International Journal of Current Trends in Engineering & Technology
ISSN: 2395-3152
Volume: 01, Issue: 05 (July - August 2015)

To Enhance Projection Scalability of Item Transactions by Parallel and Partition Projection using Dynamic Data Set

Priyanka Soni, Research Scholar (CSE), MTRI, Bhopal, priyanka.soni379@gmail.com
Dhirendra Kumar Jha, MTRI, Bhopal, dhirendrajha@gmail.com

Abstract: ***In the pattern mining FP-growth method is an efficient algorithm to mine long or short frequent patterns. By using compact tree structure and partitioning-based divide-and-conquer searching method is used to reduces the searching costs substantially. But in the analysis process of FP-tree construction, it is a strict serial computing process. The proposed algorithm performance is related to the database size, the sum of frequent patterns in the database. By using distributed parallel and partition computation technique is used to solve this problem. This method apparently increase the costs for exchanging and combining control information at the first time of execution but in the next time for item searching the performance improvement is increases and response time also improve.***

Keywords – Data Mining, Web Usage Mining, Frequent Sequential Patterns, divide-and-conquer, partitioning-based, parallel Projection, AI, Classification.

1. INTRODUCTION

Web mining is the application of machine learning (data mining) techniques. It is web-based data for the purpose of learning or extracting knowledge. It encompasses a wide variety technique, including soft computing. This methodologies can generally be classified into one of three distinct categories: web usage mining, web structure mining, and web content mining examine web page usage patterns in order to learn about a web system's users or the relationships between the documents. In web usage mining the goal is to examine web page usage patterns in order to learn about a web system's users or the relationships between the documents.

Web usage mining is defined as the process of applying data mining techniques to the discovery of usage patterns from web logs data, to identify web users' behavior. In Web mining, data can be collected at the server-side, client-side and proxy servers.

1.1 Clustering Technique

Clustering and classification have been useful and active areas of machine learning research that promise to help us cope with the problem of information overload on the Internet. With clustering the goal is to separate a given group of data items (the data set) into groups called clusters such that items in the same cluster are similar to each other and dissimilar to the items in other clusters.

1.2 K-Mean Algorithm

The K-Means algorithm is one of a group of algorithms called partitioning clustering algorithm**.** The most commonly use partition clustering strategy is based on square error criterion. The general objective is to obtain the partition that, for a fixed number of clusters, minimizes the total square errors. Suppose that the given set of *N* samples in an n-dimensional space has somehow been partitioned into *K*-clusters {C_1, C_2, C_3... C_K}. Each C_K has n_K samples and each sample is in exactly one cluster, so that $\sum n_K = N$, where k=1… K. The mean vector M_k of cluster C_K is defined as the centroid of the cluster

$$M_K = (1/n_k)\sum_{i=1}^{n_k} x_{ik} \qquad \text{------------ (1)}$$

Where x_{ik} is the i^{th} sample belonging to cluster C_K. The square-error for cluster C_K is the sum of the squared Euclidean distances between each sample in C_K and its centroid. This error is also called the within-cluster variation [5]:

$$e_k^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2 \qquad \text{------------ (2)}$$

The square-error for the entire clustering space containing K cluster is the sum of the within-cluster variations

$$E_k^2 = \sum_{k=1}^{K} e_k^2 \qquad \text{------------ (3)}$$

The basic steps of the K-mean algorithm are:

- Select an initial partition with K clusters containing randomly chosen sample, and compute the centroids of the clusters,
- Generate a new partition by assigning each sample to the closest cluster centre,
- Compute new cluster centre as the centroids of the clusters,
- Repeat steps 2 and 3 until optimum value of the criterion function is found or until the cluster membership stabilizes.

II. RELATED WORK

The database projection growth based approach, FreeSpan, was developed. Although FreeSpan outperforms the Apriori based GSP algorithm, FreeSpan may generate any substring combination in a sequence. The projection in FreeSpan must keep all sequences in the original sequence database without length reduction.

The *Partitioned Candidate Common Database* (PCCD) algorithm, by Zaki et al. is essentially a data distribution algorithm supports task parallelism over shared memory distributed machines, and is a Data Distribution version of CCPD algorithm.

PrefixSpan, a more efficient pattern growth algorithm was proposed which improves the mining process. The main idea of PrefixSpan is to examine only the prefix subsequences and project only their corresponding suffix subsequences into projected databases.

In WUM we find the behavior of user either it is registered or not. If a website requires users to sign in before they can start browsing, it will be very easy not only to differentiate between users but also to identify each single user. The problem arises when a website allows visitors to anonymously browse its content, which is common place. In this paper differentiate between visitors activity as a challenging task in the log using common log format.

III. PROPOSED WORK

The Data mining tasks is used to discover different kinds of patterns. There are two types of data mining tasks: descriptive and predictive. Descriptive data mining tasks is used to describe the general properties of the existing data and predictive data mining tasks do predictions based on inference on available data. The Taxonomy of Data Mining tasks are shown by Figure 1.

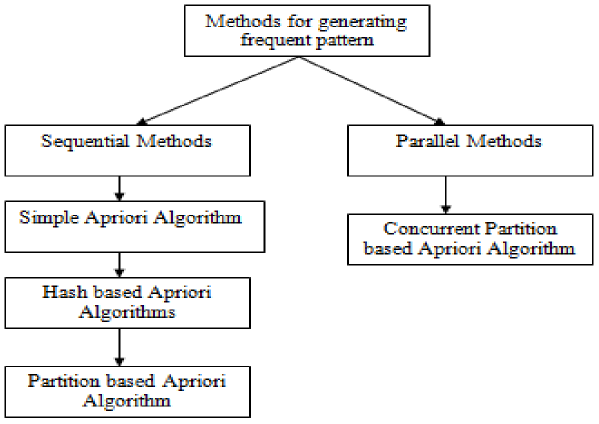

Methods for generating frequent pattern
Sequential Methods
Parallel Methods
Simple Apriori Algorithm
Concurrent Partition based Apriori Algorithm
Hash based Apriori Algorithms
Partition based Apriori Algorithm

Figure 3.1: Data Mining Taxonomy

A partition of the database refers to any subset of the transactions contained in the database D. Any two different partitions are non-overlapping, i.e. any item set that is potentially frequent in DB must be frequent in at least one of the partitions of DB. Partition scans DB only twice:
Scan-1: Partition database and find local frequent patterns.
Scan-2: Consolidate global frequent patterns.

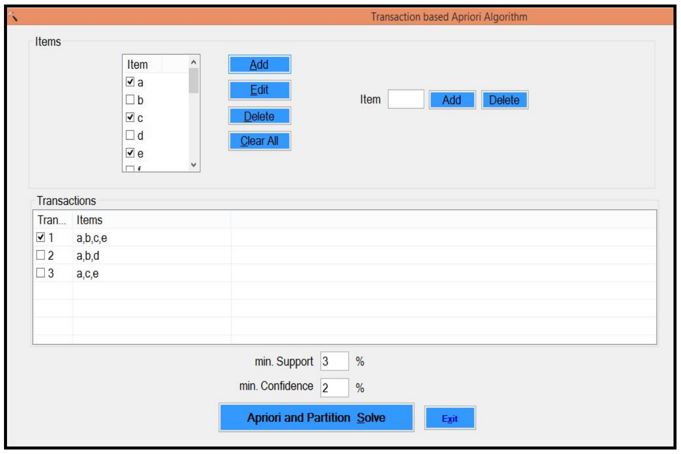
Initially the database *D* is logically partitioned into n partitions.

Phase-I: Read the entire database once, takes *n* iterations
    input: *pi, where i = 1... n.*
    output: local large item sets of all lengths,
        as the output.

Merge Phase:
    input: local large item sets of same lengths from all *n* partitions
    output: combine and generate the global candidate item sets. The set of global candidate item sets of length j is computed as
Phase-II: read the entire database again, takes *n* iterations
    input: *pi, where i = 1... n;*
    output: counters for each global candidate item set and counts their support
Algorithm output: item sets that have the minimum global support along with their support. The algorithm reads the entire database twice.

IV. EXPERIMENTAL RESULT

All the experiments are performed on a 3-GHz Pentium PC machine with 1 GB megabytes main memory, running on Microsoft Windows/NT. All the programs are written in Microsoft Visual Studio 2010. The experiments are pursued on both synthetic and real data sets. This paper showing the result used frequent sequential pattern for behavior of similar pattern access of user.


Transaction based Apriori Algorithm
Items
Add
Edit
Delete
Clear All
Item
Add
Delete
Apriori and Partition Solve
Exit

Item
✓ a
☐ b
✓ c
☐ d
✓ e

Transactions

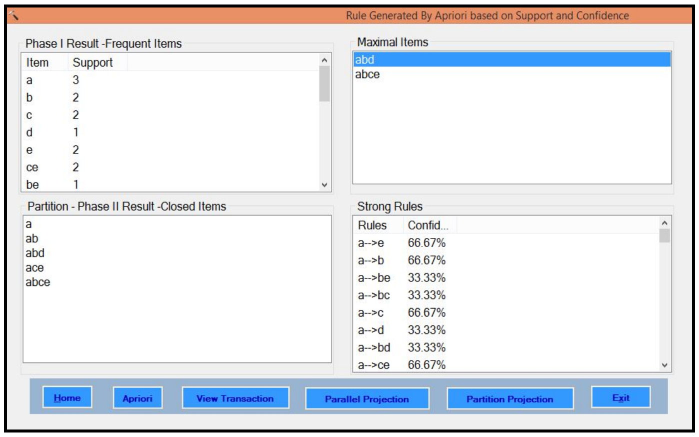
Tran...	Items
✓ 1	a,b,c,e
☐ 2	a,b,d
☐ 3	a,c,e

min. Support 3 %
min. Confidence 2 %

Figure 4.1: Execution of FP-growth with Support and Confidence

The figure 4.1 is showing the output after the execution of the complete algorithm. This snapshot display menu in which user interact easily. A message display for his choice where user select first option after it again a message is display enter the number of transaction to be added after that he give the id of the item.

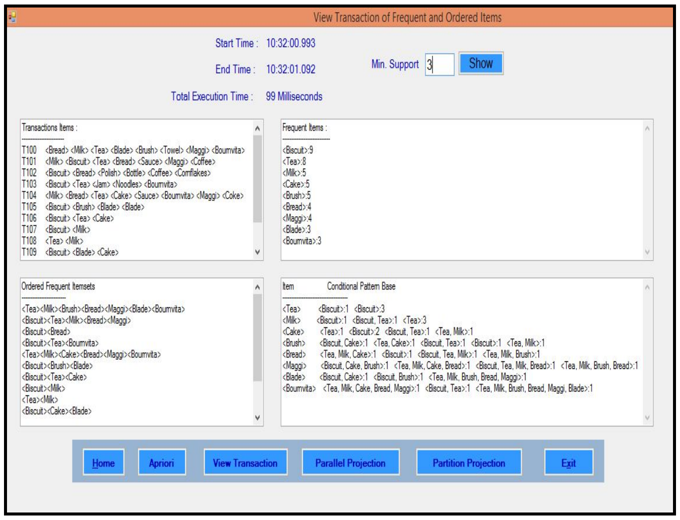
International Journal of Current Trends in Engineering & Technology
ISSN: 2395-3152
Volume: 01, Issue: 05 (July - August 2015)



```
Rule Generated By Apriori based on Support and Confidence

Phase I Result -Frequent Items
Item  Support
a     3
b     2
c     2
d     1
e     2
ce    2
be    1

Maximal Items
abd
abce

Partition - Phase II Result -Closed Items
a
ab
abd
ace
abce

Strong Rules
Rules    Confid...
a-->e    66.67%
a-->b    66.67%
a-->be   33.33%
a-->bc   33.33%
a-->c    66.67%
a-->d    33.33%
a-->bd   33.33%
a-->ce   66.67%
```

Figure 4.2: Rules Generation of Items with Menu Interaction

The figure 4.2 is showing the output after the execution of the complete algorithm. This snapshot display menu in which user interact easily and a message display for his choice. User select second option all the transactions with items and transaction id is displayed.

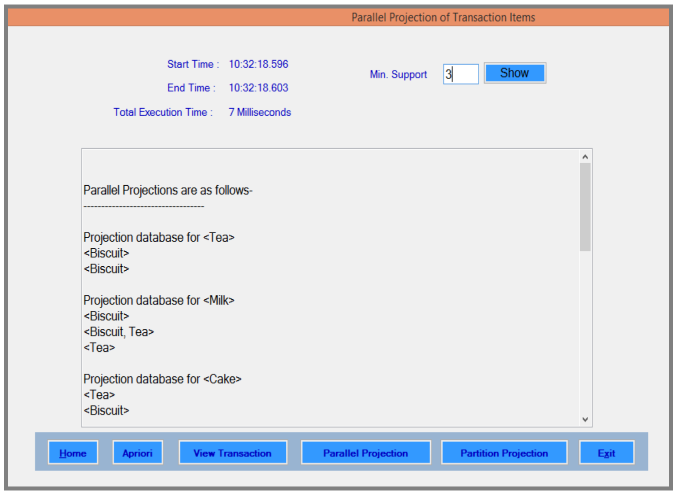


```
View Transaction of Frequent and Ordered Items

Start Time :  10:32:00.993
End Time :    10:32:01.092
Total Execution Time :  99 Milliseconds
Min. Support 3

Transactions Items :
--------------------
T100   <Bread> <Milk> <Tea> <Blade> <Brush> <Towel> <Maggi> <Bournvita>
T101   <Milk> <Biscuit> <Tea> <Bread> <Sauce> <Maggi> <Coffee>
T102   <Biscuit> <Bread> <Polish> <Bottle> <Coffee> <Cornflakes>
T103   <Biscuit> <Tea> <Jam> <Noodles> <Bournvita>
T104   <Milk> <Bread> <Tea> <Cake> <Sauce> <Bournvita> <Maggi> <Coke>
T105   <Biscuit> <Brush> <Blade> <Blade>
T106   <Biscuit> <Tea> <Cake>
T107   <Biscuit> <Milk>
T108   <Tea> <Milk>
T109   <Biscuit> <Blade> <Cake>

Frequent Items :
--------------------
<Biscuit>:9
<Tea>:8
<Milk>:5
<Cake>:5
<Brush>:5
<Bread>:4
<Maggi>:4
<Blade>:3
<Bournvita>:3

Ordered Frequent Itemsets
--------------------
<Tea><Milk><Brush><Bread><Maggi><Blade><Bournvita>
<Biscuit><Tea><Milk><Bread><Maggi>
<Biscuit><Bread>
<Biscuit><Tea><Bournvita>
<Tea><Milk><Cake><Bread><Maggi><Bournvita>
<Biscuit><Brush><Blade>
<Biscuit><Tea><Cake>
<Biscuit><Milk>
<Tea><Milk>
<Biscuit><Cake><Blade>

Item        Conditional Pattern Base
------------------------------
<Tea>       <Biscuit>:1  <Biscuit>:3
<Milk>      <Biscuit>:1  <Biscuit, Tea>:1  <Tea>:3
<Cake>      <Tea>:1  <Biscuit>:2  <Biscuit, Tea>:1  <Tea, Milk>:1
<Brush>     <Biscuit, Cake>:1  <Tea, Cake>:1  <Biscuit, Tea>:1  <Biscuit>:1  <Tea, Milk>:1
<Bread>     <Tea, Milk, Cake>:1  <Biscuit>:1  <Biscuit, Tea, Milk>:1  <Tea, Milk, Brush>:1
<Maggi>     <Biscuit, Cake, Brush>:1  <Tea, Milk, Cake, Bread>:1  <Biscuit, Tea, Milk, Bread>:1  <Tea, Milk, Brush, Bread>:1
<Blade>     <Biscuit, Cake>:1  <Biscuit, Brush>:1  <Tea, Milk, Brush, Bread, Maggi>:1
<Bournvita>  <Tea, Milk, Cake, Bread, Maggi>:1  <Biscuit, Tea>:1  <Tea, Milk, Brush, Bread, Maggi, Blade>:1
```

Figure 4.3: Conditional Pattern Tree Projection with Rule Generation of Transaction

The figure 4.3 is showing the output after the execution of the complete algorithm. This snapshot display menu in which user interact easily and a message display for his choice. User select "View Transaction" option the output show the conditional Pattern of the FP –growth tree.

4.1 TREE WITH DATABASE PARALLEL PROJECTION



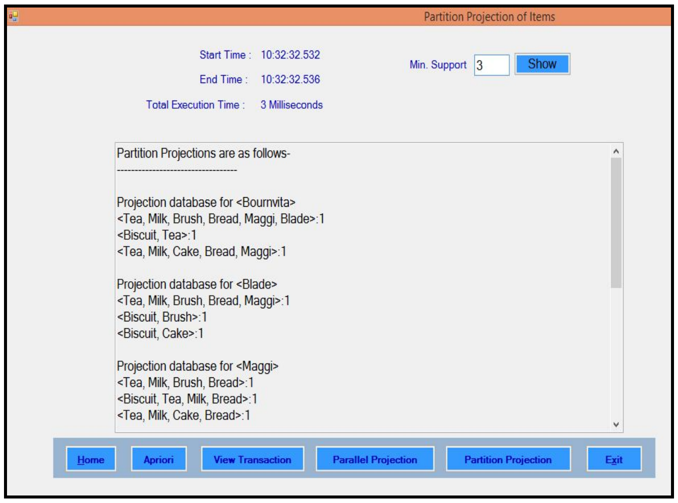
```
Parallel Projection of Transaction Items

Start Time :  10:32:18.596
End Time :    10:32:18.603
Total Execution Time :  7 Milliseconds
Min. Support 3

Parallel Projections are as follows-
------------------------------

Projection database for <Tea>
<Biscuit>
<Biscuit>

Projection database for <Milk>
<Biscuit>
<Biscuit, Tea>
<Tea>

Projection database for <Cake>
<Tea>
<Biscuit>
```

Figure 4.4: Parallel Projection with Transaction-Database using Support of Items

The figure 4.4 is showing the output after the execution of the complete algorithm. This snapshot display menu in which user interacts easily and a message display for his choice when user select a option according to the select option the operation is performed.

4.2 FP-GROWTH TREE WITH DATABASE PARTITION PROJECTION

The figure 4.5 is showing the output after the execution of the complete algorithm. This snapshot display menu in which user interacts easily and a message display for his choice when user select an option according to the select option the operation is performed.



```
Partition Projection of Items

Start Time :  10:32:32.532
End Time :    10:32:32.536
Total Execution Time :  3 Milliseconds
Min. Support 3

Partition Projections are as follows-
------------------------------

Projection database for <Bournvita>
<Tea, Milk, Brush, Bread, Maggi, Blade>:1
<Biscuit, Tea>:1
<Tea, Milk, Cake, Bread, Maggi>:1

Projection database for <Blade>
<Tea, Milk, Brush, Bread, Maggi>:1
<Biscuit, Brush>:1
<Biscuit, Cake>:1

Projection database for <Maggi>
<Tea, Milk, Brush, Bread>:1
<Biscuit, Tea, Milk, Bread>:1
<Tea, Milk, Cake, Bread>:1
```

Figure 4.5: Partition Projection with Transaction-Database using Support of Items

4.3 COMPARISON ON THE BASIS OF EXECUTION TIME AND MINIMUM SUPPORT COUNT AMONG CONDITIONAL PATTERN, PARALLEL

International Journal of Current Trends in Engineering & Technology
ISSN: 2395-3152
Volume: 01, Issue: 05 (July - August 2015)

PROJECTION AND DATA BASE PARTITION PROJECTION OF FP-GROWTH TREE

Number of Records	Time taken to execute (In ms) **FP-Growth Tree with Conditional Pattern**	Time taken to execute (In ms) **FP-Growth Tree with Database Parallel Projection**	Time taken to execute (In ms) **FP-Growth Tree with Database Partition Projection**
2	102	133	152
3	85	123	132
4	60	85	96
5	45	73	79

Table 4.1: Comparison among Conditional Pattern, Database Parallel Projection and Database Partition Projection of FP-Growth Trees with different support

The Table 4.1 shows the comparison among Conditional Pattern, Database Parallel Projection and Database Partition Projection by using different support and we found that execution time for conditional pattern projection is less as compare to parallel and partition projection of FP-Growth Tree.

Figure 4.6 shows the comparison among conditional pattern, database parallel projection and database partition projection of FP-Growth Tree with different support.

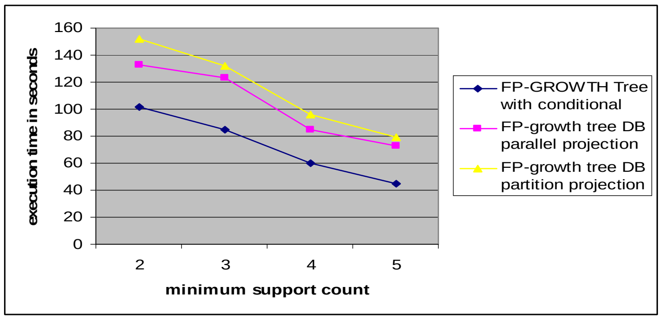


Figure 4.6: Comparison among Conditional Pattern, Parallel Projection and Partition Projection of FP-Tree with different support

4.4 COMPARIOSION ON THE BASIS NUMBER OF RECORDS AND MINIMUM SUPPORT COUNT BETWEEN FP-GROWTH TREE CONDITIONAL PATTERN, FP-GROWTH TREE WITH DATABASE PARALLEL PROJECTION AND FP-GROWTH TREE WITH DATABASE PARTITION PROJECTION

Number of Records	Time taken to execute (In ms) **FP-GROWTH Tree with Conditional Pattern**	Time taken to execute (In ms) **FP-GROWTH Tree with Database Parallel Projection**	Time taken to execute (In ms) **FP-GROWTH Tree with Database Partition Projection**
200	67	85	92
300	70	97	103
400	130	145	157
500	187	197	221

Table 4.2: Comparison among Conditional Pattern, Database Parallel Projection and Database Partition Projection of FP-Growth Trees with number of records

The Table 4.2 shows the comparison among Conditional Pattern, Database Parallel Projection and Database Partition Projection by using number of records and we found that execution time for conditional pattern projection is less as compare to parallel and partition projection of FP-Growth Tree.

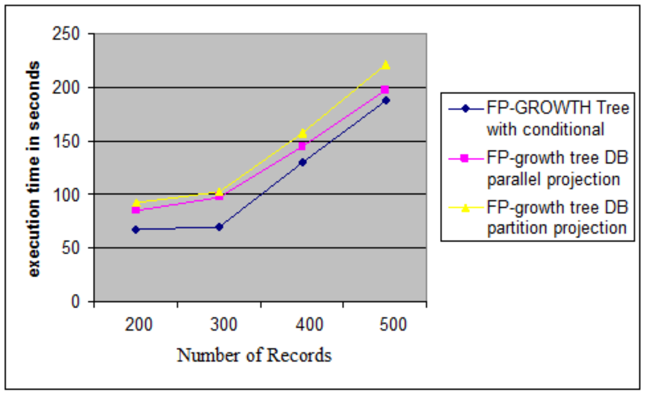


Figure 4.7: Comparison among Conditional Pattern, Database Parallel Projection and Database Partition Projection with different number of records

Figure 4.7 shows the comparison among conditional pattern, database parallel projection and database partition projection of FP-Growth Tree using different number of records. We found that execution time of condition pattern is very low.

V. CONCLUSION

FP-tree achieves good compactness most of the time. Especially in dense datasets, it can compress the database many times. Clearly, there is some overhead for pointers and counters. However, the gain of sharing among frequent projections of transactions is substantially more than the overhead and thus makes FP-tree space more efficient in many cases. When support is very low, FP-tree becomes bushy. In such cases, the degree of sharing in branches of FP-tree becomes low. The overhead of links makes the size of FP-tree large. Therefore, instead of building FP-tree, we should construct projected databases. That is the reason why we build FP-tree for transaction database/projected database only when it passes certain density threshold. From the experiments, one can see that such a threshold is pretty low, and easy to touch. Therefore, even for very large and/or sparse database, after one or a few rounds of database projection, FP-tree can be used for all the remaining mining tasks. In the following experiments, we employed an implementation. Finally FP-Growth Tree is more efficient then tree projection but it is difficult to maintain it in memory so tree projection is used in tree projection two types of projection is used parallel projection is good but it takes more memory but partition projection takes more time is execution but takes less space compare to parallel projection.

.

REFERENCES

[1] Savasere A., Omiecinski E., and Navathe S, ***"An Efficient Algorithm for Mining Association Rules in Large Databases"***, Proceedings of the VLDB Conference, 1995.

[2] Agrawal R. and Srikant R, ***"Fast algorithms for mining association rules"***, VLDB, pp. 487-499, 1994.

[3] Han J., Pei J., and Yin Y, ***"Mining Frequent Patterns without Candidate Generation"***, SIGMOD, pp. 1-12, 2000.

[4] Brin S., Motwani R., Ullman Jeffrey D., and Tsur Shalom, ***"Dynamic itemset counting and implication rules for market basket data"***, SIGMOD, 1997.

[5] Brin S., Motwani R., and Silverstein, C, ***"Beyond market baskets: Generalizing association rules to correlations"***, SIGMOD 26[2], pp. 265-276, 1997.

[6] Antonie M.-L. and Zaïane O. R., ***"Text Document Categorization by Term Association"***, IEEE-ICDM-2002, pp. 19-26, Maebashi City, Japan, December 9 - 12, 2002.

[7] Han J., Pei J., Mortazavi-Asl B., Chen Q., Dayal U., and Hsu M.-C, ***"FreeSpan: Frequent pattern-projected sequential pattern mining"***, ACM SIGKDD, 2000.

[8] Beil F., Ester M., Xu X., ***"Frequent Term-Based Text Clustering"***, ACM SIGKDD, 2002.

[9] Orlando S., Palmerini P., and Perego R, ***"Enhancing the Apriori Algorithm for Frequent Set Counting"***, Proceedings of 3rd International Conference on Data Warehousing and Knowledge Discovery, 2001.

[10] Piatetsky-Shapiro G., Fayyad U., and Smith P., ***"From Data Mining to Knowledge Discovery: An Overview"***, in Fayyad, U., Piatetsky-Shapiro, G., Smith, P., and Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining* AAAI/MIT Press,, pp. 1-35, 1996.

[11] Huang H., Wu X., and Relue R, ***"Association Analysis with One Scan of Databases"***, Proceedings of the 2002 IEEE International Conference on Data Mining, 2002.

[12] Wang K., Tang L., Han J., and Liu J, ***"Top down FP-Growth for Association Rule Mining"***, Proc. Pacific-Asia Conference, PAKDD 2002, pp. 334-340, 2002.

[13] Zaki M. J. and Hsiao C.-J, ***"CHARM: An Efficient Algorithm for Closed Itemset Mining"***, SIAM International Conference on Data Mining, 2002.

[14] Pei J., Han J., Nishio S., Tang S., and Yang D, ***"H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases"***, Proc. 2001 Int. Conf. on Data Mining, 2001.

[15] Pei J., Han J., and Mao R, ***"CLOSET: An efficient algorithm for mining frequent closed itemsets"***, SIGMOD, 2000.

[16] Agrawal R., Aggarwal C. C., and Prasad V. V. V, ***"A Tree Projection Algorithm for Generation of Frequent Itemsets"***, Journal on Parallel and Distributed Computing[(Special Issue on High Performance Data mining)], 2001.

[17] Goulbourne G., Coenen F., and Leng P. H, ***"Computing association rule using partial totals"***, In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 54-66, 2001.

[18] Cheung W., ***"Frequent Pattern Mining without Candidate generation or Support Constraint"***, Master's Thesis, University of Alberta, 2002.

[19] Lin J. L. and Dunham, M. H., ***"Mining association rules: Anti-skew algorithms"***, The 1998 14th International Conference on Data Engineering, pp. 486-493, 1998.

[20] Zaki M. J., Parthsarathy S., Ogihara M., and Li W., ***"New Algorithms for Fast Discovery of Association Rules"***, KDD, pp. 283-286, 2001.

[21] Agrawal R., Imielinski T., and Swami A., ***"Mining association rules between sets of items in large databases"***, In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93), Washington, DC, pp. 207–216, 1993.

[22] Agrawal R., Mannila H., Srikant R., Toivonen H., and Verkamo, ***"Fast discovery of association rules"***, In Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, pp. 307–328, A.I. 1996.