FP-Growth

Fatima Radi

Farah Al-Tufaili

University of Kufa – facility of computer science and math

computer science department 2015

First let’s look back to The Apriori Algorithm

- Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases.
- It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.
- The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

The Apriori Algorithm — Example

Database D

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

Scan D $\rightarrow$

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

$\rightarrow$

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D $\leftarrow$

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

$\leftarrow$

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D $\rightarrow$

L_3

itemset	sup
{2 3 5}	2

Is Apriori fast enough?

- Basics of Apriori algorithm
  - Use frequent (k-1)-itemsets to generate k-itemsets candidates
  - Scan the databases to determine frequent k-itemsets

- It is costly to handle a huge number of candidate sets

- If there are 10^4 frequent *1-itemsts*, the Apriori algorithm will need to generate more than 10^7 *2-itemsets* and test their frequencies

Solving Apriori Algorithm Problems? Another Method for Frequent Itemset Generation

- ECLAT
- FP- Growth
- AprioriDP
- Context Based Association Rule Mining Algorithm
- Node-set-based algorithms
- GUHA procedure ASSOC
- OPUS search

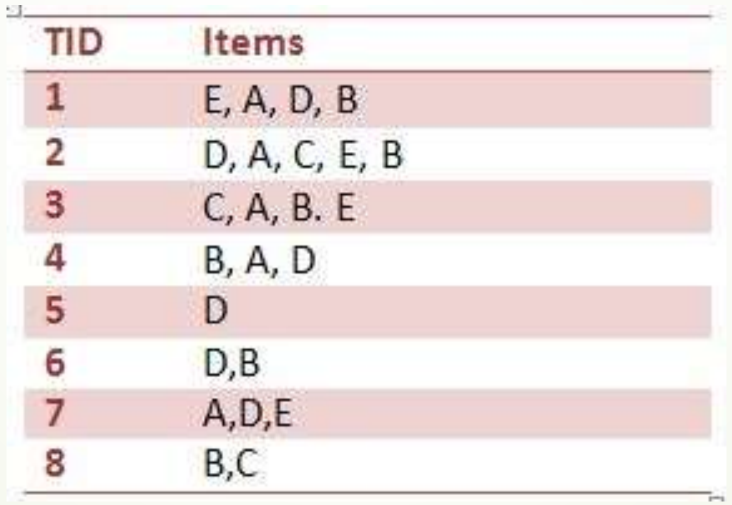
What Is FP-Growth

- An efficient and scalable method to complete set of frequent patterns.
- It allow frequent item set discovery without candidate item set generation.
  - Two step approach:

  1. Build a compact data structure called the FP-Tree.
  2. Extracts frequent item set directly from the FP-Tree.

FP tree example (How to identify frequent patterns using FP tree algorithm

Suppose we have the following DataBase



TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B. E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C

Step 1 - Calculate Minimum support

- First should calculate the minimum support count. Question says minimum support should be 30%. It calculate as follows:

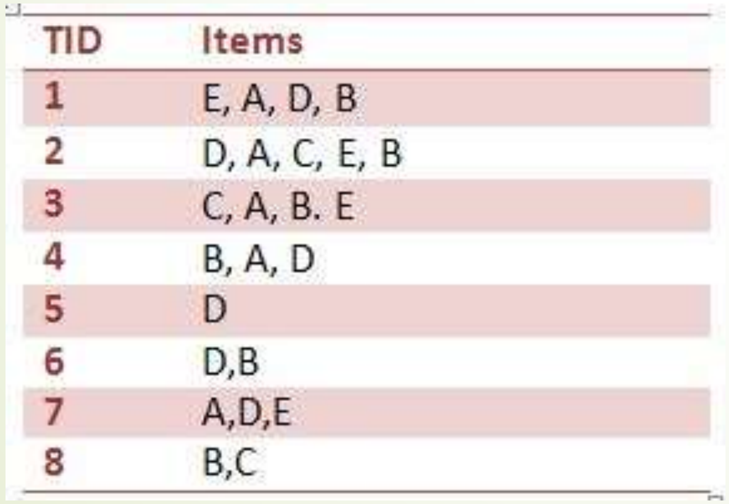
  Minimum support count(30/100 * 8) = **2.4**

  As a result, 2.4 appears but to empower the easy calculation it can be rounded to to the ceiling value. Now,

  Minimum support count is **ceiling**(30/100 * 8) = **3** - See more at: http://hareenlaks.blogspot.com/2011/06/fp-tree-example-how-to-identify.html#sthash.4yIugLAd.dpuf

Step 2 - Find frequency of occurrence

- Now time to find the frequency of occurrence of each item in the Database table. For example, item A occurs in row 1,row 2,row 3,row 4 and row 7. Totally 5 times occurs in the Database table. You can see the counted frequency of occurrence of each item in Table 2



TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B. E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C

Table 1 - Snapshot of the Database

TID	frequency
A	5
B	6
C	3
D	6
E	4

Table2 -Frequency of Occurrence

Step 3 - Prioritize the items

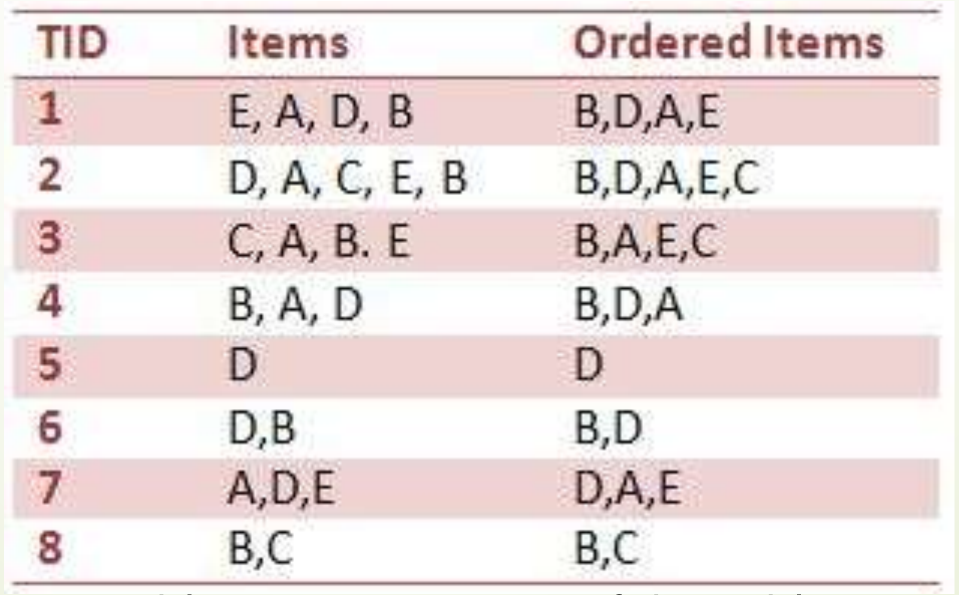
- In Table 2 you can see the numbers written in Red pen. Those are the priority of each item according to it's frequency of occurrence. Item B got the highest priority (**1**) due to it's highest number of occurrences. At the same time you have opportunity to drop the items which not fulfill the minimum support requirement. For instance, if Database contain **F** which has frequency 1, then you can drop it.

TID	frequency	priority
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4

Table2 -Frequency of Occurrence

Step 4 -Order the items according to priority

- As you see in the Table 3 new column added to the Table 1. In the Ordered Items column all the items are queued according to it's priority, which mentioned in the Red ink in Table 2. For example, in the case of ordering row 1, the highest priority item is B and after that D, A and E respectively.



TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Table 3 - New version of the Table 1

Step 5 -Order the items according to priority

- As a result of previous steps we got a ordered items table (Table 3). Now it's time to draw the FP tree. We will mention it row by row
- **Row 1:**
  Note that all FP trees have 'null' node as the root node. So draw the root node first and attach the items of the row 1 one by one respectively. (See the Figure 1) And write their occurrences in front of it.

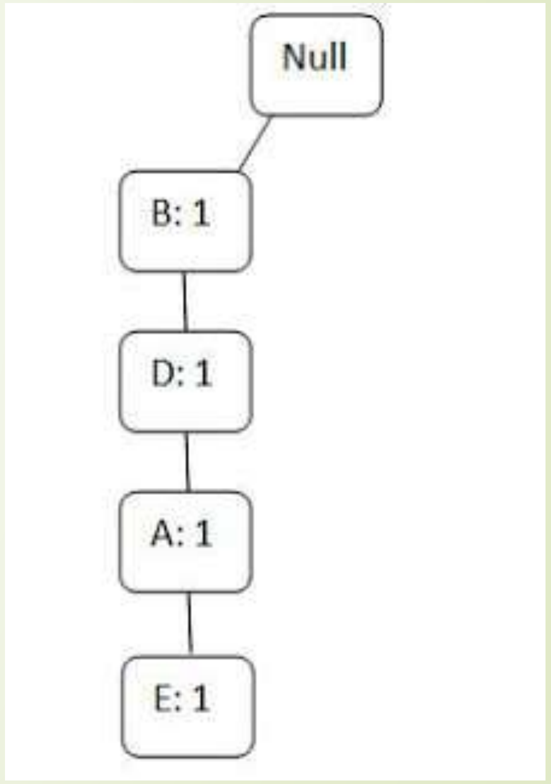


Figure 1- FP tree for Row 1

- Row 2:
  Then update the above tree (Figure 1) by entering the items of row 2. The items of row 2 are B,D,A,E,C. Then without creating another branch you can go through the previous branch up to E and then you have to create new node after that for C. This case same as a scenario of traveling through a road to visit the towns of the country. You should go through the same road to achieve another town near to the particular town.
  When you going through the branch second time you should erase one and write two for indicating the two times you visit to that node. If you visit through three times then write three after erase two

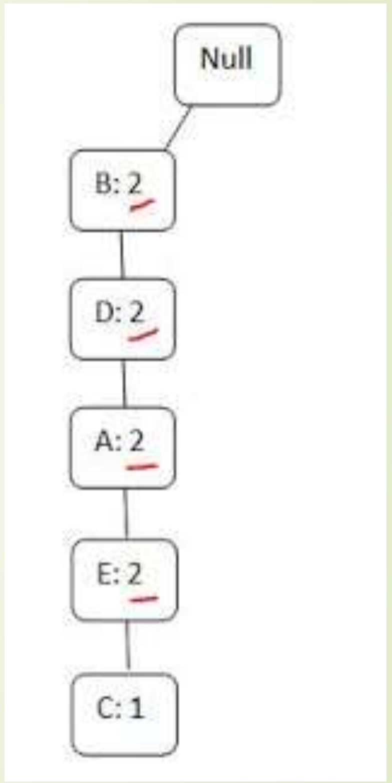


Figure 2- FP tree for Row 1,2

- **Row 3:**
  In row 3 you have to visit B,A,E and C respectively. So you may think you can follow the same branch again by replacing the values of B,A,E and C . But you can't do that you have opportunity to come through the B. But can't connect B to existing A overtaking D. As a result you should draw another A and connect it to B and then connect new E to that A and new C to new E.

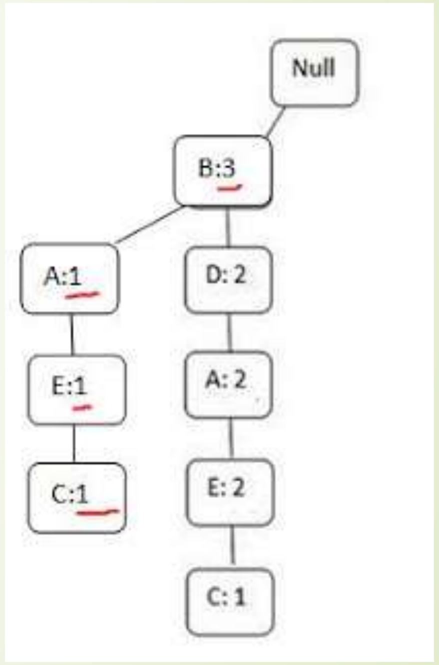


Figure 3 - After adding third row

- **Row 4:**
  Then row 4 contain B,D,A. Now we can just rename the frequency of occurrences in the existing branch. As B:4,D,A:3.
- **Row 5:**
  n fifth raw have only item D. Now we have opportunity draw new branch from 'null' node. See Figure 4.

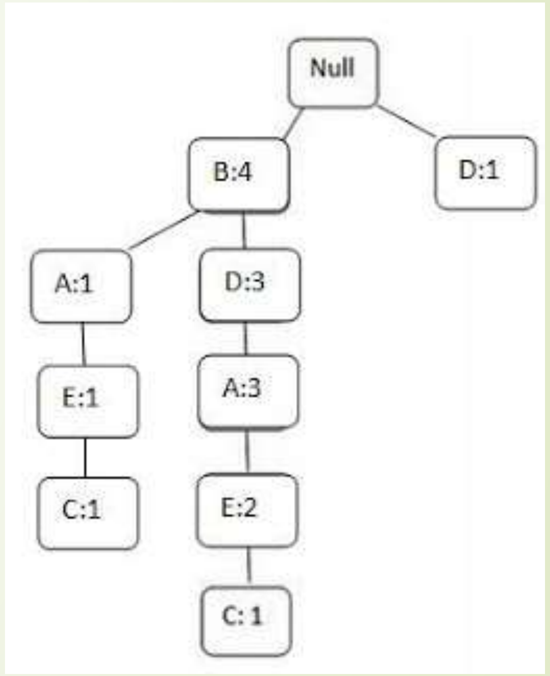


Figure 4- Connect D to null node

- **Row 6:**
  B and D appears in row 6. So just change the B:4 to B:5 and D:3 to D:4.

- **Row 7:**
  Attach two new nodes A and E to the D node which hanging on the null node. Then mark D,A,E as D:2,A:1 and E:1.

- **Row 8 :(Ohh.. last row)**
  Attach new node C to B. Change the traverse times.(B:6,C:1

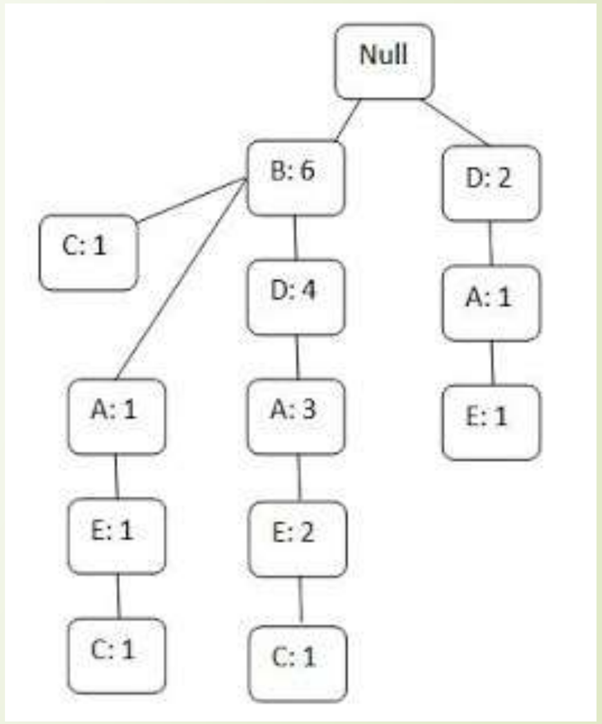


Figure 5 - Final FP tree

Step 6 - Validation

- After the five steps the final FP tree as follows: Figure 5.

  How we know is this correct?

  Now count the frequency of occurrence of each item of the FP tree and compare it with Table 2. If both counts equal, then it is positive point to indicate your tree is correct.

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - There exists examples of databases, where compression ratio could be over 100

FP-Growth Complexity

- Therefore, each path in the tree will be at least partially traversed the number of items existing in that tree path (the depth of the tree path) * the number of items in the header.
- Complexity of searching through all paths is then bounded by O(header_count^2 * depth of tree)