Itemset Materializing for Fast Mining of Association Rules

Marek Wojciechowski, Maciej Zakrzewicz

Institute of Computing Science
Poznan University of Technology
ul. Piotrowo 3a, 60-965 Poznan, Poland
{marek, mzakrz}@cs.put.poznan.pl

Abstract. Mining association rules is an important data mining problem. Association rules are usually mined repeatedly in different parts of a database. Current algorithms for mining association rules work in two steps. First, the most frequently occurring sets of items are discovered, then the sets are used to generate the association rules. The first step usually requires repeated passes over the analyzed database and determines the overall performance. In this paper, we present a new method that addresses the issue of discovering the most frequently occurring sets of items. Our method consists in materializing precomputed sets of items discovered in logical database partitions. We show that the materialized sets can be repeatedly used to efficiently generate the most frequently occurring sets of items. Using this approach, required association rules can be mined with only one scan of the database. Our experiments show that the proposed method significantly outperforms the well-known algorithms.

1 Introduction

Data mining, also referred to as database mining or knowledge discovery in databases (KDD), is a new research area that aims at discovery of useful information from large datasets. Data mining uses statistical analysis and inference to extract interesting trends and events, create useful reports, support decision making etc. It exploits the massive amounts of data to achieve business, operational or scientific goals. An important goal of current research is to provide methods for on-line analytical mining (OLAM) [6]. On-line analytical mining implies that data mining is performed in a way similar to on-line analytical processing (OLAP), i.e. mining can be performed interactively, for different portions of a database and at different conceptual levels. On-line analytical mining requires a high-performance and rapid-response environment that assists users in data selection, rule generation and rule filtering [5], [8], [11].

One of the most significant data mining problems is mining association rules. Association rules are interesting class of database regularities, introduced by Agrawal, Imielinski, and Swami in [1]. Association rules approaches address a class of problems typified by a market basket analysis. Classic market basket analysis treats the purchase of a number of items (the contents of a shopping basket) as a single transaction. Basket data usually consists of products bought by a customer along with the date of transaction, quantity, price, etc. Such data may be collected, for example, at supermarket checkout counters. The goal is to find trends across large number of purchase transactions that can be used to understand and exploit natural buying patterns, and represent the trends in the form of association rules. Each association rule identifies the set of items that is most often purchased together with another set of items. For example, an association rule may state that "80% of customers who bought items A, B and C also bought D and E". This information may be used for promotional displays design, optimal use of shelf and floor space, effective sales strategies, target marketing, catalogue design etc.

1.1 Association Rules

Let $L=\{l_1, l_2, ..., l_m\}$ be a set of literals, called items. Let a non-empty set of items T be called an *itemset*. Let D be a set of variable length itemsets, where each itemset $T\subseteq L$. We say that an itemset T *supports* an item $x\in L$ if x is in T. We say that an itemset T *supports* an itemset $X\subseteq L$ if T supports every item in the set X.

An *association rule* is an implication of the form $X\rightarrow Y$, where $X\subset L$, $Y\subset L$, $X\cap Y=\varnothing$. Each rule has associated measures of its statistical significance and strength, called *support* and *confidence*. The support of the rule $X\rightarrow Y$ in the set D is:

$$support(X \rightarrow Y, D) = \frac{\left|\{T \in D \,|\, T \text{ supports } X \cup Y\}\right|}{|D|}. \quad \textbf{(1)}$$

In other words, the rule $X\rightarrow Y$ holds in the set D with support s if s% of itemsets in D support $X\cup Y$. The confidence of the rule $X\rightarrow Y$ in the set D is:

$$confidence(X \rightarrow Y, D) = \frac{\left|\{T \in D \,|\, T \text{ supports } X \cup Y\}\right|}{\left|\{T \in D \,|\, T \text{ supports } X\}\right|}. \quad \textbf{(2)}$$

In other words, the rule $X\rightarrow Y$ has confidence c if c% of itemsets in D that support X also support Y.

1.2 Previous Work on Association Rules

The problem of generating association rules was first introduced in [1] and an algorithm called *AIS* was proposed. In [13], an algorithm *SETM* was proposed for mining association rules using relational operators. In [3], Agrawal and Srikant presented two new algorithms, called *Apriori* and *AprioriTid*, that are fundamentally different from the previous ones. The algorithms achieved significant improvements over *SETM* and *AIS* and became the core of many new algorithms for mining association rules

In the existing approaches [3], [6], [7], [9], [12], [14], [15] the problem of mining association rules is decomposed into the following two steps:

1. Discover the large itemsets, i.e. the sets of itemsets that have support above a predetermined minimum support σ.
2. Use the large itemsets to generate the association rules for the database.

It is noted that the overall performance of mining association rules is determined by the first step. After the large itemsets are identified, the corresponding association rules can be derived in a straightforward manner.

Much research has focused on deriving efficient algorithms for discovering large itemsets. Generally, to show that an itemset is large we can count its occurrences in the database D. If the count is greater than σ |D|, then the itemset is large. The problem is that the number of all possible itemsets is huge and it is infeasible to count them all (e.g. for 1000 different items there are: c.a. 500 000 of possible 2-itemsets, c.a. 160 000 000 of possible 3-itemsets, etc.). If we knew, say, a few thousands of itemsets which are *potentially* large we could count them in only one scan of the database. All well-known algorithms rely on the property that an itemset can only be large if all of its subsets are large. It leads to a level-wise procedure. First, all possible 1-itemsets (itemsets containing 1 item) are counted in the database to determine *large 1-itemsets*. Then, large 1-itemsets are combined to form potentially large 2-itemsets, called *candidate 2-itemsets*. Candidate 2-itemsets are counted in the database to determine *large 2-itemsets*. The procedure is continued by combining the large 2-itemsets to form *candidate 3-itemsets* and so forth. A disadvantage of the algorithm is that it requires *K* or *K+1* passes over the database to discover all large itemsets, where *K* is the size of the greatest large itemset found.

Since it is costly to discover association rules in large databases, there is often a need for techniques that incrementally update the discovered association rules every time the database changes. In general, database updates may not only invalidate some existing strong association rules but also turn some weak rules into strong ones. Thus it is nontrivial to maintain such discovered association rules in large databases. In [4], Cheung, Han, Ng and Wong presented an algorithm called *FUP* (Fast Update Algorithm) for computing the large itemsets in the expanded database from the old large itemsets. The major idea of *FUP* algorithm is to reuse the information of the old large itemsets and to integrate the support information of the new large itemsets in order to reduce the pool of candidate itemsets to be re-examined. Unfortunately, the method cannot be used to mine association rules in a part of a database because the large itemsets that hold in the entire database may not hold in a part of it.

Another way of reducing the number of database passes was proposed by Savasere, Omiecinski and Navathe in the algorithm called *Partition* [14]. *Partition* algorithm reads the database in portions into main memory and discovers large itemsets inside each portion. Then, by scanning the whole database, the actual support values for these itemsets are computed.

1.3 Problem Description

Given a database of sets of items, the problem of mining association rules is to discover all rules that have support and confidence above the user-defined minimum values. In practice, association rules can be mined repeatedly in different parts of the database. A straightforward, though ineffective way to solve this problem is to run (each time) a well-known algorithm for mining association rules on the part of the database. Note that if the large itemsets could be precomputed and stored in a database, the algorithm for mining association rules would be simpler and more efficient. This is known as the *itemset materializing*.

In this paper, we propose a new method that addresses the issue of discovering the most frequently occurring sets of items. Our method consists in materializing precomputed sets of items discovered in logical partitions of a large database. We show that the materialized sets of items can be repeatedly used to efficiently generate the sets of items that most frequently occur in the whole database or only in a part of it. Using this approach, the required association rules can be interactively mined with only one scan of the database.

1.4 Outline

The structure of the paper is the following. In Section 2, the method of itemset materializing is described and the algorithm for mining association rules is given. In Section 3 we give our experimental results showing the performance of the new method. Section 4 contains final conclusions.

2 Itemset Materializing Method

The key idea behind itemset materializing is the following. Recall that the reason that limits the well-known algorithms is that if itemset counting should be done in a single scan of a database, the number of itemsets to count would be exponentially large. However, if we could easily select a small set of *potentially* large itemsets, say a few thousand itemsets, then they could be counted in only one scan of a database. We present the method that uses materialized itemsets to select potentially large itemsets that can be verified in a single database scan.

Itemset materializing method divides the database into user-defined, non-overlapping partitions and discovers all large itemsets inside each partition. The positive borders of the large itemsets are computed and stored along with the partitions in the database. We use the positive borders as a condensed representation of the itemsets. Later on, when association rules are to be discovered in a set of database partitions, the positive borders for those partitions are merged to generate the *global positive border*. Then all the itemsets described by the global positive border (potentially large itemsets) are counted in the database partitions to find their supports. Thus, the itemsets materialized only once can be used repeatedly to efficiently select potentially large itemsets.

2.1 Basic Definitions

Positive border
The concept of the positive border was introduced by Manilla and Toivonen in [10]. Given a set S of itemsets, the *positive border* $BD^+(S)$ consists of those itemsets from S which are not contained by any other itemsets from S:

$$Bd^+(S) = \{\, X \in S \mid \text{for all } Y \in S, \text{ we have } X \not\subset Y \,\} . \tag{3}$$

The positive border can play a role of the condensed representation of the itemsets.

Combination of positive borders
Combination $\theta\,(S_1, S_2)$ of the two positive borders S_1 and S_2 is the positive border of $S_1 \cup S_2$:

$$\theta\,(S_1, S_2) = Bd^+(S_1 \cup S_2) . \tag{4}$$

We will use the combination of the positive borders to generate the global positive border for the partitions.

2.2 Generation of Materialized Positive Borders

Given is a database D and a minimum support value σ. The minimum support value σ should be less or equal to a predicted minimum value that users can set on support of their mined association rules.

We divide the database D into a set of non-overlapping partitions: $d_1, d_2, ..., d_n$. The database can be divided according to e.g. dates, locations, customer types. For each partition d_i all large itemsets $L_1^i, L_2^i, ..., L_k^i$ are discovered be means of a well-known algorithm (L_1^i refers to large 1-itemsets in the partition d_i, L_2^i refers to large 2-itemsets in the partition d_i, etc.). Then, for each partition d_i a positive border Bd^+_i of $L_1^i \cup L_2^i \cup ... \cup L_k^i$ is computed and stored in the database together with the partition. The algorithm is shown in Figure 1.

```
for each partition $d_i$ of $D$ do
 begin
  discover all itemsets whose support > $\sigma$
  compute the positive border $Bd^+_i$ for the discovered itemsets
  store $Bd^+_i$ together with $d_i$
 end
```

Fig. 1. Generation of materialized positive borders

Example 1
To illustrate the generation of materialized positive borders, consider the database *D* in Figure 2. Assume that σ is 0.2. Let us first divide the database *D* into three partitions: d_1, d_2, d_3. For each partition all itemsets with support $\geq$ 0.2 are discovered (not depicted here). Then the positive borders of the itemsets are computed. For the partition d_1 we have the positive border $Bd^+_1 = \{\{4,5,6\}, \{1,3,5,7\}, \{2,3,4,7\}\}$, for the partition d_2 we have $Bd^+_2 = \{\{1,3,7\}, \{2,3,5\}, \{1,2,5,7\}\}$ and for the partition d_3 we have $Bd^+_3 = \{\{1,4,6\}, \{3,5,7\}, \{2,4,6,7\}\}$. Then, the positive borders are stored together with the partitions in the database.

D

TID	*Item*
100	1 3 5 7
101	2 3 4 7
102	4 5 6
103	1 2 5 7
104	2 3 5
105	1 3 7
106	3 5 7
107	1 4 6
108	2 4 6 7

d_1

TID	*Item*
100	1 3 5 7
101	2 3 4 7
102	4 5 6

d_2

TID	*Item*
103	1 2 5 7
104	2 3 5
105	1 3 7

d_3

TID	*Item*
106	3 5 7
107	1 4 6
108	2 4 6 7

Fig. 2. Example database and its partitions

2.3 Generation of Large Itemsets from Materialized Itemsets

Given is the database *D*, divided into the set of *n* partitions d_1, d_2, .., d_n, and the positive borders for the partitions Bd^+_1, Bd^+_2, ..., Bd^+_n, derived for the minimum support value σ. Let $I = \{i_1, i_2, ..., i_l\}$ denote a set of partition identifiers. Below we present the algorithm that discovers all large itemsets that hold in $d_{i1} \cup d_{i2} \cup ... \cup d_{il}$ with support above σ', σ' ≥ σ. Once the large itemsets and their supports are determined, the rules can be discovered in a straightforward manner [1].

First, the positive borders of all the partitions described in *I* are *combined* to form the *global positive border S*. Then, for each itemset in *S* all unique subsets are generated to form the set of *potentially large itemsets C*. In the next step the partitions $d_{i1}, d_{i2}, ..., d_{il}$ are scanned to count the occurrences of all the itemsets in *C*. The result of the algorithm consists of the itemsets in *C* with support greater or equal to σ', i.e. large itemsets. The algorithm is shown in Figure 3.

```
$S = \emptyset$
for each partition identifier $i_k$ in $I$ do
  $S = \theta(S, Bd^{+}_{ik})$
$C$ = all unique subsets of the itemsets in $S$
for each transaction $t \in d_{i1} \cup d_{i2} \cup ... \cup d_{il}$ do
  increment the count of all itemsets in $C$ that are contained in $t$
Answer = all itemsets in $C$ with support $\geq \sigma'$
```

Fig. 3. Mining association rules

Example 2

To illustrate the large itemset generation algorithm, consider the database partitions and the positive borders from Example 1. We are looking for all association rules that hold in the database partitions d_1 and d_3 with support ≥ 0.4. First, the positive borders $Bd^{+}_{1} = \{\{4,5,6\}, \{1,3,5,7\}, \{2,3,4,7\}\}$ and $Bd^{+}_{3} = \{\{1,4,6\}, \{3,5,7\}, \{2,4,6,7\}\}$ are combined to form the global positive border $S = \{\{1,4,6\}, \{4,5,6\}, \{1,3,5,7\}, \{2,3,4,7\}, \{2,4,6,7\}\}$. Then, for each itemset in *S* all unique subsets are generated to form the set of potentially large itemsets *C*. Thus we get $C = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{1,3\}, \{1,4\}, \{1,5\}, \{1,6\}, \{1,7\}, \{2,3\}, \{2,4\}, \{2,7\}, \{2,6\}, \{3,4\}, \{3,5\}, \{3,7\}, \{4,5\}, \{4,6\}, \{4,7\}, \{5,6\}, \{5,7\}, \{6,7\}, \{4,5,6\}, \{1,4,6\}, \{1,3,5\}, \{1,3,7\}, \{1,5,7\}, \{3,5,7\}, \{2,3,4\}, \{2,3,7\}, \{2,4,7\}, \{3,4,7\}, \{2,4,6\}, \{2,6,7\}, \{4,6,7\}, \{2,3,4,7\}, \{2,4,6,7\}\}$. Now, the partitions d_1 and d_3 are scanned and all the itemsets in *C* are counted. The itemsets in *C* with support greater or equal to 0.4 are returned as the result. In this example the resulting large itemsets are: {3}, {4}, {5}, {6}, {7}, {3,7}, {4,6}. The association rules that can be derived from those large itemsets are the following:

$3 \rightarrow 7$, support = 0.50, confidence = 1.00
$7 \rightarrow 3$, support = 0.50, confidence = 0.75
$4 \rightarrow 6$, support = 0.50, confidence = 0.75
$6 \rightarrow 4$, support = 0.50, confidence = 1.00

Note that the database has been scanned only once to get this result.

Correctness

Our algorithm relies on the property that an itemset can only be large if it is large in at least one partition. To prove this property formally, we show that if an itemset is not large in any of the partitions, then it is not large in the whole database.

Let s_i denote the count of the itemset in the partition d_i. If the itemset is not large in any of the database partitions, then:

$$s_1/|d_1| < \sigma \wedge s_2/|d_2| < \sigma \wedge \ldots \wedge s_n/|d_n| < \sigma \,, \qquad \textbf{(5)}$$

therefore:

$$s_1 < \sigma\,|d_1| \wedge s_2 < \sigma\,|d_2| \wedge \ldots \wedge s_n < \sigma\,|d_n| \,. \qquad \textbf{(6)}$$

When we add those inequalities together we get the following:

$$s_1 + s_2 + \ldots + s_n < \sigma\,|d_1| + \sigma\,|d_2| + \ldots + \sigma\,|d_n| \,, \qquad \textbf{(7)}$$

therefore:

$$(s_1 + s_2 + \ldots + s_n)/(\,|d_1| + |d_2| + \ldots + |d_n|) < \sigma \,. \qquad \textbf{(8)}$$

The last inequality says that the itemset is not large. Thus, we have shown that an itemset that is not large in any of the partitions can not be large in the whole database.

2.4 Database Partitioning

Let us now consider how the database can be divided into partitions to efficiently use the presented method. The limitation of the method is that association rules can be mined in union of selected database partitions only, not in whichever part of the database. However, it is obvious that in real systems users usually mine association rules in semantic ranges of a database - i.e. in data from selected weeks, months, supermarkets. Then, each partition can refer to a week or to a supermarket and flexible queries over the partitions can be formulated. The similar problem is often discussed in OLAP communities.

Flexibility of the presented method is greater when the number of the database partitions is large. On the other hand, large number of partitions results in storage overhead. Thus, users can evaluate the number and size of the partitions individually, depending on the storage cost and flexibility requirements. We evaluated that a real-life number of database partitions in which users mine association rules is 20-50.

Another important feature of the itemset materializing method is that it is easy to maintain the materialized positive borders for the database partitions. When a database or data warehouse is *updated*, the materialized positive borders for the updated database partitions should be updated too. When new partitions are *appended* to a database or data warehouse, then the positive borders for the new partitions must be computed, however, none of the previously materialized positive borders need to be updated. Besides, computing the positive borders for a partition can be done fast, because the whole partition is likely to fit in main memory.

3 Experimental Results

To assess the performance of the proposed method, we conducted several experiments on large itemset discovering by using a 2-processor Sun SPARCserver 630MP with 128 MB of main memory. The database was implemented in Oracle 7.3.1 DBMS. Experimental data sets were generated by the synthetic data generator *GEN* from Quest project [2]. *GEN* generates textual data files that contain sets of numerical items. Several parameters affect the distribution of the synthetic data. These parameters are shown in Table 1. To load the contents of the data files into the database, *Oracle SQL*Loader* program was used.

Table 1. Synthetic data parameters

parameter	value	parameter	value
n_{trans}	number of item sets, 1,000 and 10,000	n_{pats}	number of patterns, 50 and 500
n_{items}	number of different items, 100	patlen	average length of maximal pattern, 6
t_{len}	average items per set, 10 and 15	corr	correlation between patterns, 0.25

Figures 4 and 5 show the execution time of discovering the large itemsets in the synthetic database for different minimum support values. The positive borders were materialized before the experiment with the traditional *Apriori* algorithm. We have compared the performance of our method (for different numbers of partitions) with the performance of the algorithm *Apriori*. Our method, for 5-20 partitions and minimum support > 0.05, beat out *Apriori*, running 2-3 times faster. As we expected, its performance decreases for decreasing the minimum support value and for increasing the number of partitions. This behavior can be explained by larger number of itemsets that are described by the global positive border.

Figure 6 shows the storage overhead for the materialized positive borders for different numbers of partitions. The space needed to store the positive borders is linearly proportional to the number of partitions. In our experiments, the storage overhead for 30 partitions and minimum support = 0.1 was c.a. 50% of the size of the database. This is the cost of faster mining association rules.

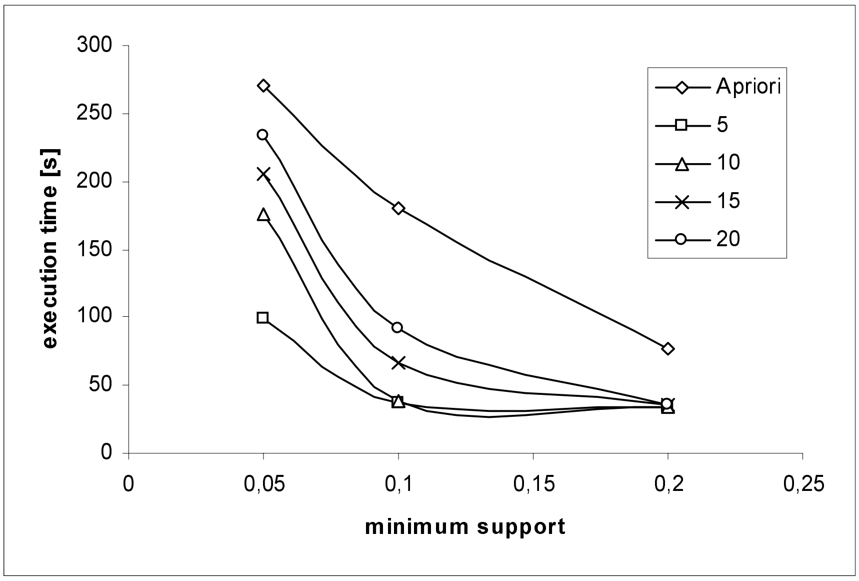

Fig. 4. Execution time for different minimum support values

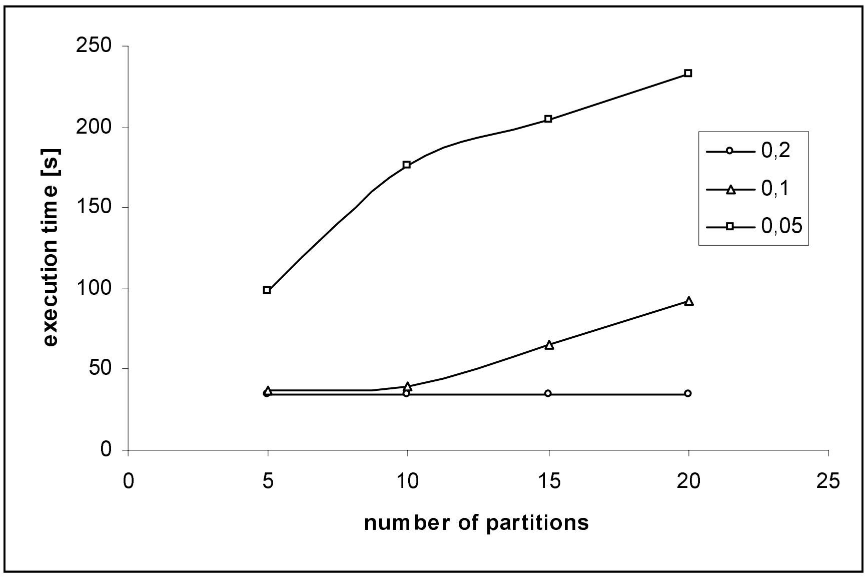

Fig. 5. Execution time for different numbers of partitions and different minimum supports

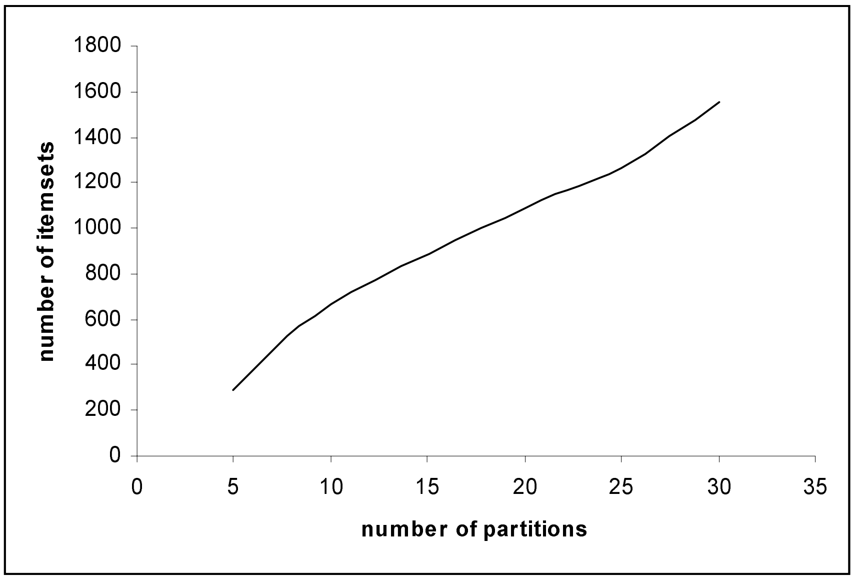

Fig. 6. Size of positive borders for different numbers of partition

4 Conclusions and Future Work

We have introduced and discussed the use of materialized itemsets in the task of discovering association rules in large databases. Our algorithm uses the materialized itemsets to efficiently generate the large itemsets that occur in the whole database or only in a part of it. Thus, the required association rules can be interactively mined with only one scan of the database.

We have presented our experimental results for synthetic databases to show that itemset materializing can be an effective tool for data mining. The itemset materializing method was compared with *Apriori* algorithm and significantly outperformed it. We note that there is a trade-off between the flexibility of mining association rules (in different parts of a database) and the storage overhead.

For the future work, we plan to extend the presented method along the following dimensions:

- develop other data structures that can be used for fast sequential patterns discovery and classification,
- study the buffer management techniques for data mining to efficiently process sequences of association rules mining requests.

References

1. Agrawal R., Imielinski T., Swami A., “Mining Association Rules Between Sets of Items in Large Databases”, Proc. ACM SIGMOD, pp. 207-216, Washington DC, USA, May 1993
2. Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T., "The Quest Data Mining System", Proc. of the 2nd Int’l Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon, August 1996
3. Agrawal R., Srikant R., “Fast Algorithms for Mining Association Rules”, Proc. 20th Int’l Conf. Very Large Data Bases, pp. 478-499, Santiago, Chile, 1994
4. Cheung D.W., Han J., Ng V., Wong C.Y., “Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique”, Proc. Int’l Conf. Dana Eng., New Orleans, USA, February 1996
5. Fayyad U., Piatetsky-Shapiro G., Smyth P., "The KDD Process for Extracting Useful Knowledge from Volumes of Data", Communications of the ACM, Vol. 39, No. 11, Nov. 1996
6. Han J., "Towards On-Line Analytical Mining in Large Databases, SIGMOD Record", Vol. 27, No. 1, March 1998
7. Houtsma M., Swami A., “Set-Oriented Mining of Association Rules”, Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, USA, October 1993
8. Imielinski T., Manilla H., "A Database Perspective on Knowledge Discovery", Communications of the ACM, Vol. 39, No. 11, Nov. 1996
9. Manilla H., Toivonen H., Inkeri Verkamo A., “Efficient Algorithms for Discovering Association Rules”, Proc. AAAI Workshop Knowledge Discovery in Databases, pp. 181-192, July 1994
10. Manilla H., Toivonnen H., "Levelwise Search and Borders of Theories in Knowledge Discovery", Report C-1997-8, University of Helsinki, Finland
11. Morzy T., Zakrzewicz M., " SQL-Like Language For Database Mining", ADBIS’97 Symposium, St. Petersburg, September 1997
12. Park J.S., Chen M.-S., Yu P. S., "An Effective Hash-Based Algorithm for Mining Association Rules", SIGMOD'95, San Jose, CA, USA, 1995
13. Piatetsky-Shapiro G., Frawley W.J., editors, Knowledge Discovery in Databases, MIT Press, 1991
14. Savasere, E. Omiecinski, S. Navathe, “An Efficient Algorithm for Mining Association Rules in Large Databases”, Proc. 21th Int’l Conf. Very Large Data Bases, pp. 432-444, Zurich, Switzerland, September 1995
15. Toivonen H., “Sampling Large Databases for Association Rules”, Proc. 22nd Int’l Conf. Very Large Data Bases, Bombay, India, 1996