Mining Frequent Patterns Without Candidate Generation  
Paper Review

In data mining research, mining frequent patterns in different types of databases using Apriori-like candidate set generation and test approach requires a huge cost especially when prolific or long patterns exist. During the process of finding pattern of 104 frequent item sets, Apriori algorithm generates more than 107 length-2 candidates. Besides that, discovering 100 size frequent pattern, more than 1030 candidates need to be generated. These numbers are very big when this technique is applied and definitely cost a long time to generate the output pattern. This problem leads to the discovery of what caused the high cost of candidate generation. The cause is located at the candidate set generation and test process. By avoiding the generation of huge set of candidates, the cost can be significantly reduced. The writer proposed a solution by using frequent pattern tree structure (FP-tree). FP-tree structure is expected to achieve efficiency of mining by using three techniques. The first technique is to compress large database into a highly condensed and much smaller data structure. The second technique is to adopt a pattern fragment growth method to avoid the high cost generation of large number candidate set. While the last technique is by using divide and conquer method to decompose mining task into a set of smaller task. Using this method, mining frequent patterns is much more efficient and scalable.

FP Tree requires two main step in order to produce an output. The first step is constructing the FP Tree. This step requires a database as an input and results in a frequent pattern tree. There are 2 method of producing FP Tree. The first one by scanning the database while collecting the set of frequent items and their supports. Then the frequent items are sorted in descending order L. The second process is creating the root of the FP-Tree and labeling it as null. A function called insert\_tree contains two conditions. The first condition is If T has a child N such that N.item-name = p.item-name, then increment N 's count by 1. The second condition is create a new node N , and let its count be 1. Since FP-Tree is a highly compact structure tree, the size of the tree is smaller than the size of the database. Besides that, since the frequent items are sorted according to support-descending order, more frequently occurring items are arranged closer to the top of the FP-Tree. This makes the FP-Tree compact and easier to find the pattern.

The second step is mining frequent pattern with FP-Tree by pattern fragment growth. The input of this step is the result of the first step. While the output of this step is the complete set of frequent pattern. The method of obtaining the output is by calling FP-growth function based on the paper. Since the FP-growth algorithm involves divide and conquer, the scale of shrinking is usually significant. Even though the database may generate an exponential number of frequent patterns, the size of FP-tree will still be small and never grow exponentially because of divide and conquer. It can be said that this way, the process of obtaining complete set of frequent pattern is much more efficient than Apriori algorithm.

Based on the paper, it can be concluded that FP-Tree method is much more faster and efficient than Apriori in order to mine frequent pattern. Th FP-Tree constructed is highly compact which means the size is smaller than the original database. This way, scanning the data will require shorter time. FP-Tree also uses pattern growth method which avoid costly candidate generation and test by successively concatenating frequent item set. Last but not least FP-Tree uses divide and conquer method which is proven to be very efficient. The paper suggested to explore FP-Tree implementation in SQL-based database, highly scalable FP-Tree and the extension of FP-Tree based mining method for mining sequential patterns, max-patterns and partial periodicity.