International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-3 Issue-11 September-2013

Association Rule Mining Analyzation Using Column Oriented Database

D. P. Rana[1], N. J. Mistry[2], M. M. Raghuwanshi[3]

Abstract

The logical view of data is a two dimensional table and the physical storage is a single dimensional. Two approaches exist to map two dimensional data on to a single dimensional storage: Row oriented and Column oriented. Common database applications are developed using traditional row-oriented database systems. Data Mining (DM) is a promising research area, deals with huge data with large numbers of attributes and records. DM algorithms are more analytical in nature with the goal of reading through the data to gain new insight and use it for planning make Column oriented database systems more preferable. The Column oriented database systems show better performance than traditional database on analytical workloads such as those found in data warehouses, decision support, and business intelligence applications. The Column oriented databases like MonetDB is utilized for performance analysis of SQL queries. This paper is focused on the utilization of Column oriented databases like MonetDB with Oracle 11g - the famous Row oriented database for execution time analysis for famous DM algorithm: APRIORI. Experiment results show the faster execution time of MonetDB compare to Oracle for different supports and justifies the suitability of the Column oriented database for such data mining algorithm.

Keywords

Apriori algorithm, Association Rule Mining, Column store database, Row store database.

1. Introduction

From last decades, all areas of our society strongly depend on the information technology. The amount of data stored and processed worldwide in information systems has grows enormously.

D. P. Rana, Computer Engineering Department, Sardar Vallabhbhai National Institute of Technology, Surat, India.
N. J. Mistry, Civil Engineering Department, Sardar Vallabhbhai National Institute of Technology, Surat, India.
M. M. Raghuwanshi, Rajiv Gandhi College of Engineering and Research, Nagpur, India.

In data bases, data stores in tabular form where rows correspond to relationships or entity (Records or instance of a table) and columns are the attribute or features. If the expected process tends to access data on the granularity of an entity e.g., Display complete information of a student, Add new student, Delete complete information of a Student etc., then the row-by-row storage is preferable since all of the needed information will be stored together i.e. RDBMS (Relational Database Management System). On the other hand, if the expected process tends to deal with only a few attributes from many records of a table, e.g., a query that finds the most common favorite book, then column-by-column storage is preferable since other attributes are irrelevant for a particular query and need not have to be accessed. This type of application needs Columned Database [1] - [3].

Data mining is the creation of new knowledge in natural or artificial form, by using business knowledge to discover and interpret patterns in data [4]. Major data mining task like Outlier analysis, Classification, Association Rule mining etc., need to analyze attribute(s) individually so column oriented database (Column store) is well suited for many of these algorithms.

In 1993, the author R. Agrawal et al. proposed association rule mining algorithm to discover the relation between the items/attribute values [5]. This paper is targeting only association rule mining technique.

This paper is describing the study and performance analysis of the Row store vs. Column store DBMS and performance impact of column store DBMS with association rule mining. The next Section 2 discusses the study of column database storage with its advantages and disadvantages. Section 3 provides brief introduction and literature survey of Association rule mining approaches and their limitations with respect to column database. The Section 4 analyzes the execution time of the association rule mining algorithm with row oriented and column oriented database, which is followed by conclusion in Section 5.

2. Column vs. Row Data Store System

Row data store system access whole record with every query despite of involvement of attributes in the query, as every tuple is stored in a bundle in the secondary storage. While the column data store system will access individual columns quickly as every column is stored in a bundle in the secondary storage. In column store system, the two dimensional data table will be vertically partitioned and stored in the multiple tables containing two columns. The new table name contains: Table identifier and Column identifier. First column of the table will contain the tuple identifier and the second column will contain the column value. This makes write process and complete tuple access more expensive in columned database. So it is found that column store is read-optimized and row store is write-optimized. Column store is more efficient for analytics because it is not pulling in all of the unnecessary columns which are not the part of the query **[**3]. The advantages of column store over row store are as follows:

1. Data compression: Column data is of uniform type. Therefore it is much easier to compress than row data and NULL values need never be stored. Row stores cannot omit columns from any row and still achieve direct random access to a table, because random access requires that the data for each row be of fixed width. In column stores, this is trivially true because of type uniformity within a single column's storage, allowing omission of NULL values and therefore efficient storage of wide, sparsely populated tables. In practice, row store system can and does implement tables with variable-width rows, but this requires either some form of indirect access or giving up random access in favor of some type of fast ordered access.
2. Improved Bandwidth Utilization: For tables with many columns and queries that use only few of them, a column store can confine its reads to the columns required, whereas a row store must read the entire table. The Row stores are extremely "write friendly" as adding a row of data to a table requires a simple file appending I/O and column stores perform better for complex read queries.
3. Improved Code Pipelining: The storage efficiency properties of column stores can greatly reduce the number of actual disk reads required to satisfy a query. The reduced irrelevant column access saves CPU cycle performance as we use the performance only for the required attributes.
4. Improved cache locality: The cache in the column oriented contains only the required data instead of the unnecessary data which is the case for the row oriented database.

Though, the column store system has a number of advantages, it also has disadvantages which are as follows:

1. Reduced Disk Performance: Multiple columns access needs parallel read operation and thus it will increase the seek time.
2. Poor insertion efficiency: Single record insertion needs to access every column tables and thus the higher insertion time is due to higher seek time and sparse storage of a record.

A. Column Oriented Databases

Numbers of column store system are available nowadays. HBase is an open source column oriented database system modeled on Google's BigTable [6]. Infobright is column oriented MySQL engine and almost all MySQL api's/interfaces/tools can be used though it's column oriented [7]. Infobright has its own proprietary data storage and query optimization layers. InfiniDB is an open source (GPLv2) by Calpont which supports most of the MySQL API and stores data in a column-oriented fashion, and is optimized for large-scale analytic processing [8]. MonetDB is a relational database management system that stores data in columns [9], [10]. C-Store is a read-optimized relational DBMS storage of data by column rather than by row with overlapping collection of column-oriented projections, rather than the current fare of tables and indexes [11], [12].

B. MonetDB System

In this section, we provide the necessary information of MonetDB database system as we have used it for the analysis. In MonetDB, every n-ary relational table is represented as a collection of Binary Association Tables called BATs [9] without any hole. For a relation R of k attributes, there exists k BATs, each BAT storing the respective attribute as (key, attr) pairs. The 'key' is system generated and identifies all attributes of the relational tuple. In MonetDB, SQL queries are translated by the compiler and the optimizer into a query execution plan that consists of a sequence of relational algebra operators. One or more MonetDB Assembly Language (MAL) instructions will be generated for each relational

operator. Each MAL instruction performs a single action using one or more columns in a bulk processing mode. Intermediate results are also maintained as temporary BATs in a column format. For example, SELECT R.z FROM R WHERE R.x < 10 AND R.y BETWEEN 11 AND 20;
This query is translated into the following (partial) MAL plan:

```
R.x1 := algebra.select(R.x, 0, 10);
R.y1 := algebra.select(R.y, 11, 20);
R.x2 := algebra.KEYintersect(R.x1, R.y1);
R.z1 := algebra.project(R.z, R.x2);
```

The MonetDB query processing scheme consists of three software layers. The pipeline is used to identified by the SQL global variable optimizer, which can be modified using a SQL assignment
1. The top layer is formed by the query language parser that outputs a logical plan expressed in MAL.
2. The code produced by MonetDB/SQL is passed and massaged by a series of optimization steps, denoted as an optimizer pipeline.
3. The MAL plans are transformed into more efficient plans enriched with resource management directives.

C. SQL Query Analysis

The analysis process carried out by comparing different query execution time on the column DBMS (MonetDB) with row DBMS (Oracle11g enterprise) for the real and synthetic dataset. The real dataset-GSS [11] is having 121 attributes and 10047 records and the synthetic dataset is created with 257 attributes and 1362 records. Since the datasets are not much large compared to real datasets, a milliseconds count can affect the result for this type of dataset.

The execution time analysis for a set of SQL queries applied on these two dataset on MonetDB and Oracle11g enterprise version is performed, which are as follows:

The execution time of MonetDB is much faster than the Oracle for the queries which include single column, two or more columns. The better performance of MonetDB is also achieved when execute the query with single or few column with condition on columns. MonetDB performance is excellent compared to Oracle on queries having group by, order by clause. For simple select all columns or entire row, query without aggregate function like sum, max, count, avg etc. the performance of MonetDB is quite poor compare to Oracle. But, if this query is with aggregate function than, the MonetDB performance is excellent.
MonetDB performance is more time consuming compared to Oracle when using complex queries like use of subqueries, join and view where only few columns is being accessed. From the result of query analysis we concluded that Oracle or row database is suitable for applications where we need to access entire row at any time, while MonetDB or columned database is suitable for applications where we need to access only set of columns instead of entire row access.

3. Association Rule Mining

The performance of column database is very much satisfactory where column access is more than the row access. This inference emphasized to work on some algorithms where column access is more. From literature survey it is observed that data mining algorithms have more column access (items or features) rather than entire row (all items or features).
The association rule mining is utilized in application domains such as market basket analysis, finance (to identify patterns that help be used to decide the result of a future loan application), environmental and satellite research (to identify potential undetected natural resources or to identify disaster situations like oil slicks), health care (to predict outbreaks of infectious diseases), web traffic analysis (to recommend the next web page), network (to detect botnet) etc. [14], **[**15**]**.

A large number of association rule mining methods have been reported in the literature, which have different mining efficiencies. Their resulting sets of rules are however all the same based on the definition of association rules. That is, given a transaction data set T, a minimum support and a minimum confidence, the set of association rules existing in T is uniquely determined. Any algorithm should find the same set of rules although their computational efficiencies, the number of times it scan the database, the structure it used to represent the transaction pages and memory requirements may be different.

In Apriori algorithm the classical association rule mining approach is used to discover the frequent item sets, which is the sets of items that have minimum support and then association rules are generated considering confidence threshold [16]. It follows the Apriori Property, “if an itemset is frequent, then all of its subsets must also be frequent“ and it proceeds by identifying the frequent individual items in the

database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined will be used to generate association rules which highlight general trends in the database.
Combinatorial explosion of the number of possible frequent itemsets and making of multiple passes over the data affects the performance of Apriori algorithm. Many variations of the Apriori algorithm have been proposed in literatures to improve the efficiency of the original algorithm. In 2000, the author J. Han et al. [17] proposed the FP-growth algorithm which uses only two data scans and also avoids the generation of a large number of candidate itemsets by storing complete FP-tree into the memory and in 2003, the author A. Pietracaprina et al. [18] proposed the modification to this using trie structure. A hash technique is very efficient. In 1995, the author Park et al. [19] proposed DHP algorithm using hash-based itemset counting to reduce the size of the candidate 2-itemsets and more. In 1997, the author Soo et al. [20] proposed the improvement over DHP in which during the process of candidate itemsets generation, it progressively also reduces the transaction database size by effective pruning techniques. It is useful particularly for the large two-itemsets, greatly improves the performance of the entire process.

In 2005, the author L. Zhi-Chao et al. [21] proposed transaction reduction technique in AprioriTid algorithm which relies on a concept that a transaction that does not contain any frequent k-itemset is useless in subsequent scans. It is generating candidate transaction database D' of the candidate frequent itemsets and mining is performed on the database D'. Thus, it reduces the time of I/O operation because D' is smaller than D.

In 1995, the author A. Savasere et al. [22] proposed partitioning technique to mine the frequent itemsets from the data within two database scan by dividing the database into small non overlapping partitions such that each partition can be handled in the main memory and finding the local large itemsets by using Apriori algorithm. And thus reducing disk I/O for each partition after loading the partition into the main memory.

Major works are discussed here which deal with boolean association rules, but still other research works are there. In 2012, the author E. Duneja et al. [23] has discussed the research review of association rule mining approaches focused on mining of multilevel association rules, multidimensional association rules and quantitative association rules.
In association rule mining, major database access is done during frequent itemset generation. Because of this the overall performance of mining association rules is determined by this process. Thus, in all above discussed approaches, the majority of related research has focused upon the efficient discovery of frequent itemsets as its level of complexity is greater than to generate association rule. These approaches are minimizing the execution time using different techniques like usage of special structure, transaction reduction, less number of data scans, etc. from transactional and/or relational databases.

From these approaches we can conclude that these techniques have not considered the physical storage of transaction data which is utilized in process of frequent item generation. From the study of column database and association rule mining approaches, we motivated to utilize the column database for storage instead of relational row database where it tries to access the complete transaction information rather than just to have required partly transaction information.

Up to the knowledge of the author the exploitation for the performance analysis of association rule mining using column database is untouched. Thus, this paper is analyzing the Boolean association rule mining with the column database.

4. Execution Time Analysis

The implementation is performed on Intel core i5 processors, 4 GB RAM with 64 bit windows operating system using JAVA language. We need to check the execution time of Apriori algorithm with more number of instances, so prepared Aprdata-synthetic data having more number of instances compare to market basket dataset. So, prepared the Aprdata which is binary dataset having values either 0 (not purchased item) or 1 (purchased item) with 10000 number of instances and 101 number of attributes are considered for the performance analysis.

The following Fig. 1 shows the execution time analysis of Apriori algorithm on both MonetDB and Oracle. From the analysis of result we can see that MonetDB performance is very much better than Oracle for Apriori algorithm. On analysis of Apriori algorithm we can notice that algorithm does only frequent column access. More column pairs are formed after each phase and only those column pairs are accessed rather than entire row. We can also see that on decreasing minsup value time for database increases and time difference become clearer. Decreasing minsup will allow more column pair to be form in each phase as a result the database has to perform more and hence more execution time.

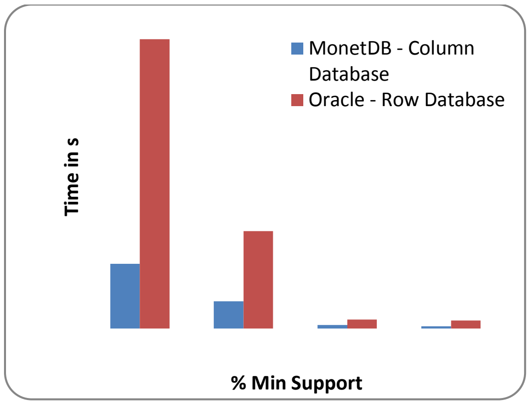

Fig. 1: Execution Time analysis of Apriori algorithm for 'Aprdata' dataset

Here, the noticeable execution time result is observed in Fig. 1, for Oracle (Row database). For Aprdata MonetDB and Oracle is showing less time difference in terms of 100th due to more number of instances. Thus, we can conclude that the execution time will be much less in column database compare to row database for large number of instances. Our result concludes that performance analysis of association rule mining using Apriori algorithm is much better with column oriented database compare to row oriented database. And hence we can say that access of data from column database for data mining algorithm results in faster than row database.

5. Conclusion

From the study and analysis in the previous sections, we conclude that Column oriented database performance is better than row oriented database when column related queries are more than row by execution of SQL queries on MonetDB and Oracle. Expectation of faster execution time with the column oriented database for data mining algorithms compared to row oriented database is analyzed for the Apriori algorithm on MonetDB and Oracle. And from the experimental results we achieved faster execution performance for MonetDB than Oracle. The faster execution for the algorithm on synthetic dataset shows the suitability of the column oriented database for such data mining algorithm. The research can be extended for Apriori algorithm with real values and to be tested on other real datasets.

Reference

[1] K. Venkat and K. Rakesh, "Column oriented databases vs row oriented databases," Special Interest Activity, ITK – 478, 2007.
[2] D. Abadi, "Column-stores for wide and sparse data," 3rd Biennial Conf. on Innovative Data Systems Research, California, USA, pp. 7 – 10, January 2007.
[3] D. J. Abadi, S. R. Madden and N. Hachem, "Column stores vs. row stores: How different are they really?," SIGMOD 2008, Vancouver, BC, Canada, pp. 9–12, 2008.
[4] J. Han and M. Kamber, Data mining: Concepts and techniques, 2nd ed. Morgan Kaufmann, 2006, ch. 6.
[5] R. Agrawal, T. Imielinski and A. Swami, "Mining association rules between sets of Items in large databases" in *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pp. 207 – 216, 1993.
[6] [Online]. Available: http://www.hbase.apache.org.
[7] [Online]. Available: http://www.infobright.org.
[8] Calpont, What is infinidb? http://infinidb.org/resources/what-is-infinidb, 2010.
[9] [Online]. Available: http://www.monetdb.org.
[10] S. Idreos, F. Groffen, N. Nes, S. Manegold, S. Mullender and M. Kersten: "MonetDB: Two decades of research in column-oriented database architectures," Data Engineering IEEE Computer Society, 2012.
[11] S. Kanade and A. Gopal, "Choosing right database system: Row or column-store," Intl. Conf. on Information Communication and Embedded Systems (ICICES), ISBN: 978-1-4673-5786-9, pp. 16 – 20, 2013.
[12] A. H. Jennifer, L. Beckmann, J. F. Naughton and D. J. DeWitt, "A comparison of c-Store and row-store in a common framework," in Proc. of the 32nd VLDB Conference, Seoul, Korea, 2006.
[13] [Online]. Available: http://www.mathcs.org/statistics/statcrunch/gss2008/.
[14] D. P. Rana, N. J. Mistry and M. M. Raghuwanshi, "Association rule mining for environmental data: A study," Intl. Congress of Environmental Research (ICER-11), India, December-2011.
[15] S. S. Garasia, D. P. Rana and R. G. Mehta, "HTTP botnet detection using frequent patternset mining," Intl. Journal of Engineering Science and Advanced Technology (IJESAT), ISSN: 2250–3676, vol. 2, no. 3, pp. 619 – 624, 2012.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," Proc. of the Twentieth Intl. Conf. on Very Large Databases, Santiago, Chile, pp. 487–499, 1994.
[17] J. Han and J. Pei, "Mining frequent patterns by pattern-growth: methodology and implications," ACM SIGKDD Explorations, vol. 2, pp. 14–20, 2000.
[18] A. Pietracaprina and D. Zandolin, "Mining frequent itemsets using Patricia Tries," in Proc. of IEEE ICDM Workshop Frequent Itemset Mining Implementations (FIMI), vol. 80, 2003.
[19] J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules," in Proc. ACM-SIGMOD, Int. Conf. Management of Data (SIGMOD), San Jose, CA, pp. 175–186, May 1995.
[20] J. Soo, M. S. Chen, and P. S. Yu, "Using a hash-based method with transaction trimming and database scan reduction for mining association rules," IEEE Transactions on Knowledge and Data Engineering, vol. 5, pp. 813–825, 1997.
[21] L. Zhi-Chao, H. Pi-Lian and M. Lei, "A high efficient AprioriTid algorithm for mining association rule," in Proc. of Intl. Conf. of Machine Learning and Cybernetics, pp. 1812 – 1815, 2005.
[22] A. Savasere, E. Omiecinski and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," in Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, pp. 432–443, 1995.
[23] E. Duneja and A. K. Sachan, "A Survey on Frequent Itemset Mining with Association Rules", Intl. Jrnl. of Computer Applications, ISSN:0975 – 8887, Vol. 46, Iss.23, pp. 18-24, 2012.

Dipti P. Rana is Assistant Professor at Computer Engineering Department, S. V. National Institute of Technology, Surat, Gujarat-395007, India. She obtained her M. Tech.(R) degrees from S. V. National Institute of Technology with specializations in Computer and is currently pursuing her PhD degree. Her research interest is in the field of security in web applications, computer architecture, database management system, data mining and web data mining. She is a life member of ISTE and CSI.

Naresh J. Mistry is Professor at Civil Engineering Department, S. V. National Institute of Technology, Surat, Gujarat-395007, India. His research interests are water and wastewater treatment, solid waste management, environmental impact assessment and environmental audit. He is a member of CES.

Mukesh M. Raghuwanshi is working as a principal at Rajiv Gandhi College of Engineering and Research, Nagpur, India. He completed his PhD in Computer Science, 2007, at Visvesvaraya National Institute of Technology (VNIT), Nagpur, India and M.Tech. in Computer Science & DP, 1991, at Indian Institute of Technology (IIT), Kharagpur, India. His research interests are evolutionary computing, genetic algorithm, data structures, algorithm, compilers, programming Languages, data mining and data warehouse, web crawling, text summarizing, image processing. He is having reputed 16 journal publications and 26 conferences publications. He is a member of IEEE, ISTE and CSI.