**Chapter Four**

**Related Works**

**4.1 : Introduction**

This chapter contains some of previous research and related works in the same association rules field and reviews quickly three topics about advanced algorithms have been developed of Apriori algorithm. The paper [5] by Rakesh Agarwal and Ramakrishnan Srikant is presented in Sections 4.2 and 4.3, and the paper [6] by Unil Yun and John J.Leggett is presented in Section 4.4. At the end there is a conclusion summary the editors findings of their papers.

**4.2 : Apriori-TID Algorithm**

Apriori TID has the same candidate generation function as Apriori. The interesting feature is that it does not use database for counting support after the first pass. An encoding of the candidate itemsets used in the previous pass is used. In later passes the size of encoding can become much smaller than the database,thus saving reading effort. [5]

**4.3 : Apriori-Hybrid Algorithm**

Apriori and AprioriTid use the same candidate generation procedure and therefore count the same itemsets Apriori examines every transaction in the database. On the other hand, rather than scanning the database, AprioriTid scans candidate itemsets used in the previous pass for obtaining support counts. Apriori Hybrid uses Apriori in the initial passes and switches to AprioriTid when it expects that the candidate itemsets at the end of the pass will be in memory. [5]

**4.4 : WFIM (Weighted Frequent Itemsets Mining)**

Apriori approaches based on the downward closure property, if any length k pattern is not frequent in a transaction database, superset patterns can not be frequent. Using this characteristic, Apriori based algorithms prune candidate itemsets. However, Apriori based algorithms need to generate and test all candidates. Moreover, they must repeatedly scan a large amount of the original database in order to check if a candidate is frequent or not. This is inefficient and ineffective. [6]

To overcome this problems; FP-tree based methods mine the complete set of frequent patterns using a divide and conquer method to reduce the search space without generating all the candidates. An association mining algorithm generates frequent patterns and then makes association rules satisfying a minimum support. One of the main limitations of the traditional model for mining frequent itemsets is that all the items are treated uniformly, but real items have different importance. For this reason, weighted frequent itemset mining algorithm have been suggested. The items are given different weights in the transaction database. The main focus in weighted frequent itemset mining concerns satisfying the downward closure property. The downward closure property is usually broken when different weights are applied to the items according to their significance. [6]

WFIM adopts an ascending weight ordered prefix tree. The tree is traversed bottom-up because the previous matching can not maintain the downward closure property. A support of each itemset is usually decreased as the length of an itemset is increased, but the weight has a different characteristic. An itemset which has a low weight sometimes can get a higher weight after adding another item with a higher weight, so it is not guaranteed to keep the downward closure property. [6]

**4.5 : Conclusion in Points**

* Both Apriori and AprioriTid need minsup and minconf to be specified in advance. The algorithms have to be rerun each time these values are changed, throwing everything away that was obtained in previous runs. If no appropriate values for these thresholds are known in advance and we want to know how the results change with these values without rerunning the algorithms, the best we can do is to generate and count only those itemsets that appear at least once in the database without duplication and store them all in an efficient way. Note that Apriori generates candidates that do not exist in the database. [2]
* Apriori and almost all other association rule minings use two-phase strategy: first mine frequent patterns and then generate association rules. This is not the sole way. There are another strategies that immediately generates a large subset of all association rules. [2]
* Low support threshold can result in more large itemsets and increase the number of candidate itemsets. [5]
* Apriori makes multiple passes, run time of algorithm may increase with number of transactions. [5]
* Apriori and AprioriTID reduces the number of itemsets to be generated each pass by reducing the number of candidate itemsets. [5]