Volume 4, Issue 4, April 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# An Approach towards Information Extraction Based on Partitioning

**Sonam S.Chauhan, Prashant R.Deshmuk**

[1]*Department of Information Technology,*

*SGBAU University, India*

***Abstract— In this paper we describe an approach for information extraction in which we partition the dimensions (attributes), i.e., a higher dimension of large data set can be transformed into relatively smaller subsets of data in certain numbers that might be processed easily. Many techniques have been proposed in this but works on single database. Here we use heterogeneous database as large data set. Thereafter, based on the separation of dimensions, the discernible dataset of all data are computed so as to get their core attribute sets. Furthermore, the attribute reduction and data redundancy methods are used to obtain the partition results. In our approach towards information extraction we partition are data sets so that information can be extracted easily and correctly with query processing which generate results in less time. Feasibility of our incremental extraction approach can be demonstrated by performing experiments to spotlight two important aspects of an information extraction system: quality of extraction results and efficiency.***

***Keywords— Heterogeneous databases, multidimensional data, partitioning, association rules, algorithm, itemsets.***

## I. INTRODUCTION

In the recent years our capabilities of both collecting and generating data have been grown rapidly. In business management, government administration, scientific and engineering data management, and much other application millions of databases have been used. It has been noted that the number of such databases is growing rapidly as a result of the availability of affordable and powerful database system. This rapid growth in databases and data has resulted in urgent need for new tools and techniques that can rapidly, intelligently and automatically transform the processed data into more useful information and knowledge. Consequently, data extraction has become a research area with increasing importance [1, 2, 3]. By knowledge discovery in database, interesting knowledge, regularities, or high-level information can be extracted from the relevant sets of data in database and can be investigated from different angles and large databases thereby serve as rich and reliable sources for knowledge generation and verification. Knowledge and Information from huge databases has been reorganized as a key research topic by many researchers database system. The discovered knowledge can be applied to information management, query processing, decision making, process control and much application. Furthermore several emerging application in information providing services, also call for various data extraction techniques to better understand user behaviour, to meliorate the service provided, and to increase opportunities. In response to such a demand, here we describe an approach for information extraction in which we partition the dimensions (attributes), i.e., a large data set having a higher dimension can be transformed into certain number of relatively smaller data subsets that can be easily processed. Here we use heterogeneous database as large data set. Then, based on the partitioning of dimensions, the discernible dataset of all data are computed to obtain their core attribute sets. Furthermore, the attribute reduction and data redundancy methods are used to obtain the partition results. In our approach towards information extraction we have design such an algorithm which automatically partitions the data from heterogeneous database and then combine this data into single one.

## II. PREVIOUS WORK

Extracting knowledge and information from huge databases has been recognized as a key research topic by many researchers in database system and machine learning and by large number of industrial companies as an important area with an opportunity of major revenues. Many papers and techniques have been proposed for providing solutions in data extraction from bulky data stored. The major source of our inspiration is early work by Jin Zhou, Liang Hu, Feng Wang, Huimin Lu, and Kuo Zhao in [1]. In this paper, an efficient multidimensional fusion algorithm is proposed for IoT data that is based on partitioning. Partitioning is the basic concept involved here. In rough set theory the discernible matrixes of all data subsets are computed based on the partitioning of dimensions to obtain their core attribute sets. Moreover, a global core attribute set can be determined. Finally, fusion result is obtained by the rule extraction and attributes reduction methods. By means of proving a simulation and a few theorems, the effectiveness and correctness of this algorithm is illustrated. Many caching mechanism was also described in many papers like DB Cache [4] which could extract information from bulky data in less time, however many times threatened the goal of quality of extraction result.

We also referred to some papers describing the algorithms for association rules like Apriori algorithm [2], DHP algorithm [3]. However it was discovered that the step of determining itemsets by scanning the whole database and testing each transaction against the hash tree build is very expensive.

## III. PROPOSED WORK

The Partition Algorithm has three major features: one is efficient generation of partition database, second is update in partition database and third is maintaining Threshold value. Initially at first iteration the data is brought from the original database. An item count is maintained in the history table. These items are added to the partition database. After a fix number of interval (transactions) the partition database is updated. Updates are performed with respect to individual item value as well as number of attributes. After each transaction a new threshold value is computed and accordingly the attributes are inserted in partition database. The new threshold value is calculated as,

Threshold=tcount/10

Where tcount = Total number of transactions.
Value 10 = update interval.

According to the threshold value attributes are inserted or removed from the partition database. The algorithm also maintains hit rate i.e., how many time the Query is answered from partition database and query frequency i.e., how many time the same query is asked in n number of transactions.

The algorithm scans through the original database only when the partition database does not answer the query. This reduces the scanning overheads. However by using the traditional methods of answering query, it needs to scan the whole database again and again to answer each query, overhead increaser when working with large, bulky database. Using heterogeneous databases with same is more time consuming.

Using partition approach we can partition are data sets so that information can be extracted easily and correctly with query processing which generate results in less time**.** The figure1 shows the working of partition algorithm. The itemsets and their support are maintained in the history table. A scan is made for maximum asked item. Threshold value is calculated and accordingly the items are inserted in the partition database. This happens after a fixed number of transactions. Here we have set update after every 10 transactions. Whenever a query is fired search is made in the partition database first if the partition database is unable to answer the query the search is made in original heterogeneous databases. It is noted that given large databases, the initial extraction of useful information from the databases is usually the most costly part. A scan is made for the total support of each item. Maximum support items are scan out and threshold is applied. Attribute with max support according to the threshold value is added in the partition database. Figure1 gives the algorithm for Partitioning. Implementation detail is omitted in figure1.Partation reduces the database size progressively by not only trimming each individual transaction size but also pruning the number of transactions in database.

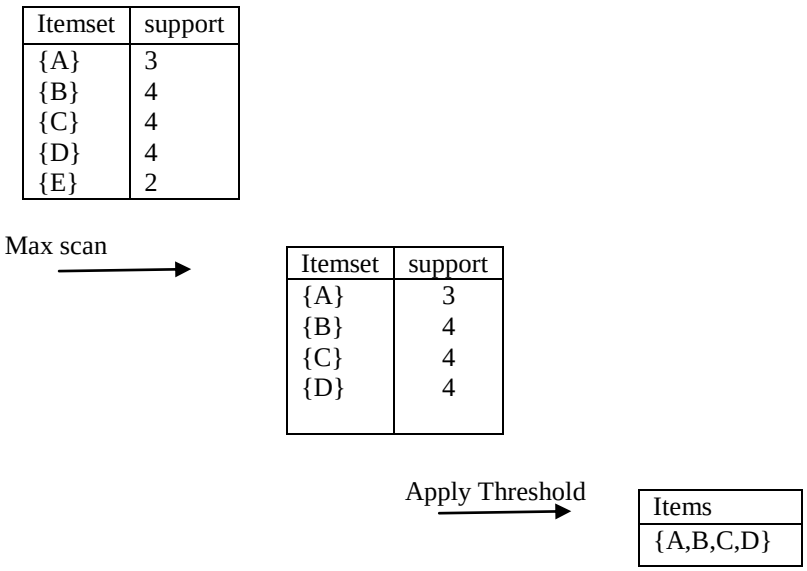



Figure1: Generation of Itemset

## IV. PERFORMANCE

The table1 gives the performance of partition algorithm against different number of updates. As shown in the table as the number of transaction increases, the time taken to retrieve data from database decreases. This is so because as the transaction increases items are added in the database according to the partition algorithm. Once an item is added to the partition database it needs not to be searched against different heterogeneous databases. However the item can be removed from the partition database if it is not frequently asked in queries. This will depend on the history maintained by the algorithm. The table gives the Partition database results after Updates, where updates are made after 10 Transactions each and Database size is 100items.The Figure2 graphically shows the performance of partition algorithm.

Table1: Partition database results after Updates, Updates after each 10 Transactions (DB size 100)

| No. Of Updates | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Execution time (millisecond) | 64 | 74 | 52 | 32 | 32 | 32 | 32 | 29 | 16 | 16 |

Table2 gives the performance of algorithm as the number of transaction increases. Here we have used database size of 50 items at first to note the execution time against different number of transaction. It can be seen from the table that as the number of transaction increases the time taken by original database is much more than the time taken by the partition database. The Figure3 graphically shows the performance of partition algorithm by comparing the execution time against different number of transactions.

Table2: Performance of Partition Database as number of Transactions increases (DB size 50)

| Transaction | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|---|---|
| Original DB | 108 | 110 | 120 | 130 | 140 | 150 | 160 | 170 |
| Partition DB | 16 | 17 | 15 | 16 | 15 | 14 | 14 | 14 |

Table3 gives the performance of algorithm by compare the execution time against different Database Sizes. It is noted that as the size of the database increases the time taken by original database to retrieve data is much more than the time taken by the partition database. The Figure4 graphically shows the performance of partition algorithm by comparing the execution time against different size of database.

Table3: Compare against Different Database Size (no. of records)

| DB Size | 100 | 200 | 300 | 400 | 500 | 600 |
|---|---|---|---|---|---|---|
| Original DB | 108 | 425 | 1015 | 1025 | 1034 | 1045 |
| Partition DB | 16 | 21 | 31 | 32 | 36 | 40 |

Table4 shows the execution time when the update interval is varied. Here we have examine the execution time by setting update after every 5, 10, 15 transaction resp. It is noted that update after every5 transaction give the best result. However frequent updating in database also consumes time as the updated data need to be brought from main database to partition database. Hence we computed are result with update interval of 10 transactions. Figure5 give the execution time.

Table4: Database size=100, Update value: 5, 10, and 15

| No. of Update | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 64 | 49 | 46 | 32 | 16 | 16 | 16 | 16 | 16 | 16 |
| 10 | 64 | 74 | 52 | 32 | 32 | 32 | 32 | 29 | 16 | 16 |
| 15 | 64 | 32 | 32 | 28 | 28 | 28 | 16 | 16 | 16 | 16 |

Support for an item is the number of time the item is asked in n numbers of transaction.in this paper we have computed the support for 2, 4, 6, 8, and 10 for an item and recorded the execution time. It is noted that as the support for an item increaser its retrieval time decreases. Table5 gives execution time against different support value. Figure6 represents its graph.

| Support | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| Execution time(millisecond) | 150 | 130 | 16 | 16 | 16 |

Table 5: Support (Average No. of times an item is asked in N transactions), no updates, DB100 and reading recorded when transaction =50 therefore; threshold value= 50/10=5

We have also recorded execution time when threshold is varied. It is noted that as support increases the execution time decreases with decreasing value of threshold. Hence when threshold is set to lower value with increasing value of support, the performance of database increases. . It is as shown in figure7.

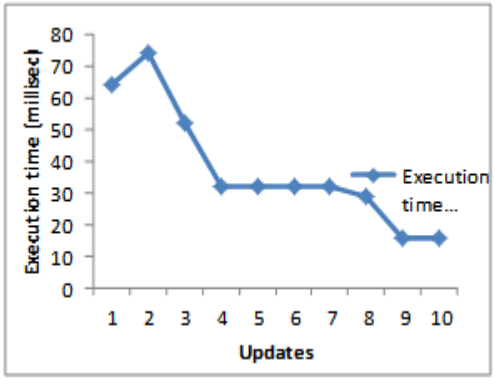



Figure2: Execution Time

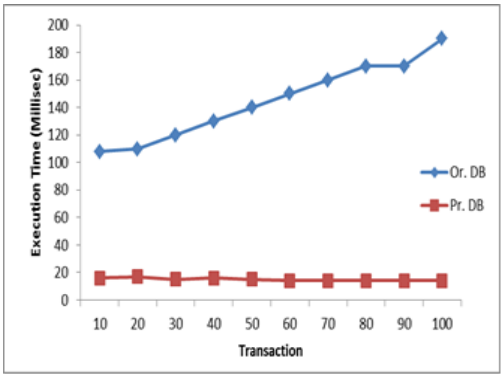
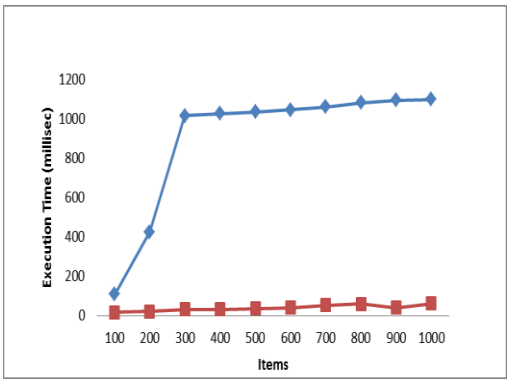



Figure3: Execution Time




Figur4: Execution Time

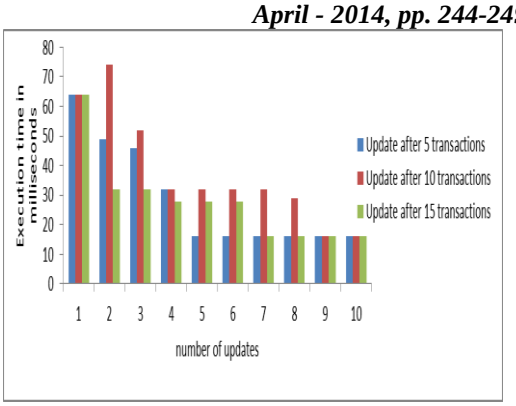



Figure5: Execution Time

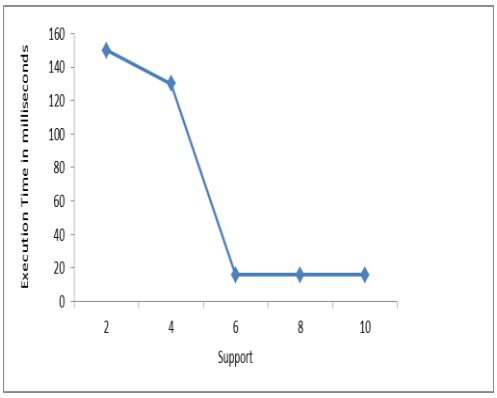



Figure6: Execution Time

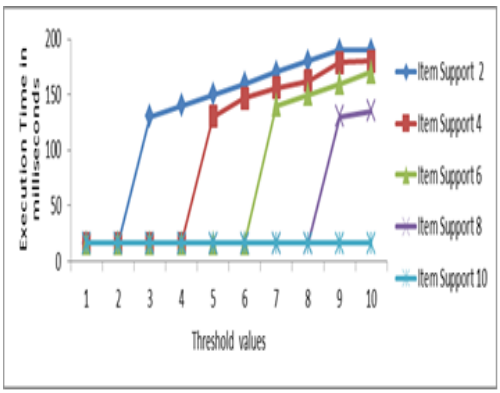



Figure7: Execution Time

## V. RELATIVE PERFORMANCE

Figure8 shows the execution time for 5 different size of database. In this comparison is made among three different algorithms as DHP, Cache, and Partition. It is noted that in the initial iteration DHP performs better than the other two algorithms but as the size of database increases Partition start giving better results. However cache beets DHP and Partition Algorithms when size of database increases.

In our approach, we have exploited complementary tactics for reducing the response time of a query (i.e. speeding up a query). Response time may be reduced by decreasing the total work to compute a query which is reduced by partitioning work. There is no need to go through each database and search for result. We simply use a partition table. The "hit rate" describes how often a searched-for item is actually found in the Partition Database. In our work algorithm maintains a history of more usage information in order to improve the hit rate .Query frequency is determined and result are noted against hit rate. The figure9 show the hit rate, query frequency and its retrieval time for different cache mechanism [4] and partition approach. It is noted that Partition performs the best when hit rate and retrieval time is taken together.

Figure10 compares the partition algorithm with Apriori, AprioriTid [2], and Cache algorithm [4]. It is noted that Apriori and Partition Algorithm beets the other two at initial iteration but Partition performs the best as number of transactions increases.

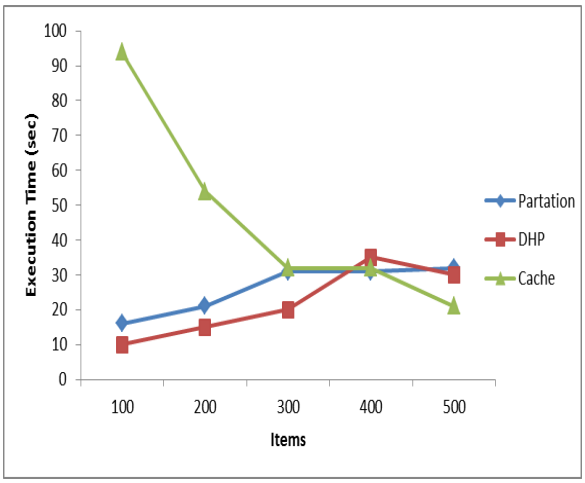



Figure8: Execution Time (Ref. 3, 4)

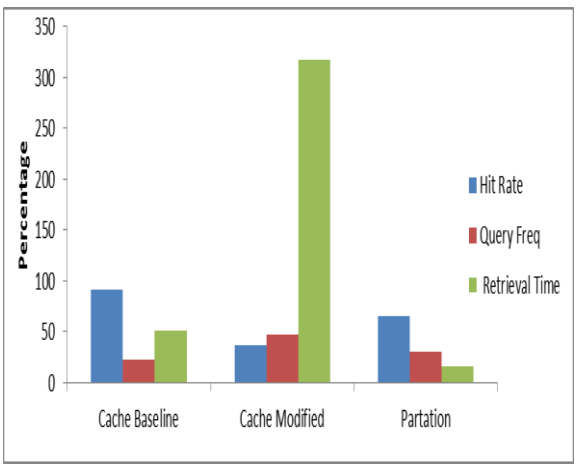



Figure9: Comparing partition and Cache Mechanism (Ref: 4)

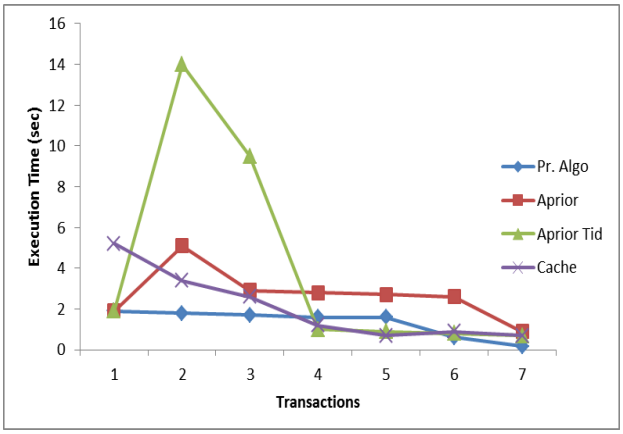



Figure10: Execution Time (Ref: 2, 4)

The Partition Algorithm proposed by us not only deals with text data but can also be used to extract images form database in less time. The mechanism implemented on text is also implemented on image and results are computed. It is seen that the partition mechanism also gives better results when extracting large images from bulky database. The table6 show the execution time against different updates interval.Figure11 shows its graphical representation. The Table7 gives the execution time needed when extracting image from original database and that from partition database using partition algorithm.it is noted that the extraction of images from original database takes uh more time then Partition database. Thus our partition database is also suitable or working on image database.Figure12 gives its graphical illustration.

| No. Of Updates | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Execution Time(millisecond) | 72 | 63 | 54 | 32 | 32 |

Table6: For Image database partition database results after Updates, DB size 300, Updates after each 10 Transactions

| Transaction | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| Original DB | 162 | 123 | 108 | 120 | 108 |
| Partition DB | 72 | 63 | 52 | 32 | 32 |

Table7: For Image database Performance of Partition Database as number of Transactions increases (DB size 300

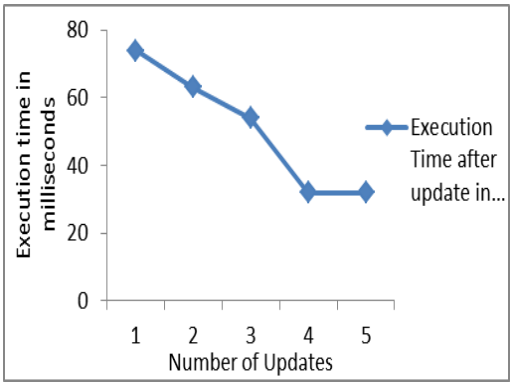



Figure11: Execution Time

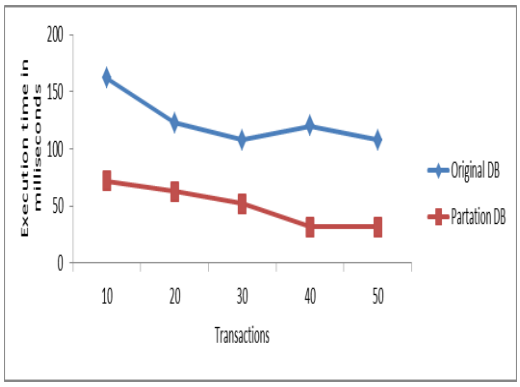



Figure12: Execution Time

## VI CONCLUSION

We proposed new algorithm, Partition Algorithm, for extracting data from large database in less time. We compared this algorithm with number of previously known algorithms. The experimental results show that the proposed algorithm outperforms many of the existing algorithms. The performance gap increases with the increasing number of transactions and database size. These experiments demonstrate the feasibility of using Partition Approach in real application involving very large database. The Partition Approach proposed by us can be used to extract data from any Heterogeneous databases in less time. However it is more suitable to databases were updating of database is less as large updating results in frequent updating in Partition database, increment in estimation cost. The advantage of this approach is that it transforms a big problem into smaller problems that are easier to process; this substantially reduces the temporal and spatial computational complexities.

## REFERENCES

**[1]** Jin Zhou, Liang Hu, Feng Wang, Huimin Lu, and Kuo Zhao in "An Efficient Multidimensional Fusion Algorithm for IoT Data Based on Partitioning" Aug 2011.

**[2]** Rakesh Agrawal and Ramakrishnan Srikant: Fast Algorithm for Mining Association Rules.1994.
**[3]** Jong Soo park "An Efficient Hash Based Algorithm for Mining Association Rules"
**[4]** Khalil Amiri "DB Proxy: A dynamic data cache for web application"
**[5]** U.M. Fayyad, G.Piatetsky-shapiro, P.Smyth, and R. Uthurusamy, Advances in knowledge Discovery and Data Mining, Mining, AAAI 1996.
**[6]** G.Piatetsky-shapiro Advances in knowledge Discovery and Data Mining, AAAI/MIT 1991.
**[7]** A.Sliberschatz, M.Stonebraker, and J.D.Ullman,"database Research: Achievement and opportunities into 21st Century,"May1995.
**[8]** H. Patni, C. Henson, and A. Sheth, Linked sensor data in Proc. 2010.
**[9]** A. K. Joshi, P. Jain, P. Hitzler, P. Z. Yeh, K. Verma, A.P. Sheth, and M. Damova, Alignment-based querying of linked open data, in Proc. on the Move to Meaning Internet Systems: OTM 2012, Confederated International Conferences: Coop IS, DOA-SVI, and ODBASE 2012, Rome, Italy, 2012, pp. 1-18.
**[10]** A. Rajaraman, J. Leskovec, and J. D. Ullman, Mining of Massive Datasets. Cambridge University Press, 2010.
**[11]** N. Pham and R. Pagh, A near-linear time approximation algorithm for angle-based outlier detection in high dimensional Data, in Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12), Beijing, China, 2012, pp. 877-885.
**[12]** S. Kahramanli, M. Hacibeyoglu, and A. Arslan, Attribute reduction by partitioning the minimized discernibility function, 2011.