12-2002

VDL: A language for active mining variants of association rules

Kok-Leong ONG

Wee-Keong NG

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Citation

ONG, Kok-Leong; NG, Wee-Keong; and LIM, Ee Peng. VDL: A language for active mining variants of association rules. (2002). *International Workshop on Active Mining AM 2002, in conjunction with the IEEE International Conference on Data Mining ICDM 2002, 9-12 December*. 1-6. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/905

VDL: A Language for Active Mining Variants of Association Rules

Kok-Leong Ong Wee-Keong Ng Ee-Peng Lim
Nanyang Technological University, Nanyang Ave, N4-B3C-14
Singapore 639798, SINGAPORE
ongkl@pmail.ntu.edu.sg

Abstract

The popularity of association rules has resulted in several variations being proposed. In each case, additional attributes in the data are considered so as to produce more informative rules. In the context of active mining, different types of rules may be required over a period of time due to knowledge needs or the availability of new attributes. The present approach is the ad-hoc development of algorithms for each variant of rules. This is time consuming and costly, and is a stumping block to the vision of active mining. We argue that knowledge needs and the changing characteristics of the data requires the ability to re-specify the type of rules to rediscover over time. This paper proposes a novel approach to specify the "how-to" of mining different rule variants without the cost of developing new algorithms. Called the VDL, it is SQL-like and has the expressive power demonstrated by our examples, some of which are classical and others novel. We also give a discussion on the theoretical model underpinning our proposal.

1. Introduction

The discovery of association rules has been a popular domain of study in data mining. We call a rule $X \rightarrow Y$, where X and Y are sets of items, a *base* rule [1]. Over the years, different attributes [2, 5, 8, 9, 11, 16] were considered to create more informative rules which we refer to as *variants*. Usually, a variant of the base rule considers additional attributes to produce more informative rules as illustrated in our examples below.

Example 1 Let $X = \{a, d\}$ and $Y = \{e, y\}$ such that X and Y satisfies the support and confidence requirements, then the base rule would be $X \rightarrow Y$ or $\{a, e\} \rightarrow \{e, y\}$. If we consider the number of times an item occurs in X or Y (i.e., the recurrence), we have a more informative rule that may be represented as $\{2\,a, 1\,d\} \rightarrow \{3\,e, 5\,y\}$. We call this rule [16] a variant of the base rule, and it means that two occurrences of 'a' and an occurrence of 'd' leads to three occurrences of 'e' and five occurrences of 'y'.

Example 2 Assuming the same X and Y in Example 1, another variant of $X \rightarrow Y$ may be obtained if we consider their spatial relationship. One such rule would be $\{\texttt{left}(a, d)\} \rightarrow \{\texttt{top}(e, y)\}$ which means that whenever 'a' appears on the left of 'd', 'e' will appear on top of 'y'. This rule is another variant [8] from the base rule, where each transaction contains the spatial relationship of each item with respect to the others.

In the context of active rule mining, the above examples illustrate an important motivation. Data attributes (e.g., the recurrence value in Example 1 and the spatial relationships in Example 2) are continuously added or remove, and knowledge needs changes over time. While the analyst may find the base rules useful today, the inclusion of a new attribute may motivate the need to rediscover rules with the new attribute in mind. A survey of existing literature shows various attributes been considered, and each presents their ad-hoc solution of an algorithm to address the new attribute's contribution to mining. Clearly, the cost of developing algorithms is both costly and time consuming. This is evident with the availability of *only* base rules in most commercial data mining tools. And this is a stumping block towards the vision of active mining where new variants may be needed over time. To address this problem, the ad-hoc approach taken by the current community must be re-evaluated.

Looking at the database industry, we have taken for granted the availability of SQL to retrieve relevant tuples from the database. Before SQL and the relational model, this has been done in an ad-hoc fashion, where individuals write code to retrieve relevant tuples from proprietary data models. This chaotic situation is similar to the current state of data mining for rules. While data mining languages has been proposed, non to our knowledge propose a declarative approach on the "how-to" of finding different rule variants. Taking the cue from the database industry, our contribution in this paper is to address this issue. The Variant Description Language (VDL) is our effort to declaratively describe different approaches to finding variants of rules using the Apriori [1] as the model to eliminate the need for developing algorithms. This in turn will enable speedy re-specification of rules to mine and hence, achieve active mining of new rules from ever changing data sources.

The remaining sections of the paper is organized as follows. In the next section, we discuss some related work on declarative languages. We then illustrate the expressiveness of our language in Section 3 using examples of which some are classical and others novel. Section 4 then gives a brief description of the theoretical models that the language is built upon. Finally, we summarize our work in Section 5.

2. Related Work

The work in this paper was motivated from Mannila's [10] discussion on a theoretical framework for data mining. He commented on the ad-hoc situation of data mining research and called for a systematic framework to develop KDD applications. A framework for mining rules was discussed but lacked concrete details. Our work continues from where Mannila stopped. Although far from a theoretical foundation, the proposal is a step closer towards a systematic approach of knowledge discovery.

Closest to our proposal is the `MINE RULE` operator proposed by Meo et al [12]. Also SQL-like, the operator enables a uniform and consistent description of the problem of discovering association rules. Using the `MINE RULE` operator, the authors demonstrate how it can be used to describe the different rule mining tasks such as mining base rules with data constraints or rules at multiple concept levels (which is one of the variants known [5]). Of course, the difference lie in the objective of the language. Meo's `MINE RULE` operator is concerned with the uniform description of different but similar tasks of rule mining where each implementation of the algorithm is in place. Although similar in motivation, our proposal describes how different parameters are considered to find different variants of rules. In our case, no algorithms exist, and in place is an algorithmic engine that can, with the VDL, mine future variants.

Around the same time, a more generalized declarative language to describe different data mining tasks was proposed by Han et al [6]. The language is also SQL-like and arises from the **DBMiner** [6] project where different data mining algorithms were implemented. Unlike Meo's `MINE RULE` operator, the Data Mining Query Language, or DMQL in short, presents a consistent and uniform view to describing different data mining models ranging from classification, clustering to association rule mining.

3. VDL By Examples

We begin with the conceptual model needed to describe the task of association rule mining. Let $\mathcal{D}$ be a database of transactions. A transaction $T \in \mathcal{D}$ contains items from the universal of all items $\mathcal{I}$ in $\mathcal{D}$ such that $T \subseteq \mathcal{I}$ and $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ where $i_j, 1 \leq j \leq n$ are known as items. Let $P = \{p_1, p_2, \ldots, p_m\}$ be the universe of attributes associated with each item in T in $\mathcal{D}$. Given an item i_x in a transaction T_y, $T_y.i_x.p_z$ returns the attribute value of p_z of i_x in T_y. Using this object oriented notation, the universe of P is never an empty set and contains at least one attribute called `Label` representing the description of an item. Except for the VDL, the label is always implicitly written in this paper (i.e., an item e means that it has the label 'e').

With the above definitions, we can now discuss the VDL using a practical case to illustrate its relevance to active mining of informative rules. The practical case is the transaction database of a typical supermarket. When a customer buys some products, the whole purchase is mapped to the concept of a transaction and each product is an item. In the beginning, what is stored in each purchase are the unique items bought. Conceptually, the database may looked like the one in Table 1.

TID	Items
1	milk bread butter
2	toothbrush milk bread coke
3	diapers bread coke
4	milk bread beer

Table 1. The transaction database where only unique items of each purchase are stored.

With the available data, the analyst can obtain a set of base rules. This can be done using existing data mining packages. In our proposal, the same set of base rules would be discovered using the VDL and the algorithmic engine. Thus, the VDL would be the SQL of databases, and the algorithmic engine is the implementation of the Apriori model similar to tables as implementation of the relational model. In the Apriori model, the idea of finding an association rule lies in the discovery of frequent itemsets. A frequent itemset can be obtained by (1) generating potential candidate itemsets that are (2) scanned against the database to (3) collect the support count. Since a pass through the database is expensive, itemsets are (4) pruned if we know that it cannot be frequent in the coming pass. Once the support is collected, an itemset is (5) evaluated to determine if it is frequent. If so, it forms the basis of candidate generation in the next round, and is used to obtain rules satisfying the confidence measure.

3.1. Mining Base Rules

Base on the above model, the role of the VDL is to express the five main steps to determine whether a candidate itemset would be frequent, and hence become the basis of confidence evaluation. To extract the frequent itemsets for the base rules, the formulation of the VDL will be as follows:

```
GENERATE BaseRules USING
CANDIDATES FROM AprioriJoin(f, Lp)
PRUNE IF EXISTS C - C.c NOT IN Lp
VOTE IF Subset(C, T)
INCREMENT C.Count BY 1
SELECT IF C.Count >= MinSupp
```

A run of the above VDL produces the set of itemsets that are frequent. These itemsets form the basis of rule generation in which the confidence measure is evaluated. Observe that we have consciously omitted the description of rule generation as the idea is similar and paper space is an issue. However, the reader should be able to extrapolate the methodology to rule generation. The `GENERATE` clause attaches the description “BaseRules” to the set of evaluation criteria after it. The `CANDIDATES` clause defines how candidates should be generated. In this case, we have two predefined concepts namely `f` and `Lp`. The set of frequent itemsets found in the previous pass is represented by `Lp` while `f` refers to the candidate 1-itemsets in the bootstrapping phase and subsequently, are the frequent itemsets of size 1.

The bootstrapping is needed as frequent 1-itemsets are not known initially, and hence `Lp` is empty. The engine first identifies the universe of items $\mathcal{I}$ in $\mathcal{D}$. In the case of Table 1, this is determined to be $\mathcal{I} = \{$milk, bread, butter, toothbrush, coke, diapers, beer$\}$. Since we are in the bootstrapping phase, `f` are the candidate 1-itemsets. This means that all items in $\mathcal{I}$ will be evaluated according to the remaining set of criteria specified in the VDL. At the end of the bootstrapping phase, `Lp` will be properly populated with the frequent 1-itemsets and `f` refers to the frequent 1-itemsets from now on. In Apriori, candidates are scan in a level-wise manner. Hence, once all frequent 1-itemsets are found, the next step is to scan for the frequent 2-itemsets. Notice that the `GENERATE` clause will generate candidate itemsets containing two items each based on the definition of `f` and `Lp` (a formal definition of the `AprioriJoin` is given in Section 4).

To obtain optimum performance, pruning is used to determine if a candidate has the possibility of becoming frequent. If we are sure that the candidate cannot be frequent, then there is no need to waste unnecessary CPU time. This pruning criteria is expressed in the `PRUNE` clause. For finding base rules, the expression means that if any item `c` in the candidate itemset `C` (another predefined concept) fail to exists as a frequent itemset in the last pass, then we know that it fails to satisfy the *a-priori* property and hence cannot be frequent. Thus, the engine proceeds to collect the support count of the current candidate `C` only if the evaluation in the `PRUNE` clause fails.

If the candidate survives the pruning test, it is then checked against each transaction to determine if it supports the itemset. This is represented by the symbol `T` to represent the current transaction under inspection. If the subset test holds, then the `VOTE` clause becomes `true` and the engine determines how it should increment the support count of the current candidate. Notice that the support count of a candidate is also modelled (similar to items) as an attribute named `Count`, and the increment is computed following the expression after the `BY` clause. In this case, the count increments by `1` for each transaction that declares support for the candidate. At the end of the pass through the database, the data mining engine determines if the item is frequent by using the test criteria in the `SELECT` clause. In this case, the VDL states that the candidate should be selected if its support count has surpassed the given minimum support.

Clearly, this approach is preferred over writing code to achieve the same task. The time and resources saved can also be better use by the analyst in other aspects of the KDD process. Of course, the reader may argue that the facilities of finding base rules are available in data mining packages. The sub-sections that follows will better illustrate how the VDL addresses the changing needs of knowledge, and why the VDL approach becomes an effective solution to addressing active mining of relevant and informative rules.

3.2. Mining Weighted Rules

As the holiday season approaches, the marketing manager of the store decides to use data mining to prepare a marketing campaign. When the analyst presents the results, the manager was overwhelmed by the number of rules available. In his opinion (which is also ours), it will take too much time and effort to draft a campaign based on these rules. Within limited time, the manager would like to focus on items of interest such as those under promotion, or items that give a higher profit margin. The smart analyst knows that he is able to accomplish this easily with the VDL.

He first requested the marketing manager to identify the products that he would be interested in gaining insights. He then creates a new attribute call “weight” such that each item has a numerical value to indicate its importance with respect to other items in the database. For example, `milk` may be given a weight of `1.2`, while `toothbrush` has a weight of `0.5`. The two numbers model the manager’s preference in knowing rules containing `milk` over rules containing `toothbrush`. The analyst then formulates a new way to mine rules that considers the weight of each item. This formulation is given as follows.

```
GENERATE WeightedRules USING
CANDIDATES FROM AprioriJoin(f,
    S(k-1) - {S(k-1).s
    WHERE S(k-1).s.Count < S(k-1).s.MSB}
VOTE IF Subset(C, T)
INCREMENT C.Count BY C.Count * SUM({C.c.Weight})
SELECT IF C.Count >= MinSupp
```

In the above, we introduce another variant that ranked the importance of each item relative to the others through the notion of "weights". To our knowledge, this was first explored (expressed above) in [2], and a similar idea based on multiple minimum support was later introduced in [9]. By now, it should be clear that the requirements of the manager can be easily met by writing the VDL. This is where data mining packages fail if support for such consideration is unavailable. From the perspective of active mining, it is thus possible for the analyst to produce the "best fitting" set of rules quickly. This is important as a slow turnaround period may impede the use of the insights in a timely manner.

In this VDL, we introduce two additional concepts `S` and `k`. As mentioned in Section 3.1, frequent candidates in the Apriori model are discovered in a level-wise manner. As such, the concept `k` represents the current size of the candidate itemsets been investigated. As candidates of size `k` are generated, the algorithmic engine places them into the collection `S(k)`. Hence, `S` represents the collection of itemsets of all sizes, and `S(j)`, $\mathtt{j} \geq 1$ refers to the collection of candidates of size `j`.

Interestingly, the notion of weights invalidates the *a-priori* property. To maintain the downward closure property, a measure called *minimum support bound* was introduced. This is a scalar value which we represented as an attribute (i.e., `MSB`) of the itemset. From the view point of the algorithmic engine, it is not concerned with how the `MSB` is computed. In the model, it is simply a value obtained by means of some computable function. This value may be a constant, a value populated via SQL, or complex routines that computes the result. In any case, the engine assumes the availability of this value when the process starts. Compared to the generation of candidates in the base rules, all candidates, except those whose support count is less than its own `MSB` value, are used for candidate generation.

We also introduce the notion of *set* operations in this formulation. When used together with the braces (i.e., {}), the scalar operators such as "`+`" and "`-`" become set operators as demonstrated in the `GENERATE` clause. It is also used to determine the range of variables affected by an operation. For example, the `INCREMENT` clause computes the total weight of each item in the candidate `C`. Without the braces, `C.c.Weight` would simply refer to the weight of one of the item in the candidate at each point of evaluation for the `INCREMENT` clause. Using the brace, the semantic of `SUM` is instilled, and all the weights of the items in `C` are evaluated collectively.

3.3. Mining Recurrent Rules

Suppose the same supermarket upgraded their point-of-sales terminals a year later to include the capability of storing the number of items purchased in each transaction. With this information, the analyst realizes that a more informative version of the base rule can be obtained by considering the quantity of each item as illustrated in Example 1. The inclusion of this new attribute, as a result, motivated the need to reflect a new set of rules that may give the organization better competitive advantage. Intuitively, if the rules are used for target marketing, then we see that the quantity given in a rule will help decide how much of each item should go into a bundle. With the base rules initially, the number of items to include in each bundle is at best a guess from the experience of the marketing manager. To extract rules containing recurrent items, we have the following VDL.

```
GENERATE RecurrentRules USING
CANDIDATES FROM RecurrentJoin(f, Lp)
PRUNE IF EXISTS C - C.c NOT IN Lp
VOTE IF Subset(C, T) AND C.c.Qty <= T.c.Qty
    AND C.c.Label = T.c.Label
INCREMENT C.Count BY
    Min({Floor(T.c.Qty / C.c.Qty)
         WHERE T.c.Label = C.c.Label})
SELECT IF C.Count >= MinSupp
```

Since the quantity of an item is now considered, the way candidates are generated is thus different from that of the candidates generated for the base rules. Using the practical example, we must now consider candidates such as the occurrence of 2 loafs of bread and 3 packets of milk (which the `AprioriJoin` will miss). This consideration is taken into account using `RecurrentJoin` instead. While the pruning condition remains unchanged, the test for transaction support has been modified. The test `C.c.Qty <= T.c.Qty` (of the `VOTE` clause) ensures that the transaction declares support if and only if it has that number of items recorded. As an example, suppose we want to test whether a candidate containing 2 loafs of bread and 2 packets of milk is a frequent itemset in the new database depicted in Table 2. Then transaction T_1 can safely declare support since it has 4 loafs of bread and 6 packets of milk. However, T_2 cannot claim support since it has 3 packets of milk, but only a loaf of bread. In other words, we cannot "create" this candidate from T_2 and thus, T_2 cannot claim support.

To conclude this section, we would like to point the reader to the `BY` clause of the above VDL. Notice that it is not a trivial constant that increments the support count of a candidate by `1`. Instead, a transaction that supports an itemset may have a different support contribution. Continuing from the earlier example, the candidate contains 2 loafs of bread and 2 packets of milk. This means that we can "create" two such candidates from T_1. That is, we can divide the bread into two sets of 2 loafs and for each set, we can

TID	Items
1	milk(6) bread(4) butter(1)
2	toothbrush(1) milk(3) bread(1) coke(6)
3	diapers(1) bread(1) coke(6)
4	milk(6) bread(1) beer(12)

Table 2. The new database where the number in the bracket is the quantity of that item purchased.

assign two packets of milk with two more available. Thus, the support contribution from T_1 would be 2 instead. This is what has been mathematically modelled in the VDL's `BY` clause.

3.4. Mining Recurrent Weighted Items

It should be clear that as new attributes are considered, the real needs of the analyst will evolve (and vice versa). As such, active mining requires the use of the VDL to quickly achieve the updated set of rules not possible with incremental algorithms proposed to date. More importantly, the power of a declarative language goes beyond that of describing individual attributes. In fact, the discussion up to this point has been based on existing rule variants where algorithms are in placed. The VDL is thus a "summary" of the existing algorithm's behavior.

Here, we show that the VDL's expressiveness goes beyond describing existing variants. In fact, the formulation below is a novel representation of combining different attributes to produce a set of informative rules without coding. This particular variant is the result of combining the discussions in the preceding sections. The resultant VDL considers the interest of the manager (i.e., `Weight`), and the quantity (i.e., `Qty`) of each item in the process of discovery. The formulation below is obtained by observing some rules that allows semi-mechanical construction of the VDL involving variants that were individually described. We have elaborated the steps in [15], and we shall leave it as an exercise for the reader to interpret the VDL.

```
GENERATE RecurrentWeightedRules USING
CANDIDATES FROM AprioriJoin(f,
    S(k-1) - {S(k-1).s
    WHERE S(k-1).s.Count < S(k-1).s.MSB}
VOTE IF Subset(C, T) AND C.c.Qty <= T.c.Qty
    AND C.c.Label = T.c.Label
INCREMENT C.Count BY
    Min({Floor(T.c.Qty / C.c.Qty)
         WHERE T.c.Label = C.c.Label}) *
         SUM({C.c.Weight})
SELECT IF C.Count >= MinSupp
```

So far, the individual examples demonstrate how different variants can be described. We then show, with the above formulation, the expressiveness of the language by creating a novel, but informative variant of association rule through a simple composition of their VDL description. And as we unveil each variant's VDL, we also demonstrate how a declarative approach is useful in the context of active mining. Through the discussion of mining weighted rules, we show how we can reflect the knowledge needs of the analyst, and the mining of rules containing recurrent items illustrates how the evolution of the data influence the discovery of new knowledge. Finally, we conclude with a novel example, where attributes are combined to demonstrate how the proposal can scale towards new knowledge needs, beyond what the ad-hoc approach can deliver.

4. Theoretical Model

In this section, we briefly discuss the theoretical models underpinning the design of the VDL. In particular, we explore the data model representing the database, the algorithmic model that describes the mining methodology, and the semantics of the language.

The data model, representing the database, has three tables. In the first, we have a collection of transactions where each has a unique identifier called the `TID` and a set of item descriptions similar to Table 1. In the second table, the unique key is a combination of the `TID` and the item description. Each key, i.e., a $\langle$`TID`, `Label`$\rangle$ pair, returns a tuple containing a set of attribute values P (e.g., `Weight`) as defined in Section 3. Conceptually, the first two table are "read-only" during the execution of the VDL and the third is an auxiliary table where candidates are entered as they are generated. Each unique itemset has an entry (in this table) that contains its set of attribute values (e.g., `Count`, `Qty`, `MSB`). Some of these values are updated when the VDL is executed while others steer the behavior of the data mining engine using the conditions defined in the same VDL.

Based on the above, the VDL formulation determines how the algorithmic engine manipulates the contents of the three tables. As mentioned, this was designed using the Apriori as the basis of our work. The motivation of selecting the Apriori approach comes from our observation that most algorithms proposed for various variants were extension of the Apriori. Hence, it is thus natural and logical to design our proposal from this model. More importantly, we observe that the generality of the Apriori provided room for consideration of new attributes that arises in the future. Such consideration is difficult with other methods.

As a final note, the expressions expressed in the VDL has foundations in discrete mathematics, in particular, first order logic and set theory. Even the functions `AprioriJoin` and `RecurrentJoin` can be described mathematically giving the VDL a strong foundation in its implementation. For example, `AprioriJoin` can be mathematically described as $\mathtt{f} \times_a \mathtt{Lp} = \{\mathtt{f} \cup \mathcal{X} \mid \mathcal{X} \in \mathtt{Lp} \wedge \forall\, x_i, x_j \in \mathcal{X},\ x_i.\mathtt{Label} \leq x_j.\mathtt{Label} \wedge (i < j)\}$, and a VDL expressed using mathematical notations is thus possible as shown below.

```
GENERATE RecurrentRules USING
CANDIDATES FROM $\mathtt{f} \times_a \mathtt{Lp}$
PRUNE IF $\exists c \in C,\ C - \{c\} \notin \mathtt{Lp}$
VOTE IF $C \subseteq T \wedge C.c.\mathtt{Qty} \leq T.c.\mathtt{Qty} \wedge C.c = T.c$
INCREMENT $C.\mathtt{Count}$ BY $\min(\{\lfloor \frac{T.c.\mathtt{Qty}}{C.c_i.\mathtt{Qty}} \rfloor \mid C.c_i = T.c\}_{i=1}^{|C|})$
SELECT IF $C.\mathtt{Count} \geq \mathtt{MinSupp}$
```

5. Summary

In this paper, we proposed the VDL as the mechanism to specify the "how-to" of finding rules in databases. With this approach, we eliminate the need to develop algorithms which are often costly and time consuming to implement. By the time an implementation completes, the value of the knowledge obtain may no longer be relevant or useful. In the competitive economy, knowledge must be constantly updated. Incremental updates of similar knowledge has been relatively well addressed with incremental algorithms [3, 4]. However, the changing needs of relevant knowledge from the same data source cannot be ignored. Data changes as attributes are created or removed, and knowledge needs of an organization also changes with time. In the context of rule mining, this new variants of rules may be needed to reflect these changes. Hence, a method to specify the mining of new rules will be required.

The approach proposed in this paper uses the Apriori algorithm as the conceptual model to finding association rules. The Apriori model is simple to understand and sufficiently expressive for various variants of rules and their compositions. While we have demonstrated a few in this paper, we have actually addressed several others in our technical report [15] which has not been reflected in this paper due to the lack of space. The positive aspects of the Apriori model aside, many readers may be concerned with the performance of the data mining engine. Compared to newer methods such as the `FP-Tree` [7], the Apriori solution appears to be unsuitable. Fortunately, this is not the case.

Favoring the Apriori model, it is easier to write the VDL for expressing several known variants which uses the Apriori algorithm as the basis of extension. At the same time, the model supports a more general approach allowing more complex rules to be specified and new attributes, never considered before, to be included. Our experience to use the `FP-Tree` as the basis of such works proved to be unnecessarily complicated and futile [14]. This is due to the high degree of optimizations made in favor of generality. Fortunately, the model contains room for performance enhancements without loosing the generality to support the VDL. As a matter of fact, we already have an initial implementation of a data structure, called `T-Graph` [13], which builds a "transaction graph" representing the database in a compact manner. By traversing the graph, the VDL needs to scan only a subset of the database and a part of each transaction. Our current implementation shows a performance improvement of nearly an order of magnitude. Effectively, this leverages our proposal to that of modern mechanisms such as the `FP-Tree`, all without giving up the flexibility of a declarative approach based on a simple model described in this paper.

References

[1] R. Agrawal and R. Srikant. Fast Algorithm for Mining Association Rules. In *Proc. of VLDB*, pages 487–499, 1994.

[2] C. H. Cai, A. W. C. Fu, C. H. Cheng, and W. W. Kwong. Mining Association Rules with Weighted Items. In *Proc. of IDEAS Symp.*, 1998.

[3] D. Cheung, S. Lee, and B. Kao. A General Incremental Technique for Updating Discovered Association Rules. In *Proc. of DASFAA*, Australia, 1997.

[4] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In *Proc. of ICDE*, pages 106–114, USA, 1996.

[5] J. Han and Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *Proc. of VLDB*, Zurich, Swizerland, 1995.

[6] J. Han, Y. Fu, W. Wang, K. Koperski, and O. S. Zaine. DMQL: A Data Mining Query Language for Relational Databases. In *SIGMOD DMKD Workshop*, Canada, 1996.

[7] J. Han, J. Pei, and Y. Yin. Mining Frequent Pettern Without Candidate Generation. In *Proc. of SIGMOD*, Dallas, 2000.

[8] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proc. of the Int. Symp. on Large Spatial Databases*, Maine, 1995.

[9] B. Liu, W. Hsu, and Y. Ma. Mining Association Rules with Multiple Minimum Supports. In *Proc. of SIGKDD*, San Diego, 1999.

[10] H. Mannila. Methods and Problems in Data Mining. In *Proc. of ICDT*, Greece, 1997.

[11] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering Frequent Episodes in Sequences. In *Proc. of SIGKDD*, Canada, 1995.

[12] R. Meo, G. Psaila, and S. Ceri. A New SQL-like operator for Mining Association Rules. In *Proc. of VLDB*, India, 1996.

[13] K.-L. Ong, W.-K. Ng, and E.-P. Lim. A Framework for Efficient Scalable Mining of Rule Variants. CAIS Technical Report, Nov. 2001.

[14] K.-L. Ong, W.-K. Ng, and E.-P. Lim. Mining Multi-Level Rules with Recurrent Items Using FP'-Tree. In *Proc. of ICICS*, Singapore, Oct. 2001.

[15] K.-L. Ong, W.-K. Ng, and E.-P. Lim. Mining Variants of Rules Using the CrystalBall Framework. *CAIS Technical Report*, Nov. 2001.

[16] O. R. Zaiane, J. Han, and H. Zhu. Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. In *Proc. of ICDE*, San Diego, 2000.