Mining Multi-dimensional Frequent Patterns Without Data Cube Construction*

Chuan Li[1], Changjie Tang[1], Zhonghua Yu[1], Yintian Liu[1], Tianqing Zhang[1], Qihong Liu[1], Mingfang Zhu[1], and Yongguang Jiang[2]

[1] The Data Base and Knowledge Engineering Lab (DBKE)
Computer School of Sichuan University
{lichuan, tangchangjie}@cs.scu.edu.cn
[2] Chengdu University of Traditional Chinese Medicine
cdtcm@163.com

Abstract. Existing approaches for multi-dimensional frequent patterns mining rely on the construction of data cube. Since the space of a data cube grows explosively as dimensionality or cardinality grows, it is too costly to materialize a full data cube, esp. when dimensionality or cardinality is large. In this paper, an efficient method is proposed to mine multi-dimensional frequent patterns without data cube construction. The main contributions include: (1) formally proposing the concept of multi-dimensional frequent pattern and its pruning strategy based on Extended Apriori Property, (2) proposing a novel structure called Multi-dimensional Index Tree (MDIT) and a MDIT-based multi-dimensional frequent patterns mining method (MDIT-Mining), and (3) conducting extensive experiments which show that the space consuming of MDIT is more than *4* orders of multitudes smaller than that of data cube along with the increasing of dimensionality or cardinality at most cases.

1 Introduction

Multi-dimensional frequent patterns mining is the essential step in multi-dimensional association rules mining and has wide applications [1, 2]. Let's see a motivating example. A store manager may ask a question like "what groups of customers would like to buy what groups of items ?" What he wants is the patterns like the following:

{*Pricai Proceedings, DBMiner*}∧{*Occupation*('*Student*'), *Major*('*Computer*')}: *2%*

{*Hamburg, Egg, Edible Oil*}∧{*Occupation*('*House Wife*'), *Income*('*Middle*')}: *4%*

* This work was supported by National Science Foundation of China (60473071), Specialized Research Fund for Doctoral Program by the Ministry of Education (SRFDP 20020610007), the grant from the State Administration of Traditional Chinese Medicine (SATCM 2003JP40) and National Science Foundation of China (90409007). Chuan Li, Tianqing Zhang, Yintian Liu, Qihong Liu, and Mingfang Zhu are Ph. D Candidates at DB&KE Lab, Sichuan University. YU Zhonghua is a professor at Sichuan University. JIANG Yongguang is a professor at Chengdu University of TCM. And TANG Changjie is the associate author.

Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 251 – 260, 2006.
© Springer-Verlag Berlin Heidelberg 2006

These patterns are useful as they are helpful to analyze the structure of the purchasing power and the major interests of different customer groups [1]. Such patterns are called multi-dimensional frequent patterns as they contain not only items but also the multi-dimensional values combination and they are frequent. Regretfully, this concept has never been formally proposed. And it is only deemed as an intermediate step in multi-dimensional association rules mining [2]. Almost all existing approaches for multi-dimensional frequent patterns mining rely on the construction of data cube [2, 3, 7, 8, 9].

Unfortunately, the storage space of data cube grows explosively as dimensionality or cardinality (i.e. the number of distinct values in a dimension) grows. For example, a data cube of *6* dimensions, cardinality of *1000* (i.e. each dimension having *1000* distinct values), may contain as many as $(1001)^6$ cube cells. Although adoption of partial materialization (such as iceberg cube, condensed cube, etc.) can delay the growth, it does not solve the fundamental problem [7, 8, 9, 10]. Some may suggest the introduction of concept hierarchies and concept generalization techniques such as AOI [11, 12], attribute reduction [12], etc. to simplify the relation and hence alleviate the disaster caused by large cardinality. However, there are applications with large cardinality dimensions, which have no hierarchies and are too important to be deleted, such as the analysis of Traditional Chinese Medicine Prescriptions [13].

In this paper, an efficient method is proposed to mine multi-dimensional frequent patterns without data cube construction. Our main contributions include: (1) formally proposing the concept of multi-dimensional frequent pattern and its pruning strategy based on Extended Apriori Property, (2) proposing a novel structure called Multi-dimensional Index Tree (MDIT) and a MDIT-based multi-dimensional frequent patterns mining method (MDIT-Mining), and (3) conducting extensive experiments, which show the space consuming of MDIT is more than *4* orders of multitudes smaller than that of data cube along with the increasing of dimensionality or cardinality at most cases.

2 Formal Definition of Multi-dimensional Frequent Pattern

Suppose there are D dimensions, $\{P_1, P_2, \ldots P_D\}$. Let I be a set of items, and p_i be a possible value of the i_{th} dimension. We have the following formal definitions.

Definition 2.1 (Multi-dimensional Pattern). Let $i, i_1, i_2, \ldots i_k$ be items of I, $l_1, l_2, \ldots l_j$ be distinct integers between *1* and D. $i \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$ is called **a Multi-dimensional Item** and $\{i_1, i_2 \ldots i_k\} \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$ is called **a Multi-dimensional Pattern,** where $\{p_{l1}, p_{l2}, \ldots p_{lj}\}$ is called **a Multi-dimensional Values Combination**.

Definition 2.2 (The Universal Set). Let $l_1, l_2, \ldots l_j$,be distinct integers between *1* and D. The union of all distinct multi-dimensional patterns of the form $I \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$ is called **the Universal Set** of the multi-dimensional space, denoted as ***MI***, $MI = \bigcup_{possible\ l1,l2,\ldots lj} \left(I \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}\right)$.

Definition 2.3 (Sub-pattern Relationship). Let $MA=\{i_1, i_2 \ldots i_j\} \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$, $MB=\{i_1', i_2' \ldots i_k'\} \wedge \{p_{l1'}, p_{l2'}, \ldots p_{lj'}\}$ be two multi-dimensional patterns. *MA* is called a **Sub-pattern of** *MB*, denoted as $\boldsymbol{MA \subseteq MB}$, if the following two criteria hold simultaneously:

(1) $\{i_1, i_2 \ldots i_j\} \subseteq \{i_1', i_2' \ldots i_k'\}$, and
(2) For any element $p_l \in \{p_{l1}, p_{l2}, \ldots p_{lj}\}$, there exists an element $p_{l'} \in \{p_{l1'}, p_{l2'}, \ldots p_{lj'}\}$[1] such that (a) p_l and $p_{l'}$ are values of the same dimension, and (b) $p_l = p_{l'}$.

A pattern $MA=\{i_1, i_2 \ldots i_k\} \wedge \{p_{l1'}, p_{l2'}, \ldots p_{lj'}\}$ is called a **Sub-pattern of** *MI*, denoted as $\boldsymbol{MA \subseteq MI}$, if one of the following two criteria holds:

(1) *MA* is of the form $I \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$ or
(2) There exists an multi-dimensional pattern, $MB= I \wedge \{p_{l1}, p_{l2}, \ldots p_{lj}\}$ satisfying the following two criteria simultaneously:
   (a) $MB \subseteq MI$, and
   (b) $MA \subseteq MB$

Based on Definition 2.2 and 2.3 it's clear that for any multi-dimensional pattern *MA*, $MA \subseteq MI$.

Definition 2.4 (Multi-dimensional Transaction Database). Given the universal set *MI*, a set of multi-dimensional patterns $\{mt_1, mt_2, \ldots mt_n\}$ is called a **Multi-dimensional Transaction Database**, denoted as ***MTDB***, if for any $i \in [1..n]$, $mt_i \subseteq MI$. The support η (or occurrence frequency) of a multi-dimensional values combination $\{p_{l1}, p_{l2}, \ldots p_{lj}\}$ is the number of transactions containing $\{p_{l1}, p_{l2}, \ldots p_{lj}\}$ in *MTDB*. The support η (or occurrence frequency) of a multi-dimensional pattern *MA* is the number of transactions containing *MA* in *MTDB*.

Definition 2.5 (Multi-dimensional Frequent Pattern). Let η be the predefined minimum support threshold for the given *MTDB*. A multi-dimensional pattern is **a Multi-dimensional Frequent Pattern** if its support is no less than η. The problem of multi-dimensional frequent patterns mining is to generate the complete set of multi-dimensional frequent patterns from the multi-dimensional database *MTDB*.

3 Pruning Strategy

Based on Definition 2.4 and 2.5, each multi-dimensional pattern should be investigated as to judge whether it is frequent. However, for any given multi-dimensional pattern with *J* dimensions and *K* items, the possible non-empty sub-patterns are as many as $(2^J-1)*(2^K-1)$. The search space is too huge to be handled. Thus, optimization is in great need. Inspired by the single dimensional case, we consider extending the *Apriori* property [5] to multi-dimensional landscape and using it to reduce the search space. Then we have the following lemmas.

[1] Just having $\{p_{l1}, p_{l2}, \ldots p_{lj}\} \subseteq \{p_{l1'}, p_{l2'}, \ldots p_{lj'}\}$ is incorrect because it may not satisfy (a).

Lemma 3.1 (Extended Apriori Property). Let *MA*, *MB* be two multi-dimensional patterns, $MA \subseteq MB$. If *MA* is not a multi-dimensional frequent pattern, *MB* is not a multi-dimensional frequent pattern, either.

Lemma 3.2 (Pruning Strategy). Multi-dimensional patterns containing infrequent multi-dimensional values combination cannot be frequent.

Observation 3.1. The complete set of multi-dimensional frequent patterns could be partitioned into a series of non-overlapped sets such that patterns in each set share the same frequent multi-dimensional values combination.

Based on Lemma 3.2 and Observation 3.1, the multi-dimensional frequent patterns mining process can be fulfilled in the following summarized steps:

(1) Discover all the multi-dimensional values combinations, $\{p_{l1}, p_{l2}, \ldots p_{lj}\}$, whose support is above the minimum support threshold, η.
(2) For each multi-dimensional values combination, $\{p_{l1}, p_{l2}, \ldots p_{lj}\}$, track down all the multi-dimensional transactions containing it, which forms a projected database of *MTDB*.
(3) Mine single-dimensional frequent patterns from the projected databases and output the patterns in conjunction with their corresponding multi-dimensional values combinations.

4 The Introduction of MDIT: Design, Construction and Analysis

Stemming from the above discussion, we propose a new approach, called MDIT, and two new algorithms: one for computing MDIT, and one for multi-dimensional frequent patterns mining based on MDIT. This new approach will be able to handle databases of reasonable dimensionality and extremely high cardinality. The general idea is to partition the base dataset into a series of projected databases according to different multi-dimensional values combinations. With MDIT, one can search the corresponding sets of transaction IDs (*TIDs*) according to frequent multi-dimension values combinations. After that, one can mine frequent patterns from each projected database (represented as a *TID list*). Finally, after assembled with multi-dimensional values combinations, one can get the complete set of multi-dimensional frequent patterns. The whole process is accomplished highly efficient with MDIT.

4.1 Multi-dimensional Index Tree (MDIT)

Definition 4.1. A Inverted Index Item is a *2*-tuple $R=\{I, P\}$, where *I* denotes a dimension value and *P* denotes a pointer leading to a child node.

Definition 4.2. Multi-dimensional Index Tree (MDIT) is a tree-like structure defined as follows:

(1) **MDIT** consists of a root node, a set of sub-trees, a set of transaction buckets and a global *MTDB* array.
(2) The **Root** node contains a set of inverted index items.
(3) Each node in the **Sub-trees** contains a set of inverted index items.

(4) **A Transaction Bucket** is a tuple *B={S, TIDs}*, where frequency field *S* denotes the support of the bucket, *TIDs* denotes a *TID list*.
(5) **A Global MTDB Array** is a global array $A=\{t_1, t_2,\ldots t_m\}$, where $t_1, t_2,\ldots t_m$ are the itemset part of transactions of the *MTDB*.

Example 4.1. We roughly sketch the MDIT construction process for the *MTDB* given in Example 4.1. Initially, MDIT is an empty root node and empty global array. We sequentially process all the transactions. For each transaction, we investigate its dimension values one by one, and after that, we store its *TID* and itemset part into the global *MTDB* array. See transaction 1. The first dimension value is *A1*, thus we add *A1* into root and generate a path *Root ->D ->E ->F* to memorize the inverted index {*A1, Any, Any: 1:1*}[2]. No further updates are made to MDIT by transaction *1* since the rest dimension values are "*Any*". Then we store {*1: I001, I002*} into the global *MTDB* array. The following transactions are processed until transaction *7*. The first dimension value is *A1*, so we update inverted index {*A1, Any, Any: 1:1*} into {*A1, Any, Any: 2: 1, 7*} by modifying transaction bucket *F*. The second dimension value is *B1*. We add *B1* into node *D* and generate a path *B1 ->A ->B* to store inverted index {*A1, B1, Any: 1:7*}. What's more, we need to update inverted index in transaction bucket *C* by registering *7* into its *TIDs* field and adding to its frequency field. The third dimension value is "*Any*", so we store {*7: I023, I025*} into the global *MTDB* array. In this way, after all *11* transactions are processed, we got the MDIT shown in Figure 1.

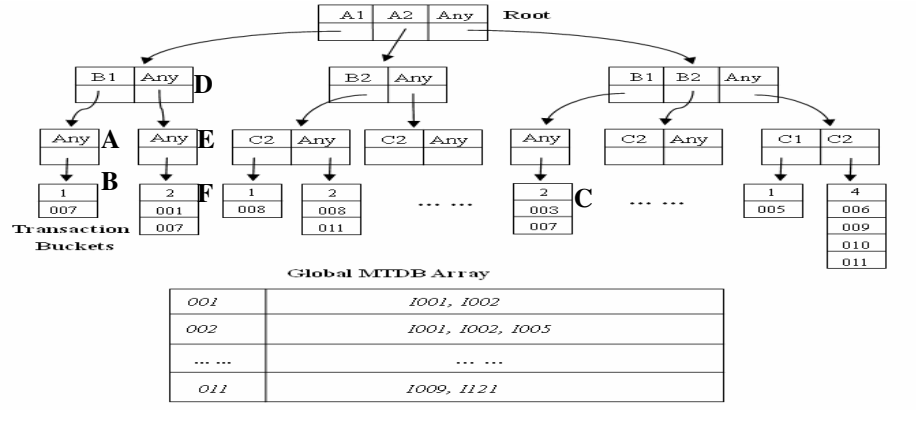

Fig. 1. Multi-dimensional Index Tree (MDIT)

4.2 Construction Algorithm of MDIT

Based on above illustration, the MDIT building algorithm can be summarized below.

Algorithm 1 (**Build-MDIT**)**.** Construction of MDIT
Input: A multi-dimensional database $MTDB=\{mt_1, mt_2,\ldots, mt_m\}$ with *n* dimensions $(P_1, P_2,\ldots, P_n)$.

[2] The inverted is represented as *{multi-dimensional values combination: frequency:TID list}*.

Output: A MDIT consisting of: (1) A MDIT tree[3] and (2) A global *MTDB* array
Method:
(1) Initialize the MDIT with *root* and an empty global *MTDB* array;
(2) **for** each $mt_i \in MTDB$
(3) **for** each dimension value p_j of dimension P_j in mt_i sequentially
(4) **if** there is no p_j 's multi-dimensional index item in level *j*
(5) Add p_j 's multi-dimensional index items into level *j*;
(6) Register the *TID* of mt_i into corresponding set of transaction buckets associated with p_j, $BS=\{B_1, B_2, \ldots B_k\}$;
(7) **for** each bucket $B \in BS$, $B->S++$;
(8) **else**
(9) Register the *TID* of mt_i into corresponding set of transaction buckets associated with p_j, $BS'=\{B_1, B_2, \ldots B_{k'}\}$;
(10) **for** each bucket $B \in BS'$, $B->S++$;
(11) Add the *TID* and the itemset part of mt_i into the global *MTDB* array;

5 Algorithm MDIT-Mining

5.1 Algorithm for Multi-dimensional Frequent Patterns Mining

Given the MDIT, one can mine multi-dimensional frequent patterns directly. Suppose the user wants to find all multi-dimensional frequent patterns from MDIT shown in Figure 1. Assume the minimum support threshold, η to be *2*. Just as what we did in case of inverted indices, firstly we search the MDIT to find all the buckets whose support is equal to or above *2*. These buckets are *{1, 7: 2}*[4], *{8, 11: 2}*,…, *{6, 9, 10, 11: 4}*. Secondly, for each frequent bucket, we get its projected database via its *TID list* and mine frequent patterns wherefrom. Finally, we output them in conjunction with their corresponding multi-dimensional values combinations.

Example 5.1. For the transaction bucket *{003, 007: 2}* (*C* in Figure 1), transaction *3* and *7* constitute a projected database as shown in *Table 1*, and they share the same frequent multi-dimension values combination {*Any, B1, Any*}.

Table 1. Projected database represented by *TID 3* and *7*

TID	*Items Sold*
003	*I023,I334*
007	*I023,I025*

Then we mine frequent patterns from this projected database and get frequent pattern *{I023: 2}*[5]. Concatenated with its multi-dimensional values combination, we

[3] For convenience, we call the tree part of MDIT (consisting of root, sub-trees, and transaction buckets) a MDIT tree.
[4] The format is *{TID list: frequency}*.
[5] The format of frequent pattern is *{Itemset: frequency}*.

get a multi-dimensional frequent pattern *{I023}∧{Any, B1, Any}* whose support is *2*. In this way, we can get all the multi-dimensional frequent patterns in *Table 2*.

Table 2. The resultant multi-dimensional frequent patterns

Multi-dimensional frequent patterns	*Support*
{I023}∧{Any, Any, C2}	*2*
{I025}∧{Any, Any, C2}	*2*
{I023}∧{Any, B1, Any}	*2*

This leads to our algorithm for multi-dimensional frequent patterns mining as below.

Algorithm 2 (MDIT-Mining). Mining multi-dimensional patterns based on MDIT
Input: MDIT and the minimum support threshold, η.
Output: The complete set of multi-dimensional frequent patterns, where each pattern consists of: (1) the multi-dimensional values combination and (2) a set of items.
Method:

```
(1)  for each transaction bucket B{
(2)    if (B->S>= η){
         //get the multi-dimensional values combination
(3)      Mdvc=the set of dimensional values of the inverted indices from root to B;
         //mine the frequent patterns in the projected database
(4)      Construct FP-Tree with the projected database represented by B->TIDs;
(5)      FP=FP_Growth(FP-Tree, η);
(6)      for each pattern p□FP output p in conjunction with Mdvc.
(7)    }
(8)  }
```

6 Performance Study

This section reports the performance study of the two algorithms: Build-MDIT and MDIT-Mining in comparison with data cube method on various datasets. All theses algorithms are implemented in *Microsoft Visual C++ 6.0*. Experiments are performed on a *1.6GHz* AMD PC with *448MB* main memory, running *Microsoft Windows XP Professional*. All algorithms together with data generator are available through address [14]. Regarding the data cube method, we develop a simulating algorithm to estimate its running costs on basis of paper [2]. We have implemented various simplifications to reduce the estimation of both its time and space consuming including:

(1) Ignoring the space consumed by all the other cuboids generated except the base one. i.e. For the data cube with dimensionality of D, among the (2^D-1) cuboids, we only consider the base cuboid and ignore all the other (2^D-2) cuboids.

(2) Supposing each cube cell only takes the space of an integer (which should actually store a *TID list*).
(3) Ignoring the space cost by the multi-dimensional database.
As a notational convention, the following symbols are adopted.

Table 3. Symbols

Symbol	*Meaning*	*Symbol*	*Meaning*
D	*Number of dimensions*	*C*	*Cardinality of each dimension*
T	*Number of transactions*	*M*	*Max_item in the MTDB*
L	*Maximum transaction length*	*S*	*Minimum support*

6.1 Space Scalability of MDIT

Figure 2 shows how the storage size of MDIT scales as dimensionality increases in comparison with data cube. The experiments were conducted on dataset (*10000-T*). The storage space of MDIT only approximately doubles as dimensionality increases by *1*. However, the storage space of data cube would increase about $(C+1)=1001$ times as dimensionality increases by *1* (Note that time axis is in logarithmic scale).

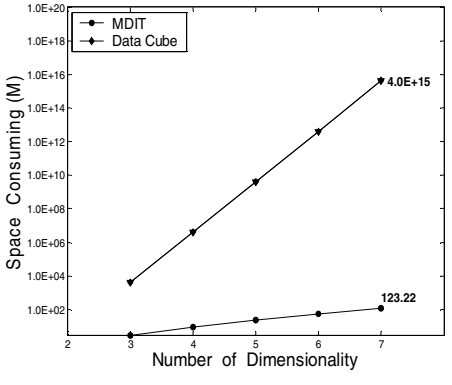

Fig. 2. Storage size of MDIT vs. data cube (10000-*T*)*T*=10000, *C*=1000, *M*=10000, *L*=20

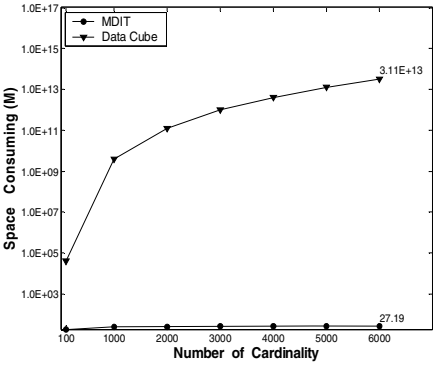

Fig. 3. Storage size of MDIT vs. data cube (5-*D*) *D*=5, *T*=5000, *M*=5000, *L*=10

Figure 3 demonstrates how the cardinality affects the space growth of MDIT and that of data cube on dataset (*5-D*). The space of data cube is $S=O((C+1)^{D})$[1, 2]. Although it does not grow exponentially along with number of cardinality, the growth rate actually is greater and greater as cardinality increases ($S'= O(D(C+1)^{D-1})$), which ultimately leads to even worse space explosion. However, the space of MDIT

almost keeps unchanged with cardinality[6]. When cardinality reaches *6000*, the storage of MDIT remains under *30 MB*, while the space of data cube has reached *12* orders of multitudes that of MDIT and surpassed the capacity summation of all existing storage devices. Overall, space consuming of MDIT is at least *4* orders of multitudes smaller than that of data cube. In addition, this gap grows wider as dimensionality or cardinality increases.

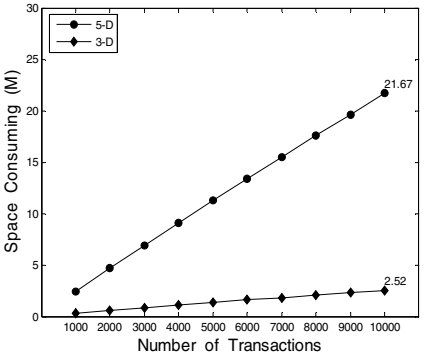

Fig. 4. Storage size of MDIT: (5-*D*) *D*=5, *C*=200, *M*=10000, *L*=20. (3-*D*) *D*=3, *C*=200, *M*=5000, *L*=10

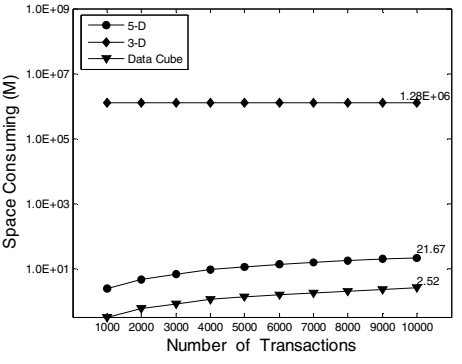

Fig. 5. Storage size of MDIT vs. Data cube: (MDIT 5-*D*) *D*=5, *C*=200, *M*=10000, *L*=20. (3-*D*) *D*=3, *C*=200, *M*=5000, *L*=10. (Data cube 5-*D*) *D*=5, *C*=200

Fig. 4 and Fig. 5 show the space scalability of MDIT along with number of transactions. Our experiments were conducted on two different datasets (*5-D*) and (*3-D*). From Figure 4 we can see that storage space grows linearly as number of transactions grows. This is expected because each transaction would at most increase a certain amount of storage, $(2^D\text{-}1)*(sizeof(tid)+sizeof(I)+sizeof(P) = 12*(2^D))$, as explored in space complexity analysis. In Figure 5, we compare the space consuming of MDIT with that of complete data cube. In the range transactions number varies from *1000* to *10000*, the space of MDIT is less than *22MB*, while the storage of data cube reaches 1.28×10^6 *MB*, more than 10^4 times that of MDIT.

7 Conclusion

In this paper, an efficient method is proposed to mine multi-dimensional frequent patterns without data cube construction. Our contributions are (1) proposing the concept of multi-dimensional frequent pattern, (2) proposing a novel MDIT structure and a MDIT-based multi-dimensional frequent patterns mining method (MDIT-Mining), and (3) conducting extensive experiments that show space of MDIT is more than 4 orders of multitudes smaller than data cube along with the increase of dimensionality or cardinality.

[6] There is slight growth because larger cardinality causes lower sharing rate of multi-dimensional index items, and hence more multi-dimensional index items are generated.

References

1. Data Mining: Concepts and Techniques. Jiawei Han, Micheline Chamber, Morgan Kaufmann, Hardcover, ISBN 1558604898
2. Kamber, J. Han, and J.Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KKD'97.
3. R. Baeza-Yates. et. al Modern Information Retrieval. Addison-Wesley, 1999.
4. G. Piatetski-Shapiro. Discovery, analysis, and presentation of strong rules. In Knowledge Discovery in Databases, pages 229-248, 1991.
5. Rakesh Agrawal, et.al Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD 1993
6. Han, J., Pei, J., and Yin, Y. Mining Frequent Patterns without Candidate Generation. SIGMOD, 1-12. 2000.
7. D. Barbara and M. Sullivan. Quasi-cubes: Exploiting approximation in multidimensional databases. SIGMOD Record, 26:12-17, 1997.
8. S. Agarwal, et. al. On the computation of multidimensional aggregates. In Proc. 22nd VLDB, pages 506--521, Mumbai, Sept. 1996.
9. Y. Sismanis and N. Roussopoulos. The dwarf data cube eliminates the high dimensionality curse. TR-CS4552, University of Maryland, 2003.
10. Xiaolei Li, Jiawei Han, and Hector Gonzalez. High-Dimensional OLAP: A Minimal Cubing Approach. In VLDB'04.
11. J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. VLDB 95, pages 547--559,
12. Y.D. Cai, N. Cercone, and J. Han. *Attribute-oriented induction in relational databases*. In Knowledge Discovery in Databases, 1991.
13. Peng Huairen. The First Volume of Great Formula Dictionary of TCM. People's Medical Publishing House. December 1993
14. http://cs.scu.edu.cn/~lichuan/lichuan.rar