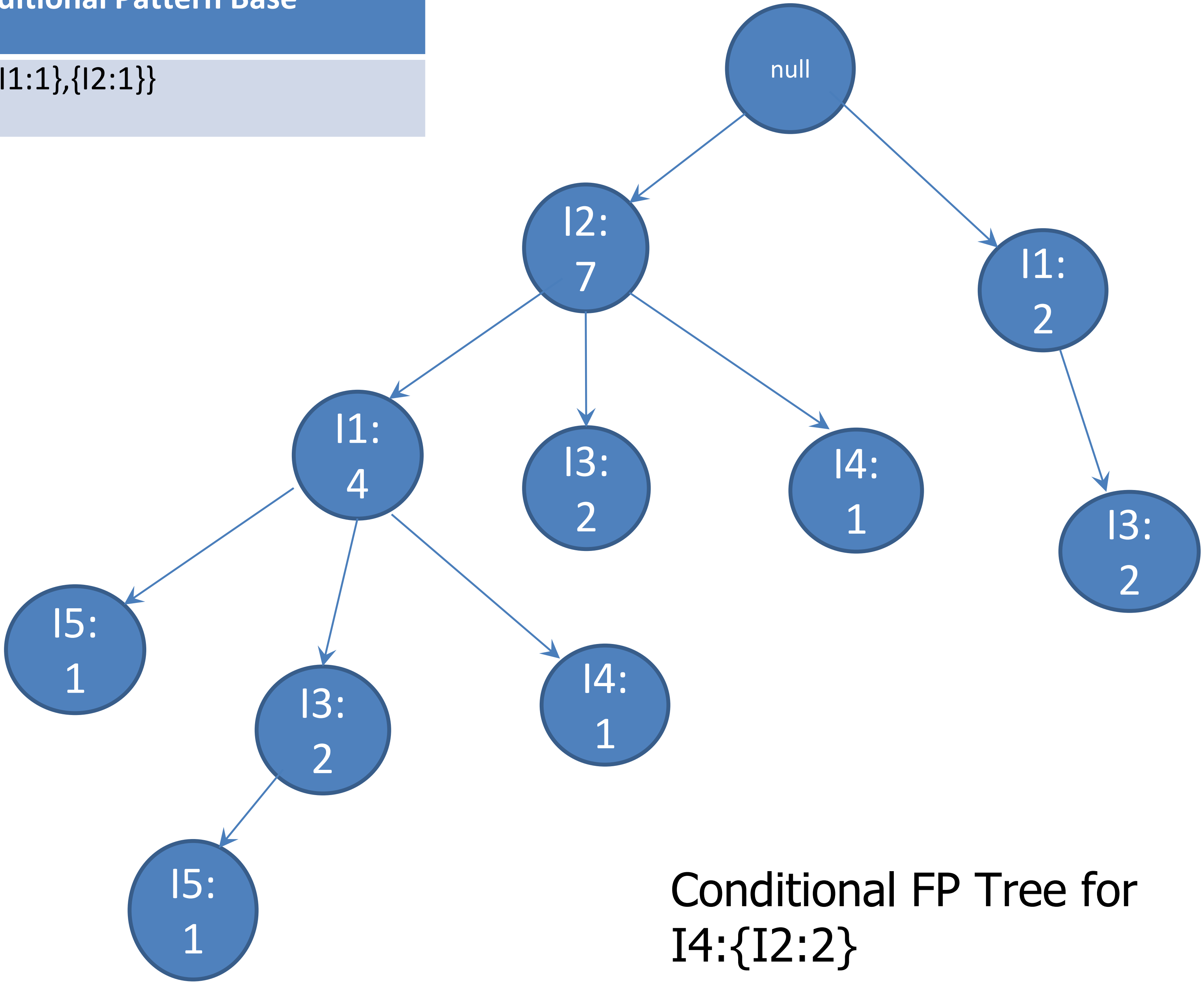
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base

	Conditional Pattern Base
I5	{{I2,I1:1},{I2,I1,I3:1}}



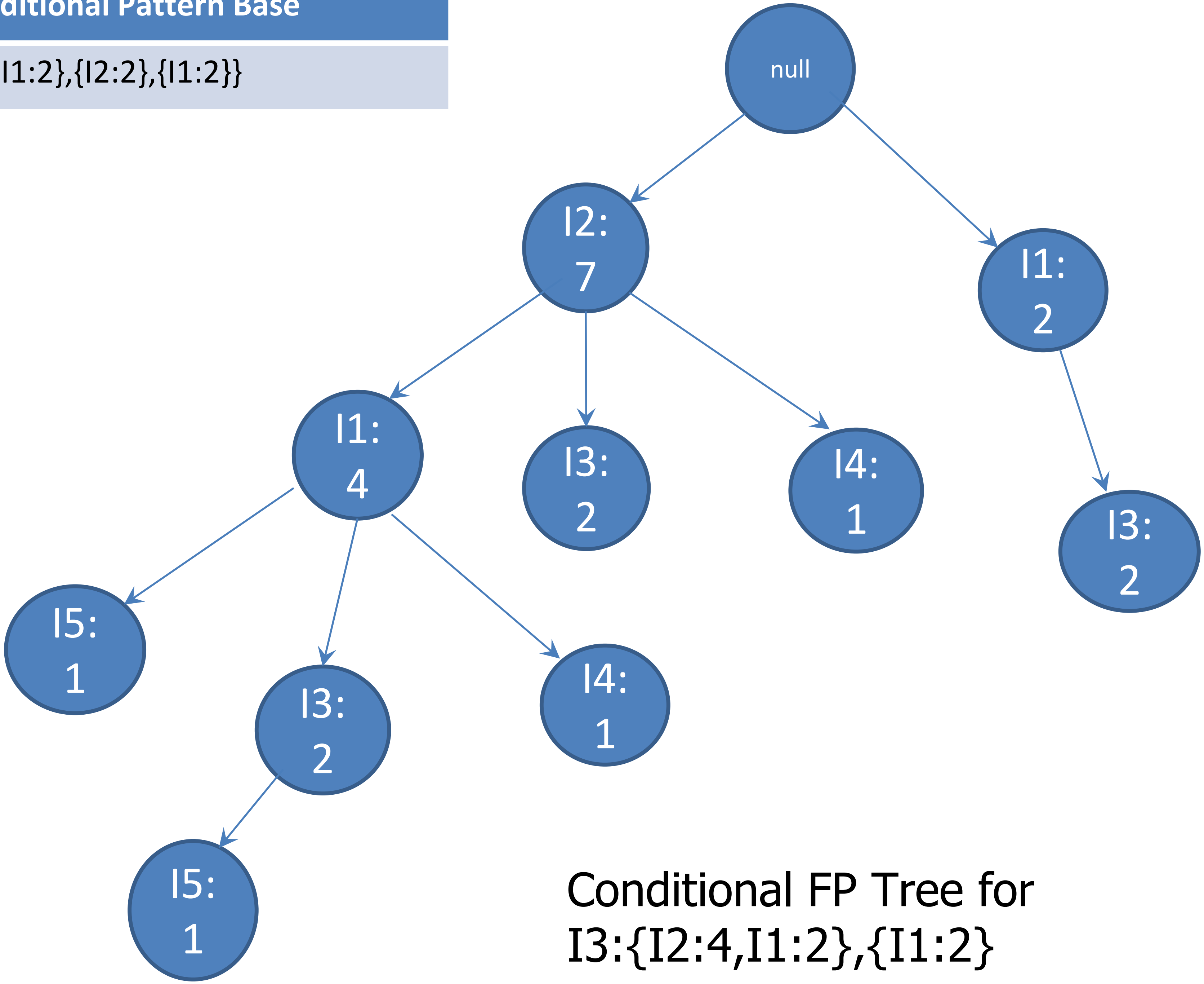
Conditional FP Tree for I5:{I2:2,I1:2}

	Conditional Pattern Base
I4	{{I2,I1:1},{I2:1}}



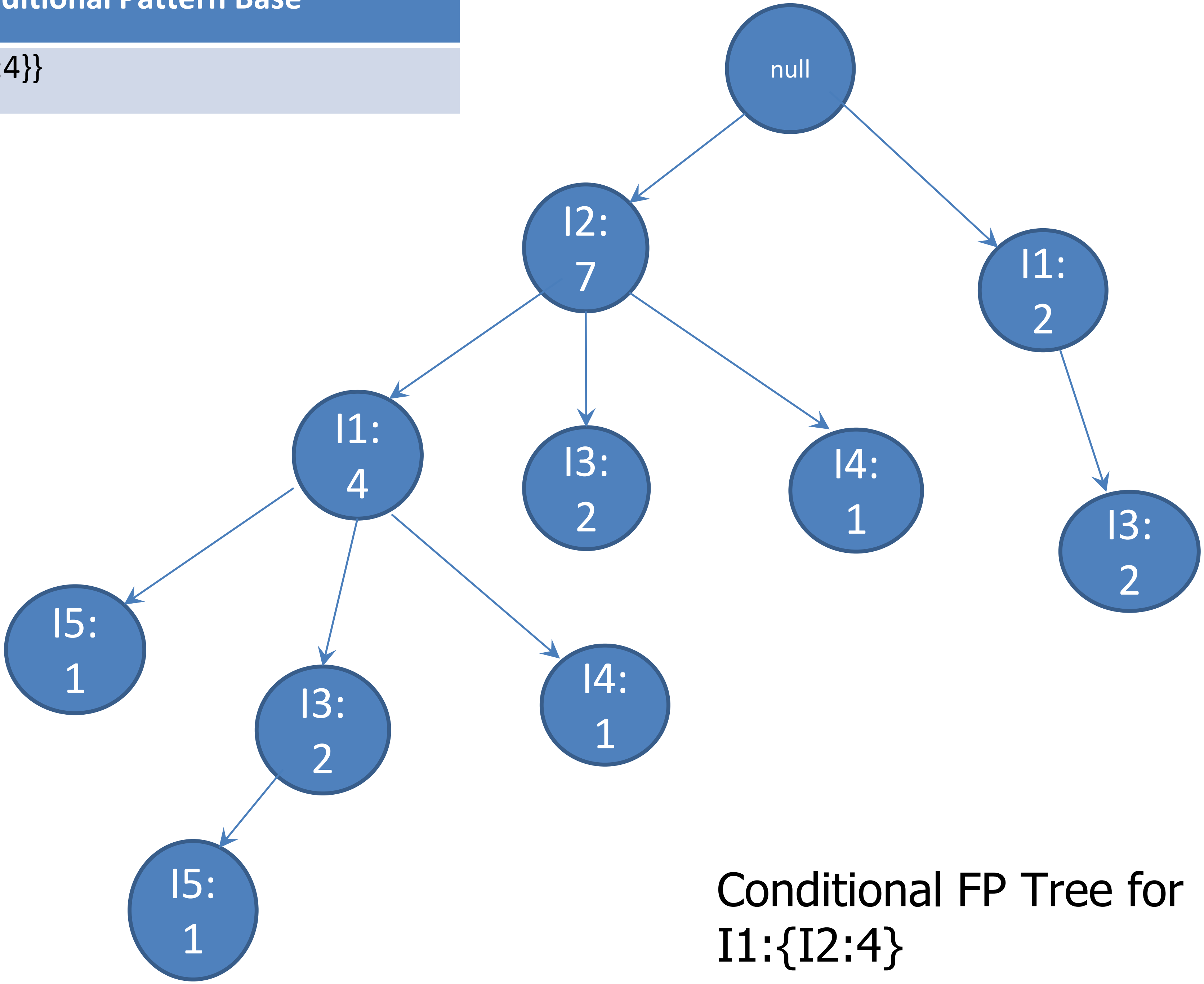
Conditional FP Tree for I4:{I2:2}

	Conditional Pattern Base
I3	{{I2,I1:2},{I2:2},{I1:2}}



Conditional FP Tree for I3:{I2:4,I1:2},{I1:2}

	Conditional Pattern Base
I1	{{I2:4}}



Conditional FP Tree for I1:{I2:4}

Frequent Patters Generated

	Conditional Pattern Base
I5	{{I2,I1:1},{I2,I1,I3:1}}

	Conditional Pattern Base
I4	{{I2,I1:1},{I2:1}}

	Conditional Pattern Base
I3	{{I2,I1:2},{I2:2},{I1:2}}

	Conditional Pattern Base
I1	{{I2:4}}

	Frequent Pattern Generated
I5	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{I2, I4: 2}
I3	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{I2, I1: 4}

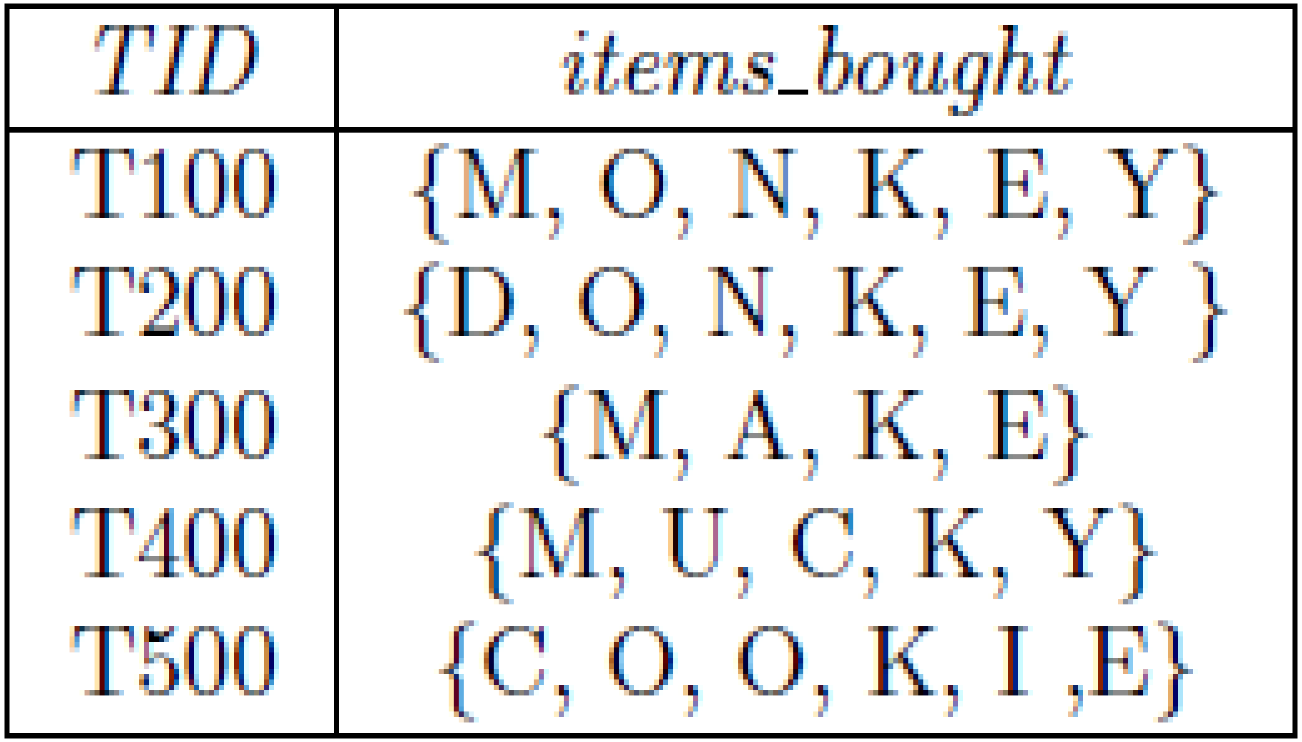
Exercise

- A database has 5 transactions. Let min sup = 60% and min conf = 80%.

(a) Find all frequent itemsets using FPGrowth

(b) List all of the strong association rules

Exercise



TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

Transactions Storage

Consider eight transactions with transaction IDs {10, 20, 30, 40, 50, 60, 70, 80}. This set of eight transactions with six items can be represented in at least three different ways as follows

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

Horizontal List Format

Horizontal Representation

Consider only eight transactions with transaction IDs {10, 20, 30, 40, 50, 60, 70, 80}. This set of eight transactions with six items

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

Horizontal vector Format:

Vertical Representation

Consider only eight transactions with transaction IDs {10, 20, 30, 40, 50, 60, 70, 80}. This set of eight transactions with six items

Vertical List Format:
For each item, a list of TID s of records containing that item is stored; is also referred as the tidlist of that item.

Items	Transactions
A	10,30,80
B	10,40
C	40,70,80
D	10,20,40,60,70,80
E	20,50,60
F	20,30,50,60,70,80

Vertical list Format:

Vertical Representation

Consider only eight transactions with transaction IDs {10, 20, 30, 40, 50, 60, 70, 80}. This set of eight transactions with six items

Items	10	20	30	40	50	60	70	80
A	1	0	1	0	0	0	0	1
B	1	0	0	1	0	0	0	0
C	0	0	0	1	0	0	1	1
D	1	1	0	1	0	1	1	1
E	0	1	0	0	1	1	0	0
F	0	1	1	0	1	1	1	1

Vertical vector Format:

Mining frequent itemsets using vertical data format

- Transform the horizontally formatted data to the vertical format by scanning the data set once.
- The support count of an itemset is simply the length of the TID_set of the itemset.
- Starting with k=1 the frequent k-itemsets can be used to construct the candidate (k+1)-itemsets.
- This process repeats with k incremented by 1 each time until no frequent itemsets or no candidate itemsets can be found.
  - The tidlist of an itemset is computed as the intersection of the tidlists of its items

Minimum support count=2

Itemset	TID_set
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

TID	List of item 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

The 2-itemsets in vertical data format:

Itemset	TID_set
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T300, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

The 3-itemsets in vertical data format

{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

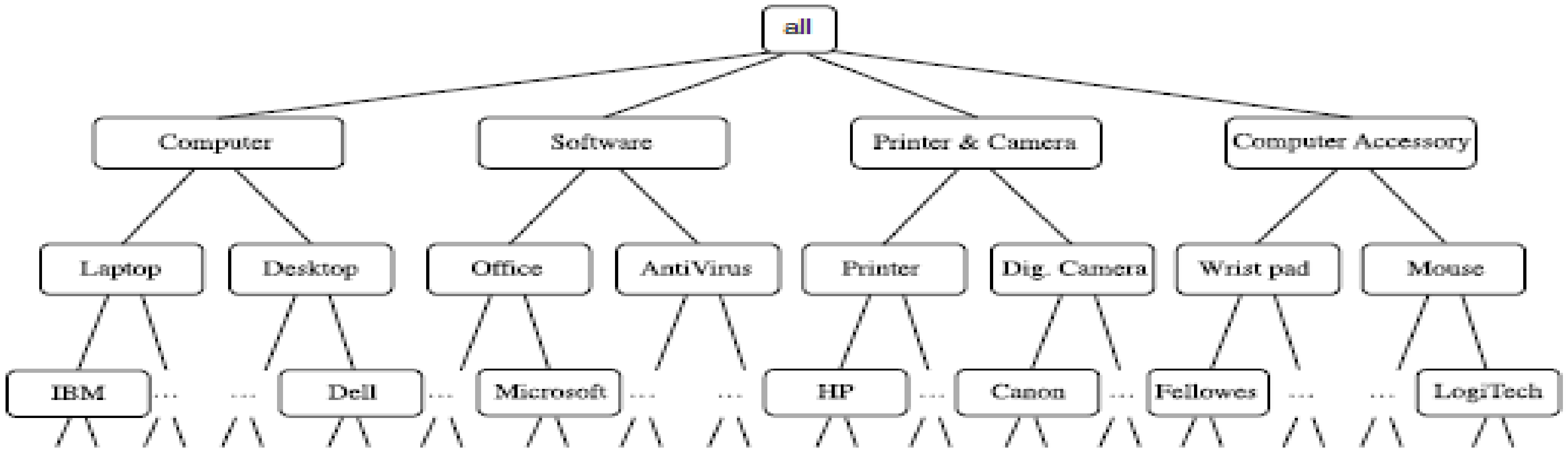
Mining frequent itemsets using vertical data format

- Advantages of this algorithm: -
- Better than Apriori in the generation of candidate (k+1)-itemset from frequent k itemsets –
- There is no need to scan the database to find the support (k+1) itemsets (for k>=1).
- This is because the TID_set of each k-itemset carries the complete information required for counting such support.

Introduction to Mining Multiple-level Association Rules and Multidimensional Association Rules

Multilevel Association Mining

TID	*Items Purchased*
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...



Single- vs. multilevel rules

- **Single-level association rules:**
- Associations between items or attributes at the same level of abstraction, i.e., at the same level of hierarchy
- Butter, Chips -> Bread [0.4%, 52%]
- **Multilevel association rules:**
- Associations between items or attributes at different levels of abstraction, i.e, at different levels of concept hierarchy.
- Butter:Amul, Chips:Balaji -> Bread [0.1%, 74%]

Introduction:
Why Multiple-Level Association Rules?

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}

Frequent itemset:
{b2}

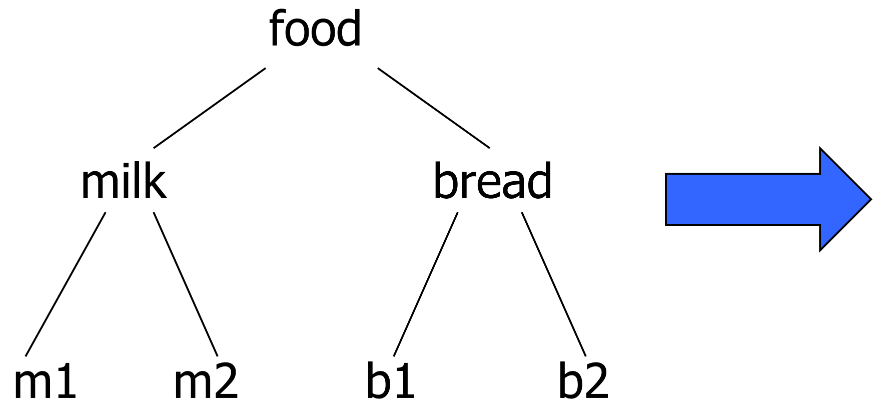
Association rules: none

Is this database useless?

Introduction:

Why Multiple-Level Association Rules?

What if we have this abstraction tree?



TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}

minisup = 50%   miniconf = 50%

Frequent itemset: {milk, bread}

A.A rules: milk <=> bread

Introduction:

Why Multiple-Level Association Rules?

- Sometimes, at primitive data level, data does not show any significant pattern. But there are useful information hiding behind.

- The goal of Multiple-Level Association Analysis is to find the hidden information in or between levels of abstraction

Introduction:

Requirements in Multiple-Level Association Analysis

Two general requirements to do multiple-level association rule mining:

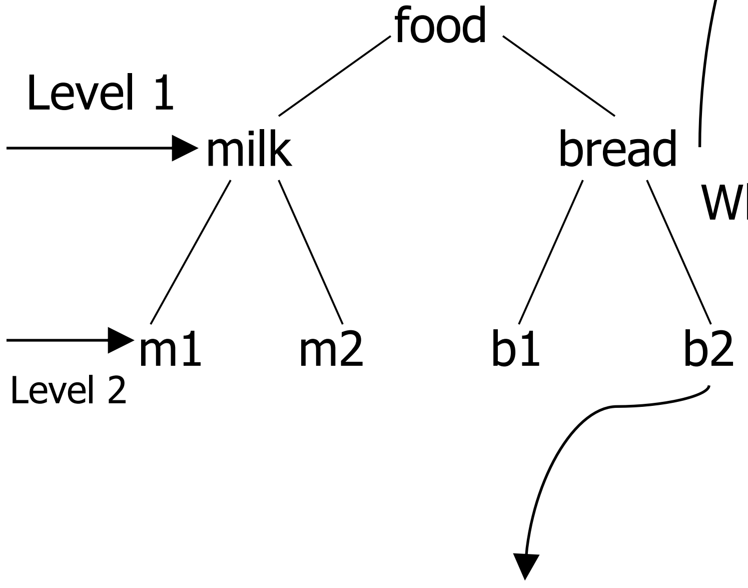
1) Provide data at multiple levels of abstraction.
2) Find efficient methods for multiple-level rule mining.

Multiple-Level Association Rules

TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}
T4	{milk, bread}
T5	{milk}

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}
T4	{m2, b1}
T5	{m2}

minisup = 50%   miniconf = 50%



Frequent itemset: {milk, bread}
A.A rule: milk <=> bread

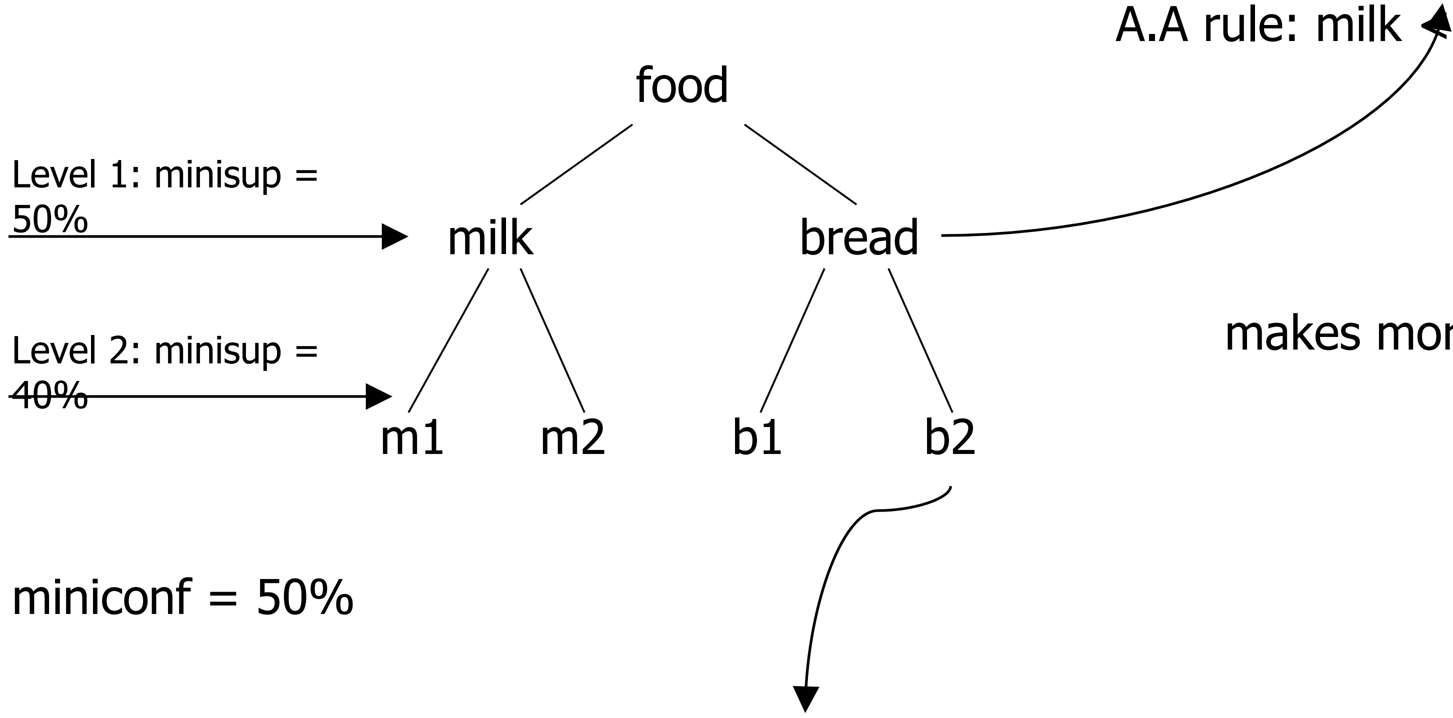
What about {m2, b1}?

One minisup for all levels?

Frequent itemset: {m2}
A.A rule: none

Multiple-Level Association Rules

Frequent itemset: {milk, bread}
A.A rule: milk <=> bread



makes more sense now

miniconf = 50%

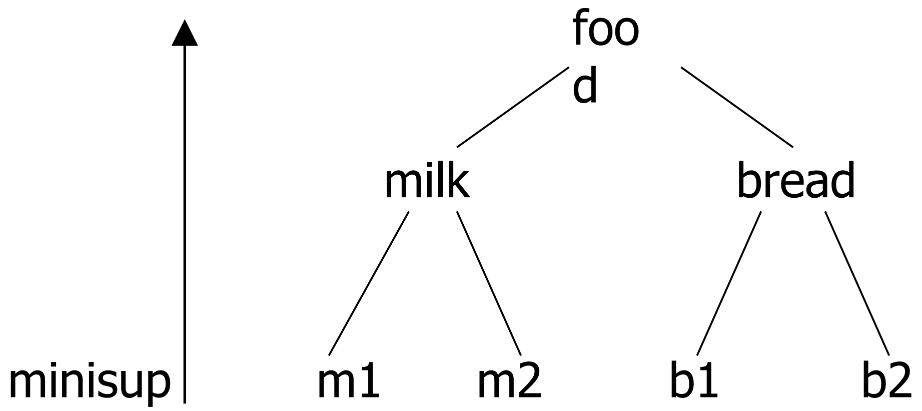
Frequent itemset: { m2, b1, b2}
A.A rule: m2 <=> b1

Multiple-Level Association Rules

Generalizing/specializing values of attributes:
from specialized to general: support of rules increases and new rules may become valid.
from general to specialized: support of rules decreases and rules may become not valid

- On a too low level of abstraction :
  high support = too few rules
  low support = too many rules, most uninteresting
- On a too high level of abstraction => uninteresting rules

Approach: ascending minisup on each level



Multiple-Level Association Rules

- A basic approach (top-down progressive deepening):
- Calculate frequent itemsets at each concept level, until no more frequent itemsets can be found
- For each level use, e.g., Apriori for finding frequent itemsets
- Example:
- First mine high-level frequent items - milk , bread – and rules based on them.
- Then mine their lower-level "weaker" frequent itemsets -
- 2% milk ,wheat bread - and the rules based on them.

Multiple-Level Association Rules

- Variations of the basic approach:
- Uniform minimum support for all levels
  - one minimum support threshold; simplified search
  - if too high, we can miss low level associations.
  - if too low, we can generate too many uninteresting high level associations.
- Reduced minimum support at lower levels different strategies possible: level-by-level, level cross filtering by single item, level-cross filtering by k-itemset, and controlled level-cross filtering by single item

Algorithm: An Example

An entry of **sales_transaction** Table

Transaction_id	Bar_code_set
351428	{17325,92108,55349,88157,…}

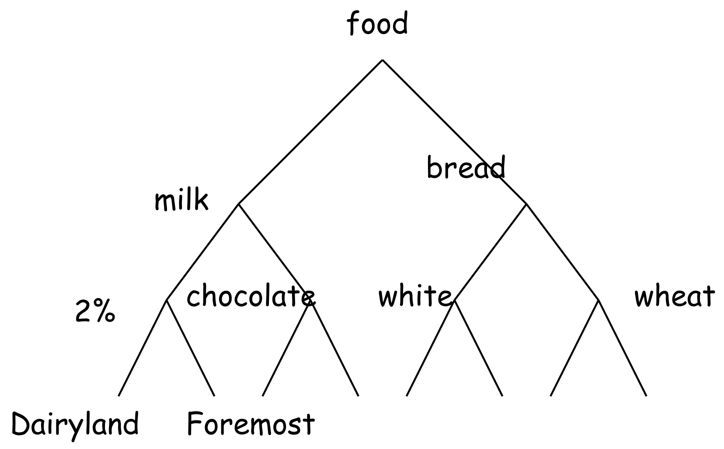
A **sales_item** Description Relation

Bar_code	category	Brand	Content	Size	Storage_pd	price
17325	Milk	Foremost	2%	1ga.	14(days)	$3.89

Algorithm: An Example

Encode the database with layer information

GID	bar_code	category	content	brand
112	17325	Milk	2%	Foremost



First 1: implies milk

Second 1: implies 2% content

2: implies Foremost brand

Algorithm: An Example

Encoded Transaction Table:T[1]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222,323}
T3	{112,122,221,411}
T4	{111,121}
T5	{111,122,211,221,413}
T6	{211,323,524}
T7	{323,411,524,713}

Algorithm: An Example

The frequent 1-itemset on level 1

L[1,1]

Itemset	Support
{1**}	5
{2**}	5

Level-1 minsup = 4

only keep items in L[1,1] from T[1] $\longrightarrow$

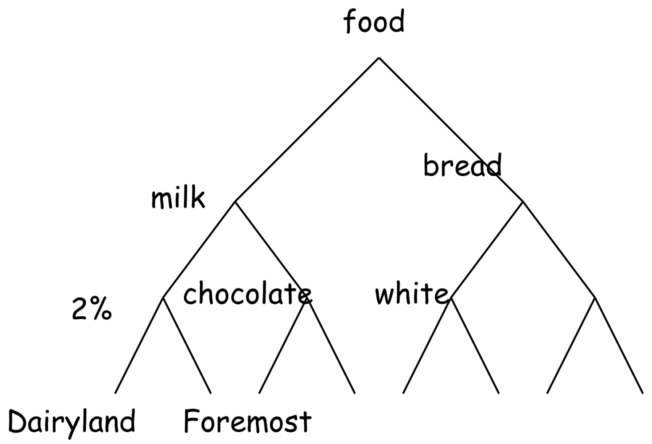
T[2]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222}
T3	{112,122,221}
T4	{111,121}
T5	{111,122,211,221}
T6	{211}

L[1,2]

Itemset	Support
{1**,2**}	4

Use Apriori on each level



Algorithm: An Example

Level-2 minsup = 3

L[2,1]

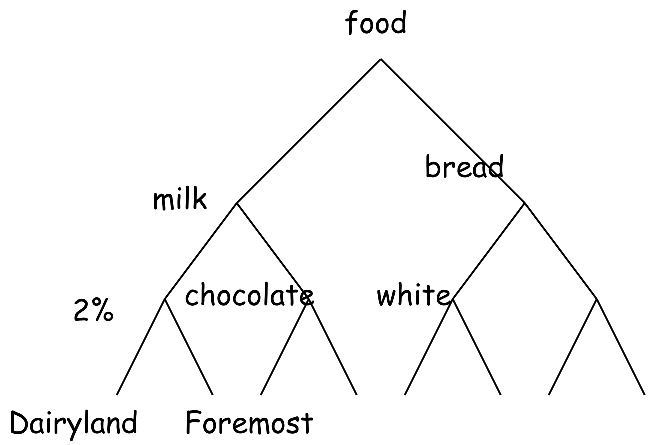
Itemset	Support
{11*}	5
{12*}	4
{21*}	4
{22*}	4

L[2,2]

Itemset	Support
{11*,12*}	4
{11*,21*}	3
{11*,22*}	4
{12*,22*}	3
{21*,22*}	3

L[2,3]

Itemset	Support
{11*,12*,22*}	3
{11*,21*,22*}	3



Frequent Item Sets at Level 3

Level-3 minsup = 3

L[3,1]

Itemset	Support
{111}	4
{211}	4
{221}	3

L[3,2]

Itemset	Support
{111,211}	3

E.g.

Level-1:

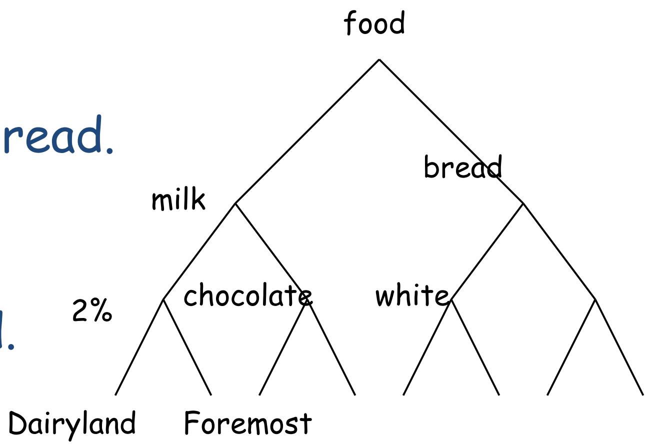
80% of customers that purchase milk also purchase bread.

milk ➔ bread with Confidence= 80%

Level-2:

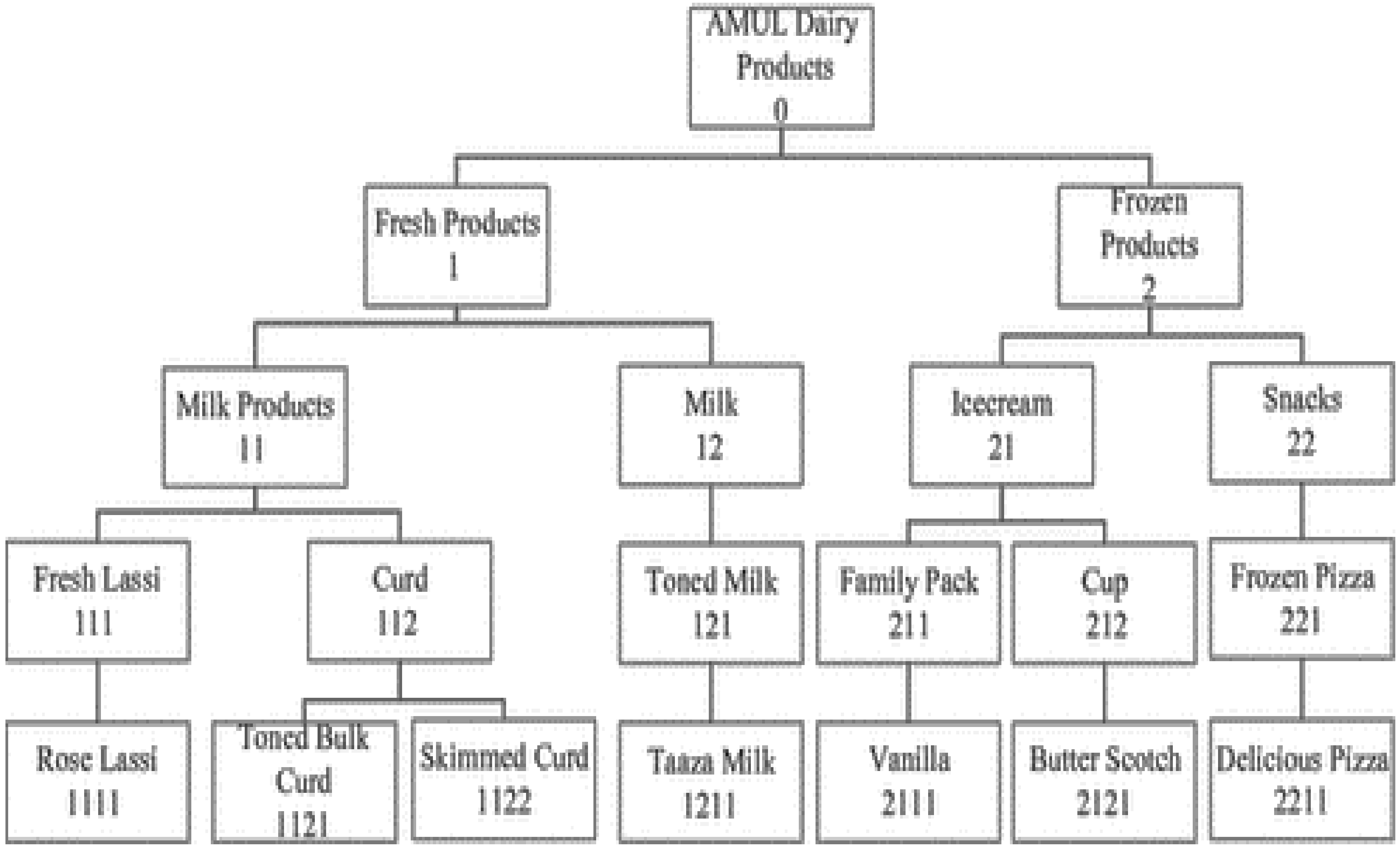
75% of people who buy 2% milk also buy wheat bread.

2% milk ➔ wheat bread with Confidence= 75%



Redundancy of multilevel association rules

- Some rules may be redundant due to "ancestor" relationships between items.
- Example (milk has 4 subclasses):
- milk -> wheat bread [support = 8%, confidence = 70%]
- 2% milk -> wheat bread [support = 2%, confidence = 72%]
- The first rule is an ancestor of the second rule.
- The second rule could be redundant.

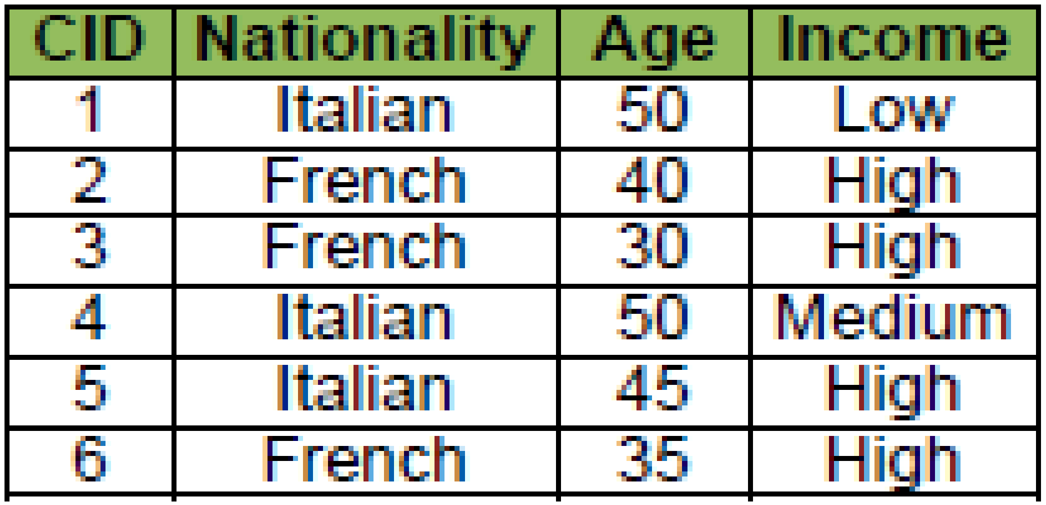

AMUL Dairy Products 0
Fresh Products 1
Frozen Products 2
Milk Products 11
Milk 12
Icecream 21
Snacks 22
Fresh Lassi 111
Curd 112
Toned Milk 121
Family Pack 211
Cup 212
Frozen Pizza 221
Rose Lassi 1111
Toned Bulk Curd 1121
Skimmed Curd 1122
Taaza Milk 1211
Vanilla 2111
Butter Scotch 2121
Delicious Pizza 2211

Multidimensional Association rules

- **Single-dimensional rules:**
- Items in the rule refer to only one dimension or predicate, e.g., to "buys".
  - buys(x, “milk") ^ buys(x, "Chips") -> buys(x, "Bread") [0.4%, 52%]
- **Multidimensional rules:**
- Items in the rule refer to two or more dimensions or predicates, e.g.,
  - "buys", "time_of_transaction", "customer_category".

Multidimensional Association rules

Database



CID	Nationality	Age	Income
1	Italian	50	Low
2	French	40	High
3	French	30	High
4	Italian	50	Medium
5	Italian	45	High
6	French	35	High

- Rules:
- “Nationality = French” -> “Income = high” [50%, 100%]
- “Income = high” -> “Nationality = French“ [50%, 75%]
- “Age = 50” -> “Nationality = Italian” [33%, 100%]

Multidimensional Association rules

- **Single-dimensional rules (intradimensional rules):**
  - Transactional data is used.
  - Items in a rule are assumed to belong to the same transaction.
- **Multidimensional rules:**
  - Relational data is used.
  - Attribute A in a rule is assumed to have value a, attribute B value b and attribute C value c in the same tuple.

Multidimensional Association rules

- Two types of rules
- Interdimensional rules: same attributes/predicates may not be repeated in a rule
  - *age*(*X*, “20:::29”)^*occupation*(*X*, “*student*”))*buys*(*X*, “*laptop*”).
- Hybrid-dimensional rules: same attributes/predicates can be repeated in a rule
  - *age*(*X*, “20:::29”)^*buys*(*X*, “*laptop*”))*buys*(*X*, “*HP printer*”)

From association mining to correlation analysis

- 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.
- *correlation rules* of the form
  - $A \Rightarrow B$ [*support*, *confidence*. *correlation*].

Interestingness Measure: Correlations (Lift)

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

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

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

Exercise 1

The following contingency table summarizes supermarket transaction data, where *hot dogs refers to the* transactions containing **hotdogs**, ***hotdogs (bar)*** *refers to the transactions that do not contain hot dogs, hamburgers* refers to the transactions containing **hamburgers**, and ***hamburgers (bar)*** *refers to the transactions that do not* contain hamburgers

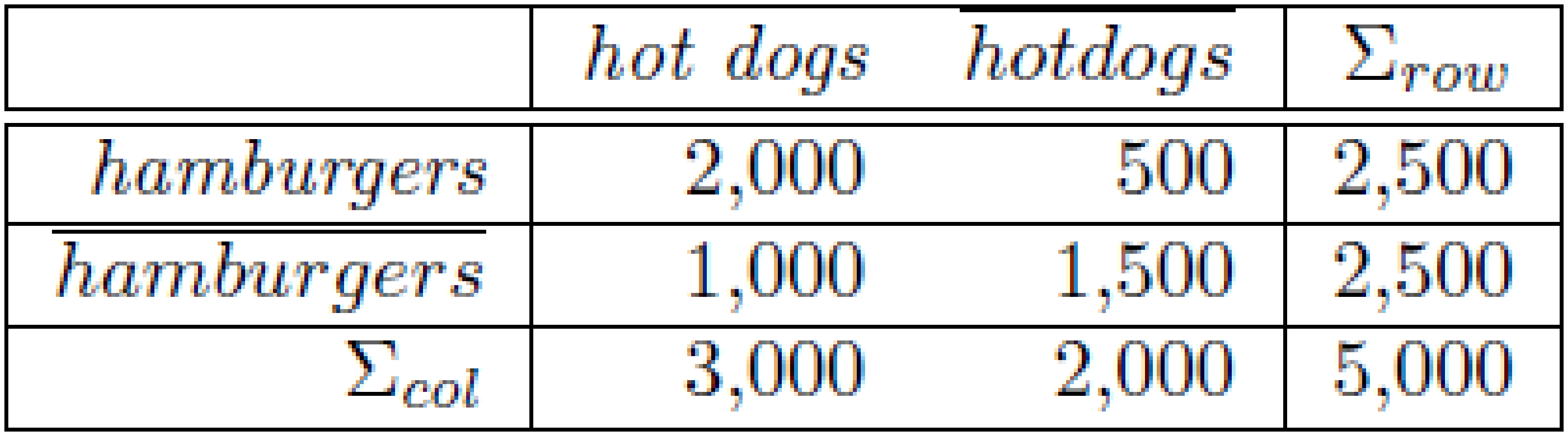
(a) Suppose that the association rule *"hot dogs =>hamburgers" is mined. Given a minimum support*

threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?

(b) Based on the given data, is the purchase of *hot dogs independent of the purchase of hamburgers? If*

not, what kind of *correlation relationship exists between the two?*

Conti..



	$hot\ dogs$	$\overline{hotdogs}$	Σ_{row}
$hamburgers$	2,000	500	2,500
$\overline{hamburgers}$	1,000	1,500	2,500
Σ_{col}	3,000	2,000	5,000

Solution 1

(a) For the rule, support = 2000/5000 = 40%, and confidence = 2000/3000 = 66.7%. Therefore, the association rule is strong.

(b) *Corrf (hotdog;hamburgerg)= P(fhot dog, hamburgerg)/(P(fhot dogg) P(hamburgerg))=0.4/(0.5 / 0.6) =*1.33*>1.*

So, the purchase of hotdogs is not independent of the purchase of hamburgers. There exists a positive correlation between the two.

Exams Questions

- *Q2: What are the problems in using normal Apiori methods??*
- A: One may apply the Apriori algorithm to examine data items at multiple levels of abstraction under the same minimum support and minimum confidence thresholds. This direction is simple, but it may lead to some undesirable results.

  First, large support is more likely to exist at high levels of abstraction. If one wants to find strong associations at relatively low levels of abstraction, the minimum support threshold must be reduced substantially; this may lead to the generation of many uninteresting associations at high or intermediate levels.

  Second, since it is unlikely to find many strong association rules at a primitive concept level, mining strong associations should be performed at a rather high concept level, which is actually the case in many studies. However, mining association rules at high concept levels may often lead to the rules corresponding to prior knowledge and expectations, such as "milk => bread", (which could be common sense), or lead to some uninteresting attribute combinations if the minimum support is allowed to be rather small, such as "toy => milk", (which may just happen together by chance).

Exams Questions

- *Q3: What are the 2 general steps to do multiple-level association rule mining?*
- A: To explore multiple-level association rule mining, one needs to provide:
- 1) Data at multiple levels of abstraction, and
- 2) Efficient methods for multiple-level rule mining.