CORRECTED COPY

Materialized View Generation and Its Applications

Thesis submitted to the Degree of

Doctor of Philosophy (Science)

In Computer Science

By

Debabrata Datta

Post Graduate and Research Department of Computer Science
St. Xavier's College (Autonomous), Kolkata
Affiliated to The University of Calcutta

2025

Table of Contents

7 Conclusion

List of Figures

List of Tables

ABSTRACT

The modern era is entirely data driven. Different applications demand accessing big data for optimizing their performances. Text based data and image based data are two types based on which a substantial amount of researchers has been putting their efforts at least for the last two decades. The challenge of data processing and subsequent analysis become gruesome specifically in the domains which are query intensive in nature. Different end users have different types of requirements and the challenge lies in accessing those data in an optimal time. The present research work has mainly highlighted two aspects regarding this. The first aspect is about storing the frequently accessed data in such a way that they can be accessed in minimal time for satisfying customers' requirements. For this purpose, the usefulness of a database object called materialized view has been analyzed in the research work. But it is computationally complex to choose the sets of views to be materialized. Different methodologies have been proposed here regarding this. For the generation of materialized view which is used for physically storing the result set of a query or a series of queries as a separate database object, two principal algorithms have been explored and utilized and these are genetic algorithm and apriori algorithm. Two different techniques based on genetic algorithm have been proposed and subsequently comparative studies have been done with some existing methodologies. The effectiveness of Boolean Association rules has been investigated to optimally choose views to be materialized based on apriori algorithm. The second aspect of the research work is about utilization of materialized views in two domains – one in the field of text based data and another in the field of image classification. The former application has immense influence in the areas of incremental data mining and the later one is based on content based image retrieval. The efficacy of content based image retrieval for image classification has also been elaborately done as well in the research.

ACKNOWLEDGEMENT

With immense pleasure and gratitude, I would like to acknowledge each and everyone who has guided and helped me to complete the research work leading towards this dissertation.

Words cannot express my sincere and heartfelt gratitude to my PhD supervisor Dr. Anal Acharya and co-supervisor Prof. Kashi Nath Dey for their invaluable suggestions, guidance and patience. Both of them have endowed me with their knowledge and expertise since the inception of the research work. Without their support, this thesis would not have been possible. Their influence will definitely enlighten and encourage me in doing further research in future.

It is with much appreciation that I humbly thank Reverend Father Dr. Dominic Savio, SJ, Principal, St. Xavier's College (Autonomous), Kolkata for his continuous encouragement and help in pursuing my research work.

I would also like to take this opportunity to convey my earnest gratefulness to other Professors and Staff members of the Post Graduate and Research Department of Computer Science, St. Xavier's College (Autonomous), Kolkata and Computer Science and Engineering Department, University of Calcutta.

I am also happy to specially mention Dr. Samrat Roy, Coordinator, PhD Programs, St. Xavier's College (Autonomous), Kolkata, who has helped me throughout the tenure in speeding up the procedure right from the registration to thesis submission.

Finally, my sincere gratitude goes to my parents whose encouragements have been my blessings in pursuing my PhD work.

DECLARATION

Name of Supervisor: Dr. Anal Acharya

Designation: Associate Professor

Post Graduate and Research Department: Computer Science

Address: 47/51, Ramkrishna Ghosh Road, Kolkata – 700050.

CERTIFICATE

I, certify that the thesis entitled "**Materialized View Generation and Its Applications**" submitted by **Debabrata Datta** for the degree of Doctor of Philosophy (Ph.D.) in **Computer Science** in the area of **Data Mining** is the record of research work carried out by him during the period from **18.08.2022** to **20.02.2025** under my guidance and supervision, and that this work has not formed the basis for the award of any Degree, Diploma, Associateship, Fellowship, Titles in this University or any other University or other similar institution of Higher learning.

OVERALL SIMILARITY PERCENTAGE: 7%

Signature of Principal Supervisor: Anal Acharya 20/02/2025

Signature of Co-Supervisor: 24/02/2025

Signature of Ph.D Coordinator:

Associate Professor (Retired)
Computer Science & Engineering
University of Calcutta

DECLARATION

I do hereby declare that the work represented in this thesis entitled MATERIALIZED VIEW GENERATION AND ITS APPLICATIONS is the outcome of my own research work which has not been previously included in a thesis submitted to this or any other institution for any degree. The research work has been carried out under the supervisions of Dr. Anal Acharya of the Post Graduate and Research Department of Computer Science, St. Xavier's College (Autonomous), Kolkata and Prof. Kashi Nath Dey of the Department of Computer Science and Engineering, University of Calcutta.

DEBABRATA DATTA

Post Graduate and Research Department of Computer Science

St. Xavier's College (Autonomous), Kolkata.

LIST OF PAPERS CONTRIBUTING TO THE THESIS

1. Debabrata Datta, Anal Acharya, Kashi Nath Dey, “Content Based Image Retrieval through Materialized View”, ZKG International **(Web of Science Indexed)**, Volume VIII, Issue II, December, 2023, pp. 553 – 561, ISSN: 2366 – 1313.
2. Debabrata Datta, Kashi Nath Dey, "Application of Materialized View in Incremental Data Mining Operation", International Journal of Information Technology and Computer Science (**Indexed by IET Inspec, IndexCopernicus, EBSCO, CrossRef, Google Scholar, Microsoft Academic Research**), Volume 9, Number 6, pp. 43 – 49, June, 2017, ISSN: 2074 – 9007, DOI: 10.5815/ijitcs.2017.06.06.
3. Debabrata Datta, Kashi Nath Dey, “A Soft Computing Approach of Materialized View Creation”, Third International Conference on Business and Information Management held on 9^{th} to 11^{th} January, 2016 at National Institute of Technology, Durgapur; **adjudged as the Best Paper Award in Information Management 1 track**.
4. Debabrata Datta, Kashi Nath Dey, “Materialized View Generation using Apriori Algorithm”, International Journal of Database Management Systems (**Indexed by SCIRUS, EBSCO, Google Scholar**), Volume – 7, Number – 6, December, 2015, pp: 17 – 27, ISSN: 0975 – 5705.
5. Debabrata Datta, Kashi Nath Dey, Payel Saha, Sayan Mukhopadhyay, Sourav Kumar Mitra, “A Method of View Materialization using Genetic Algorithm”, IOSR Journal of Computer Engineering (**Indexed by J-Gate, CrossRef**), Volume – 18, Issue – 2, March – April, 2016, pp: 125 – 133, ISSN: 2278 – 0661.
6. Kashi Nath Dey, Debabrata Datta, “Materialized View – A Novel Approach”, IEEE International Conference on Computing and Systems – 2013 held on 21st to 22nd September, 2013 at The University of Burdwan, pp: 280 – 282, ISBN: 978 – 9 – 35 – 134273 – 1.

CHAPTER 1
INTRODUCTION

Commercial application of database has become diversified because of its manifold applications. Data scientists, database engineers and knowledge workers are constantly in the process of applying different features of a database and its allied topics like data warehouse in business intelligence. So, data analysis has become an important topic of research now-days and has a huge potential, especially in the e-commerce sector. The data scientists and the knowledge workers are constantly in the lookout for a better usefulness of the huge volume of data that a data warehouse or a large database may store. For this, the roles of different database objects have become significant. A view is such a database object. The present chapter has been organized as follows:

Section 1.1 gives an overview of a database object called Materialized View and its generation processes in the context of the present research work. Section 1.2 briefly discusses about some real life application areas of the materialized view in connection with the research work. The subsequent section highlights the main aspects of genetic algorithm. Next two sections respectively put forward the aspects related to incremental data mining and content based image retrieval. The final section concludes by raising different research questions addressed in this thesis.

1.1 MATERIALIZED VIEW

A materialized view is a database object used to store the output of a query or a set of queries as a separate physical entity [1]. This means even if the original tables, from which queries were generated, get removed from the database the materialized views remain in the database and effectively they can later be referred for a faster query execution [2]. These views are mainly used to physically store the frequently asked data, which are retrieved mainly from a data warehouse or any master database site which is a repository of historical data. Figure 1.1 roughly sketches the scenario.

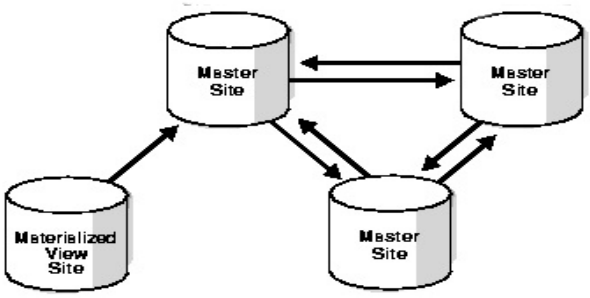

Figure 1.1: Communication between a materialized view site with other master sites [3]

Specifically for OLAP operations on a data warehouse, a materialized view can be used to store data in the form of aggregation from the base tables [4]. View materialization for aggregated outputs is more essential because queries involving an aggregated form of data are the costliest ones. So instead of referring to the base tables each time, a materialized view will do the needful without executing the queries. It may also be required to create index structures on the materialized view for accessing data in an improved way.

There are specific advantages of using materialized views in applications which are query intensive in nature. These are described next:

- Whenever a query, accessing the data gets executed, it scans through the relevant tables to fetch the necessary records. But data can be fetched directly from the materialized views instead of having to execute the queries every time. The more complex a query is, the more time will be saved using a materialized view.
- Materialized views provide a consolidated output from complex query logic. This makes data transformations and understanding easier for analysts. Hence, with materialized views, any complex query is more manageable.
- Materialized views also provide a specific data captured from the base tables. It may also be possible to configure read consistency in materialized views. With materialized views, data can even be made available in multi-user environments where concurrency control is essential.
- With materialized views, distribution of data across many geographical locations is possible by data replication. The people requiring the data may directly interact with the replicated data store which is closest to them geographically. Data replication and distribution with the help of materialized view often significantly reduce the network load.

- Materialized views may also provide time-stamped snapshots of datasets in a periodic interval of time. These help in decision support system by dynamically understanding the changing scenarios in a business platform.
- In distributed database systems, materialized views may be used to optimize queries involving data from remote servers. Data can be fetched and stored from a remote source in the form of materialized views. Later data may also be updated, if required. So rather than repeatedly fetching data from remote locations, they can be fetched and stored in a local materialized view. This reduces the network load and helps in optimally utilizing the network bandwidth. So this application directly helps in performance enhancement.
- Materialized views are also helpful for situations where periodic batch processing is required.

To be very specific, a materialized view can be very effective in the applications where the queries involve natural join operation. This is explained below:

In a database, a natural join operation deals with the referential integrity constraint on the selected relations. Let R1 and R2 be two relations having n1 and n2 number of tuples respectively. If R2 is dependent on R1 then R2 has a foreign key which relates R1 through referential integrity. Since R1 has n1 number of tuples then the primary key of R1 has n1 number of unique entries. A natural join is primarily based on Cartesian product operation which when applied on R1 and R2, will generate (n1 × n2) number of tuples. But after natural join, this number will reduce depending on the value equality of the foreign key and the corresponding primary key. After natural join of R1 and R2, let the number of generated tuples be [(n1 × n2) – n], where n is the number of times the referential integrity constraint does not match and n is a positive integer. Let J be the number of times the natural join between R1 and R2 has been executed from different users. So, the number of generated tuples from all these operation is J × [(n1 × n2) – n]. If the cost of processing natural join is C then the total cost is J × C. Considering the generation of all the tuples, this cost may reach a large value specifically when the datasets of the participating relations are huge. A materialized view plays its role exactly here. If a materialized view stores the joined output then the cost of overall operation is reduced by a factor of J. If there are N number of such natural join operations on a given database with Ci is the cost of performing the i^{th} join operation and Ji is the number of times the i^{th} natural join is executed then the overall reduction of cost, is materialized view is created, is Σ Ji , where $1 \leq i \leq N$.

Hence generation of materialized views can significantly reduce query execution time. For analyzing the effectiveness of materialized view, the following example is demonstrated.

Let us consider a portion of a normalized database that deals with the billing process of a manufacturing house and has three tables namely Bill, Bill_Item and Product_Costing. The sizes of the tables are calculated on the basis of the number of attributes and the number of bytes occupied by the attributes. The tables are shown below:

Attribute	Type	Size (in Bytes)
Bill_Id	Integer	4
Bill_Date	Date	8
Customer_Id	Integer	4
Status	Integer	4
Comment	Text	50

Table 1.1: Bill table in the database

So, the size of a single tuple for the above table is 70 bytes.

Attribute	Type	Size (in Bytes)
Bill_Id	Integer	4
Bill_Line_No	Integer	4
Product_Id	Integer	4
Description	Text	50
Ship_Date	Date	8
Quantity_Delivered	Integer	4
Unit_Price	Long	8
Sell_Price	Long	8

Table 1.2: Bill_Item table in the database

So, the size of a single tuple for the above table is 90 bytes.

Product_Costing		
Attribute	Type	Size (in Bytes)
Product_Id	Integer	4
Operation_Id	Integer	4
Employee_Id	Integer	4
Start_Date	Date	8
Finish_Date	Date	8
Elapsed_Time	Date	8
Rate	Long	8
Cost_Price	Long	8

Table 1.3: Product_Costing table in the database

So, the size of a single tuple for the above table is 52 bytes.

For considering the advantage of using materialized view, the following information for the above database is considered.

Average products manufactured per day = javg; average bills per day = iavg; average product_costing rows per product = ravg; day on record = d.

Since the profit analysis is a major operation that is asked for the management of the concerned organization, this analysis is based on creating materialized view which will store this profit analysis information. For a better and efficient profit analysis, the management keeps track of job profitability. This analysis requires cost, sell and profit for each job. To get this output, the required SQL statement is as follows -

Select p.product_id, sum(cost) “Cost Price”, sum(sell) “Sell Price”, sum(sell) – sum(cost) Profit from product_costing p, bill_item b where p.product_id = b.product_id group by p.product_id.

The above query will give product wise profit details.

Since there are j_{avg} manufactured products per day and d number of days on records, there are (j_{avg} × d) number of rows in the bill_item table. Considering b bytes per block of prefetch buffers for I/O operations, number of rows per prefetch buffer in case of Bill_Item table = floor((b bytes per block) / (90 bytes per row)) = floor(b/90) rows per buffer. So, the total number of buffers required for this table is ceil((javg × d)/floor(b/90)).

On the other hand, the profit_costing table contains ($j_{avg} \times d \times r_{avg}$) number of rows as there are r_{avg} number of rows per product. So, the number of rows per prefetch buffer = floor(b bytes per block) / (52 bytes per row)) = floor(b/52) rows per buffer. Hence, the number of buffers required to store all the rows of that table = ceil(($j_{avg} \times d \times r_{avg}$) / floor(b/52)).

Since the SQL requires to join two tables, sequential scans on both the tables may be performed. Now, assuming t unit of time as I/O time for b bytes of prefetch buffer, total join cost = T_{join} = [{ceil(($j_{avg} \times d$)/floor(b/90)) + ceil(($j_{avg} \times d \times r_{avg}$) / floor(b/52))} × t] unit of time. This will generate a moderately large value of time for a standard size of organization. Instead, if the output of the given query is stored as a materialized view, the query generation time will be reduced drastically.

The output of the above SQL statement has only four attributes namely product_id, cost_price, sell_price and profit. This output is stored as a view and let the name of the view be Product_wise_Profit which is shown below:

Attribute	Type	Size (in Bytes)
Product_Id	Integer	4
Cost_Price	Long	8
Sell_Price	Long	8
Profit	Long	8

Table 1.4: Product_wise_Profit view

So, the size of a single tuple of the above view is 28 bytes.

Naturally, the number of rows in this view will be as same as that of Bill_Item table. Therefore, the number of rows per prefetch buffer = floor((b bytes per block) / (28 bytes per row)) = floor(b/28) rows per buffer. This value is less than floor(b/90) which was obtained earlier. Therefore, the number of buffers = ceil(($j_{avg} \times d$) / floor(b/28)) which is less than ceil(($j_{avg} \times d$)/floor(b/90)), the value obtained for Bill_Item table.

To create the materialized view, the output of the above SQL statement needs to be written to the disk and this will take [ceil(($j_{avg} \times d$) / floor(b/28)) × t] unit of time. Let this be denoted by T_{write}. So, the materialized view Product_wise_Profit creation cost = $T_{create} = T_{join} + T_{write}$. The above result is obtained for shared disks as it is assumed that different tables may be stored in different disks for storing constraints.

Now, if a dedicated disk is considered compared to a shared disk for creating the materialized view, as it may be created in business off-hours, the creation time may further be reduced. Then the join time will be T'_{join} = [{ceil((j_{avg} × d)/floor(b/90)) + ceil((j_{avg} × d × r_{avg}) / floor(b/52))} × t′] unit of time. Clearly, $T'_{join} < T_{join}$ as t′ < t. So, the new creation time $T'_{create} = T'_{join} + T'_{write}$, where T'_{write} is the required time for write operation for a dedicated disk.

Now, the query I/O time with Product_wise_Profit = T′ = [ceil((j_{avg} × d) / floor(b/28)) × t′] unit of time, considering dedicated disk of storage. Hence the number of accessed buffer is significantly reduced here, T′ < T.

If the frequency of the above query is q per day, the total disk I/O time before Product_wise_Profit = $T_{I/O}$ = query frequency × I/O time per query = (q × T_{join}) unit of time. Again, after Product-wise_Profit is created, the total disk I/O time = $T'_{I/O}$ = creation cost + query frequency × I/O time per query = (T'_{create} + q × T′) unit of time.

Now, T′ << T_{join} and T'_{create} is to be considered only once. So, the materialized view improves an overall I/O performance and query response time. As the value of q increases, this advantage becomes more evident.

In this regard, there is an inherent problem associated with the generation of materialized views. From a list of great number of outputs coming from different queries, identification of the required data for materialization is a major area of concern and hence is a subject of research. The present research work deals with two categories of algorithms pertaining to the selection of views to be materialized. One category of the algorithm is purely based on the features of genetic algorithm which is one of the heuristic search algorithms and is a part of the family of evolutionary algorithms. Another proposed algorithm for view materialization is based on a seminal algorithm called Apriori algorithms.

1.2 APPLICATIONS OF MATERIALIZED VIEW

There is a significant role of materialized view in applications associated with data mining. Creating a materialized view can significantly reduce the costs required for expensive or frequently executable queries and hence also reduce the costs involved in the mining tasks. But there is a necessity to subsequently upgrade the outputs incurred from the data mining operation. A process called incremental data mining can play a crucial role in this scenario. It is a technique which needs a periodical refresh operation on the outputs that have been generated by the knowledge discovery module of a Decision Support System associated with an

enterprise which needs to analyze the historical data. In this area of applications, a materialized view can be optimally used to effectively reduce I/O cost instead of running a costly data mining algorithm each time for pattern generation. The proposed method in the present document talks about the role of materialized view in incremental data mining operations.

Another application area which has been addressed in the thesis is related to the utilization of the outcomes stored in the materialized views in the domain of content based image retrieval. This image retrieval process depends on the significant and the defining contents of an image. Hence the image database does not necessarily store the entire image. So it subsequently reduces both the storage as well as the retrieval time to tackle the image classification problem.

1.3 GENETIC ALGORITHM

Genetic algorithm is a method based on the principles of natural selection dependent on the biological evolution [5]. This method can be used to solve both the constraint and the unconstraint optimization problems by repeatedly modifying a population of individual solutions. For a given problem, a genetic algorithm needs to encode the decision variables of the problem into finite-length strings of alphabets of certain cardinality. These strings, which are candidate solutions to the problem under consideration, are referred to as the chromosomes and the alphabets are referred to as the genes. Each gene is associated with a value known as the alleles. In contrast to the traditional optimization techniques, a method based on the genetic algorithm works with encoding of parameters, rather than the parameters themselves. To identify and evolve good solutions and to implement natural selection, there should be a measure for distinguishing good solutions from bad solutions. This measure could be an objective function that is a mathematical model or it can be a subjective function where the better solutions are manually chosen over the worse ones. The algorithm is driven by a measure called the fitness function which determines a candidate solution's relative fitness over others and this will subsequently be led towards the evolution of the most optimal solution or a set of solutions [6]. Another important concept used by the genetic method is the concept of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms [7]. A relatively smaller population size may lead to a premature convergence to the solution resulting in a poor quality solution.

On the other hand, an unnecessarily large population size may lead towards misuse of the valuable computational time. Once the given problem is encoded in the form of a chromosome and a proper fitness function has been chosen, evolving the solutions to the search problem need following essential steps:

Initialization: The initial population of candidate solutions is usually generated randomly across the search space. However, domain specific and application specific knowledge may have to be incorporated.

Evaluation: Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.

Selection: This step allocates more copies of the solutions with higher fitness values. It effectively implements the survival-of-the-fittest mechanism on the candidate solutions.

Recombination: Recombination is a set which combines parts of two or more parental solutions to create new and possibly better solutions known as the offspring. Practically, the offspring generated from recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner.

Mutation: Mutation is an essential step of genetic algorithm through which a local but random modification is imposed on the chromosomes. So it tries to modify the features of the chromosome in such a way that getting an optimum candidate solution becomes easier and smoother.

Replacement: The offspring population created by selection, recombination and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement and steady-state replacement methods are used in genetic algorithm based solutions.

The above steps are repeated until a terminating condition is met.

1.4 INCREMENTAL DATA MINING

The patterns or the outputs generated from a set of queries may be time specific and hence need to be periodically refreshed. Each update operation can potentially invalidate some of the previously obtained outputs

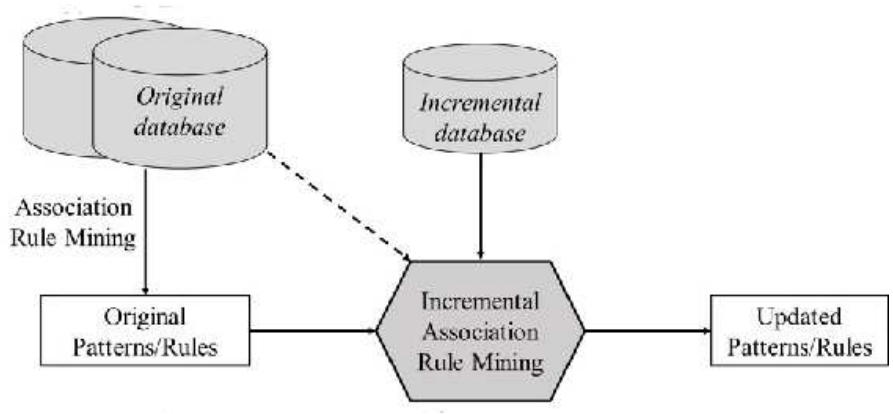

Figure 1.2: Process flow of incremental data mining [8]

So it is necessary to maintain an incremental approach for materializing views for online analytical processing and various data mining applications. One of the major problems in data mining is the processing time of the queries. It has been observed from different mining practices that the majority of data mining queries are only minor modifications of previous queries. Given these circumstances, data mining systems should try to exploit the results of previous queries, instead of running a complete mining algorithm for each query. Figure 1.2 depicts the process of incremental data mining. An incremental mining algorithm requires the results of previous queries to be available. One way to preserve those results is to use materialized data mining views. Materialized data mining views store the mined patterns and refresh them as the underlying data change. Incremental data mining technique is an approach to get the updated output obtained from a data mining operation.

1.5 CONTENT BASED IMAGE RETRIEVAL

Content-based image retrieval is a technique used to search images from large scale image databases by using their contents which may be both visual as well as semantic [9]. The visual contents may be general or domain specific. A general visual content may include color, shape, texture and spatial layout of the image whereas a domain specific visual content is application dependent and may involve domain knowledge. Semantic content is obtained either by textual annotation or by complex inference procedures based on the visual content. Considering the color of the image, some of the parameters that generally affect this process include the color space, the color moments, the color histogram the color coherence vector, the color correlogram and the invariant color features. On the other hand, if the shape of the image is considered then the parameters that affect the process include the moment invariants,

the turning angles, the Fourier descriptors, the circularity, the eccentricity and the major axis orientation. The content based image retrieval technique may also be used to index images. In this approach, after extracting the visual contents of the images they are described by multi-dimensional feature vectors. The feature vectors of the images in the database form a feature database. Once retrieval of an image is required, some example images are fed into the system which represents the images in the form of feature vectors. The similarities or distances between the feature vectors of the image under query and those of the images in the database are then calculated and subsequently retrieval is performed with the help of an indexing scheme effective for the system under consideration. Figure 1.3 demonstrates the process.

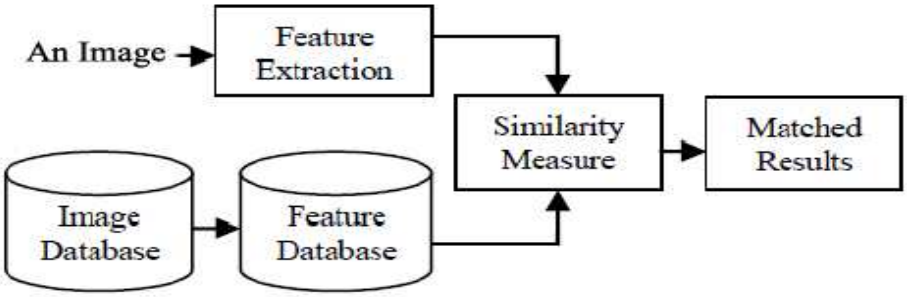

Figure 1.3: Process flow of content based image retrieval [9]

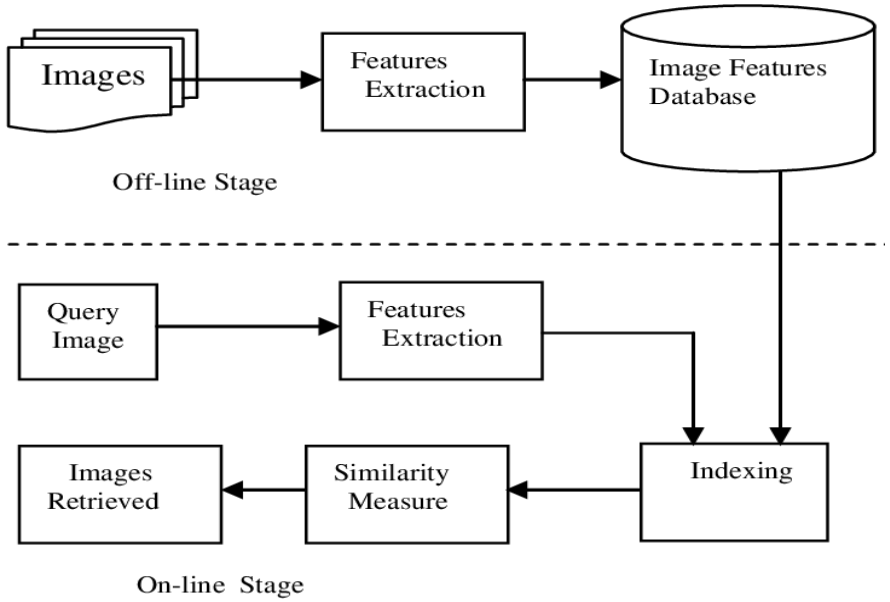

Figure 1.4: Different internal procedures under content based image retrieval [10]

As evident from figure 1.4, the detailed operation under content based image retrieval may broadly be divided into two stages – an offline stage which creates a database involving various aspects related to the features of an image and an online stage which tries to identify the similarity measure by comparing the features already stored in the feature database with the ones extracted from the new image under consideration.

1.6 RESEARCH OBJECTIVES

Storing and retrieving the required data set for analysis has become a major research objective specifically in the applications related to end users. Hence to summarize, the thesis aims to answer the following research questions:

1. How can the features of genetic algorithm be utilized to generate a materialized view?
2. How can the feature of frequent itemset generation of Apriori algorithm be utilized in the generation of a materialized view?
3. How can a materialized view be optimally used for an incremental data mining operation?
4. How can a materialized view be an effective way of classifying different images with the concept of content based image retrieval?

So the research work has two perspectives – one is to utilize different soft computing and seminal methodologies to build materialized views and another to apply the characteristics of a materialized view in different domains of data mining and analysis.

1.7 THESIS ORGANIZATION

The purpose of this section is to concisely state the way different chapters of the thesis is built upon the previously done researches on the same domain and hence tries to derive the answers put forward in section 1.6. Figure 1.5 gives a flow of the work described in the thesis. As mentioned in the previous sections of this chapter, the thesis revolves around the concept of materialized view creation strategies and its various applications in the possible domain. Hence one set of tasks gives focus on the few novel approaches of materialized view creation using genetic algorithm and Apriori algorithm. The other set of tasks describes some areas of applications where the benefits of materialized views can be utilized. One such application highlighted is related to incremental data mining and other application is in the field of content based image retrieval.

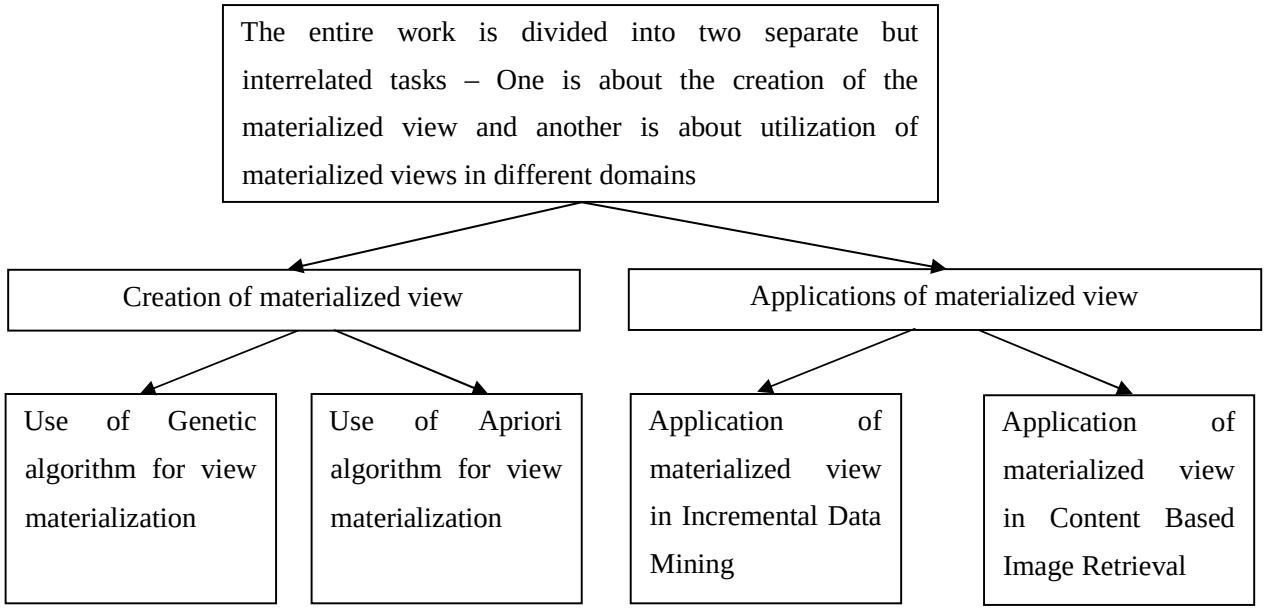

Figure 1.5: The flow of the work developed in the thesis

CHAPTER 2
RELATED LITERATURE SURVEY

The chapter is dedicated to the survey of literature leading towards different algorithms and applications proposed in the entire thesis. It is evident from the first chapter that the basis of the work proposed in the thesis is materialized view with its generation processes and hence most of the surveyed research papers have been on this topic and section 2.1 is dedicated on this. Section 2.2 highlights the contents described in the research papers dedicated to the various applications on the incremental data mining whereas section 2.3 gives the summary of various studies related to the content based image retrieval process.

2.1 MATERIALIZED VIEW AND ITS GENERATION ALGORITHMS

Since a materialized view is a database object which stores and maintains the outputs of the queries on the data warehouse, different algorithms have been developed for efficiently extracting the required views that run on historical data, to be materialized. The architectural issues concerning the efficient design of a materialized view to be a part of a data warehouse hugely depend on a problem known as the Materialized View Selection (MVS) problem, which is the problem of choosing an optimal set of views to materialize under resource constraints. The MVS problem has been shown to be NP-Complete in the data warehouse literature, considering data cubes for identifying materialized views [11]. Another method for view materialization using streamed data was proposed in a research work [12]. Another research work [13] had used a greedy algorithm and this proposal was based on some heuristics to design data cubes which would ultimately lead to an optimal materialized view selection. Considering all these ideas, an approach was described in a research work [14]. By utilizing a data mining technique called clustering, a framework for materialized view selection was also highlighted [2]. A number of other research works have been done for view materialization using greedy algorithms [15]. A tree based materialized view selection algorithm was proposed in a research work [16] keeping the overall workload for execution of queries in mind. Different research work had also exploited the potential of materialized view by designing a fast and scalable algorithm [17][18]. For query optimization using materialized views, a dynamic programming model was proposed in another research work [19][20].

Description of the use of materialized views in Oracle databases was found in [21][22]. The role of evolutionary algorithms, specifically genetic algorithmic for materialized view selection were proposed in many research papers [23][24][25][26][27][28][29]. These algorithms aim to select views for higher dimensional data sets with the key challenge of selecting views of high quality i.e. low total view evaluation cost (TVEC). Another research work has applied randomized algorithm for view materialization [30]. Along with view materialization, maintenance of the same is also required and accordingly, different methods were proposed in a number of research papers [31].

2.2 APPLICATION OF MATERIALIZED VIEW IN INCREMENTAL DATA MINING

Since the present research document discusses about two specific applications of materialized view – one in the domain of incremental data mining and content based image retrieval, researches that have been done in these areas are highlighted here. There have been very few researches done on the field of incremental data mining so far. A research work [32] gave a method to discover the association rule among the sets of data present in a large database. The idea of incremental data mining was at first discussed in a research work where a novel method called Fast Update Algorithm (FUP) was discussed [33]. This research was further expanded in another research work [34], where an algorithm was proposed to minimize the re-computation operation to find out the updated mined data from a transaction database when some new data sets are added or some existing data sets are removed. An approach of implementing an incremental data mining operation on the sequential patterns of data was discussed in a research work [35].

2.3 APPLICATION OF MATERIALIZED VIEW IN CONTENT BASED IMAGE RETRIEVAL

The methods in the field of content based image retrieval have been discussed in a very few research papers [36] and all of the research papers had discussed methods of performing content based image retrieval using color-based, shape-based and depth-based features of the images [37][38]. In this work also, the color-based approach, based on color moments algorithm [39] and shape-based approach, based on height and width ratio have been used to perform image retrieval and image classification.

In the same domain, to increase the accuracy as well as the efficiency of the method, a number of critical areas like feature extraction and selection was considered in a research work [40]. Another work on this domain dealt with data representation and similarity measure [41]. To query an image from an image database, clustering techniques were proved to be a better solution [42].

CHAPTER 3
MATERIALIZED VIEW CREATION USING GENETIC ALGORITHM

3.1 INTRODUCTION

Commercial applications of database have, of late, become diversified because of manifold applications. Data scientists, database engineers and knowledge workers are constantly in the process of applying different features of a database and its allied topics like data warehouse in business intelligence. Since huge amount of data is an important and integral part of the process, use of data warehouse automatically comes into the scenario. A data warehouse is a very large database system that collects, summarizes and stores data from multiple remote and heterogeneous information sources. Data warehouses may also be used for supporting decision making operation on an intelligent system. The decision making queries are complex in nature and take a large amount of time when they are run against a large data warehouse. To analyze the historical data present in the data warehouse, data mining is an effective way which can be used to study the pattern of the huge amount of data which affect the business scenario. For the purpose of analyzing the data, views play significant role. A view, which is a database object, can be used to logically store the output of a database query executed on the tables from a database. This database object can be used to identify the nature of the queries that the users run. A materialized view, which is another database object, is an extension of a view and is mainly used to physically store the frequently asked data, which are retrieved from a database through queries and even from a data warehouse. A materialized view can be used like a cache which can be rapidly accessed [2]. So the basic purpose of using a materialized view in business process model should be to minimize the query execution time. This will be of great importance specially if the nature of the query is very complex, i.e., if a query contains any join operation or any aggregate operation. At the same time, it is to be considered that a materialized view is very much time specific and hugely depends upon the historical dataset. This type of view should store the outputs of the frequently asked queries from a database or it should store the outputs of the queries of major importance. This is a point of concern to identify the queries to be considered for a materialized view from a set of queries. But the problem is that from a list of great number of outputs coming from different queries, identification of the required data for materialization. Different algorithms have been proposed in this regard.

All of these algorithms that have been proposed so far in connection with the selection of the proper data come from queries or stored in logical views, which are to be identified for materialization. From these algorithms, the greedy ones have been proved to be more successful in efficiently identifying the list of queries to be considered for the formation of materialized views. Genetic algorithm has also been used in this context. A method for materialized view selection using a genetic algorithmic approach has been proposed in [23], considering the candidate views as the genes in the chromosomes. The present research work, as described in this chapter, has been done on the basic ideas that was proposed in [23] with some additional constraints.

In this chapter, two different methods have been discussed for view materialization. Both the algorithms described are fundamentally dependent on an evolutionary class of algorithm called genetic algorithm which can be effectively used for solving optimization problems. Section 3.2 gives an overview of this algorithm that consists of processes mimicking the biological evolution and hence involves the steps like crossover and mutation. Any problem based on genetic algorithm repeats the population of individual solutions hoping to get a better outcome in the current instance compared to the previous one. The first method, described in section 3.3 primarily considers two factors for view materialization – the size of each view eligible to be materialized and the time to generate each view. On the other hand, the second approach, described in section 3.4 focuses on the frequencies of all the attributes present in the views eligible for materialization. A comparative study with existing ones has also been done considering few parameters..

3.2 AN OVERVIEW OF GENETIC ALGORITHM

Genetic algorithms (GAs) are adaptive heuristic search methods based on the evolutionary ideas of natural selection and genetics. They are inspired by Darwin's theory about evolution – "Survival of the fittest." They represent an intelligent exploitation of random search used to solve optimization problems. GAs, although randomized, exploit historical information to direct the search into the region of better performance within the search space. In solving problems, some solution(s) will be the better than others. The set of all possible solutions is called the search space or the state space. Each point in the search space represents one possible solution, which can be marked by its value. This is known as the fitness value for the problem. GA looks for the best solution among a number of possible solutions.

Figure 3.1 gives a block diagram related to the working principle of genetic algorithm.

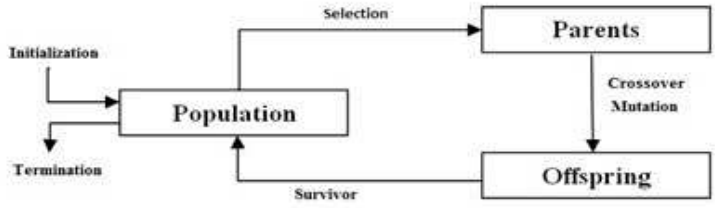

Figure 3.1: Working principle of Genetic Algorithm [43]

The basic steps of a genetic algorithm are stated below:

Initialization: *An initial population of all chromosomes is usually generated randomly within the search space.*

Evaluation: *Once the population is initialized or an offspring population is created, the fitness values of all individuals within that population are evaluated using a proper fitness function.*

Selection: *Selection of two parents with higher fitness values for recombination. The main idea is to prefer better solutions to worse ones.*

Crossover (Recombination)*: Recombination combines parts of two or more parental solutions (or individuals) to create new, possibly better solutions (offspring).*

Mutation: *Mutation basically alters one or more gene values in a chromosome to maintain genetic diversity from one generation of a population to the next.*

Replacement: *The offspring population created by selection, recombination and mutation replaces the original parental population.*

Repeat steps 2–6 until a terminating condition is met.

3.3 A SOFT COMPUTING APPROACH OF MATERIALIZED VIEW CREATION

A materialized view is dependent on the data that are fetched from the database using different forms of queries. To form a materialized view, both the space and the time should play important roles as each of the materialized views will take some time to generate depending on the complexity of the query and it will occupy some space in the disk depending on the attributes that are involved in the formation of the materialized view.

The algorithm that was proposed in [23] had considered the space constraint only in the form of the view size in connection with the views considered for selection in the chromosomes. The present work has additionally considered the time of view selection along with the consideration done in [23]. The proposed algorithm, that is named as View Materialization with Soft Computation (VMSC) in this chapter paper is based on a closed lattice where each node represents a view. This lattice is a representation of a data cube in the form of a graph data structure.

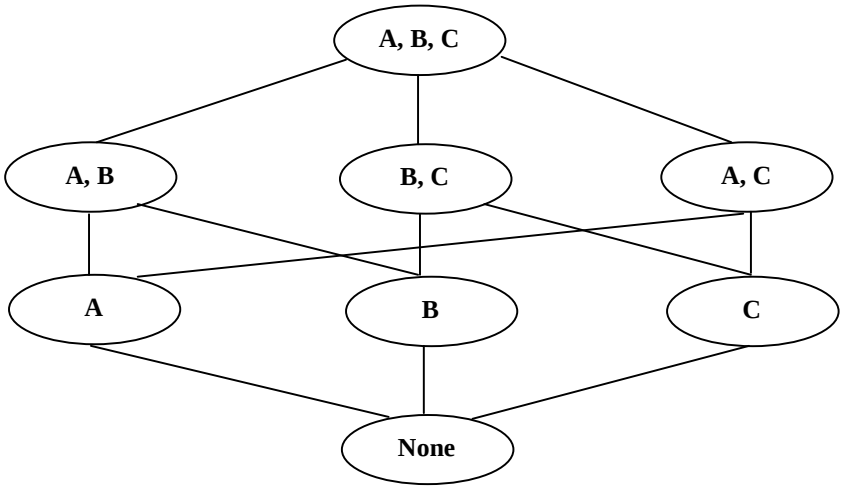

Figure 3.2: A sample lattice with three attributes A, B and C

Figure 3.2 depicts a lattice of three attributes A, B and C. Since there are three attributes, the lattice contains eight vertices, also known as the cuboids in this context. In general, if a lattice deals with n number of attributes, it will have 2^n number of cuboids. An attribute may also be referred as a dimension in explaining the formation of the lattice. In the context of the database structure and the subsequent formation of data warehouse from that database, a lattice contains the aggregation of attributes in the form nodes or vertices. The root node basically contains the fact table which is formed by taking all the key attributes from the corresponding dimension tables. So a fact table is a data structure from which the views are to be considered and the leaf node stores the information related to aggregation of all the attributes present. The results generated after applying the steps proposed in the present algorithm has been compared with the results that would have been obtained after applying the algorithm as stated in [23]. The number of views has been determined from the number of attributes that led to the formation of the relation or fact.

In accordance with the property of a lattice, the number of views to be considered for materialization has been taken as 2^{dmnsn}, where 'dmnsn' is the dimension or the number of attributes of the lattice or data cube to be considered.

In this algorithm, the following three parameters have been considered for defining each node in the lattice structure:

a) Index: Each node or view has been indexed for convenience in terms of reference.

b) Size: Each node or view has been assigned a randomly generated value as size which is within the range between 1 and 1000. The root node has been assigned with the value of 1000 while the leaf nodes have been assigned with the value of 1 by default. The value of size for each node reduces as nodes are visited from the root node to leaf nodes of the cube. This is simply because the root node itself contains all the attributes whereas the number of attributes decreases as the cube is traversed down from the root node. All the intermediate nodes have been assigned with the values depending on their positions in the lattice.

c) Time: Each node or view has been assigned a randomly generated value as time (in milliseconds) which is within the range from 0 to 1000. The root node has been assigned the time value of 0 while the leaf nodes have been assigned with the time value of 1000 by default. The value of time for each node increases as traversed down from root to leaf in the lattice. This is because root node contains all the attributes, so materialization process will take the least amount of time if the root node is considered and at the same time, since the number of attributes starts decreasing as the nodes are visited downward in the data cube, more time will take for materialization as the number of attributes already available in the node is less compared to the number of attributes already available in the lower level nodes in the lattice structure.

Each node in the lattice has been assigned another value, termed as Value_Node, based on the product of the absolute values of the size and the time.

So, Value_Node = |Size| × |Time|

The detailed argument in favor of considering the processing time to increase as traversing from the root node to the leaf node in the lattice has been put forward in the following points:

a) The processing time for each view increases as traversing from the root node to the leaf node is done in general scenarios as the initial information is there for the root node only. For this node, its child nodes can be traversed. In this way, reaching the leaf nodes will take the maximum time.

b) In considering the increasing format from top to bottom, the following three cases may be realized as test cases:

i) Value of a node may be same as its ancestor node in spite of having different time value.

Example – Let there be two nodes A and B where A is B's ancestor node with the parameters as specified below:

Size (A) = 60 unit, Size (B) = 40 unit, Time (A) = 0.4 unit and Time (B) = 0.6 unit.

So, Value (A) = 24 and Value (B) = 24.

ii) Value of a node may be less than its ancestor node.

Example – Let there be two nodes A and B where A is B's ancestor node with the parameters as specified below:

Size (A) = 60 unit, Size (B) = 40 unit, Time (A) = 0.4 unit and Time (B) = 0.5 unit.

So, Value (A) = 24 and Value (B) = 20.

iii) Value of a node may be greater than its ancestor node.

Example – Let there be two nodes A and B where A is B's ancestor node with the parameters as specified below:

Size (A) = 60 unit, Size (B) = 40 unit, Time (A) = 0.4 unit and Time (B) = 0.7 unit.

So, Value (A) = 24 and Value (B) = 28.

Hence it is realized that having time in increasing order helps the work with varying possibilities of views within the lattice. At the same time, with the consideration of time as an additional constraint, a more versatile comparative graph generation would be possible with respect to the previous algorithm as discussed in [23], where space was the only constraint for view selection.

With the development of the algorithm the following terms are related:

Fact Table – It is a database object which identifies the relation between the numbers of tables participating in the warehouse formation.

Number of dimension – It is the number of attributes in a fact table. These are nothing but the keys of the corresponding dimension tables based on which the fact table is formed.

Lattice – It is a set of interconnected nodes, each node identifies an aggregation of the attributes.

Lattice Size – It is the number of nodes present in a lattice. In this case it is 2dmnsn, where 'dmnsn' is the variable which identifies the dimension.

Lattice Time – It is the summation of the processing times of all the views, represented by the nodes in the lattice.

Chromosome – It is a data structure which is a collection of objects known as genes. In this context, views are analogous to genes. So, here, in the present work description, a chromosome is nothing but a collection of views.

Population – It is defined as the collection of chromosomes.

Fitness Function – It is the main function used for calculating the fitness value of each chromosome in a given population.

The procedure for calculating fitness function for each chromosome is as follows:

Step 1: *The views considered to form a chromosome are taken to be the materialized views for that chromosome only and the views that are not in consideration are identified as the non-materialized views. The root node is never considered for materialization as this node basically contains the whole table structure as it is expected that the materialized view should contain a proper subset of the entire table structure. This is also never considered as a non-materialized one, thus it is free from the random selection required for the calculation of the fitness function. The leaf nodes are never considered as materialized as each leaf node contains a single attribute from the fact table.*

Step 2: *For a single chromosome, the values of all the views considered as the materialized views are added up together. The added values of the materialized views are stored as a new parameter called Selected_Views_Value.*

Step 3: *Now the views that are not present in a given chromosome are to be considered. For this set of views, their values are not considered. Instead the values of their respective ancestor nodes in the lattice having the smallest value among all of its ancestors are considered. This approach was also used in [23]. Then their values are added up together and this result is stored as another parameter called Unselected_Views_Value.*

Step 4: *After the values against the different Selected_Views_Value and different Unselected_Views_Value are obtained, the fitness function for each chromosome is calculated using the following expression:*

Total_View_Selection_Cost = Selected_Views_Value + Unselected_Views_Value
This is the parameter used as the fitness function in the present work. In general, in any genetic algorithmic approach the more is the value of the fitness function, the better is the process and the more is its acceptability. But here, in the proposed work, since the fitness function is dependent on the values of the views to be considered for materialization, the less is the Total_View_Selection_Cost, the more optimal is the materialized view selection. So, if a chromosome has a less fitness value than another, the content of the former chromosomes should be selected for the materialization.

Step 5: *This is done for all the chromosomes present in a population. After calculating the fitness value for each chromosome in a given population, it is required to select the best chromosome out of a given population based on a repetitive computation. For obtaining such a chromosome, it is required to run the algorithm, as given below, for a given number of times, known as the generation, identified as the variable 'g' and for a given dimension, identified as the variable 'd'.*

The steps that have been described above are the main steps involved in the algorithm.

There are two more things that control two important phases of the algorithm. These are described below:

a) A control parameter has been used in the form of Crossover Probability which ranges between 0 and 1. In genetic algorithmic based applications, crossover is an important operation required to bring in random improvement in chromosomes. There are mainly two different strategies used for crossover operation. They are one-point crossover and m-point crossover, where m > 1. In the present research work, one point crossover method has been used because of its straightforward approach. The role of crossover probability is to determine the number of winner chromosomes that will take part in a crossover. The use of this parameter is explained with the help of the following example:

If there are 8 winner chromosomes and there is a crossover probability of 0.5 then the number of chromosomes participating in a single crossover is 4.

b) Another control parameter that has been used in the work is in the form of Mutation Probability. Like Crossover Probability, here the value ranges between 0 and 1.

Mutation is another important operation in a genetic algorithmic approach and it is required to exchange randomly chosen views from randomly chosen winner chromosomes in order to obtain a new set of chromosomes with a different fitness function. The role of mutation probability is to determine the number of views of winner chromosomes that will take part in mutation. The use of this parameter is explained with the help of the following example:

If there are 8 winner chromosomes each having 10 views and a mutation probability of 0.5 is considered then the number of random views to be mutated is $0.5 \times 8 \times 10 = 40$.

The algorithm to execute the overall process related to the present research work is depicted next:

Step 1: *The process is initiated.*

Step 2: *A population of size 'm' is generated where each chromosome is of size 'n'.*

Step 3: *A dimension 'dmnsn' is set and that identifies the number of attributes in the fact table.*

Step 4: *The number of nodes is found out and its value is 2dmnsn.*

Step 5: *A lattice having 2dmnsn number of nodes is generated.*

Step 6: *The size and the processing time of each of the nodes in the lattice are assigned with the methods described above.*

Step 7: *The total view selection cost of each chromosome based on a pre-defined fitness function is calculated. The role of the fitness function has already been discussed.*

Step 8: *The binary tournament selection on the selected chromosomes is performed. This method is used for introducing more randomization in the process.*

Step 9: *The winner chromosomes after the process executed in step 8 is selected.*

Step 10: *The crossover operation and the mutation operation on the selected chromosomes are performed, i.e., the winner chromosomes are based on some pre-assigned crossover probability and mutation probability.*

Step 11: *Steps 7 to 10 are repeated for a given generation 'g'.*

Step 12: *The outcome is the finally selected views to be materialized.*

Step 13: *The process is terminated.*

After all the above-mentioned steps are executed, the finally selected chromosome contains the views to be materialized.

3.3.1 RESULTS OBTAINED AND ANALYSIS

Dimension of the lattice	Fitness values on different sets		Minimum Fitness Value	
	Algorithm as discussed in [23]	VMSC (after normalization)	Algorithm as discussed in [23]	VMSC
4	9834	311.086	**9694**	**276.519**
	10820	276.519		
	10170	315.284		
	9694	304.095		
5	15715	367.229	**14520**	**353.003**
	14608	353.003		
	14520	599.877		
	15733	714.608		
6	42908	2244.698	**39363**	**1296.19**
	39532	1296.190		
	39363	1343.929		
	40114	1914.246		
7	62306	1869.262	**62306**	**1869.262**
	84373	2453.765		
	78161	1882.012		
	66592	2580.362		
8	169703	6391.550	**149996**	**2774.217**
	149996	4308.359		
	172558	7170.775		
	165614	2774.217		
9	380139	9075.732	**350811**	**5703.695**
	362494	8908.453		
	370860	5703.695		
	350811	7052.475		
10	849385	7727.303	**672581**	**7727.203**
	672581	27847.944		
	784704	21050.733		
	701525	17663.188		

Table 3.1: Results obtained after applying the proposed algorithm and the algorithm depicted in [23]

The algorithm, named as VMSC in the present paper and discussed in section 3 has been implemented with the values assigned against each of the views of the lattice structures considered. The values against all of these views have been assigned based on the logic already explained in the previous section. After applying the steps mentioned in the previous section, a result has been obtained and this is discussed in this section. For the implementation purpose, it has been considered that there is a population size of 8 chromosomes with each chromosome having 10 views. The mutation probability that has been considered is 0.025 and the crossover probability has been chosen to be 0.5. The algorithm was implemented using Java program on a system with the following configuration:

Processor Configuration: Intel i5 2500k with 3.0 GHz

Primary Memory Capacity: 4 GB

Hard Disk Space: 1 TB

The previous table, i.e., table 3.1 depicts the outputs obtained from the present work and the outputs obtained with the same set of inputs applied on the algorithm as discussed in [23]. The table also shows a comparative study between the fitness values of the winners of the proposed algorithm VMSC and that of the algorithm as discussed in [23] from each dimension after running for 1000 generations. The experiment has been conducted for 8 chromosomes, so after applying binary tournament selection method, for each dimension, there will be 4 winner chromosomes and hence the second column and the third column of the following table store four different values for four different winners after applying the algorithm discussed in [23] and the VMSC algorithm respectively. For each dimension, 1000 generations are considered, i.e., the value of g that was mentioned in Step 11 of the aforesaid algorithm discussed in the previous section, is 1000. The work has been done from dimension 4 to dimension 10 and has been applied on both the algorithms. From the last two columns under ‘Minimum Fitness Value’ of Table 3.1, it is clear that always the present algorithm has given substantially better values against the fitness functions. So for the optimization process, VMSC will give a better result compared to the algorithm as discussed in [23].

In this research work for the implementation purpose, lattice structures from dimensions 4 to 10 have been considered. Further, in the working environment, the time of each node is considered in milliseconds and so the fitness values in this algorithm has been divided by 1000 to get the normalized result in seconds. This has been shown in the above table as well against the values obtained after applying the present algorithm VMSC.

From the data as demonstrated in the table 3.1, it is clear that in comparison to the algorithm as proposed in [23], the present algorithm has a better performance. So, it can be concluded that the overall optimization in the process of materialized view generation has been satisfactorily improved with the introduction of processing time as an additional parameter along with the view space that was already considered in the earlier algorithm.

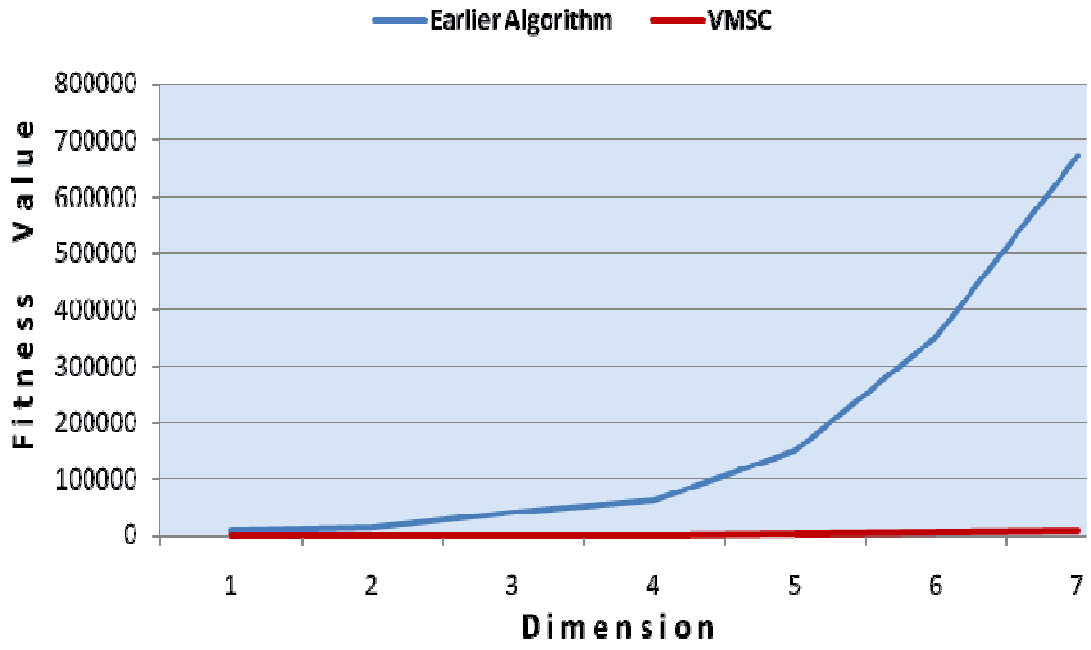

Figure 3.3: A graphical comparison of the proposed algorithm and the algorithm depicted in [23]

Both figures 3.3 and 3.4 show graphical comparisons between the present work and the work that was depicted in [23], which was also an approach based on the genetic algorithm. In both of these graphs, the result obtained in the present work is shown in the red line and the results obtained in the previous work as discussed in [23] is shown in blue line. From the first graph, it is clearly seen that the rate of increase of the fitness values in the present work is negligible in comparison with that of the previous algorithm. So, fitness values to find the optimum chromosomes based on which the materialized views are to be constructed, do not significantly depend on the dimensions of the lattice structure. The next graph shows a dimension-wise comparison of the fitness values of the algorithms. Again, it has been shown that, irrespective of the dimensions of the lattice structures considered, the fitness values of the present work for all the dimensions are much smaller than those of the previous algorithm.

So, it is clear that considering the time as a parameter, the performance becomes substantially better.

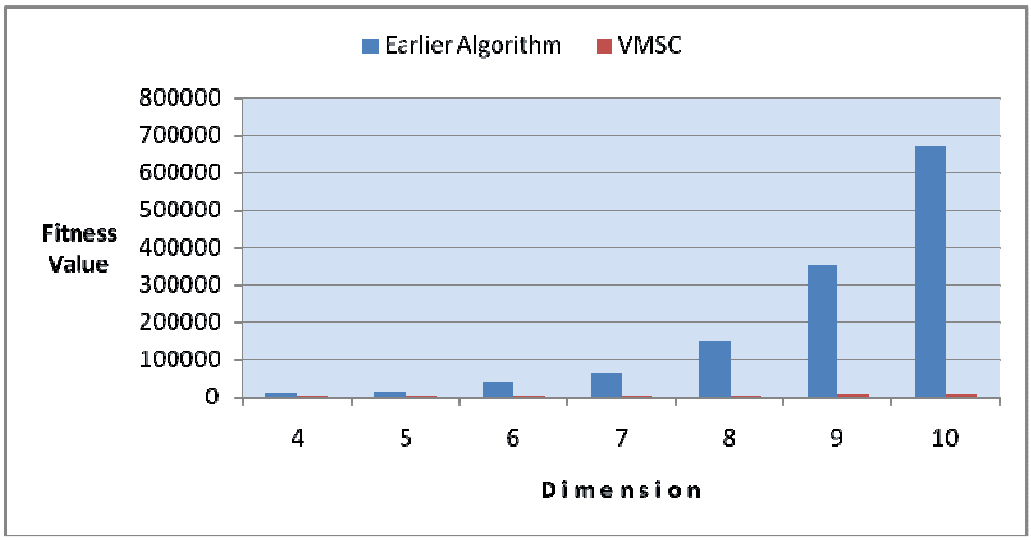

Figure 3.4: Dimension-wise comparisons of the proposed algorithm and the algorithm depicted in [23]

3.4 A METHOD OF VIEW MATERIALIZATION USING GENETIC ALGORITHM

Different researches have been done to select the views for materialization. Genetic Algorithm (GA) is one of the prominent domains based on which view materialization methods have been proposed but the majority of research in the area of materialized view selection has been focused around greedy heuristics based techniques. These techniques are unable to select good quality views for higher dimensional data sets because their total view evaluation cost is high. On the other hand, genetic algorithm is a widely used evolutionary technique suitable for solving complex problems involving the identification of a good set of solutions from within a large search space [7]. Several GA based view selection algorithms have been proposed in literature [24][25][26][27][28][29][30]. These algorithms aim to select views for higher dimensional data sets with the key challenge of selecting views of high quality i.e. low Total View Evaluation Cost (TVEC). In this section of the chapter, a GA based algorithm is proposed that selects views from a multidimensional lattice. Each chromosome is represented as a string of views selected for materialization. The length of each chromosome is T for selecting top-T views for materialization.

GA is applied with a pre-defined crossover and mutation probability and a set of top-T views are generated after a pre-specified number of generations.

3.4.1 THE MULTIDIMENSIONAL LATTICE

Views involved in On-Line Analytical Processing (OLAP) queries can be represented as nodes of a multidimensional lattice [13][44]. The multidimensional lattice is a graph with no self-loops or parallel edges. The root node of the lattice represents the base facts table. All the views in the lattice either directly or indirectly depend on the root view. A view VA is said to be dependent on another view VB when the queries on VA can be answered using VB. In the graph, direct dependencies are represented by an edge between two views & indirect dependencies are discovered transitively.

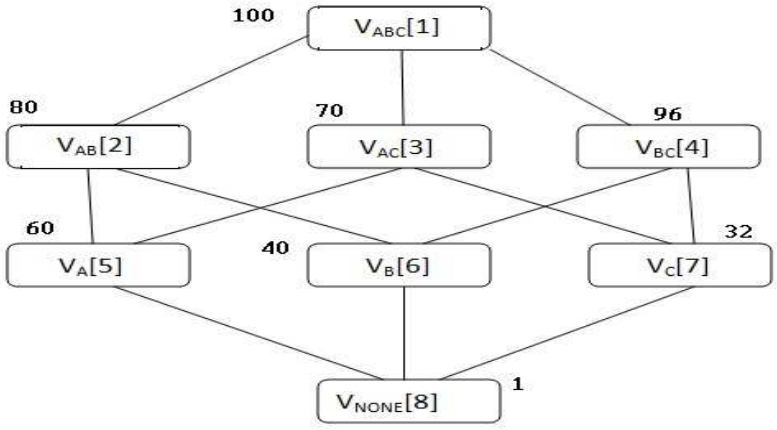

Figure 3.5: A sample multidimensional lattice with 3 dimensions

In figure 3.5, a 3-dimensional lattice is shown. The index is shown in brackets inside the node & the frequency of the variables is shown on the top left corner of each node. The root view contains all the variables A, B & C.

3.4.2 THE PROPOSED ALGORITHM

The proposed method tries to apply the basic steps of any genetic algorithm. In this regard, the chromosome selection process and the fitness function generation are critical tasks. The steps of the proposed algorithm are mentioned next.

1. [Start] *Generate random population of n chromosomes.*

2. [Fitness] *Use the fitness function f(x) to calculate the fitness value of each chromosome x in the population.*

3. [Test] *If the end condition is satisfied, i.e. , a chromosome with the desired fitness value is found in the population or a value close to that value is found (& further repetition does not generate a chromosome with a better fitness value) then stop & return the best solution that is found.*

4. [New Population] *Create a new population by repeating the following steps until the new population is complete.*

> ***a. [Selection]*** *Select two parent chromosomes from a population according to their fitness value. The higher the fitness value, the better the chances for it to be selected.*
>
> ***b. [Crossover]*** *With a crossover probability pc, combine the parent chromosomes to generate a new offspring.*
>
> ***c. [Mutation]*** *With a mutation probability pm, mutate the offspring at each locus.*
>
> ***d. [Accepting]*** *Place the offspring in the new population.*

5. [Replace] *Replace the new population with the previous population & go to step 2.*

3.4.3 CHROMOSOME REPRESENTATION

Each chromosome is represented as a string of distinct views. Each distinct gene corresponds to a view. The chromosome string contains only the index values of those views that are materialized. The chromosome representation for selecting the top-5 views for materialization is shown below:

3	5	1	4	8
1	7	2	3	5
8	6	2	1	4

Figure 3.6: Chromosome representation for selecting top-5 views

3.4.4 GENERATION OF FITNESS FUNCTION

The fitness function is used to give an estimate of how much fit a chromosome is for being selected for crossover. The quality of solution obtained by the GA based algorithm depends on the correctness & the appropriateness of the fitness function. Here, the fitness function is based on the frequency of attributes used in a view. The higher the total frequency of a chromosome, the better is its chances to be selected for crossover. The fitness function is given below:

$$TFV = \sum_{i=1}^{x} Mat(V_i) + \sum_{j=1}^{y} AncNMat(V_j), where\ i \neq j$$

TFV = Total Frequency Value

n = Total number of genes or views

x = Number of materialized views

y = Number of non-materialized views

n = x + y

Mat(Vi) = Frequency of attributes for view Vi where Vi is materialized

AncNMat(Vj) = Frequency of a materialized ancestor with maximum frequency among other ancestors of a non-materialized view Vj

For example, if views V_{AC}, V_{BC}, V_A, V_B & V_C are selected for materialization and x=5, y=3 then the TFV computation is as follows:

$$\sum_{i-1}^{5} Mat(V_i)$$

= Mat(V_{AC}) + Mat(V_{BC}) + Mat(V_A) + Mat(V_B) + Mat(V_C)

= 70 + 96 + 60 + 40 + 32

= 298

$$\sum_{j=1}^{3} Mat(V_j)$$

= Mat(VABC) + Mat(VAB) + Mat(V_{NONE})

= 0 +0 + 60

= 60

$$TFV = \sum_{i=1}^{5} Mat(V_i) + \sum_{j=1}^{3} Mat(V_j)$$
$$= 298 + 60$$
$$= 358$$

The fitness function takes the following parameters as its inputs:

1. The adjacency matrix adj[][] for the entire multidimensional lattice,
2. freq[] array which contains the frequencies of attributes for each view,
3. level[] which contains the level of each view in the graph(lattice),
4. The total number of distinct genes (tot_gene),
5. The number of genes to be materialized (mat_gene), and
6. The size of the population popl.

Along with the above mentioned parameters, the fitness function generation process also considers a list of procedures as depicted below:

- The procedure RAND (1, tot_gene) generates a random number between 1 & n where n is the total number of distinct genes.
- The procedure LINEAR (topt, k, i, mat_gene) checks if the item k is present in the ith chromosome or not. If it is, it returns true. Otherwise, it returns false.
- The procedure MAX (anc) finds out the maximum number in the anc [] array, i.e. the materialized ancestor of a non-materialized gene which has the maximum frequency.

The proposed algorithm proceeds in the following way:

- At first, it generates the initial population and stores the chromosomes in an array.
- It then calculates the total frequency value for each chromosome in the population. This value is termed as TFV1.
- Next, it finds the non-materialized views in each chromosome and also finds the maximum frequency materialized ancestor for each of those non-materialized views.
- Then the algorithm calculates the sum of the attribute frequencies of these ancestors for each chromosome. This value is termed as TFV2.
- It finally adds TFV1 and TFV2 for each chromosome to generate the Total Frequency Value (TFV).

The pseudocode of the steps mentioned above are shown next:

```
Algorithm FITNESS (adj[][], freq[], level[], tot_gene, mat_gene, popl)
    begin
    for i = 1 to popl do
      for j = 1 to mat_gene do
```

```
 while (true) do          /* Loop for removing duplicate genes */
    flag = false;
    rand = RAND(1, tot_gene);
          for k = 1 to j do
                if (topt[i][k] == rand)  then     /* If the gene is already present
in the chromosome then generate a new number */
                      flag = true;
                      break;
          end for
          if (flag == false) then
             topt [i][j] = rand;
             break;
    end while
  end for
end for
    for i = 1 to popl do
          tv1=0;
                for j = 1 to mat_gene do
          k = topt[i][j];
                      tv1 = tv1 + freq[k];
    end for
tfv1 [i] = tv1;
    end for
```

```
        flag = false;
        for i = 1 to popl do
k = 1; l = 0;
  while (k <= tot_gene  and  l < n_mat_gene) do
flag = LINEAR(topt, k ,i, mat_gene);
if(flag == false) then
        mma [i][l]=k;
        l++;
    k++;
  end while
    end for
        for i = 1 to popl do
tv2 = 0 ;
for j=1 to n_mat_gene do
k = mma[i][j];
q=0;
anc[]={0,0,0,0,0};
for  p= 1 to tot_gene do
        if(level[k]==1)
                break;
        else if ((adj[k][p] == 1) and (level[p] == (level[k]-1))) then        /* If the entry in the adjacency matrix is 1 & the node is in the previous level then it is an ancestor */
```

anc[q++] = freq[p]; /* Add the ancestor to anc[] */

end for

m =**MAX***(anc);* /* Calculate the freq that is max because we need max freq ancestor */

tv2 = tv2 + m;

end for

tfv2[i]=tv2;

end for

for i=1 to popl **do**

TFV[i]=tfv1[i]+tfv2[i];

end for

end

3.4.5 SELECTION

Selection is a process of choosing individuals from a population for performing crossover. The fitter individuals are more likely to produce fitter offsprings in the subsequent generations. Selection of only fitter individuals might hinder exploration of the search space thereby leading to early convergence.

Chromosomes (top-T Views)	Fitness value(TFV)
[3 6 7 8 2]	303
[2 5 4 1 8]	629
[2 3 1 8 6]	541
[7 4 8 5 1]	595
[7 3 8 1 6]	513
[6 2 1 7 4]	568

Table 3.2: TFV of top-T views in V_{TopT}

Therefore, there is a need to randomly select individuals where individuals, having higher fitness value, have greater likelihood of being selected for crossover. The approach uses binary tournament selection method where two individuals are randomly selected from the population and a tournament is conducted among them. A value r is randomly generated between 0 and 1. If r is less than the pre-defined value of k (say k = 0.75), taken between 0 and 1, the individual having higher TFV is selected else the individual with lower TFV is selected. The steps for the algorithm used for Selection process are stated below:

Step1: Given: parameter k = 0.75.
Step 2: Choose randomly two individuals from the population.
Step 3: Choose: a Random number r between 0 and 1.
Step 4:
a) If r < k, Select the fitter individual among the two individuals.
b) Else, Select the less fitter individual among the two individuals.

The result of the above algorithm is reflected in table 3.3.

Randomly generated Indexes [i] [j]	Tournament between individuals [P(i)] [P(j)]	Fitness [TFV(P(i))] [TFV(P(j))]	Random Number (r)	Individual selected
[2] [3]	[2 3 1 8 6] [7 4 8 5 1]	541 585	0.9765590312995504	[2 3 1 8 6]
[5] [0]	[6 2 1 7 4] [3 6 7 8 2]	303 568	0.7209919519618984	[3 6 7 8 2]

Table 3.3: Selection of top-T views using Binary Tournament Selection process

3.4.6 CROSSOVER

After the selection process, crossover comes to create a new solution by selecting parameters or genes from two parent solutions. In the proposed methodology, two individuals are randomly selected and are recombined using single point crossover method with crossover probability, $p_c = 0.5$. A uniform random number, r, is generated and if $r \leq p_c$, the two randomly selected individuals undergo crossover. Otherwise, the two offspring are simply copies of their parents [45].

The process is depicted through figure 3.7.

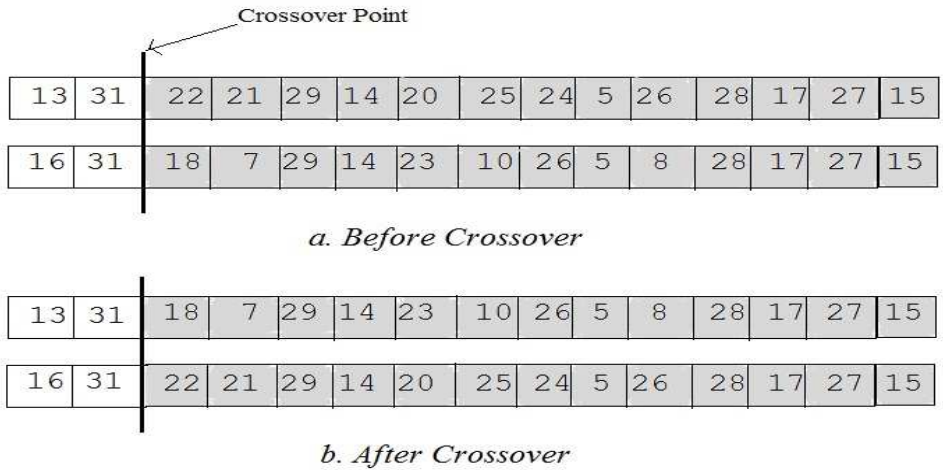

Figure 3.7: Outcome of Single Point Crossover

The steps for Crossover process are mentioned next.

Step 1: Select two parents randomly for crossover from the population

Step 2: Generate random number *r*

Step 3: If r < = pc then do

a. Randomly generate a point for crossover

b. Perform single point crossover between the parents corresponding to the randomly generated point

c. If there are duplicate genes within either of the children after crossover, then repeat Step 1-3

d. Else

If any of the children have already been generated before to be placed in the new population then repeat steps 1-3

Else

Do not perform crossover. Simply copy the parents to their corresponding children.

3.4.7 MUTATION

Mutation is performed using random resetting method with predefined mutation rate, pm. A random number rand is generated & if it is less than the mutation rate only then mutation is performed. Otherwise, mutation is not performed. So, the actual number of genes to be mutated is not fixed. For a chromosome of length L, on average L*pm genes will be mutated [46]. The method ensures that there are no duplicate genes in the chromosome after mutation. This process is demonstrated through figure 3.8.

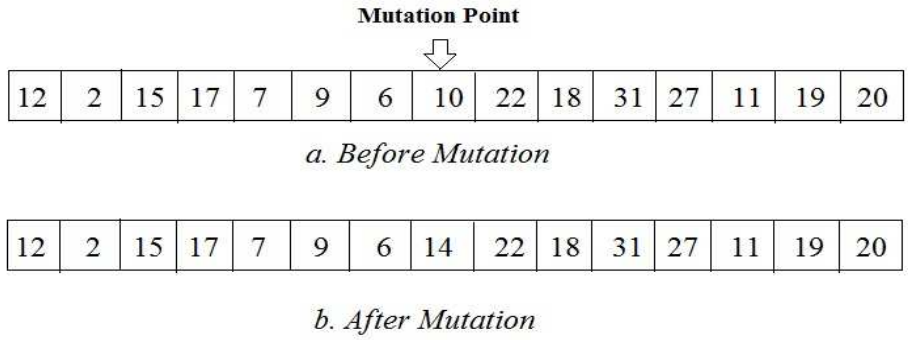

Figure 3.8: Outcome of Mutation process

The steps for Mutation process are stated below:

Step 1: for i=1 to L do

a. Generate *rand*

b. if rand < pm then

i. Generate a random gene value

ii. Mutate gene[i] of the selected chromosome with the random gene value generated in the previous step

iii. if the newly added gene is a duplicate then repeat steps i and ii

Step 2: Repeat Step 1 for all the chromosomes until the end of the population.

3.4.8 REPLACEMENT

The final step for Genetic Algorithm is Replacement where after the generation of the new population, the old population is replaced with the new population. Then the entire cycle of fitness value calculation, selection, crossover, mutation, and replacement for a pre specified number of generations is performed.

3.4.9 RESULTS OBTAINED AND ANALYSIS

The GA based view selection algorithm was implemented using jdk 1.7.0_55 in Windows 7(x64) environment. The algorithm was successfully tested for a 3-dimensional and a 4-dimensional lattice of views. The result for selecting top-5 views for materialization from a 3-dimensional lattice is shown in figure 3.9. From the figure it can be observed that the matrix for top-T views contains distinct views in each row of the matrix. Duplicate views are eliminated since the data warehouse does not require redundant data for recovery. TFV1 is the first part of the fitness function that calculates the total frequency value for only materialized views. TFV2 is the second part of the fitness function that calculates the total frequency value for non-materialized views. These two are added to give the TFV for each chromosome.

The size of the adjacency matrix increases exponentially with the increase in lattice dimensions. Hence, with the increase in dimensions of the lattice, the complexity increases. When we calculate the attribute frequencies for a non-materialized view, we use the frequency of attributes of the maximum materialized ancestor of that non-materialized view because the non-materialized view is dependent on the ancestor. This means that all the queries for the non-materialized view can be answered using its materialized ancestor, without requiring any other views to be materialized. But, if the root view itself was not materialized then we use a frequency value of zero in the calculation because there are no other views in the lattice which can give answers to all the queries that can be made to the root view as no other view but the base or root view contains all the attributes.

A frequency value of zero is also used when a non-materialized view has no ancestors. This is because only an ancestor view that the child view is dependent on or the child view itself can give answers to all the queries that can be made to the child view.

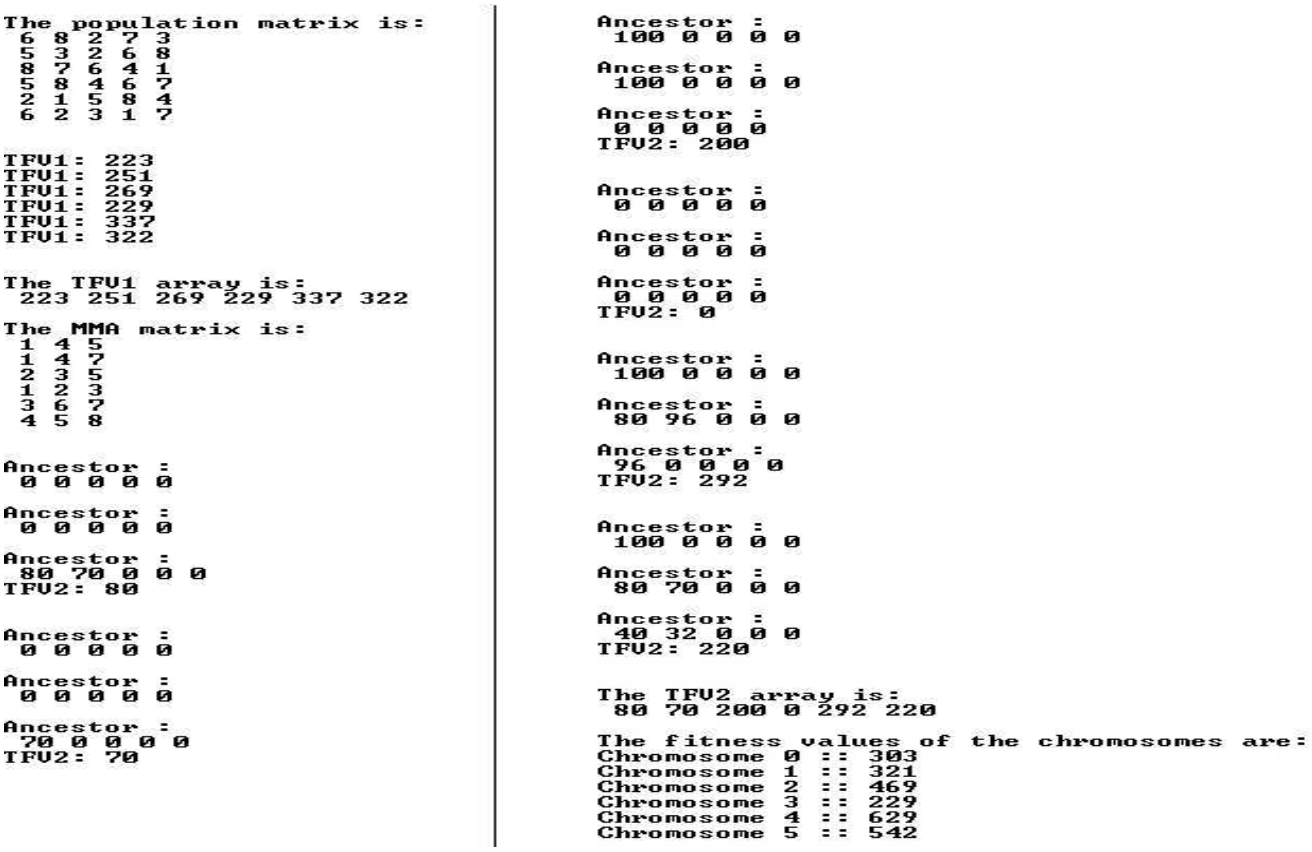

Figure 3.9: Result for selecting top-5 Views from a 3 Dimensional Lattice

The proposed GA based view selection algorithm using frequency of attribute (FA) and another GA based view selection algorithm using size of attribute (PA) are compared. The comparisons are carried out on Fitness value due to views selected by the two algorithms. The experiments were performed for selecting the top-T views for materialization for dimensions 2 to 5 over 1000 generations. First, the graphs showing Fitness value for different crossover and mutation probabilities for selecting top-6 views, were plotted and compared with Fitness value of selecting top-6 views using PA. These graphs, plotted for pair of crossover and mutation probabilities (0.5, 0.05), (0.6, 0.05), (0.5, 0.1), (0.6, 0.1), (0.55, 0.1), (0.55, 0.05), (0.65, 0.1), (0.65, 0.05), are shown in figure 3.10. The graphs show that FA, in comparison to PA, is able to select views that are Fitter than that obtained by PA for different crossover and mutation probabilities. This difference in Fitness value becomes significant for higher dimensions. Further, it can be observed from the graph that, for crossover probability 0.6 and mutation probability 0.05, the best result is obtained by FA in the Fitness value across all dimensions.

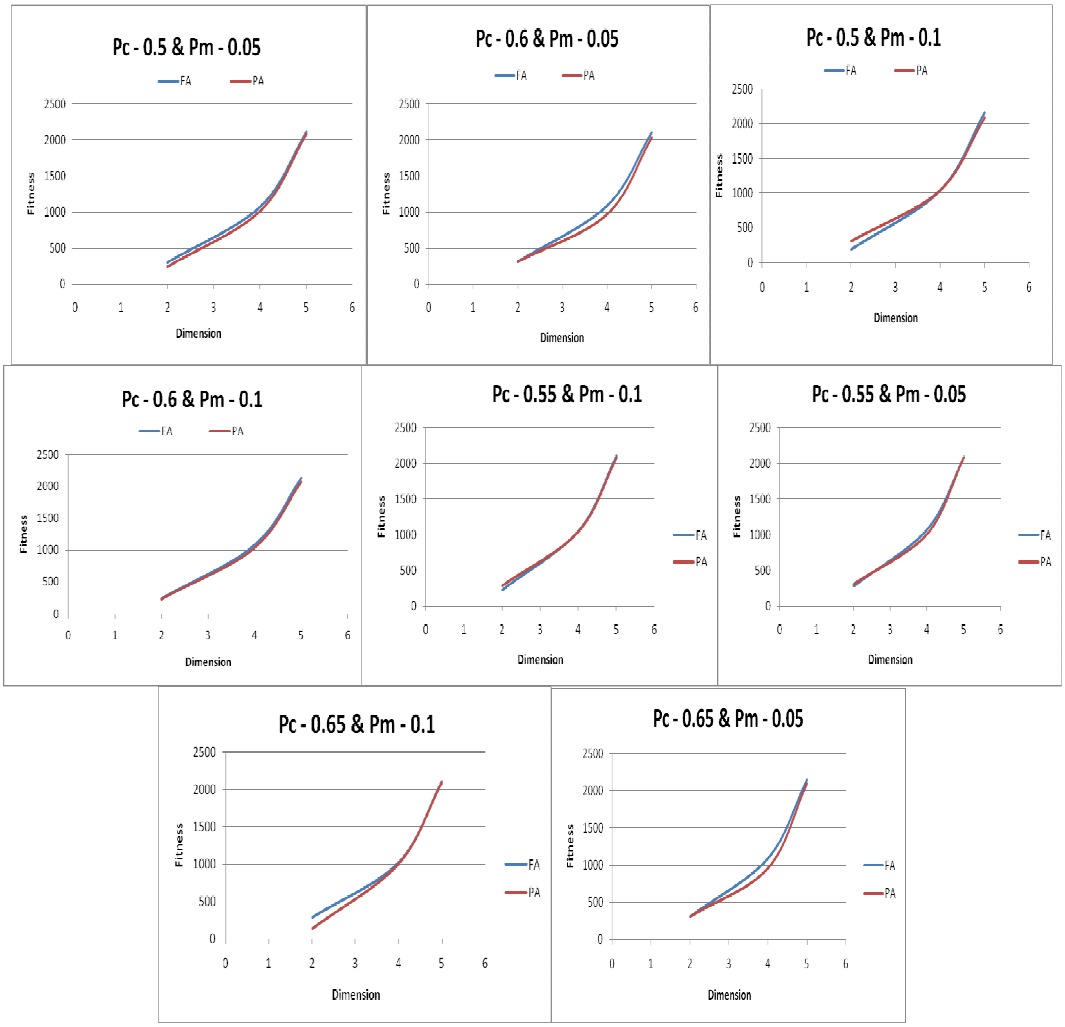

Figure 3.10: Comparison of FA and PA – Fitness Vs. Dimensions for different Pc's and Pm's

3.5 CONCLUSION

This chapter has discussed about two different approaches for view materialization based on the steps followed by a genetic algorithm. The first work generates a materialized view after considering both space and time as two influencing parameters and also compares the present work with a previous work based on genetic algorithm. Observing the result it can be concluded that after the inclusion of the time factor, materialized view generation becomes more optimized. The work can further be expanded to include a constraint in selecting a particular set of views to be materialized instead of considering all possible views that can be generated. This constraint can be a single view generation cost which can be merged with the view size that is already considered in the lattice structure. In a nutshell, the present algorithm has considered time domain as well as the space domain for materialized view generation.

There is a further scope of introducing the frequency domain with this. In the frequency domain, the frequencies of occurrence of individual queries consisting of the participation dimensions have to be considered. Moreover, on the results obtained in the present work, this research work can be expanded to utilize materialized view in incremental data mining applications.

The other approach described in this chapter has used genetic algorithm and has used a fitness function to evaluate the fitness values of the possible solutions. One of challenging directions of future work aims at addressing the view selection problem in a distributed setting. Randomized algorithms can be applied to complex problems dealing with large or even unlimited search spaces. Using this approach of view materialization enables exploration and exploitation of the search space. As a result, the views so selected are likely to have a higher total frequency value. Further experimental results show that the views selected using the proposed algorithm have a comparatively higher frequency value to those selected using the sizes of the attributes for the observed crossover and mutation probabilities. That is, the genetic algorithm based methodology using frequency is able to select comparatively better quality views. This in turn results in reduced query response time enabling efficient decision making.

CHAPTER 4
MATERIALIZED VIEW CREATION USING APRIORI ALGORITHM

4.1 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 chapter is based on Apriori algorithm proposed in [47]. 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.

4.2 AN OVERVIEW OF BOOLEAN ASSOCIATION RULES

The backbone of the proposed methodology is Boolean Association rules which can have two different but dependent measures – support and confidence which respectively reflect the usefulness and certainty of discovered rules. An association rule is considered to be significant.

But it is to be ensued that this rule satisfies both a minimum support threshold and a minimum confidence threshold set by the strategists or the analysts.

Let I = {I1, I2,..., Im} be an itemset. Let D, the task-relevant data, be a set of database transactions where each transaction T is a nonempty itemset such that $T \subseteq I$. Let A be a set of items. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, $A \neq \phi$, $B \neq \phi$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain both A and B. The rule $A \Rightarrow B$ has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. Rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong. The rule confidence is associated with the rule support using the following relation:

confidence $(A \Rightarrow B)$ = support$(A \cup B)$ / support(A)

4.3 AN OVERVIEW OF APRIORI ALGORITHM

Apriori algorithm was proposed in [47] 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 => 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 => B) = $P(A \cup B)$ and confidence(A => 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, which is a seminal algorithm as described in [47], identifies the frequent item sets for Boolean association rules. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k + 1)-itemsets. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space.

Apriori property tells that all the nonempty subsets of a frequent itemset must also be frequent. 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 entire method is divided into two basic steps: join step and prune step.

4.3.1 JOIN STEP OF APRIORI ALGORITHM

To find L_k , a set of candidate k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}. The notation $l_i[j]$ refers to the j^{th} item in l_i. The join operation, denoted as $L_{k-1} * L_{k-1}$, is performed, where members of L_{k-1} are joinable if their first $(k-2)$ items are in common. That is, members l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ ensures that the join operation does not generate any duplicate entry. The resulting itemset formed by joining l_1 and l_2 is $\{l_1[1], l_1[2],..., l_1[k-2], l_1[k-1], l_2[k-1]\}$.

4.3.2 PRUNE STEP OF APRIORI ALGORITHM

C_k is a superset of L_k , that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A database scan to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). Since C_k may contain a huge item set, this may involve heavy computation. To reduce the size of C_k, the Apriori property is used. Any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any $(k-1)$-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This is done through a process call subset testing.

4.3.3 PSEUDOCODE OF APRIORI ALGORITHM

Apriori algorithm primarily aims to find the frequent itemsets using an iterative level-wise approach based on candidate itemset generation. The algorithm takes two parameters as inputs – D, a database of transactions and min_sup, the minimum support threshold. The output of the algorithm is L, the frequent itemsets in D. The pseudocode of the algorithm is divided into three methods as given below:

L_1 = find frequent 1-itemsets(D);

for ($k = 2; L_{k-1} \neq \varphi; k++$) {

C_k = apriori_gen(L_{k-1});

for each transaction $t \in D$ {

C_t = subset(C_k , t);

for each candidate $c \in C_t$

c.count++;

}

$L_k = \{c \in C_k \mid c.count \geq min\ sup\}$

}

return $L = \cup_k L_k$;

procedure apriori_gen(L_{k-1}:frequent (k − 1)-itemsets)

for each itemset $l_1 \in L_{k-1}$

for each itemset $l_2 \in L_{k-1}$

if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then{

c = l1 ✶ l2; //The join step of Apriori algorithm

if has_infrequent_subset(c, L_{k-1}) then

delete c; // The prune step of Apriori algorithm

else add c to C_k;

}

return C_k ;

procedure has_infrequent_subset(c: candidate k-itemset, L_{k-1}: frequent (k − 1)-itemsets) // use of Apriori property

for each (k − 1)-subset s of c

if s ∉ L_{k-1} then

return TRUE;

return FALSE;

4.4 THE PROPOSED METHODOLOGY

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 [47].

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 is considered for materialization. In the final case, the output of the second last 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 A1 to A5 and only A1 and A3 are selected for materialization after applying the first phase then in the second phase, the confidence values of A2, A4 and A5 on A1 and A3 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 [47], 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 = {T_1, T_2, T_3, …, T_n}

Initialize M by Φ, i.e., null set

for i =1 to n

do

Let A = A set of attributes involved in i^{th} transaction T_i

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 U 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 U 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.

4.5 RESULTS AND ANALYSIS

In the context of the present chapter, 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 A1 to A6. The following is the description of the results obtained.

Test case 1:

Table 4.1 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 A1, A2, A3 and A4 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.

Transaction	Attribute Set	Binary	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

Table 4.1: The attributes in all transactions along with their binary and decimal values.

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

Table 4.2: The outputs of Apriori Algorithm applied on the transaction set as shown in table 4.1

For example, from table 4.1, if the tuple corresponding to the transaction id 4 is considered, only attributes A3 and A4 are selected, so its equivalent binary entry will be 1100, where the leftmost 1s identify that two attributes A4 and A3 are selected in this transaction and the rightmost 0s signify that in this transaction, two more attributes A1 and A2 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 4.2, is split into two pages and it stores all the frequency values against iterations which are based on Apriori algorithm. According to this algorithm, 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 chapter, the threshold frequency value has been chosen to be 2. From table 4.2, it is clear that the required frequent attribute sets are 15 and 23, i.e., 11112 and 101112 respectively because these two sets have frequency, which is termed as the support value, above the threshold of 2. In other words, attributes A1, A2, A3, A4 and A1, A2, A3 and A5 are identified separately. Since two different sets of attributes are selected, their intersection is found out and it generates A1, A2 and A3, which has an equivalent decimal value of 7. So, according to the algorithm MVG_AA ( ), the attributes A1, A2 and A3 are to be materialized.

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

Table 4.3: The list of confidence values obtained from the result as shown in table 4.1

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 4.3. 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 4.3 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 4.3, the attribute or the attribute set that is dependent on 7, i.e., A1, A2 and A3 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 A1, A2 and A3.

Test case 2:

Table 4.3 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 4.1. The next table, i.e., table 4.4 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 table 4.5, 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 A1, A2, A3, A4 and A1, A4, A5 and A6 are identified separately. Since two different sets of attributes are selected, their intersection is found out and it generates A1 and A4 or 1001_2 which has an equivalent decimal value of 9. So, A1 and A4 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 4.6, the attribute or the attribute set that is dependent on 9, i.e., A1 and A4 is marked as bold. It is clear that only 6 (or 110_2), i.e., the set, containing A2 and A3 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 A1, A2, A3 and A4.

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 4.4: The attributes in all transactions along with their binary and decimal values

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.

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 4.5: The outputs of Apriori Algorithm applied on the transaction set as shown in table 4.4

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

Table 4.6: The list of confidence values obtained from the result as shown in table 4.5

4.6 CONCLUSION

This chapter 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 work in this chapter 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.

CHAPTER 5
APPLICATION OF MATERIALIZED VIEW IN INCREMENTAL DATA MINING OPERATION

5.1 INTRODUCTION

A view is a database object which is used to store the results of a query and can be thought of as a logical table often used for query processing. A materialized view as described in the previous chapters is a relatively newer concept and is another database object that, unlike a normal view, physically stores the results of a query set. It can also be defined as cache which is mainly used for faster query processing. Materialized views help in eliminating the overhead involving expensive joins and aggregations that need to be performed for a large class of queries and offer significant improvements in query processing time, particularly for aggregation queries over tables of considerable size and also in consolidating complex query logic. Effectively, this helps developers in easier data transformation.

Materialized views essentially materialize the results produced by the query in order to reduce access times [48]. Use of materialized view is mainly associated with data warehouse containing a huge amount of historical data. From this large data set, many data may not be actually useful for the future use. Here is the exact role of data mining which has become an important branch of database applications, especially with big data handling and is extensively used in the fields of decision support system, market analysis or any type of data processing. It is an iterative method of extracting knowledge from data set through some mathematical modelling and statistical analysis.

Incremental data mining is a form of data mining in which the existing mined data set is updated with the newly mined data with the process being repeated iteratively. This becomes evident when the users are not immediately satisfied with the results generated by a specific database queries. Hence the process of getting the exact resultset may be an iterative approach where in each consecutive step the users evaluate the patterns obtained from the previous iteration and based on the needs and expectations modify either the mined dataset or the parameters the inherent algorithm requires, or both. Because of this iterative nature of data processing through mining applications, a data mining system should utilize the resultset generated from the previous queries to accomplish the actual user requirements.

For any online analytical processing, where there may be an involvement of data warehouses, the resultset is materialized for the use of next iteration. But the challenge lies in the changing pattern of the generated outputs that may occur in handling substantially large amount of data. Each refresh or update operation may invalidate some previously stored patterns stored in the form of materialized views. The incremental data mining technique tries to tackle this situation. In incremental data mining process, the already mined data sets need to be modified depending on the users' ever changing requirements. So, a challenge in this field is to find out the relevance of the outputs of new user queries with the existing set of data. To do this, the association between the past data and the present data is to be identified. The proposed method as discussed in this chapter is related to the use of materialized view in implementing incremental data mining operations. The method deals with a database transaction set which consists of a set of attributes. The main algorithm on which the present method has been set up is Apriori algorithm as described in [47] and using this algorithm, the frequencies of occurrences of these attributes are obtained and the attributes which are above a particular threshold value of the frequency level are selected for materialization. Later with a new set of database transactions with the same set of attributes, the frequencies are calculated and if any change is observed with the frequencies, the materialized view gets updated.

5.2 THE PROPOSED METHODOLOGY

The backbone of the proposed methodology in connection with the incremental data mining operation is Apriori algorithm, which has already been described in the previous chapter. In any transaction processing, the daily activities performed by the different users on a particular database are monitored. Considering this aspect, the present work as described here has two main components as mentioned below:

Initial transaction processing component: This contains the frequently occurring sets of attributes which may be identified to form the materialized views.

Revised transaction processing component: This contains the attributes which may be added to the exiting materialized views. This revision is done using an iterative approach.

The Initial Transaction processing component is obtained by applying the algorithm MVG_AA, described in chapter 4. The way MVG_AA utilizes the Apriori algorithm is briefly presented next.

The Apriori algorithm consists of two basic steps, viz., join step and the prune step and since the basic idea of the Apriori algorithm is to select a confidence value and a support value, the same has been done here as well. The support value is the minimum number of times that an attribute has to occur in the set of transactions to be considered as a frequently occurring attribute and the confidence value is used to determine if an attribute should be added to the existing set of materialized views consisting of the already identified frequent attribute sets. If the percentage of transactions containing the attributes in the materialized view that also contains the concerned attribute is greater than or equal to the confidence value, then this new attribute has been added to the materialized view. This updated materialized view is then used in the subsequent iterations.

To implement the proposed method, the input data set from a transaction processing environment is taken as a text file containing a set of 'n' number of transactions T, where T = {T1, T2, T3, …, Tn}, where Ti contains a set of attributes participating in the ith transaction. In the proposed method, each participating attribute is identified by a positive number. So, if the number of participating attributes is stored in a variable called 'm' then the first attribute is identified by the number 1, the second one is identified by the number 2 and so on and so forth with the last attribute is identified by the number m. Moreover, since each transaction contains a set of attributes, they are mentioned in the transaction set. For example, if the i^{th} transaction contains the first attribute, the fourth attribute, the fifth attribute and the eighth attribute (with the assumption that the value of the previously considered variable 'm' >=8) then Ti = {1, 4, 5, 8}.

The proposed algorithm to implement incremental data mining operation has been named as Materialized View in Incremental Data Mining or MV_IDM. This method depends on the output generated by the algorithm called Materialized View Generation using Apriori Algorithm or MVG_AA which has been described in chapter 4. With the help of MVG_AA, the algorithm for the work depicted in this chapter has been developed and the pseudo-code of the proposed algorithm is given next.

Algorithm MV_IDM ( )

{

Input: no_iter = The number of iterations required to find out the incremental result on an already mined data set, DT = A set consisting of 'no_iter' numbers of different sets of database transactions at different time frame on the same database and ATR = A set of attributes on which different transactions are to be executed.

Output: R = The set of materialized views to be considered for incremental data mining operation and R is initialized by Φ, i.e., null set.

Let DT = {ST_1, ST_2, ST_3, …, ST_{no_iter}}

for i = 1 to no_iter

do

Mi = MVG_AA (ST_i, ATR) /* It returns the materialized view containing the most frequent itemsets for a single transaction set STI and all the participating attributes are there in the set ATR */

R = R U M_i

done

return R

}

As evident from the above-mentioned pseudo-code, MD_IDM iteratively invokes MVG_AA. The number of such iterations depends on the number of times the transactions are to be considered to generate the finally mined attribute set. In MV_IDM, the set DT contains the sets of transactions. So, each element of DT is itself a set of transactions. For example, if ST_i, which is the i^{th} set of transactions to be passed as a parameter to MVG_AA, contains the first transaction, the third transaction and the seventh one (with the assumption that the value of the variable 'n' >=7.

Since this variable stores the number of transactions in MVG_AA) then $ST_i = \{T_1, T_3, T_7\}$. So, the i^{th} call to MVG_AA from MV_IDM would take the transactions T_1, T_3 and T_7 only to identify the attributes to be considered for storing in the materialized view for that particular iteration. Similarly, in other iterations also, other transactions are considered and accordingly new materialized view is returned.

The algorithm MV_IDM has been executed on a few randomly generated test cases as described in the next section. Each test case has been assumed to be a database transaction. The final outcome of the algorithm has been chosen to be the concluding data set to be mined.

5.3 RESULTS AND ANALYSIS

The algorithm MV_IDM, as described in the previous section, has been implemented on a system having Windows Operating System with Intel Core 2 Duo 2.0 GHz CPU (x86) processor and 2 GB of primary memory. For the testing purpose of the algorithm, four sets of different transactions have been chosen, with each of the transactions having twenty different attributes of a randomly generated database. For the implementation purpose, each attribute has been assigned with a number starting from 1. So the twenty different attributes have been numbered from 1 to 20. Moreover, in each set, eighteen different transactions have been chosen. To justify the applicability of the proposed algorithm, the participating attributes in each of the transactions have been randomly chosen. The first four tables, i.e., table 5.1 to table 5.4 store the participating attributes in each transaction under each set. So, the value of the 'no_iter' variable, as specified by the algorithm MV_IDM, is 4 here for the above-mentioned test case. In each of the iterations, a new set of 18 different transactions have been added to the existing list of transactions which is initially set to empty.

Moreover, in the very first iteration, the number of transactions considered is eighteen; in the second iteration, the number of transactions considered has been 36, because a new set of 18 different transactions have been added with the existing 18 transactions which have already participated in the first iteration. In the same way, the steps of the next iterations have been executed and accordingly the number of transactions has also increased.

The algorithm MV_IDM has been iteratively applied on the different transaction sets as depicted from table 5.1 to table 5.4 and the result obtained after each of the iterations is described after the results shown in table 5.4. The minimum confidence value in each of the iterations has been chosen to be 0.7, which has been a standard value in different applications.

On the other hand, the minimum support value has been determined to be dependent on the number of transactions considered in each of the iterations. In this case, it has been chosen to be 1/3rd of the total number of the participating transactions. This value has been chosen because it is a moderate value in the sense it is less than ½ which may become relatively high to consider participation of the attributes and also it is more than ¼ which may be relatively a small number for consideration. According to this specification, in the first iteration, since the number of participating attributes is 18, the minimum support value has been chosen to be 6 (1/3rd of 18). In the next iteration, as the number of participating attributes has been increased to 36, the minimum support value also got increased and has been set to 12 (1/3rd of 36). In this way, the minimum support values for the other iterations have also been chosen.

Transaction number	**Participating attributes**
1	1,3,4,5,6,7,8,10,11,12,15,16,17,20
2	2,3,4,6,7,8,10,12,13,14,15,16,17,1
3	2,9,14,16
4	1,2,3,4,5,6,7,8,9,10,11,12,14,16,17
5	1,3,5,6,7,8,9,10,11,12,13,14,15,16,
6	2,4,5,6,7,8,9,10,11,12,13,14,15,18,
7	1,2,3,4,5,6,8,9,10,11,12,13,14,15,1
8	1,2,3,8,13,15
9	2,6,16
10	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
11	2,3,5,6,7,8,11,12,13,15,18
12	1,3,10,11,15,18,19
13	13
14	2
15	3,4,5,6,10,11,12,13,14,16,17,18,20
16	9,10
17	1,2,3,4,6,7,8,9,12,13,14,15,16,17,1
18	2,3,4,7,8,10,12,16,18,20

Table 5.1: The first set of transactions under consideration

For the implementation purpose, the set of attributes participating a single transaction is assigned with a binary number as a whole. This approach follows the methodology described in chapter 4.

The participating attributes have been identified by ‘1’s and the attributes which are not participating in a transaction have been identified by ‘0’s and thus forming a binary pattern. For example, if in a transaction attribute numbers 6, 4 and 3 are only participating and the attributes with numbers 5, 2 and 1 are not participating then the corresponding binary pattern is formed to be 101100, which has a decimal equivalent value of 44. So, the number ‘44’ identifies the participating attributes in this transaction.

Transaction number	Participating attributes
1	5,8,9,10,15,17,20
2	2,20
3	1,2,3,4,5,6,8,9,10,12,13,14,16,17,1
4	6,7,12,13,14
5	2,4,7,8,10,13,14,15,20
6	1,2,3,5,6,7,8,9,10,11,12,14,15,16,1
7	4,16,18
8	1,4,14
9	1,2,3,5,6,9,10,11,12,13,14,16,17,1
10	1,2,4,7,8,9,12,13,14,17,20
11	1,2,3,4,7,8,9,11,13,14,15,17,18,19
12	1,2,3,5,7,8,9,11,12,13,14,15,17,19,
13	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
14	1,2,4,12,15,16
15	1,4,6,7,8,10,11,12,13,14,15,16,17,
16	1,2,3,4,6,7,8,11,12,13,14,15,16,17,
17	1,5,6,9,10,12,14,15,16,18,20
18	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

Table 5.2: The second set of transactions under consideration

In the first iteration, shown in table 5.1, since the number of transactions was 18, the minimum support value was chosen to be 6 as already explained and after this iteration, only one frequent set was identified and that was [6, 8, 12, 14, 16, 18, 19] which had the minimum support value of 6, i.e., this set has occurred 6 times in the whole set of transactions as depicted in table 5.1. In accordance with the reason explained above, this set is identified by the binary number 01101010100010100000 (with the MSB corresponds to the 20^{th} attribute and the LSB corresponds to the 1^{st} attribute) which has the equivalent decimal number 436384.

The next step was to find out the confidence value of the other attributes on [6, 8, 12, 14, 16, 18, 19]. This calculation is based on Boolean Association rules as described in the previous chapter. It was calculated that no other attributes crossed the minimum confidence value of 0.7 and hence after the first set of 18 transactions, the materialized view set has become [6, 8, 12, 14, 16, 18, 19] and the non-materialized view set has become [1, 2, 3, 4, 5, 7, 9, 10, 11, 13, 15, 17, 20]. So, it can be concluded that the seven attributes present in the materialized view have got preference over others and thus only these attributes can be mined. Now, the aim of the algorithm is to identify whether any new attribute is mined in the next iteration or not. It may also happen that after the next iteration, any existing attribute may have to be omitted from the set with the introduction of the new ones.

Transaction Number	**Participating attributes**
1	6,10
2	4,5,7,15
3	1,3,4,6,8,9,11,12,14,15,16,17,18,1
4	2,3,5,8,9,11,18,20
5	1,2,3,4,5,7,8,9,11,12,13,14,15,17,1
6	1,2,4,7,9,10,11,13,14,15,16,17,18,
7	2,3,4,7,10,11,15,16,17,18,19,20
8	1,2,3,5,6,7,8,9,10,11,12,13,14,15,1
9	1,2,3,4,6,8,9,12,14,15,17,20
10	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
11	3,6,8,12,13,14,16,17,19
12	1,2,3,5,6,7,11,16,17,19,20
13	2,3,5,6,7,8,10,12,15,17,19,20
14	2,3,4,5,6,8,9,10,11,12,14,15,16,18,
15	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
16	1,4,6,8,11,12,14,15,20
17	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
18	8

Table 5.3: The third set of transactions under consideration

Similarly, in the second step of the iteration, as shown in table 5.2, a new set of 18 transactions was considered along with the already considered set and hence in this iteration, the total number of participating transactions was 36 and the minimum support values has been increased to 12 (1/3rd of 36).

After applying the algorithm, only the set [10, 14, 16, 18] was identified to be materialized, i.e., crossed the threshold value of the minimum support. It was also found that the above-mentioned set has got the minimum support value of 13. So, like the previous iteration, the next step was to find out the confidence value of the other attributes on [10, 14, 16, 18], i.e., on 00101010001000000000_2 or 172544_{10}.

Transaction	Participating attributes
1	1,5,14
2	1,3,5,10,12,14,19,20
3	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
4	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
5	1,2,3,4,5,7,8,9,11,12,13,14,15,17,1
6	5,20
7	1,3,4,6,7,8,9,10,11,12,13,14,15,16,
8	4,10,11,14,16,17,18
9	7,12,14
10	1,2,6,8,10,12,13,17,19
11	5,12,15
12	1,2,3,6,11,12,13,14,15
13	3,4,14,17,18
14	1,3,7,15,19
15	2,3,4,5,6,7,8,9,10,12,13,14,15,16,1
16	2,6,9,11,12,15
17	4
18	1,2,3,4,5,6,7,8,10,11,12,13,14,15,1

Table 5.4: The fourth set of transactions under consideration

Again, it was identified that no other attribute has confidence value above or equal to the predefined minimum confidence value of 0.7. So, after the second iteration, the materialized view set became [10, 14, 16, 18] and hence this set contained the attributes to be mined.

The interesting point is that one new attribute, i.e., attribute number 10, has been added in the materialized set and the four attributes (6, 8, 12 and 19) got removed from the materialized set that was generated after the first iteration. After the third iteration also, as shown in table 5.3, no other attribute could cross the minimum confidence value and the materialized view was identified to be the set [10, 17, 19].

But after the fourth iteration, i.e., the final iteration, as shown in table 5.4, as it has happened for the test case, more sets have satisfied the minimum support value as well as the minimum confidence value. In this iteration, the total number of transactions involved was 72 and hence the minimum support value was chosen to be 24 (1/3rd of 72). With the result of this final iteration, the next steps are to be executed.

It was found that more than one set of attributes have crossed the threshold value of 24. The sets and their corresponding support values are given in table 5.5. This table also lists the decimal values of all the selected sets of attributes.

Set of attributes	**Decimal Value**	**Support Value**
[3, 12, 14, 16]	43012	25
[4, 16]	32776	28
[3, 14, 16]	40964	25
[3, 16, 19]	294916	25
[15, 16]	49152	26
[14, 16, 18]	172032	25
[14, 16, 19]	303104	25
[6, 12, 14, 16, 19]	305184	24
[8, 14, 16]	41088	24
[12, 15, 16]	51200	24
[12, 14, 16, 19]	305152	24

Table 5.5: The sets with their respective support values after the fourth iteration

As a part of the algorithm, the next step was to find the final materialized view and to do that, it was required to check the confidence values of the attributes on each of the attribute set already chosen as displayed in table 5.5.

If the confidence value of any such attribute crosses the threshold value of the minimum confidence value chosen then only the attribute will be considered in the final view materialization set.

It may be observed from table 5.5 that in all of the transactions attribute number 16 has appeared. So, in the next phase of the implementation, it was required to check whether any other attribute or attribute set in each transaction had confidence value above the minimum confidence value.

Set of Attributes	**Set of other attributes on attribute number 16**	**Decimal value of the set in column 2**	**Confidence value of the set in column 2 on attribute number 16**
[3, 12, 14, 16]	[3,12,14]	10244	0.675675675675675
[4, 16]	[4]	8	0.756756756756756
[3, 14, 16]	[3.14]	8196	0.675675675675675
[3, 16, 19]	[3.19]	262148	0.675675675675675
[15, 16]	[15]	16384	0.702702702702702
[14, 16, 18]	[14.18]	139264	0.675675675675675
[14, 16, 19]	[14,19]	270336	0.675675675675675
[6, 12, 14, 16, 19]	[6,12,14,19]	272416	0.648648648648648
[8, 14, 16]	[8,14]	8320	0.648648648648648
[12, 15, 16]	[12.15]	18432	0.648648648648648
[12, 14, 16, 19]	[12,14,19]	272384	0.648648648648648

Table 5.6: The confidence values of all other attributes on the attribute number 16

The next table, i.e., table 5.6, depicts the confidence values obtained in all of the above-mentioned cases. It is observed from this table that only in the second and the fifth rows, the confidence value is above the minimum confidence value of 0.7. These two rows correspond to the attribute number 4 and the attribute number 15 respectively.

Hence, it may be concluded that along with the attribute number 16, two more attributes, i.e., attribute 4 and attribute 15 may be considered for the final set of materialized view and hence in the final set of mined data.

So, the finally selected set of attributes for data mining after the above-mentioned four iterations is [4, 15, 16]. This selection of attributes for data mining is purely based on the frequency of occurrence of the attributes.

Thus, materialized view has been successfully used in implementing data mining operations, specifically when the input data sets are changing dynamically.

There is a limitation of the proposed methodology. Since this method is based on Apriori algorithm, only the frequencies of the attributes are considered for view materialization. There may be other possibilities and factors for considering materialized view in incremental data mining. The factors may include the relevance of the output of a query with the existing mined data set and the space constraint of the mined data. With the introduction of these as the additional parameters, the updation of the mined data may be more specific.

5.4 CONCLUSION

The proposed method has discussed about the possible use of materialized view in incremental data mining where periodic update operations are required. Since this method is based on Apriori algorithm, only the frequencies of the attributes are considered for view materialization. The frequencies of the attributes have been calculated based on the historical transactions performed on the database. There may be other possibilities and factors for considering materialized view in incremental data mining. The factors may include the relevance of the output of a query with the existing mined data set and the space constraint of the mined data. With the introduction of these as the additional parameters, the updation of the mined data may be more specific. One major drawback of the proposed algorithm has been with regards to the number of transactions considered in each of the iterations. Here, eighteen new transactions have been considered in each of the iterations. In practice, the number of transactions may be much more than that. Further, in each of the new iterations, the participating set of transactions of the previous iteration has also been considered. So, if the ‘no_iter’ value if large, then in the final iteration, the number of transactions to be considered would have been huge. In this work, the Apriori algorithm has been invoked iteratively within MVG_AA ( ) algorithm. Moreover, MVG_AA ( ) has been invoked iteratively from MVG_IDM ( ). So, asymptotically, the Apriori algorithm is invoked in the polynomial time of ‘no_iter’. Additionally, the complexity of the Apriori algorithm can be controlled by modifying the minimum support value.

CHAPTER 6
CONTENT BASED IMAGE RETRIEVAL THROUGH MATERIALIZED VIEW

6.1 INTRODUCTION

A view or often termed as a logical view is a database object which temporarily stores the result of a query. On the other hand, a materialized view is also a database object but unlike a logical view, it physically stores the output of a query or even of a query set [2]. Many commercial database packages have, of late incorporated the generation of a materialized view and the use of the same [21][22], as described in the previous chapters as well. Materialized views help in eliminating the overhead involving expensive joins and aggregations that need to be performed for a large class of queries and offer significant improvements in query processing time, particularly for aggregation queries over tables of considerable size [23][52]. There are different approaches of identifying the optimal ways of storing data to be materialized. In general, the use of materialized view lies in the field of faster query processing and a methodical data analysis. Since most of the areas where materialized view is often used deal with text-based data, there are different ways or algorithms to do that. This chapter introduces a method of utilizing materialized view in image classification and retrieval which find applications in a wide variety of different domains. Because of the exponential growth in the field of multimedia applications, storing and retrieving the desired digital data in the form of images have become an area of concern, of late. The goal of the proposed method is to improve the way the images may be stored, retrieved based on their different classes. Image classification is the process of classifying images into one of two or more classes based on certain features extracted from the images. Retrieving images based on their defining features leads to the concept of Content Based Image Retrieval or CBIR which is the process of searching and retrieving images from a database or collection of images based on an image query. In this chapter, a method has been proposed to utilize materialized view in the domain of content based image retrieval. Here, the image query consists of the features of the images to be classified and they are compared against the features of the images stored in the form of materialized views. Instead of retrieving matching images as is the case in the traditional CBIR, the new images are added to the materialized view that contains the matching images.

So, the main objective of the proposed research work is to classify the unclassified images and determine in which materialized view the image should be placed. Any subsequent new images are classified with the materialized views updated with the previously classified images. In general, the classification technique consists of two phases – training and testing where in the training phase, the classifier is trained using a number of images whose classes are already known and identified by class labels. Once the training phase is complete, the testing phase is performed in which the now trained classifier is used to determine the class of new, unclassified images i.e. images whose class label is not known. Since classification involves a training phase involving labelled samples, it falls under the category of supervised learning. There are various methods that can be used for performing classification based on supervised learning. The basic method that has been used here in the proposed research work is support vector machine which is a widely used supervised machine learning algorithm. CBIR can be implemented in various ways. One of the most common ways of achieving that is through some defining image features like colour, shape and depth. This method described in this chapter mainly focuses on the colour-based approach, based on colour moments algorithm and shape-based approach, based on height and width ratio have been used to perform image retrieval and image classification.

6.2 SUPPORT VECTOR MACHINE

The methodology discussed in this chapter mainly depends on the approach followed by a supervised category of machine learning model called the Support Vector Machine or SVM [53]. A Support Vector Machine is a technique that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space [53][54]. It can classify both the linear as well as the nonlinear data. When the data are not linearly separable, kernel functions are used to transform the data to a higher-dimensional space to enable linear separation. There can be many kernel functions like linear kernel, polynomial kernel, Radial Basis Function or RBF kernel and sigmoid kernel [55]. The choice of a specific kernel function may depend on the characteristics of the data and the specific use case. Within the new dimension obtained, it searches for the linear optimal separating hyperplane or a decision boundary. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. An SVM finds this hyperplane using support vectors which are the training tuples and the margins that are defined by the support vectors.

Since an SVM is based on finding a hyperplane between two different data classes, it is a binary classifier, but can be augmented by various methods to classify data into additional number of classes.

6.3 PROPOSED METHODOLOGY

The main problem related to the research work is to classify the uncategorized images into one of two or three materialized views each containing images belonging to a particular category, i.e., a particular class. So, the proposed method can classify two different images and even three different images. To solve this problem, a classifier has been designed based on the features of a support vector machine which is trained using the images in the materialized views. Once the training phase is complete, the classifier accepts as input the image to be classified and then places it in the correct materialized view. The updated materialized view is then used to retrain the classifier before classifying the next uncategorized image and the process is repeated for the subsequent iterations. Image classification involving two classes and three classes both consist of two major phases – an initialization phase in which the training set of images are read and the training matrix is constructed using the features selected from these images, and a training and classification phase in which the classifier is trained and new images are classified using the trained classifier. So the method consists of the following sub-problems:

Feature extraction, classifier training and classification of new images. Each of these sub-problems is described below:

Feature extraction – This process involves extracting or selecting features from the images that will be used to train the classifier and then to classify new images after training. This was implemented in two different ways, at first; feature selection process that was designed was based on the intensity of the images. Later, the feature extraction process has been modified and instead of considering the intensity of the images, the shapes of the images have been considered and the latter process has given better results. In this research work, the shape of an image has been taken in the form the ratio of its height with its width. In the initial attempt of classifying the images based on their intensities, the color moments algorithm [56] was used to derive the features for each image. The features produced by this algorithm are based on the intensity distribution of the image. At first, the algorithm has converted the input image into the HSV (Hue, Saturation, Value) color space from the RGB (Red, Green, Blue) color space.

The algorithm has then proceeded to calculate the mean pixel intensity, the variance of the pixel intensities and the skewness of the pixel intensities for each channel, i.e., for Hue, Saturation and Value and this has been done for each image. Thus, for each image, this process has generated nine feature values which together have constituted the feature vector of an image. These feature values have been used to classify each image.

The pseudocode for the color moments algorithm is given below:

COLOR_MOMENTS ( )

{

Initialization: The feature vector to be zero

Input: The image whose feature vector is to be calculated

The input image is converted from the RGB color space to HSV color space

Extract three individual channels in the form of Hue, Saturation and Value

For each channel

Loop

The mean pixel intensity is calculated

The variance of the pixel intensity is calculated

The skewness of the pixel intensity is calculated

End Loop

The feature vector is stored with nine values obtained through the above iteration

The features vector is returned

}

Although the classification process by means of the color moments algorithm may produce satisfactory results, the process may not be optimal since it is possible for two classes of images to have similar intensity distributions, particularly if images are in grayscale. Thus, a second approach has been generated based on the shapes of the images. Although two different classes of images may share similar intensity distributions, their shapes should be significantly different in most instances. Thus, generating features based on the shapes of the principal object in the image would be more effective in performing image classification. Using the shape as the principal feature for classification would allow the classifier to be able to differentiate between different classes of images regardless of similar intensity distributions, if any, especially in the case of grayscale images as discussed above. Since the classifier can only work with numerical feature values, a numerical representation for the image shape would be required. Such a representation has been produced by taking the ratio of the height of the image shape to the width of the shape and this ratio has been considered to be the feature for each image. Thus, while the previous method required needed to store a nine-element feature vector for each image, in this method, only one feature value for each image needs to be stored, thus saving disk space as well. Since the image shape is also a better discriminator between images than intensity distribution, the classification accuracy has also been improved, as will be discussed in a later section. The following is the pseudocode for the shape based feature selection process:

EXTRACTION_SHAPE ( )

{

Input: The image whose feature vector is to be calculated

The input image is converted into a grey-level image

The points of interest are detected on the converted image

The most important, i.e., the most useful 100 points are identified on the detected points

The coordinates of the selected points are determined

x ← the x-coordinates of all the points

y ← the y-coordinates of all the points

height ← max(y) – min(y) /* max ( ) and min ( ) are tow function to respectively calculate the maximum and the minimum values from a list */

width ← max(x) - min(x)

ratio ← height / width

The value stored in ratio is returned

}

Classifier training – The classifier is trained using the image features of all the images in the training data set.

Classification of new images – The same features are extracted from the images to be classified in order to classify the images using the previously trained classifier.

The primary difference between the two-class classification and the three-class classification is that for the latter three training matrices are constructed and correspondingly, three different classifiers are trained. The class label for the new image is determined by taking the majority of the labels output by the three classifiers. The algorithm for the process is given below:

Step 1: Images present in the training set are read.

Step 2: The features for each of the images to be trained are calculated.

Step 3: The training matrix using the computed image features and associated class label is computed. The computation is done for each image.

Step 4: For the first iteration, the classifier is trained using the data obtained in the training matrix and for any subsequent iteration, the classifier is retrained with addition featured data, if any.

Step 5: The image to be classified is read.

Step 6: The features of the newly accepted image, as given in step 5 are calculated.

Step 7: The class label of the new image is computed after invoking the classifiers that have been created in step 3.

Step 8: The training matrix is augmented with the features and the class level of this new image. The augmentation is done for the incremental mining of the image features.

Step 9: If any more image is to be classified, then the control is transferred to step 4; otherwise the process is stopped.

In the above mentioned algorithm, the first three steps may be considered to be the initialization phase whereas the next steps may be identified as the training and classification phase. This algorithm has been designed with the help of the characteristics of supervised machine learning approach, specifically, support vector machine as already mentioned in the previous sections. The algorithm, named as IMG_CLASSIFICATION for image classification using the features of an SVM is given below. This algorithm classifies two different classes of images and hence this is dependent on two different materialized views, viz., MV1 and MV2. These two views already store the features of the images corresponding to two different classes.

IMG_CLASSIFICATION( )

{

Initialization: The feature vector to be empty and the list of labels to be empty

Input: Two materialized views MV1 and MV2

For each image I in MV1

Loop

The required features are selected using the feature selection process

The features are added to the feature vector

The label corresponding to MV1 is added to the list of labels

End Loop

The above loop iterated for each image in MV2 too.

The classifier is trained with the feature vector and the list of labels as parameters

For each image J to be classified

Loop

The required features are selected using the same selection process

The image J is classified using the selected features

Image J is added to either MV1 or MV2 depending on the features obtained

If J is classified to be in MV1

Then

MV1 is updated

Else

MV2 is updated

End If

The classifier is retrained using the newly updated materialized view

End Loop

}

Since a support vector machine is capable of only binary classification, to design an algorithm for more than two-class classification, the method of binary classification has to be used, i.e., a ternary classification has to be performed by means of a binary classifier. Since in a ternary classification, three classes are required, three different SVMs have been trained, one for each of the three pairs of classes possible among the three different classes. Thus, each of these SVMs is a binary classifier as before. Once these three classifiers have been trained, new images having three different types of attributes can be classified.

This is done for each image by attempting to classify it using each of the three classifiers. Each classifier would output the class which it has classified the new image into. To determine the correct class of the new image, the class which has been output by a majority of the classifiers has been chosen and in the present method of the ternary classification, a majority means two classes out of three classes. The following example demonstrates this:

Assuming that there are three classes C1, C2 and C3, three binary SVMs have been trained for classifying between C1 and C2, between C2 and C3 and finally between C3 and C1. Now, if an image that should be classified as belonging to class C1 is taken as an input, each of the three SVMs are executed on the unclassified image. The C1 versus C2 classifier would output class C1, the C3 versus C1 classifier will output class C1 and the C2 versus C3 classifier will output either C2 or C3 depending on which of these two classes the image is closer to. Since class C1 is the majority output (i.e., output by 2 of the 3 classifiers) class C1 is to be taken as the correct class of the image and this is the desired output. The scenario of generating three different outputs for three different binary classifiers is impracticable as explained below:

If for an image, the C1 versus C2 classifier outputs C1 and the C3 versus C1 classifier outputs C3, then there would be no majority output class if the C2 versus C3 classifier now outputs C2. But this is not possible, as the test image is closer to C1 than to C2 as determined by the first classifier, and also because it is closer to C3 than to C1 as determined by the second classifier, it logically follows that the image must be closer to C3 than to C2 and hence the C2 versus C3 classifier would definitely output C3 and thus the resultant class of the new image would be class C3.

The following is the pseudocode for three-class classification where MV1, MV2 and MV3 are three materialized views containing the required feature of three different image classes:

THREE_CLASS_CLASSIFICATION ( )

{

Initialization: Three feature vectors C1, C2 and C3 to be empty and three lists of labels L1, L2 and L3 to be empty

For each image I in MV1

Loop

The required features are selected using the feature selection process

The features are added to the feature vectors C1 and C2

The label corresponding to MV1 is added to the lists of labels L1 and L2

End Loop

For each image I in MV2

Loop

The required features are selected using the feature selection process

The features are added to the feature vectors C3 and C1

The label corresponding to MV2 is added to the lists of labels L3 and L1

End Loop

For each image I in MV3

Loop

The required features are selected using the feature selection process

The features are added to the feature vectors C2 and C3

The label corresponding to MV3 is added to the lists of labels L2 and L3

End Loop

The C1 versus C2 classifier is trained with the feature vectors C1 and C2 and the lists of labels L1 and L2 as inputs

The C3 versus C1 classifier is trained with the feature vectors C3 and C1 and the lists of labels L3 and L1 as inputs

The C2 versus C3 classifier is trained with the feature vectors C2 and C3 and the lists of labels L2 and L3 as inputs

For each image J to be classified

Loop

The required features are selected using the same selection process

The image J is classified using the selected features using each of the three classifiers.

From the three outputs produced by the three classifiers, the output that occurs twice is selected

Image J is added to MV1, MV2 or MV3 depending on the output obtained

The feature vectors of MV1, MV2 or MV3 are updated using the features of the newly classified image

The classifiers are retrained using the newly updated materialized view

End Loop

}

6.4 RESULTS AND ANALYSIS

All the algorithms as mentioned in the previous section have been implemented and the application has been tested with standard data sets. The system that was used to implement the algorithm had Intel Core 2 Duo processor with a speed of 2.0 Ghz. The operating system used was Windows 7 and the software tool used to write the code was MATLAB 2015b. Though the application can directly executed in a system having any version of Windows operating system above the XP version and the runtime environment that is required is MATLAB runtime. The initial testing for the two-class image classifier made use of the color moments feature selection process. To test it, the images of elephants and pandas from a dataset that was accessible at [57] were used. Some of the used images are given from figure 6.1 to figure 6.4.

Figure 6.1: Four sample images of pandas used for the training purpose

Figure 6.2: Four sample images of elephants used for the training purpose

Figure 6.3: Four sample images of pandas used for the testing purpose

Figure 6.4: Four sample images of elephants used for the testing purpose

In the second approach of the image classification as mentioned in the previous section, the method was based on the image shape and accordingly, testing was done with the new process using the same set of images as above and the latter process gave better results. Having successfully tested the classifier on animal images, leaf classification was conducted. The leaf images were taken from the dataset as mentioned in [58]. The leaf images were mainly used for the ternary classifier as proposed in the previous section. To do this, three different types of leaves were taken and to train the classifier, 80% of images under each category were taken. After training the classifier, rest of the images were used for the testing purpose. Since the ternary classifier depends on the binary classifier, to train the former, the latter were automatically tested. So, as far as the binary classifier is concerned, training and testing were conducted for three different pairs of image classes. Some of the images used for the purpose are given next from figure 6.5 to 6.10.

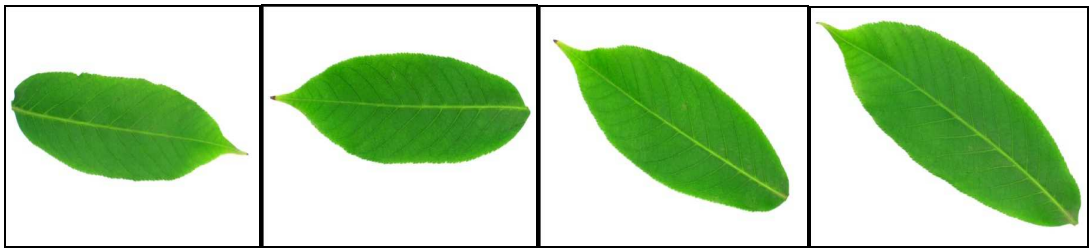

Figure 6.5: Four sample images of class 1 leaves used for the training purpose

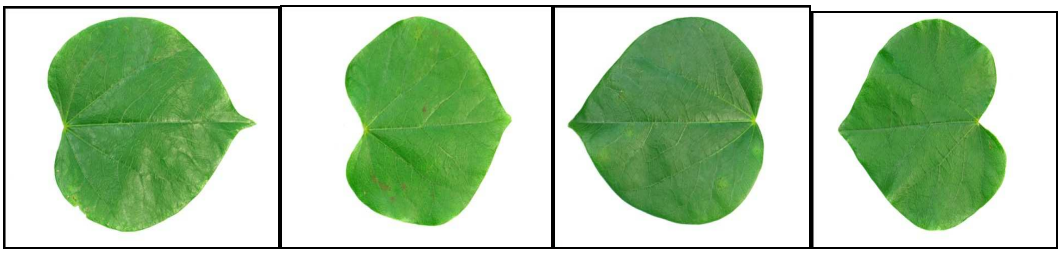

Figure 6.6: Four sample images of class 2 leaves used for the training purpose

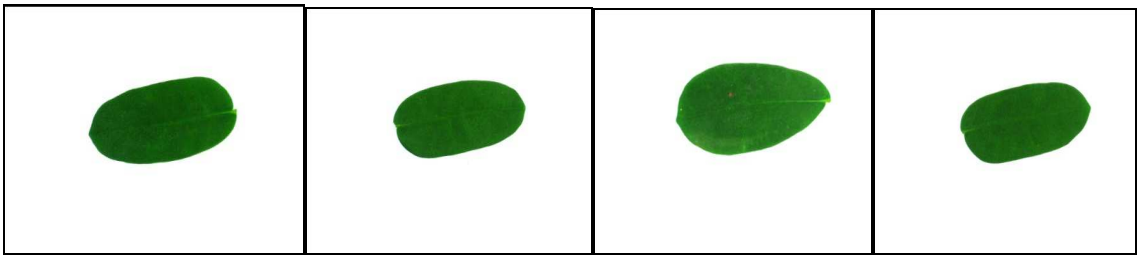

Figure 6.7: Four sample images of class 3 leaves used for the training purpose

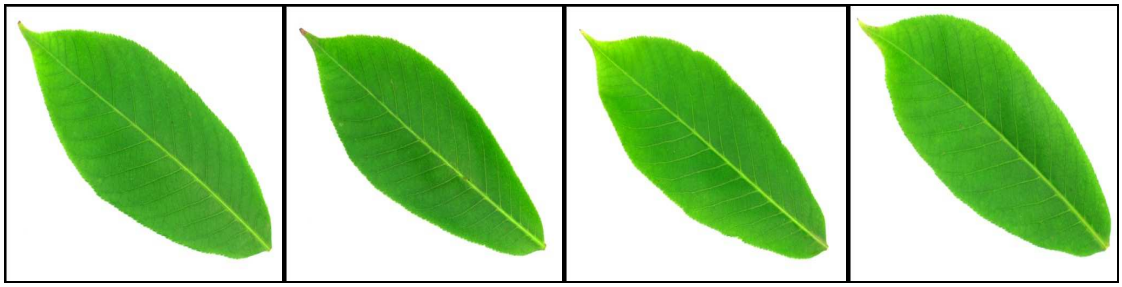

Figure 6.8: Four sample images of class 1 leaves used for the testing purpose

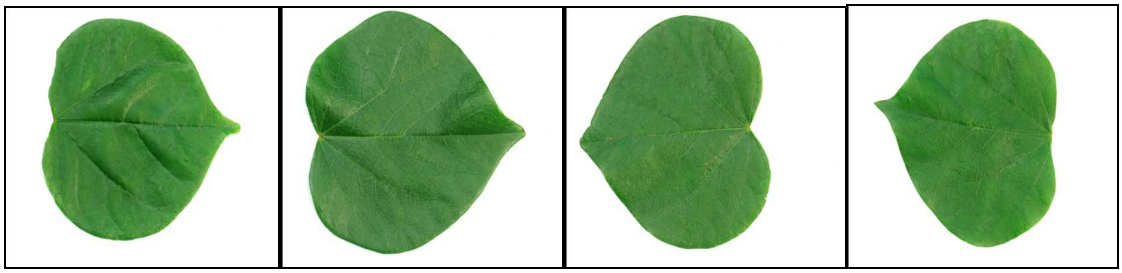

Figure 6.9: Four sample images of class 2 leaves used for the testing purpose

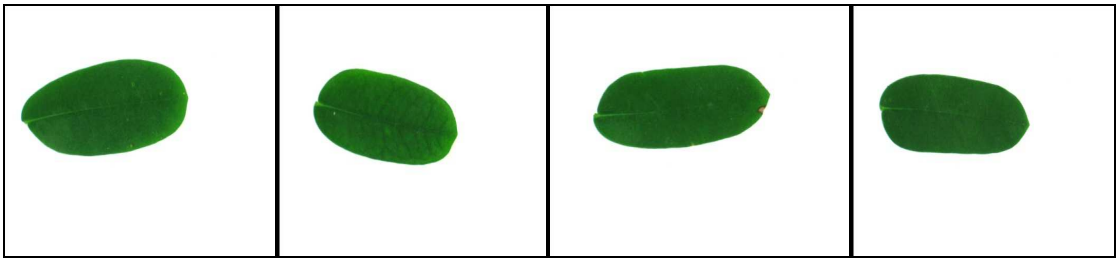

Figure 6.10: Four sample images of class 3 leaves used for the testing purpose

With all the above mentioned datasets, the results that were obtained after implementing algorithms as mentioned in the previous section are given next. Table 6.1 shows only one snapshot of the result containing thirty images obtained after applying COLOR_MOMENTS ( ) with IMG_CLASSIFICATION ( ) and EXTRACTION_SHAPE ( ) with IMG_CLASSIFICATION ( ) on the animal images. Table 6.1 also shows a comparative study of the outputs obtained when binary classification was implemented after considering the features related to the colors of the images as well as the shapes of the images. All the file names have been shown in table 6.1 are from that dataset. The algorithms COLOR_MOMENTS ( ) and IMG_CLASSIFICATION ( ) were applied on the animal images from the dataset obtained from [57] to consider color related features and the algorithms EXTRACTION_SHAPE ( ) and IMG_CLASSIFICATION( ) were used to the shape related features of the images.

Test No.	File Name	Class	color_moments( ) and img_classification( )	extraction_shape( ) and img_classification( )	
			Classified correctly (Yes/No)?	Height/Width ratio	Classified correctly (Yes/No)?
1	image_0013	Panda	No	1.04	Yes
2	image_0032	Panda	No	0.691	No
3	image_0033	Panda	No	0.632	No
4	image_0035	Panda	No	1.52	Yes
5	image_0036	Panda	Yes	1.07	Yes
6	image_0037	Panda	Yes	0.753	No
7	image_0038	Panda	Yes	1.03	Yes
8	image_0038_2	Elephant	No	0.723	Yes
9	image_0039	Elephant	Yes	0.676	Yes
10	image_0040	Elephant	No	0.614	Yes
11	image_0041	Elephant	Yes	0.807	Yes
12	image_0042	Elephant	Yes	0.877	Yes
13	image_0043	Elephant	Yes	0.723	Yes
14	image_0044	Elephant	Yes	0.814	Yes
15	image_0045	Elephant	Yes	0.661	Yes
16	image_0046	Elephant	No	0.912	Yes
17	image_0047	Elephant	Yes	0.849	Yes
18	image_0048	Elephant	Yes	0.858	Yes
19	image_0049	Elephant	Yes	0.690	Yes
20	image_0050	Elephant	Yes	0.690	Yes
21	image_0051	Elephant	Yes	0.678	Yes
22	image_0052	Elephant	Yes	0.727	Yes
23	image_0053	Elephant	Yes	0.937	Yes
24	image_0054	Elephant	Yes	0.956	Yes
25	image_0055	Elephant	Yes	0.794	Yes
26	image_0056	Elephant	Yes	0.764	Yes
27	image_0057	Elephant	No	1.13	No
28	image_0058	Elephant	Yes	0.646	Yes
29	image_0059	Elephant	Yes	0.871	Yes
30	image_0062	Elephant	No	0.735	Yes

Table 6.1: Animal image classification

The classification accuracy after considering color related features was 70% whereas the same for the shape related features was 86.7%, which is a reasonable improvement. Figure 6.11 gives a graphical comparison of the outputs obtained.

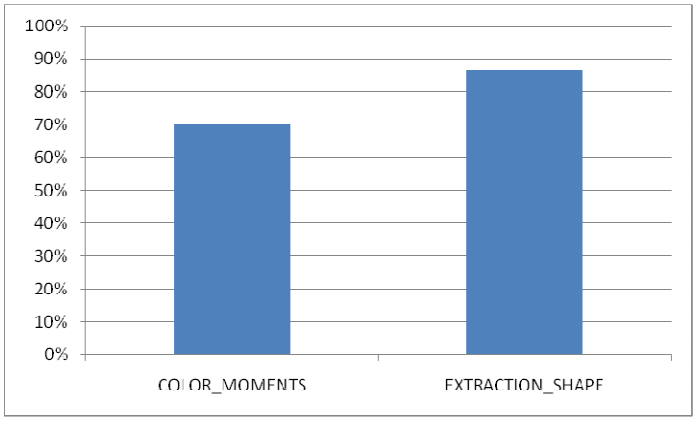

Figure 6.11: A comparative study of the outputs obtained from two different methods shown in table 6.1

Test No.	File Name	Class	color_moments( ) and img_classification( )	extraction_shape( ) and img_classification( )	
			Classified correctly (Yes/No)?	Height/Width ratio	Classified correctly
1	1110	Class 1	Yes	0.701	Yes
2	1111	Class 1	Yes	0.707	Yes
3	1112	Class 1	Yes	0.756	Yes
4	1113	Class 1	Yes	0.714	Yes
5	1114	Class 1	Yes	0.768	Yes
6	1115	Class 1	Yes	0.801	Yes
7	1116	Class 1	Yes	0.729	Yes
8	1117	Class 1	Yes	0.731	Yes
9	1118	Class 1	Yes	0.674	Yes
10	1119	Class 1	Yes	0.691	Yes
11	1120	Class 1	Yes	0.681	Yes
12	1121	Class 1	Yes	0.674	Yes
13	1122	Class 1	Yes	0.407	Yes
14	1181	Class 2	Yes	0.891	Yes
15	1182	Class 2	Yes	0.892	Yes
16	1183	Class 2	Yes	0.986	Yes
17	1184	Class 2	Yes	0.967	Yes
18	1185	Class 2	Yes	1.04	Yes
19	1186	Class 2	Yes	0.919	Yes
20	1187	Class 2	Yes	1.06	Yes
21	1188	Class 2	Yes	0.918	Yes
22	1189	Class 2	Yes	0.944	Yes
23	1190	Class 2	Yes	0.960	Yes
24	1191	Class 2	Yes	0.970	Yes
25	1192	Class 2	Yes	0.931	Yes
26	1193	Class 2	Yes	0.970	Yes
27	1194	Class 2	Yes	0.971	Yes

Table 6.2: Leaf image classification

Test No.	File Name	Class	Height/Width ratio	Classified correctly (Yes/No)?
1	1110	Class 1	0.701	Yes
2	1111	Class 1	0.707	Yes
3	1112	Class 1	0.756	Yes
4	1113	Class 1	0.714	Yes
5	1114	Class 1	0.768	Yes
6	1115	Class 1	0.801	Yes
7	1116	Class 1	0.729	Yes
8	1117	Class 1	0.731	Yes
9	1118	Class 1	0.674	Yes
10	1119	Class 1	0.691	Yes
11	1120	Class 1	0.681	Yes
12	1121	Class 1	0.674	Yes
13	1122	Class 1	0.407	No
14	1181	Class 2	0.891	Yes
15	1182	Class 2	0.892	Yes
16	1183	Class 2	0.986	Yes
17	1184	Class 2	0.967	Yes
18	1185	Class 2	1.04	Yes
19	1186	Class 2	0.919	Yes
20	1187	Class 2	1.06	Yes
21	1188	Class 2	0.918	Yes
22	1189	Class 2	0.944	Yes
23	1190	Class 2	0.960	Yes
24	1191	Class 2	0.970	Yes
25	1192	Class 2	0.931	Yes
26	1193	Class 2	0.970	Yes
27	1194	Class 2	0.971	Yes
28	1251	Class 3	0.501	Yes
29	1252	Class 3	0.554	Yes
30	1253	Class 3	0.446	Yes
31	1254	Class 3	0.450	Yes
32	1255	Class 3	0.536	Yes
33	1256	Class 3	0.456	Yes
34	1257	Class 3	0.476	Yes
35	1258	Class 3	0.508	Yes
36	1259	Class 3	0.454	Yes
37	1260	Class 3	0.546	Yes
38	1261	Class 3	0.528	Yes

Table 6.3: Output of the ternary classification

Table 6.2 shows a snapshot of the comparative study of the outputs obtained when binary classification was implemented after considering the features related to the colors of the images as well as the shapes of the images. The algorithms COLOR_MOMENTS ( ) and IMG_CLASSIFICATION ( ) were applied on the leaf images from the dataset obtained from [58] to consider color related features and the algorithms EXTRACTION_SHAPE ( ) and IMG_CLASSIFICATION( ) were used to the shape related features of the images. All the file names have been shown in table 6.2 are from that dataset. Incidentally, the classification accuracies in both the cases here were 100%. So, in this case, no graphical comparison needs to be shown.

On the other hand, table 6.3 distinguishes images belonging to one of the three classes – class 1, class 2 and class 3 from the dataset obtained from [58]. So, this table shows an instance of the result obtained after applying ternary classification based on the shapes of the images. As shown in table 6.3, in the ternary classification process, there is only one case where the classifier has failed to correctly classify the image and the accuracy of the result is 97.37%.

In general, by going through the results shown in the above tables, it may be inferred that shape-based classification process performs in a better way than the algorithm based on the color features of the images and the ternary classification process which in turn invokes the binary classifier, gives a satisfactory result. Moreover, as only some defining features against a particular image was required to be stored in the above-mentioned methodologies, there has been a significant improvement in terms of amount of memory space used in storing an image . So for classification and retrieval purpose, an image of size in the order of kilobytes or megabytes has been stored in a memory space whose size is in the order of bytes. The number of classes can be increased by incorporating further processing of the binary classifier and the resultant steps will also increase. The classifier gave more accurate result when shape-based features were considered instead of the color-based features of the images. Few constraints are to be maintained in the shape-based feature selection. Among them, one point is the orientation of an image. The classification accuracy may be adversely affected if the orientation of the images changes. Moreover, to reduce the heterogeneity of the image orientation, some established image registration algorithms may be used before actually using the images in the classification of the images.

6.5 CONCLUSION

The image classifier system based on materialized view as proposed in this paper stores only the defining points or characteristics of the images instead of storing entire images. This has been the major improvement over the existing systems of image classification. This proposed classifier has been able to classify images belonging to three different classes. The number of classes can be increased by incorporating further processing of the binary classifier and the resultant steps will also increase. The classifier gave more accurate result when shape-based features were considered instead of the color-based features of the images. Few constraints are to be maintained in the shape-based feature selection. Among them, one point is the orientation of an image. The classification accuracy may be adversely affected if the orientation of the images changes. Moreover, to reduce the heterogeneity of the image orientation, some established image registration algorithms may be used before actually using the images in the classification of the images. As a future prospect, these things may be taken into consideration for proposing better algorithms to develop image classifiers based on materialized view.

CHAPTER 7

CONCLUSION

The thesis initially discusses about the materialized views with their generation methodologies where two different types of algorithms have been highlighted. These are genetic algorithm and apriori algorithm. It also addresses the issues related to two different application areas of materialized views. One application area is related to incremental data mining another is in connection with content based image retrieval. The salient features of the these have been addressed in the first sub section whereas the second sub section highlights the finding of the thesis. The final sub section gives a detailed discussion on the limitations of the work described in the thesis with some future scopes of work.

7.1 Summary of the work

The study has mainly two aspects – the first one is associated with the generation process of the materialized view and the second one gives a detailed discussion on the application areas. Different sub sections under this section summarize the novelty of the work discussed in the thesis.

7.1.1 Materialized View generation using Genetic algorithm

The first category of algorithm used for materialized view generation was genetic algorithm and two different approaches have been presented in chapters 2 and 3. Different researchers had earlier done work in the said domain and also proved that the materialized view generation process was an NP-hard problem but the presented work could generate the views in polynomial time and exponential time. Few earlier work had either used space or time to generate materialized views in a multidimensional space but the first work discussed in the thesis considered both the parameters and hence could generate views in an optimal way. The second work using genetic algorithm considered the frequency of view generation as the defining parameter. Both of these methodologies could generate materialized views in optimal time and hence could solve the problem of Materialized View Selection (MVS).

7.1.2 Materialized View generation using Apriori algorithm

A different approach to generate materialized views was also explored and that was based on a seminal algorithm called Apriori where the major aspect considered for view materialization was the frequency of occurrences of the records under consideration. No prior research work had used this approach for materialized view generation. The major factor achieved through this approach was the computational time complexity for view materialization. Since Apriori algorithm could itself generate the desired outcome in polynomial time, its use in the problem under consideration could also be possible in polynomial time and this was explained in chapter 4.

7.1.3 Application of materialized view in incremental data mining

The second part of the research work depicted in the thesis was dedicated to the different areas of applications of materialized views. A lot of applications have already been highlighted in the previously published research work but no specific application was described in the field of incremental data mining. This area of application was described and analyzed in the thesis. With an iterative approach of caching the outcomes of frequently executed queries in the form of materialized views, an effective of incremental data mining could be achieved. Chapter 5 elaborates this application with supporting examples.

7.1.4 Application of materialized view in content based image retrieval

Image handling is now being considered as an area of research which has manifold applications in the domains of medical science, computer vision, pattern recognition and others. This primarily includes retrieval of images and thereafter processing according to the requirements. Image retrieval is a technique which requires an entire image and therefore a large amount of memory space may be required when an application needs to process a huge collection of images. Content based image retrieval is a relatively newer approach where some visual contents of an image need to be processed for retrieving an image without storing the entire image and hence requires much less amount of memory space. In chapter 6, the thesis has explored an area of utilizing materialized views in the field of content based image retrieval by considering the ratio of the height and the width of the images. This approach could not only save space for storing the details of the images but also could substantially reduce the required time to retrieve the images. The work was further expanded in image classification as described in the same chapter.

7.2 ANSWERS TO THE RESEARCH QUESTIONS

The basis of the research work described in this thesis is a database object called materialized view. The initial questions, as put forward in section 1.6 were related to the different approaches of generating the same. As described in sections 3.3 and 3.4 of chapter 3, genetic algorithms were successfully used for materialized view generation. Hence the methodologies described in these two chapters answer the first research question. The answer to the second research question lies in the process described in chapter 4 where the Boolean Association rules and frequent itemset generation process of Apriori algorithm were utilized for view materialization in a novel way. Table 7.1 summarizes the parameters considered for view materialization using different approaches as described in this thesis.

Approach	**Parameters considered**	**Described in which chapter**
Genetic Algorithm	Space and time of the generated views	Section 3.3 of chapter 3
Genetic Algorithm	Frequency of view generation	Section 3.4 of chapter 3
Apriori Algorithm	Frequency of occurrences of each attribute in the participating transactions	Chapter 4

Table 7.1: A summary of the different approaches considered for view materialization

The next questions, mentioned in section 1.6 were related to few possible application areas of materialized views. There have been various applications of materialized views in query processing and in the field of knowledge discovery from the database. One of the challenging application areas has been in the field of incremental data mining. Chapter 5 elaborates the use of materialized view for the said application and hence this answers the third question of the research objective.

The answer to the fourth question under the research objective has been elaborated through another application area in the field of content based image retrieval. Chapter 6 elucidates an approach of image retrieval process based on some defining features of an image without storing the entire image.

This effectively saves the storage space and at the same time, reduces the image fetch operation. Table 7.2 summarizes this discussion related to the initial research questions pertaining to the possible application areas of materialized views.

Application Domain	Described in which chapter
Incremental Data Mining	Chapter 5
Content Based Image Retrieval	Chapter 6

Table 7.2: A summary of the different application areas of materialized view

7.3 SCOPE OF FUTURE DEVELOPMENT

Each chapter entails a scope for further development in the field of data processing, handling, retrieving and analysis.

As far as the use of genetic algorithm is concerned in materialized view generation, the work can further be expanded to include a constraint in selecting a particular set of views to be materialized instead of considering all possible views that can be generated. This constraint can be a single view generation cost which can be merged with the view size that is already considered in the lattice structure. Moreover, there is a further scope of introducing the frequency domain with this. In the frequency domain, the frequency of occurrence of individual queries will have to be considered. Another challenging direction of future work aims at addressing the view selection problem in a distributed setting. Randomized algorithms can be applied to complex problems dealing with large or even unlimited search spaces.

The next task has been view materialization using Apriori algorithm. 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. When materialized view was used for incremental data mining, only frequency of occurrences of an item or an itemset was considered to be the defining parameter. There may be other possibilities and factors for considering materialized view in incremental data mining. The factors may include the relevance of the output of a query with the existing mined data set and the space constraint of the mined data. With the introduction of these as the additional parameters the updation of the mined data may be more specific. Some factors may also be added in future for the use of materialized view in content based image retrieval. Since the proposed classifier gave more accurate result when only the shape-based features were considered instead of the colour-based features of the images, few constraints are to be maintained in the shape-based feature selection. Among them, one point is the orientation of an image. The classification accuracy may be adversely affected if the orientation of the images changes. Moreover, to reduce the heterogeneity of the image orientation, some established image registration algorithms may be used before actually using the images in the classification of the images. As a future prospect, these things may be taken into consideration for proposing better algorithms to develop image classifiers based on materialized view.

References

1. S. Sen, D. Datta, and N. Chaki, “An Architecture to Maintain Materialized View in Cloud Computing Environment”, International Conference on Computing Sciences, Turing100, 2012, pp. 361 – 366.

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

3. https://docs.oracle.com/cd/B14117_01/server.101/b10732/repmview.htm, last accessed on 01.01.2025.

4. Thomas P. Nadeau and Toby J. Teorey, “Achieving Scalability in OLAP Materialized View Selection”, In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, 2002, pp. 28 – 34.

5. J.H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975, Michigan; re-issued by MIT Press, 1992.

6. D.E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley, Reading, Massachusetts.

7. Black, T., Fogel, D. B. and Michalewicz, Z., “Handbook of Evolutionary Computation”, Oxford University Press, Oxford, 1997.

8. Siddharth Shah et al, “Incremental Mining of Association Rules: A Survey”, International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012, 4071 – 4074.

9. Yadav, Ashwani & R.Roy, & Yadav, Vaishali & Kumar, Archek, “Survey on Content-based Image Retrieval and Texture Analysis with Applications”, International Journal of Signal Processing, Image Processing and Pattern Recognition, 2014, pp. 41 – 50..

10. Kanithan, S., Vignesh, N.A., Karthick SA, “Visual Object Segmentation Improvement Using Deep Convolutional Neural Networks”, In: Kumar, A., Jain, R., Vairamani, A.D., Nayyar, A. (eds) Object Tracking Technology. Contributions to Environmental Sciences & Innovative Business Technology, Springer, Singapore. 2023, pp. 63 – 85.

11. E. Baralis, S. Paraboschi and E. Teniente, “Materialized view selection in a multidimensional database”, In Proceeding of the 23rd International Conference on Very Large Data Bases, 1997, pp. 156 – 165.

12. S. Chandrasekaran and M.J. Franklin, “Streaming queries over Streaming Data”, In Proceedings of the 28th International Conference on Very Large Data Bases, 2002, pp. 203 – 214.

13. V. Harinarayan, A. Rajaraman and J. Ullman, “Implementing data cubes efficiently”, In Proceedings of ACM SIGMOD International Conference on Management of Data, 1996, pp. 205 – 216.

14. H. Gupta and I. Mumick, “Selection of Views to Materialize in a Data Warehouse”, IEEE Transactions on Knowledge and Data Engineering, 17(1), 2005, pp. 24 – 43.

15. T.V. Vijay Kumar and A. Ghosal, “Greedy Selection of Materialized Views”, International Journal of Communication Technology, Volume 1, Number 1, 2009, pp. 156 – 172.

16. A.P. Bhagat and B.R. Harle, “Materialized View Management in Peer to Peer Environment”, In Proceedings of International Conference & Workshop on Emerging Trends in Technology, 2011, pp. 480 – 484.

17. Thomas P. Nadeau and Toby J. Teorey, “Achieving Scalability in OLAP Materialized View Selection”, In Proceedings of the 5th ACM International workshop on Data Warehousing and OLAP, 2002, pp. 28 – 34.

18. Jonathan Goldstein and Per-Åke Larson, “Optimizing Queries Using Materialized Views: A Practical, Scalable Solution”, In Proceedings of the ACM SIGMOD International Conference on Management of Data, 2001, pp. 331 – 342.

19. S. Chaudhuri, S. Krishnamurthy, S. Potamianos and K. Shim, “Optimizing Queries with Materialized Views”, In Proceedings of the International Conference on Data Engineering, 1995, pp. 190 – 200.

20. C. Zhu, Q. Zhu and C. Zuzarte, “Efficient processing of monotonic linear progressive queries via dynamic materialized views”, In Proceedings of Conference of Center for Advanced Studies on Collaborative Research, 2010, pp. 224 – 237.

21. R.G. Bello, K. Dias, J. Feenan, J. Finnerty, W.D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin, “Materialized Views in Oracle”, In Proceedings of the 24th International Conference on Very Large Data Bases, 1998, pp. 659 – 664.

22. Docs.Oracle.com, “Materialized View Concepts and Architecture”, 2002, available: https://docs.oracle.com/cd/B10501_01/server.920/a96567/repmview.htm, accessed: 17 – April - 2023.

23. T.V. Vijay Kumar and Santosh Kumar, “Materialized View Selection using Genetic Algorithm”, In Proceedings of the 5th International Conference on Contemporary Computing, August, 2012, pp. 225 – 237.

24. J.T. Horng, Y.J. Chang, B.J. Liu, and C.Y. Kao, “Materialized view selection using genetic algorithms in a data warehouse system”, In Proceedings of the World Congress on Evolutionary Computation, 1999, pp. 2221 – 2227.

25. C. Zhang, X. Yao and J. Yang, “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, 2001, pp. 282 – 294.

26. J.T. Horng,, Y.J. Chang and B.J. Liu, “Applying evolutionary algorithms to materialized view selection in a data warehouse”, Soft Computing, 2003, Volume 7, Issue 4, pp.574 – 581.

27. W.Y. Lin and I.C. Kuo, “A genetic selection algorithm for OLAP data cubes”, Knowledge and Information Systems, Volume 6, Issue 1, 2004, pp. 83 – 102.

28. Z. Wang and D. Zhang, “Optimal genetic view selection algorithm under space constraint”, International Journal of Information Technology, Volume 11, Number 5, 2005, pp. 44 – 51.

29. X. Yu, Y.X. Yao, C. Choi and G. Gou, “Materialized view selection as constrained evolutionary optimization”, IEEE Transactions on Systems, Man, and Cybernetics, Volume 33, Number 4, 2003, pp. 458 – 467.

30. M.Lee and J. Hammer, “Speeding up materialized view selection in data warehouses using a randomized algorithm”, International Journal of Cooperative Information Systems, Volume 10, Number 3, 2001, pp. 327 – 353.

31. H. Mistry, P. Roy, S. Sudarshan, and K. Ramamritham, “Materialized view selection and maintenance using multi-query optimization”, In Proceedings of ACM SIGMOD International Conference on Management of Data, 2001, pp. 307 – 318.

32. R. Agrawal, T. Imielinski and A. Swami, “Mining association rules between sets of items in large databases”, In Proceedings of the ACM SIGMOD Conference on Management of Data, 1993, pp. 207 – 216.

33. D. W.-L. Cheung, J. Han, V. Ng, and C. Y. Wong, “Maintenance of discovered association rules in large databases: An incremental updating technique”, In Proceedings of the 12th International Conference on Data Engineering, 1996, pp. 106 – 114.

34. S. Thomas, S. Bodagala, K. Alsabti and S. Ranka, “An efficient algorithm for the incremental updation of association rules in large databases”, In Proceedings of the International Conference on Knowledge Discovery in Databases, 1997, pp. 263 – 266.

35. M. Parthasarathy, J. Zaki, M. Ogihara and S. Dwarkadas, “Incremental and interactive sequence mining”, In Proceedings of the ACM CIKM Conference, 1999, pp. 251 – 258.

36. M. Ali, A. Adnan, M. Saqib and Zahidullah, “Content based image retrieval (CBIR) using materialized views”, In Proceedings of the International Conference of Computer Science and Information Technology, 2011, pp. 116 – 119.

37. R. Chaudhuri and A. M. Patil, “Content based image retrieval using color and shape features”, IJAREEIE, Volume 1, Issue 5, 2012, pp. 386 – 392.

38. A. Jain, R. Muthuganapathy and K. Ramani, “Content based image retrieval using shape and depth from an engineering database”, In Proceedings of the 3rd International Conference on Advances in Visual Computing, 2007 – Volume Part II, pp. 255 – 264.

39. N. A. C. Hussin, N. Jamil, S. Nordin, K. Awang, “Geometrical-Invariant grid-based colour moment (GBCM) for plant identification”, Advanced Science Letters, Volume 4, Issue 10 – 11, 2014, pp. 1914 – 1917.

40. Zhou H, Yuan Y and Sadka A.H., ”Application of semantic features in face recognition”, Pattern Recognition, Volume 41, Issue 10, 2008, pp. 3251 – 3256.

41. Eakins J. and Graham M., “Content based Image Retrieval”, A report to the JISC Technology Applications program, University of Northumbria, Newcastle, 1999.

42. Jin J., Kurniawati R., Xu G. and Bai X., “Using browsing to improve content based image retrieval”, Journal of Visual Communication and Image Representation, 2001, pp. 123 – 135.

43. Rajan, M., Parameswaran, L. “Key frame extraction algorithm for surveillance videos using an evolutionary approach”, Sci Rep 15, 536 (2025). https://doi.org/10.1038/s41598-024-84324-0.

44. Mohania, M., Samtani, S., Roddick, J. and Kambayashi, Y., “Advances and Research Directions in Data Warehousing Technology”, Australian Journal of Information Systems, Volume 7, Number 1, 1999, pp. 41 – 59.

45. Sastry, K., Goldberg, D., Kendall, G., “Search Methodologies, Introductory Tutorials in Optimization and Decision Support Techniques”, Springer US, 2005.

46. Eiben, A.E., Smith, J.E., “Introduction to Evolutionary Computing”, Springer-Verlag, 2003, pp. 42 – 43.

47. R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”, In Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp. 487 – 499.

48. T. Morzy, M. Wojciechowski and M. Zakrzewicz, “Materialized Data Mining Views”, Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery, 2000, pp. 65 – 74.

52. S. Agrawal, S. Chaudhuri and V. R. Narasayya, “Automated Selection of Materialized Views and Indexes in SQL Databases”, In Proceedings of VLDB, 2000, pp. 496 – 505.

53. J. Nayek, B. Nayek and H. S. Behera, “A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications and Challenges”, International Journal of Database Theory and Application, Volume 8, Number 1, 2015, pp. 169 – 186.

54. Y. Zhan and D. Shen, “Design efficient support vector machine for fast classification”, Pattern Recognition, Volume 38, Issue 1, 2005, pp. 157 – 161.

55. F. Lauer and G. Bloch, “Incorporating prior knowledge in support vector machines for classification: A review”, Neurocomputing, Volume 71, Issues 7 – 9, 2008, pp. 1578 – 1594.

56. N. A. C. Hussin, N. Jamil, S. Nordin, K. Awang, “Geometrical-Invariant grid-based colour moment (GBCM) for plant identification”, Advanced Science Letters, Volume 4, Issues 10 – 11, 2014, pp. 1914 – 1917.

57. L. Fei-Fei, R. Fergus and P. Perona, “Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories”, IEEE.CVPR 2004, Workshop on Generative-Model Based Vision, retrieved from http://www.vision.caltech.edu/datasets/, accessed: 18 – June - 2023.

58. Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu- Xuan Wang, Yi-Fan Chang and Chiao-Liang Shiang, “A Leaf Recognition Algorithm for Plant classification Using Probabilistic Neural Network”, IEEE 7th International Symposium on Signal Processing and Information Technology, Cairo, Egypt, retrieved from https://sourceforge.net/projects/flavia/, accessed: 03 – November - 2023.

59. Kotidis Y., Roussopoulos N., "DynaMat: A Dynamic View Management System for Data Warehouses", ACM SIGMOD Record, Volume 28, Issue 2, 1999, pp. 371 – 382.

60. Shukla A., Deshpande P.M., Naughton J.F., "Materialized View Selection for Multidimensional Datasets", In Proceedings of the 24th International Conference on Very Large Data Bases, 1998, pp. 488 – 499.

61. Zhou H, Yuan Y, Sadka A.H., "Application of semantic features in face recognition", Pattern Recognition 41(10), 2008, pp. 3251 – 3256.

62. Pentland A, Picard R, Scaroff S., "Photobook: Content-based manipulation for image databases", International Journal of Computer Vision, Volume 18, 1996, pp. 233 – 254.

63. Jain AK, Vailaya A., "Image retrieval using color and shape", Pattern Recognition, Volume 29, Issue 8, 1996, pp. 1233 – 1244.

64. Wojciechowski M., Zakrzewicz M., "Itemset Materializing for Fast Mining of Association Rules", In Proceedings of the 2nd Conference on Advances in Databases and Information Systems, 1998, pp. 284 – 295.

65. R. Agrawal, T. Imielinski, A. Swami., "Database mining: A performance perspective", IEEE Bansactions on Knowledge and Data Engineering, Special Issue on Learning and Discovery in KnowledgeBased Databases, 1993, pp. 914 – 925.

66. Morzy, T., Wojciechowski, M., Zakrzewicz, M., "Data Mining Support in Database Management Systems", In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds) Data Warehousing and Knowledge Discovery, 2000, Lecture Notes in Computer Science, Volume 1874, Springer, Berlin, Heidelberg.

67. H. Mannila, H. Toivonen, and A. I. Verkamo, "Efficient algorithms for discovering association", In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, 1994, pp. 181 – 192.

68. G.K.Y.Chan, Q.Li, L.Feng, “Design and selection of materialized views in a data warehousing environment: A case study”, In Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP, 1999, pp. 42 – 47.

69. P.P. Karde and V.M.Thakare, “Selection & Maintenance of Materialized View and Its Application for Fast Query Processing: A Survey”, International Journal of Computer Science & Engineering Survey, Volume 1, Number 2, 2010, pp. 16 – 29.

70. Chaudhuri, S. & Dayal, U., “An Overview of Data Warehousing and OLAP Technology”, In ACM Sigmod Record, Volume 26, Issue 1, 1997, pp. 65 – 74.

71. Czejdo, Bogdan, Morzy, Mikolaj, Wojciechowski, Marek and Zakrzewicz, Maciej, “Materilaized View in Data Mining”, Proceedings of the 13th International Workshop on Database and Expert Systems Applications, 2002, pp. 827 – 831.

ADDENDUM

Question 1: What specific challenges in materialized view selection motivated the use of Genetic Algorithms in your approach?

Answer: The algorithms that have been proposed in connection with the view materialization have mostly worked on the selection of the proper data coming from different queries executed on the data set or stored in logical views. From these algorithms, the greedy ones have been proved to be more successful in efficiently identifying the list of queries to be considered for the formation of materialized views. But these algorithms have specifically not tested for a large search space. Since genetic algorithm is a widely used evolutionary technique suitable for getting an optimum set of solutions from a large search space because of its involvement in utilizing multiple generations, this algorithm has been chosen in the research work.

Question 2: How does your View Materialization with Soft Computation (VMSC) algorithm improve over the previous method [23]? Explain with respect to fitness function and time/space trade-offs.

Answer: The algorithm that was proposed in [23] had considered the space constraint only in the form of the view size in connection with the views considered for selection in the chromosomes and ultimately used for view materialization. On the other hand, the present algorithm, named as View Materialization with Soft Computation or VMSC has considered both the space as well as the time of view selection for chromosome formation which has led to view materialization. On considering this additional parameter, a significant improvement was noticed on the results obtained. A comparison on the outputs obtained in the research work depicted in [23] and those in VMSC and the relevant analysis have been done in details in section 3.3.1.

Question 3: What role does the confidence metric play in your selection of views for materialization and how is it different from conventional association rule mining?

Answer: Conventional association rule mining method discovers the relationships between itemsets present in a large dataset. It identifies the likelihood of one item being associated with another item. In the research work depicted in chapter 4 of the thesis, a Boolean Association Rule parameter called Support has been used to check the how frequently a set of attributes appear together from a series of query transactions.

But it is also required to identify whether any other attribute is there to be materialized along with the attributes already chosen. For this, the Confidence values of other attributes on the already selected attributes are to be calculated. Effectively, after applying both the Support values and the Confidence values, all the important and associated attributes could be identified for materialization.

Question 4: How does using materialized views reduce the cost of incremental data mining operations? Can you quantify the improvement?

Answer: With the use of materialized views, incremental data mining operation could be completed with a set union operation only after considering the only the new transactions and applying MVG_AA( ) algorithm as described in chapter 4 of the thesis. Instead of searching all the transactions from the scratch, only new transactions are considered and hence it reduces the cost of iterating through the entire set of transactions.

Question 5: In your proposed methodology, how do you ensure consistency and correctness of mined patterns after database updates?

Answer: The consistency and the correctness are ensured because of the following steps taken during the design and the implementation of the algorithm:

a) All the data have been collected, validated and preprocessed in a uniform way.
b) All the data have been processed through a common set of algorithms.
c) The results obtained after adding new set of transactional data have been standardized with those obtained from the previous instance.

Question 6: What are the key differences in materialized view selection between using Genetic Algorithms and Apriori? Under what conditions would one outperform the other?

Answer: Two algorithms have used distinct features of materialized view selections. Whereas the methods based on Genetic Algorithms have mainly been dependent on the design of the fitness functions, crossover points and the number of generations, the one based on Apriori Algorithm has considered Boolean Association Rules as the deciding factors of view materialization.

The view materialization techniques based on Genetic Algorithms should be preferred over the ones based on Apriori Algorithm if the dependency of the attributes or transactions could be represented in a lattice based structure.

On the other hand, the view materialization techniques based on Apriori Algorithm should outperform the others based on GA if the transactions are represented in the form of association rules

Question 7: Suppose a wildlife conservation organization uses a content-based image retrieval (CBIR) system built with materialized views and SVMs to classify thousands of trail camera images daily. Over time, new species start appearing in the images due to climate-driven migration. Given that the current system uses materialized views for feature storage and a pre-trained SVM for classification, how should the organization update its system to:

a. Efficiently incorporate new species into the retrieval and classification pipeline without rebuilding the entire materialized view and retraining the SVM from scratch, and

b. Maintain real-time query performance in the face of a rapidly growing and evolving image dataset?

Answer:

a) The methodology described in chapter 6 of the thesis basically highlights the usefulness of multi-class classification problem using SVM. Hence if new species keep on appearing because of climate driven migration, the features of the new species are to be collected as were done for the older ones and the new features should be incorporated under a new class. Effectively, it will not be required to rebuild the entire materialized view and retraining the SVM from the scratch.

b) The CBIR algorithm proposed in chapter 6 did not consider storing the entire image. Instead it needs to store only the defining parameters of the images as mentioned in the chapter. In fact, this is the usefulness of using CBIR techniques for image classification problems. So even if the image dataset size increases and evolves, the performance of the algorithm should not be affected.