Abstract—Construction of a compact FP-tree ensures that subsequent mining can be performed with a rather compact data structure. This does not automatically guarantee that it will be highly efficient since one may still encounter the combinatorial problem of candidate generation if one simply uses this FP-tree to generate and check all the candidate patterns. We study how to explore the compact information stored in an FP-tree, develop the principles of frequent-pattern growth by examination of our running example, explore how to perform further optimization when there exist a single prefix path in an FP-tree, and propose a frequent-pattern growth algorithm, FP-growth, for mining the complete set of frequent patterns using FP-tree.

I. INTRODUCTION

There are two methods for database projection:
1. Parallel projection
2. Partition projection.

A. Parallel projection

Parallel projection is implemented as follows: Scan the database to be projected once, where the database could be either a transaction database or a α-projected database. For each transaction T in the database, for each frequent item ai in T, project T to the ai -projected database based on the transaction projection rule, specified in the definition of projected database. Since a transaction is projected in parallel to all the projected databases in one scan, it is called parallel projection. The set of projected databases shown in of Example 2 demonstrates the result of parallel projection. This process is illustrated in figure 1. Parallel projection facilitates parallel processing because all the projected databases are available for mining at the end of the scan, and these projected databases can be mined in parallel. Since each transaction in the database is projected to multiple projected databases, if a database contains many long transactions with multiple frequent items, the total size of the projected databases could be multiple times of the original one. Let each transaction contains on average l frequent items. A transaction is then projected to 1 - 1 projected database. The total size of the projected data from this transaction is 1 + 2 + \cdots + (l- 1) = l (l-1) 2. This implies that the total size of the single item-projected databases is about l-1 2 times of that of the original database.

B. Partition projection

Partition projection is implemented as follows. When scanning the database (original or α-projected) to be projected, a transaction T is projected to the ai -projected database based on the transaction projection rule, specified in the definition of projected database. Since a transaction is projected in parallel to all the projected databases in one scan, it is called parallel projection. The set of projected databases shown in of Example 2 demonstrates the result of parallel projection. This process is illustrated in figure 1. Parallel projection facilitates parallel processing because all the projected databases are available for mining at the end of the scan, and these projected databases can be mined in parallel. Since each transaction in the database is projected to multiple projected databases, if a database contains many long transactions with multiple frequent items, the total size of the projected databases could be multiple times of the original one. Let each transaction contains on average l frequent items. A transaction is then projected to 1 - 1 projected database. The total size of the projected data from this transaction is 1 + 2 + \cdots + (l- 1) = l (l-1) 2. This implies that the total size of the single item-projected databases is about l-1 2 times of that of the original database.
list of frequent items. That is, the projected database of the least frequent item is mined first, and so on. Each time when a projected database is being processed, to ensure the remaining projected databases obtain the complete information, each transaction in it is projected to the adjacent projected database, where aj is the item in the transaction such that there is no any other item after aj in the list of frequent items appearing in the transaction. The partition projection process for the database in Conclusion: distributed parallel computation technique or multi-CPU to solve this problem. But these methods apparently increase the costs for exchanging and combining control information, and the algorithm complexity is also greatly increased, cannot solve this problem efficiently. Even if adopting multi-CPU technique, raising the requirement of hardware, the performance Improvement is still limited.

REFERENCES


[3] Han, J., Pei, J., and Yin, Y. Mining Frequent Patterns without Candidate Generation. SIGMOD, 1-12. 2000.


[14] Pei, J., Han, J., Nishio, S., Tang, S., and Yang, D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc. 2001 Int. Conf. on