A Framework for the View Selection Problem in Data Warehousing Environment

B. Ashadevi [1]
Assistant Professor, Department of MCA.
Velalar College of Engineering And Technology
Thindal, Erode - 638012

Dr. R.Balasubramanian [2]
Dean Academic Affairs, PPG Institute of Technology,
Saravanampatty, Coimbatore-35

Dr. P.Navaneetham [3]
Professor & Head of the Department,
Department of MCA.
Velalar College of Engineering And Technology
Thindal, Erode - 638012

Abstract : A set of essential new concepts and tools have evolved into a new technology that makes it possible to access and produce accurate and timely management information for the competitive world. The phrase that has come to characterize this new technology is Data Warehousing (DW). The general problem of selecting an appropriate set of views to materialize is called the materialized view selection problem. In order to acquire a precise and quick response to an analytical query, proper selection of the views to materialize in the data warehouse is crucial. In traditional view selection algorithms, all relations are considered for selection as materialized views. Due to the space constraint and maintenance cost constraint, the materialization of all views is not possible. The primary goal of data warehousing is to select a suitable set of views that minimizes the total cost associated with the materialized views. In this paper, we present a framework, an optimized version of our previous work, for the view selection problem, which intends to achieve the best combination of low query processing cost, low view maintenance cost and good query response. All the cost metrics associated with the materialized views selection that comprise the query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints are considered by this framework. This framework optimizes the maintenance, storage and query processing cost and selects the most cost effective views to materialize. Thus, an efficient data warehousing system is the outcome.

Keywords: **Data Warehousing, Views, Materialization, View Selection, View-Maintenance, Query processing cost, Storage space.**

I. INTRODUCTION

A data warehouse is an information base that stores a large volume of extracted and summarized data for On-Line Analytical Processing and Decision Support Systems. To reduce the cost of executing queries in a data warehousing environment, frequently used aggregates queries are often pre-computed and materialized into summary views so that future queries can utilize them directly. Undoubtedly, materializing these summary views can minimize query response time. However, if the once data changes frequently, keeping these materialized views updated will inevitably incur a high maintenance cost. Furthermore, for a system with limited storage space and/or with thousands of summary views, we may be able to materialize only a small fraction of the views. Therefore, a number of parameters, including the query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints, should be considered in order to select an optimal set of summary views to be materialized.

According to Inmon, W.H [1], a data warehouse is a subject-oriented, integrated, time-varying, nonvolatile collection of data that is used primarily in organizational decision making. The Data Warehouse is the heart of the architected environment, and is the foundation of all decision support system (DSS) processing. On-Line Analytical Processing (OLAP) and Decision Support Systems utilize the large volume of extracted and summarized data stored in an information base referred as a data warehouse [2]. The data warehousing technologies is the basis for the effective embarking of many industries, for instance, manufacturing financial services, transportation, telecommunications, utilities and healthcare.

In order to collect data from many data sources, a data warehouse uses an update-driven approach that communicates through networks both locally and internationally. A solid platform of consolidated historical data is provided for analysis by the data warehouse system and it also distributes such analysis to local and remote users [3]. In order to provide effective solution for the queries posted to the data warehouse, the intermediate results obtained in the query processing are stored in the data warehouse. This can avert the access of the original data sources by the users [4]. A view is a derived relation defined in terms of base (stored) relations. A data warehouse holds multiple views and we have referred the materialized views as the views stored in the data warehouse.

Materialized views are physical structures that precompute the intermediary results, thereby improving data access time. However, additional storage space and maintenance overhead when refreshing the data warehouse is necessitated by the employment of materialized [5]. Owing to the direct availability of integrated information at the warehouse has the ability to answer queries and perform analysis efficiently and quickly [6]. The data warehouse research community provides effective solutions for the analytical queries, however other performance issues such as query response time for a given aggregated query, view maintenance time, etc are not entirely dealt with.

The materialization of views is the most important ordeal in data warehousing. It is impossible to materialize all possible views as large computation and space is necessitated. Consequently, the primary concern in data warehousing is the "view selection problem" that deals with the selection of suitable set of views to materialize that strikes a stability among computational cost and increased query performance [7]. In a dynamic environment, the selection of appropriate set of views to materialize necessitate considerations of additional factors, hence, it is a demanding task. The selection of the materialized views is affected by numerous factors. Thus, the process of selecting the suitable views to materialize in warehouse implementation is a critical issue.

In this paper, we present an algorithm for selecting views to materialize based on the cost model. A cost model was developed to enable the evaluation of the total costs and benefits involved in selecting each materialized view. We applied the algorithm and cost model to the given query set. Based on the cost analysis, a set of materialized views are selected to optimize the total cost, so that the best combination of good performance and low maintenance cost can be achieved.

The remainder of the paper is organized as follows. Section 2 discusses the related work. The cost model and procedures for the view selection problem are presented in section 3. Section 4 provides the experimental results. We conclude the paper in section 5.

II. RELATED WORK

The accumulation of data has led to the recent availability of outsized archives of data in industry and organization. The decision making process is faced by critical problems due to the employment of these bulk data. These problems can be managed by the developing new data models and decision support systems. The problem of finding views to materialize to answer queries has traditionally been studied under the name of view selection. Its Original motivation comes up in the context of data warehousing. Harinarayan, V et al. [8], they presented a greedy algorithm for the selection of materialized views so that query evaluation costs can be optimized in the special case of "data cubes". However, the costs for view maintenance and storage were not addressed in this piece of work. Chakravarthy, U.S et al. [9], proposed an approach for reducing the cost of query evaluation on a given database by grouping queries. Techniques applied to optimizing individual queries, such as semantic information, syntactic information and knowledge based information can also be applied. Chaudhuri, S et al. [10], discussed a rich set of execution alternatives that can significantly enhance the quality of the plans produced. They also discussed how one can choose among the alternatives. Zhuge, Y et al. [11], proposed an "Eager Compensating Algorithm" for new view maintenance problem that can be used to eliminate the anomalies. But they did not explain how ECA can be adapted to views over multiple sources.

Yang, J et al. [12] proposed a heuristic algorithm which utilizes a Multiple View Processing Plan (MVPP) to obtain an optimal materialized view selection, such that the best combination of good performance and low maintenance cost can be achieved. However, this algorithm did not consider the system storage constraints. Gupta. H et al. [13] developed a greedy algorithm to incorporate the maintenance cost and storage constraint in the selection of data warehouse materialized views. "AND-OR" view graphs were introduced to represent all the possible ways to generate warehouse views such that the best query path can be utilized to optimize query response time. Shukla. A et al. [14] proposed a simple and fast heuristic algorithm, PBS, to select aggregates for precomputation. PBS runs several orders of magnitude faster than BPUS, and is fast enough to make the exploration of the time-space tradeoff feasible during system configuration.

Zhang. C and J. Yang [15] proposed a completely different approach, Genetic Algorithm, to choose materialized views and demonstrate that it is practical and effective compared with heuristic approaches. Stillger, M et al. [16] proposed a genetic programming model for one of the hardest problems in databases, the query optimization problem. They explored the nature of the problem makes it particularly appropriate for Genetic Programming (GP), since the Query Execution Plan (QEP) of a query can be conveniently observed as a genetic program. They specified the search space of their odel and presented the GP operators applied on the QEPs of this space to produce the successors of each population.

Lee. M and Hammer. J [17] proposed an efficient solution to the maintenance-cost view selection problem using a genetic algorithm for computing a near optimal set of views used to search for a near optimal solution. Ziqiang Wang and Dexian Zhang [18] proposed a modified genetic algorithm for the selection of set views for materialization.

J. X. Yu et al. [19] proposed a new constrained evolutionary algorithm for the maintenance-cost view-selection problem. Constraints were incorporated into the algorithm through a stochastic ranking procedure. No penalty functions were used. Kamel Aouiche et al. [5] proposed a framework for materialized view selection that exploits a data mining technique (clustering), in order to determine clusters of similar queries. They also proposed a view merging algorithm that builds a set of candidate views, as well as a greedy process for selecting a set of views to materialize.

III. Proposed work

This Paper is focused on development a framework for choosing views to materialize in order to achieve improved query response in low time by the reduction of the total cost involved with the materialized views. All the cost metrics associated with materialized views like the query execution frequency, query access cost, base-relation update frequency, view maintenance cost and the system's storage space constraints are utilized by the proposed framework. Existing materialized views are maintained by the system from time to time by confiscating views with low access frequency and high storage space. The queries having high access frequencies are chosen for the view selection problem. The intermediary views in the queries are represented in a simple format and results in reduced computational complexity. An algorithm is projected for choosing the views to materialize on basis of their weightage in the provided query set.

A. *Conceptual Framework for Optimized View Selection Problem*

This section explains the proposed optimized cost effective framework for materialized view selection. The proposed framewok exploits all the cost metrics associated with materialized views such as query frequency, query access cost, base-relation update frequency, view maintenance cost and the systems's storage space constraints. The materialized view selection problem can be described as follows: given a set of queries Q and a quantity S (available storage space) and maintenance time MT and existing materialized views Mv, the view selection problem is to select a set of views M to be materialized, that minimize the total cost associated with materialized views under storage space and maintenance cost constraints. The storage space constraint is the space which should not be exceeded by materializing the views. The view maintenance cost is the sum of the cost propagating each source relation change to the materialized views. This sum can be weighted, each weight indicating the frequency of propagation of the changes of the associated source relation.

The framework sustains existing materialized views periodically by removing views low access frequency and high storage space. The queries with high access frequencies are selected for the view selection problem. The intermediary views are represented in a comparatively simpler format than that our previous work, which in itself was an enhancement over the conventional representation with the aid of AND-DAG that makes use of a tree based structure resulting in computational complexity and additional traversal time. An algorithm is proposed for the selection of views to materialize based on their weightage in the given query set and storage space. Then the query access cost and maintenance cost of selected views are calculated. The total cost of each view is calculated and views with optimum cost under the maintenance and space constraints are selected for materialization. The proposed framework is discussed detailed in the following subsections.

B. *Preservation of Existing Materialized Views*

This sub-section details the preservation of the existing materialized views. Before selecting new views for materialization, the existing materialized views are sustained based on their access frequency and storage space. The steps for the above process are given in Procedure 1.

Assumptions:

Mv	-	Vector of Materialized Views
N	-	Total Number of Materialized Views
MS	-	Memory size of materialized Views
Thres	-	Threshold Value
AF	-	Access Frequency of Materialized views.

Procedure 1:

Procedure remove-existing-materialized-view (existing materialized views)

```
// Input : Existing Materialized Views
// Output : Selects materialized views with high access frequency and less storage space
 for each materialized view in Mv
 begin
        W ← 2log(AF)-log(MS)
        if (W<Thres) then
                Remove current Materialized view
end
```

C. *Selection of Views Based on Weightage*

This sub-section details the initial selection of views based on their weightage in the given query set and storage space. Instead of selecting all the queries, the queries which have high access frequency are selected for the view selection problem. The queries are selected from the given query set using Procedure 2.

Assumptions:

Q	-	Given set of Queries
SS	-	Storage Space of Given Query
Φ	-	Threshold Value
AF	-	Access Frequency of Given Query.
SQ	-	Vector of Selected Queries

Procedure 2:

Procedure Select-Materialized –Views (set-of-queries)

```
// Input : Given set of queries Q
// Output: Gives views with high access frequency and low storage space based on their weightage SQ
for  each query in Q
begin
    W ← 2log(AF)-log(SS)
If (W < Φ)
```

```
        add  query to vector SQ
    end
```

The queries having high access frequency less than the threshold value Φ are selected for materialized view selection problem. After that the conditional clauses in each query are represented in a simple conditional clause structure format using Procedure 3.

Assumptions:

SQ	-	Selected set of Queries
Q_C	-	2D Array of Conditional Clauses
Q_{IC}	-	2D Array of integer values of Q_C

Procedure 3:

Procedure Extract-Conditional-Clauses(SQ)

```
    // Input : Selected query which is an result of  procedure 2
    // Output : Conditional Clauses which is stored into $Q_C$
    for each query in SQ
    begin
            if  SQ has conditional clauses then
               $Q_C$ [i] ← Conditional Clause
    end
```

Each distinct conditional clause in QC is mapped to an integer value and the count of each distinct clause is calculated using Procedure 4.

Assumptions:

DCC	-	Distinct Conditional Clauses
CC	-	Count of Conditional Clauses
C_C	-	Conditional Clause

Procedure 4:

The conditional clauses in each query are represented in two dimensional format using procedure 3. This two dimensional is converted into one dimensional representation and their counts are taken simultaneously for further processing. The algorithm for the above is as follows:

Procedure Count-Distinct-Conditional-Clause (Q_{Ci})

```
    \\ Input : conditional clause $Q_{Ci}$
    \\ Output : Produce distinct conditional clause
    Initialize index to 0
    for each row(i) in $Q_C$
    begin
    for each conditional clause CC in row
    begin
         if(DCC<>$C_C$)
                DCC << $C_C$
                CC ← CC +1;
         else
                index ← DCC[$C_C$]
         end if
    end
    end
```

Then the views are selected based on their weightage in the given query set and storage space using procedure 5. then views with weightage greater than a threshold value α are selected for further process.

Assumptions :

M_U	-	Vector of storage space needed to store result
M_{Tot}	-	Total storage space needed
CC_{Tot}	-	Total count
SV	-	Selected set of views
a	-	Threshold value

Procedure 5 :

```
    // Input : Distinct Conditional Clause DCC
    // Output : Set of Materialized Views
    Procedure Select-Materialized-Views (DCC)
    for each conditional conditional clause in DCC
    begin
        $F1 = CC / CC_{Tot}$
        $F2 = M_U / M_{Tot}$
        $W = \log(F1) + \log(F2)$
         If(W<α)
            add current conditional clause based view to SV for
            further process
    end
```

D. *Query Processing Cost*

The cost of query processing is query frequency multiplied by the cost of query access from the materialized views. The query processing cost of each view from SV is calculated using the following formula.

$$QP_{COST} = 1 / \sum_{i=1}^{N} Freq * Ca(V)$$

Where N is the total number of queries, Freq is the frequency of query and Ca(V) is the cost of access for query q using view V.

E. *View Maintenance Cost*

View maintenance is the process of updating precomputed views when the base fact table is updated. The maintenance cost for materialized view is the cost used for refreshing this view whenever a change is made to the base table. The maintenance cost is calculated using update frequency and the priority value of the base table. The view maintenance cost is calculated using procedure 6.

Assumptions:

P	-	Priority of Base table
UF	-	Update frequency of Base Tables

Procedure 6:

Procedure view-maintenance-cost (SV)

```
\\ Input : Selected view SV
\\ output : View maintenance cost
For each view in SV
begin
    For each base table
begin
        $VM_{COST[i]} = 1 / P[i] * ( 1/ UF[i] )$
end
end
```

F. ***View Selection Problem***

The total cost of each view is calculated by summing the query processing cost and view maintenance cost. Then the view are sorted in ascending order according to the value of the Total Cost

$$\text{Total Cost} = QP_{COST} + VM_{COST}$$

Then the views with minimal cost whose maintenance time and storage space falls within the given constraints are selected for materialization.

IV. EXPERIMENTAL RESULTS

In this section, we have presented the results of our experimental analysis. We have implemented all the procedures of our proposed approach in Java. The Procedure 1 has successfully removed the existing materialized views with low access frequency and high storage space and thus freed the space for the materialization of new views. The Procedure 2 has successfully selected the queries with high access frequencies and low storage space for the view selection problem. This work is an optimized work of our previous work [20]. The conditional clauses from each selected query were extracted by Procedure 3. Then these conditional clauses were represented using a single different representation instead of using AND-DAG graph. The Figure 1 depicts the simple conditional clause structure representation for eleven queries. We count the distinct conditional clause using Procedure 4. From the available views, some views were initially selected based on procedure 5. We can conclude that, our framework finally selects view with minimum cost for materialization under the storage space constraints and maintenance cost constraints by considering all the cost metrics associated with the materialized views.

We have compared our proposed approach OVSP (Optimized View Selection Problem) against OCEMS (Optimized Cost Effective approach for Materialized View Selection) with the aid of time. The optimized View Selection Problem consumes less time than the OCEMS. The results of our experiments have been clearly shown in the table and the analysis is presented in the graph. In Table 1 and Table 2, the result of both procedures is given. In Table 3, the size of the files having queries and the time taken to materialize are given. The graphical representation (Figure 2, 3) shows that the optimized algorithm is better than our previous work in terms of time.

Table 1: The results of OCEMS

Table Size (in KB)	With Actual Number of Row	OCEMS(Optimized Cost Effective approach for View Selection Problem)				
		Selected Queries	With actual no.of conditional clauses	With actual no.of distinct conditional clauses	Selected conditional clause	Selected Materializ-ed Views
0.5	11	10	29	19	5	11
1	21	10	27	19	5	11
1.5	30	29	86	75	3	9
2	38	37	114	103	3	9
2.5	45	147	133	03	9	7
4	71	70	220	220	3	9
4.5	79	78	262	251	3	9
5	80	79	269	258	3	9
8	130	464	452	1	1	13
10	162	158	571	558	1	1

Table 2: The results of OVSP

Table Size (in KB)	With Actual Number of Row	OVSP (Optimized View Selection Problem)				
		Selected Queries	With actual no.of conditional clauses	With actual no.of distinct conditional clauses	Selected conditional clause	Selected Materializ-ed Views
0.5	11	7	21	14	5	9
1	21	7	23	15	5	9
1.5	30	7	23	15	5	9
2	38	7	23	15	5	9
2.5	45	7	23	15	5	9
4	71	8	27	19	1	1
4.5	79	10	35	27	4	10
5	80	10	35	27	4	10
8	130	13	45	37	4	10
10	162	23	82	74	2	6

Table 3: comparative results of execution time for OCEMS and OVSP

Table Size (in KB)	**Execution Time (sec) in OCEMS**	**Execution Time (sec) in OVSP**
0.5	.219	.250
1	.375	.218
1.5	.390	.219
2	.469	.234
2.5	.421	.235
4	.516	.265
4.5	.641	.468
5	.687	.265
8	.328	.368
10	.234	.328

Figure 1: Simple representation of conditional clause structure format

a=b 0	b=c 1	c=a 2	
a=b	c=d 3	b=c	e>c 4
c=d	e>c 5		
a2<e2 6	c2=d2 7	b2like%a 8	
b1=d1 9	d1=a1 10	b1>c1 11	e1like%use% 12
a=b	b=c	c>a 13	
b1<d1 14	d1=a1		

(where, 0, 1, 2, …..14 distinct conditional clauses)

Figure 2: Time analysis Graph of OCEMS

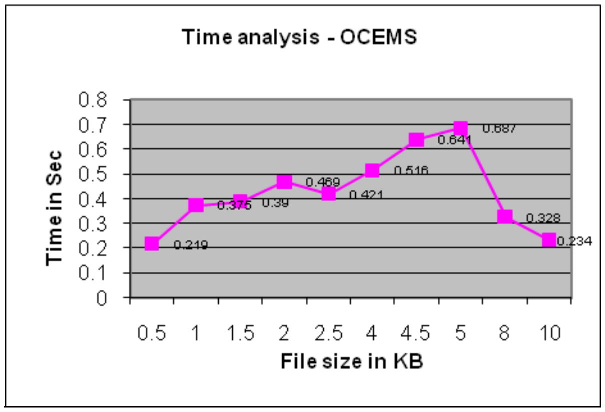
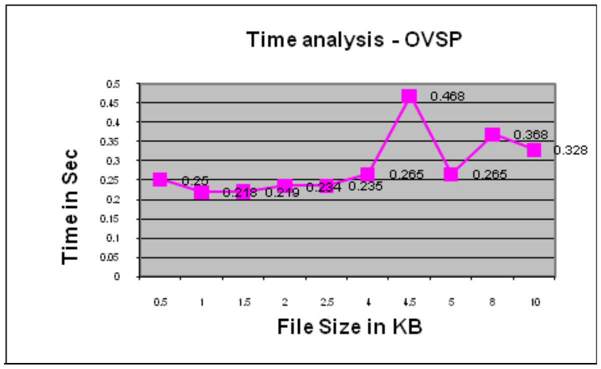


Figure 3: Time analysis Graph of OVSP



V. CONCLUSION

The selection of views to materialize is one of the most important issues in designing a data warehouse. The view-selection problem has been addressed in this paper by means of taking into account the essential constraints: maintenance cost and storage space. In this paper, we have presented a framework, which is an optimized version of our previous work, for selecting views to materialize so as to achieve the best combination of good query response, low query processing cost and low view maintenance cost in a given storage space constraints. The presented framework considered all the cost metrics associated with materialized views such as query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints. The most cost effective views have been selected for materialization by the framework and the maintenance, storage and query processing cost of the views have been optimized. We have compared the results with our previous work in terms of time.

ACKNOWLEDGMENT

The authors appreciate the editors and anonymous referees for their invaluable suggestions and comments which help us improve the paper's quality and presentation.

REFERENCES

[1] Inmon. W.H. (1996). Building the data warehouse. Second Edition., John Wiley and Sons, Canada, ISBN:0471-14161-5.
[2] Chaudhuri. S., and Dayal. U. (1997). An overview of data warehousing and OLAP technology. ACM SIGMOD, pp:65-74.
[3] Yu, J.X., Yao. X., Choi. C.H ., and Gou. G. (2003). Materialized view selection as constrained evolutionary optimization. IEEE transactions on System Man Cybernetics Part C, pp:458-467.
[4] Zhang. C., Yao. X., and Yang. J. (2001). An Evolutionary approach to materialized views selection in a data warehouse environment. IEEE Transactions on System Man Cybernetics, pp:282-294.
[5] Kamal Aouiche. P., Jouve, and J. Darmont. (2006). Clustering-based materialized view selection in data warehouses. In ADBIS'06, volume 4152 of LNCS, pages 81–95.
[6] Zhuge. Y., Garcia-Molina. H., Hammer. J., and Widom. J. (1995). View Maintenance in a Warehousing Environment. Proceedings of SIGMOD Conference. pp:316-327.

[7] Gupta. H., and Mumick. I.S. (1999). Selection of Views to Materialize under a Maintenance-Time constraint. Proceedings of International Conference on Database Theory. pp:453-470.

[8] Harinarayan.V., Rajaraman. A., and Ullman. J. (1996). Implementing data cubes efficiently. ACM SIGMOD, pp:205-216.

[9] Chakravarthy. U.S., and Minter. J. (1982). Processing Multiple Queries in Database Systems. Journal of Database Engineering. pp:38-44.

[10] Chaudhuri, S., Krishnamurthy. R., Potamianos. S., and Shim. K. (1995). Optimizing Queries with Materialized Views. Proceedings of the International Conference on Data Engineering. pp:190-200.

[11] Zhuge.Y., Garcia-Molina. H., Hammer. J., and Widom. J. (1995). View Maintenance in a Warehousing Environment. Proceedings of SIGMOD Conference. pp:316-327.

[12] Yang. J., Karlapalem. K., and Li. Q. (1997). Algorithms for Materialized view design in data warehousing environment. Proceedings of the Twenty third International conference on Very Large Data Bases, USA, pp:136-145.

[13] Gupta. H., and Mumick. I.S. (1999). Selection of Views to Materialize under a Maintenance-Time constraint. Proceedings of International Conference on Database Theory. pp:453-470.

[14] Shukla. A, P. Deshpande, and J. F. Naughton. (1998). “Materialized view selection for multidimensional datasets,” in Proc. 24th Intl. Conf. Very Large Data Bases, pp. 488–499.

[15] Zhang. C and J. Yang. (1999). Genetic algorithm for materialized view selection in data warehouse environments. Proceedings of the International Conference on Data Warehousing and Knowledge Discovery , LNCS, vol. 1676, pp. 116–125.

[16] Stillger. M and Spiliopoulou. M. (1996). Genetic Programming in Database Query Optimization. Proceedings of first national conference on Genetic Programming. pp:388-393.

[17] Lee. M., and Hammer. J. (2001). Speeding up warehouse physical design using a randomized algorithm. International Journal of Cooperative Information System, pp:327-353.

[18] Ziqiang Wang and Dexian Zhang. (2005). Optimal Genetic View Selection Algorithm Under Space Constraint, International Journal of Information Technology, vol. 11, no. 5, pp. 44 – 51.

[19] J. X. Yu, X. Yao, C. Choi and G. Gou. (2001). Materialized view selection as constrained evolutionary optimization. IEEE Transactions on Systems, Man and Cybernetics, vol.31, no. 3, pp. 282-293.

[20] Ashadevi. B., and R. Balasubramanian. (2009). Optimized Cost effective Approach for Selection of Materialized Views in Data Warehousing. International Journal of Computer Science and Technology, vol. 9, no. 1, pp. 21-26.

Mrs. B. Ashadevi received B.Sc., MCA from the Kamaraj University Madurai, in 1997, 2000 respectively and M.Phil from the Mother Teresa Womens University, Kodaikanal 2004, where she is currently pursuing the PhD. She was a Lecturer between 2000 and 2006. Currently, she is working as an Assistant Professor in Velalar College of Engineering and Technology at department of MCA. Her current research interests include Knowledge and Database Engineering. She has authored a book on Database Management systems. She has presented papers in national and international conferences and published research articles in International journals.

Dr. R. Balasubramanian received B.Sc (Mathematics) – 1967 at Govt.Arts College, Coimbatore, M.Sc (Mathematics) – 1969 at PSG Arts College, Coimbatore and Ph.D – 1990 at PSG College of Tech. On completion of his M. Sc program he served CIT, Coimbatore for two years as Associate Lecturer. In June 1971 he joined PSG Tech as Associate Lecturer served PSG Tech till November 2000. During his stay at PSG Tech he saw few phases of professional caderes like, Lecturer, Special Temporary Assistant Professor and Assistant Professor. During November, he opted for voluntarily retirement and joined Sri Krishna College of Engineering and Technology as Professor and Head of the Department of Mathematics and Computer Applications. He has published more than 15 research papers in national and international journals. Attended a number of short-term courses and conferences to enrich his knowledge. He has organized a National level conferences and chaired many technical sessions of the mathematical conferences organized elsewhere in the country. He has authored a series of books on Engineering Mathematics and Computer Science. He has supervised one PhD thesis and several M. Sc (App. Math), MCA project works. His interest includes, applied mathematics, partial differential equations, Data structures. He was the principal investigator of UGC sponsored research projects. He is a member of board of studies of Avinashilingam deemed University and chairman board of examinations of several Universities. He is a life member of many professional bodies like ISTE, ISTAM, CSI. His mission is to impart highest quality of mathematics and mould younger generation.

Dr. P. Navaneetham received B.Sc (Computer Science) – 1993 at Erode Arts College, Erode, M.Sc.,(Computer Science) – 1995 at Urumu Dhanalakshmi College, Trichy, M.Phil.,(Computer Science) – 2002 at Erode Arts College. On Completion of his M.Sc., Program he served as Lecturer in Cherraan’s Arts Science College, Kangayam. During his period of stay he saw few phases of caders Sr. Lecturer and Head of the Department. In Dec. 2005 he joined RVS College of Arts and Science (Autonomous), Coimbatore as Head of the Department. Since Dec 2006 he joined at Velalar College of Engineering and Technology as Assistant Professor cum Head of MCA department. Currently, he is designated as Professor at Velalar College of Engineering and Technology. He has presented more than 10 papers in national and international level conferences and organized many national level conferences. He has supervised several MCA project works and more than 40 M.Phil thesis works. His interest includes, Image processing, Compiler Design, Applied Mathematics. He is the member of board of studies in various deemed universities and autonomous colleges. He is a life member of many professional bodies like ISTE.