Analysis of View Selection Problem in Data Warehousing Environment

Ashadevi. B [#1]

Associate Professor, Department of Computer Applications, Velalar College of Engineering and Technology, Tindal, Erode-12

[1] aashadeviphd@gmail.com

Abstract—**A Data Warehouse (DW) can be seen as a set of materialized views defined over the source relations. During the initial design and evolution of a DW, the DW designer is faced, on many occasions, with the problem of selecting views to materialize in the DW. This study presents the critical survey of the methodologies to select materialized view in more efficient way. In this study, we are summarizing all these methodologies with critical analysis. Advanced solutions are particularly focusing the evolutionary optimization methods. I have analyzed and compartmentalized the available literature on the basis of relevant evaluation parameters. Important books, PhD thesis, links, etc. are also given in study. To work out this study studied more than fifty research papers. This study may be helpful to the researchers, who are working in the domain of the Data Warehouse focusing on the view selection problem.**

Keyword-Data Warehouse, Materialized View, View Selection Problem

I. INTRODUCTION

A Data Warehouse (DW) is a repository of information collected from multiple, possibly heterogeneous, autonomous, distributed databases and other information sources for the purpose of complex querying, analysis and decision support. Data warehousing is an emerging approach for effective decision support. According to [1], a DW is a subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making. The need for data warehousing techniques is justified due to the decision support queries, which are ad-hoc user queries in various business applications. In these applications, current and historical data are comprehensively analysed and explored. A class of queries typically involves group-by and aggregation operators.

During the initial design and evolution of a DW, the DW designer / administrator is, on many occasions, faced with the problem of selecting view to materialize in the DW. This problem has been addressed in the literature for different classes of queries / views and with different design goals.

From a computer science perspective, a data warehouse is a collection of materialized views derived from base relations that may not reside at the warehouse. Therefore, a data warehouse is considered as a definer and storage of views. When a view is defined, the database system stores the definition of the view itself, rather than the result of evaluation of the relational algebra expression that defines the view. Hence, a view is a derived relation defined in terms of base relations. A view thus defines a function from a set of base tables to a derived table; this function is typically recomputed every time the view is referenced. According to the perspective of materialized views, at the abstract level the contents of the data warehouse are regarded as a set of materialized views defined over the data sources. These materialized views are designed based on the user's requirements (e.g., frequently asked queries). The benefit of using materialized views is significant. Since index structures can be built on materialized views, consequently, database access to the materialized view is just a cache, which is copy of the data that can be accessed quickly. Integrity checking and query optimization can also benefit from materialized views. In short, when a view is defined, normally the database stores only the query defining the view. In contrast, a materialized view is a view whose contents are computed and stored. It is cheaper in many cases to read the contents of a materialized view than to compute the contents of the view by executing the query defining the view. Materialized views are important for improving performance in some applications.

There are many review papers are available, but most of them are from year 1980 to 1990. Many researchers have proposed the solution strategies for materialized view selection problem. But year 2000 onwards, some advance techniques are used to find the solution (example, simulated annealing, genetic algorithm etc.). This study is intended for the beginners in the domain of materialized view selection problem. This study provide the details of basic structures, mathematics, cost modelling, books PhD thesis, links, glossary of key terms, benchmark database and critical analysis on past and present methods.

A. Materialized View Selection Problem and Cost Model

The general problem of selecting an appropriate set of views to materialize is called the materialized view selection problem. There are many research issues related to DW [2], among them materialized view selection is

one of the most challenging ones. On one hand, materialized views speed up query processing. On the other hand, they have to be refreshed when changes occur to the data sources. Therefore, there are two costs involved in materialized view selection: the query evaluation cost and materialized view maintenance cost. The main objective of materialized view selection problem is either the minimization of a constraint or a cost function. A constraint can be system oriented (space constraint) or user oriented (query response time constraint). Most of the approaches are designed for minimization of a cost function. Gupta, H (1997), and Barlis. E. et al. (1997) defined view selection problem and take as input the queries that the data warehouse has to satisfy for an initial or an incremental design.

- The view maintenance cost is the sum of the cost of 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 expressions used to compute the changes of the source relations and are called maintenance expressions. When the source relation changes affect more than one materialized view, multiple maintenance expressions need to be evaluated. Shim, K et al. (1994) proposed a technique for multi query optimization that can be used to detect common sub expression between maintenance expressions in order to derive an efficient global evaluation plan for these maintenance expressions.
- The overall query evaluation cost is the sum of the cost of evaluating each input query rewritten (partially or completely) over the materialized views. This sum can also be weighted, each weight indicating the frequency, or importance of the associated query. The aim of approach is minimizing the query evaluation cost (Harinarayan, V et al. 1996; Shukla. A et al. 1998). The materialized views are maintained using an incremental approach. In an incremental approach, only the changes that must be applied to the view are computed using the changes of the source relation (Ashish Gupta and Mumick. I. 1995). These view changes are then applied to the materialized view.

Low view maintenance cost can be achived by replicating source relations at the Data warehouse; in this case the query evaluation cost is high. Low query evaluation cost can be obtained by materializing at the Data Warehouse all the input queries. In this case the view maintenance cost will be high. In this case the view maintenance cost will be high. For this reason, one can choose a linear combination of the query evaluation and view maintenance cost. In most of the research related to the materialized view selection used the linear cost model [6], which is used for view cost evaluation. Only difference occurs at the assumptions which are used in the evaluation of mathematical model of the cost. Analytical justification of linear cost model based on graph theory is given in [9]. General linear cost model is described as follows:

Let us assume that a set of queries Q = (Q1,Q2,.....,Qn) are defined over a set source relations S = (S1,S2,....,Sn) and a multi query graph G. Let GQi be the query DAG for Qi, i=1,...,m in G. E(GQi) denotes the cost of evaluating Qi, using GQi. The query evaluation cost of G is:

$$E(G) = \sum_{i=1}^{n} fQiE(GQi) \quad (1)$$

Where,fQi is query frequency.

Let GSi where i = 1,...,n be the change propagation DAGs for Q i=1,...,m in G. M(GSi) denotes the cost of propagating the changes of Si to the materialized views using GSi. The view maintenance cost of G is:

$$M(G) = \sum_{i=1}^{n} fSiM(GSi) \quad (2)$$

Where, fSi is source relation updation frequency. The values of query frequency and base relation updation frequency can be assumed for the experimentation. View selection problem can be described as follows:

How to select an appropriate set of materialized views from a certain graph G, so that the total query processing cost for the supported queries and the total maintenance cost of these materialized views is minimal.

Given a G, let M be a set of views in a G to be materialized, fq, fs the frequency of executing queries and frequency of updating base relations, respectively. Furthermore for each v∈M, let E(GQi(V)) and M(GSi (v)) denote the cost of access for query using v and the cost of maintenance of view v base on changes to base relation s, respectively (where, v∈QN is the set of queries and s∈ SRN is the set of base relations). Then the query processing cost will be:

$$E\big(G(v)\big) = \sum_{i=1}^{n} fQiE\big(GQi(v)\big) \quad (3)$$

And the materialized view maintenance cost will be:

$$M(G(v)) = \sum_{i=1}^{n} fSiM\big(GSi(v)\big) \quad (4)$$

Thus the total cost of materializing a view is

$$\text{Total cost (v)} = E(G(v)) + M(G(v)) \quad (5)$$

Therefore, the total cost of materializing a set of views M is Total cost:

$$\text{TOTAL COST} = \sum_{V \in M} \text{TOTAL COST}(V) \qquad (6)$$

II. REPRESENTATION OF VIEW SELECTION PROBLEM FOR THE MATERIALIZED VIEWS

The goal of view selection problem is to find a set of views that minimizes the expected cost of evaluating the queries. When designing a data warehouse, it is extremely important to minimize the cost of answering queries because the warehouse is very large. The selection of the optimal collection of views for available storage space and minimum query cost is referred to as the view selection problem. There are many numbers of the base tables (with schemas in hundreds attributes) from dozens of data sources, it would be very challenging to decide which views should be materialized. To solve the view selection problem, mathematical formulation is the first step. In view selection problem, data structures are required to represent the view selection. For this, the following subsections are generally used.

A. Relational Algebra

Relational algebra is a procedural query language. A set of operations are used to express a query. Each operation takes one or more relations as arguments and produces a new relation as the result. This property makes it easy to compose operations to form a complex query. The fundamental set of Relational Algebra operations are Selection (sigma σ), Projection (pi $\prod$), Union ($\bigcup$), Set-difference (-), Cartesian – product (X), Rename (rho ρ). These fundamental operations are involved in the query processing for the query optimization process.

B. Directed Acyclic Graph

In mathematics and computer science, a directed acyclic graph (dag or DAG), is a directed graph with no directed cycles, which is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is no way to start at some vertex V and follow a sequence of edges that eventually loops back to V again. For example, if an edge u<=v indicates that v is a part of u, such a path would indicate that u is a part of itself, which is impossible.

C. AND / OR Graph

A form of graph or tree used in problem solving and problem decomposition. The nodes of the graph represent states or goals and their successors are labelled as either AND or OR branches. The AND successors are sub goals that must all be achieved to satisfy the parent goal, while OR branches indicate alternative sub goals, any one of which could satisfy the parent goal.

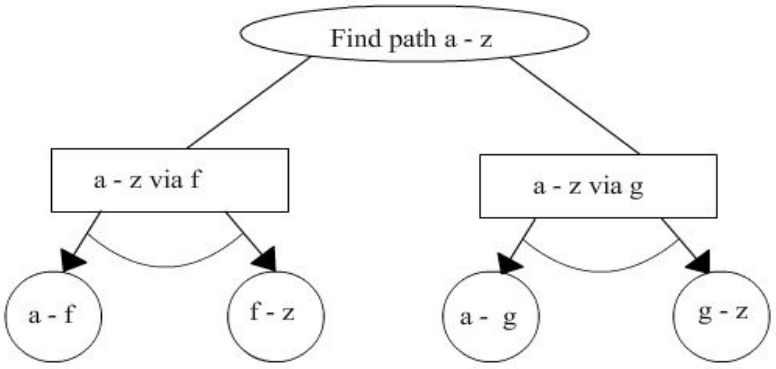

Fig. 1: AND – OR graph: or-nodes as ellipse, and-nodes as boxes

A problem: Find path a-z can be solved by either solving a-z via f or a-z via g. A problem a-z via f can be solved by both the sub problem a-f and f-z and a problem a – z via g can be solved by both the sub problems a-g and g-z. Groups of sub problems are joined together by an arc.

D. Lattices

On-line analytical processing (OLAP) systems builds data cubes with multiple dimensions. Data cubes are made up of two elements: dimensions and measures. The dimensions and measures are simply the actual data values. Most OLAP systems can build data cubes with many more dimensions. The property of cubes is that the n-D data can be represented as a series of (n-1)-D cubes. A d-dimensional base cube is associated

with 2^d cuboids (i.e., sub cubes). Many researchers proposed the data cube operator as a means of simplifying the process of data cube construction. Most were based upon the exploitation of the data cube lattice, a directed graph that depicts the relationship between all 2^d cuboids in a given d-dimensional space. The $\leq$ operator imposes a partial ordering on the queries. Consider two queries Q_1 and Q_2. $Q_1 \leq Q_2$ can be defined if Q_1 can be answered using only the results of Q_2. It is said that Q_1 is dependent on Q_2.

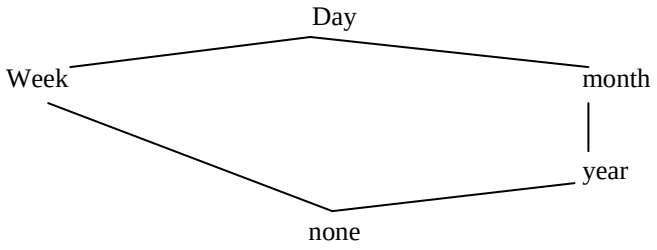

Fig. 2: Time dimension

The dimension of the data cube consists of more than one attribute and the dimensions are organized as hierarchies of these attributes. The lattices for the time dimension shown in Fig. 2 is that of organized by the time dimension into the hierarchy: day, month and year.

Based on whether the current materialized views will be used in computing the new views, and whether the data warehouse will query the remote data sources for additional data to do the computation, the data warehouse view maintenance techniques are classified into four major categories: self-maintainable recomputation, not self-maintainable recomputation, self-maintainable incremental maintenance and not self-maintainable incremental maintenance (Wang, X et al. 2004). Their approach provided a comprehensive comparison of the techniques in these four categories in terms of the data warehouse space usage and number of rows accessed in order to propagate an update from a remote data source to a target materialized view in the data warehouse. The comparison of advantages and disadvantages of these categories is given in Table 1. It is shown that self-maintainable incremental maintenance performs the best in terms of both space usage and number of rows accessed.

Both self-maintainable recomputation and self-maintainable incremental maintenance approaches totally separate the data warehouse view maintenance operations from the OLTP operations. Therefore, the view maintenance operations will not consume data sources' local resources. These operations only consume the data warehouse's resources. Even if the remote data sources are not available, the data warehouse view maintenance process can continue running. However, a part or all source data are replicated at the data warehouse to make the data warehouse view maintenance process self-maintainable. These replicated data take space. Data transfer processes are implemented to transfer data from the remote data sources to the data warehouse. Design, implement and maintain these processes are time-consuming. A lot of unnecessary data may be duplicated at the data warehouse.

TABLE I COMPARISON OF FOUR CATEGORIES

CATEGORY	ADVANTAGE	DISADVANTAGE
Self-Maintainable Recomputation	* Data warehouse view maintenance operations are totally separated from OLTP operations. * Unavailable source will not block the data warehouse view maintenance process.	* Data are replicated at data warehouse. * Need extra data storage for replicate data. * Have to implement and maintain data transfer process to transfer data from sources to data warehouse.
Not Self-Maintainable Recomputation	* Very simple to implement * No replicate data at the data warehouse * No extra data storage for replicate data. * Do not have to implement and maintain data transfer processes to transfer data from sources to data warehouse.	* unavailable source will block the data warehouse view maintenance process. * Evaluating queries at the data sources consumes local sources.
Self-maintainable incremental maintenance	* Data warehouse view maintenance operations are totally separated form OLTP operations * Unavailable source will not block the data warehouse view maintenance process. * In the worst case, the number of rows	* Data warehouse view maintenance operations are not separated form OLTP operations. * Data are replicated at data warehouse. * Need extra data storage for replicate data. * Have to implement and maintain data transfer processes to transfer data from sources to data

	accessed to maintain a view is the lowest.	warehouse.
Not self-maintainable incremental maintenance	* No replicate data at the data warehouse. * No extra data storage for replicate data. * Do not have to implement and maintain data transfer processes to transfer data from sources to data warehouse.	* Unavailable source will block the data warehouse view maintenance process. * Evaluating queries at the data sources consume local resources. * Data warehouse view maintenance operations are not separated from OLTP operations. * Have to design the view maintenance process carefully to avoid the anomaly problem. * In the worst case the number of rows accessed is the highest * Performance is down-graded rapidly. * Need extra storage for intermediate data

III. CONCEPTUAL BACKGROUND OF THE MATERIALIZED VIEW SELECTION PROBLEM

A data cube is a multi-dimensional modelling construct. It contains many cuboids. A cuboid is also commonly known as a "view". In this context, a view is a set of aggregated data for a particular set of dimensions. Essentially, a view is the result of a "GROUP BY" query.

In a given data cube, the following implementation alternatives are possible:

1. Physically materialize the whole data cube. This is known as 100% materialization of a data cube. This approach will give the best possible query response time. Obviously, 100% materialization may be infeasible for a large data cube because it will require an excessive amount of disk space. Also, the time required to materialize a view is considerable. So 100% view materialization might take a long time to accomplish, which might not be affordable in today's decision support environment. Also one needs to maintain indices, if any, which will further add to overall cost. Once views are materialized, they need to be maintained to reflect the current or the latest updates in the source data. Hence, as more views are materialized, the view maintenance costs will also increase.
2. Do not materialize any view. In this case, one needs to access the raw data and answer each query. This approach will result in long retrieval times due to high CPU and disk load. But it does not need any extra storage space for the view materialization.
3. The third alternative is to materialize only a part of the data cube. But selecting the right set of views to materialize is the challenge. In a data cube, many views could be derived from other views. Consequently, one may want to materialize a relatively infrequently accessed view if it helps in obtaining many other views quickly. We consider this problem as the materialized view selection problem.

Research interest in materialized views started in the early eighties. One of the early investigations was to speed up the data retrieval process for running queries on views in very large databases. Subsequently, further research studies were reported in view and index maintenance along with comparative evaluations of materialized views on the performance of queries.

The materialized view selection problem formally studied by Harinarayan, V et al. 1996, where major features of the materialized view selection problem are discussed elaborately. There is more recent review of literature for this problem. Typically, a lattice framework is used to capture the dependencies among views. The most common case of the hypercube lattice is considered and examined the choice of materialized views for hypercube in detail, giving some good tradeoffs between the space used and the average time to answer a query. In this research the problem of deciding which set of cells (views) in the data cube to materialize in order to minimize the query response time investigated. Materialization of views is an essential query optimization strategy for decision support application. Right selection of the views to materialize is critical to the success of the strategy (6).

The size of the views is an important component in the formulation of the materialized view selection problem (15). Gupta, H et al. (1997) produced the combined view and index selection problem under a given space constraints. In order to keep a materialized view consistent with the data at sources, the view has to be incrementally maintained. This maintenance of views is known as view maintenance or update costs [13].

Gupta, H et al. (2005) proposed a theoretical framework for the general problem of selection of views in a data warehouse. They have presented competitive polynomial-time heuristics for a selection of views to optimize total query response time under a storage space constraints, for some important special cases of the problem that occur in practice. They have also addressed in detail the view selection problem under the maintenance cost constraint and presented provably competitive heuristics.

In (16), the researchers proposed a framework, which is based on specification of multiple view processing plan (MVPP), to present the problem formally and they proposed a heuristic algorithm based on individual optimum query plans. But they did not use any resource constraint.

Liabio, W et al. (1997) explained an A* search to pick the best set of views when only the maintenance cost is to be minimized. The problem of materialized view selection under a disk space constraint S explained in Gang Gou et al. (2006). However, the proposed A* algorithm can find the optimal solution very efficiently when S is small, and observed that A* algorithm might coverage slowly when S is large. To avoid this problem, developed a new competitive A* algorithm in order to improve the quality of solution. There are many other approaches on selection of common views to be materialized.

Shukla, A et al. (1998) proposed a simple and fast heuristic algorithm called pick by size (PBS) to solve the materialized view selection problem and explored its performance. They pointed out that PBS runs several orders of magnitude faster than the heuristic algorithm proposed by Harinarayan, V et al. (1996) and is fast enough to make the exploration of the time-space trade-off feasible during system configuration. Furthermore, they have examined the view selection problem when subsets of aggregates can be computed using chunks (Shukla, A et al. 1998) and showed that the benefit of the views selected by PBS can be greater then the ones selected without chunk based precomputation.

Barlalis, E et al. (1997) explained the number of representative queries is extremely small with respect to the total number of elements of the complete data cube. Using such indications (inputs), they have explained the technique to select views and an algorithm to perform selection that will reduce the solution space by considering only the relevant elements of the multidimensional lattice.

In order to improve the efficiency of problem, Lee. M and Hammer. J. (2001) assume that the set of materialized views and then ask the question: How do we to rewrite the given OLAP query to make the best use of existing materialized views? They have developed algorithms for the rewrite as well as identifying the materialized views that will best answer the query.

Gray, J et al. (1997) proposed the data cube as a relational aggregation operator generalizing group-by, cross-tabs, and sub-totals. Dynamic view selection problems are an important constituent for supporting fast online queries on such databases. In order to solve view selection problems, one needs the sizes of the various views which are obtained from running group-by queries. Time required for running such queries can be reduced by an order of magnitude by running parallel group-by queries.

An interesting variant has the objective of minimizing the maximum weighted number of rows to be retrieved in responding to any query from the set of queries (21). This version of the view selection problem may be denoted as the bottleneck view selection problem, which provides a guaranteed quality of service to all users.

Chirkova, R et al. (2001) have pointed out that the complexity of the materialized view selection problem depends crucially on the quality of the estimates that a query optimizer has on the size of the views it is considering to materialize. They have shown that when a query optimizer has accurate size estimates of the views, the cardinality of an optimal view selection may be exponential in the size of the database schema. On the other hand, when optimizer uses standard estimation heuristics, they have shown that the cardinality of an optimal view selection is polynomially bounded. For very large databases, it is very time consuming to generate the actual sizes of the views.

Theodoratos, D et al. (1999) proposed a generic method that given a set of SPJ (Select-Project-Join) queries to be satisfied by the data warehouse, generates all the essential sets of materialized views that satisfy all the input queries. In addition, algorithms have been developed so that a materialized view set selected in this way fits in the space allocated to the data warehouse for materialization and minimizes the combined overall query evaluation and view maintenance cost.

Ligoudistianos, S et al. (1998) proposed an approach, which focused on the experimental evaluation of an exhaustive algorithm and developed greedy and heuristic algorithms that expand only a small fraction of the states produced by the exhaustive algorithm. The algorithms are explained in terms of a state space search problem. The data warehouse configuration problem is formulated as a state space search problem based on a representation of view and queries using conjunction of selection and join atomic predicates. A realistic cost model for query processing and view maintenance has been developed.

Mohania, M et al. (1999) explained the problem of incremental maintenance of materialized views in data warehouses. The relational algebraic operators and aggregate functions were used to define views. It is shown that a materialized view can be maintained without accessing the view itself by materializing and maintaining additional relations. These relations are derived from the intermediate results of the view computation.

To find the solution to the view selection problem, an evolutionary approach is described (Horng, J.T et al. 1999). Genetic Local Search (GLS) algorithm is a hybrid heuristic that combines both genetic algorithm and local optimization. A hybrid evolutionary algorithm is applied to solve three related problems. The first is to optimize queries. The second is to choose the best global processing plan from multiple global processing plans. The third is to select materialized views from a given global processing plan.

A randomized approach for incrementally selecting a set of views that are able to answer a set of input user queries locally while minimizing a combination of the query evaluation and view maintenance cost is developed (26). In this process common sub-expressions among new queries and between new queries and old views have been exploited. The approach is based on the simulated Annealing process.

Mistry, H et al. (2001) proposed an efficient plan for the maintenance of a set of materialized views by exploiting common sub expressions between different view maintenance experiences. In particular, it has been shown how to efficiently select (i) expressions and indices that can be effectively shared, by transient materialization, (ii) additional expressions and indices for permanent materialization and (iii) the best maintenance plan-incremental or recomputation for each view. These three decisions are highly interdependent and the choice of one affects the choice of the others. A framework was developed that cleanly integrates the various choices in a systematic and efficient manner.

A scalable algorithm for determining whether part or all of query can be computed from materialized views and describes how it can be incorporated in transformation-based optimizers is presented (Goldstein, J and Larson. P.A. 2001). The main contributions of this paper are (i) an efficient view matching algorithm for views composed of selections, projections, joins and a final group-by (SPJG views) and (ii) a novel index structure (on view definitions, not view data) that quickly narrows the search to a small set of candidate views on which view-matching is applied. The version of the algorithm described here is limited to SPJG views and produces single-view substitutes.

Due to the space constraint and maintenance cost constraint, the materialization of all views is not possible. Therefore, a subset of views needs to be selected to be materialized. The problem noticed is NP-hard, therefore, exhaustive search is infeasible. A View Relevance Driven Selection (VRDS) algorithm based on view relevance to select view is developed (Valluri, S.R et al. 2002). The VRDS algorithm strikes a balance between the query processing cost and the view maintenance cost, whereas greedy algorithm is focused mainly on updates and MVPP algorithm on selecting all beneficial views.

A constrained evolutionary algorithm is proposed by Yu, J.X et al. 2003. Constraints are incorporated into the algorithm through a stochastic ranking procedure where no penalty functions are used and constraint handling technique, i.e., stochastic ranking, can deal with constraints effectively. The algorithm proposed is able to find a near-optimal feasible solution and scales with the problem size well. First, pools of bit string gnomes are generated randomly. This is the initial population. Each gnome represents a candidate solution to the problem to be solved. The length of this gnome is the total number of vertices in the lattice; 1 and 0 mean that the vertices need to be materialized or not, respectively.

In (31), the uses of genetic algorithm for the selection of materialized views are explained based on multiple global processing plans for many queries.

IV. ANALYSIS OF VIEW SELECTION PROBLEM

The data warehouse problem through materialized views is usually stated as the view selection problem. When designing a data warehouse, it is extremely important to minimize the cost of answering queries because the warehouse is very large, queries are often ad hoc and complex and decision support application requires short response time. The determination of the optimal collection of views for available storage space and minimum query cost is referred to as the view selection problem. This view selection problem is totally different from the view selection problem under the disk space constraint. With numerous numbers of the base tables (with schemas in hundreds attributes) from dozens of data sources, it would be very challenging to decide which views should be materialized.

View selection problem can also be solved under the different types of constraints. For example, space constraints, time constraints, aggregation and grouping constraints, source availability constraints and currency constraints etc. The problem of selecting a set of materialized views for answering queries under the presence of updates and a global space constraint is explored in [11], [3], [12], [13], and [14].

TABLE II ANALYSIS OF APPROACHES AND CONSTRAINTS

APPROACH	*REFERENCES*	*CONSTRAINTS*	*REMARKS*
View cost evaluation based on Linear cost model	Harinarayan, V et al.(1996)	Which set of cells (views) in the data cube to materialize in order to minimize the query response time investigated? Space constraints, Maintenance time constraints	Gives some good tradeoffs between the space used and the average time to answer a query.
Proposed algorithm that automate the selection of summary tables and the indexes.	Gupta, H et al. (1997)	The view has to be incrementally Maintained to keep a materialized view consistent with the data at sources	Algorithm that select which subcubes and indexes precomputed for improved query performance. show the trade-off between the performance bounds and the

		Space constraints.	complexity of the algorithm.
Proposed a theoretical framework for the general problem of selection of views in a data warehouse.	Gupta, H et al. (2005)	Space constraints, Maintenance Time constraints.	Competitive polynomial-time heuristics for a selection of views to optimize total query response time.
Proposed a heuristics algorithm based on individual optimum query plans.	Yang, J et al. (1997)	Without any resource constraints.	Framework is based on specification of multiple view processing plan(MVPP), which is used to present the problem formally.
View maintenance techniques are classified into four major categories : self-maintainable recomputation, not self-maintainable recomputation, self-maintainable incremental maintenance and not self-maintainable incremental maintenance.	Wang, X et al. (2004)	Space usage and No.of rows accessed.	Self-maintainable Incremental maintenance performs the best in terms of both space usage and number of rows accessed.
A method for dealing with the problem was developed by formulating it as a state space optimization problem and then solving it is using as exhaustive incremental algorithm as well as a heuristics algorithm.	Theodoratos, D and Sellis. T (1997)	That there are no space restrictions in the data warehouse and space is not the problem does not discuss the complexity of the view selection problem.	An exhaustive algorithm was designed and has provided heuristics for pruning the search space in different cases.
Randomized approach based on the Simulated Annealing process.	Theodoratos, D et al. (2001)	Constraint that the new views and the old views together must be able to answer all the new queries has been imposed.	Simulated Annealing has been used in a variety of optimization problems. In the database area, it has been used for query optimization.
The problem is formulated as a state space search problem by taking in to account multiquery optimization over the maintenance queries and the use of auxiliary views for reducing the view maintenance cost.	Theodoratos, D et al. (1999)	Space constraints.	The static case of the DW design problem in detail.
An exhaustive algorithm with greedy and heuristic algorithms that expand only a small fraction of the states produced by the exhaustive algorithm.	Ligoudistianos, S et al. (1998)	As the number of implications increases the r-greedy performs worse and the heuristic algorithm becomes the winner.	r-greedy algorithm that prune the state space and a new heuristic algorithm that searches a small fraction of the state space and reports a sub-optimal solution.
The problem noticed is NP-hard. A View Relevance Driven Selection(VRDS) algorithm based on view relevance.	Valluri, S.R et al. (2002)	The query processing cost and the view maintenance cost was taken in to consideration.	VRDS algorithm performs better than greedy and MVPP algorithm when there is a space constraint and update frequency is high.
By exploiting common sub expressions between different view maintenance expressions is presented.	Mistry, H et al. (2001)	Increase in cost of optimization is acceptable.	Extended the volcano query optimization framework to generate optimal maintenance plans. A greedy heuristic has been proposed.
The main contributions of this stuffy are (i) an efficient view matching algorithm for views composed of selections, joins and a final group-by (SPJG views) and (ii) a novel index structure	Goldstein, J and Larson. P.A. (2001)	1000 views in the system.	View-matching algorithm was developed, including the filter tree, in SQL server. An index structure is presented, called a filter tree.

(on view definitions, not view data) that quickly narrows the search to a small set of candidate views on which view-matching is applied.			
Proposed Genetic Local search (GLS) technique. GLS approach to solve view selection problem has been adopted.	Horng, J.T et al. (1999)	NP-complete problem.	Genetic algorithm based solution.
A new constrained evolutionary algorithm is proposed.	Yu, J.X et al. (2003)	Constraints are incorporated in to the algorithm through stochastic ranking procedure. No penalty functions are used.	Stochastic ranking can deal with constraints effectively.
Proposed framework for selecting views to materialize(i.e., View selection problem), which takes in to account all the cost metrics associated with the materialized views selection, including query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints	Ashadevi, B and Balasubramanian. R (2008)	Space constraints.	Selects the most cost effective views to materialize and thus optimizes the maintenance storage, and query processing cost.

To help the other researchers in the materialized view domain, the collection of important books, Ph.D thesis and links related to the materialized view selection in data warehouse are given below:

Important Books:

1. W.H. Inmon., Building the Data Warehouse, John Wiley, 1992
2. Effective Database Design Phi 1981.
3. The Data Warehouse Life Cycle Toolkit by Ralph Kimball.
4. The Microsoft Data Warehouse Toolkit with SQL server 2005 and Business Intelligence Toolset by Ralph Kimball.
5. Building a Better Data Warehouse by Don Meyer And Casey Cannon.
6. R. Kimball., The Data Warehouse Toolkit, John Wiley, 1996.

Important PhD Thesis:

1. Materialized Views in Data Warehouses By Dallan Wendell Quass August 1997.
2. Selection and Maintenance of Views in a Data Warehouse by Himanshu Gupta September 1999.
3. Optimization Strategies for Data Warehouse Maintenance in Distributed Environments by Bin Lin April 2002.
4. Efficient Incremental View Maintenance for Data Warehousing by Songing Chen December 2005.
5. Efficient Materialization and use of views in data warehouses by M.F. de Souza and M.C. Sampaio 1999.

Evaluation of approaches using Benchmark database:

1. Providing OLAP to user Analyst : An IT Mandate
   http://www.arborsoft.com/OLAP.html
2. Multidimensional Analysis: Converting corporate Data into strategic information.
   http://www.arborsoft.com/papers/multiToc.html.
3. TPC Benchmark H (Decision Support) Revision 1.1.0.
   http://www.tpc.org/
4. The TPC-H is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications:
   www.it.iitb.ac.in/~chetanv/personal/acads/db/report_html/node3.html
5. The TPC-D benchmark, which simulates a complex DSS workload with 17 queries areas including data warehousing, high performance OLTP and web/E-Commerce:
   www.taborcommunications.com/dsstar/98/1110/100406.html
6. The TPC-E benchmark simulates the data processing associated with a real warehouse.
   www.itjungle.com/two/two080807-story04.html

V. CONCLUSION

Analyzed more than fifty research papers and books. Through this study I tried to give the basic content which is required to be known to the beginner who is going to work in this domain. I have discussed about the basic mathematical background including the cost model formulation and critically analyzed the past and present methods or techniques to solve the materialized view selection problem by providing the summary in Table 1 and 2. We have also provided the details of books, thesis, important links and the benchmark database which can be used to check anybody's approach or technique. In addition to these I have proposed some solutions based on the new advanced techniques of searching and optimization.

One typical area of this problem is the cost model formulation i.e., mathematical formulation. Everyone used the linear cost model. There are two important costs viz., view maintenance cost and query processing cost. Finally, I can say that this study can be the initial guide for the beginner in this domain. Recently many researchers are working in this domain. Still there is a scope to contribute much more in this domain.

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] Gupta, H. (1997). Selection of views to materialize in a Data Warehouse. Proceeding of the 6th International Conference on Database Theory, pp:98-112.

[4] Baralis, E., Paraboschi. S., and Teniente. E. (1997). Materialized view selection in a multidimensional database. Proceeding of the Twenty fourth International conference on Very Large Data Bases, pp:488-499.

[5] Shim, K., Sellis. T.K., and Nau. D. (1994). Improvement on a heuristic algorithm for multiple-query optimization. Data knowledge. Data knowledge Engineering, pp:197-222.

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

[7] Shukla, A., Deshpande. P.M., and Naughton. J.F. (1998). Materialized view selection for multidimensional datasets. Proceedings of the Twenty fourth International Conference on Very Large Data Bases, USA, pp:488-499.

[8] Ashish Gupta., and I.Mumick. (1995). Maintenance of Materialized Views: Problems, Techniques, Applications. pp:1-16.

[9] Dhote, C.A and M.S Ali.(2007). Materialized view selection in data warehousing. Fourth International Conference on Information Technology, ITNG 2007, Aril 2-4, IEEE Computer Society Washington, DC, USA, pp:843-847

[10] Wang, X., Gruenwalda. L., and Zhu.G. (2004). A performance analysis of view maintenance techniques for data warehouses. Data Warehouse knowledge, pp:1-41.

[11] Roussopoulos, N. (1982). View indexing in relational databases. ACM Trans. Database System, 7 : 258-290.

[12] Theodoratos, D. and M. Bouzeghoub. (1999). Data currency quality factors in data warehouse design. Proceedings of the International Workshop on design and management of Data Warehouses, June 14-15, Heidelberg, Germany, pp:1-1.

[13] Gupta, H,, V. Harinarayan, A.Rajaraman and J.D. Ullman. (1997). Index selection for OLAP. Proceeding of International Conference on Data Engineering, April 7-11, Birmingham, UK, pp:208-219.

[14] Liang, W, H. Wang and M.E Orlowska. (2001). Materialized view selection under the maintenance time constraints. Data Knowledge Engineering, 37, pp:203-221.

[15] Gupta, H., and Mumic. I.S.(2005). Selection of views to Materialize in a Data warehouse. IEEE Transaction on Data and Knowledge Engineering, pp:24-43.

[16] Yang, J., Karlapalem. K., and Li. Q. (1997). A framework for designing materialized views in a data warehousing environment. Proceedings of the Seventieth IEEE International Conference on Distributed Computing systems, USA, pp:458.

[17] Labio, W., Quass. D., and Adelberg. B.(1997). Physical database design for data warehouse., Proceedings of the thirteenth International Conference on Data Engineering, Birmingham, UK, pp:277-288.

[18] Gang Gou., YU. J.X., and Hongjun Lu. (2006). A* search : An efficient and Flexible approach to Materialized view selection. IEEE Transactions on Man, and Cybernetics, Part C, pp:411-425.

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

[20] Gray, J., Chaudhuri. S., Bosworth. A., Layman. D., Reichart. D., and Venkatrao. M. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining and Knowledge discovery, pp:29-54.

[21] Agrawal, V., Nadhakeolyar. U., Sundararaghavan. P.S., and Ahmed. M. (2004). Optimal view materialization in a data warehouse. Proceedings of the Decision Sciences Institute, USA, pp:5931-5936).

[22] Chirkova, R., Halevy. A., and Suciu. D. (2001). A formal perspective on the view selection problem. Proceedings of the ACM SIGMOD International conference on Management data, pp:259-270.

[23] Ligoudistianos, S., Theodoratos. D., and Sellis. T.(1998). Experimental evaluation of data warehouse configuration algorithms. Proceedings of the ninth International Workshop on Database and Expert systems Applications, pp:218-223.

[24] Mohania, M., Konomity. S., Kambayashiz. Y., and Vincentx. M.(1999). Designing view maintenance algorithm in data warehousing environment. Proceedings of the ninth International conference on Management of data(COMAD), pp:117-133.

[25] Horng, J.T., Chang. Y.J., Lin. B.J., and Kao. C.Y. (1999). Materialized view selection using genetic algorithms in a data warehouse system. Proceedings of the Congress on Evolutionary Computation, pp:22-27.

[26] Theodorates, D., Dalamagas. T., Simitsis. A., and Stavropoulos. M. (2001). A randomized approach for the incremental design of an evolving data warehouse. Proceedings of the Twentieth International conference on Conceptual Modeling, pp:325-338.

[27] Mistry, H., Roy. P., Sudarshan. S., and Ramamritham. K. (2001). Materialized View selection and maintenance using multi query optimization. ACM SIGMOD, pp:307-318.

[28] Goldstein, J., and Larson. P.A. (2001). Optimizing queries using materialized views: A practical, scalable solution. ACM SIGMOD, pp:328-339.

[29] Valluri, S.R., Vadapalli. S., and Karlapalam. K. (2002). View relevance Driven Materialized view selection in data warehousing Environment, IEEE Computational Society, USA, pp:187-196.

[30] 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.

[31] Zhang, C., Yao. X., and Yang. J. (1999). Evolving materialized views in data warehouse. Proceedings of the 1999 congress on Evolutionary Computation, pp:829-829

[32] Dhote, C.A., and M.S. Ali (2009). Materialized View Selection in Data Warehousing: A Survey. Journal of Applied Sciences, pp:401-414.