Abstract—Recently, high utility pattern (HUP) mining is one of the most important research issues in data mining due to its ability to consider the nonbinary frequency values of items in transactions and different profit values for every item. On the other hand, incremental and interactive data mining provide the ability to use previous data structures and mining results in order to reduce unnecessary calculations when a database is updated, or when the minimum threshold is changed. In this paper, we propose a tree structure, called IIUT (incremental and interactive utility tree), by considering High, medium and low utility patterns. It uses a pattern growth mining approach to avoid the level-wise candidate set generation-and-test problem, and it can efficiently capture the incremental data without any restructuring operation. Moreover, IIUT has the “build once mine many” property and therefore it is highly suitable for interactive mining. Experimental results show that our tree structure is very efficient and scalable for incremental and interactive high, medium and low utility pattern mining.

Keywords—Data mining, Knowledge discovery, Frequent pattern mining, High, Medium and Low utility pattern mining, Incremental mining, Interactive mining

I. INTRODUCTION

Data mining techniques can efficiently discover hidden knowledge from databases. Frequent pattern mining [1, 3,4,9,12,14] plays an essential role in many data mining tasks such as association rule mining, classification, clustering, time-series mining, graph mining, web mining, and so on. The initial solution of the frequent pattern mining, the Apriori algorithm [3,4] is based on the candidate generation-and-test methodology and requires several database scans. In the first database scan, it finds all the 1-element frequent itemsets and based on that, it generates candidates for 2-element frequent itemsets. In the second database scan, it finds all the 2-element frequent itemsets and based on that, it generates the candidates for 3-element frequent itemsets and so on. This level-wise candidate generation process may create the problems of several database scans and huge candidate pattern generation. Han et al. [14] solved these problems by introducing a prefix tree (FP-tree)-based algorithm without candidate set generation and testing. This algorithm is called the frequent-pattern growth or FP-growth algorithm and needs two database scans.

Although frequent pattern mining plays an important role in data mining applications, it has two limitations. First, it treats all items with the same importance/weight/price and second, in one transaction each item appears in a binary (0/1) form, i.e. either present or absent. In the real world, however, each item in the supermarket has a different importance/price and one customer can buy multiple copies of an item. Therefore, finding only traditional frequent patterns in a database cannot fulfill the requirement of finding the most valuable customers/itemsets that contribute the most to the total profit in a retail business.

The high medium and low utility mining model was defined to solve the above limitations of frequent pattern mining by allowing the user to conveniently measure the importance of an itemset by its utility value. With utility mining, several important decisions in business area like maximizing revenue, minimizing marketing or inventory costs can be made and more important knowledge about itemsets/customers contributing to the majority of the profit can be discovered. In addition to real world retail markets, we can also consider biological gene databases and web click streams, where the importance of each gene or website is different and their occurrences are not limited to a 0/1 value. These techniques can also be applied to many other areas, including stock tickers, network traffic measurements, web-server logs, data feeds from sensor networks.

On the other hand, by using incremental and interactive pattern mining, we can use the previous data structures and mining results and avoid unnecessary calculations when the database is updated or the mining threshold is changed. However, the previous high utility pattern mining techniques are based on a fixed database and so they do not consider that one or more transactions may be deleted, inserted or modified in the database. To understand the necessity of today’s incremental databases, where additions, deletions, and modifications are frequent operations, we can consider a small example in the real world market. Say customer X has bought 3 pens, 4 pencils and 1 eraser and customer Y has bought 1 computer mouse. After some time, customer Z may come to buy 2 loaves of bread and 1 carton of milk, customer X may return 2 pencils, and customer Y may return the mouse. As one can see, in the real world, new
transactions can be added, and old transactions may be frequently modified or deleted. As a result, we must consider additions, deletions and modifications to the transactions in real world datasets.

Moreover, the existing data structures do not have the “build once mine many” property (by building the data structure only once, several mining operations can be done) for interactive mining. As a result, they cannot use their previous data structures and mining results for the new mining threshold. In our real world, however, the users need to repeatedly change the minimum threshold for useful information extraction according to their application requirements. Therefore, the “build once mine many” “build once mine many” property is essentially needed to solve these interactive mining problems.

Motivated by these real world scenarios, we propose a novel tree structure, called IIUT (Incremental and Interactive Utility Tree), for high medium and low utility pattern mining in incremental databases. It exploits a pattern growth mining approach to avoid the level-wise candidate generation-and-test problem of the existing high utility pattern mining algorithms. It needs only two database scans, in contrast to several database scans needed for the existing algorithms. We design the IIUT structure according to the item’s appearance order, and it can efficiently handle the incremental data without any restructuring operations. Moreover, IIUT has the “build once mine many” property and therefore it is very efficient in interactive mining. Extensive experimental results show that our tree approach is very efficient for incremental and interactive high medium and low utility pattern mining, and it outperforms the existing algorithms.

The remainder of this paper is organized as follows. In Section II, we describe the Objective. In Section III, we describe the Problem definition. In Section IV, we describe the IIUT advantages. In Section V, we have IIUT Properties. In Section VI, we have proposed tree structure for high, Medium, and Low utility pattern mining. In Section VII, our experimental results are presented and analyzed. Finally, in Section VIII, conclusions are drawn and in Section IX, Acknowledgment follows.

II. OBJECTIVE

Mining of IIUT and finding high, medium and low utility patterns which can be applied in maximizing the profit and maintaining the good will of the customer in a super market.

<table>
<thead>
<tr>
<th>Item id</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Trans utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>T2</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>T3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>76</td>
</tr>
<tr>
<td>T4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>T5</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>T6</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit ($/per unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
</tr>
</tbody>
</table>

Transcational Database

Utility Table

Figure 1. Example of a transaction database and utility table.

III. PROBLEM DEFINITION

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and $D$ be a transaction database $\{T_1, T_2, \ldots, T_n\}$ where each transaction $T_i \in D$ is a subset of $I$.

**Definition 1:** The local transaction utility value $l(ip, T_q)$, represents the quantity of $ip$ in transaction $T_q$. For example, in Figure 1(a) $l(b, T_2) = 8$.

**Definition 2:** The external utility $p(ip)$ is the unit profit value of item $ip$. For example, in Figure 1(b), $p(b) = 6$.

**Definition 3:** Utility $u(ip, T_q)$, is the quantitative measure of utility for item $ip$ in transaction $T_q$, defined by

$$u(ip, T_q) = l(ip, T_q) \times p(ip) \quad (1)$$

For example, $u(b, T_2) = 8 \times 6 = 48$ in Figure 1.

**Definition 4:** The utility of an itemset $X$ in transaction $T_q$, $u(X, T_q)$ is defined by,

$$u(X, T_q) = \sum_{ip \in X} u(ip, T_q) \quad (2)$$

**Definition 5:** The utility of an itemset $X$ is defined by,

$$u(X) = \sum_{T_q \in D} \sum_{ip \in X} u(ip, T_q) \quad (3)$$

For example, $u(ab) = u(ab, T_2) + u(ab, T_4) + u(ab, T_6) = 56 + 16 + 42 = 114$ in Figure 1.

**Definition 6:** The transaction utility of transaction $T_q$ denoted as $tu(T_q)$ means the total profit of that transaction and it is defined by,

$$tu(T_q) = \sum_{ip \in T_q} u(ip, T_q) \quad (4)$$

For example, $tu(T_1) = u(b, T_1) + u(c, T_1) + u(d, T_1) = 12 + 24 + 16 = 52$ in Figure 1.

**Definition 7:** The minimum utility threshold $\delta$, is given by percentage of the total transaction utility values of the database. In Figure 1, the summation of all the transaction utility values is 323. If $\delta$ is 25% or we can also express it as 0.25, then minimum utility value can be defined as

$$minutil = \delta \times \sum_{T_q \in D} tu(T_q) \quad (5)$$

So, in this example $minutil = 0.25 \times 323 = 80.75$ in Figure 1.

**Definition 8:** An itemset $X$ is a high utility itemset, if $u(X) \geq minutil$. Finding high utility itemsets means find out all the itemsets $X$ having $u(X) \geq minutil$. For $minutil = 80.75$, pattern “ab” is a high utility pattern, as $u(ab) = 114$ (calculated in the example of Definition 5)
The main challenging problem of high utility pattern mining area is itemset utility does not have the anti-monotone property. For example, if\( \text{minutil} = 80.75 \) in Figure 1, then “d” is a low utility item as \( u(D) = 48 \), but “bd” is a high utility itemset as \( u(bd) = 90 \) according to equation (3) and Definition 8. According to Definition 5 and equation (3), the definition of itemset’s utility is based on the actual sum of product calculation similar to our real world business calculations. Maintaining the anti-monotone property is more challenging here compared to the weighted frequent itemset mining problem for two reasons. First, weighted frequent itemset mining considers the 0/1 appearance of an item inside a transaction, and second, the weight of an itemset is calculated using the average of all the items’ weights inside of it. In high utility pattern mining we can maintain the anti-monotone property by transaction weighted utilization.

**Definition 9:** The transaction weighted utilization of an itemset \( X \), denoted by \( twu(X) \), is the sum of the transaction utilities of all the transactions containing \( X \)

\[
    twu(X) = \sum_{T \subseteq \mathcal{D} \mid X \subseteq T} u(T)
\]

For example, \( twu(ac) = tu(T4) + tu(T5) = 37 + 28 = 65 \) in Figure 1. The anti-monotone property can be maintained using transaction weighted utilization. Here, for \( \text{minutil} = 80.75 \) in Figure 1, as \( twu(ac) < \text{minutil} \), any super pattern of “ac” cannot be high twu itemset and obviously can not be high utility itemset.

**Definition 10:** \( X \) is a high transaction weighted utilization itemset if \( twu(X) \geq \text{minutil} \).

In our algorithm, after finding all the high twu pattern maintaining the anti-monotone property using only one database scan, we calculate all the high utility patterns from the high twu patterns by scanning the database a second time and performing the original utility calculations (according to equation 3) for those high twu patterns.

**IV. IIUT ADVANTAGES**

- Maximizes the profit:
- Each item appears in multiple number:
- Difference importance/price/weight to different items

**V. IIUT PROPERTIES**

The following properties are true for IIUT.

**Property 1:** The total count of twu value of any node in IIUT is greater than or equal to the sum of total counts of twu values of its children.

**Property 2:** The ordering of items in IIUT is unaffected by the changes in twu values caused by incremental updating.
After inserting up to T6

Figure 2. Construction process of IIUT

(a) Incremental Database after adding db1 and db2

<table>
<thead>
<tr>
<th>Item id</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Trans Utility($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>T2</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>T3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>T4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>T5</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>T6</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>74</td>
</tr>
<tr>
<td>T7</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>102</td>
</tr>
<tr>
<td>T8</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>T9</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>T10</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

T1 to T6 Original db, T7 to T8 db1, T9 to T10 db2

(b) Updated Database

<table>
<thead>
<tr>
<th>Item id</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Trans Utility($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>T2(new)</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>T3(new)</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>T4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>T6</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>74</td>
</tr>
<tr>
<td>T7</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>102</td>
</tr>
<tr>
<td>T8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>T10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

After update operations

Figure 3. Incremental maintenance of IIUT

Figure 3(a) shows our initial database of Figure 1 is increased by adding two groups of transactions, db1 + and db2 +. Figure 3(b) shows our current database of Figure 3(a) is updated by deleting transactions T5 and T9 and by modifying transaction T2 and T3. Figure 3(c) shows IIUT can be easily incremented to capture db1 + and db2 +. The insertion process is same as transactions are inserted in the construction process.

B. MINING PROCESS

As our IIUT has the important property of FP-tree stated in property 1, pattern growth mining approach can be directly applicable to IIUT by using two values. Consider the last updated database of Figure 3(b) and $ = 0.25$. According to equation 5, $minutil = 106.75$. As in pattern growth, we start from the bottom most item.

Conditional Tree for item “e”
Conditional Tree for item “a”

<table>
<thead>
<tr>
<th>Tid</th>
<th>A</th>
<th>B</th>
<th>D</th>
<th>Trans Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>T4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>T6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>74</td>
</tr>
<tr>
<td>T7</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>102</td>
</tr>
</tbody>
</table>

H-Table

B:257
D:176

Conditional Tree for item “d”

<table>
<thead>
<tr>
<th>Tid</th>
<th>B</th>
<th>D</th>
<th>Trans Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>2</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>T6</td>
<td>5</td>
<td>4</td>
<td>74</td>
</tr>
<tr>
<td>T7</td>
<td>4</td>
<td>5</td>
<td>102</td>
</tr>
</tbody>
</table>

H-Table

B:228

Min util=0.25*427 =106.75

High twu patterns:

From item “e”:
1. \{be:140\}
2. \{de:166\}
3. \{e:204\}

From item “a”:
4. \{ab:257\}
5. \{ad:176\}
6. \{abd:176\}
7. \{a:257\}

From item “d”:
8. \{bd:228\}
9. \{d:308\}
10. \{b:347\}

Result: The High utility patterns,

We got this High utility patterns by considering the 2 non zero values. Consider in conditional tree “e” take items b and e in T3 and T 7 .We have 32 and 43 and those items transcation utility values are 38+102=140. In the simillar way consider item d and e in T7 and T8 53 and 34 and those transcation utility values are 102+64=166 and at last consider only item e values and add them 38+102+64=204.In this way we will get the First database scan of the items.

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit ($) per unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>10</td>
</tr>
</tbody>
</table>

Utility Table

In the second daabase scan take conditional item “e” in T3 and T7 In T3 take value of b that is 3 and in T7 value of b is 4 and add both of them that is 3+4=7 amd now consider the value of e in T3 and in T7 that is 2+3=5.After doing in this way take the profit values of “b and “e” in utility table and...
multiply with that value and check whether it is a Hup,Mup or Lup mining.

From (1) high twu pattern:b:7 ,e:5

- Utility value=7*6+5*10
  =92<106.5,
It is not a high utility pattern.

From (2) high twu pattern:d:8 ,e:7

- Utility value=8*8+7*10
  =134>106.5,
there fore ,{de:134} is a high utility pattern.

Similarly other high utility patterns:

- {a,b:134}
- {a,b,d:146}
- {b:132}
- {b,d:154}
- {d:128}

By using IIUT The high utility patterns, medium utility patterns and low utility patterns are obtained using the utility threshold values.

The High utility patterns (Hup):

If the utility value of the items are greater than the High utility threshold value then that items are known as High utility patterns.

The Medium utility patterns (Mup):

If the utility value of the items are greater than the Medium utility threshold value and lesser than High utility threshold value then that items are known as Medium utility patterns.

The Low utility patterns (Lup):

If the utility value of the items are lesser than the Medium utility threshold value then that items are known as Low utility patterns.

The utility threshold value :

It is given by percentage of the total transaction utility values of the data base.

VII. EXPERIMENTAL RESULTS

To evaluate the performance of our proposed tree structure, we have performed several experiments on the IBM synthetic T10I4D100K dataset, and the real-life mushroom and kosarak datasets from frequent itemset mining dataset repositor (http://fimi.cs.helsinki.fi/data/) and UCI Machine Learning Repository (http://kdd.ics.uci.edu/). These datasets do not provide the profit values or quantity of each item for each transaction. As like the performance evaluation of the previous weight/utility based pattern mining we have generated random numbers for the profit values of each item and quantity of each item in each transaction. Observed from real world databases that most of the items carry low profit, we generate the profit values using a lognormal distribution. Our programs were written in Java and run with the Windows XP operating system on a Pentium dual core 2.13 GHz CPU with 1GB main memory. The below example database shows how we have acheived the high ,medium and low utility pattern mining.

Example(DB):

```
0 2 8 2 0
4 6 0 0 0
0 3 0 0 2
2 2 7 0 0
6 5 0 4 0
4 4 0 5 3
0 0 0 3 4
0 0 0 2 0
```

The high utility threshold value (25% of total utility) : 106.75

The medium utility threshold value (7% of total utility) : 29.89

- The total items are :16
- The total items are :22
- The total items are :15
- The total items are :16
- The total items are :9

- The utility is :32
- The utility is :132
- The utility is :45
The utility patterns are used to apply in Super Market System to maximize the profit by using the high utility patterns, based on the profit values of the products the high utility products will be obtained then this products can be concentrated more marketing strategies are implemented in order to attract the customers. And to maintain the good will of the customer the medium utility products and the low utility products are useful because by using this two products all the items can be concentrated including the products which gives less profit so that all the time the stock will be maintained and customer will get any item which results in good will.

VIII. CONCLUSION

In this Mining Process of IIUT using Pattern Growth, an efficient tree structure called IIUT is obtained that adjusts itself dynamically for incremental and interactive high utility pattern mining without any restructuring operations. IIUT has the “build once, mine many” property and is highly suitable for interactive mining. Moreover, it exploits a pattern growth mining approach to avoid the level-wise candidate generation-and-test problem, and uses a maximum of two database scans. The high utility, medium utility and low utility patterns are obtained using the high and medium utility threshold values. These patterns can be applied in the Super Market System in maximizing the profit and in maintaining the good will of the customer.

IX. ACKNOWLEDGMENT

I express my earnest and deep sense of gratitude to Mrs. D.Sujatha, Head Of the Department, Computer Science & Engineering without Her valuable suggestions and guidance, preparation of this Paper wouldn’t have been possible. It was a great pleasure and honor to work under Her and I hope many more fruitful associations in the years to come. Her kindness and affection will remain with me forever. I also owe the gratitude due towards T. Naga lakshmi Lecturer, Coordinator for me. And I want to thank my guide K.Chandra Shekar and other faculty members of ATRI college, for their noteworthy suggestions and encouragement given to me in completing the paper.. I would also like to thanks the members of PRC for their help, through provoking discussions invigorating suggestions extended to me with immense care, zeal throughout the work.

C.B. Raghavendar born on April 14th 1987. He is pursuing M.Tech ( Software Engineering ) specialization at Aurora’s Technologocal and Research Institute, Hyderabad, India Affiliated to JNTU-H. His expertise includes Data Mining, Computer Networks.

K. Chandra Shekar, born on 9th September 1983. He completed his M.Tech in Engineering from JNTU-H, Hyderabad and has an experience of 8 years. Presently, he is working as Associate Professor in the Department of CSE at Aurora’s Technological Research Institute, Hyderabad, India. His areas of interest are Data Mining, Theory of Computation.