International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 3 | ISSN : 2456-3307

Implementation and Performance analysis UP-Growth for Mining High Utility Itemsets in Transactional Database

Bhavana Lokhande, Prof. Hemlata Dakhore

Department of Computer Science & Engineering, G.H. Raisoni Institute of Engineering & Technology, Nagpur, India

ABSTRACT

Data mining can be described as a development that thinks some learning contained in far reaching exchange databases. Standard data mining procedures have focused, as it were, on finding the things that are more successive in the exchange databases, which is furthermore called visit itemset mining. These data mining procedures relied on upon support conviction show. Itemsets which appear to be more as often as possible in the database must be of all the all the more proposing to the customer from the business viewpoint. In this paper we demonstrate a creating domain called as High Utility Itemset Mining that finds the itemsets considering the repeat of the itemset and also utility associated with the itemset. Each itemset have regard like sum, advantage and other customer's favourable position. This regard associated with everything in a database is known as the utility of that itemset. Those itemsets having utility qualities more vital than given edge are called high utility itemsets. This issue can be recognized as mining high utility itemsets from exchange database. In various areas of expert retail, stock et cetera fundamental initiative is key. So it can help in mining count, the closeness of everything in an exchange database is addressed by a matched regard, without considering its sum or a related weight, for instance, cost or advantage. However sum, advantage and weight of an itemset are important for recognizing certifiable decision issues that require extending the utility in an affiliation. Mining high utility itemsets from exchange database presents a more imperative test as differentiated and regular itemset mining, since unfriendly to monotone property of incessant itemsets is not fitting in high utility itemsets. In this paper, we analyse the performance of UP-Growth for efficient discovery of high utility itemset.

Keywords : High Utility Pattern, Closed High Utility Itemset, Utility Mining, Lossless And Concise Representation, Pattern Mining

I. INTRODUCTION

Finding fascinating examples has been an imperative information mining undertaking, and has an assortment of uses, for instance, genome investigation, condition checking, cross promoting, and stock forecast, where intriguing quality measures [17] assume a critical part. With successive example mining [2], [3], [18] an example is viewed as fascinating if its event recurrence surpasses a user specified limit. For instance, mining incessant examples from a shopping exchange database alludes to the disclosure of sets of items that are often obtained together by clients. In any case, a client's advantage may identify with many components that are not really communicated as far as the event recurrence. For instance, a general store supervisor might be keen on finding mixes of items with high benefits or an income, which identifies with the unit benefits and bought amounts of items that are not considered in incessant example mining.

Utility mining rose as of late to address the impediment of continuous example mining by considering the client's desire or objective and in addition the crude information. Utility mining with the itemset share structure [19], [39], [40], for instance, finding blends of items with high benefits or incomes, is substantially harder than different classes of utility mining issues, for instance, weighted itemset mining [10] and target

arranged utility-based affiliation mining [11]. Solidly, the intriguing quality measures in the last classes watch a hostile to monotonicity property, that is, a superset of a uninteresting example is additionally uninteresting. Such a property can be utilized in pruning look space, which is likewise the establishment of all continuous example mining calculations [3]. Shockingly, the counter monotonicity property does not make a difference to utility mining with the itemset share structure. In this way, utility mining with the itemset share system is more testing than alternate classifications of utility mining and in addition visit design mining.

The majority of the earlier utility mining calculations with the itemset share system [4], [15]adopt a two-stage, competitor era approach, that is, first discover candidates of high utility examples in the primary stage, and after that output the crude information once again to recognize high utility examples from the candidates in the second stage. The test is that the quantity of candidates can be immense, which is the versatility and effectiveness bottleneck.

Despite the fact that a considerable measure of exertion has been made [4], [15] to lessen the quantity of candidates created in the primary stage, the test still continues when the crude information contains many long exchanges or the base utility edge is little. Such countless cause's adaptability issue in the main stage as well as in the second stage, and subsequently corrupts the effectiveness. One exemption is the HUIMiner calculation [28], which is however even less productive than two-stage calculations when mining substantial databases because of wasteful join operations, absence of solid pruning, and versatility issue with its vertical information structure.

To address the test, this paper proposes another calculation, UP-Growth, for utility mining with the itemset share system, which utilizes a few methods proposed for mining incessant examples, including investigating a normal set count in a turnaround lexicographic request and heuristics for requesting things [18]. Our commitments are as per the following:

A high utility example growth approach is proposed, which we contend is one without competitor era on the grounds that while the two-stage, applicant era approach utilized by earlier calculations first produces high TWU designs (candidates) with TWU being a between time, hostile to monotone quantify and after that distinguishes high utility examples from high TWU designs, our approach specifically finds high utility examples in a solitary stage without creating high TWU designs (candidates). The quality of our approach originates from effective pruning strategies in light of tight upper limits on utilities.

A look ahead procedure is joined with our approach, which tries to distinguish high utility examples prior without recursive specification. Such a system depends on a conclusion property and a singleton property, and upgrades the effectiveness in managing thick information.

II. LITERATURE REVIEW

Around there we show a concise review of the unmistakable figurings, techniques, thoughts and systems that have been described in various research journals and circulations. Agrawal, R., Imielinski, T., Swami, A. [1] proposed Frequent itemset mining count that uses the Apriori standard. Standard strategy relies on upon Support-Confidence Model. Support measure is used. A hostile to monotone property is used to decrease the request space. It produces visit itemsets and finds alliance oversees between things in the database. It doesn't perceive the utility of an itemset [1]. Yao, H., Hamilton, H.J., Buzz, C.J. [2] proposed a framework for high utility itemset mining. They aggregate up past work on itemset share measure [2]. This recognizes two sorts of utilities for things, exchange utility and outside utility. They recognized and dismembered the issue of utility mining. Close by the utility bound property and the support bound property. They described the numerical model of utility mining in perspective of these properties. The utility bound property of any itemset gives an upper bound on the utility estimation of any itemset. This utility bound property can be used as a heuristic measure for pruning itemsets as early stages that are not expected that would qualify as high utility itemsets [2]. Yao, H., Hamilton, H.J., Buzz, C.J. [3] proposed a count named Umining and another heuristic based computation UminingH to find high utility itemsets. They apply pruning systems in perspective of the logical properties of utility constraints. Counts are more beneficial than any past utility based mining computation. Liu, Y., Liao, W.K., Choudhary A. [4] proposed a two phase figuring to mine high utility itemsets. They used an exchange weighted utility (TWU) measure to prune the request

space. The counts in light of the confident period and-test approach. The proposed estimation encounters poor execution when mining thick datasets and long cases much like the Apriori [1]. It requires minimum database inspects, generously less memory space and less computational cost. It can without quite a bit of an extend handle broad databases. Erwin, A., Gopalan, R.P., N.R. Achuthan [5] proposed a powerful CTU-Mine Algorithm in perspective of Pattern Growth approach. They introduce a decreased data structure called as Compressed Transaction Utility tree (CTU-tree) for utility mining, and estimation called CTU-Mine for mining high utility itemsets. They show CTU-Mine works more viably than Two Phase for thick datasets and long case datasets. If the breaking points are high, then Two Phase runs respectably snappy appeared differently in relation to CTU-Mine, however when as far as possible gets the chance to be lower, CTUMine beats Two-stage. Erwin, A., Gopalan, R.P., N.R. Achuthan [7] proposed a capable computation called CTU-PRO for utility mining using the case improvement approach. They proposed another limited data portrayal named Compressed Utility Pattern tree (CUP-tree) which builds up the CFP-tree of [11] for utility mining. TWU measure is used for pruning the request space yet it avoids a rescan of the database. They show CTU-PRO works more successfully than Two-stage and CTU-Mine on thick data sets. Proposed computation is in like manner more capable on inadequate datasets at low reinforce edges. TWU measure is an overestimation of potential high utility itemsets, along these lines requiring more memory space and more estimation when appeared differently in relation to the illustration improvement computations. Erwin, R.P. Gopalan, and N.R. Achuthan [14] proposed an estimation called CTU-PROL for mining high utility itemsets from endless datasets. They used the illustration advancement approach [6]. The figuring first finds the broad TWU things in the exchange database and if the dataset is pretty much nothing, it makes data structure called Compressed Utility Pattern Tree (CUP-Tree) for mining high utility itemsets. In case the data sets are excessively enormous, making it impossible to be in any capacity held in central memory, the computation makes subdivisions using parallel projections that can be thusly mined self-rulingly. For each subdivision, a CUP-Tree is used to mine the aggregate course of action of high utility itemsets. The counter monotone property of TWU is used for pruning the chase space of subdivisions in CTU-PROL, yet not in any manner like Two-period of Liu et al. [4], CTU-PROL computation avoids a rescan of the database to choose the genuine utility of high TWU itemsets. The execution of figuring is taken a gander at against the Two-stage count in [4] moreover with CTU-Mine in [5]. The results show that CTU-PROL beats past figuring on both sparse and thick datasets at most support levels for long and short illustrations.

In the second database look at, the figuring finds all the two segment exchange weighted utilization itemsets and it realizes three part exchanges weighted utilize itemsets. The drawback of this figuring is that it encounters level clever cheerful period and test reasoning [18].

J Hu et al developed a count for successive thing set mining that recognize high utility thing blends. The goal of this computation is to find areas of data, portrayed through mixes of a couple of things (guidelines), which satisfy certain conditions as a get-together and help a predefined target work. The high utility case mining issue considered is not the same as past techniques, as it practices control exposure concerning particular qualities and furthermore with respect to the general standard for the mined set, attempting to find social events of such cases that together adds to the most to a predefined target work [19].

Y-C. Li, J-S. Yeh and C-C. Chang proposed a separated thing discarding system (IIDS). In this paper, they discovered high utility itemsets moreover reduced the amount of hopefuls in every database analyze. They recouped gainful high utility itemsets using the mining estimation called FUM and DCG+. In this framework they showed an unrivaled execution than all the past high utility case mining technique. Nevertheless, their figurings still continue with the issue of level shrewd period and test issue of Apriori and it require different database channels [20].

Liu Jian-ping, Wang Ying, Yang Fan-ding et al proposed a count called tree based incremental alliance oversee mining estimation (Pre-Fp). It relies on upon a FUFP (fast upgrade visit illustration) mining strategy. The huge target of FUFP is the re-usage of heretofore mined continuous things while moving onto incremental mining. The advantage of FUFP is that it diminishes the amount of cheerful set in the updating methodology. In FUFP, all associations are bidirectional while in FP-tree, associations are quite recently unidirectional. The advantage of bidirectional

is that it is definitely not hard to incorporate, clear the youth center point without much entertainment. The FUFP structure is used as a commitment to the pre-broad tree which gives positive check differentiate at whatever point little data is added to one of a kind database. It oversees few changes in database if there ought to emerge an event of inserting new exchange. In this paper the figuring orchestrates the things into three characterizations: visit, rare and pre-far reaching. Pre-incomprehensible itemsets has two sponsorships restrain regard i.e. upper and lower edge. The drawback of this approach is that it is monotonous [21].

Ahmed CF, Tanbeer SK, Jeong BS et al made HUC-Prune. In the present high utility illustration mining it deliver a level clever candidate period and test theory to keep up the cheerful case and they require a couple database looks at which is particularly dependent on the contender length. To vanquish this, they proposed a novel tree based candidate pruning system called HUC-tree, (high utility contender tree) which gets the basic utility information of exchange database. HUC-Prune is absolutely free of high utility candidate case and it requires three database compasses to figure the result for utility case. The drawback of this approach is that it is to a great degree difficult to keep up the figuring for greater database check areas [22].

Shih-Sheng Chen et al (2011) proposed a procedure for continuous discontinuous illustration using diverse slightest sponsorships. This is a capable approach to manage find visit case since it relies on upon various base breaking point reinforce in light of continuous event. Each something in exchange is planned by slightest thing support (MIS), and it doesn't hold download conclusion property, rather it uses sorted conclusion property in light of climbing solicitation. By then PFP (discontinuous successive case) estimation is associated which is same as that of FP-advancement where prohibitive illustration base is used to discover visit cases. This computation is more capable to the extent memory space, thusly decreasing the amount of database yields [23].

Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, Young-Koo Lee, and Ho-Jin Choi et al proposed a Single-pass incremental and natural burrowing for finding weighted successive cases. The current weighted regular illustration (WFP) burrowing can't be associated for incremental and canny WFP burrowing moreover for stream data mining since they rely on upon a static database and its require different database analyzes. To thrashing this, they proposed two novel tree structures IWFPTWA (Incremental WFP tree in light of weight rising solicitation) and IWFPTFD (Incremental WFP tree in perspective of diving solicitation) and two new counts IWFPWA and IWFPFD for incremental and natural mining using a single database channel. IWFPFD ensures that any non-candidate thing can't appear before contender thing

III. PROPOSED SYSTEM

We implement and algorithm called UP-Growth, for finding high utility thing sets and keeping up imperative data identified with utility examples inside databases are proposed. We also implement FP-Growth & Apriori algorithm for performance analysis. We first present the proposed information structure, named UP-Tree, and after that depict the proposed calculation, rang Growth, in subtle elements. The system of the proposed strategy comprises of three sections:

1. Construction of UP-Tree
2. Generation of potential high utility itemsets from the UP-Tree by UP-Growth
3. Identification of high utility itemsets from the arrangement of potential high utility itemsets.

Take note of that we utilize another term potential high utility itemsets (PHUIs) to recognize the found examples found by our approach from the HTWUIs since our approach is not in view of the customary system of exchange weighted use mining model. In our proposed display, the arrangement of PHUIs is significantly littler than the arrangement of HTWUIs in stage I. Figure 1 Show the basic system architecture.

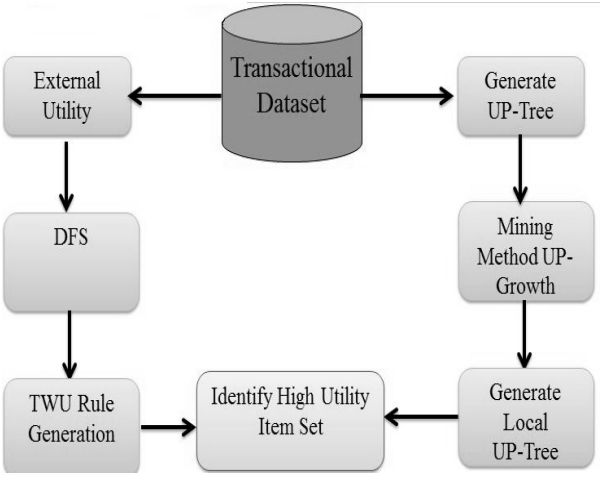

Figure 1: System Architecture

Another high utility example growth approach is proposed (d2HUP), which we contend is one without applicant era. While the two-stage, applicant era approach utilized by earlier calculation first produces high TWU designs (candidates) with TWU being a break, against monotone measure and after that recognizes high utility examples from high TWU designs.

Our approach straightforwardly finds high utility examples in a solitary stage without creating high TWU designs (candidates). A look-ahead procedure is joined with our approach, which tries to distinguish high utility examples prior without recursive specification and upgrades the effectiveness in managing thick information.

IV. IMPLEMENTATION

A. UP-Growth:

In UP-Tree, each node N incorporates N.name, N.count, N.nu, N.parent, N.hlink and an arrangement of kid hubs. The points of interest are presented as takes after. N.name is the thing name of the hub. N.count is the bolster tally of the hub [5]. N.nu is called hub utility which is a gauge utility estimation of the hub. N.parent records the parent hub of the hub. N.hlink is a hub connection which focuses to a hub whose thing name is the same as N.name. Header table is utilized to encourage the traversal of UP-Tree. In the header table, every passage is made out of a thing name, a gauge utility esteem, and a connection. The connection focuses to the last event of the hub which has an indistinguishable thing from the section in the UP-Tree. By taking after the connection in the header table and the hubs in UP-Tree, the hubs whose thing names are the same can be crossed productively.

TID	Transaction	TU
T_1	(A,1) (C,1) (D,1)	8
T_2	(A,2) (C,6) (E,2) (G,5)	27
T_3	(A,1) (B,2) (C,1) (D,6) (E,1) (F,5)	30
T_4	(B,4) (C,3) (D,3) (E,1)	20
T_5	(B,2) (C,2) (E,1) (G,2)	11

Table 1. Example Dataset

Disposing of worldwide unpromising things amid the development of a worldwide UP-Tree

The development of UP-Tree can be performed with two outputs of the first database. In the principal sweep of database, the exchange utility of every exchange is registered. In the meantime, TWU of each single thing is likewise collected. Subsequent to examining database

Item	A	B	C	D	E	F	G
TWU	65	61	96	58	88	30	38

Table 2. Items and their TWUs

When, things and their TWUs are gotten. By TWDC properties, if the TWU of a thing is not as much as least utility limit, its supersets are unpromising to be high utility itemsets. The thing is called unpromising things. Definition [A] gives a formal definition for unpromising things and promising things.

TID	Reorganized transaction	RTU
T_1'	(C,1) (A,1) (D,1)	8
T_2'	(C,6) (E,2) (A,2)	**22**
T_3'	(C,1) (E,1) (A,1) (B,2) (D,6)	**25**
T_4'	(C,3) (E,1) (B,4) (D,3)	20
T_5'	(C,2) (E,1) (B,2)	**9**

Table 3. Reorganized exchanges and their RTUs

Definition [A] (Promising thing and unpromising thing) A thing ip is known as a promising thing if TWU (ip) >=min_utility. Something else, the thing is called an unpromising thing.

After the primary output of database, promising things are sorted out in the header table in the plunging request of TWU qualities. Take note of those different requests can be utilized. In this paper, we propose the TWU sliding request since [2] shows that this request encourages the mining execution. Amid the second sweep of database, exchanges are embedded into UP-Tree. At first, the tree is made with a root R. At the point when an exchange is recovered, unpromising things are expelled from the exchange and their utilities are wiped out from the TU of the exchange since just the supersets of promising things are conceivable to be the high utility itemsets. The staying promising things in the exchange are sorted in the plummeting request of

TWU. The exchange after the above revamping is called rearranged exchange and its TU is called RTU (redesigned exchange utility). The RTU of a revamped exchange Td is signified as RTU (Td).

Illustration 1: Consider the exchange database in Table 1 and the profit table in Table 2. Assume the base utility edge min_util is 40. In the primary output of database, TUs of the exchanges and the TWUs of the things are processed. They are appeared in the last segment of Table 1 and in Table 3, individually. As appeared in Table 3, {F} and {G} are unpromising things since their TWUs are not as much as min_util. The promising things are revamped in the header table in the slipping request of TWU. Table 4 demonstrates the redesigned exchanges and their RTUs for the database in Table 1. As appeared in Table 4, unpromising things {F} and {G} are expelled from the exchanges T2, T3 and T5, individually. Moreover, the utilities of {F} and {G} are disposed of from the TUs of T2, T3 and T5, individually. The staying promising things {A}, {B}, {C}, {D} and {E} in the exchange are sorted in the sliding request of TWU. At that point, we embed redesigned exchanges into the UP-Tree by an indistinguishable procedure from IHUP-Tree [2]. We utilize the accompanying case to portray the operation of inclusion.

Illustration 2: Consider the revamped exchanges in Table 4. The initially rearranged exchange T1' = {C, A, D} prompts to make a branch in UP-Tree. The primary hub {C} is made under the root with {C}.count = 1 and {C}.nu = 8. The second hub {A} is made under hub {A} with {A}.count = 1 and {A}.nu = 8. The third hub {C} is made as an offspring of hub {A} with {C}.count = 1 and {C}.nu = 8. At the point when the following rearranged exchange T2' = {C, E, A} is recovered, the hub utility of the hub {C} is expanded by 22 and {C}.count is expanded by 1. At that point, another hub {E} is made under {C} with {E}.count=1 and {E}.nu = 22. Likewise, another hub {A} is made under the hub {E} with {A}.count=1 and {A}.nu = 22. The redesigned exchanges T3', T4' and T5' are embedded similarly. Subsequent to embedding every revamped exchange, the development of a worldwide UP-Tree with procedure DGU is finished. The worldwide UP-Tree is appeared in Figure 2.

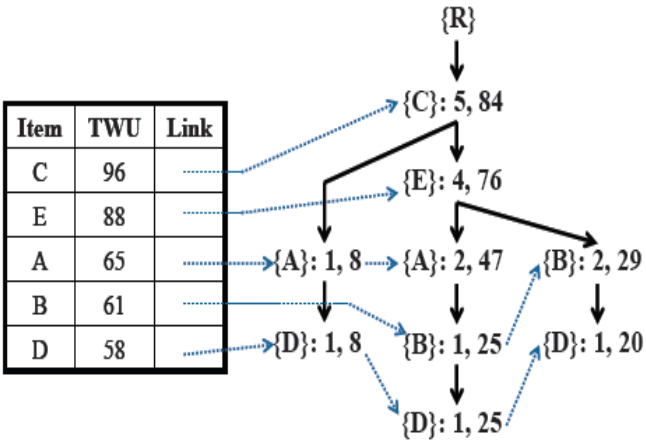

Figure 2: A UP-Tree by applying strategy DGU

Algorithm 1: Pseudo Code for UP-Growth:
Subroutine: UP-Growth (Tx, Hx, X)
Input: A UP-Tree Tx, a header table Hx for Tx and an itemset X.
Output: All PHUIs in Tx.
Procedure UP-Growth (Tx, Hx, X)
(1) For each entry ai in Hx do
(2) Generate a PHUI Y = X □ ai ;
(3) The estimate utility of Y is set as ai's utility value in Hx;
(4) Construct Y's conditional pattern base Y-CPB;
(5) Put local promising items in Y-CPB into Hy
(6) Apply strategy DLU to reduce path utilities of the paths;
(7) Apply strategy DLN and insert paths into Ty;
(8) If Ty not equals null then call UP-Growth (Ty, Hy, Y);
(9) End for

V. EXPERIMENTAL RESULTS

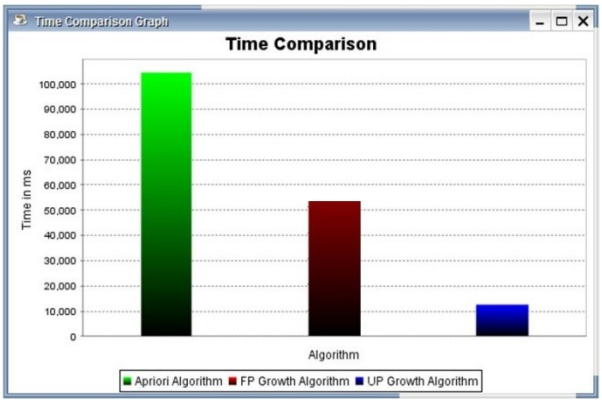

Figure 3: Time Comparison of Apriori, FP-Growth & UP-Growth

Figure 3 shows the performance of the three algorithms with respect to time consumption for execution. Figure justifies that UP-Growth takes lesser time as compared

to other two and reason being the fast and easy accessible UP-Tree Data structure.

In terms of memory consumption also UP-Growth is better in performance. Figure 4 shows the graph for memory utilization of the three algorithms.

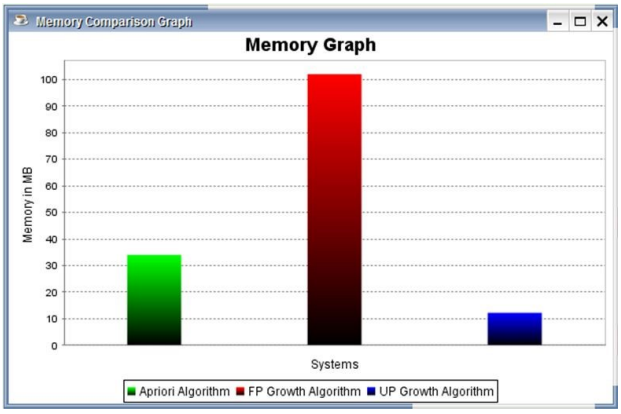

Figure 4: Memory Utilization of Apriori, FP-Growth & UP-Growth

VI. CONCLUSIONS

In this paper, we have proposed an effective calculation named UP-Growth for mining high utility itemsets from exchange databases. An information structure named UP-Tree is proposed for keeping up the data of high utility itemsets. Henceforth, the potential high utility itemsets can be effectively produced from the UP-Tree with just two outputs of the database. Also, we create four methodologies to diminish the assessed utility esteem and upgrade the mining execution in utility mining. In this paper, a distributed and dynamic method is proposed to produce finish set of high utility itemsets from vast databases. Mining high utility itemsets from databases alludes to finding the itemsets with high benefit. In dispersed, it arranges the unpromising things in light of the base utility itemsets from transactions database. This approach makes appropriated environment with one ace hub and two slave hubs examines the database once and numbers the event of everything. The huge database is disseminated to all slave hubs. The worldwide table has the last resultant. Incremental Mining Algorithm is utilized where consistent overhauling continues showing up in a database. At last incremental database is revised and the high utility itemsets is found. Subsequently, it gives speedier execution, that is diminished time and cost.

VII. REFERENCES

[1]. R. Agarwal, C. Aggarwal, and V. Prasad, "Depth first generation of long patterns," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2000, pp. 108–118.

[2]. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1993, pp. 207–216.

[3]. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. 20th Int. Conf. Very Large Databases, 1994, pp. 487–499.

[4]. C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Trans. Knowl. Data Eng., vol. 21, no. 12, pp. 1708– 1721, Dec. 2009.

[5]. R. Bayardo and R. Agrawal, "Mining the most interesting rules," in Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 1999, pp. 145–154.

[6]. F. Bonchi and B. Goethals, "FP-Bonsai: The art of growing and pruning small FP-trees," in Proc. 8th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2004, pp. 155–160.

[7]. F. Bonchi and C. Lucchese, "Extending the state-of-the-art of constraint-based pattern discovery," Data Knowl. Eng., vol. 60, no. 2, pp. 377–399, 2007.

[8]. C. Bucila, J. Gehrke, D. Kifer, and W. M. White, "Dualminer: A dual-pruning algorithm for itemsets with constraints," Data Mining Knowl. Discovery, vol. 7, no. 3, pp. 241–272, 2003.

[9]. C. H. Cai, A. W. C. Fu, C. H. Cheng, and W. W. Kwong, "Mining association rules with weighted items," in Proc. Int. Database Eng. Appl. Symp., 1998, pp. 68–77.

[10]. R. Chan, Q. Yang, and Y. Shen, "Mining high utility itemsets," in Proc. Int. Conf. Data Mining, 2003, pp. 19–26.

[11]. S. Dawar and V. Goyal, "UP-Hist tree: An efficient data structure for mining high utility patterns from transaction databases," in Proc. 19th Int. Database Eng. Appl. Symp., 2015, pp. 56–61.

[12]. T. De Bie, "Maximum entropy models and subjective interestingness: An application to tiles in binary databases," Data Mining Knowl. Discovery, vol. 23, no. 3, pp. 407–446, 2011.

[13]. L. De Raedt, T. Guns, and S. Nijssen, "Constraint programming for itemset mining," in Proc. ACM SIGKDD, 2008, pp. 204–212.
[14]. A. Erwin, R. P. Gopalan, and N. R. Achuthan, "Efficient mining of high utility itemsets from large datasets," in Proc. 12th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2008, pp. 554–561.
[15]. P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility cooccurrence pruning," in Proc. 21st Int. Symp. Found. Intell. Syst., 2014, pp. 83–92.
[16]. L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," ACM Comput. Surveys, vol. 38, no. 3, p. 9, 2006.
[17]. J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2000, pp. 1–12.
[18]. R. J. Hilderman, C. L. Carter, H. J. Hamilton, and N. Cercone, "Mining market basket data using share measures and characterized itemsets," in Proc. PAKDD, 1998, pp. 72–86.
[19]. R. J. Hilderman and H. J. Hamilton, "Measuring the interestingness of discovered knowledge: A principled approach," Intell. Data Anal., vol. 7, no. 4, pp. 347–382, 2003.
[20]. M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen, "A perspective on databases and data mining," in Proc. 1st Int. Conf. Knowl. Discovery Data Mining, 1995, pp. 150–155.
[21]. S. Krishnamoorthy, "Pruning strategies for mining high utility itemsets," Expert Syst. Appl., vol. 42, no. 5, pp. 2371–2381, 2015.
[22]. G.-C. Lan, T.-P. Hong, and V. S. Tseng, "An efficient projectionbased indexing approach for mining high utility itemsets," Knowl. Inf. Syst., vol. 38, no. 1, pp. 85–107, 2014.
[23]. Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," Data Knowl. Eng., vol. 64, no. 1, pp. 198–217, 2008.
[24]. T. Y. Lin, Y. Y. Yao, and E. Louie, "Value added association rules," in Proc. 6th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2002, pp. 328–333.
[25]. J. Liu, Y. Pan, K. Wang, and J. Han, "Mining frequent item sets by opportunistic projection," in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2002, pp. 229–238.