MATERIALIZED VIEW GENERATION USING APRIORI ALGORITHM

Debabrata Datta[1] and Kashi Nath Dey[2]

[1]Department of Computer Science, St. Xavier's College, Kolkata, India
[2]Department of Computer Science and Engineering, University of Calcutta, Kolkata, India

ABSTRACT

Data analysis is an important issue in business world in many respects. Different business organizations have data scientists, knowledge workers to analyze the business patterns and the customer behavior. Scrutinizing the past data to predict the future result has many aspects and understanding the nature of the query is one of them. Business analysts try to do this from a big data set which may be stored in the form of data warehouse. In this context, analysis of historical data has become a subject of interest. Regarding this, different techniques are being developed to study the pattern of customer behavior. Materialized view is a database object which can be extensively used in data analysis. Different approaches are there to generate optimum materialized view. This paper proposes an algorithm which generates a materialized view by considering the frequencies of the attributes taken from a database with the help of Apriori algorithm.

KEYWORDS

Data Warehouse, OLAP, Materialized View, Apriori Algorithm, Minimum Support Value

1. INTRODUCTION

Business enterprises deal with a large amount of data and their profits significantly depend on how the data are actually interpreted. So, data analysis has become an important topic of research now-days and has a huge potential, especially in the e-commerce sector. Moreover, it has a notable contribution in the field of social media as well. In this regard, data analysts and data scientists are in the process of developing different algorithms to analyze data and store the data that are of more importance. So, data analysis operation is executed to increase the business intelligence of a commercial organization. From the different approaches that are prevalent today, materialized view can be substantially used to store the important data. A materialized view is used to store the outputs of the queries. But unlike a logical view, this can store the outputs permanently in a physical memory. Because of this nature, this database object can be extensively used to store the results of the queries which are frequently asked for. So, instead of fetching data each time from the database itself, with the help of a materialized view, results can be directly obtained. This type of view can be used as a cache which can be quickly accessed. It will effectively reduce the network load, if the data are stored in distributed environment and at the same time, it will reduce the query execution time. But the problem remains that from a huge set of data transactions, which data are to be materialized. Different algorithms have been proposed to identify the optimal data set for materialization and the most of these algorithms are mainly based on greedy approach of selection. Of late, genetic algorithm has also been used to select data for materialization. The research work that is presented in this paper is based on Apriori algorithm proposed in [4]. This algorithm has been used to design a method to identify the data to be materialized based on their frequencies and the dependencies on other data. The next section gives an overview of some useful algorithms discussed in different research papers in connection

with materialized view selection. Section 3 gives an overall idea about the steps of Apriori algorithm that is applied in the process of development of the present work. Section 4 describes the steps followed in this research work and also puts forward the algorithm for selection of materialized view. The next section shows the results obtained after applying the proposed algorithm on different data sets and these results are analyzed. Finally, the last section focuses on the concluding points.

2. RELATED WORK

A data warehouse is a collection of historical data gathered from previously used data from a number of queries executed on a specific database. The data stored in a data warehouse is actually used for On Line Analytical Processing or OLAP which is a method for decision support system.

A materialized view is normally created on the data available in a data warehouse. Since a data warehouse contains a huge amount of data, extracting a specific set of data often becomes time consuming and thus may lead to an inefficient processing. A materialized view has its role exactly in this case. This type of view is a database object stores data physically for minimizing data processing time. Since the materialized views are essentially used with data warehouses only, the usefulness of constructing a data warehouse is also a point of concern. A detailed study on this aspect was done in [19] to explain the role of OLAP along with data warehouse. Further, different research has been going on to extract the optimal data set to be used for materialization. Earlier research work, as described in [7] had shown that the optimal materialized view selection was an NP-complete problem and the same research work had also proposed a greedy algorithmic based approach for view materialization to optimize query evaluation cost. The approach shown in [7] was dependent on a data structure called data cube. Another data structure, tree was used in view materialization in another research work that was proposed in [8]. The work, as discussed in [8] took a decisive parameter for view generation using tree and that parameter was the overall workload for query execution. Since the nature of the query may change from time to time, more data may have to be added with the existing data set available with the materialized view. So, a materialized view has to be scalable. In this issue, an approach had been discussed in [5] for OLAP processing. Another suck work was also described in [9] and the main characteristic of that work was to deal with a portion of the queries, instead of considering the entire query. Use of materialized view can also be extended into knowledge discovery of data, i.e., related to data mining applications. Quite a few researches have been done in this field. Using data clustering techniques, view materialization for data mining was proposed in [1] and the method shown in [1] could generate effective results. If the data are continuously processes, if the data are streamed data then also materialized view can be formed in a dynamic way. One such method was proposed and discussed in [6]. Use of dynamic programming model was seen in the same domain and as described in [10], this model can be effectively used for view materialization. As the first commercial database package, Oracle databases have used the materialized view with a large volume of data and this was discussed in [11]. Different research papers have done comparative studies on different approaches for view selection. One such review study was done in [15] and it was shown that a greedy algorithmic based approach with a polynomial time complexity would have been an optimal way for view selection for materialization. Based on the greedy algorithmic approach, a cost model was developed in [18]. In that work, different calculations were made on evaluation of the total cost and the benefits involved in each materialized view selection and based on the outcome, the most optimized materialized view was selected for a data warehouse.

Along with the selection of views to be materialized, maintenance of the same is also very important and a subject matter of research. One such research work was done in [17]. In that research work, common sub expressions were used for selecting and maintaining materialized view and that work described about three different kinds of materialization – transient, permanent

and incremental which were very much inter-dependent. That paper had mainly focused on the maintenance of the materialized view generated optimally. A research was done in [20] where the features of on dynamic materialized view were used in designing a special type progressive query, which is a set of step-queries, known as monotonic linear progressive query. Of late, modern approaches like evolutionary algorithms are used in view selection process. One of the initial papers regarding this is [14] where an evolutionary approach was proposed to find out the optimal set of views based on the total view election cost. The paper also discussed the proposed method in details. A genetic algorithmic based method was proposed and discussed in [3] where the views were represented in the form of chromosomes where each gene had represented a selected view. The selection of views to be incorporated within a chromosome from a population of chromosomes was done on the basis of a fitness function. The main parameter that was considered for the formation of the fitness function was the size of each view in the question.

Different steps like crossover and mutation were used to reorganize the chromosomes. The same paper also showed with some graphical representations that the approach of generating materialized view using genetic algorithm would have generated more optimal materialized view compared to earlier greedy based approaches.

3. A BRIEF OVERVIEW OF APRIORI ALGORITHM

Apriori algorithm was proposed in [4] to find out the frequent item sets from a large data set. This algorithm uses the association rules by identifying the relations among items that are involved in large data sets. The association rule is briefly described below:

Let $I = \{I_1, I_2, I_3, \ldots, I_n\}$ be a set of n number of items and let $T = \{T_1, T_2, T_3, \ldots, T_k\}$ be a set of k number of transactions where each $T_i \subseteq I$. With respect to the above defined sets, an association rule is said to be an expression of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$ with the condition that $A \cap B = \phi$. This association rule is defined by two parameters, viz., support and confidence which are defined through the following expressions: $support(A \Rightarrow B) = P(A \cup B)$ and $confidence(A \Rightarrow B) = P(B|A)$.

In other words, the parameter support identifies the percentage of transactions where both A and B occur and the parameter confidence identifies the percentage of transactions containing A that also contain B.

With all these parameters defined, Apriori algorithm identifies the frequent item sets. An item is said to be frequent if it crosses a pre-defined limit, defined as the minimum support value. This process involves multiple checking through iterations on the given large data set. The details of the process are described in [4]. The entire method is divided into two basic steps: join step and prune step. The first step, i.e., the join step generates k^{th} candidate item set from $(k-1)^{st}$ item sets after joining them. Each k^{th} candidate contains 'k' number of items considered for the final selection. This selection is based on a pre-defined parameter known as the minimum support. So, the first step finds out a larger item set from a smaller one. The next step, i.e., the prune step removes irrelevant item sets, if any. Irrelevance is identified by some predefined conditions imposed on the item sets depending upon the applicability of the considered data set. The pseudo-codes for these two steps are given and explained in [4].

4. PROPOSED WORK

From a given set of database transactions, the attributes, which are frequently accessed, can be identified by Apriori Algorithm. Each transaction is basically the execution of a query and each query deals with a set of attributes. So, each transaction can be thought of as a set of attributes on

which the query is to be executed. All of these sets of attributes are considered for Apriori Algorithm. Since the Apriori algorithm can be effectively used to find out the frequent itemsets, the present research work has considered this algorithm for finding out the frequent attributes that can be considered for materialization. So, for the sake of the research work, the attributes involved in the transactions have been considered to be analogous with the itemsets that were considered in the original description of Apriori algorithm in [4].

The output of the algorithm will be the sets of attributes containing the most frequently attributes that are asked for. There may be three different cases:

Case 1: The algorithm may generate more than one set of attributes.
Case 2: The algorithm may generate a single set of attributes.
Case 3: The algorithm may generate a null set.

In the first case, the intersection of the output sets will be considered for materialization. As far as the second case is considered, the output set itself is considered for materialization. In the final case, the output of the last but one iteration will be considered for materialization.

The number of iterations depends on a pre-defined threshold minimum support value. This threshold value is application specific and should be assigned by the business analysts depending on the nature of the business operation and the nature of the desired output.

Some other attributes may need to be attached to this set for materialization and that is to be identified next. This is done by finding out the confidence value of the attributes which are not selected initially for materialization on the attributes which are selected initially for materialization.

For example, if a transaction has five attributes A_1 to A_5 and only A_1 and A_3 are selected for materialization after applying the first phase then in the second phase, the confidence values of A_2, A_4 and A_5 on A_1 and A_3 are identified and if any confidence value is above the pre-defined threshold confidence value, which works like the minimum support value as described in [4], then the attributes corresponding to these confidence values are added with the materialized view.
The present method, which is named as Materialized View Generation using Apriori Algorithm or MVG_AA is based on the above-mentioned two steps. In the next section, two different test cases have been considered after applying the method MVG_AA. The data sets that have been considered for explaining the algorithm have been generated randomly.

The following is the pseudo-code for MVG_AA:

```
Algorithm MVG_AA ( )
{
        Input: T = A set of 'n' number of database transactions and ATR = A set of attributes on
        which different transactions are to be executed
        Output: M = A set of attributes to be materialized
        Let T = {T1, T2, T3, …, Tn}
        Initialize M by Φ, i.e., null set
        for i =1 to n
        do
                Let A = A set of attributes involved in ith transaction Ti
                R = Apriori (A) /* R is a set which stores the output generated by the Apriori
        algorithm and Apriori ( ) is a method to invoke Apriori algorithm and its takes A as its
        parameter */
                M = M ∪ R
        done
```

```
        S = ATR – R /* S is the set of attributes not selected by Apriori algorithm for
materialization */
        R´ = Check_Confidence (S) /* Check_Confidence ( ) is a method which looks for the
        confidence values of the attributes in S and returns the attributes which satisfy the
minimum        confidence threshold value and this is stored in another set R´ */
        M = M ∪ R´
        return M
}
```

The above pseudo-code is applied for different transactions which are randomly generated and a transaction may involve any number of attributes. Whatever be the transaction, it'll be able to identify the most important attributes to be considered for view materialization based on the frequency parameter of the attributes.

5. RESULTS AND THEIR ANALYSIS

In the context of the present research paper, the method MVG_AA ( ), as described in the previous section has been applied on two different test cases of transactions which have been randomly generated. It is further considered that the database on which the transactions are been executed has six attributes starting from A_1 to A_6. The following is the description of the results obtained.

Test case 1:

Table 1 stores all the transaction details from an example transaction along with their binary values and decimal values. In this table, under 'Transaction ID' column, each transaction is identified by a unique positive integer, under 'Attribute Set Involved' column, only the numbers of the attribute, used in the transaction, are mentioned. So, if a transaction has attributes A_1, A_2, A_3 and A_4 then its corresponding entry will be 1, 2, 3 and 4. The third column, i.e., the column with 'Binary Value' heading, the binary values of all the attributes involved in the transactions is stored. This is done for the implementation purpose and the binary value is generated in the way explained below:

The attribute number identifies the position in a binary string and here, a '1' is stored and for other positions, i.e., the positions where attribute is not participating in that particular transaction, a 0' is stored.

Table 1. The attributes in all transactions along with their binary and decimal values.

Transaction	Attribute Set	Binary Value	Decimal Value
1	1,2,3,4	1111	15
2	1,2,3	111	7
3	2,3	110	6
4	3,4	1100	12
5	5,6	110000	48
6	4,5,6	111000	56
7	1,2,3,4	1111	15
8	4,5,6	111000	56
9	1,2,3	111	7

10	4,6	101000	40
11	2,6	100010	34
12	3,4,6	101100	44
13	3,4,6	101100	44
14	2,4,6	101010	42
15	2,4	1010	10
16	1,2,3,5	10111	23
17	1,2,3,5	10111	23
18	1,2,3,5	10111	23

For example, from the table 1, if the tuple corresponding to the transaction id 4 is considered, only attributes A_3 and A_4 are selected, so its equivalent binary entry will be 1100, where the leftmost 1s identify that two attributes A_4 and A_3 are selected in this transaction and the rightmost 0s signify that in this transaction, two more attributes A_1 and A_2 are missing, i.e., not participating. Finally, the fourth column, i.e., the column 'Decimal Value' stores the equivalent decimal values of the binary values that are stored in the third column. The next table, i.e., table 2, is split into two pages and it stores all the frequency values against iterations which are based on Apriori algorithm. According to this algorithm, as stated in (R. Agarwal & R. Srikant, 1994), a number of iterations required to find out the most frequent item sets. The number of iterations is dependent on how fast the resultant set after the join step is a null set. After the final iteration is over, the attribute sets which have frequencies over a threshold value are identified and are chosen to be the most frequent ones. In this experimental process, described in this paper, the threshold frequency value has been chosen to be 2.

Table 2. The outputs of Apriori Algorithm applied on the transaction set as shown in Table 1.

Iteration	**Frequent Attribute**	**Frequency**
1	1	7
1	2	11
1	4	11
1	8	10
1	16	6
1	32	8
2	3	7
2	5	7
2	9	2
2	17	3
2	6	8
2	10	4
2	18	3
2	34	2
2	12	5
2	20	3
2	36	2
2	24	2

2	40	6
2	48	3
3	7	7
3	11	2
3	19	3
3	13	2
3	21	3
3	14	2
3	22	3
3	44	2
3	56	2
4	15	2
4	23	3

From the table 2, it is clear that the required frequent attribute sets are 15 and 23, i.e., 1111_2 and 10111_2 respectively because these two sets have frequency, which is termed as the support value according to (R. Agarwal & R. Srikant, 1994), above the threshold of 2. In other words, attributes A_1, A_2, A_3, A_4 and A_1, A_2, A_3 and A_5 are identified separately. Since two different sets of attributes are selected, their intersection is found out and it generates A_1, A_2 and A_3, which has an equivalent decimal value of 7. So, according to the algorithm MVG_AA ( ), the attributes A_1, A_2 and A_3 are to be materialized.

Table 3. The list of confidence values obtained from the result as shown in Table 2.

Confidence on 15	**Confidence on 23**
1=>14 = 0.2857142857142857	1=>22 = 0.42857142857142855
2=>13 = 0.18181818181818182	2=>21 = 0.2727272727272727
3=>12 = 0.2857142857142857	3=>20 = 0.42857142857142855
4=>11 = 0.18181818181818182	4=>19 = 0.2727272727272727
5=>10 = 0.2857142857142857	5=>18 = 0.42857142857142855
6=>9 = 0.25	6=>17 = 0.375
7=>8 = 0.2857142857142857	**7=>16 = 0.42857142857142855**
8=>7 = 0.2	16=>7 = 0.5
9=>6 = 1.0	17=>6 = 1.0
10=>5 = 0.5	18=>5 = 1.0
11=>4 = 1.0	19=>4 = 1.0
12=>3 = 0.4	20=>3 = 1.0
13=>2 = 1.0	21=>2 = 1.0
14=>1 = 1.0	22=>1 = 1.0

As the next step, the process will try to identify whether any other attribute is there to be materialized along with the attributes already chosen. For this, the confidence values, as defined in the association rules and stated in the previous section, of other attributes on the already selected attributes are to be calculated. The calculation is done by the standard expression as given in (Han, J. & Kamber, M., 2006). Accordingly, the confidence values are calculated and these values are shown in the next table, i.e., table 3. The confidence threshold value that has been considered for calculation is 0.5. According to the proposed method, if any other attribute has the confidence value greater than or equal to the threshold confidence value then that attribute

would be considered along with the already selected list of attributes. Table 3 contains two columns as in the previous step, two sets of attributes, having decimals values 15 and 23 respectively have satisfied the minimum support value.

From the entries as shown in Table 3, the attribute or the attribute set that is dependent on 7, i.e., A_1, A_2 and A_3 is marked in bold. It is clear that there is no attribute or set of attributes whose confidence value on 7 is above the threshold value of 0.5. So, no more attribute will be added with the already obtained list of attributes to be materialized. So, the final content of the materialized view will be A_1, A_2 and A_3.

Test case 2:

Table 4 stores all the transaction details from another example transaction along with their binary values and decimal values in the same way the data were stored in table 1. The next table, i.e., table 5 stores all the frequency values against iterations which are based on Apriori algorithm.
The same threshold vale of frequency, i.e., a threshold frequency value of 2, has been chosen for this test as well. From the table 5, it is clear that frequent attribute sets are 15 and 57, i.e., 1111_2 and 111001_2 respectively because these two sets have frequency support values above the threshold of 2. In other words, attributes A_1, A_2, A_3, A_4 and A_1, A_4, A_5 and A_6 are identified separately. Since two different sets of attributes are selected, their intersection is found out and it generates A_1 and A_4 or 1001_2 which has an equivalent decimal value of 9. So, A_1 and A_4 are to be materialized.

To find out the other attributes, if any, on the basis of the confidence values, the same confidence threshold of 0.5 has been chosen for this test as well.

From the entries as shown in table 6, the attribute or the attribute set that is dependent on 9, i.e., A_1 and A_4 is marked as bold. It is clear that only 6 (or 110_2), i.e., the set, containing A_2 and A_3 is dependent on 9 because it has a confidence value above the threshold value of 0.5. So, these two attributes will be added with the already obtained list of attributes to be materialized. So, the final content of the materialized view will be A_1, A_2, A_3 and A_4.

In this way, different attributes can be identified to be added in the final materialized view. The number of attributes and the attributes themselves may vary mainly if the confidence level and the support value are altered. These two parameters exclusive depend on the requirement of the applications for which the data analysis is to be performed.

Table 4. The attributes in all transactions along with their binary and decimal values.

Transaction	Attribute Set Involved	Binary Value	Decimal Value
1	1,2,3,4	1111	15
2	1,2,3	111	7
3	2,3	110	6
4	3,4	1100	12
5	5,6	110000	48
6	4,5,6	111000	56
7	1,2,3,4	1111	15
8	4,5,6	111000	56
9	1,2,3	111	7
10	4,6	101000	40

11	2,6	100010	34
12	3,4,6	101100	44
13	3,4,6	101100	44
14	2,4,6	101010	42
15	2,4	1010	10
16	1,4,5,6	111001	57
17	1,4,5,6	111001	57
18	1,4,5,6	111001	57

Table 5. The outputs of Apriori Algorithm applied on the transaction set as shown in Table 4.

Iteration	**Frequent Attribute**	**Frequency**
1	1	7
1	2	8
1	4	8
1	8	13
1	16	6
1	32	11
2	3	4
2	5	4
2	9	5
2	17	3
2	33	3
2	6	5
2	10	4
2	34	2
2	12	5
2	36	2
2	24	5
2	40	9
2	48	6
3	7	4
3	11	2
3	13	2
3	25	3
3	41	3
3	49	3
3	14	2
3	44	2
3	56	5
4	15	2
4	57	3

Table 6. The list of confidence values obtained from the result as shown in Table 5.

Confidence on 15	**Confidence on 57**
1=>14 = 0.2857142857142857	
2=>13 = 0.25	1=>56 = 0.42857142857142855
3=>12 = 0.5	8=>49 = 0.23076923076923078
4=>11 = 0.25	**9=>48 = 0.6**
5=>10 = 0.5	16=>41 = 0.5
6=>9 = 0.4	17=>40 = 1.0
7=>8 = 0.5	24=>33 = 0.6
8=>7 = 0.15384615384615385	25=>32 = 1.0
9=>6 = 0.4	
10=>5 = 0.5	
11=>4 = 1.0	
12=>3 = 0.4	
13=>2 = 1.0	
14=>1 = 1.0	

6. CONCLUSION AND FUTURE WORK

This research paper proposes a method to select the attributes to be considered for materialized views from a set of transactions. Since a transaction can be considered to be an outcome of a query and a query involves a set of attributes which are there in that particular query, it can be concluded that a transaction always deals with a set of attributes involved in the query. In this regard, the proposed research paper has tried to materialize attributes engaged in transactions. Since the proposed method is based on the outcome of Apriori algorithm, it works on the frequency aspect of the attributes present in the data transaction set. So, this work can further be expanded by including other parameters like time to generate a materialized view and the space to store a materialized view. The output obtained by this algorithm is based on a pre-defined set of transactions and hence the frequencies of occurrences of attributes in the transactions are also fixed. So, there is a possibility that the frequencies of the attributes may change in the future with a new set of transactions. This factor may also be included along with the present method to make the algorithm more scalable and dynamic.

REFERENCES

[1] Aouiche, K., Jouve, P. (2006). Clustering-based materialized view selection in data warehouses. In Proceedings of 10th East European conference on Advances in Databases and Information Systems (pp. 81 – 95).

[2] Dey, Kashi Nath, & Datta, Debabrata. (2013). Materialized View – A Novel Approach. In Proceedings of the 2nd International Conference on Computing and Systems (pp. 280 – 282).

[3] Vijay Kumar, T.V., & Kumar, S. (2012). Materialized View Selection using Genetic Algorithm.In Proceedings of the 5th International Conference on Communications and Information Science (pp. 225 – 237).

[4] Agarwal R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules.In Proceedings of the 20th International Conference on Very Large Data Bases (pp. 487 – 499).

[5] Thomas P. Nadeau, & Toby J. Teorey. (2002). Achieving Scalability in OLAP Materialized View Selection. In Proceedings of the 5th ACM International workshop on Data Warehousing and OLAP (pp. 28 - 34).

[6] Chandrasekaran, S., & Franklin, M.J. (2002). Streaming queries over Streaming Data. In Proceedings of the 28th International Conference on Very Large Data Bases (pp. 203 – 214).

[7] Harinarayan, V., Rajaraman, A. & J. Ullman. (1996). Implementing data cubes efficiently. In Proceedings of ACM SIGMOD International Conference on Management of Data (pp. 205 –216).

[8] Bhagat, A.P., & Harle, B.R. (2011). Materialized View Management in Peer to Peer Environment.In Proceedings of International Conference & Workshop on Emerging Trends in Technology (pp. 480 – 484). .
[9] Goldstein, J., & Larson, Per-Åke. (2001). Optimizing Queries Using Materialized Views: A Practical, Scalable Solution. In Proceedings of the ACM SIGMOD International Conference on Management of Data (pp. 331 – 342).
[10] Chaudhuri, S., Krishnamurthy, S., Potamianos, S. & Shim, K. (1995). Optimizing Queries with Materialized Views. In Proceedings of the International Conference on Data Engineering (pp. 190 – 200).
[11] Bello, R.G., Dias, K., Feenan, J., Finnerty, J., Norcott, W.D., Sun, H., Witkowski, A., & Ziauddin, M. (1998). Materialized Views in Oracle. In Proceedings of the 24th International Conference on Very Large Data Bases (pp. 659 – 664).
[12] Baralis E., Paraboschi S. & Teniente E. (1997). Materialized view selection in a multidimensional database. In Proceeding of the 23rd International Conference on Very Large Data Bases (pp. 156 – 165).
[13] Gupta, H. & Mumick, I. (2005). Selection of Views to Materialize in a Data Warehouse . IEEE Transactions on Knowledge and Data Engineering, 17(1) (pp. 24 – 43).
[14] Horng, J.T., Chang, Y.J., Liu, B.J. & Kao, C.Y. (1999). Materialized view selection using genetic algorithms in a data warehouse system. In Proceedings of the World Congress on Evolutionary Computation (pp. 2221 – 2227).
[15] Vijay Kumar, T.V., & Ghosal, A. (2009). Greedy Selection of Materialized Views. International Journal of Communication Technology, Volume 1, Number 1 (pp. 156 – 172).
[16] Zhang, C., Yao, X., & Yang, J. (2001). An evolutionary approach to materialized views selection in a data warehouse environment. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Volume 31, Number 3 (pp. 282 – 294).
[17] Mistry, H., Roy, P., Sudarshan, S., & Ramamritham, K. (2001). Materialized view selection and maintenance using multi-query optimization. In Proceedings of ACM SIGMOD International Conference on Management of Data (pp. 307 – 318).
[18] Chan, G. K.Y., Li, Q., & Feng, L. (1999). Design and selection of materialized views in a data warehousing environment: A case study. In Proceedings of 2nd ACM International Workshop on Data Warehousing and OLAP (pp. 42 – 47).
[19] Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. In ACM Sigmod Record, Volume 26, Issue 1 (pp. 65 – 74).
[20] Zhu, C., Zhu, Q., & Zuzarte, C. (2010). Efficient Processing of Monotonic Linear Progressive Queries via Dynamic Materialized Views. In Proceedings of Conference of Center for Advanced Studies on Collaborative Research (pp. 224 – 237).
[21] Han, J. & Kamber, M. (2006). Data Mining: Concepts and Techniques, Second Edition. Morgan Kauffman Publisher

AUTHORS

Debabrata Datta is presently an Assistant Professor of the Department of Computer Science, St. Xavier's College(Autonomous), Kolkata. He has a teaching experience of about more than 10 years. His research interests include data warehousing and data mining. He has published fifteen research papers in different national and international conferences as well as journals

Presently, Kashi Nath Dey is an Associate Professor in the Department of Computer Science & Engineering, University of Calcutta, India. He has about 8 years of industrial experience in different IT companies in India. He has about 26 years of experience in teaching and research. He has more than 35 research publications in different International conferences and International Journals and authored 7 books published by Pearson Education (India).