

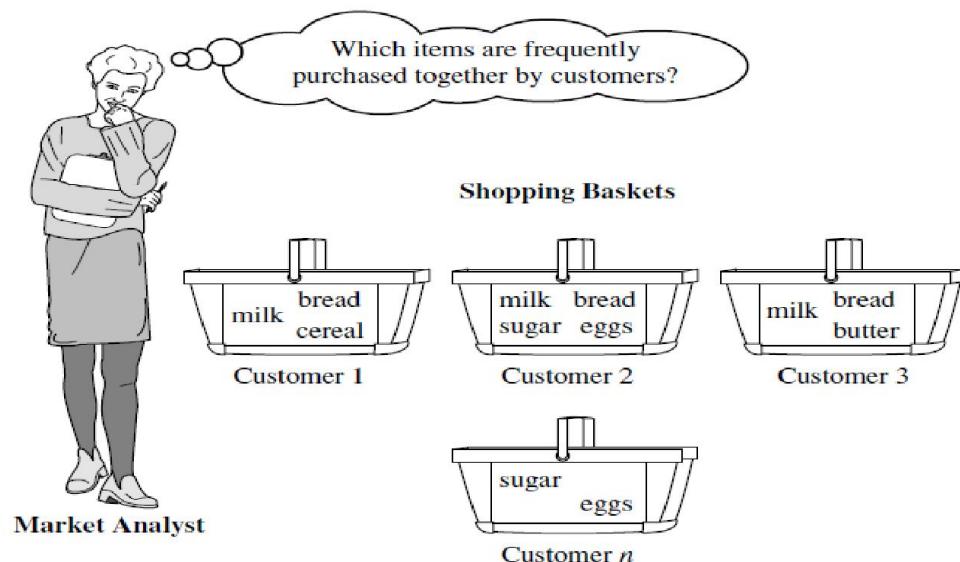
# Data Mining: Concepts and Techniques

---

Mining Frequent Patterns, Association and  
Correlations

# 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



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.

Market basket analysis.

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

**Itemset:** A collection of one or more items.

e.g. {Milk, Bread, Butter}

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

e.g.  $\sigma\{\text{Milk, Bread}\}=3$

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

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

**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

<i>Customers</i>	<i>Transactions</i>
1	milk, bread
2	bread, butter
3	beer
4	milk, bread, butter
5	bread
6	milk, bread, butter

# Support and confidence

## Association Rule

An implication expression of the form  $X \rightarrow Y$  where X and Y are itemsets

E.g. {Milk, Bread}  $\rightarrow$  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}  $\rightarrow$  Butter

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

<i>Customers</i>	<i>Transactions</i>
1	milk, bread
2	bread, butter
3	beer
4	milk, bread, butter
5	bread
6	milk, bread, butter

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

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

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

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

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

# 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.

# Apriori Algorithm

---

- Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.
- It uses **prior knowledge** of frequent itemset properties (Aprior).
- It uses K frequent itemsets to find K+1 itemsets.
- It is based on three concept: **Frequent itemset, Apriori property and join operations**

# 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

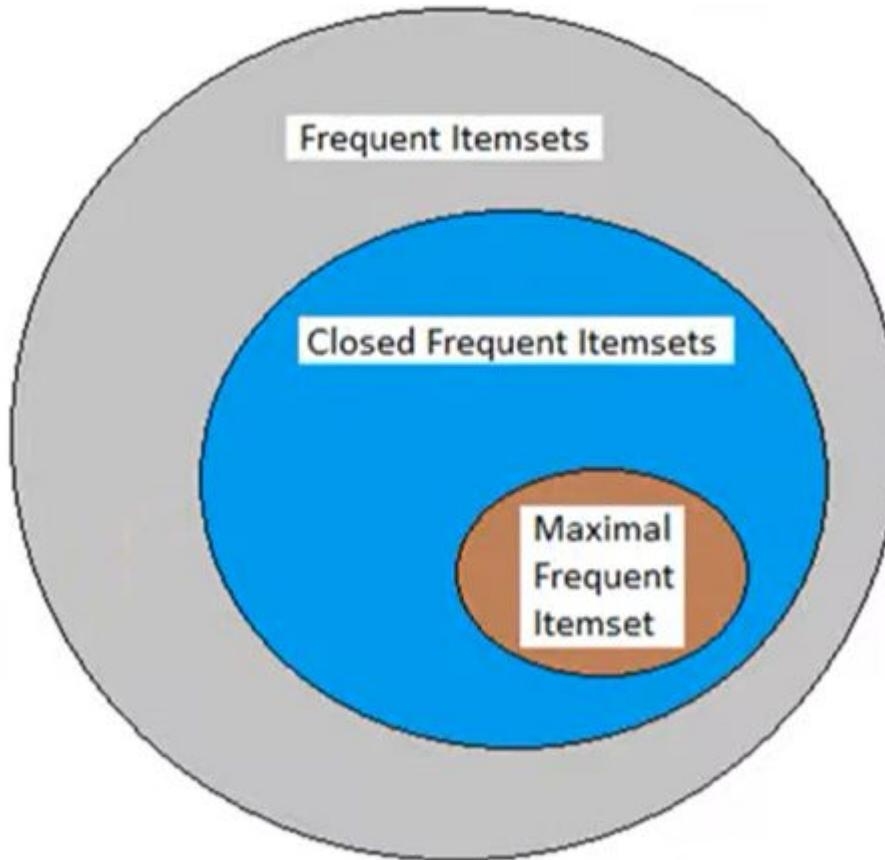
Milk, Butter

Bread, Butter

# Need of Closed and Maximal Itemset

---

- These closed frequent itemset and maximal frequent itemset are useful when **huge amount of data is used in association rule mining.**
- If the length of a frequent itemset is ‘k’, then by downward closure property, **all of its  $2^k$  subsets are also frequent.**
- When the computation is very expensive and there is no interest to find additional subsets. This can be avoided by frequent itemset with maximum length.
- One disadvantage with maximal frequent itemsets is that even all its subsets are frequent, we do not know their supports. For mining rules, support information is very important .
- Therefore closed frequent itemset is preferred.



# 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 item sets are frequent, except AD

$A(3) \rightarrow AB(3), AC(3), AD(2)$

$A(\text{count})$  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**

$B(4) \rightarrow AB(3), BC(4), BD(3)$

$B(\text{count})$  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?

$$C(5) \rightarrow AC(3), BC(4), CD(2)$$

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?

$$D(4) \rightarrow AD(2), BD(3), CD(4)$$

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	Is AB closed,Maximal?
	$AB(3) \rightarrow ABC(3), ABD(2)$
A, B,C	AB(count) is not greater than its immediate superset. <b>AB is not closed.</b>
A, B,D	In AB's immediate superset, itemset are present with min. support count i.e. 3.
A, C,D	$AB(3) \rightarrow ABC(3), ACD(2)$ AC(count) is not greater than its immediate superset. <b>AC is not closed.</b>
B,C,D	In AC 's immediate superset, itemset are present with min. support count i.e. 3. <b>AC is not maximal</b>
Is AC closed,Maximal?	$AC(3) \rightarrow ABC(3), ACD(2)$ AC(count) is not greater than its immediate superset. <b>AC is not closed.</b>
Is AD frequent?	$AD(2) \rightarrow ABD(2), ADC(2)$ <b>AD Not frequent</b>

# 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

$BC(4) \rightarrow ABC(3), BCD(3)$

$BC(\text{count})$  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**

$BD(3) \rightarrow ABD(2), BCD(3)$

$BD(\text{count})$  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**

$CD(4) \rightarrow ACD(2), BCD(3)$

$CD(\text{count})$  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?

$$ABC(3) \rightarrow ABCD(2)$$

$ABC$ (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

Is BCD closed,Maximal?

$$BCD(3) \rightarrow ABCD(2)$$

$BCD$ (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 item sets are not frequent, except BP

$B(4) \rightarrow BE(1), BM(1), BP(3)$

$B(\text{count})$  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**

$E(2) \rightarrow BE(1), EM(1), EP(0)$

$E(\text{count})$  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

$M(2) \rightarrow BM(1), EM(1), MP(1)$

$M(\text{count})$  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**

$P(3) \rightarrow BP(3), EP(0), MP(1)$

Three itemset is empty.

$E(\text{count})$  is not greater than its immediate superset.

**BP is closed and maximal**

**P is not closed.**

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

**P is not 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

---

- Algorithm for mining frequent itemsets for Boolean association rules.
- 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, the Apriori property (Any subset of frequent itemset must be frequent) is used to reduce the search space.

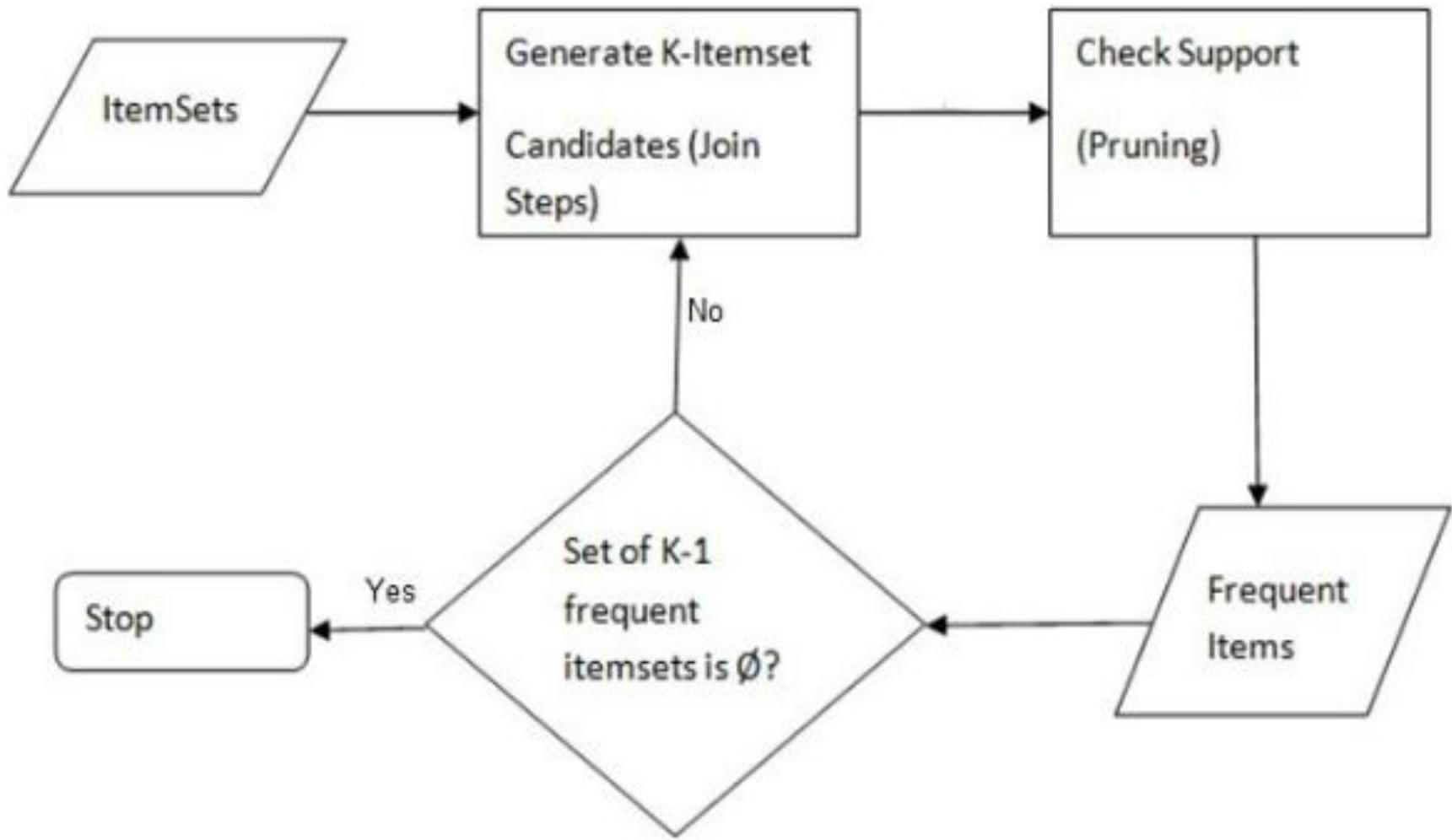
# **"How is the Apriori property used in the algorithm?"**

---

How L<sub>k-1</sub> is used to find L<sub>k</sub> for k= 2?

1. **Join Step:** To find L<sub>k</sub>, a set of candidate k-itemsets is generated by joining L<sub>k-1</sub> with itself. This set of candidates is denoted as C<sub>k</sub>.
    - The join, L<sub>k-1</sub>  $\bowtie$  L<sub>k-1</sub>, is performed, **where members of L<sub>k-1</sub> 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



# 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 Apriori Algorithm

---

- Pseudo-code:

$C_k$ : Candidate itemset of size k

$L_k$ : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **do begin**

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

**for each** transaction  $t$  in database do

increment the count of all candidates in  $C_{k+1}$

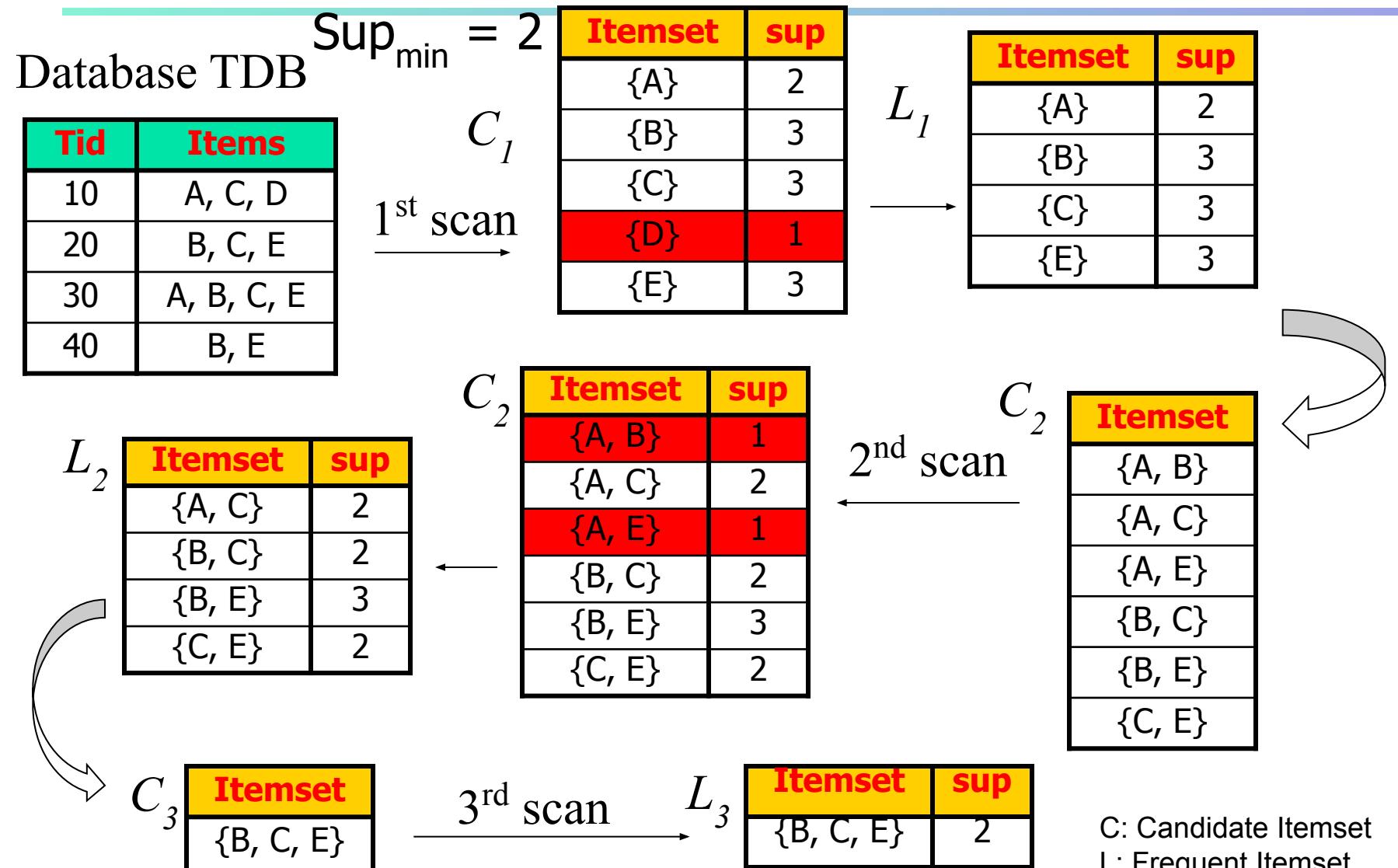
that are contained in  $t$

$L_{k+1}$  = 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**

**C: candidate itemset**

**L: Frequent itemset**

**L1**

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

**Compare candidate support count  
with minimum support count**

# 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

# 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 candidate k-itemsets is generated by joining  $L_{k-1}$  with itself.

Generate candidate set  $C_3$  ( $k=3$ ) using  $L_2$  (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  $L_2$ , first element should match.

**Prune Operation:** Select only frequent itemset.

$$C_3 = L_2 \bowtie L_2$$

**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

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

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

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

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

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

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

# 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

L3

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

Compare candidate sup. count  
with min. Sup. count

Generate C3 from L2 and scan

D for count of each candidate

# 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

L4



Compare candidate sup. count  
with min. Sup. count

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

Expand Rule from L3 as L4 is  $\emptyset$

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 (%)
$I1 \wedge I2 \rightarrow I5$	$C(I1, I2, I5) / C(I1, I2) = 2/4$	50%
$I1 \wedge I5 \rightarrow I2$	$C(I1, I2, I5) / C(I1, I5) = 2/2$	100%
$I2 \wedge I5 \rightarrow I1$	$C(I1, I2, I5) / C(I2, I5) = 2/2$	100%
$I1 \rightarrow I2 \wedge I5$	$C(I1, I2, I5) / C(I1) = 2/6$	33%
$I2 \rightarrow I1 \wedge I5$	$C(I1, I2, I5) / C(I2) = 2/7$	29%
$I5 \rightarrow I1 \wedge I2$	$C(I1, I2, I5) / C(I5) = 2/2$	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

L1

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

Scan D for count of each candidate

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

L2

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

Compare candidate sup. count with min. Sup. count

Generate C2 from L1 and scan D

# Exercise

---

**Join Operation:** To find  $L_k$ , a set of candidate k-itemsets is generated by joining  $L_{k-1}$  with itself.

Generate candidate set  $C_3$  ( $k=3$ ) using  $L_2$  (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  $L_2$ , first element should match.

**Prune Operation:** Select only frequent itemset.

**L2**

Itemset	Sup. Count
Buns, Hotdogs	2
<b>Chips, Coke</b>	3
<b>Chips, Hotdogs</b>	2
Coke, Hotdogs	2

$$C_3 = L_2 \bowtie L_2$$

$$\begin{aligned} & (Chips, Coke) \bowtie (Chips, Hotdogs) \\ & = (Chips, Coke, Hotdogs) \end{aligned}$$

# 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	$C(\text{Chips, Coke, Hotdogs})/C(\text{Chips, Coke})=2/3$	67%
Chips ^ Hotdogs → Coke	$C(\text{Chips, Coke, Hotdogs})/C(\text{Chips, Hotdogs})=2/2$	100%
Coke ^ Hotdogs → Chips	$C(\text{Chips, Coke, Hotdogs})/C(\text{Coke, Hotdogs})=2/2$	100%
Chips → Coke ^ Hotdogs	$C(\text{Chips, Coke, Hotdogs})/C(\text{Chips})=2/4$	50%
Coke → Chips ^ Hotdogs	$C(\text{Chips, Coke, Hotdogs})/C(\text{Coke})=2/3$	67%
Hotdogs → Chips ^ Coke	$C(\text{Chips, Coke, Hotdogs})/C(\text{Hotdogs})=2/4$	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**

# 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

# 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

# Exercise 2

**Join Operation:** To find  $L_k$ , a set of candidate k-itemsets is generated by joining  $L_{k-1}$  with itself.

Generate candidate set  $C_3$  ( $k=3$ ) using  $L_2$  (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  $L_2$ , 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

$$\begin{aligned}C_3 &= L_2 \bowtie L_2 \\(A, B) \bowtie (A, D) &= (A, B, D) \\(A, B) \bowtie (A, E) &= (A, B, E) \\(A, D) \bowtie (A, E) &= (A, D, E) \\(B, D) \bowtie (B, E) &= (B, D, E)\end{aligned}$$

# 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

L3

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

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

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

L4

$\emptyset$

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

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 (%)
$A \wedge B \rightarrow D$	$C(A, B, D)/C(A, B)=3/4$	75%
$A \wedge D \rightarrow B$	$C(A, B, D)/C(A, D)=3/4$	75%
$B \wedge D \rightarrow A$	$C(A, B, D)/C(B, D)=3/4$	75%
$A \rightarrow B \wedge D$	$C(A, B, D)/C(A)=3/5$	60%
$B \rightarrow A \wedge D$	$C(A, B, D)/C(B)=3/6$	50%
$D \rightarrow A \wedge B$	$C(A, B, D)/C(\text{Hotdogs})=3/6$	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

---

## **Advantages :**

- Easy to understand and implement.
- Can be easily parallelized.
- Uses large itemset property.

## **Disadvantages:**

- Requires many database scans.
- Assumes transaction database is memory resident.

# Pros and Cons of Apriori Algorithm

---

## ■ Pros:

- This is the most simple and easy-to-understand algorithm among association rule learning algorithms
- The resulting rules are intuitive and easy to communicate to an end user
- It doesn't require labeled data as it is fully unsupervised; as a result, you can use it in many different situations because unlabeled data is often more accessible
- Many extensions were proposed for different use cases based on this implementation—for example, there are association learning algorithms that take into account the ordering of items, their number, and associated timestamps
- The algorithm is exhaustive, so it finds all the rules with the specified support and confidence

# Pros and Cons of Apriori Algorithm

---

## ❑ Cons:

- ❑ It requires a significant amount of calculations if the itemsets are extremely big and the minimal support is maintained to a bare minimum.
- ❑ A full scan of the whole database is required.

# Improving Efficiency of Apriori Algorithm

---

## **Apriori Algorithm:**

Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

It uses **prior knowledge** of frequent itemset properties (Aprior).

It uses K frequent Itemset to find K+1 itemsets.

It is based on three concept: **Frequent itemset, Apriori property and join operations**

## **Disadvantages:**

- Requires many database scans.  
    ↳
- Assumes transaction database is memory resident.

# 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

---

## a) Hash Based Techniques

- Hash table is used as data structure.
- First iteration is required to count support of each itemset.
- From second iteration, efforts are made to enhance execution of Apriori by utilizing hash table concept.
- Hash table minimizes the number of itemset generated in second iteration.
- In second iteration i.e. 2-itemset generation, for every combination of two item, we map them into the diverse bucket of hash table structure and increment the bucket count.
- If count of bucket is less than min. sup. count, we remove them from candidate sets.

# 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

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

$$H(x, y) = ((\text{Order of First}) * 10 + (\text{Order of Second})) \bmod 7$$

# 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	{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

## 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

L1

	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

C2

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

L2

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

L3

	I2,I3, I4
T2	1
T4	1

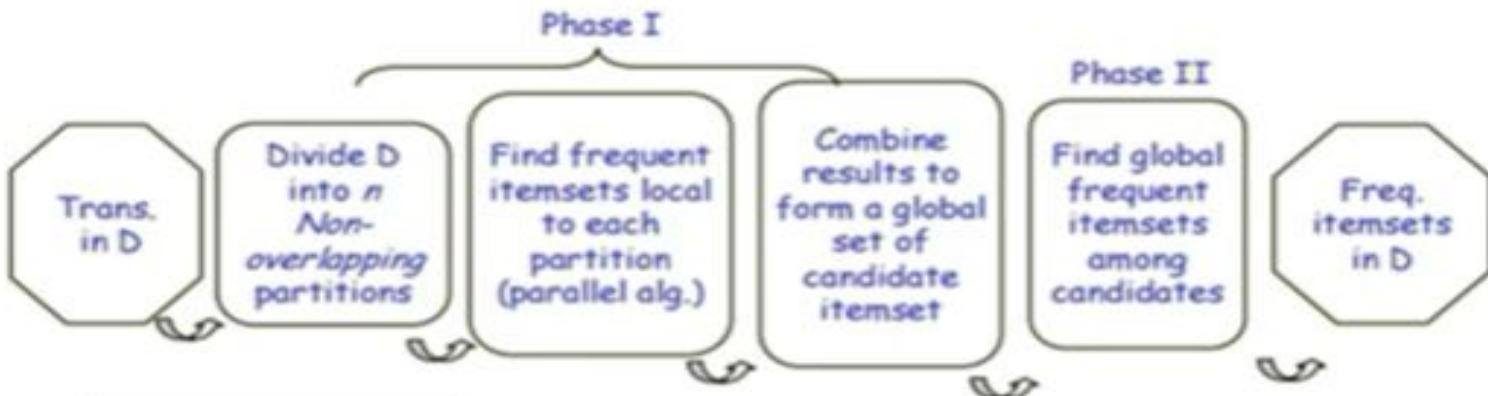
X

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

# 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

# Improving Efficiency of Apriori Algorithm: DIC

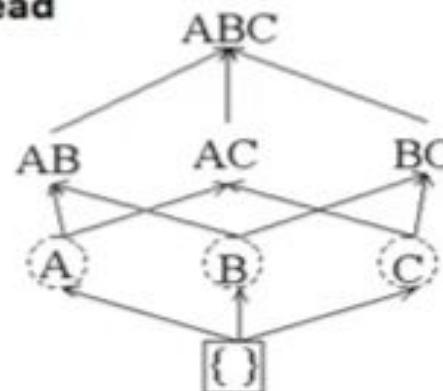
**Solid box:**  Confirmed frequent itemset - an itemset we have finished counting and exceeds the support threshold  $\text{minsupp}$

**Solid circle:**  Confirmed infrequent itemset - we have finished counting and it is below  $\text{minsupp}$

**Dashed box:**  suspected frequent itemset - an itemset we are still counting that exceeds  $\text{minsupp}$

**Dashed circle:**  suspected infrequent itemset - an itemset we are still counting that is below  $\text{minsupp}$

Itemset lattice before  
any transactions are read



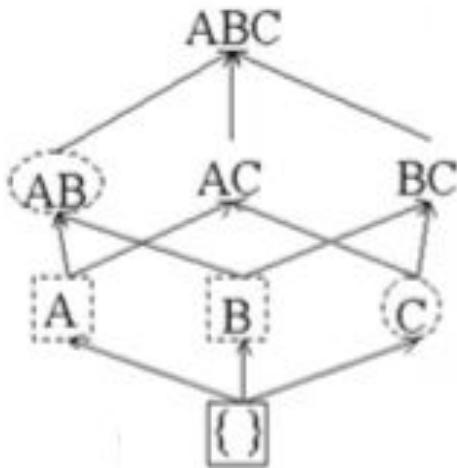
Empty itemset is marked with a **solid box**.

All 1-itemsets are marked with **dashed circles**.

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

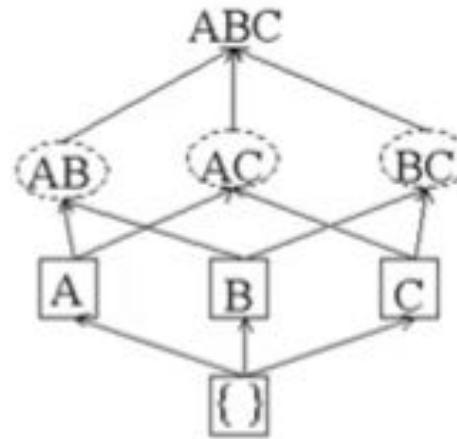
# Improving Efficiency of Apriori Algorithm: DIC

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

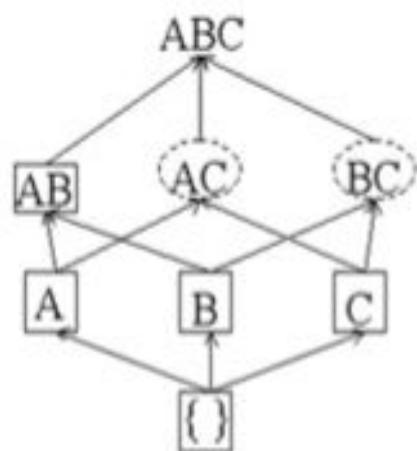
$AB = 0$ ,  $AC = 0$ ,  $BC = 0$

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

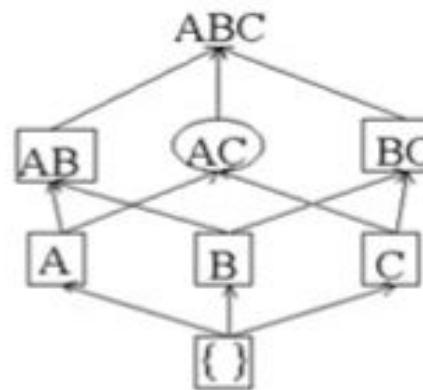


Trans.	A	B	C
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

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

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.

## e. Sampling

---

- 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.
- In this method, it can trade off some degree of accuracy against efficiency.

## **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

---

## **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

# **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 Algorithm)

---

- It is pattern growth approach for mining frequent itemsets.
- Its 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 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

# FP (Frequent Pattern 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.

$$\text{Minimum support count } (60/100 * 5) = 3$$

# FP (Frequent Pattern 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 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

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

# FP (Frequent Pattern Algorithm)

---

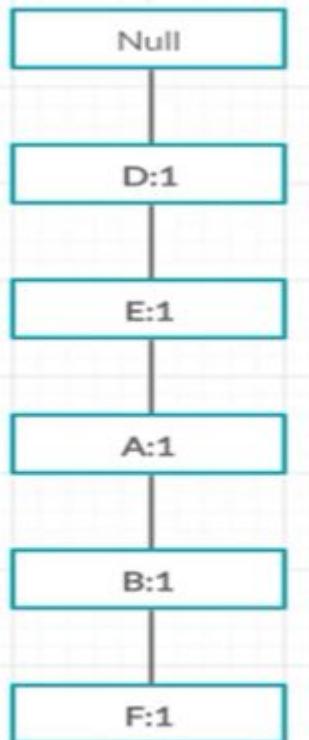
**Step 4:** Order the items according to priority

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

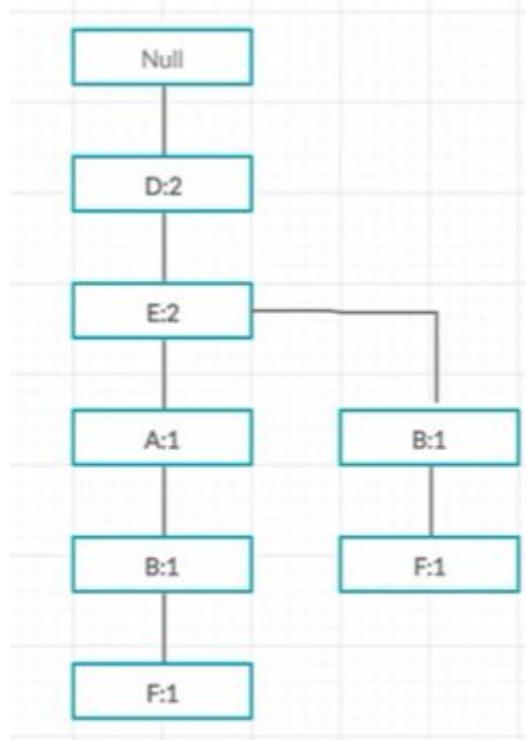
TID	List of Items
1	D, E, A, B, F
2.	D, E, B, F
3.	D, E, A
4.	D, A, F
5.	D, E, B

Original Table for Reference

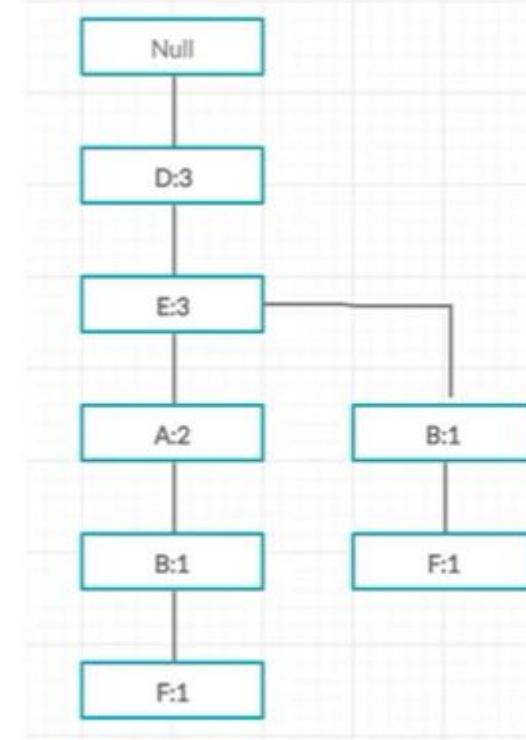
## Step 5: Build FP Tree



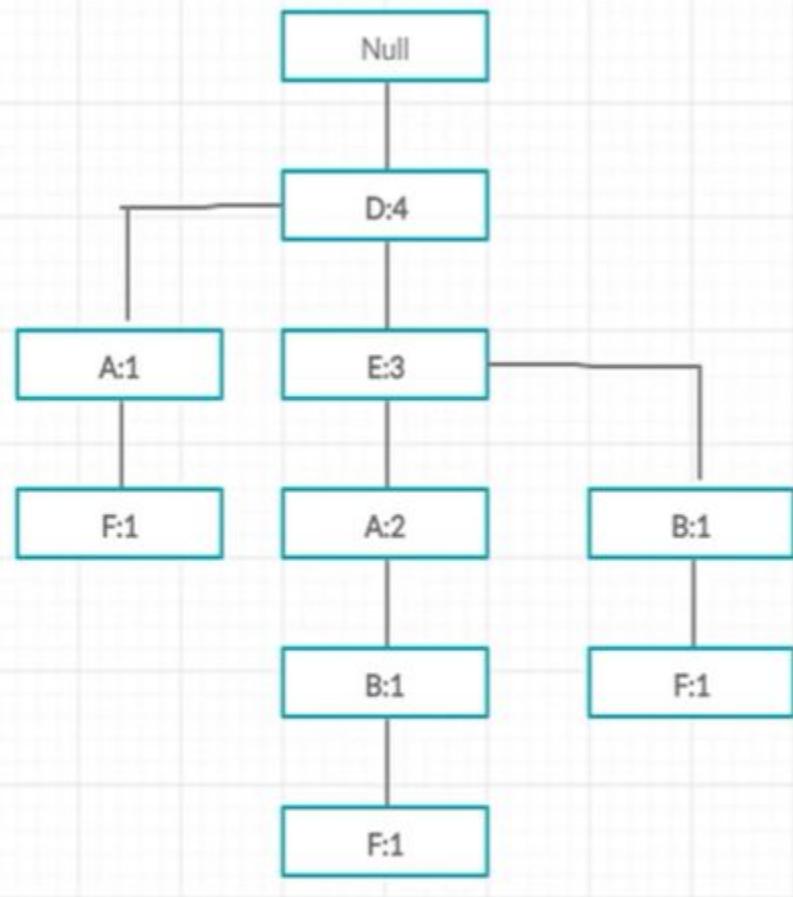
1. D, E, A, B, F



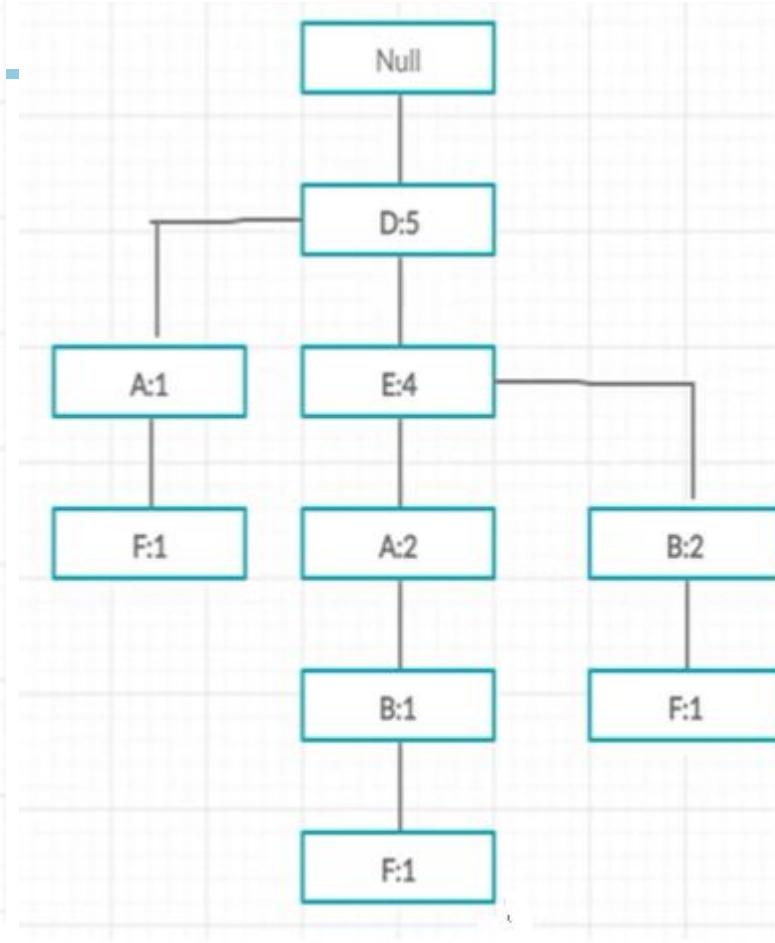
2. D, E, B, F



3. D, E, A



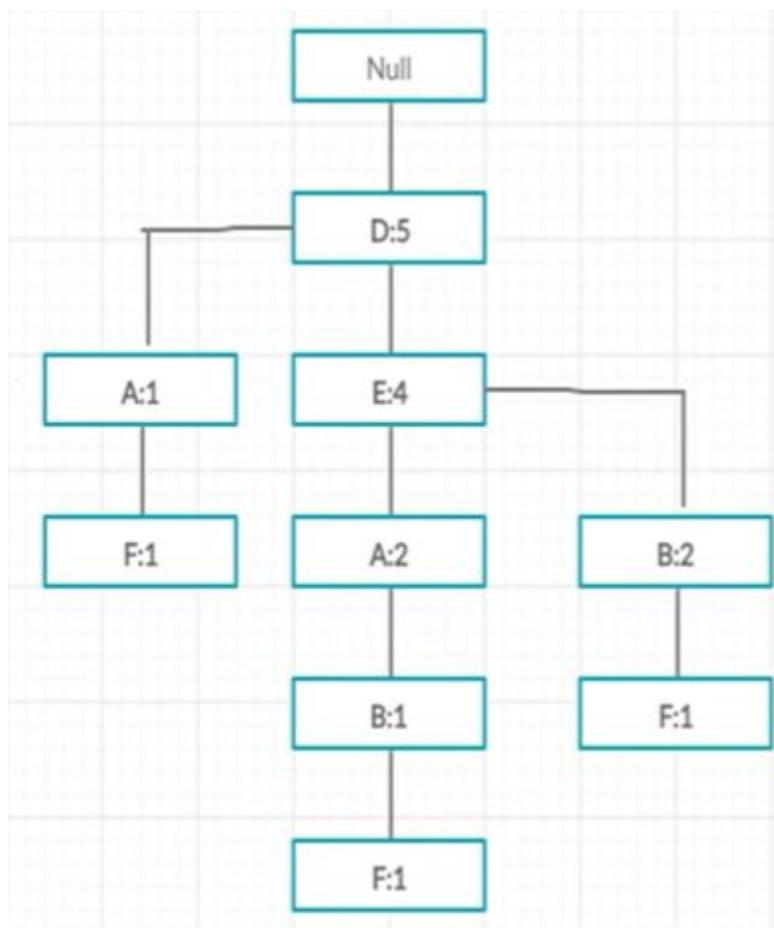
4. D, A, F



5. D, E, B

**Conditional pattern base(CPB):** This is a truncated version of the transaction database that is pertinent to a particular prefix.

### Step 6: Conditional Pattern Base



Start with least priority node

Conditional Pattern base for node F

F:3

D E B:1

D A:1

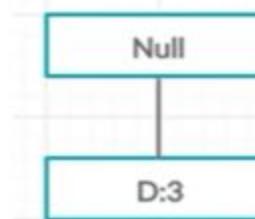
D E A B:1

Count of Node

D:3, E:2, A:2, B:2

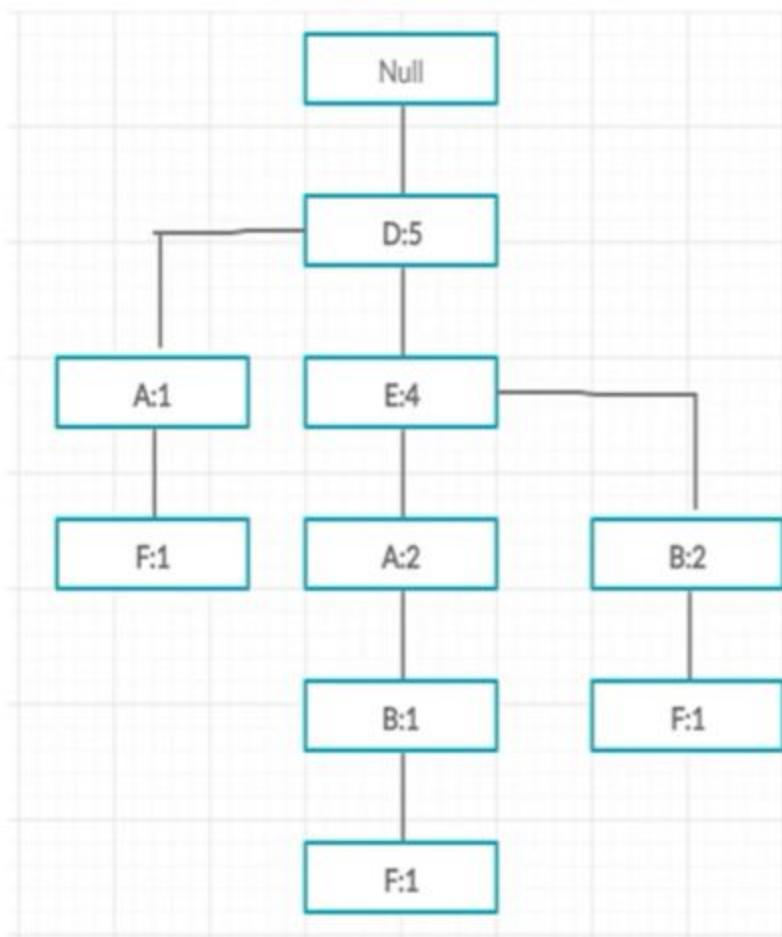
Min. Support count is 3.

Only D is satisfied



Frequent pattern are F:3, DF:3

Min. Support count is 3.



#### Conditional Pattern base for node B

B:3

D E:2

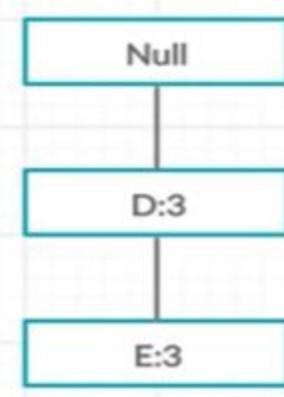
D E A:1

Count of Node

D:3, E:3, A:1

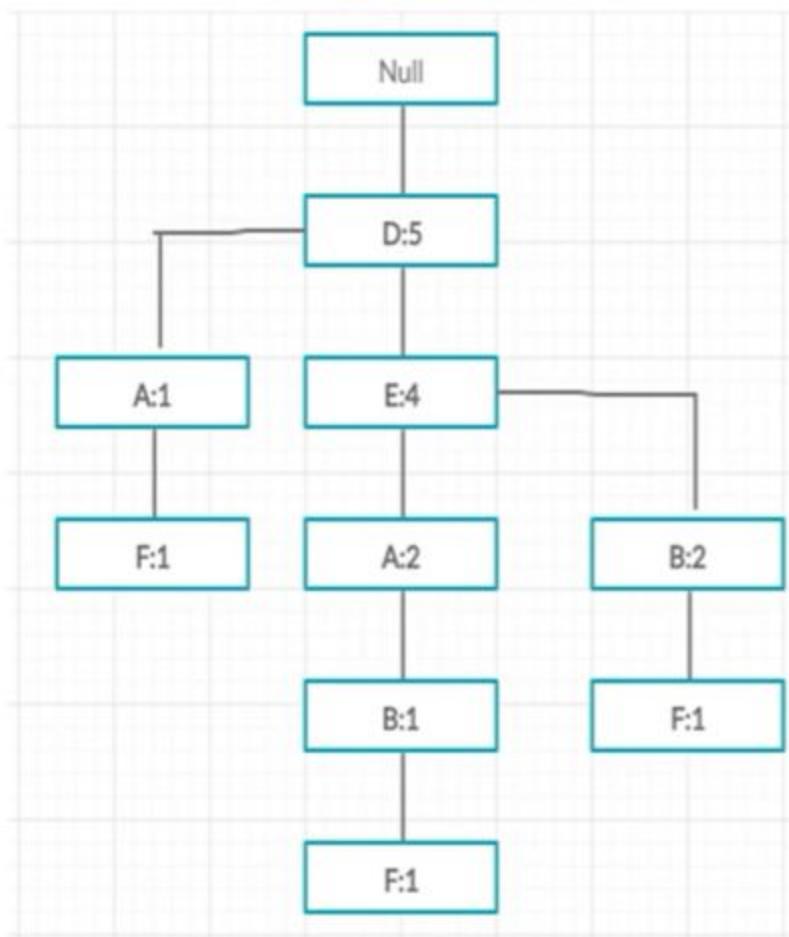
Only D & E is satisfied

DE:2 DE:1



Frequent pattern are **B:3, BD:3, BE:3, BDE:3**

Min. Support count is 3.



### Conditional Pattern base for node A

A:3

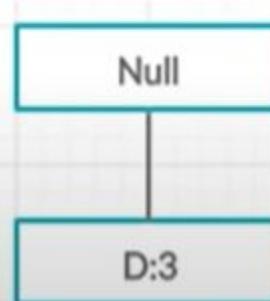
D:1

DE:2

Count of Node

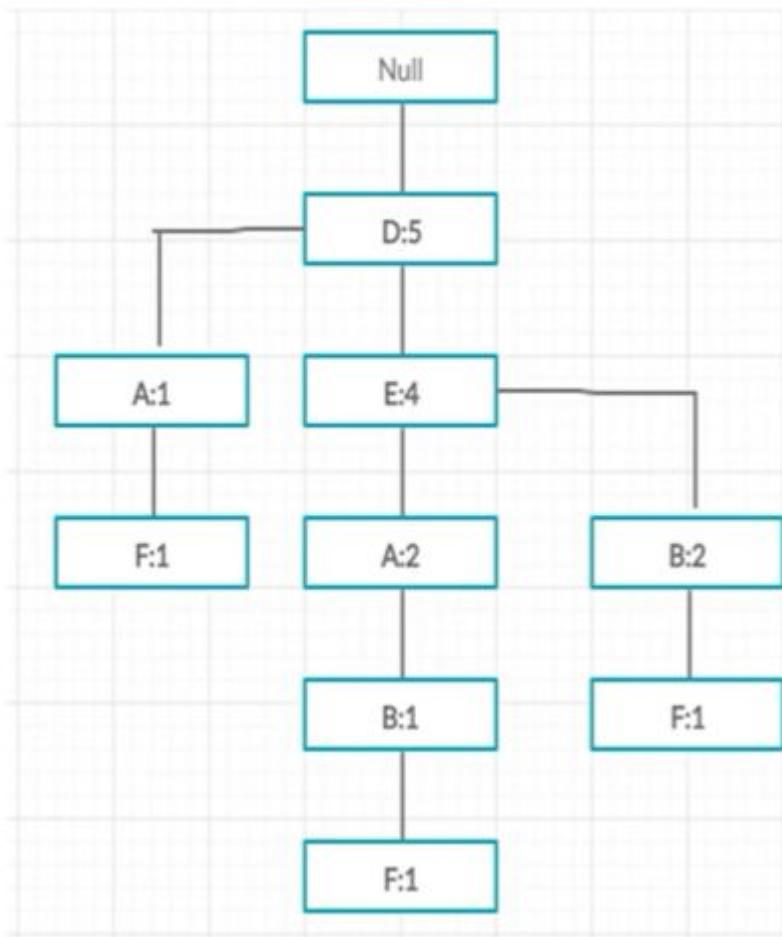
D:3, E:2

Only D satisfied



Frequent pattern are **A:3, AD:3**

Min. Support count is 3.



### Conditional Pattern base for node E

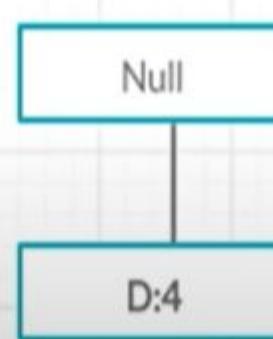
E: 4

D:4

Count of Node

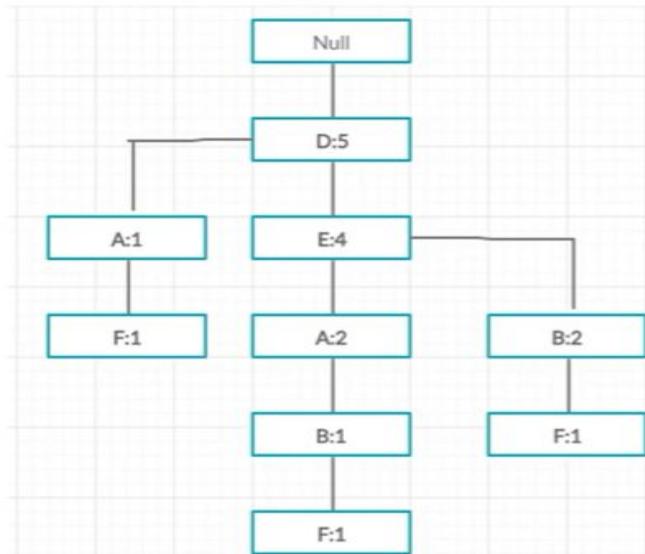
D:4

Only D satisfied



Frequent pattern are E:4, DE:4

Min. Support count is 3.



**Conditional Pattern base for  
node D**

**D:5**

Frequent pattern are **D:5**

Frequent pattern are

**F:3,DF:3**

**B:3,BD:3, BE:3, BDE:3**

**A:3, AD:3**

**E:4, DE:4**

**D:5**

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

**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

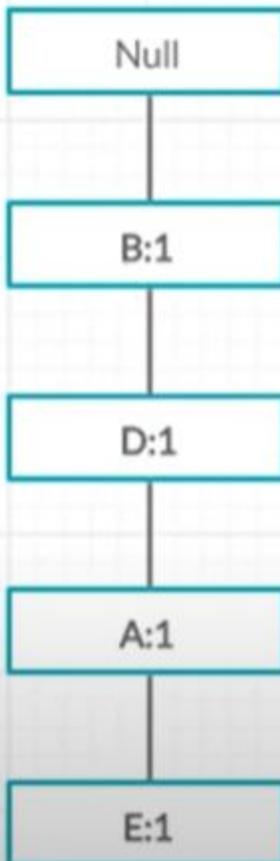
**Step 4:** Order the items according to priority

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

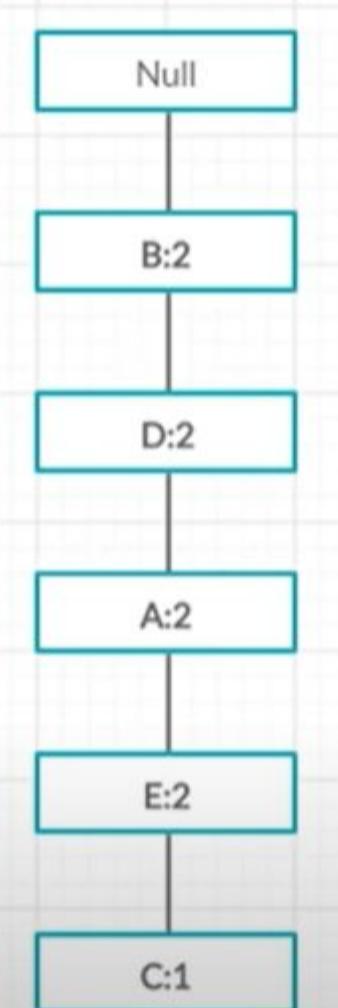
TID	List of Items
T1	B, D, A, E
T2	B, D, A, E, C
T3	B, A, E, C
T4	B, D, A
T5	D
T6	B, D
T7	D, A, E
T8	B, C

Original Table for Reference

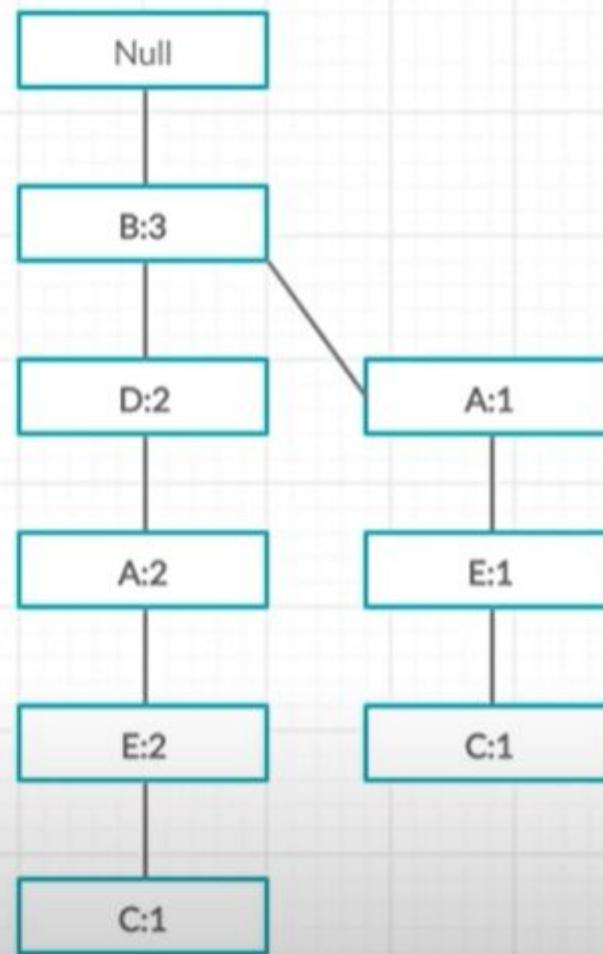
## Step 5: Build FP Tree



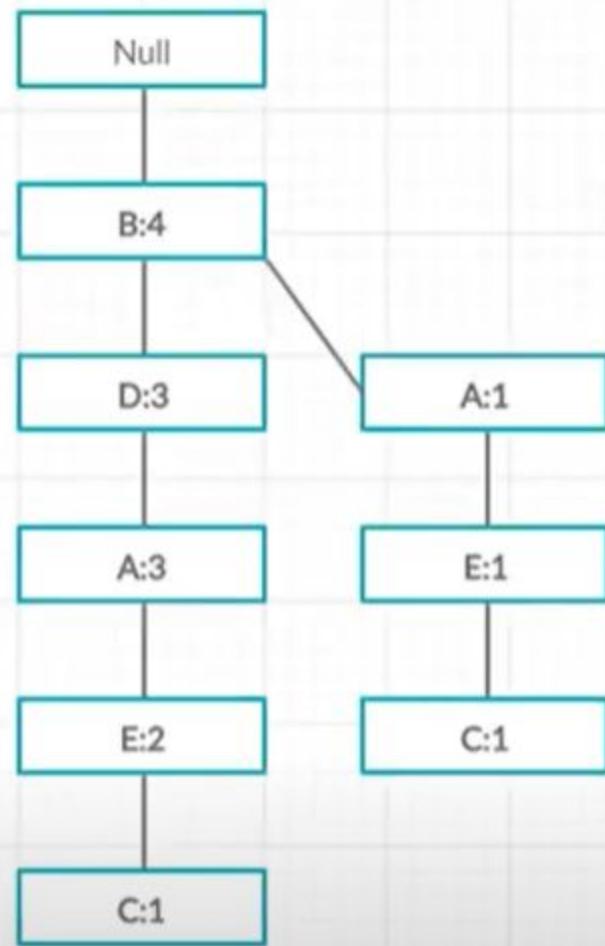
1. B, D, A, E



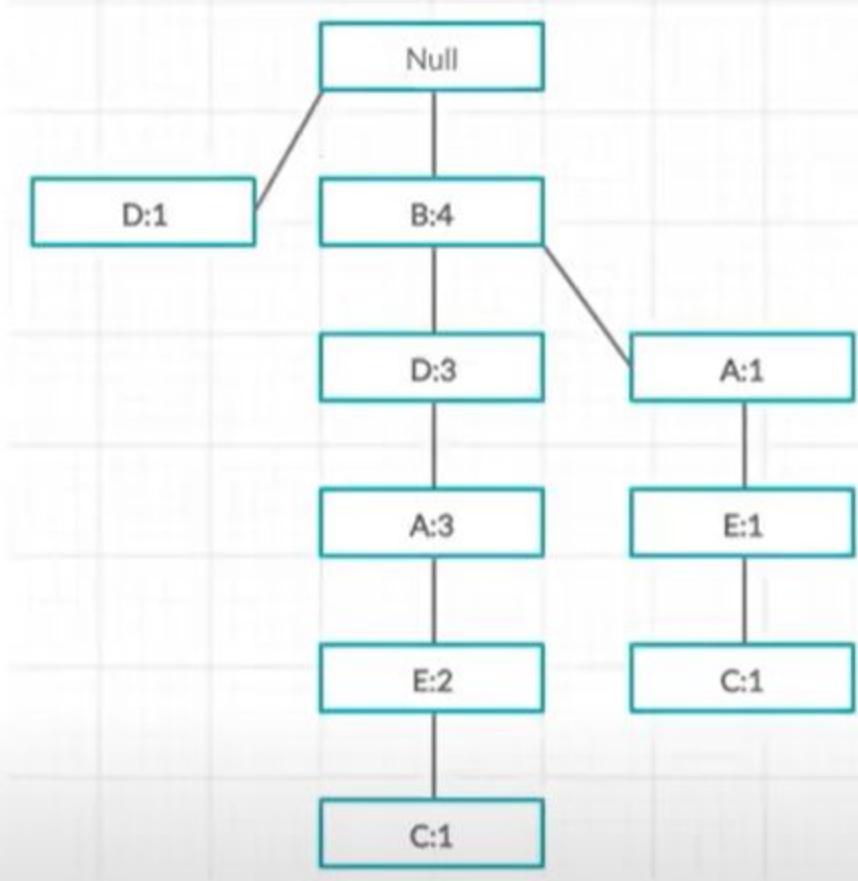
2. B, D, A, E, C



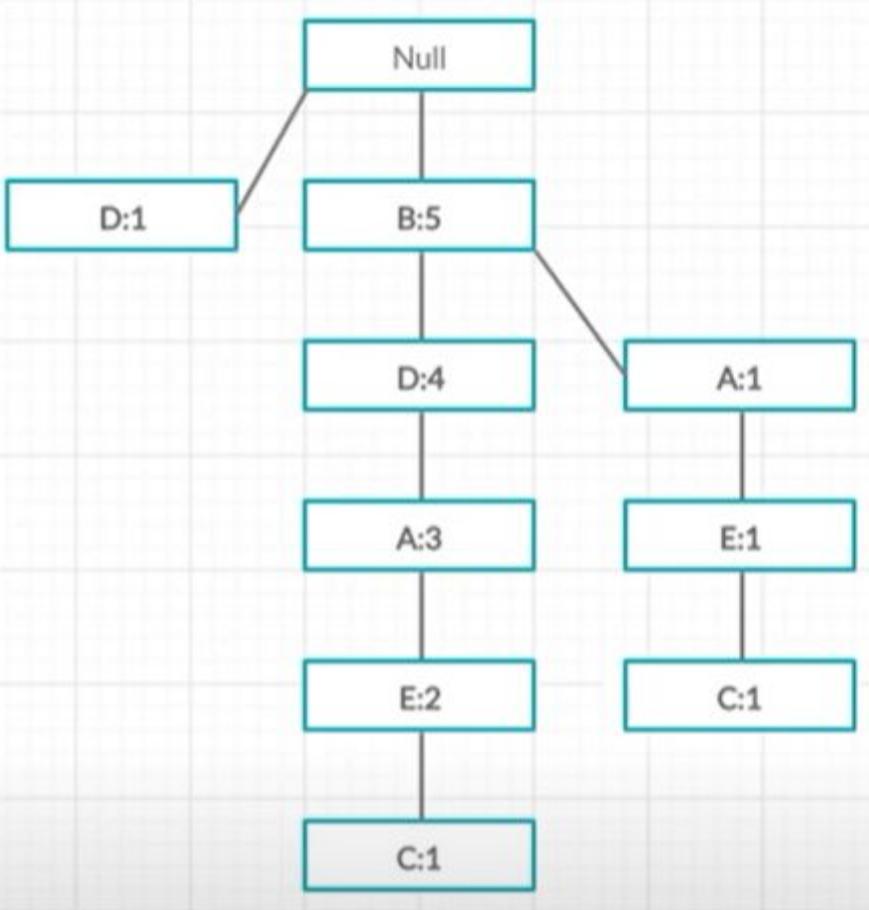
3. B, A, E, C



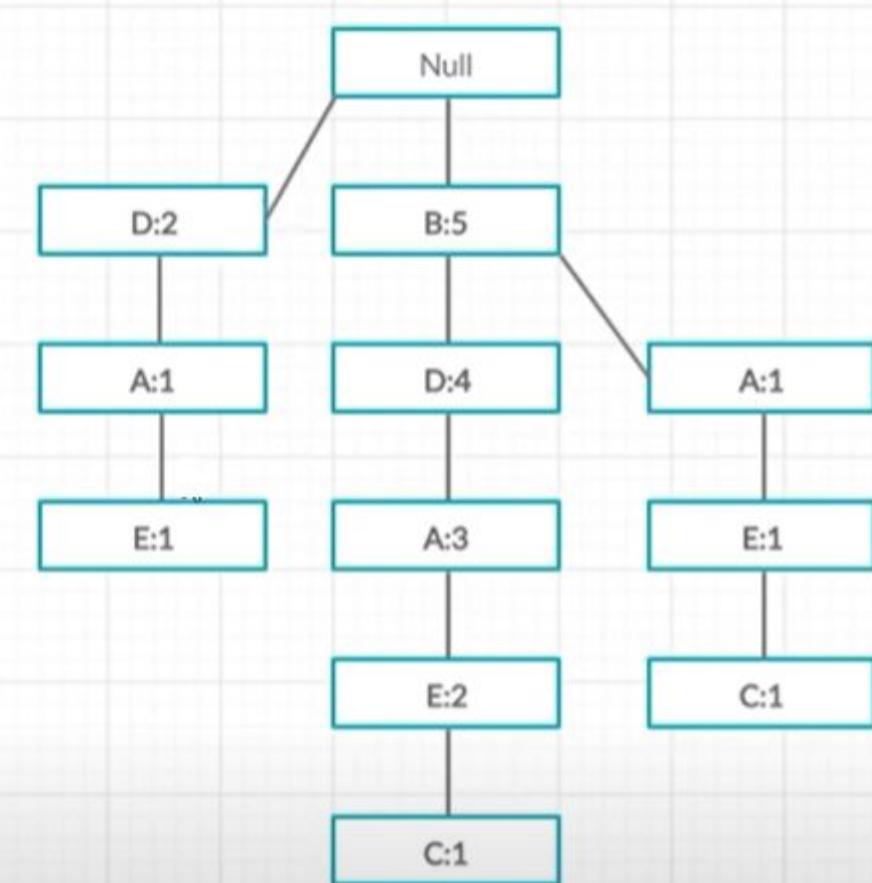
4. B, D, A



5. D

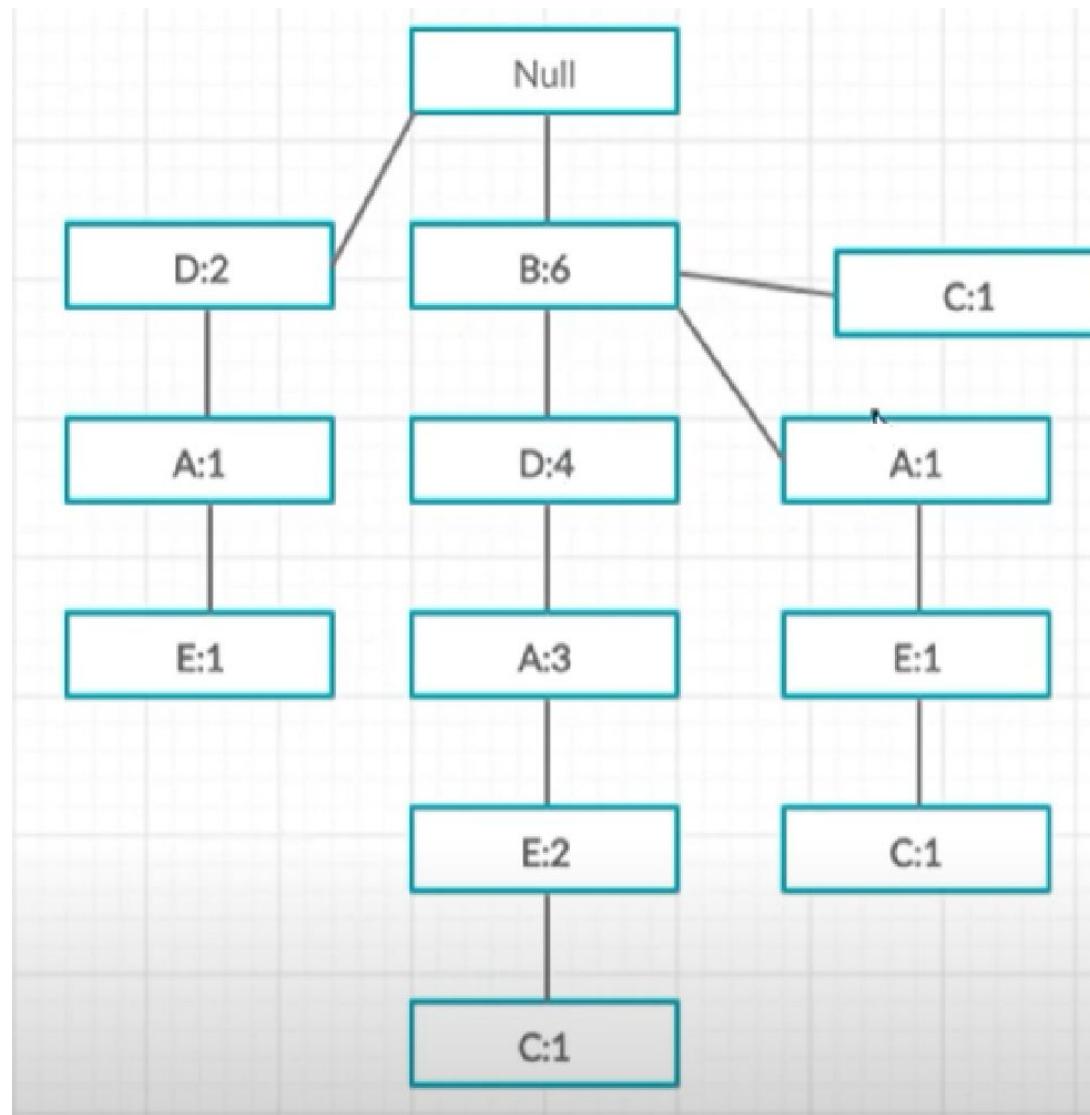


6. B, D



7. D, A, E

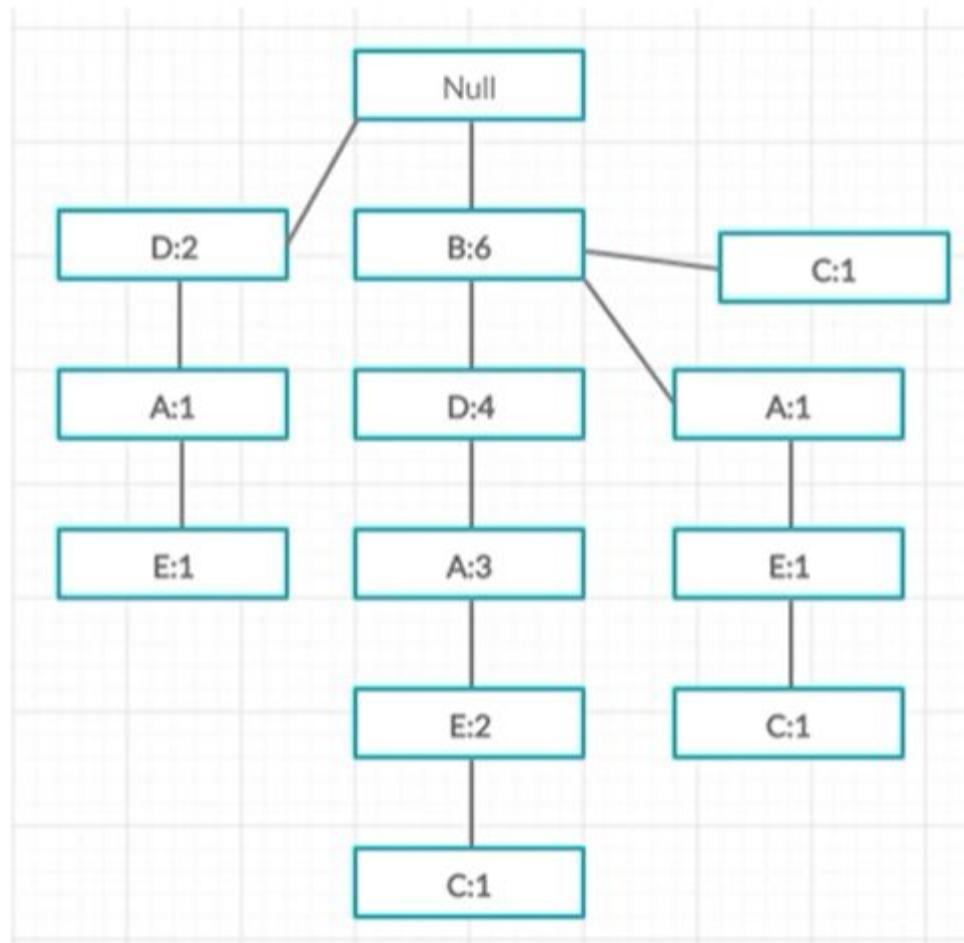
8. B, C



## Step 6: Conditional Pattern Base

Start with least priority node

Conditional Pattern base for node C



C:3

B : 1

B, D, A, E: 1

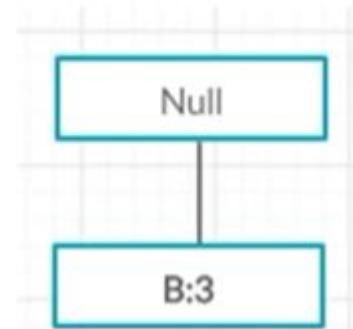
B, A, E: 1

Count of Node

B:3, A:2, E:2, D:1

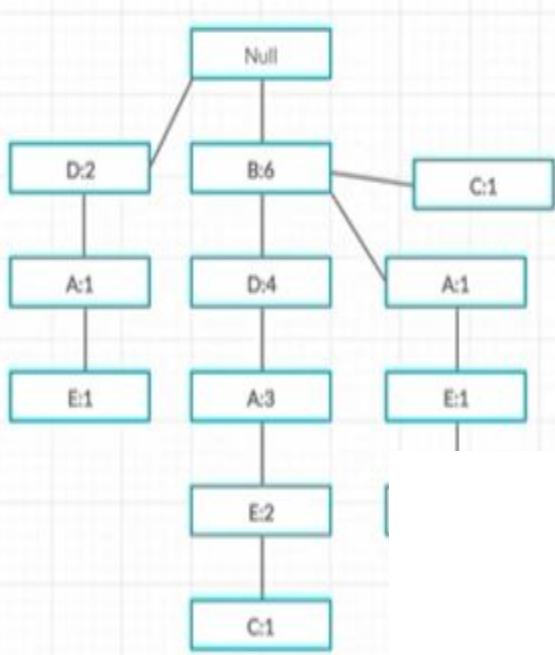
Min. Support count is 3.

Only B is satisfied



Frequent pattern are **C:3, CB:3**

Min. Support count is 3.



Conditional Pattern base for node E E:4

D, A : 1    B, D, A: 2              B, A : 1

Count of Node    B:3, D:3, A:4

B , D, A are satisfied

A:4, B:3, D:3

A B D:2

A D:1

A B:1

Frequent pattern are **E:4**

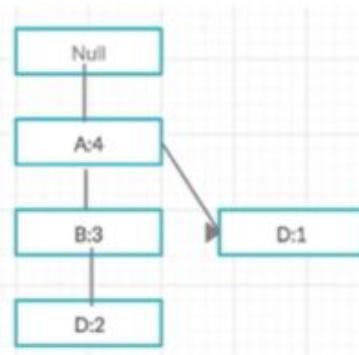
Two combinations DE:3, AE:4, BE:3

Three combinations DE: ADE BDE

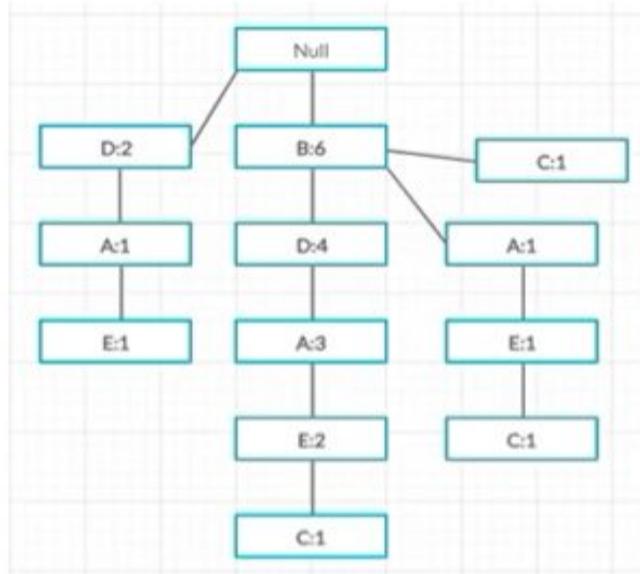
ADE: 3, BDE:2

AE: ADE,(already covered) ABE: 3

BE: ABE , DBE (already covered)



Frequent pattern are **E:4, DE:3, AE:4, BE:3, ADE:3, ABE:3**



## Conditional Pattern base for node A

**A:5**

D : 1   B, D: 3    B : 1

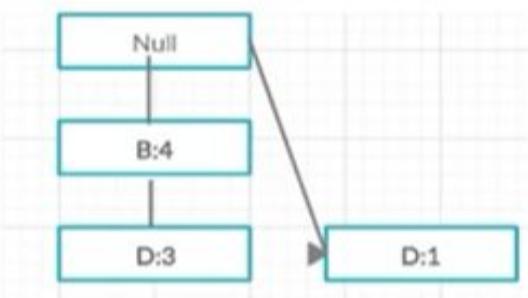
Count of Node   B:4, D:4

B , D are satisfied

B:1

BD:3

D:1



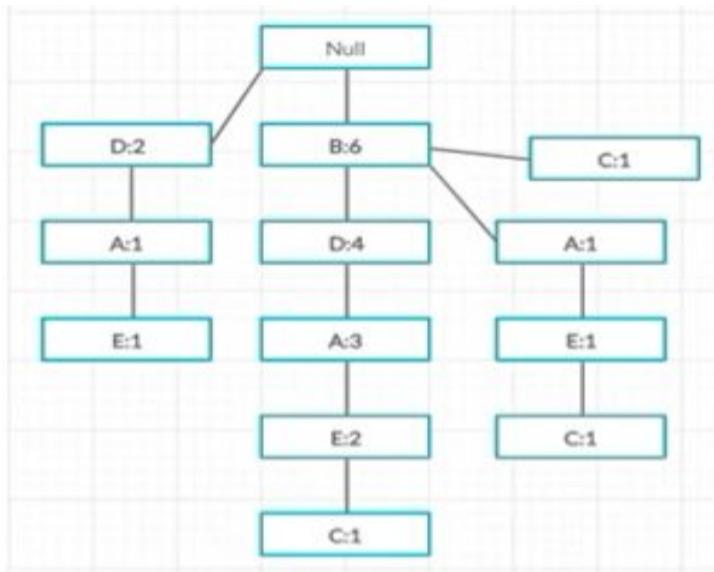
Frequent pattern are **A:5**

Two combinations AD:4, AB:4

Three combinations AD:ABD

ABD:3

Frequent pattern are **A:5, AD:5, AB:4, ABD:3**

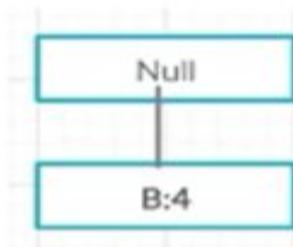


**Start with least priority node**

**Conditional Pattern base for node D:6**

B: 4

Count of Node B:4

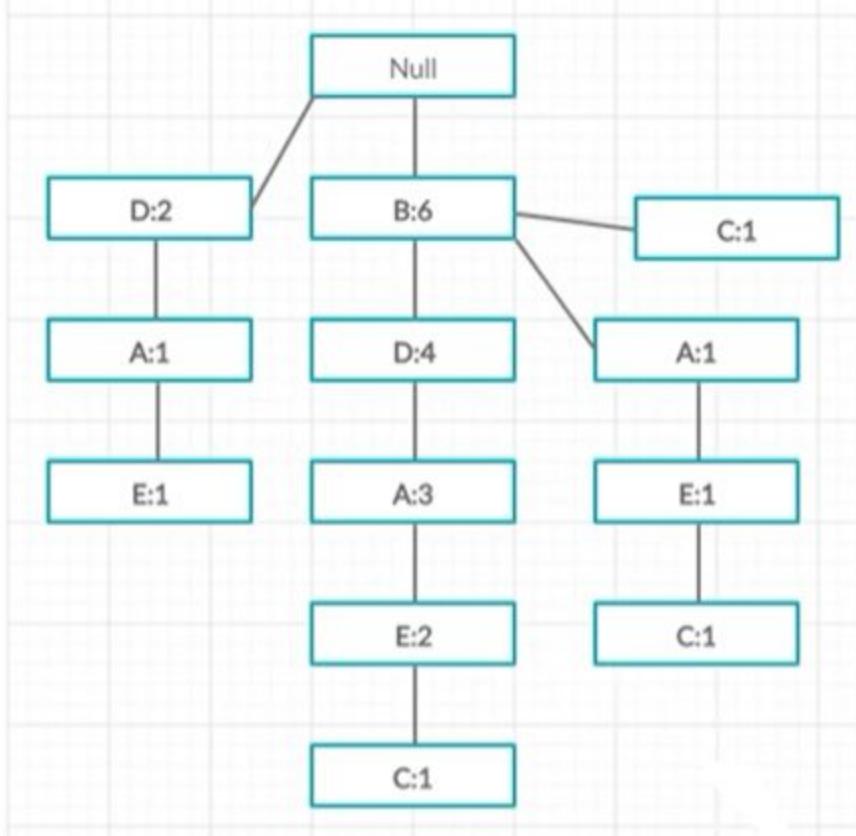


**Two combinations BD:4**

Frequent pattern are **D:6, BD:4**

**Conditional Pattern base for node B:6**

B: 6



Frequent pattern are

**C:3, CB:3**

**E:4, DE:3, AE:4, BE:3, ADE:3, ABE:3**

**A:5, AD:5, AB:4, ABD:3**

**D:6, BD:4**

**B:6**

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

# 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.

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	--

<b>Itemset</b>	<b>List of Items</b>
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	--

<b>Itemset</b>	<b>List of Items</b>
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**

<b>Itemset</b>	<b>List of Items</b>
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

<b>Itemset</b>	<b>List of Items</b>
I1, I2,I3	T8, T9
I1, I2, I5	T1, T8
I1, I3, I5	T8
I2, I3, I4	--
I2, I3, I5	T8
I2, I4, I5	--

<b>Itemset</b>	<b>List of Items</b>
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.                      Confidence = 70%  
For e.g. I1, I2 and I5

Association Rule	Confidence	Confidence (%)
$I1 \wedge I2 \rightarrow I5$	$C(I1, I2, I5) / C(I1, I2) = 2/4$	50%
$I1 \wedge I5 \rightarrow I2$	$C(I1, I2, I5) / C(I1, I5) = 2/2$	100%
$I2 \wedge I5 \rightarrow I1$	$C(I1, I2, I5) / C(I2, I5) = 2/2$	100%
$I1 \rightarrow I2 \wedge I5$	$C(I1, I2, I5) / C(I1) = 2/6$	33%
$I2 \rightarrow I1 \wedge I5$	$C(I1, I2, I5) / C(I2) = 2/7$	29%
$I5 \rightarrow I1 \wedge I2$	$C(I1, I2, I5) / C(I5) = 2/2$	100%

---

### **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.  $\text{purchase}(X, \text{"Milk"}) \rightarrow \text{purchase}(X, \text{"Bread"})$

TID	Purchase
1	Milk, Bread
2	Bread, Butter
3	Beer
4	Milk, Bread, Butter
5	Bread, Beer
6	Milk, Bread, Butter

Min. Support=3

P: Present A:Absent

	C1			
	Milk	Bread	Butter	Beer
	I1	I2	I3	I4
1.	P	P	A	A
2.	A	P	P	A
3.	A	A	A	P
4.	P	P	P	A
5.	A	P	A	P
6.	P	P	P	A
Total	3	5	3	2

# Single Dimensional Association Rules

---

Apply Apriori

(2 Itemset generation)

L1

	I1	I2	I3
1.	P	P	A
2.	A	P	P
3.	A	A	A
4.	P	P	P
5.	A	P	A
6.	P	P	P
Total	3	5	3

C2

Itemset	Support Count
I1, I2	3
I1, I3	2
I2, I3	3

# Single dimensional Association Rules

L2

Itemset	Support Count
I1, I2	3
I2, I3	3

Itemset	Support Count
Milk, Bread	3
Bread, Butter	3

Association Rule	Confidence	Confidence (%)
$I1 \rightarrow I2$	$C(I1, I2) / C(I1) = 3/3$	100%
$I2 \rightarrow I1$	$C(I1, I2) / C(I2) = 3/5$	60%
$I2 \rightarrow I3$	$C(I2, I3) / C(I2) = 3/5$	60%
$I3 \rightarrow I2$	$C(I2, I3) / C(I3) = 3/3$	100%

Predefined confidence can be used.

# Single Dimensional Association Rules

---

Confidence=70%

**Single predicate : purchase is used**

$\text{purchase}(X, \text{"Milk"}) \rightarrow \text{purchase}(X, \text{"Bread"})$

$\text{purchase}(X, \text{"Butter"}) \rightarrow \text{purchase}(X, \text{"Bread"})$

For e.g. If Min. support=2, then 3 itemset combination is possible.  
then following rule may generate

$\text{purchase}(X, \text{"Milk"}) \wedge \text{purchase}(X, \text{"Butter"}) \rightarrow \text{purchase}(X, \text{"Bread"})$

# Multidimensional Association Rules

## 2) Multi dimensional or Inter Dimensional Association Rule

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

e.g.

$\text{Student}(X, \text{"Yes"}) \wedge \text{Credit Rating}(X, \text{"Excellent"}) \rightarrow \text{buys\_Laptop}(X, \text{"Yes"})$

$\text{Student}(X, \text{"No"}) \wedge \text{Credit Rating}(X, \text{"Fair"}) \rightarrow \text{buys\_Laptop}(X, \text{"No"})$

No.	Student	Credit Rating	Buys_Laptop
1	Yes	Excellent	Yes
2	Yes	Fair	No
3	No	Excellent	Yes
4	No	Excellent	Yes
5	Yes	Fair	No
6	No	Fair	No

Attributes	Item
Student=Yes	I1
Student=No	I2
Credit Rating=Excellent	I3
Credit Rating=Fair	I4
Buys_Laptop=Yes	I5
Buys_Laptop=No	I6

Min. Support=1

P: Present A:Absent

# Multidimensional Association Rules

No.	Student	Credit Rating	Buys_Laptop
1	Yes	Excellent	Yes
2	Yes	Fair	No
3	No	Excellent	Yes
4	No	Excellent	Yes
5	Yes	Fair	No
6	No	Fair	No

Attributes	Item
Student=Yes	I1
Student=No	I2
Credit Rating=Excellent	I3
Credit Rating=Fair	I4
Buys_Laptop=Yes	I5
Buys_Laptop>No	I6

C1

I1	I2	I3	I4	I5	I6
P	A	P	A	P	A
P	A	A	P	A	P
A	P	P	A	P	A
A	P	P	A	P	A
P	A	A	P	A	P
A	P	A	P	A	P
3	3	3	3	3	3

Apply Apriori

# Multidimensional Association Rules

(2 Itemset generation)

L1

I1	I2	I3	I4	I5	I6
P	A	P	A	P	A
P	A	A	P	A	P
A	P	P	A	P	A
A	P	P	A	P	A
P	A	A	P	A	P
A	P	A	P	A	P
3	3	3	3	3	3

C2

Itemset	Count
(I1, I2)	0
(I1, I3)	1
(I1, I4)	2
(I1, I5)	1
(I1, I6)	2
(I2, I3)	2
(I2, I4)	1
(I2, I5)	2
(I2, I6)	1
(I3, I4)	0
(I3, I5)	3
(I3, I6)	0
(I4, I5)	0
(I4, I6)	3
(I5, I6)	0

# Multidimensional Association Rules

(3 Itemset generation)

**Join & Prune property:**

First elements must be common of two records.

L2

Itemset	Count
(I1, I3)	1
(I1, I4)	2
(I1, I5)	1
(I1, I6)	2
(I2, I3)	2
(I2, I4)	1
(I2, I5)	2
(I2, I6)	1
(I3, I5)	3
(I4, I6)	3

C3

Itemset	Count
(I1, I3, I4)	0
(I1, I3, I5)	1
(I1, I3, I6)	0
(I1, I4, I5)	0
(I1, I4, I6)	2
(I1, I5, I6)	0
(I2, I3, I4)	0
(I2, I3, I5)	2
(I2, I3, I6)	0
(I2, I4, I5)	0
(I2, I4, I6)	1
(I2, I5, I6)	0

# Multidimensional Association Rules

L3

Itemset	Count
(I1, I3, I5)	1
(I1, I4, I6)	2
(I2, I3, I5)	2
(I2, I4, I6)	1

**Join & Prune property:**

4 Itemset generation not possible as first two items are not same.

Select those records having I5 and I6 item as they represent final class label.

Predefined confidence can be used.

Association Rule	Confidence	Confidence (%)
I1^I3→I5	C(I1,I3,I5)/C(I1,I3)=1/1	100%
I2^I3→I5	C(I2,I3,I5)/C(I2,I3)=2/2	100%
I1^I4→I6	C(I1,I4,I6)/C(I1,I4)=2/2	100%
I2^I4→I6	C(I2,I4,I6)/C(I2,I4)=1/1	100%

# Multidimensional Association Rules

---

Four Multidimensional rules can be obtained

- 1)  $\text{Student}(X, \text{"Yes"}) \wedge \text{Credit Rating}(X, \text{"Excellent"}) \rightarrow \text{buys\_Laptop}(X, \text{"Yes"})$
- 2)  $\text{Student}(X, \text{"No"}) \wedge \text{Credit Rating}(X, \text{"Fair"}) \rightarrow \text{buys\_Laptop}(X, \text{"No"})$
- 3)  $\text{Student}(X, \text{"Yes"}) \wedge \text{Credit Rating}(X, \text{"Fair"}) \rightarrow \text{buys\_Laptop}(X, \text{"No"})$
- 4)  $\text{Student}(X, \text{"No"}) \wedge \text{Credit Rating}(X, \text{"Excellent"}) \rightarrow \text{buys\_Laptop}(X, \text{"Yes"})$

# Multidimensional 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

$\text{Student}(X, \text{"Yes"}) \wedge \text{Credit Rating}(X, \text{"Excellent"}) \wedge \text{Buys}(X, \text{"Desktop"}) \rightarrow \text{Buys}(X, \text{"Laptop"})$

$\text{Student}(X, \text{"No"}) \wedge \text{Credit Rating}(X, \text{"Excellent"}) \wedge \text{Buys}(X, \text{"Laptop"}) \rightarrow \text{Buys}(X, \text{"Printer"})$



