



电子科技大学
University of Electronic Science and Technology of China



Research Issues in Data Stream Association Rule Mining

Shasha Luo



Data Mining Lab,
Big Data Research Center, UESTC
Email: 1079473791@qq.com



目录/Contents

01

Basic concepts

02

Apriori、FP-tree algorithm

03

Data Processing Model

04

Moment algorithm

what is Association rules mining?



One day Xiao Ming's wife told him to go to the Wal-Mart supermarket to buy diapers, he found that next to the diapers was beer on the shelves, ha was very happy .So he bought a dozen of beer, so he came home after the

what is Association rules mining?



- Transaction
- Itemset
- (k) Itemset
- Frequent Itemset

TID					
1	A	E	F	G	
2	A	F	G		
3	A	B	E	F	G
4	E	F	G		

Dataset T

$\{A,F,G\}$: Support count = 3 , Support = $3/4 = 75\%$

Min_sup = 50%

what is Association rules mining?



➤ Association rules : $X \rightarrow Y (s, c), X \cap Y = \emptyset$

TID					
1	A	E	F	G	
2	A	F	G		
3	A	B	E	F	G
4	E	F	G		

$$\text{support}(X \rightarrow Y) = P(X \cup Y)$$

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \text{support}(X \cup Y) / \text{support}(X)$$

eg: **confidence** ($X \rightarrow Y$): $\{F,G\} \rightarrow \{A\}$: $C = S(\text{AFG}) / S(\text{FG}) = 3/4 = 75\%$

min_conf = 70%

Strong rules

what is Association rules mining?



- **Association rules mining**: Given a set of transactions T , association rule discovery refers to finding all rules with support greater than or equal to min_sup and greater than or equal to min_conf . **Min_sup** and **min_conf** are the corresponding support and confidence thresholds.

TID					
1	A	E	F	G	
2	A	F	G		
3	A	B	E	F	G
4	E	F	G		



01 Step one

Frequent Itemset Generation

02 step two

Rule Generation

Common algorithm : Apriori and FP-tree

➤ Introduction

Algorithm Core idea: Generate a set of candidates of length $(k + 1)$ from frequent itemsets of length k .
If an item set is frequent then its **subset** is also frequent.

AllElectronics 数据库

TID	List of item ID's
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

C_1

项集	支持度计数
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

比较候选支持度计数
与最小支持度计数



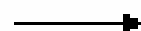
L_1

项集	支持度计数
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Min_sup=2

C_2

由 L_1 产生
候选 C_2



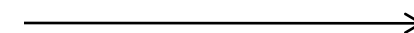
项集
{I1,I2}
{I1,I3}
{I1,I4}
{I1,I5}
{I2,I3}
{I2,I4}
{I2,I5}
{I3,I4}
{I3,I5}
{I4,I5}

扫描D, 对每个
候选计数



C_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I4}	1
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2
{I3,I4}	0
{I3,I5}	1
{I4,I5}	0



比较候选支持度计数
与最小支持度计数

L_2

项集	支持度计数
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

由 L_2 产生
候选 C_3



C_3

项集
{I1,I2,I3}
{I1,I2,I5}

扫描D, 对每个
候选计数



C_3

项集	支持度计数
{I1,I2,I3}	2
{I1,I2,I5}	2

比较候选支持度计数
与最小支持度计数



L_3

项集	支持度计数
{I1,I2,I3}	2
{I1,I2,I5}	2

How to generate a candidate

Step 1: L_k 's self-connection (连接)

$\{I1, I2\}$ 和 $\{I1, I3\}$ --> $\{I1, I2, I3\}$

Step 2: Pruning (剪枝)

3-项集

: $\{I1, I2, I3\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}$,但是
由于 $\{I3, I4\}$ 和 $\{I4, I5\}$ 没有出现在 L_2 中, 所以
 $\{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}$ 被剪枝掉了。

➤ Disadvantages:

- 1、 Scan the data repeatedly
- 2、 Generate candidate sets
- 3、 Too many transactions cause computational complexity

➤ Advantages :

Scan the data twice

Will not Generate candidate sets

➤ steps:

- 1、 **compress** the database representing the frequent itemsets onto the FP-tree
- 2、 the FP-tree is divided into a set of **conditional databases** (each data database associated with a frequent or "pattern segment"), mining each condition database to obtain frequent itemsets

TID 1 薯片,鸡蛋,面包,牛奶
TID 2 薯片,鸡蛋,啤酒
TID 3 薯片,鸡蛋,面包
TID 4 薯片,鸡蛋,面包,牛奶,啤酒
TID 5 面包,牛奶,啤酒
TID 6 鸡蛋,面包,啤酒
TID 7 薯片,面包,牛奶
TID 8 薯片,鸡蛋,面包,牛奶
TID 9 薯片,鸡蛋,牛奶

薯片:7鸡蛋:7面包:7牛奶:6啤酒:4

MinSup=4

- Support Order: **Scan** the data set once to determine the support **count** for each item. **Discard** infrequent items, and frequent items are sorted by decreasing support.

TID 1 薯片,鸡蛋,面包,牛奶

TID 2 薯片,鸡蛋,啤酒

TID 3 薯片,鸡蛋,面包

TID 4 薯片,鸡蛋,面包,牛奶,啤酒

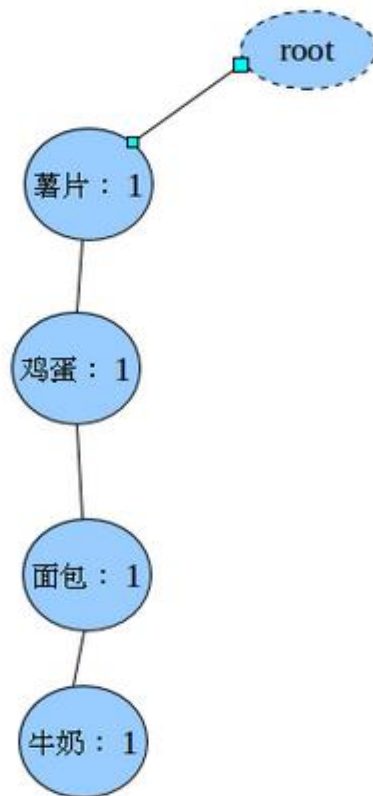
TID 5 面包,牛奶,啤酒

TID 6 鸡蛋,面包,啤酒

TID 7 薯片,面包,牛奶

TID 8 薯片,鸡蛋,面包,牛奶

TID 9 薯片,鸡蛋,牛奶

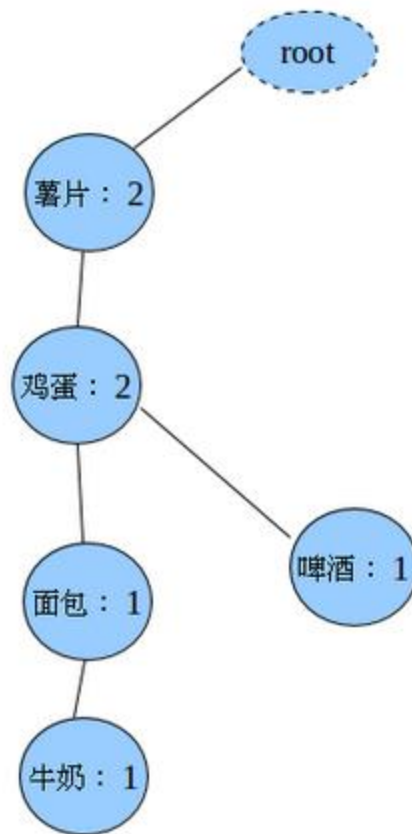


- Build the FP tree: the second scan dataset, **read** the first transaction {potato chips, eggs, bread, milk}, **create** the nodes with these names. Then form a null -> potato chips -> eggs -> bread -> milk **path**. The frequency **count** for all nodes on this path is 1.

薯片,鸡蛋,面包,牛奶
薯片,鸡蛋,啤酒
薯片,鸡蛋,面包
薯片,鸡蛋,面包,牛奶,啤酒
面包,牛奶,啤酒
鸡蛋,面包,啤酒
薯片,面包,牛奶
薯片,鸡蛋,面包,牛奶
薯片,鸡蛋,牛奶

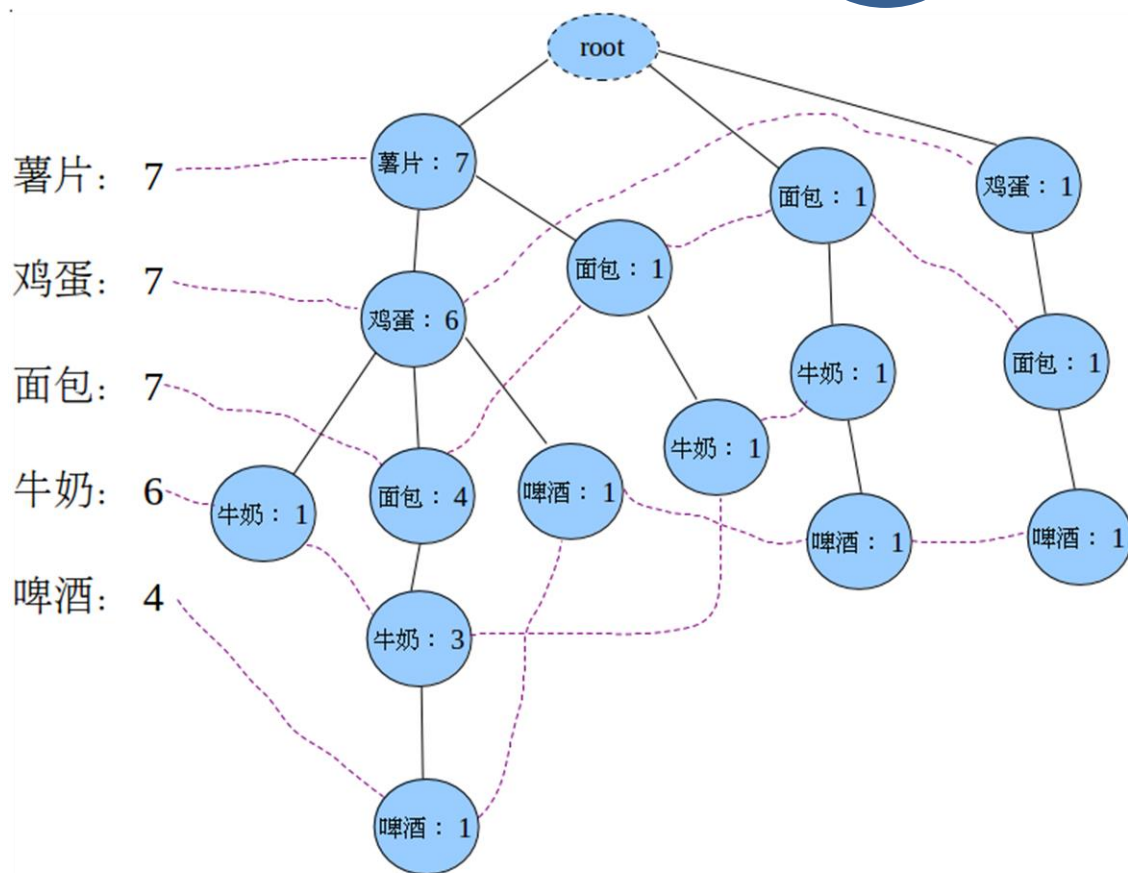
薯片:7鸡蛋:7面包:7牛奶:6啤酒:4

MinSup=4



Insert each transaction in turn,
and increase its support count.

薯片,鸡蛋,面包,牛奶
 薯片,鸡蛋,啤酒
 薯片,鸡蛋,面包
 薯片,鸡蛋,面包,牛奶,啤酒
 面包,牛奶,啤酒
 鸡蛋,面包,啤酒
 薯片,面包,牛奶
 薯片,鸡蛋,面包,牛奶
 薯片,鸡蛋,牛奶

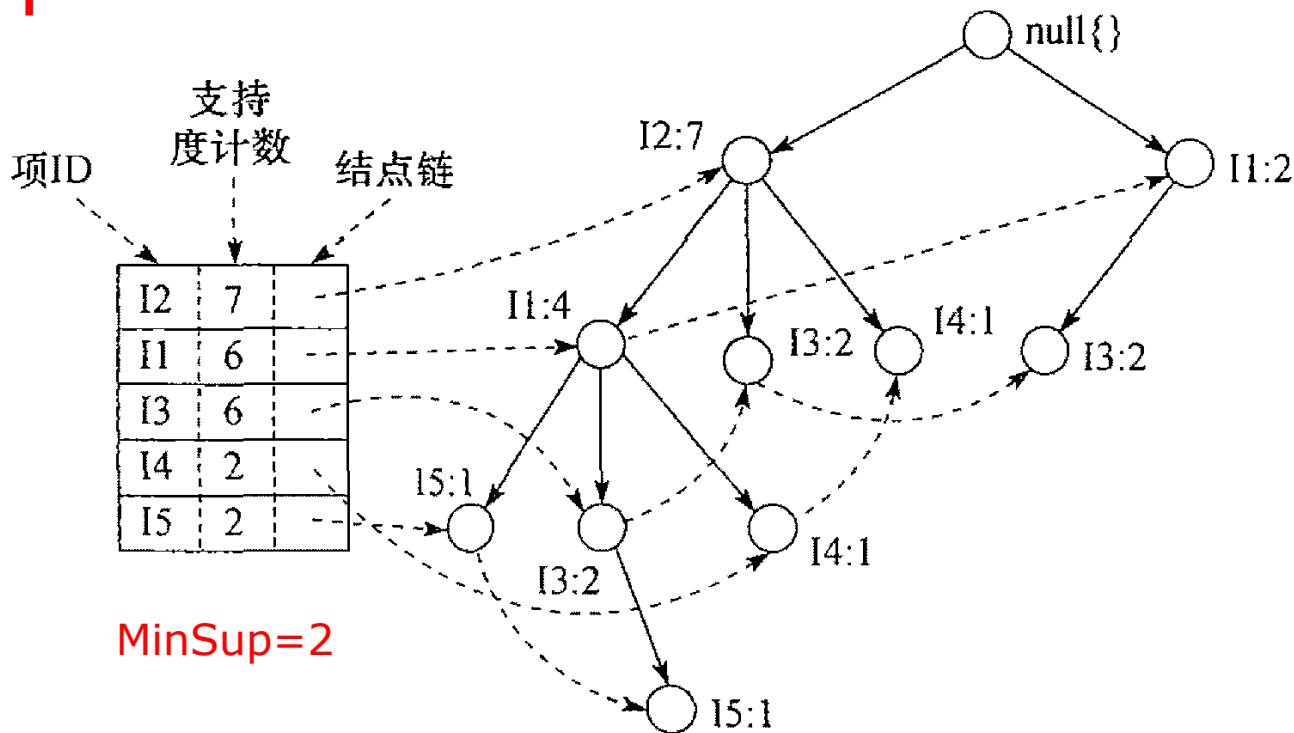


If the two transactions do not have a **common prefix**, then **another** way to open up a **path** until each transaction is mapped to the FP tree, a path is over.

➤ Another example:

AllElectronics 数据库

TID	List of item ID's
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

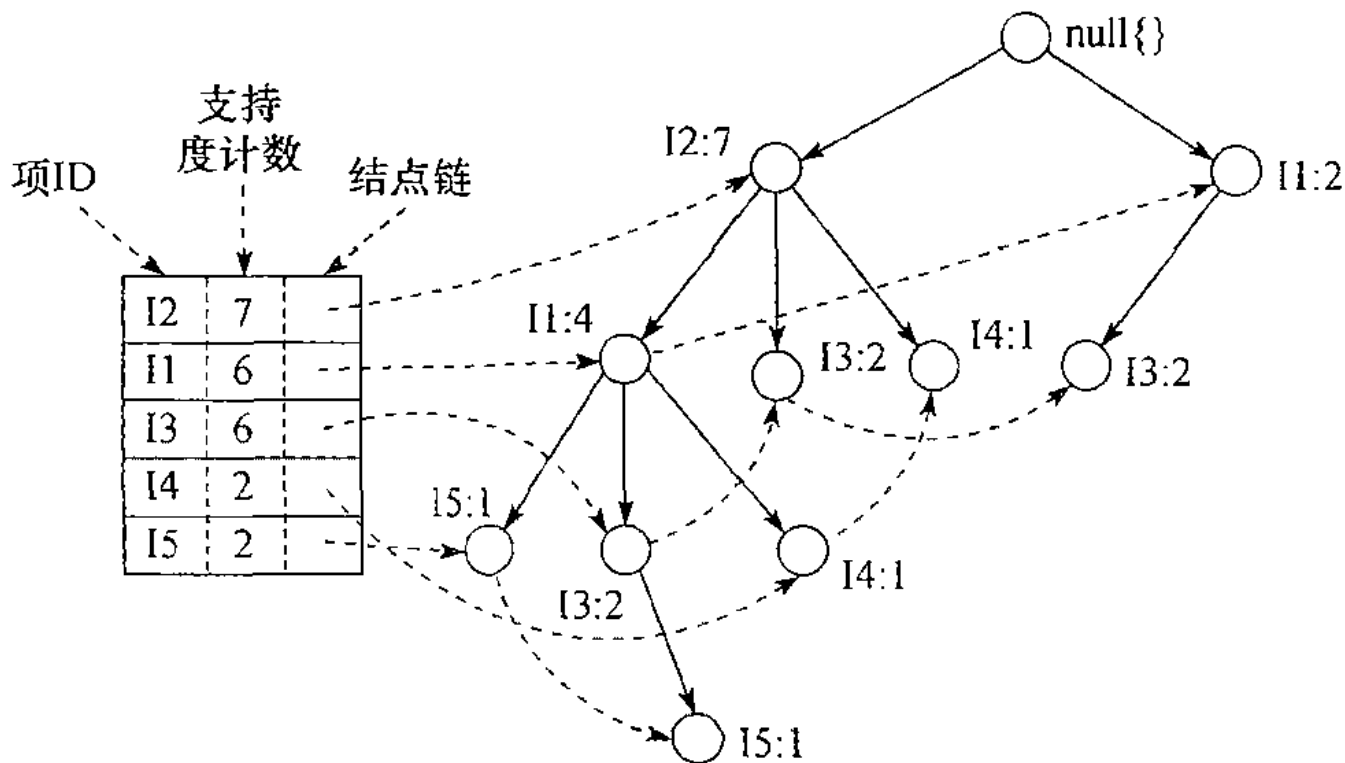


The FP tree also contains a **list** of pointers to nodes that have the same entry (**dotted line**). These pointers help to access items in the tree **quickly and easily**.

➤ How to mining frequent pattern in the FP -tree?

(1) Conditional pattern base

- Conditional pattern Base: A "sub-database" consisting of the prefix path set that appears with the suffix pattern in the FP tree.



➤ How to mining frequent pattern in the FP -tree?

(2) Conditional FP-tree

- The conditional pattern base is treated as a transaction database, construct a conditional FP tree.

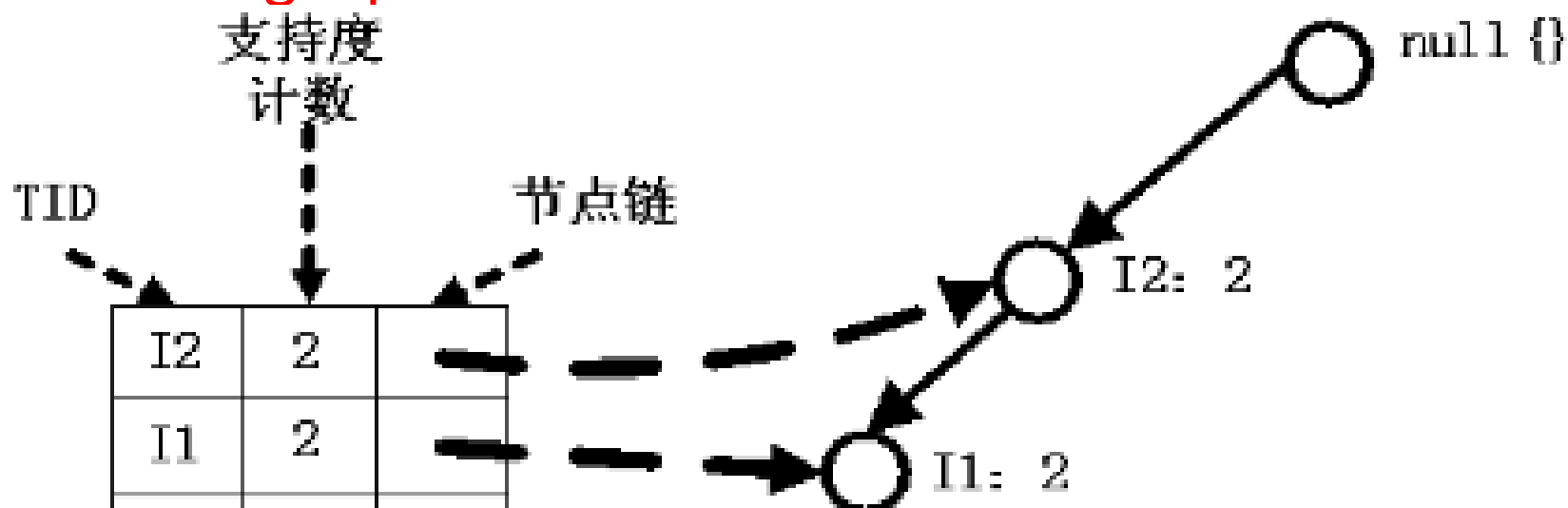
项	条件模式基
I5	{ {I2, I1: 1}, {I2, I1, I3: 1} }
I4	{ {I2, I1: 1}, {I2: 1} }
I3	{ {I2, I1: 2}, {I2: 2}, {I1: 2} }
I1	{ {I2: 4} }

➤ How to mining frequent pattern in the FP -tree?

(3) Mining frequent pattern

For each Condition FP- tree, there are two cases:

① single path



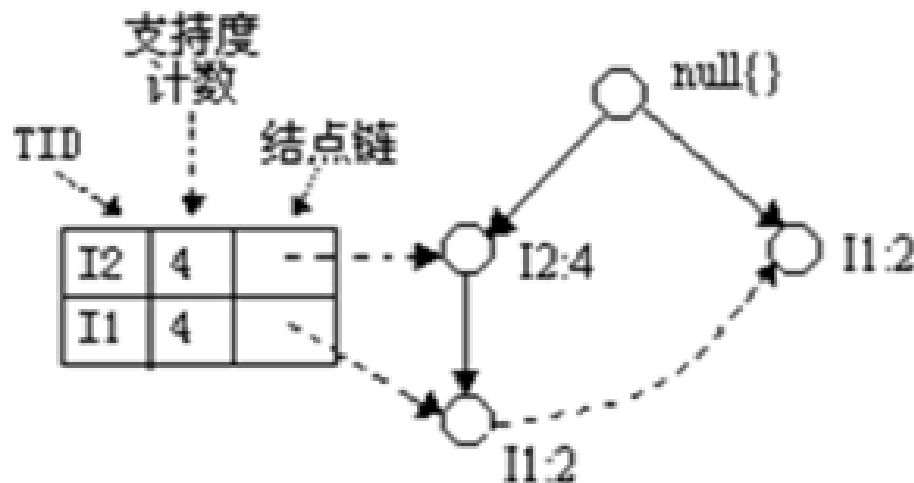
e. g. I5: {I2, I5:2} , {I1, I5:2}, {I2, I1, I5:2}

➤ How to mining frequent pattern in the FP -tree?

(3) Mining frequent pattern

② Multipath path

项 I3



Conditional FP-Tree: $\{(I2:4, I1:2), (I1:2)\}$

Part of the frequent pattern: I1 I3:4, I1 I3:4,

Note: The frequent pattern is not all nodes with the suffix I3.

➤ How to mining frequent pattern in the FP -tree?

(3) Mining frequent pattern

② Multipath path

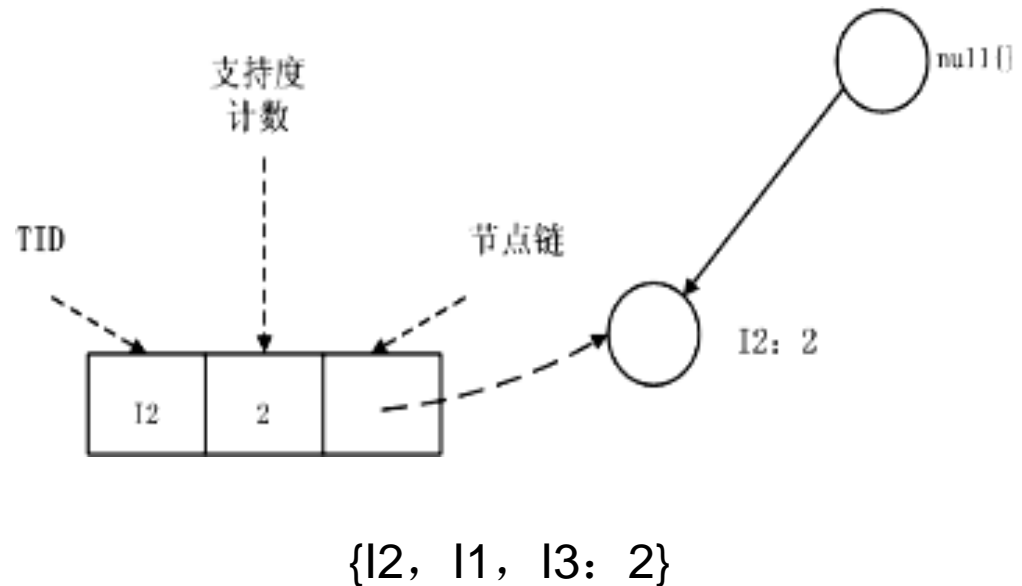
- Method of generating frequent patterns: **Rebuild** conditional pattern bases with suffixes.

Step1: {I2,I3: 4}

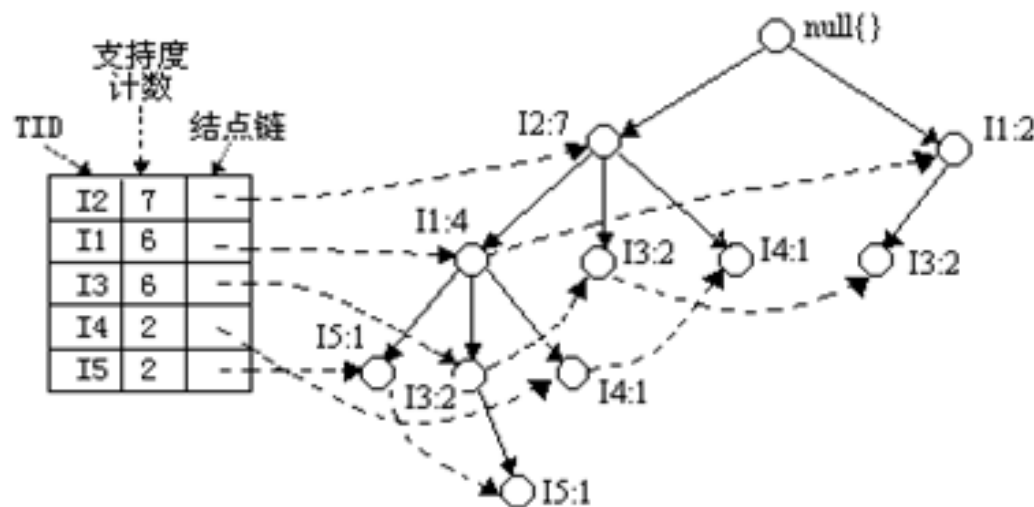
conditional pattern bases is {I2,I3}
then conditional FP- tree is empty

Step2: {I1,I3: 4}

conditional pattern bases is {I1,I3}
then conditional FP- tree is {I2:2}

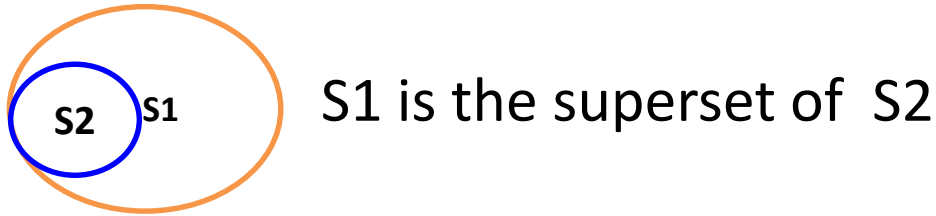


➤ How to mining frequent pattern in the FP -tree?



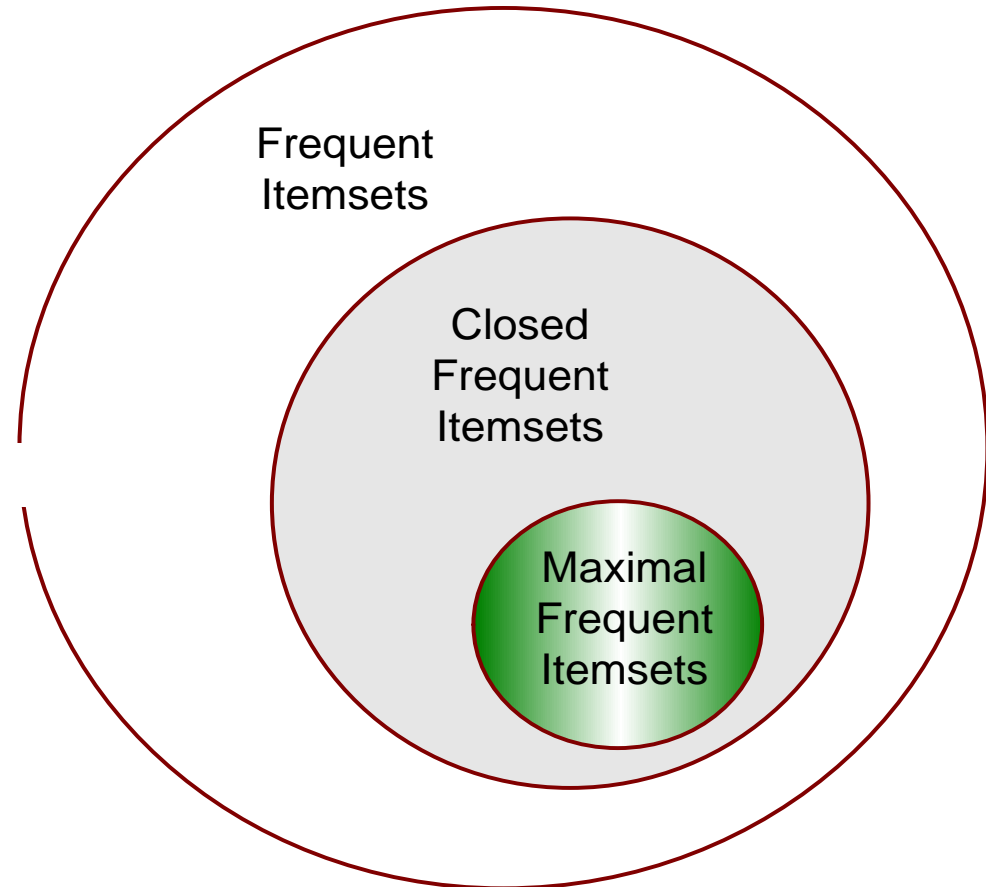
Item	Conditional pattern base(条件模式基)	Conditional FP-tree (条件FP树)	Frequent pattern (频繁模式)
I5	{(I2 I1:1), (I2 I1 I3 :1)}	(I2:2 I1:2)	I2 I5:2, I1 I5:2, I2 I1 I5:2
I4	{(I2 I1:2), (I2:1)}	(I2:2)	I2 I4:2
I3	{(I2 I1:2), (I2:2), I1:2}	(I2:4, I1:2), (I1:2)	I1 I3:4, I1 I3:4, I2 I1 I3:2
I1	{(I2:4)}	(I2:4)	I2 I1:4

➤ Superset



➤ classification

- Closed Frequent itemsets:
abc abcd bce acde de
- Maximal Frequent Itemset:
All superset of frequent itemsets L
are non-frequent itemsets

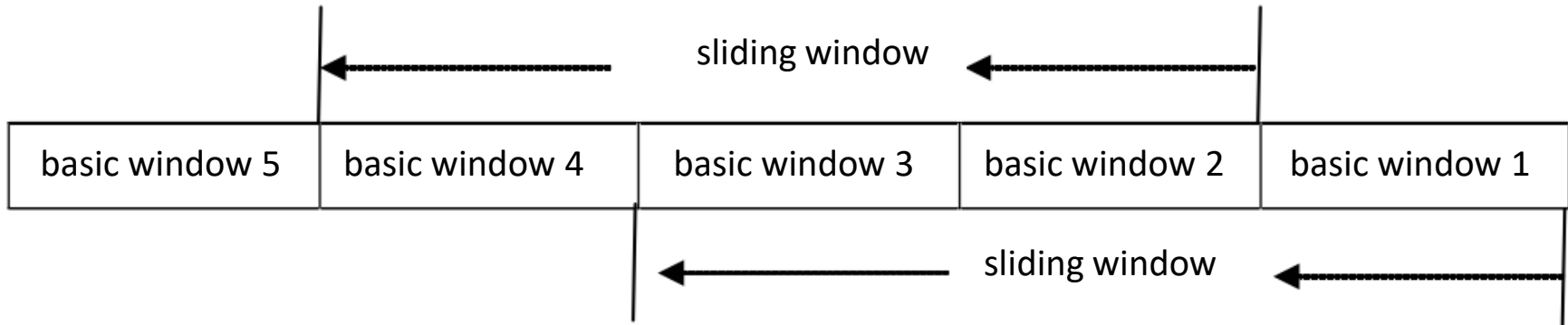


➤ Problems:

- One -scan
- The speed of the algorithm
- An incremental process to keep up with the highly update rate.

➤ Data Processing Model

Landmark ; Time-fading; sliding windows models



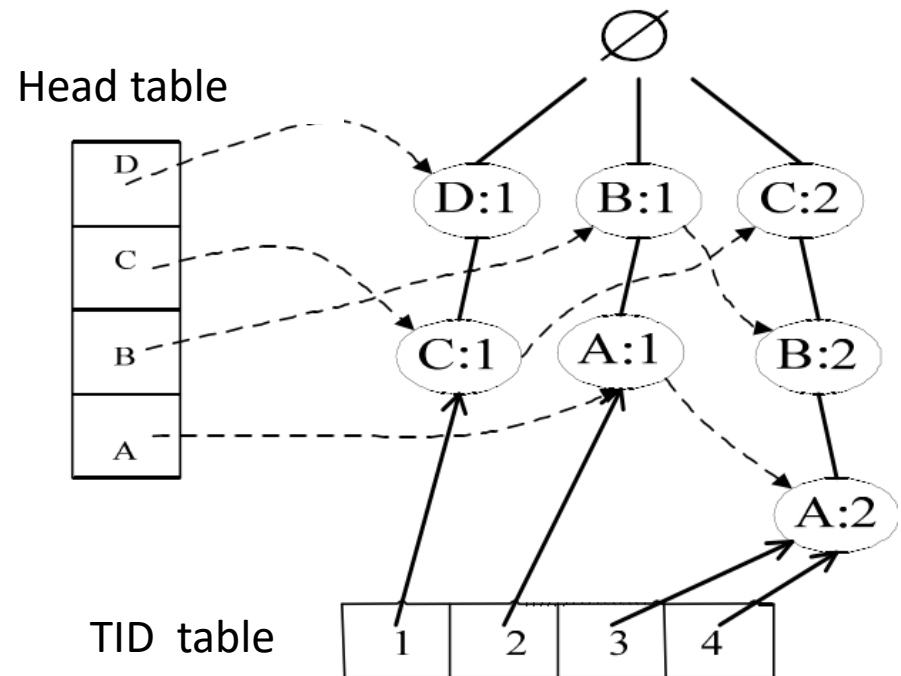
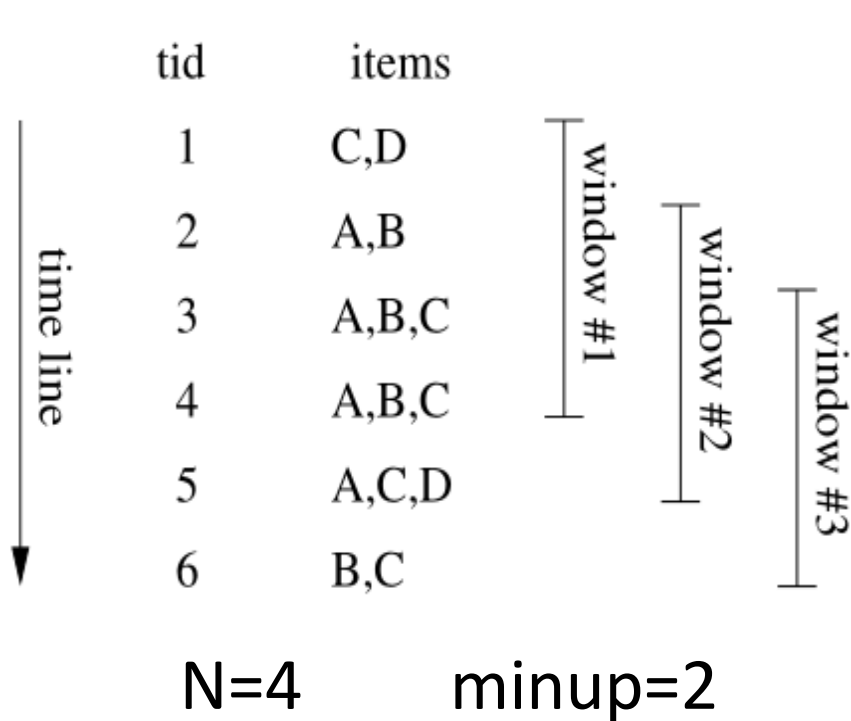
When the data arrives continuously, when the data spills, the **first-in first-out** principle is usually used to replace the old data, so the algorithm takes into account the **time weight** and expired data on the current results have no effect on the recent data .

➤ Introduction

This algorithm first requires **FP-tree** to compress the data, process the data stream based on the **transaction sliding window model**, and maintain a tree structure called **CET (closed enumeration tree)** to mine the frequent closed pattern.

➤ Steps

(1) Initialize **FP-tree** structure and **headTable** according to the sliding window size.



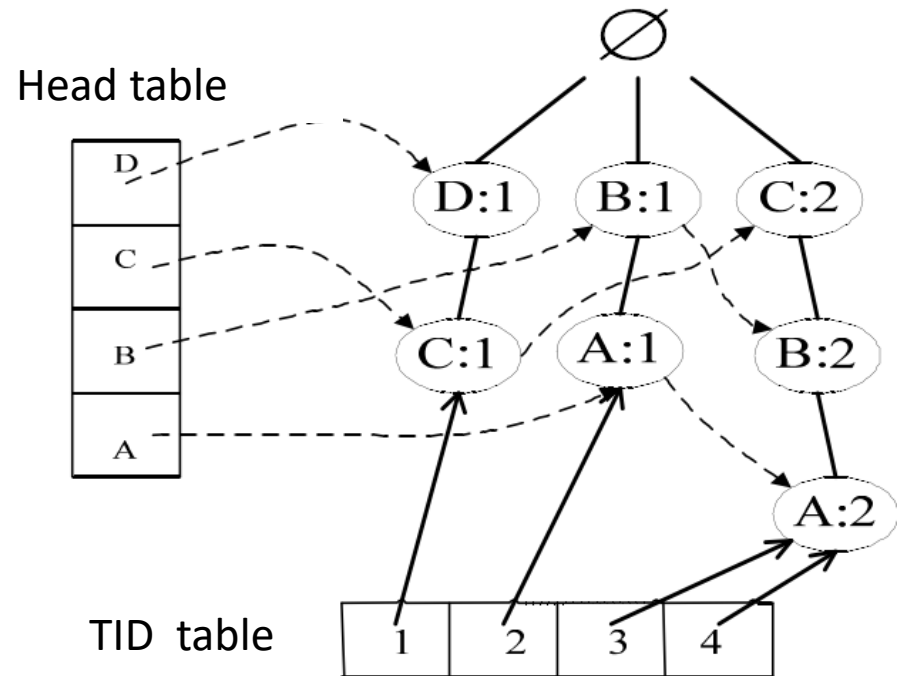
➤ Steps

(1) Initialize **FP-tree** structure and **hashTable** according to the sliding window size.

time line	tid	items
	1	C,D
	2	A,B
	3	A,B,C
	4	A,B,C
	5	A,C,D
	6	B,C

Timeline diagram showing three sliding windows of size 3:

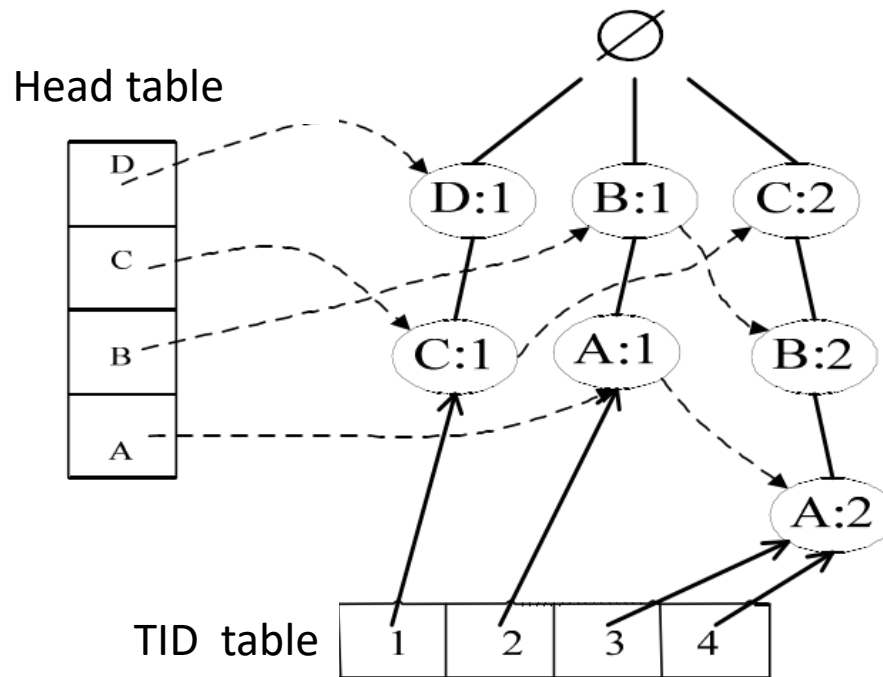
- Window #1: transactions 1, 2, 3
- Window #2: transactions 2, 3, 4
- Window #3: transactions 3, 4, 5



we use a **hash table** to maintain closed itemsets: 1) the itemset I itself, 2) the node type of n I, 3) support: the number of transactions in which I occurs, and 4) tid sum. However, the set of tids take too much space, so we instead use **(support, tid sum)** as the key. Note that tid sum of an itemset **can be incrementally updated**.

➤ Steps

(1) Initialize **FP-tree** structure and **headTable** according to the sliding window size.

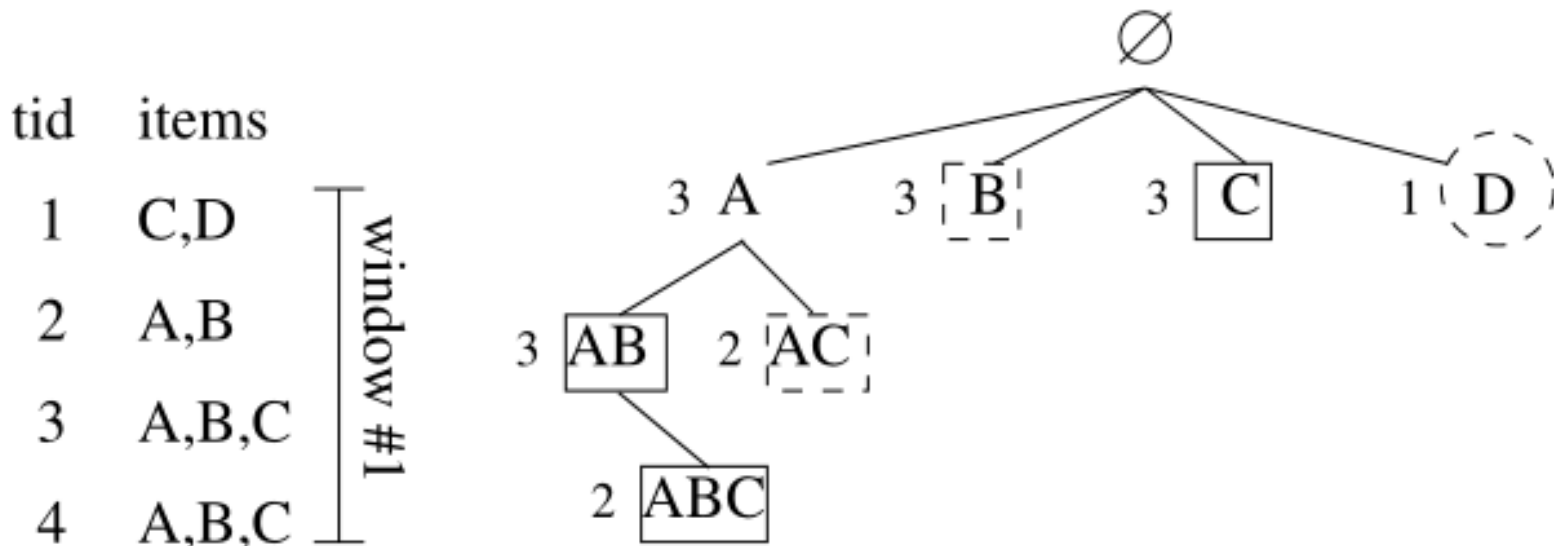


New principles:

- Sequential storage
- Horizontal pointer
- Do not cut off infrequent items
- Serial number table
- Tail node
- Insert a new transaction from the end
- Remove an old transaction from the front
- From the tail node to the root node to follow the path, update the path of the counter
- update FP-tree and headtable

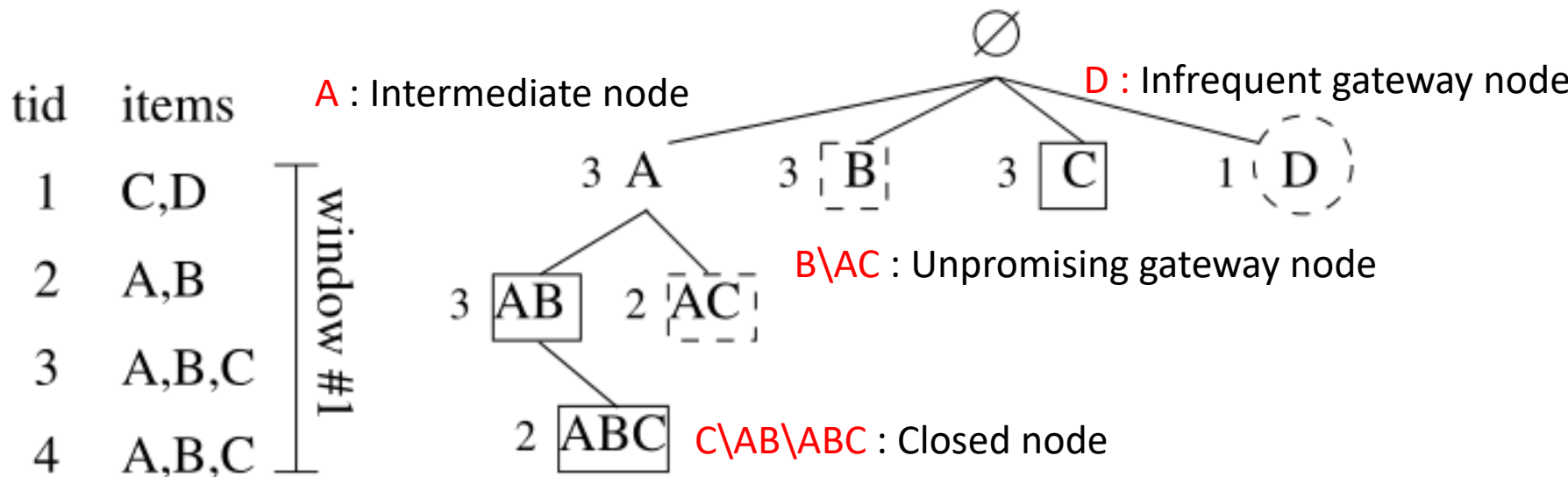
➤ Steps

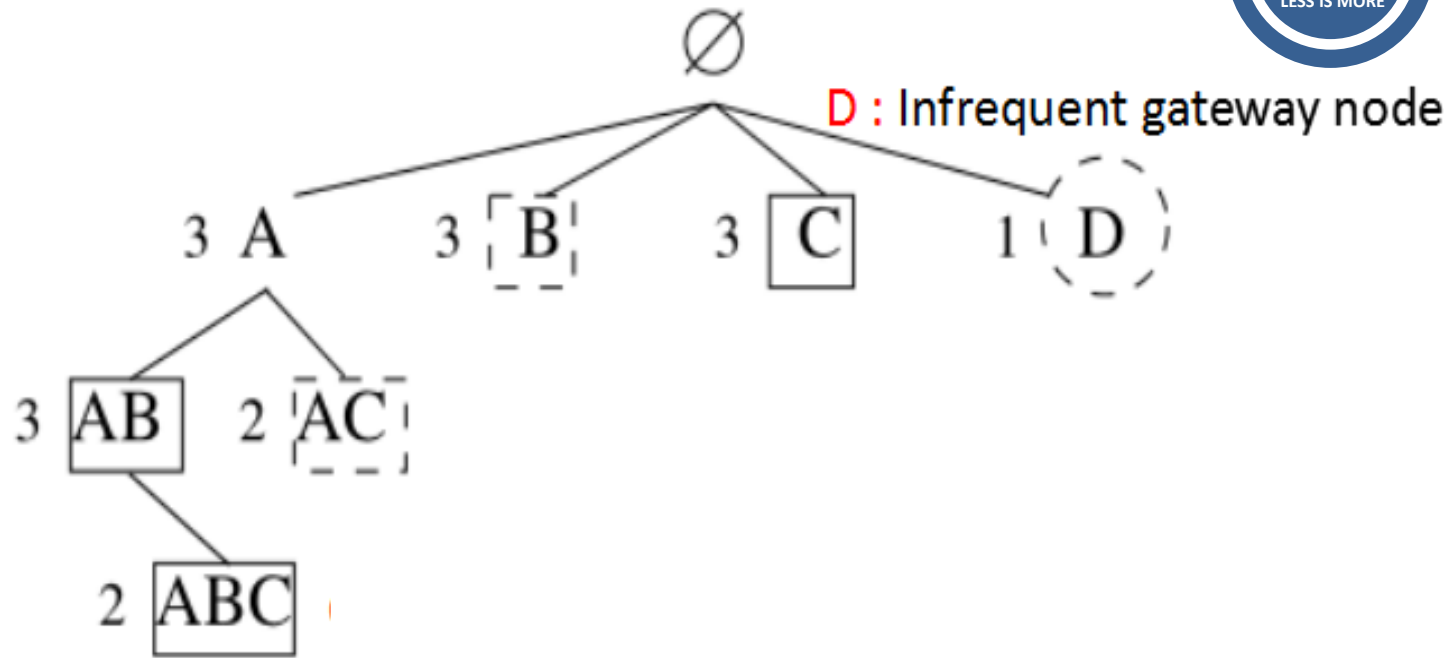
(2) Initialize the **CET structure** based on the headTable and the index of the individual project nodes pointing to the FP-tree structure.



➤ Steps

(3) In the CET structure nodes will be divided into **four categories** of nodes, mining closed frequent pattern of nodes and storage them.

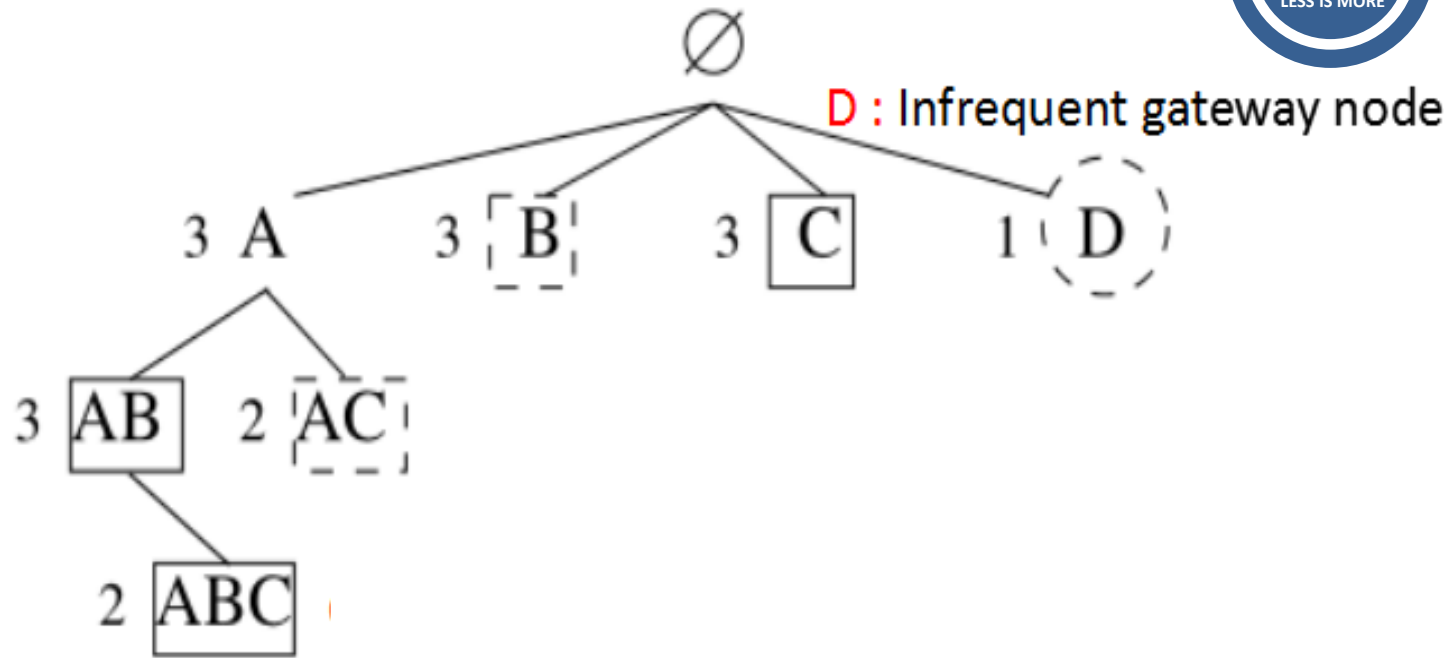




Infrequent gateway nodes (dashed circle box)

① I is a non-frequent itemset;

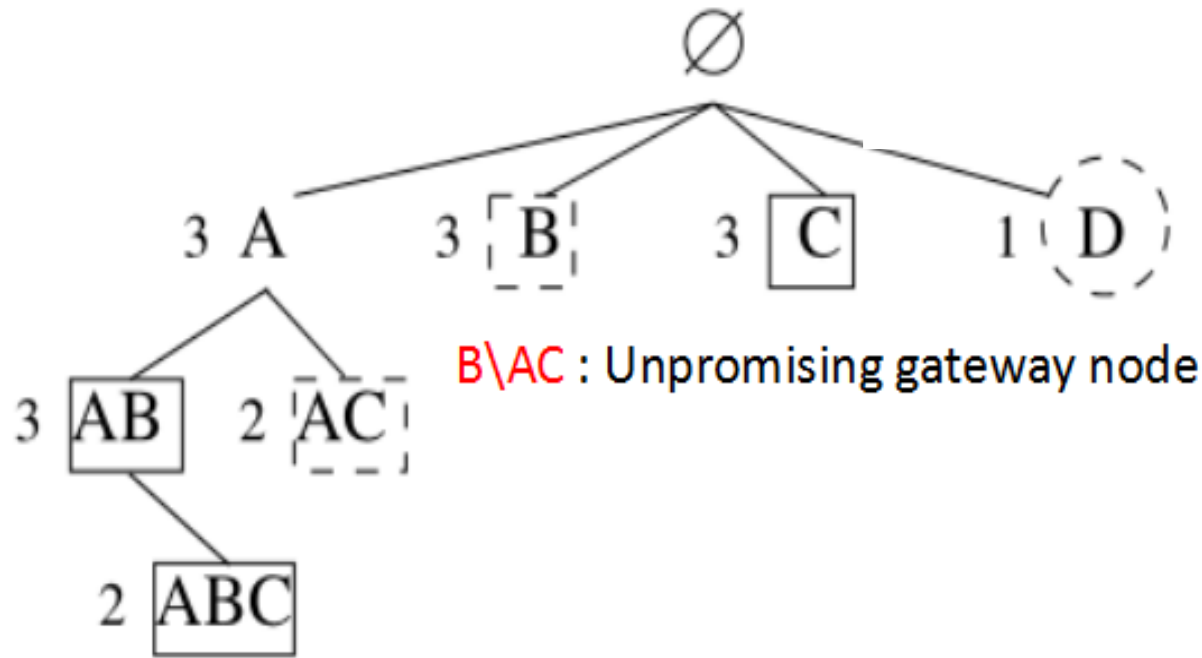
② the parent node of I_n and the parent node of its parent node are frequent.



➤ **Infrequent gateway nodes** (dashed circle box)

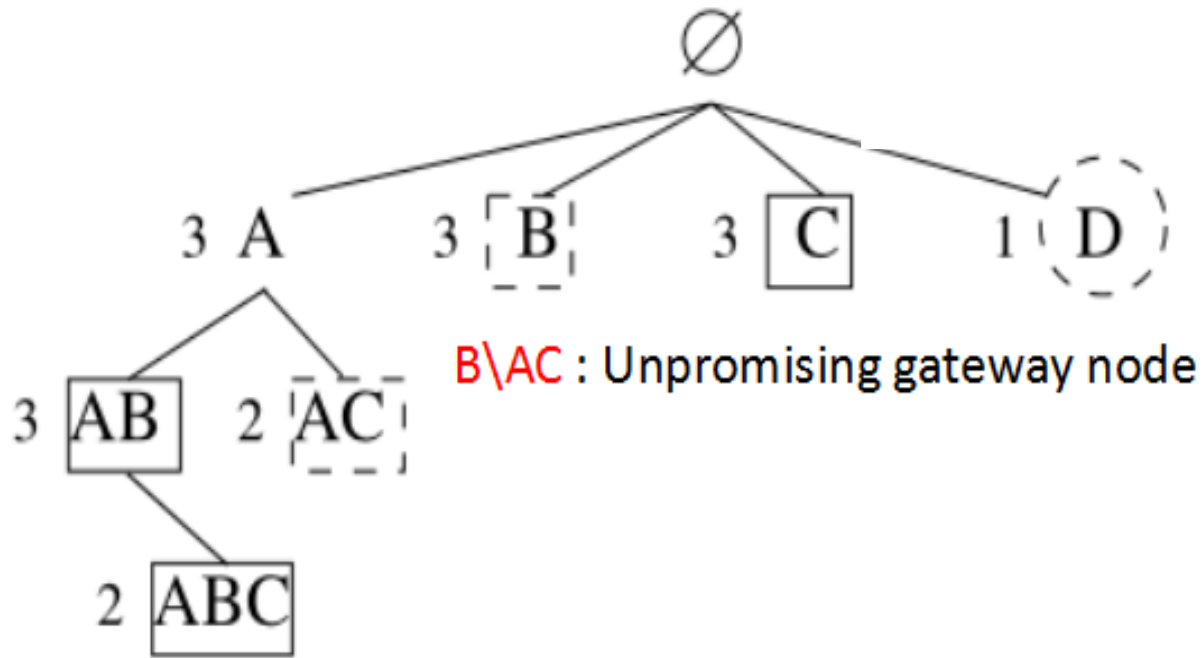
- If is an **infrequent gateway node**, then any node n_j where $J \supset I$ represents an infrequent itemset.

Moment algorithm



Unpromising gateway node (Dashed box)

- ① I is a frequent itemset
- ② there exists a closed frequent itemset J such that $J < I$, $J \supset I$, and J has the same support as I does



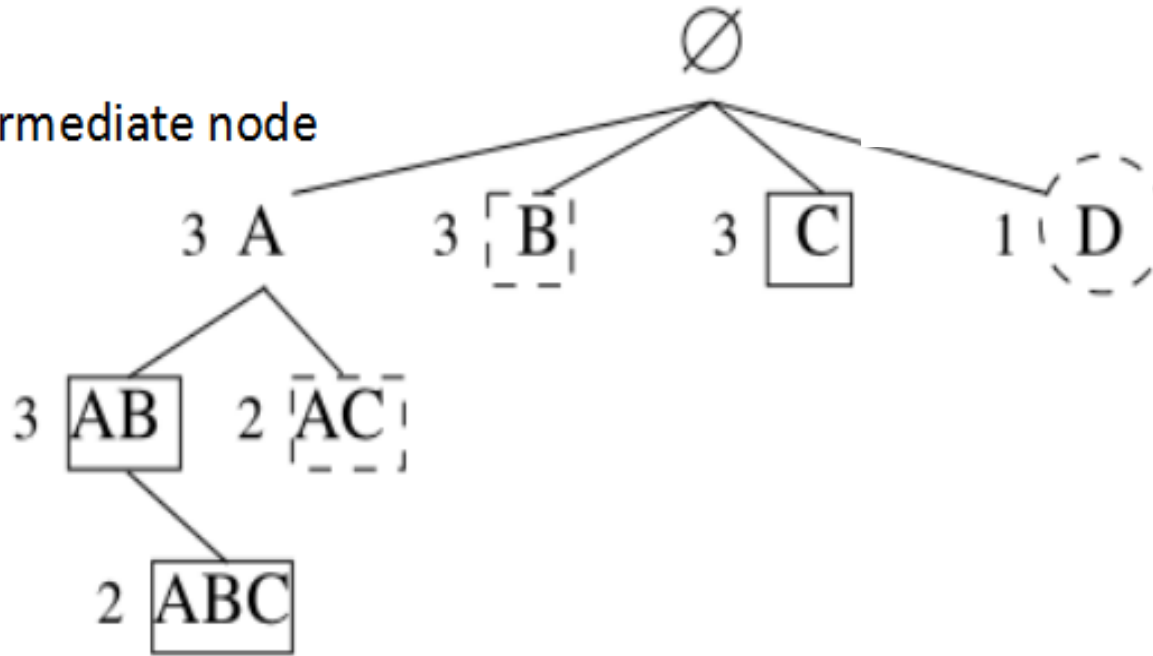
➤ **Unpromising gateway node** (Dashed box)

- If n_i is an **unpromising gateway node**, then n_i is not closed, and none of n_i 's descendants is closed.

Moment algorithm



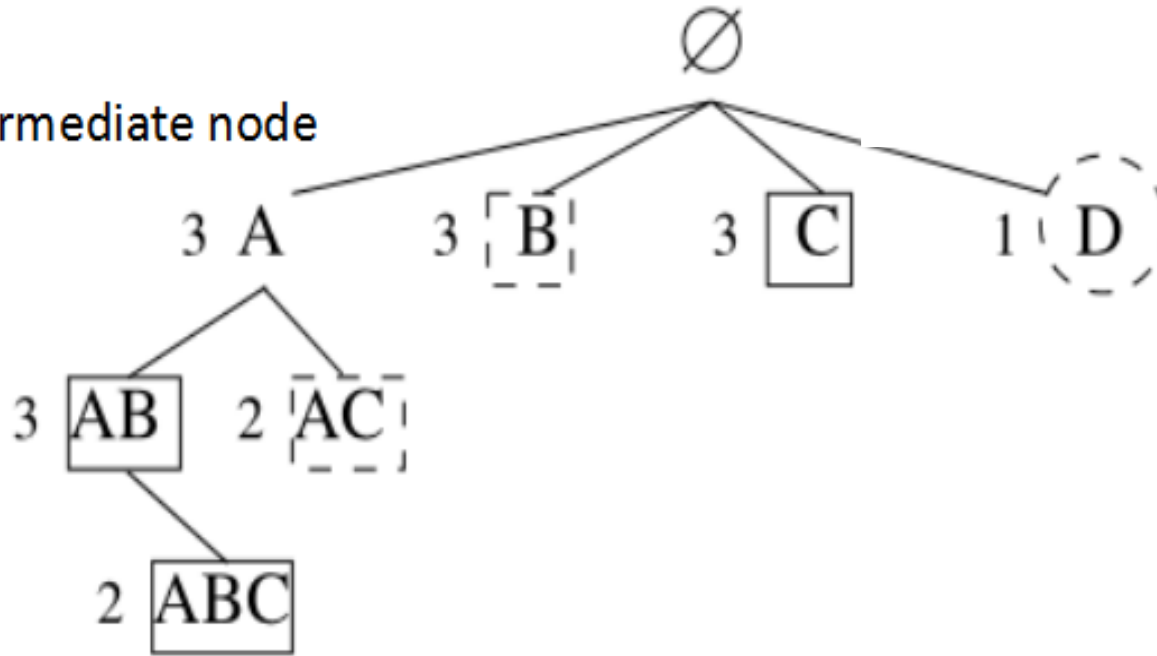
A : Intermediate node



Intermediate node

- ① I is a frequent itemset
- ② n_I has a child node n_J such that J has the same support as I does
- ③ n_I is not an unpromising gateway node.

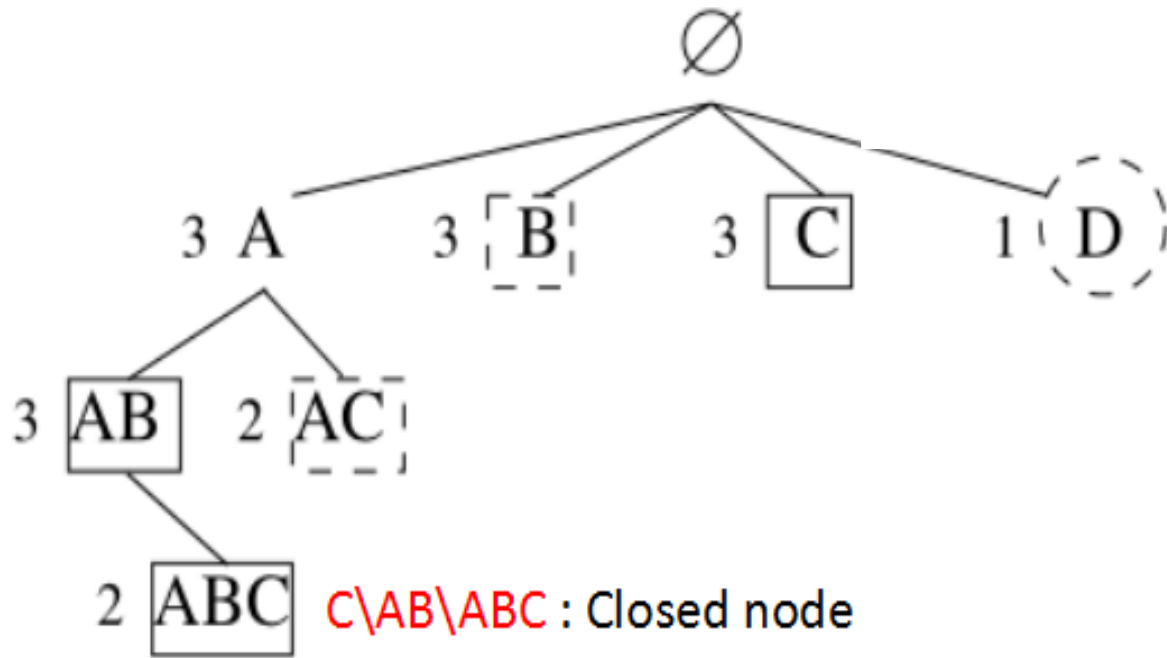
A : Intermediate node



➤ Intermediate node

- If n_i is an **intermediate node**, then n_i is not closed and n_i has closed descendants.

Moment algorithm

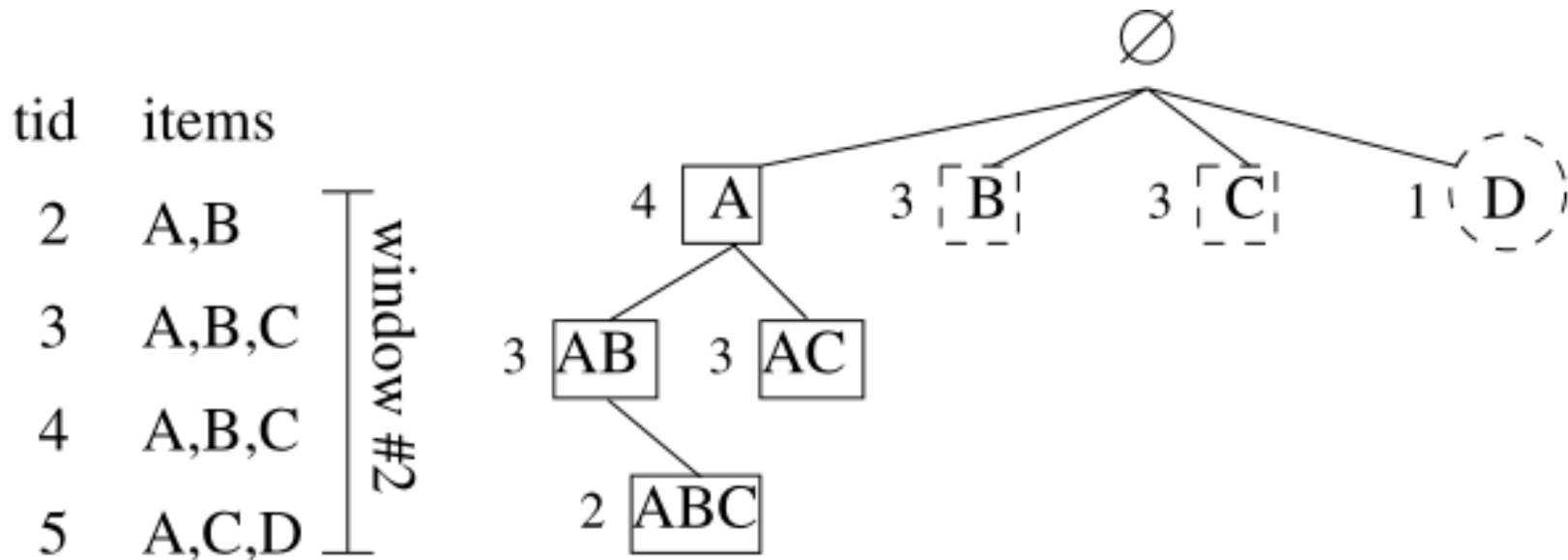


➤ **Closed nodes**(solid rectangles)

- $J < I, J \supset I$, and $\text{support}(J) = \text{support}(I)$.

➤ STEPS

(4) **Delete** old transactions, **update** FP-tree structure and CET structure.

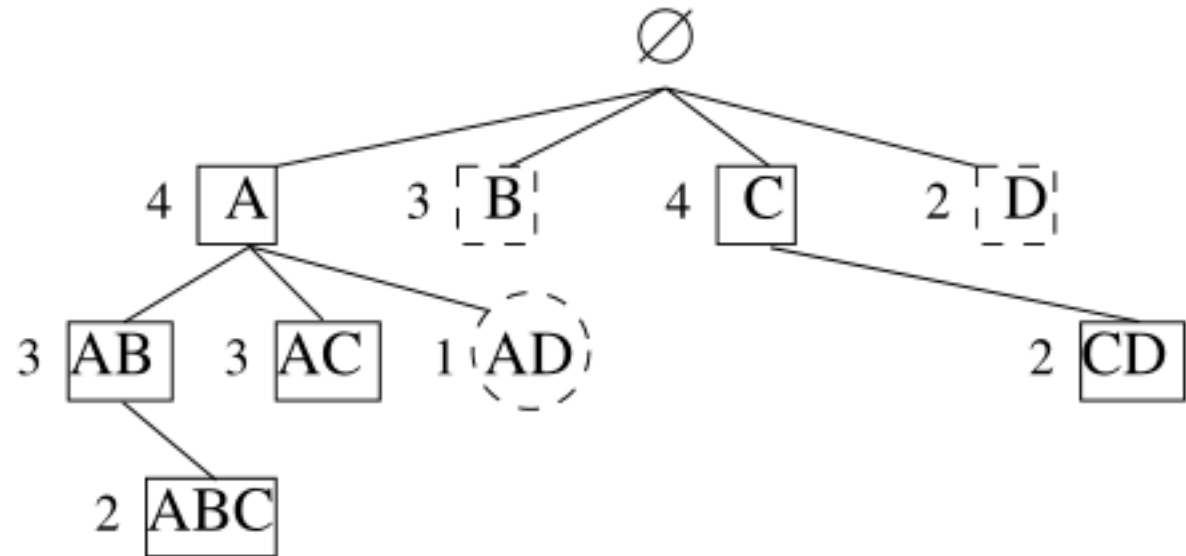


Deleting an old transaction will not change a node in the CET from non-closed to closed, and therefore it will not increase the number of closed itemsets in the sliding-window.

➤ STEPS

(5) **Adding** a new transaction, **update** the FP-tree structure and CET structure.

tid	items
1	C,D
2	A,B
3	A,B,C
4	A,B,C
5	A,C,D



Adding a new transaction will not change a node from closed to non-closed, and therefore it will not decrease the number of closed itemsets in the sliding-window.

➤ Drawbacks

- (1) storage space and time overhead is too large
- (2) It's easy to have a data turbulence problems

Thanks

