Data Mining: Concepts and Techniques

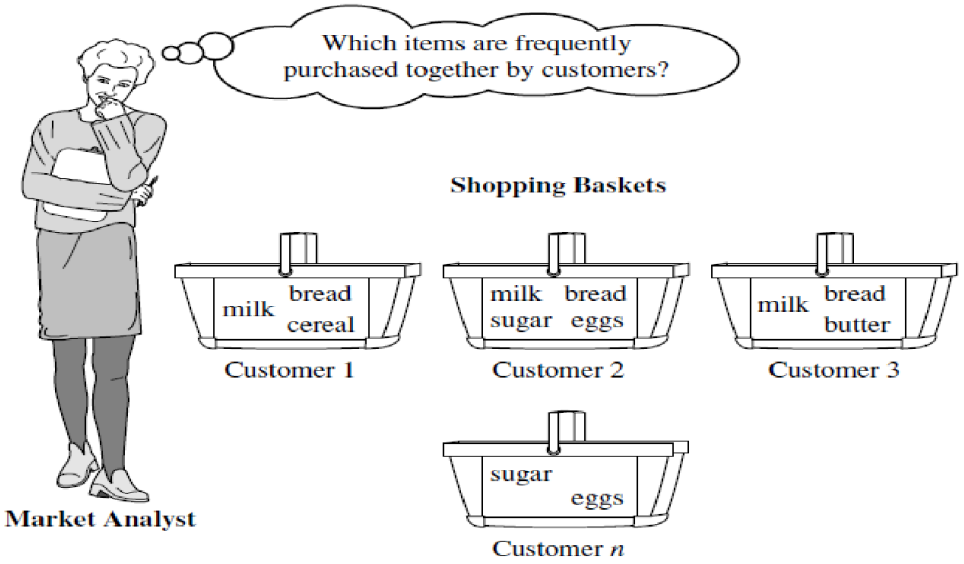
Mining Frequent Patterns, Association and Correlations

Frequent Patterns(itemsets/subsequences)

- Finding the frequent patterns plays an essential role in mining associations, correlations and some other type of interesting relationships among the data.
- Moreover, it helps in classification,clustering and other data mining tasks.

Frequent Itemset Mining:Market Basket Analysis

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



Market basket analysis.

MBA is the process of analyzing the **buying habits of customers** by finding the associations between the different items that the customers place in their basket.

The results of MBA can be used :

- To plan Marketing strategies.
- To plan Advertising strategies.
- To develop new designs (e.g Store layout designs)
- It help retailers to plan which items to put on sale at reduced prices.

Frequent Itemset Mining:Market Basket Analysis

- Frequent patterns can be represented in the form of **association rules**.

$$computer \Rightarrow antivirus\_software\ [support = 2\%, confidence = 60\%].$$

- Rule **support** and **confidence** are two measures of rule interestingness.
- They respectively reflect the usefulness and certainty of discovered rules.
- 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 a computer also bought the antivirus software.
- Typically, association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**.

Basic Terms of Association Rule Mining

Customers	*Transactions*
1	*milk, bread*
2	*bread, butter*
3	*beer*
4	*milk, bread, butter*
5	*bread*
6	*milk, bread, butter*

Itemset: A collection of one or more items.

e.g. {Milk, Bread, Butter}

Support Count(σ): It represents frequency of occurrence of an itemset. **Absolute support**

e.g. σ{Milk, Bread}=3

Support (s): Fraction of transactions that contain an itemset. **Relative support**

e.g s{Milk, Bread}=3/6

Frequent Itemset: An itemset whose support is greater than or equal to a minimum support threshold.

If minimum support threshold=3 then {Milk, Bread} are frequent

If minimum support threshold=4 then {Milk, Bread} are not frequent

Support and confidence

Association Rule

An implication expression of the form X→Y where X and Y are itemsets

E.g. {Milk, Bread}→Butter

Rule Evaluation Metrics

Support(S): Fraction of transaction that contain both X and Y.

Confidence(c): Measures how often items in Y appear in transactions that contain X.

e.g. {Milk, Bread}→Butter

Customers	*Transactions*
1	*milk, bread*
2	*bread, butter*
3	*beer*
4	*milk, bread, butter*
5	*bread*
6	*milk, bread, butter*

$$s = \frac{\sigma(\text{Milk, Bread, Butter})}{|T|} = \frac{2}{6}$$

$$c = \frac{\sigma(\text{Milk, Bread, Butter})}{(\text{Milk, Bread})} = \frac{2}{3}$$

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

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

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

- Rules that satisfy both a minimum support threshold (*min sup*) and a minimum confidence threshold (*min conf*) are called **strong**.

Support and Confidence

For association rule A => B,

Support (s)= $\sigma(A \cup B) / |T|$

Confidence (c) = $\sigma(A \cup B) / \sigma(A)$

Association Rule Mining steps

- In general, association rule mining can be viewed as 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.

Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

Frequent Itemset: A **frequent itemset** is an itemset whose support is greater than some user-specified minimum support

Closed Frequent Itemset: An itemset is closed if none of its immediate supersets has the same support as that of the itemset.

Maximal Frequent Itemset: An itemset is maximal frequent if none of its immediate supersets is frequent.

Downward Closure Property of Frequent Patterns

All subset of any frequent itemset must also be frequent.

Example:

If **Milk Bread Butter** is a frequent itemset, then the following itemsets are frequent;

Milk

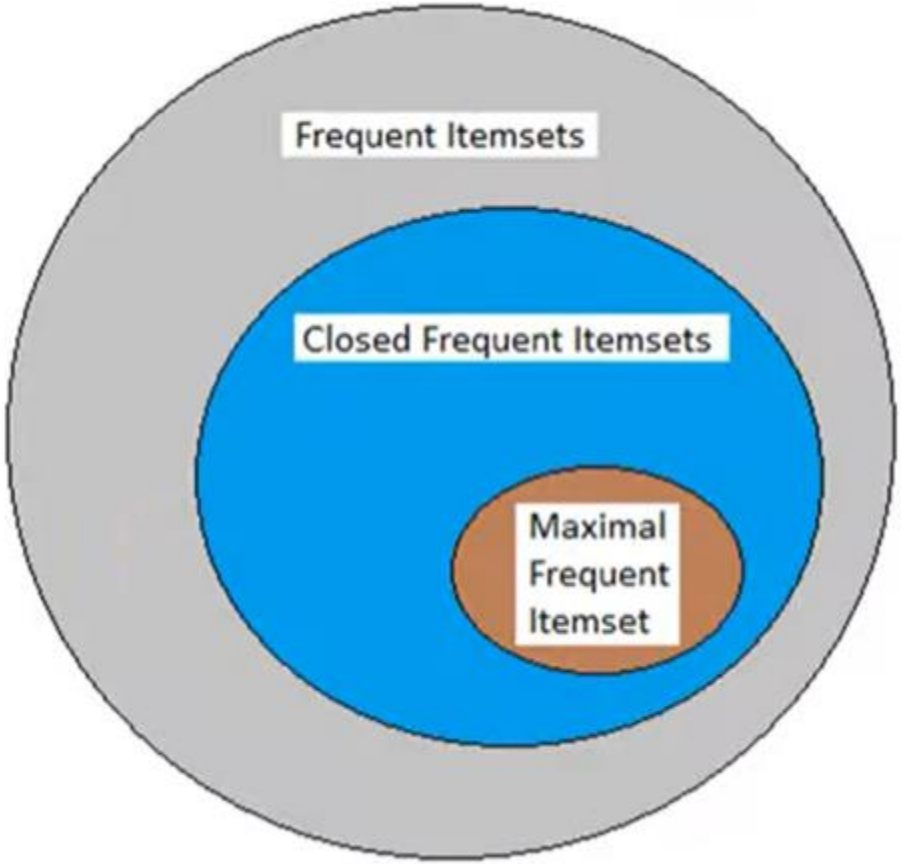
Bread

Butter

Milk, Bread

Milk, Butter

Bread, Butter


Frequent Itemsets
Closed Frequent Itemsets
Maximal
Frequent
Itemset

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

TID	List of Items
T1	A, B, C, D
T2	A, B,C, D
T3	A,B,C
T4	B, C, D
T5	C,D

Min Support Count=3,

Find Frequent, Closed, Maximal Itemset

Item	Count
A	3
B	4
C	5
D	4

All items that is A, B, C & D are **frequent** because they sup count is greater than or equal to minimum support count.

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

2-itemsets

Item	Count
A, B	3
A, C	3
A, D	2
B, C	4
B, D	3
C, D	4

All 2 items sets are frequent, except AD

A(3) → AB(3) , AC(3), AD(2)

A(count) is not greater than its immediate superset.

A is not closed.

In A's immediate superset, itemset are present with min. support count i.e. 3.

A is not maximal

B(4) → AB(3) , BC(4), BD(3)

B(count) is not greater than its immediate superset.

B is not closed.

In B's immediate superset, itemset are present with min. support count i.e. 3.

B is not maximal

Closed Frequent Itemset: An itemset is closed if none of its immediate supersets has the same support as that of the itemset.

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

2-itemsets

Item	Count
A, B	3
A, C	3
A, D	2
B, C	4
B, D	3
C, D	4

Is C closed,Maximal?

C(5) → AC(3) , BC(4), CD(2)

C(count) is greater than its immediate superset.

C is closed.

In C's immediate superset, itemset are present with min. support count i.e. 3.

C is not maximal

Is D closed,Maximal?

D(4) → AD(2) , BD(3), CD(4)

D(count) is not greater than its immediate superset.

D is not closed.

In D's immediate superset, itemset are present with min. support count i.e. 3.

D is not maximal

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

3-Itemsets

Item	Count
A, B,C	3
A, B,D	2
A, C,D	2
B,C,D	3

Is AB closed,Maximal?

AB(3) → ABC(3) , ABD(2)

AB(count) is not greater than its immediate superset.

AB is not closed.

In AB's immediate superset, itemset are present with min. support count i.e. 3.

AB is not maximal

Is AC closed,Maximal?

AC(3) → ABC(3) , ACD(2)

AC(count) is not greater than its immediate superset.

AC is not closed.

In AC 's immediate superset, itemset are present with min. support count i.e. 3.

AC is not maximal

Is AD frequent?

AD(2) → ABD(2) , ADC(2)

AD Not frequent

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

Item	Count
A, B,C	3
A, B,D	2
A, C,D	2
B,C,D	3

BC(4) → ABC(3) , BCD(3)

BC(count) is greater than its immediate superset.

BC is closed.

In BC's immediate superset, itemset are present with min. support count i.e. 3.

BC is not maximal

BD(3) → ABD(2) , BCD(3)

BD(count) is not greater than its immediate superset.

BD is not closed.

In AC 's immediate superset, itemset are present with min. support count i.e. 3.

BD is not maximal

CD(4) → ACD(2) , BCD(3)

CD(count) is greater than its immediate superset.

CD is closed.

In CD's immediate superset, itemset are present with min. support count i.e. 3.

CD is not maximal

Example: Frequent Itemset,Closed Itemset and Maximal Frequent Itemset

4-Itemsets

Item	Count
A, B,C, D	2

Is ABC closed,Maximal?

ABC(3) → ABCD(2)

ABC(count) is greater than its immediate superset.

ABC is closed.

In ABC's immediate superset, itemset are not present with min. support count i.e. 3.

ABC is maximal

What about ABCD?

ABCD is not frequent

ABD(2) → Not Frequent

ACD(2) → Not Frequent

Is BCD closed,Maximal?

BCD(3) → ABCD(2)

BCD(count) is greater than its immediate superset.

BCD is closed.

In BCD's immediate superset, itemset are not present with min. support count i.e. 3.

BCD is maximal

1-Itemset

Item	Freq.	Closed	Maximal
A (3)	☑	☒	☒
B (4)	☑	☒	☒
C (5)	☑	☑	☒
D (4)	☑	☒	☒

2-Itemset

Item	Freq.	Closed	Maximal
AB (3)	☑	☒	☒
AC (3)	☑	☒	☒
AD (2)	☒	☒	☒
BC (4)	☑	☑	☒
BD (3)	☑	☒	☒
CD (4)	☑	☑	☒

3-Itemset

Item	Freq.	Closed	Maximal
ABC (3)	☑	☑	☑
ABD (2)	☒	☒	☒
ACD (2)	☒	☒	☒
BCD (4)	☑	☑	☑

4-Itemset

Item	Freq.	Closed	Maximal
ABCD(2)	☒	☒	☒

Example

TID	List of Items
T1	B, J, P
T2	B, P
T3	B, M, P
T4	E, B
T5	E, M

Min Support Count=40%,

Find Frequent, Closed, Maximal Itemset

Example

TID	List of Items
T1	B, J, P
T2	B, P
T3	B, M, P
T4	E, B
T5	E, M

Min Support Count=40%,

Find Frequent, Closed, Maximal Itemset

All items that is B, E, M, P are **frequent** except J because they sup count is greater than or equal to minimum support count.

Item	Count
B	4
E	2
J	1
M	2
P	3

Item	Count
B	4
E	2
M	2
P	3

Frequent 1-itemsets

Frequent 2-itemsets

Item	Count
B, E	1
B, M	1
B, P	3
E, M	1
E,P	--
M, P	1

All 2 items sets are not frequent, except BP

B(4) → BE(1) , BM(1), BP(3)

B(count) is greater than its immediate superset.

B is closed.

In B's immediate superset, itemset are present with min. support count i.e. 2.

B is not maximal

E(2) → BE(1) , EM(1), EP(0)

E(count) is greater than its immediate superset.

E is closed.

In E's immediate superset, itemset are not present with min. support count i.e. 3.

E is maximal

Item	Count
B, E	1
B, M	1
B, P	3
E, M	1
E,P	--
M, P	1

All 2 items sets are not frequent, except BP

M(2) → BM(1) , EM(1), MP(1)

M(count) is greater than its immediate superset.

M is closed.

In M's immediate superset, itemset are not present with min. support count i.e. 2.

M is maximal

P(3) → BP(3) , EP(0), MP(1)

E(count) is not greater than its immediate superset.

P is not closed.

In E's immediate superset, itemset not present with min. support count i.e. 3.

P is not maximal

Three itemset is empty.

BP is closed and maximal

1-Itemset

Item	Freq.	Closed	Maximal
B (4)	☑	☑	☒
E (4)	☑	☑	☑
J (5)	☒	☒	☒
M (4)	☑	☑	☑
P (3)	☑	☒	☒

2-Itemset

Item	Freq.	Closed	Maximal
BE (1)	☒	☒	☒
BM (1)	☒	☒	☒
BP (3)	☒	☑	☑
EM(1)	☒	☒	☒
EP(0)	☒	☒	☒
MP (1)	☒	☒	☒

Frequent Itemset Mining Methods:
Apriori Algorithm: Finding Frequent Itemsets by Confined Candidate Generation

- The algorithm 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 by *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 Apriori property (Any subset of frequent itemset must be frequent) is used to reduce the search space.

"How is the Apriori property used in the algorithm?"

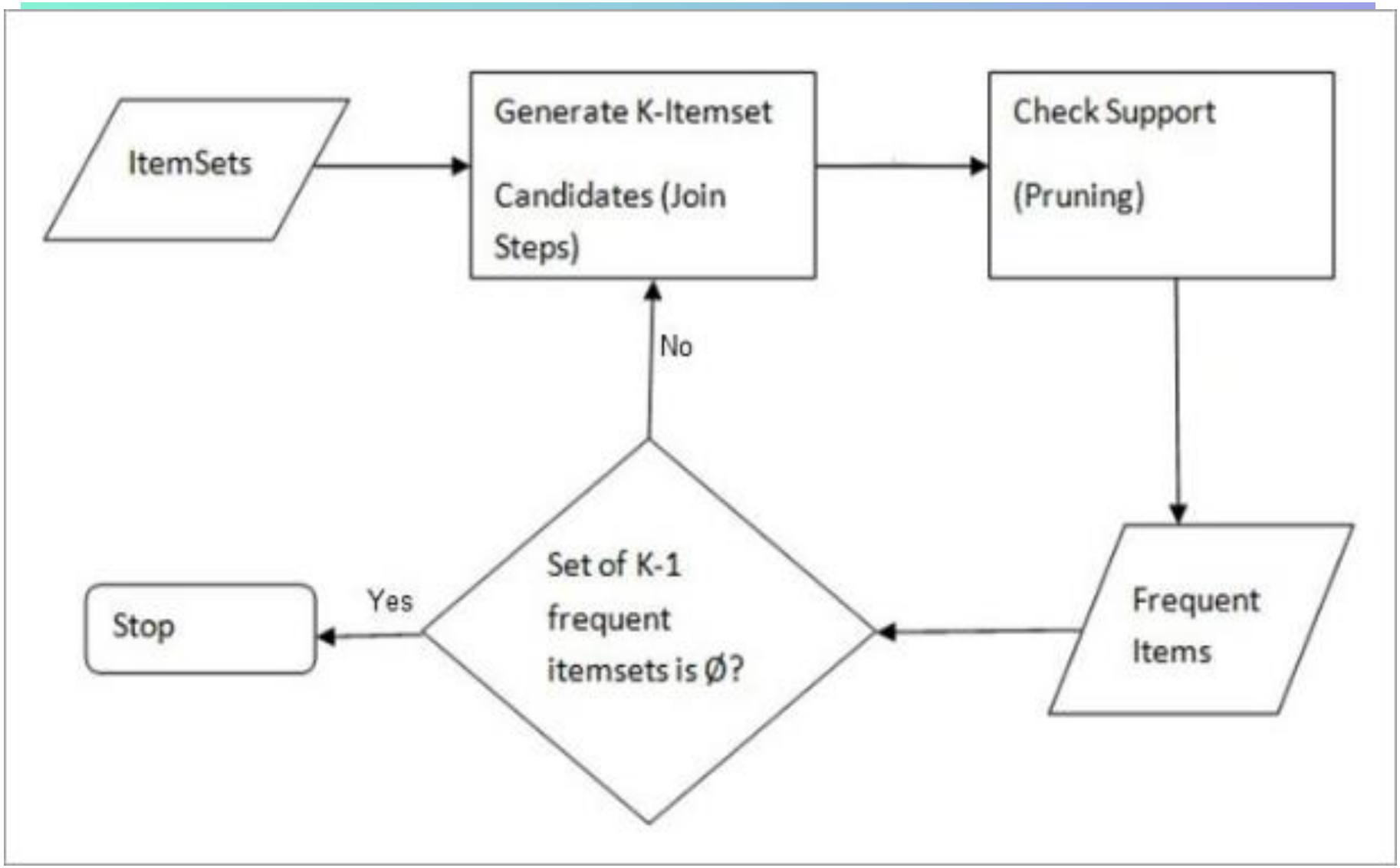
How L_{k-1} is used to find L_k for k= 2?

1. **Join Step**: To find L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted as C_k.
   - The join, $L_{k-1} \bowtie L_{k-1}$, is performed, **where members of L_{k-1} are joinable if their first (k -2) items are in common.**

2. **The prune step:** Select only frequent itemsets

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!

Apriori Algorithm Process



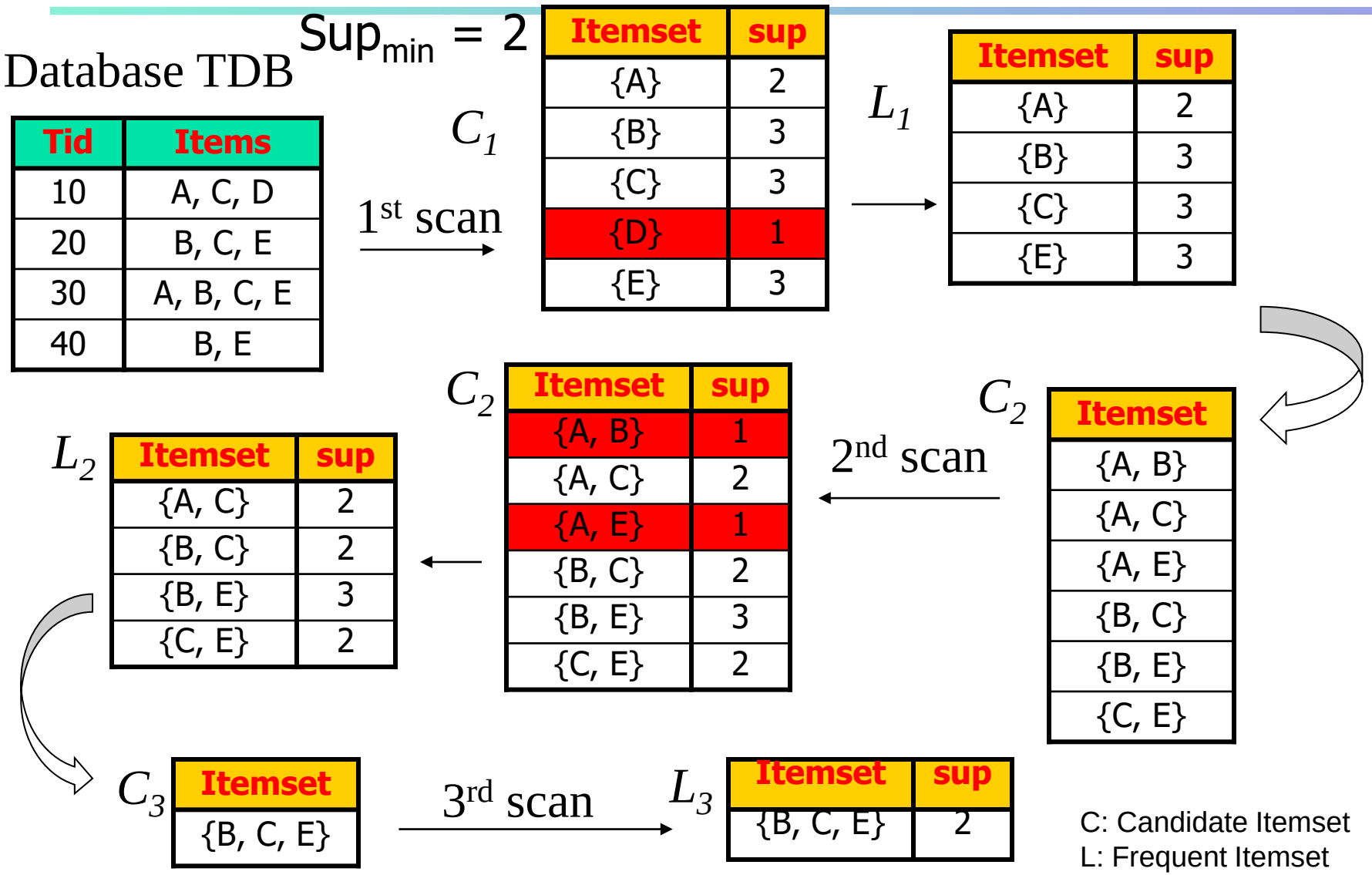
The Apriori Algorithm

- 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 $\cup_k L_k$;

The Apriori Algorithm—An Example



Example:Apriori Algorithm

TID	List of Items
T1	I1, I2, I4
T2	I1, I2, I5
T3	I1, I3, I5
T4	I2, I4
T5	I2, I3
T6	I1, I2, I3, I5
T7	I1, I3
T8	I1, I2, I3
T9	I2,I3
T10	I3, I5

Min Support Count=2, Confidence =70%

Generate Association Rule using

Apriori Algorithm

Example:Apriori Algorithm

Step 1: Generating 1-itemset frequent pattern.

C1

Itemset	Support Count
I1	6
I2	7
I3	7
I4	2
I5	4

Scan D for count of each candidate

L1

Itemset	Support Count
I1	6
I2	7
I3	7
I4	2
I5	4

Compare candidate support count with minimum support count

C: candidate itemset

L: Frequent itemset

Example:Apriori Algorithm

Step 2: Generating 2-itemset frequent pattern.

C2

Itemset	Sup. Count
I1, I2	4
I1, I3	4
I1, I4	1
I1, I5	3
I2, I3	4
I2, I4	2
I2, I5	2
I3, I4	0
I3, I5	3
I4, I5	0

L2

Itemset	Sup. Count
I1, I2	4
I1, I3	4
I1, I5	3
I2, I3	4
I2, I4	2
I2, I5	2
I3, I5	3

Compare candidate sup. count with min. Sup. count

Generate C2 from L1 and scan D for count of each candidate

Example:Apriori Algorithm

Frequent Itemsets: The sets of item which has minimum support.

Apriori Property: Any subset of frequent itemset must be frequent.

Join Operation: To find L_K, a set of candiate k-itemsets is generated by joining L_{k-1} with itself.

Generate candidate set C3 (k=3) using L2 (join step).

Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.

So here, for L2, first element should match.

Prune Operation: Select only frequent itemset.

L2

Itemset	Sup. Count
I1, I2	4
I1, I3	4
I1, I5	3
I2, I3	4
I2, I4	2
I2, I5	2
I3, I5	3

$C3 = L2 \bowtie L2$

$(I1,I2) \bowtie (I1,I3) = (I1, I2, I3)$

$(I1,I2) \bowtie (I1,I5) = (I1, I2, I5)$

$(I1,I3) \bowtie (I1,I5) = (I1, I3, I5)$

$(I2,I3) \bowtie (I2,I4) = (I2, I3, I4)$

$(I2,I3) \bowtie (I2,I5) = (I2, I3, I5)$

$(I2,I4) \bowtie (I2,I5) = (I2, I4, I5)$

Example:Apriori Algorithm

Step 3: Generating 3-itemset frequent pattern.

L2

Itemset	Sup. Count
I1, I2	4
I1, I3	4
I1, I5	3
I2, I3	4
I2, I4	2
I2, I5	2
I3, I5	3

C3

Itemset	Sup. Count
I1, I2, I3	2
I1, I2, I5	2
I1, I3, I5	2
I2, I3, I4	0
I2, I3, I5	1
I2, I4, I5	0

Generate C3 from L2 and scan D for count of each candidate

L3

Itemset	Sup. Count
I1, I2, I3	2
I1, I2, I5	2
I1, I3, I5	2

Compare candidate sup. count with min. Sup. count

Example:Apriori Algorithm

Step 4: Generating 4-itemset frequent pattern.

L3

Itemset	Sup. Count
I1, I2, I3	2
I1, I2, I5	2
I1, I3, I5	2

C4

Itemset	Sup. Count
I1, I2, I3, I5	1

Generate C4 from L3 and scan D for count of each candidate

L4

Φ

Compare candidate sup. count with min. Sup. count

Expand Rule from L3 as L4 is Φ

All three rules can be expanded

Example:Apriori Algorithm

We can expand any rule.
For e.g. I1, I2 and I5

Confidence=70%

Association Rule	Confidence	Confidence (%)
I1^I2→I5	C(I1,I2,I5)/C(I1,I2)=2/4	50%
I1^I5→I2	C(I1,I2,I5)/C(I1,I5)=2/2	100%
I2^I5→I1	C(I1,I2,I5)/C(I2,I5)=2/2	100%
I1→I2^I5	C(I1,I2,I5)/C(I1)=2/6	33%
I2→I1^I5	C(I1,I2,I5)/C(I2)=2/7	29%
I5→I1^I2	C(I1,I2,I5)/C(I5)=2/2	100%

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules are output, because these are the only ones generated that are strong.

Exercise

TID	List of Items
T1	Hotdogs, Buns, Ketchup
T2	Hotdogs, Buns
T3	Hotdogs, Coke Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	Hotdogs, Coke, Chips

Min Support =33.34%, Confidence =60%

Generate Association Rule

Using Apriori Algorithm

(x/6)*100=33.34%

x=2 Min. Support Count

Exercise

Step 1: Generating 1-itemset frequent pattern.

C1

Itemset	Support Count
Buns	2
Chips	4
Coke	3
Hotdogs	4
Ketchup	2

Scan D for count of each candidate

L1

Itemset	Support Count
Buns	2
Chips	4
Coke	3
Hotdogs	4
Ketchup	2

Compare candidate support count with minimum support count

C: Candidate item set
L: Frequent Item Set

Exercise

Step 2: Generating 2-itemset frequent pattern.

C2

Itemset	Sup. Count
Buns, Chips	0
Buns, Coke	0
Buns, Hotdogs	2
Buns, Ketchup	1
Chips, Coke	3
Chips, Hotdogs	2
Chips, Ketchup	1
Coke, Hotdogs	2
Coke, Ketchup	0
Hotdogs, Ketchup	1

Generate C2 from L1 and scan D

L2

Itemset	Sup. Count
Buns, Hotdogs	2
Chips, Coke	3
Chips, Hotdogs	2
Coke, Hotdogs	2

Compare candidate sup. count with min. Sup. count

Exercise

Join Operation: To find L_K, a set of candiate k-itemsets is generated by joining L_{k-1} with itself.

Generate candidate set C3 (k=3) using L2 (join step).

Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.

So here, for L2, first element should match.

Prune Operation: Select only frequent itemset.

L2

Itemset	Sup. Count
Buns, Hotdogs	2
Chips, Coke	3
Chips, Hotdogs	2
Coke, Hotdogs	2

$C3=L2 \bowtie L2$

(Chips, Coke) $\bowtie$ (Chips, Hotdogs)

= (Chips, Coke, Hotdogs)

Exercise

Step 3: Generating 3-itemset frequent pattern.

L2

Itemset	Sup. Count
Buns, Hotdogs	2
Chips, Coke	3
Chips, Hotdogs	2
Coke, Hotdogs	2

C3

Itemset	Sup. Count
Chips, Coke, Hotdogs	2

Generate C3 from L2 and scan D for count of each candidate

L3

Itemset	Sup. Count
Chips, Coke, Hotdogs	2

Compare candidate sup. count with min. Sup. count

C4=NULL, Hence L4=NULL

4 itemset can not be generated. Rules are formed from L3 (Chips, Coke, Hotdogs)

Exercise

{Chips, Coke, Hotdogs}

Confidence =60%

Association Rule	Confidence	Confidence (%)
Chips ^ Coke→ Hotdogs	C(Chips, Coke, Hotdogs)/C(Chips, Coke)=2/3	67%
Chips^ Hotdogs →Coke	C(Chips, Coke, Hotdogs)/C(Chips, Hotdogs)=2/2	100%
Coke ^ Hotdogs→ Chips	C(Chips, Coke, Hotdogs)/C(Coke, Hotdogs)=2/2	100%
Chips → Coke ^ Hotdogs	C(Chips, Coke, Hotdogs)/C(Chips)=2/4	50%
Coke → Chips ^ Hotdogs	C(Chips, Coke, Hotdogs)/C(Coke)=2/3	67%
Hotdogs → Chips ^ Coke	C(Chips, Coke, Hotdogs)/C(Hotdogs)=2/4	50%

Exercise 2

TID	List of Items
T1	E, A, D, B
T2	D, A, C, E, B
T3	C, A, B, E
T4	B, A, D
T5	D
T6	D, B
T7	A, D, E
T8	B, C

Min Support =30%, Confidence =75%

Generate Association Rule

Using Apriori Algorithm

min Sup=3

Exercise 2

Step 1: Generating 1-itemset frequent pattern.

C1

Itemset	Support Count
A	5
B	6
C	3
D	6
E	4

L1

Itemset	Support Count
A	5
B	6
C	3
D	6
E	4

Scan D for count of each candidate

Compare candidate support count with minimum support count

C: Candidate item set **L: Frequent Item Set**

Exercise 2

Step 2: Generating 2-itemset frequent pattern.

C2

Itemset	Sup. Count
A, B	4
A, C	2
A, D	4
A, E	4
B, C	2
B, D	4
B, E	3
C, D	1
C, E	2
D, E	3

L2

Itemset	Sup. Count
A, B	4
A, D	4
A, E	4
B, D	4
B, E	3
D, E	3

Compare candidate sup. count with min. Sup. count

Generate C2 from L1 and scan D

Exercise 2

Join Operation: To find L_K, a set of candiate k-itemsets is generated by joining L_{k-1} with itself.

Generate candidate set C3 (k=3) using L2 (join step).

Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.

So here, for L2, first element should match.

Prune Operation: Select only frequent itemset.

L2

Itemset	Sup. Count
A, B	4
A, D	4
A, E	4
B, D	4
B, E	3
D, E	3

$C3 = L2 \bowtie L2$

$(A,B) \bowtie (A, D) = (A, B, D)$

$(A,B) \bowtie (A, E) = (A, B, E)$

$(A,D) \bowtie (A, E) = (A, D, E)$

$(B,D) \bowtie (B, E) = (B, D, E)$

Exercise 2

Step 3: Generating 3-itemset frequent pattern.

L2

Itemset	Sup. Count
A, B	4
A, D	4
A, E	4
B, D	4
B, E	3
D, E	3

C3

Itemset	Sup. Count
A, B, D	3
A, B, E	3
A, D, E	3
B, D, E	2

Generate C3 from L2 and scan D for count of each candidate

L3

Itemset	Sup. Count
A, B, D	3
A, B, E	3
A, D, E	3

Compare candidate sup. count with min. Sup. count

Exercise 2

Step 4: Generating 4-itemset frequent pattern.

L3

Itemset	Sup. Count
A, B, D	3
A, B, E	3
A, D, E	3

C4

Itemset	Sup. Count
A, B, D, E	2

Generate C3 from L2 and scan D for count of each candidate

L4

Φ

Compare candidate sup. count with min. Sup. count

4 itemset is Null.

Rules are formed from L3.

We can expand all or some association rules

Exercise 2

{A, B, D}

Confidence = 75%

Association Rule	Confidence	Confidence (%)
A ^ B → D	C(A, B, D)/C(A, B)=3/4	75%
A^ D →B	C(A, B, D)/C(A, D)=3/4	75%
B ^ D→ A	C(A, B, D)/C(B, D)=3/4	75%
A → B ^ D	C(A, B, D)/C(A)=3/5	60%
B → A ^ D	C(A, B, D)/C(B)=3/6	50%
D → A ^ B	C(A, B, D)/C(D)=3/6	50%

Apriori Algorithm

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - heavy workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Improving Efficiency of Apriori Algorithm

Improving the efficiency of Apriori

a) Hash Based Techniques

b) Transaction Reduction

c) Partitioning

d) Dynamic Itemset Counting

e) Sampling

Improving 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), we can do following:

1. Generate all the 2-itemsets for each transaction,
2. Hash (i.e., map) them into the different buckets of a hash table structure, and
3. 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 (especially when k = 2).

Improving Efficiency of Apriori Algorithm: Hash Based Technique

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

Improving Efficiency of Apriori Algorithm

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

Improving Efficiency of Apriori Algorithm: Transaction Reduction (reducing the number of transactions scanned in future iterations)

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

Improving Efficiency of Apriori Algorithm: Transaction Reduction

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

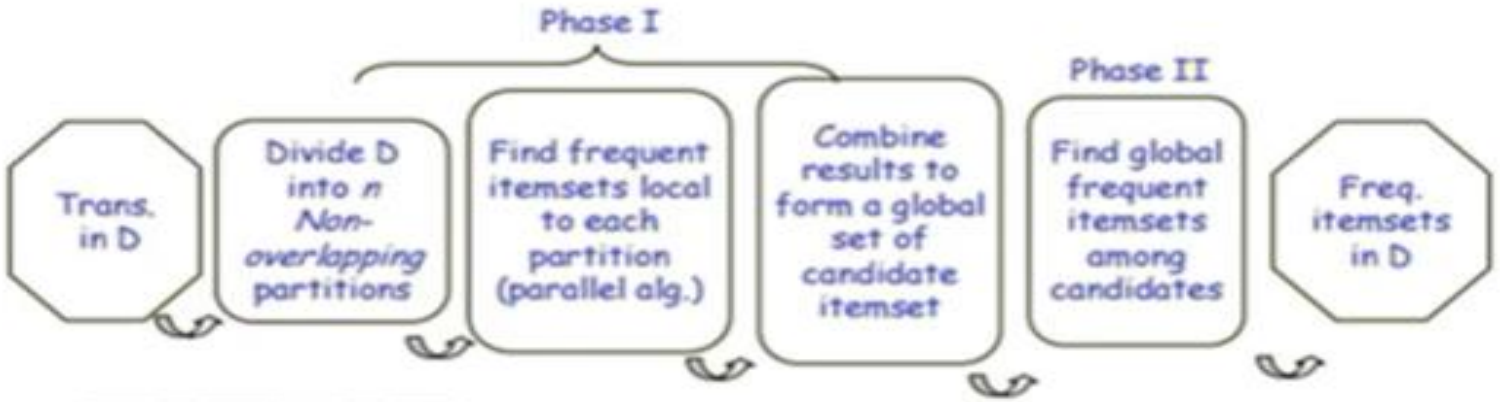
	I2,I3, I4
T2	1
T4	1

This method reduces no of **transaction** scans.

Improving Efficiency of Apriori Algorithm

c) Partitioning

Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB (2 DB Scan)



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

Database is divided into three partitions.

Each having two transactions with support of 20%

Improving Efficiency of Apriori Algorithm: Partitioning

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	{I2, I3, I4}-1	
		{I3, I4}-1 {I2, I3, I4}-1		

Improving Efficiency of Apriori Algorithm: 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 thanwith Apriori for finding all the frequent itemsets

Improving Efficiency of Apriori Algorithm:

d) Dynamic Itemset Counting

It is an algorithm which reduces the number of passes made over the data while keeping the number of itemsets which are counted in any pass relatively low.

This technique can add new candidate itemsets at any marked start point of the database during the scanning of the database.

Trans.	Items
T1	A,B
T2	A
T3	B,C
T4	-

Trans.	A	B	C
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

Min supp = 25% and M = 2

Improving Efficiency of Apriori Algorithm:DIC

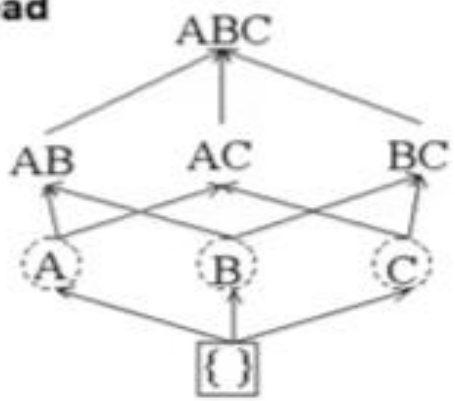
Solid box: □ Confirmed frequent itemset - an itemset we have finished counting and exceeds the support threshold *minsupp*

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 lattice before any transactions are read



Empty itemset is marked with a **solid box.**
All 1-itemsets are marked with **dashed circles.**

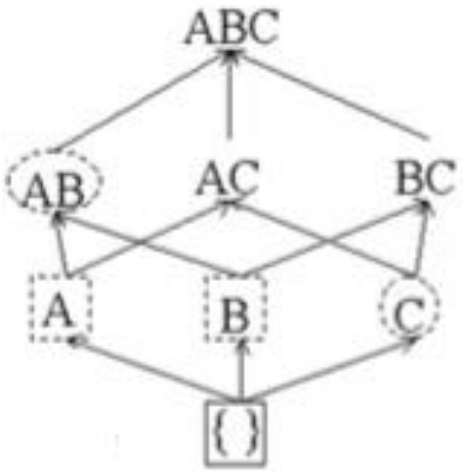
Counters: A = 0, B = 0, C = 0

DIC Algorithm

1. **Mark the empty itemset with a solid square. Mark all the 1-itemsets with dashed circles. Leave all other itemsets unmarked.**
2. While any dashed itemsets remain:
    1. 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.
    2. 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.
    3. Once a dashed itemset has been counted through all the transactions, make it solid and stop counting it.

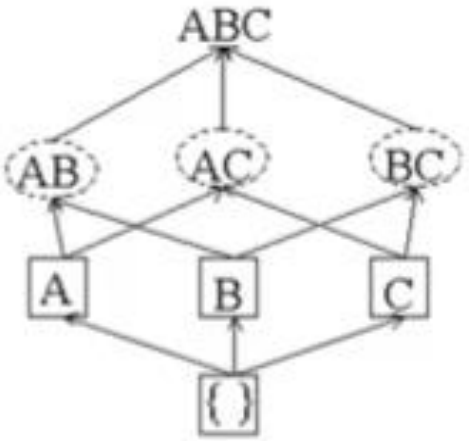
Improving Efficiency of Apriori Algorithm:DIC

After M transactions are read



Trans.	A	B	C
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

After 2M transactions are read



Counters: A = 2, B = 1, C = 0, AB = 0

Change A and B to **dashed boxes** because their counters are greater than minsup (1) and add a counter for AB because both of its subsets are boxes.

Counters: A = 2, B = 2, C = 1,

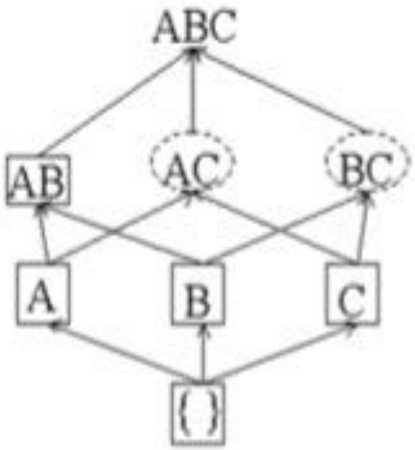
AB = 0, AC = 0, BC = 0

C changes to a square because its counter is greater than minsup.

Add counters for AC and BC because their subsets are all boxes.

Improving Efficiency of Apriori Algorithm:DIC

After 3M transactions read

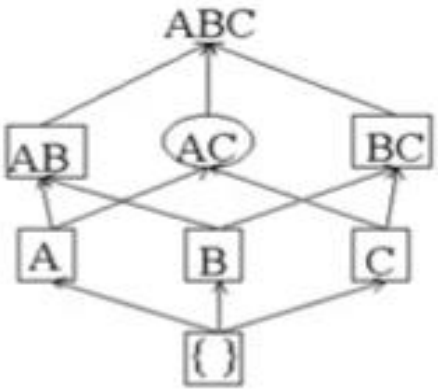


Trans.	A	B	C
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

Counters: A = 2, B = 2, C = 1,
AB = 1, AC = 0, BC = 0
AB has been counted all the way through and its counter satisfies minsup so we change it to a solid box. BC changes to a dashed box

https://www.youtube.com/watch?v=SLhLJZK6KaE

After 4M transactions read



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

AC and BC are counted all the way through. We do not count ABC because one of its subsets is a circle.
There are no dashed itemsets left so the algorithm is done.

Improving Efficiency of Apriori Algorithm: Sampling (mining on a subset of the given data)

- The fundamental idea of the sampling approach is to select a random sample S of the given data D, and then search for frequent itemsets in S rather than D.
- It may be possible to lose the global frequent itemset. This can be reduced by lowering the minimum support.
- The sample size of S is such that the search for frequent itemsets in S can be completed in main memory, and therefore only one scan of the transactions in S is needed 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

Applications Of Apriori Algorithm

1. **Improved store layout**
2. **Cross marketing**
3. **Focused attached mailings / add-on sales**
4. **Maintenance Agreement (What the store should do to boost Maintenance Agreement sales)**
5. **Home Electronics (What other products should the store stock up?)**
6. **Pharmacy to find frequent medicine itemsets**
7. For Student's Trends Analysis

Application of Apriori Algorithm in the Real World

1. Education Field

As we all know that all the students studying in the school have different characteristics and personalities like every student are different in age, gender, different names, and different parents' names, etc.

2. Medical Field

In every hospital, there are a lot of patients admitted over there every patient have a different kind of disease according to which they are given a different treatment, and they all will have a different type of characteristics and different medical history, so here it is necessary to use the computer science method of Apriori algorithm in order analyze the patients' database. So that there should be no mixing of the information of different patients

3. Forestry

In every forest, there are several kinds of flora and fauna. And, naturally, there are a variety of trees which are having different features like their sizes, texture, seasons of trees, etc. similarly, there are various types of animals having a variety of features, which makes them different from one another. So to analyze all such details, we use the method of the Apriori algorithm.

4. New Technology Business Firms

Apriori is used by many companies like Flipkart, Amazon, etc. where they have to maintain the record of various items of products that are purchased by various customers for recommender systems and by google for the autocomplete features.

5. Offices

One of the most efficient uses of this computer science technique is in the offices where they have to record a large number of day to day transactions related to sale and purchase of various good and services, like recording the transactions of creditors, sales and purchases so there is need of analysis of all such transactions so that there should not be any kind of confusion.

6. Mobile e-Commerce

The main purpose of using this technique is to make mobile e-commerce shopping easy, convenient, and increase the customer base crossing national and international borders. The deployment of the unity of real-time and recommendation accuracy is made so that mobile e-commerce gets the maximum benefit of the Apriori algorithm.

Applications Of Apriori Algorithm

Some fields where Apriori is used:

1. **In Education Field:** Extracting association rules in data mining of admitted students through characteristics and specialties.
2. **In the Medical field:** For example Analysis of the patient's database.
3. **In Forestry:** Analysis of probability and intensity of forest fire with the forest fire data.
4. Apriori is used by many companies like Amazon in the **Recommender System** and by Google for the auto-complete feature.

FP (Frequent Pattern Growth Algorithm)

- It is pattern growth approach for mining frequent itemsets.
- It's concept is basically based on divide-and-conquer strategy.
- First it compresses the frequent item database into frequent pattern tree(FP tree) which captures itemset association information.
- Set of conditional databases is obtained from FP tree. Conditional pattern base for each node represent various pattern fragment and we extract frequent pattern fragment or pattern fragment who satisfies minimum confidence criteria.
- Therefore, this approach substantially reduce the size of the data sets to be searched and also given out various frequent pattern segments

FP (Frequent Pattern Growth Algorithm)

Advantages of FP Tree

- Only 2 passes over data-set
- "Compresses" data-set
- No candidate generation
- Much faster than Apriori

Disadvantages of FP Tree

- FP-Tree may not fit in memory!!
- FP-Tree is expensive to build

Example 1

Build FP Tree and generate all frequent pattern. Min. Support is 60%

TID	List of Items
1	A, B, C, D, E, F
2.	B, C, D, E, F, G
3.	A, D, E, H
4.	A, D, F, I, J
5.	B, D, E, K

FP (Frequent Pattern Growth Algorithm)

Build FP Tree and generate all frequent pattern. Min. Support is 60%

TID	List of Items
1	A, B, C, D, E, F
2.	B, C, D, E, F, G
3.	A, D, E, H
4.	A, D, F, I, J
5.	B, D, E, K

Step 1: Calculate Minimum support .

Minimum support count (60/100 * 5) = 3

Step2: Find frequency of occurrence

Item	Frequency
A	3
B	3
C	2
D	5
E	4
F	3
G	1
H	1
I	1
J	1
K	1

FP (Frequent Pattern Growth Algorithm)

Step2: Find frequency of occurrence

Item	Frequency
A	3
B	3
C	2
D	5
E	4
F	3
G	1
H	1
I	1
J	1
K	1

Step 3: Filter out items which does not support Min. Support Count. Prioritize the items i. e Find Frequent itemsets

Item	Frequency	Priority
A	3	3
B	3	4
D	5	1
E	4	2
F	3	5

Frequent Itemsets
1: highest priority

FP (Frequent Pattern Growth Algorithm)

Step 4: Order the items according to priority i.e find the Ordered Itemsets

Here ,for each transaction, arrange the items in the descending order of their priority

TID	List of Items
1	A, B, C, D, E, F
2.	B, C, D, E, F, G
3.	A, D, E, H
4.	A, D, F, I, J
5.	B, D, E, K

Original Table for Reference

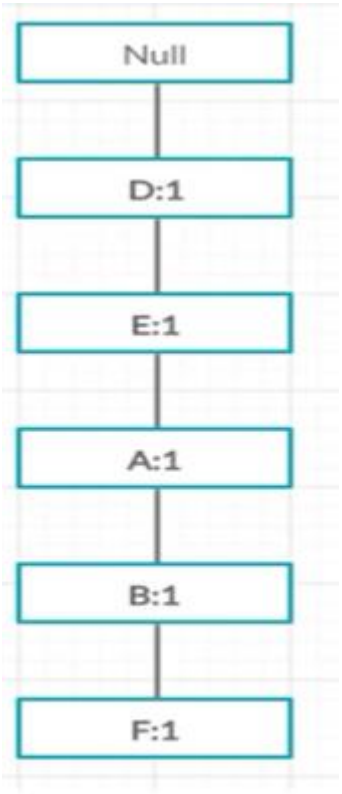
TID	List of Items
1	D, E, A, B, F
2.	D, E, B, F
3.	D, E, A
4.	D, A, F
5.	D, E, B

Ordered Itemsets

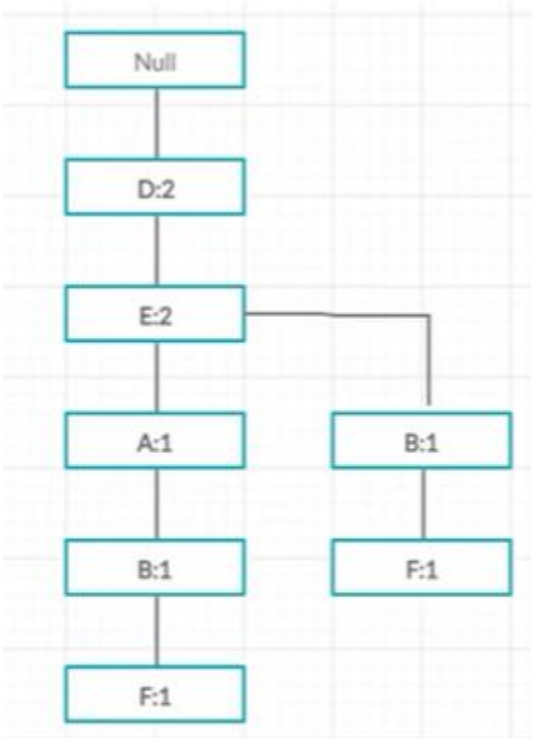
FP (Frequent Pattern Growth Algorithm)

Step 5: Build FP Tree Insert all ordered itemsets into a Trie data structure

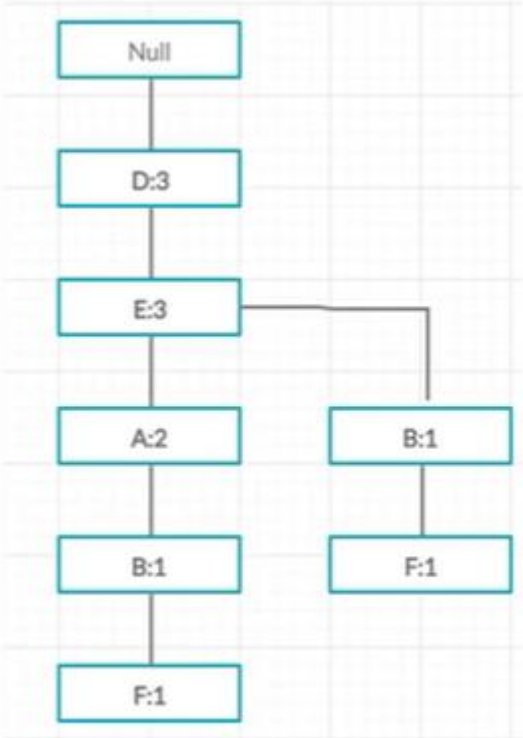
Start with NULL node and for every transaction's ordered Itemset, create a path.



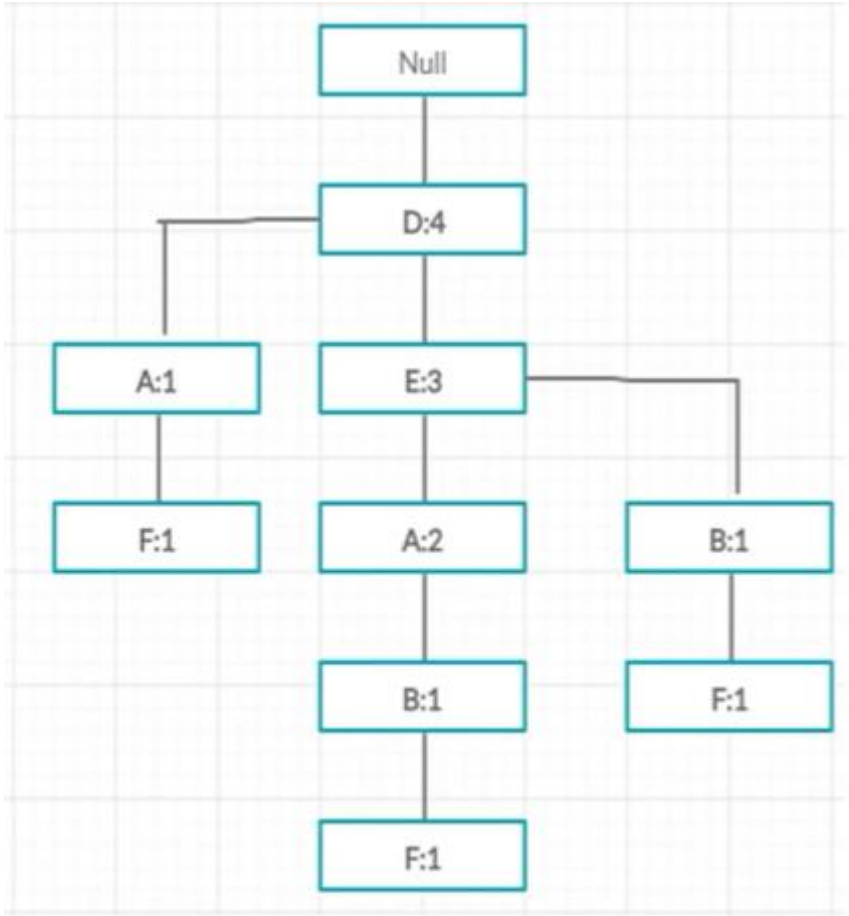
1. D, E, A, B, F



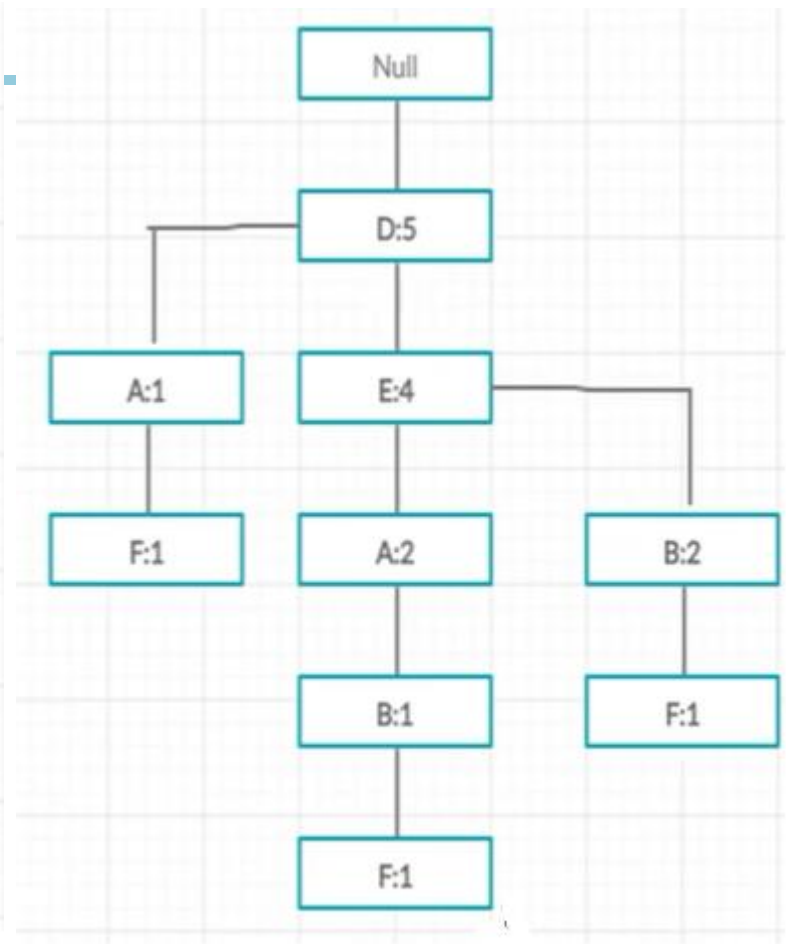
2. D, E, B, F



3. D, E, A

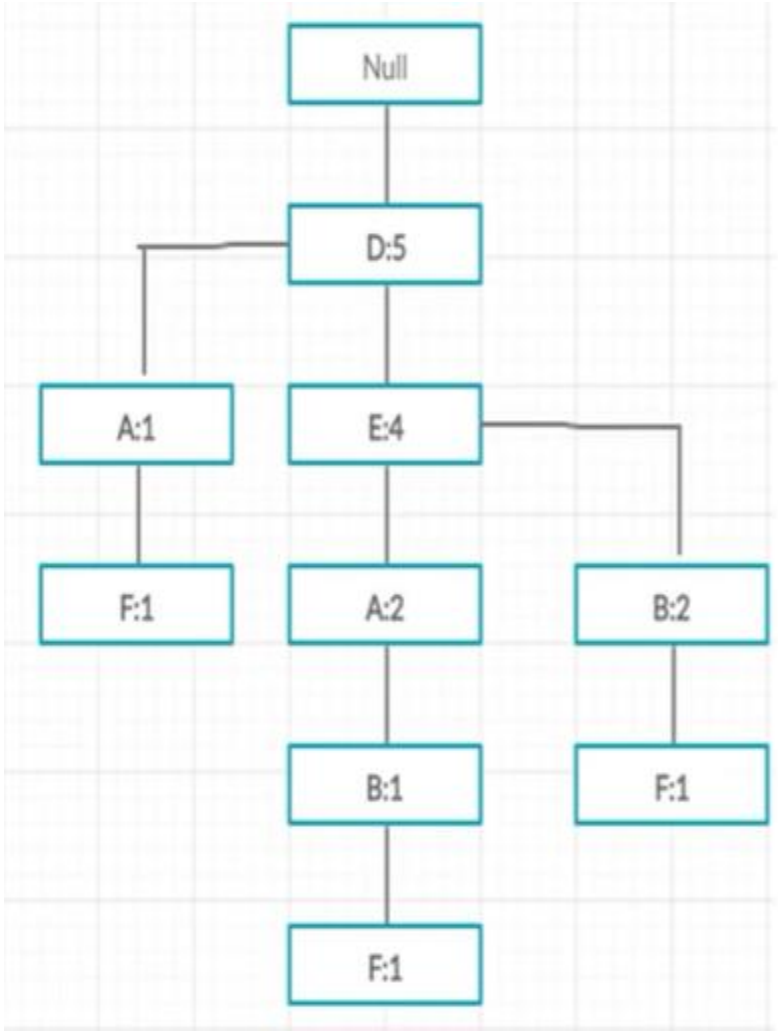


4. D, A, F



5. D, E, B

Step 6: Compute Conditional pattern base(CPB) for each item : It is apath labels of all the paths which lead to any node of the given item in the FP tree.



Item	Conditional Pattern Base(Path to reach item)
F	{{D,A:1},{D,E,A,B:1,{D,E,B:1}}
B	{{D,E,A:1},{D,E:2}}
A	{{D:1},{D,E:2}}
E	{D:4}
D	—

Step 7: Build Conditional FP Tree for each item.
For this, take the set of elements which is common in all the paths in the conditional pattern base for that item and calculate the support count by summing the support counts of all the paths in the conditional pattern base.

Item	Conditional Pattern Base(Path to reach item)	Conditional Frequent Pattern Tree
F	{{D,A:1},{D,E,A,B:1,{D,E,B:1}}	{D:3}
B	{{D,E,A:1},{D,E:2}}	{D,E:3}
A	{{D:1},{D,E:2}}	{D:3}
E	{D:4}	{D:4}
D	—	—-

Step 8: Generate Frequent Patterns

Generate frequent pattern rules by pairing the items of conditional frequent pattern tree set to the corresponding item as shown below.

Item	Conditional Pattern Base(Path to reach item)	Conditional Frequent Pattern Tree
F	{{D,A:1},{D,E,A,B:1,{D,E,B:1}}	{D:3}
B	{{D,E,A:1},{D,E:2}}	{D,E:3}
A	{{D:1},{D,E:2}}	{D:3}
E	{D:4}	{D:4}
D	—	—-

Item	Frequent Pattern Generated
F	{<D,F: 3>}
B	{<D,B:3>, <E,B:3>,<D,E,B:3>}
A	{<D,A:3>}
E	{<D,E:4>}
D	—

Step 8: Write frequent pattern Rules

The rules that satisfy minimum confidence threshold ae strong association rules .

Example 2

Build FP Tree and generate all frequent pattern. Min. Support is 30%

TID	List of Items
T1	E, A, D, B
T2	D, A, C, E, B
T3	C, A, B, E
T4	B, A, D
T5	D
T6	D, B
T7	A, D, E
T8	B, C

Step 1: Calculate Minimum support .

Minimum support count (30/100 * 8) = 2.4=3

Step2: Find frequency of occurrence

Itemset	Support Count
A	5
B	6
C	3
D	6
E	4

Step 3: Filter out items which does not support Min. Support Count. Prioritize the items

Itemset	Support Count	Priority
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4

Step 4: Order the items according to priority

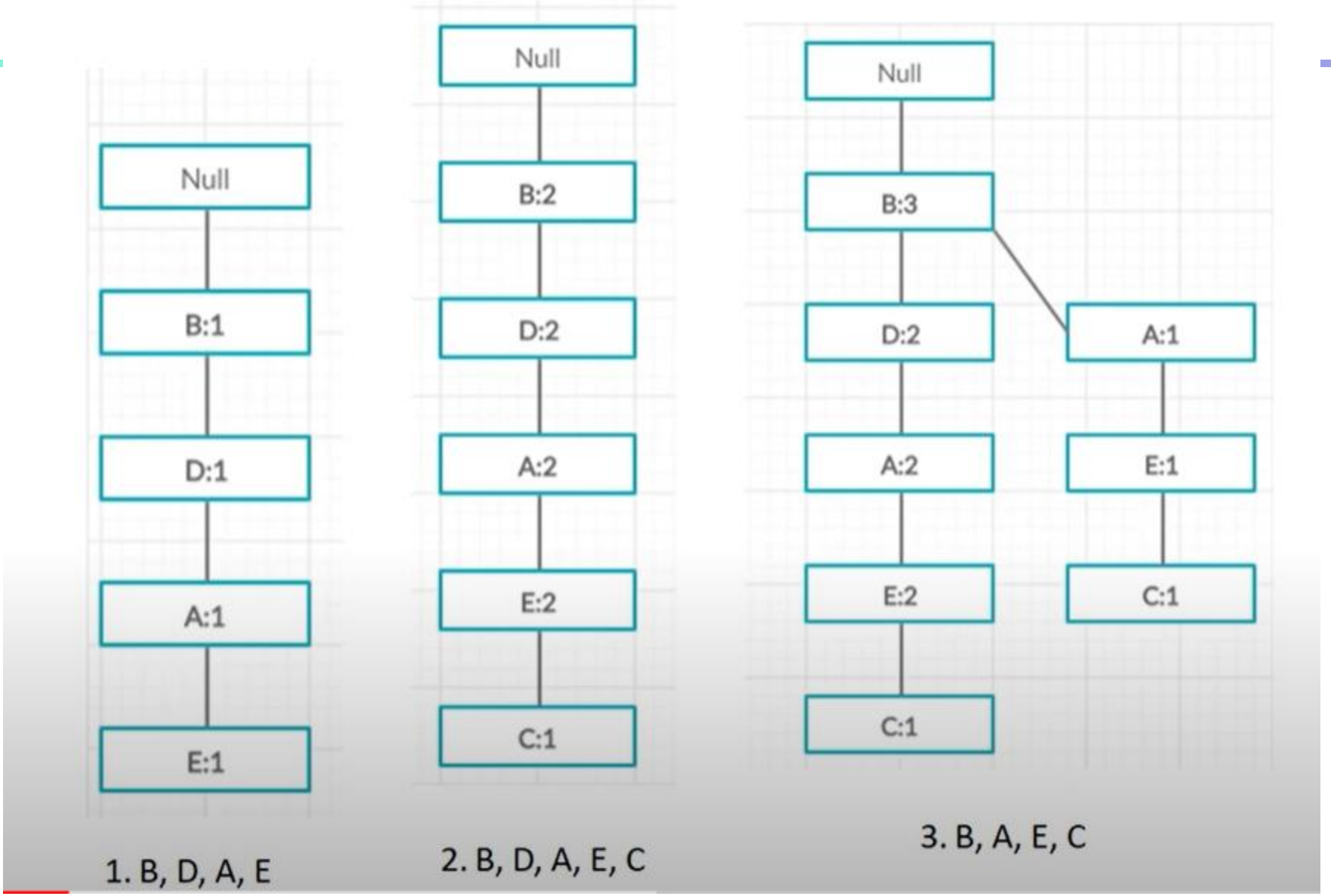
TID	List of Items
T1	E, A, D, B
T2	D, A, C, E, B
T3	C, A, B, E
T4	B, A, D
T5	D
T6	D, B
T7	A, D, E
T8	B, C

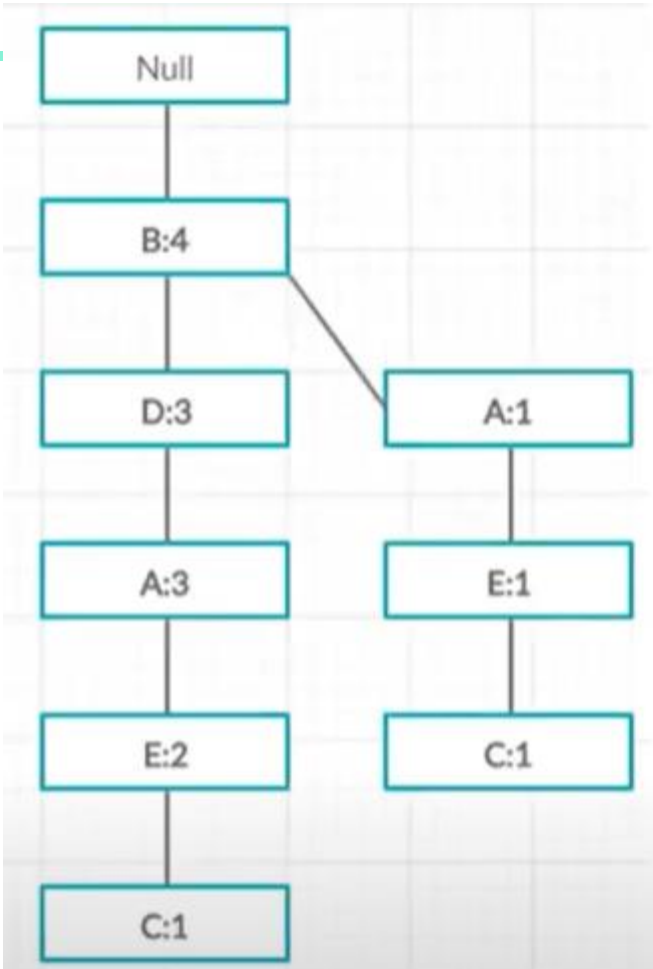
Original Table for Reference

TID	List of Items
T1	B, D, A, E
T2	B, D, A, E, C
T3	B, A, E, C
T4	B, D, A
T5	D
T6	B, D
T7	D, A, E
T8	B, C

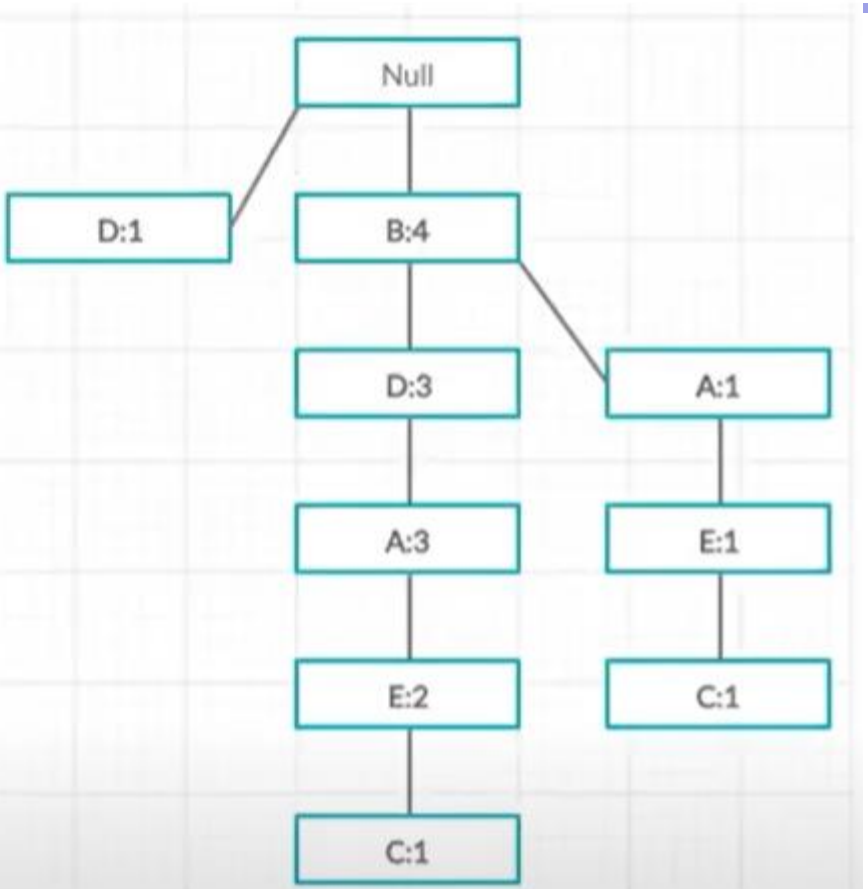
Ordered Itemsets

Step 5: Build FP Tree

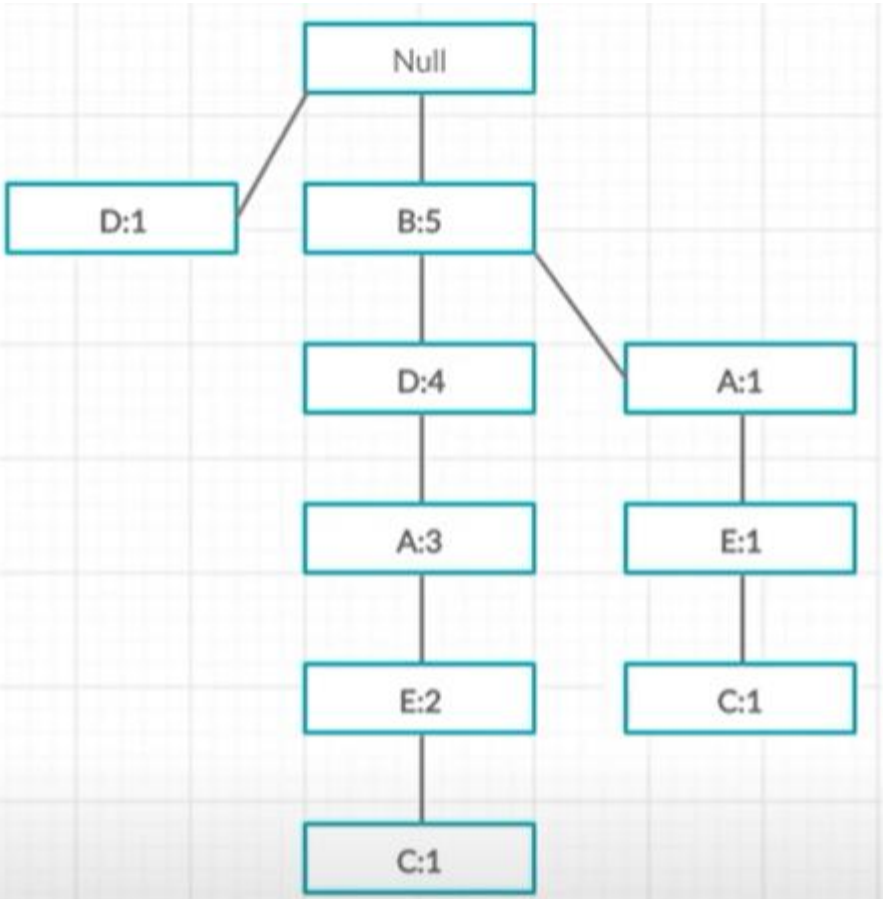




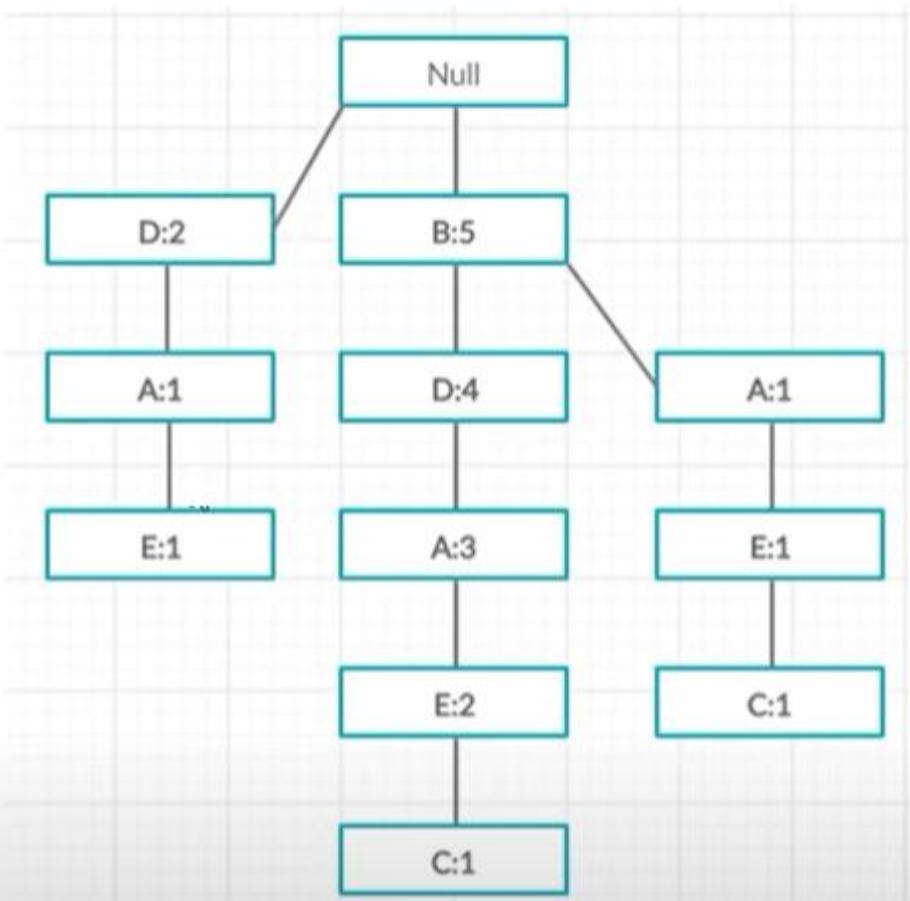
4. B, D, A



5. D

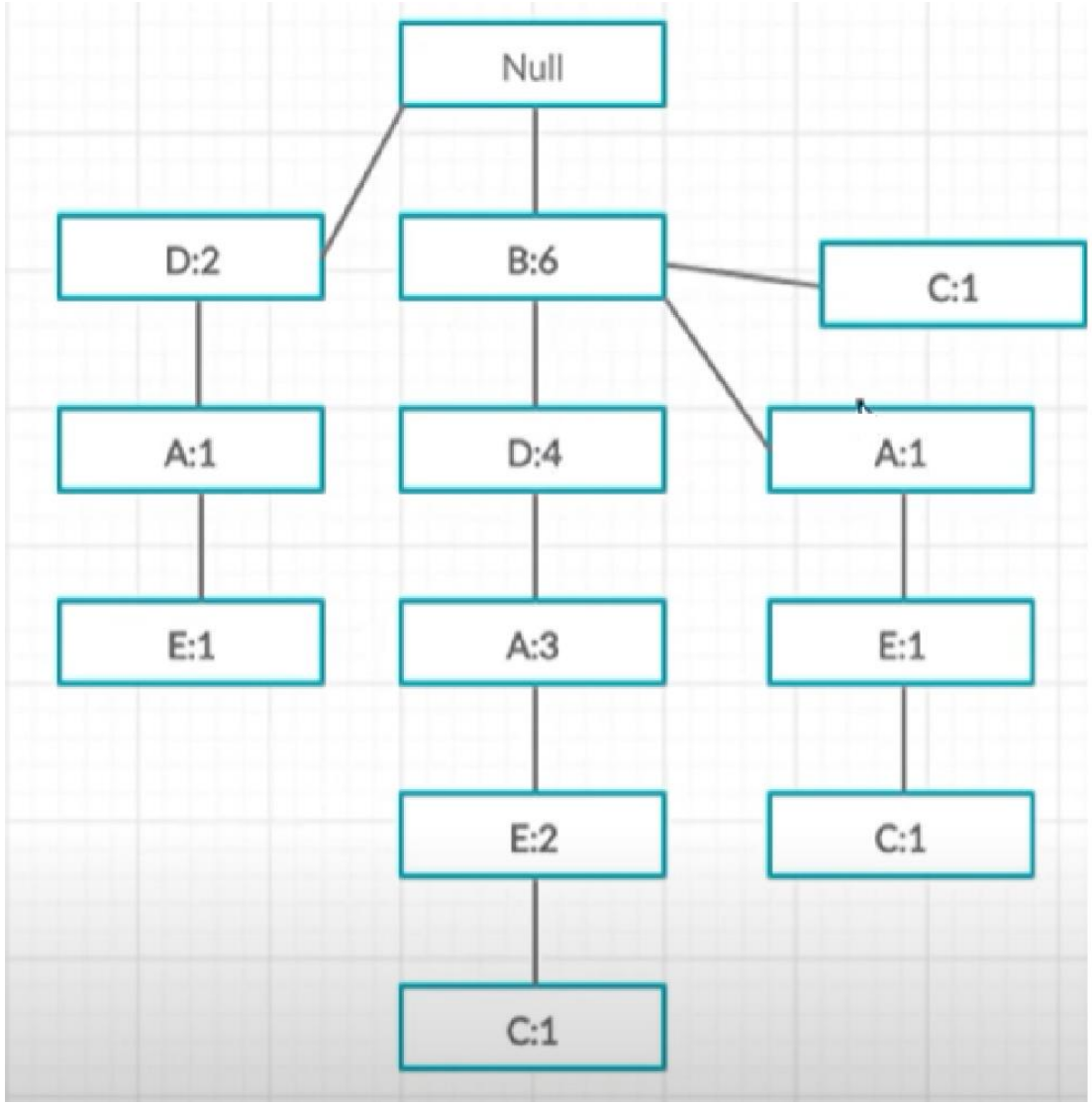


6. B, D

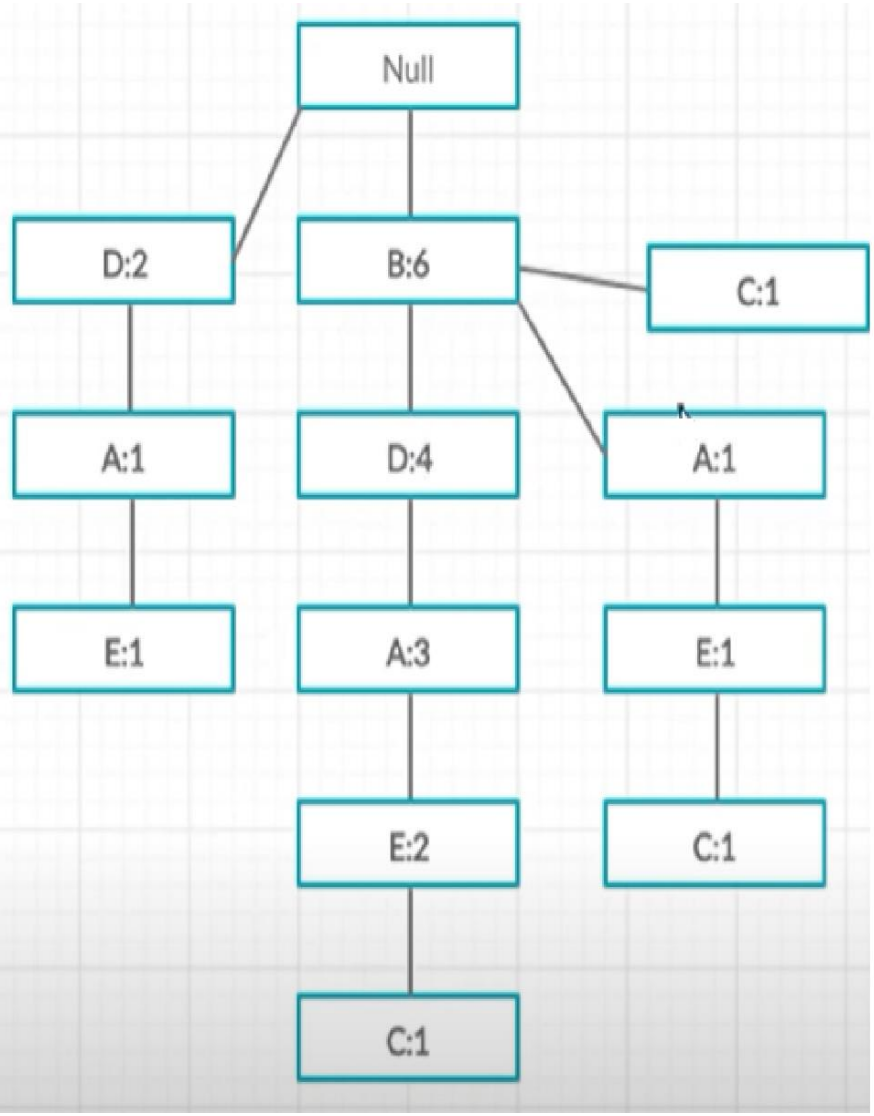


7. D, A, E

8. B, C



Step 6: Compute Conditional pattern base(CPB) for each item : It is apath labels of all the paths which lead to any node of the given item in the FP tree.



Item	Conditional Pattern Base(Path to reach item)
C	{{B,D,A,E:1},{B,A,,E:1},{B:1}}
E	{{D,A:1},{B,D,A:2},{B,A:1}}
A	{{D:1},{B,D:3},{B:1},}
D	{B:4}
B	—

Step 7: Build Conditional FP Tree for each item.
For this, take the set of elements which is common in all the paths in the conditional pattern base for that item and calculate the support count by summing the support counts of all the paths in the conditional pattern base.

Item	Conditional Pattern Base(Path to reach item)	Conditional Frequent Pattern Tree
C	{{B,D,A,E:1},{B,A,,E:1},{B:1}}	{B:3}
E	{{D,A:1},{B,D,A:2},{B,A:1}}	{A:4}
A	{{D:1},{B,D:3},{B:1},}	—
D	{B:4}	{B:4}
B	—	—-

Step 8: Generate Frequent Patterns

Generate frequent pattern rules by pairing the items of conditional frequent pattern tree set to the corresponding item as shown below.

Item	Conditional Pattern Base(Path to reach item)	Conditional Frequent Pattern Tree
C	{{B,D,A,E:1},{B,A,,E:1},{B:1}}	{B:3}
E	{{D,A:1},{B,D,A:2},{B,A:1}}	{A:4}
A	{{D:1},{B,D:3},{B:1},}	—
D	{B:4}	{B:4}
B	—	—-

Item	Frequent Pattern Generated
C	{<B,C: 3>}
E	{<A,E:4>
A	—-
D	{<B:4>}
B	—

Step 8: Write frequent pattern Rules

The rules that satisfy minimum confidence threshold ae strong association rules .

Example 3

https://www.youtube.com/watch?v=kK6yRznGTdo

ECLAT(Equivalence Class Transformation)
Vertical Apriori

- Both Apriori and FP-growth use horizontal data format .
- ECLAT mines frequent itemsets using the Vertical Data Format.
- It is a depth first search based algorithm.
- In this, each item is stored together with its T_ID (Transaction ID).
- It uses intersection based approach to compute the support an itemset.

ECLAT-Example 1

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=2, Confidence =70% Generate Association Rule using ECLAT Algorithm

Database is in Horizontal Data Format.

Step 2: Itemset generated by intersection of 1 itemset

Itemset	List of Items
I1	T1, T4, T5, T7, T8, T9
I2	T1, T2, T3, T4, T6, T8, T9
I3	T3, T5, T6, T7, T8, T9
I4	T2, T4
I5	T1, T8

Itemset	List of Items
I1, I2	T1,T4,T8,T9
I1, I3	T5,T7, T8, T9
I1, I4	T4
I1, I5	T1,T8
I2, I3	T3, T6, T8, T9
I2, I4	T2, T4
I2, I5	T1, T8
I3, I4	--
I3, I5	T8
I4, I5	--

Itemset	List of Items
I1, I2	T1,T4,T8,T9
I1, I3	T5,T7, T8, T9
I1, I4	T4
I1, I5	T1,T8
I2, I3	T3, T6, T8, T9
I2, I4	T2, T4
I2, I5	T1, T8
I3, I4	--
I3, I5	T8
I4, I5	--

Itemset	List of Items
I1, I2	T1,T4,T8,T9
I1, I3	T5,T7, T8, T9
I1, I5	T1,T8
I2, I3	T3, T6, T8, T9
I2, I4	T2, T4
I2, I5	T1, T8

Min. Support=2

Step 3: Itemset generated by intersection of 2 itemset

Itemset	List of Items
I1, I2	T1,T4,T8,T9
I1, I3	T5,T7, T8, T9
I1, I5	T1,T8
I2, I3	T3, T6, T8, T9
I2, I4	T2, T4
I2, I5	T1, T8

Itemset	List of Items
I1, I2,I3	T8, T9
I1, I2, I5	T1, T8
I1, I3, I5	T8
I2, I3, I4	--
I2, I3, I5	T8
I2, I4, I5	--

Itemset	List of Items
I1, I2,I3	T8, T9
I1, I2, I5	T1, T8

Step 3: Itemset generated by intersection of 3 itemset

Itemset	List of Items
I1, I2,I3	T8, T9
I1, I2, I5	T1, T8

Itemset	List of Items
I1, I2, I3, I5	T8

Table with four itemset is null

Rules can be formed from following three itemset

Itemset	List of Items
I1, I2,I3	T8, T9
I1, I2, I5	T1, T8

We can expand any rule.
For e.g. I1, I2 and I5

Confidence =70%

Association Rule	Confidence	Confidence (%)
I1^I2→I5	C(I1,I2,I5)/C(I1,I2)=2/4	50%
I1^I5→I2	C(I1,I2,I5)/C(I1,I5)=2/2	100%
I2^I5→I1	C(I1,I2,I5)/C(I2,I5)=2/2	100%
I1→I2^I5	C(I1,I2,I5)/C(I1)=2/6	33%
I2→I1^I5	C(I1,I2,I5)/C(I2)=2/7	29%
I5→I1^I2	C(I1,I2,I5)/C(I5)=2/2	100%

ECLAT - Example 2

https://www.youtube.com/watch?v=P5LH5HCrhMU

Advantages of ECLAT:

- It uses less memory than Apriori as its concept is based on depth first search.
- It requires less time for frequent pattern generation than Apriori. Because there is no repeated scanning of the data to compute individual support values.

Disadvantages of ECLAT:

- Suitable for small dataset.
- When T_ID list is large, then it takes more time to store candidate set. Also it requires more time for intersection.

Single Dimensional Association Rules

1) Single dimensional or Intra Dimensional Association Rule

It contains a single distinct predicate (e.g. purchase) with its multiple occurrence

For e.g. purchase(X, "Milk")→ purchase(X, "Bread")

purchase(X, "Milk") ^ purchase(X, "Butter") →purchase(X, "Bread")

Multidimensional Association Rules

2) Multi dimensional or Inter Dimensional Association Rule

It contains two or more predicate. Each predicate occurs only once.

e.g.

1) Student(X, "Yes") ^ Credit Rating (X, "Excellent") → buys_Laptop(X, "Yes")

2) Student(X, "No") ^ Credit Rating (X, "Fair") → buys_Laptop(X, "No")

3) Student(X, "Yes") ^ Credit Rating (X, "Fair") → buys_Laptop(X, "No")

4) Student(X, "No") ^ Credit Rating (X, "Excellent") → buys_Laptop(X, "Yes")

Hybrid Association Rules

3) Hybrid dimensional Association Rule

Like multi dimensional, association rules with two or more predicate. Unlike multidimensional, some predicate are repeated.

No.	Student	Credit Rating	Buys
1	Yes	Excellent	Desktop, Laptop
2	Yes	Fair	Desktop
3	No	Excellent	Laptop, Printer
4	No	Excellent	Laptop, Printer
5	Yes	Fair	Desktop

Student(X, "Yes") ^ Credit Rating (X, "Excellent") ^Buys (X, "Desktop")→Buys(X, "Laptop")

Student(X, "No") ^ Credit Rating (X, "Excellent") ^Buys (X, "Laptop")→Buys(X, "Printer")