D W Q

Foundations of **D**ata **W**arehouse **Q**uality

National Technical University of Athens (NTUA)
Informatik V & Lehr- und Forschungsgebiet Theoretische Informatik (RWTH)
Institute National de Recherche en Informatique et en Automatique (INRIA)
Deutsche Forschungszentrum für künstliche Intelligenz (DFKI)
University of Rome «La Sapienza» (Uniroma)
Istituto per la Ricerca Scientifica e Tecnologica (IRST)

S. Ligoudistianos, D. Theodoratos, T. Sellis

Experimental Evaluation of Data Warehouse Configuration Algorithms

Proc. of the 9th DEXA Workshop (DEXA'98),

Vienna, Austria, August 1998.

DWQ : ESPRIT Long Term Research Project, No 22469
**Contact Person : Prof. Yannis Vassiliou, National Technical University of Athens,
15773 Zographou, GREECE Tel +30-1-772-2526 FAX: +30-1-772-2527, e-mail: yv@cs.ntua.gr**

Experimental Evaluation of Data Warehouse Configuration Algorithms*

Spyros Ligoudistianos Dimitri Theodoratos Timos Sellis
Department of Electrical and Computer Engineering
Computer Science Division
National Technical University of Athens
Zographou 157 73, Athens, Greece
email : {spyros,dth,timos}@dblab.ece.ntua.gr

Abstract

A Data Warehouse (DW) can be seen as a set of materialized views defined over relations that are stored in remote heterogeneous database systems. When a query is posed to the DW, it is evaluated locally, using only the materialized views. The DW configuration problem is the problem of selecting an optimal set of views to materialize that answer a given set of queries. The objective is the minimization of the combination of the query evaluation and view maintenance costs. In this paper we report on the experimental evaluation of an exhaustive algorithm and we develop new greedy and heuristic algorithms that expand only a small fraction of the states produced by the exhaustive algorithm. The algorithms are described in terms of a state space search problem. Finally, we report on experimental results and discuss the observed behavior of the algorithms.

1. Introduction

Data Warehouses (DWs) typically provide access to integrated data from a set of remote heterogeneous databases [16]. A DW can been seen as a set of materialized views over the data provided by the distributed heterogeneous databases; these views are used to answer all the queries posed to the DW, thus avoiding accessing to the original sources. On the other hand the materialized views have to be refreshed when changes occur to the data of the sources.

The operational cost of a Data Warehouse depends on the cost of these two basic operations: query answering and the view refreshing. This overall cost of the DW can be reduced not only by the application of optimal query processing and view refreshment methods, but also by a careful selection of the views to be maintained in the DW. For a given set of different source databases and a given set of queries that the DW has to service, there is a number of alternative sets of materialized views that the administrator can choose to maintain. Each of these sets has different refreshment and query answering cost. Trying to select a set of views that minimizes the overall cost is the Data Warehouse Configuration problem.

* Research supported by the European Commission under the ESPRIT Program LTR project No 22469 "DWQ : Foundation of Data Warehouse Quality"

1.1 Related Work

A similar problem is the DW design as it is studied in [17]. In this paper authors propose heuristic approaches that provide a feasible solution based on merging individual optimal query plans. In [3] a formulation of the DW Configuration problem is given using AND/OR graphs. In [8] the same problem is considered for selection-join views with indexes. An A^* algorithm is provided as well as rules of thumb, under a number of simplifying assumptions. Recently many papers address this problem in the context of aggregations and multidimensional analysis [5, 4].

Answering queries using materialized views has also been studied in many papers in the past (e.g. [10] and [1]). In [6] and [7] the authors study the when using both materialized and virtual views. In a context where views are sets of pointer arrays [11] and [12] provide an A^* algorithm that optimizes the query evaluation and view maintenance cost.

1.2 Contribution and Paper Outline

In this paper we extend the work initially introduced in [15]. We re-formulate the DW Configuration problem as a state space search problem based on a representation of views and queries using conjunctions of selection and join atomic predicates. A state is a set of views and a set of the given queries rewritten over these views. As in [15], we look for the state with the minimal cost. We develop a realistic cost model for this problem for query processing and view maintenance. Given that the exhaustive state

space search algorithm of [15] is too expensive, we suggest r-greedy algorithms that prune the state space, and we compare their performance using various criteria. In addition, we develop a new heuristic algorithm which is based on the greedy algorithms and we compare its performance with the r-greedy algorithms. Our results show that unless the new heuristic algorithm visits a very limited number of states, it performs very well. Based on the experimental results and our theoretic analysis, we describe a new algorithm for the DW Configuration problem combining the previous algorithms.

The rest of the paper is organized as follows. In Section 2 we formally define the DW Configuration problem and we introduce the cost model. The problem is modeled as a state space problem in Section 3. In Section 4 we introduce r-greedy and a two-phase heuristic algorithms. Section 5 presents experimental results and discusses on them. We summarize our results in Section 6.

2. The DW Configuration Problem

The DW maintains a set of materialized views defined over a set o remote source relations. All the queries posed to the DW are answered exclusively using the views. Queries and view are defined in terms of the relational model. We will limit here the description of the DW configuration problem to the case of selection-join conjunctive queries without self-joins. Atomic formulas are of the form $x\ op\ y + c$ or $x\ op\ c$, where x, y are attribute variables, c is a constant, and op is one of the comparison operator $=, <, \leq, \geq, >$, but not $\neq$.

We assume a set **R** of source relations $R_1, ..., R_n$, and a set **Q** of queries $Q_1, ..., Q_m$, defined over **R**. The DW keeps a set **V** of views $V_1, ..., V_l$ that answer these queries without accessing the relations in **R**. Therefore, the queries in **Q**, can be rewritten over **V**. $\mathbf{Q^V} = \{Q_1^V, ..., Q_m^V\}$ is a set of rewritings of queries in **Q** over **V**. A *DW Configuration* **C** is a pair $< \mathbf{V}, \mathbf{Q^V} >$.

The DW configuration problem can now be formalized as follows: *Input:* **R**, **Q** a DW operational cost function. *Output:* $\mathbf{C} = < \mathbf{V}, \mathbf{Q^V} >$ with the minimal operational cost.

2.1 Cost Model

The DW configuration operational cost comprises the query evaluation and the view maintenance cost. In [9] a detailed description of the cost model is given. Table 1 shows the input parameters involved.

The cost of answering a query Q is $QC(Q)$. This is the optimal evaluation cost for the query Q. The query execution cost $QC(\mathbf{Q^V})$ of **C** is the weighted sum of the evaluation cost of each query in $\mathbf{Q^V}$:

$$QC(\mathbf{Q^V}) = \Sigma_i (FQ(Q_i) \times QC(Q_i^V)),\ Q_i^V \in \mathbf{Q^V}$$

$MC(V)$ is the maintenance cost of view V. This cost clearly depends on whether a rematerialization or an incremental view maintenance strategy is adopted [12, 2]. Both are supported by our model. The view maintenance cost of **C** is: $MC(\mathbf{V}) = \Sigma_i MC(V_i),\ V_i \in \mathbf{V}$.

Multiple-query optimization [13] and the use of auxiliary views is not taken under consideration in computing the total view maintenance cost.

The operational cost of the DW of **C**, $OC(\mathbf{C})$, is the weighted sum of the query evaluation cost and the view maintenance cost of **C**.

$$OC(\mathbf{C}) = c_q \times QC(\mathbf{Q^V}) + c_m \times MC(\mathbf{V})$$

Factors c_m and c_q, indicate the importance of the query evaluation and view maintenance costs.

Input Parameters	
Q	Set of queries
R	Source relations
$I(R_i)$	Insertions at the source relation R_i
$D(R_i)$	Deletions at the source relation R_i
P_i	Transmission rate between DW and source i
$FQ(Q_i)$	Frequency of query Qi
$\|\|R_i\|\|$	Size in rows of the source relation R_i
$row(R_i)$	Size of each row of the source relation R_i
ps	Page size in DW
rc	Cost required to read a disk block
wc	Cost required to write a disk block
c_q	Query evaluation cost weight
c_m	Materialization cost weight

Table 1

3. A state space search based algorithm

In this section we model the problem as a state space search problem.

3.1 States and Transitions

A state is a DW configuration $< \mathbf{V}, \mathbf{Q^V} >$. A view in **V** has a name V, a set of selection conditions **VS** and a set of join conditions **VJ** ($<V$, **VS**, **VJ** $>$). A view selection condition is a pair of a selection predicate and a source relation $R \in \mathbf{R}$, ($<$ *selection_predicate*, $R >$). A view join condition is a triple comprising a join predicate and two relations $R_1, R_2 \in \mathbf{R}$, ($<$ *join_predicate* , $R_1, R_2 >$). Similarly, a query is a triple $< Q$, **QS**, **QJ** $>$. A query selection condition has a selection predicate and a view V on which the selection is performed ($<$ *selection_predicate*, $V >$), while a query join condition has a join predicate and two views V_1, V_2 ($<$ *join_predicate*, $V_1, V_2 >$). A *selection_predicate* can be TRUE indicating that view V equals relation R or that query Q equals view V. With each state we associate a cost through the DW

configuration operational cost function introduced in the previous section.

We define state transitions based on state transformations. Each state transformation consists of a transformation of the view set and an appropriate transformation of the query set to preserve the answerability of the queries of from the views. Consider a state $<\mathbf{V}, \mathbf{Q}^{\mathbf{V}}>$. The transformations are the following:

Selection elimination:

The new state is produced by the elimination of one selection conditions from a view in **V** and the addition of an associated selection conditions into the queries that are defined over this view.

Join elimination:

The new state is produced by the elimination of one join conditions from a view *V* in **V**. This may result in the splitting of the view in two other views V_1 and V_2. If there is no view splitting, a selection conditions is added to each query defined over *V*. If a splitting occurs, a join condition is added to these queries. Query selection and join conditions of defined over *V* also needs an appropriate modifications in order to be defined over V_1 or V_2.

View Merging :

A selection (join) conditions implies another selection (join) conditions if both of them are defined over the same source relations and the former is more restrictive than the latter. A merging of two views V_1 and V_2 can take place if every conditions of V_1 (V_2) implies or is implied by a conditions of V_2 (V_1). In the new configuration, V_1 and V_2 are replaced by a view *V* that is defined over the same source relations and comprising all the implied predicates. All the queries defined over V_1 or V_2 are modified appropriately in order to be defined over *V*. Eliminated selection and join conditions are added to these queries.

There is a transition *T(s,s')* from state *s* to state *s'*, iff *s'* can be obtained by applying any of the three state transformations to *s*. It can been shown that by the application of the above three transformation rules we can get all possible DW Configurations [9].

3.2 An Exhaustive incremental algorithm

The state space of the DW configuration problem is formed by following all feasible transitions starting from the initial state s_0. The view set $\mathbf{V_0}$ of the initial state contains exactly for every $Q_i \in \mathbf{Q}$ a view definition $V_i = Q_i$. The query set $\mathbf{Q_0}^{\mathbf{V}}$ of the initial state contains exactly for every view name V_i in $\mathbf{V_0}$ a query having $\mathbf{QJ_i} = \varnothing$ and $\mathbf{QS_i} = \{<TRUE, V_i>\}$.

The exhaustive algorithm systematically examines all the states of the state space. If there are no implications between the atomic formulas of the queries, the state space contains 2^n states, where *n* is the total number of the selection and join atomic formulas in **Q**. The number of states grows with the number these implications. Figure 1 shows the number of states produced from a set of queries having 10 selection and join atomic formulas in total, as a function of the number of implications between them.

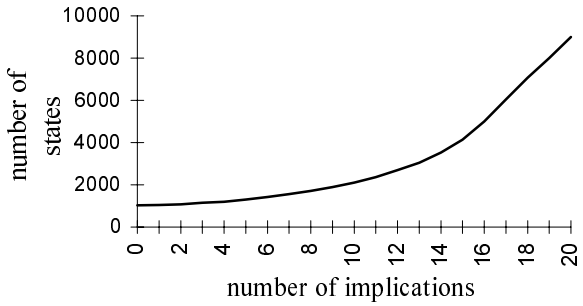

Figure 1 : The size of the state space

4. Greedy and Heuristic algorithms

Since the state space described above increases exponentially with the size of the input, the necessity for search space pruning is apparent. In this section we address this problem by suggesting greedy and heuristic algorithms.

4.1 The r-Greedy Algorithm

Our first attempt for faster algorithms is an r-greedy algorithm. This algorithm first considers state s_o and works as follows: when a state is considered, all the states to depth *r* are expanded and the one having the lowest cost is chosen for consideration. The algorithm stops when there is no more states for expansion.

This algorithm examines only a small part of the state space. The 1-greedy algorithm examines $(n\times(n+1))\div 2+1$ states in the absence of implications where *n* is the total number of selection and join atomic formulas in **Q**.

The r-greedy algorithm suffers in that it is not able to perform all the possible view merging transformations. This is due to the following reason: the application of a view merging transformation requires every atomic formula in view V_1 implies or is implied from one of view V_2. This means that the view merging transformation often can be applied only after a sequence of selection and join elimination transformations. Further more, the r-greedy algorithm sometimes eliminates atomic formulas that are involved in implications, making impossible the view merging transformation. The view merging transformation results in states having lower maintenance cost because a view is eliminated.

4.2 Heuristic Algorithms

Based on the greedy algorithms, we developed and implemented a heuristic that searches a small fraction of the state space and reports a sub-optimal solution. This heuristic increases the possibility to apply the view

merging transformation, thus reducing the problems reported previously. The new algorithm avoid the elimination of an atomic formula involved in an implication if other atomic formulas are not involved in an implication.

The heuristic can be applied to any of the different versions of the r-greedy algorithm. In the rest of our paper we apply it to the 1-greedy version. As we experimentally tested in Section 5, the heuristic algorithm examines a very limited number of states (typically an average 30% of the number of states examined by the 1-greedy algorithm). The more interesting result is that this heuristic performs well in the cases where the 1-greedy performs unacceptable and vice versa. When a small number of implications exists, r-greedy performs in acceptable levels while the heuristic does not. As the number of implications increases the r-greedy performs worse and the heuristic algorithm becomes the winner. This observation, in conjunction with the small number of the states the heuristic examines leads us to the following two-phase heuristic algorithm: For a given input apply both algorithms and choose the solution having lower cost.

5. Experimental Results

In this work we experimentally investigated the performance of the exhaustive, r-greedy and two-phase heuristic algorithms with respect to the quality of the solution returned and the time required to find this solution. The quality of the returned solution for each algorithm is the percentage of the operational cost of the optimal configuration reported by the exhaustive algorithm divided by the operational cost of the configuration the examined algorithm reports. The required time of each algorithm is expressed by the number of the expanded states.

The algorithms are compared in terms of the following parameters: (a) the complexity of the query set **Q**, and (b) the physical input parameters. The complexity of the query set depends on two orthogonal parameters: the size of the query set and the overlapping of the queries in the query set. The size of the query set is expressed by the total number of atomic selection or join conditions of all the queries in **Q**. The query overlapping is expressed by the total number of implications between the atomic formulas of the different queries.

In our experiments we simulate a DW environment where source relations and the DW are stored in separate systems. The source relations used and relation sizes are taken from the TPCD benchmark [14]. Query sets and other cost parameters are generated randomly. The evaluation cost for input queries and the queries needed for the maintenance of the view is computed by finding the optimal execution plan. The operational cost of each state is computed incrementally since, along a transition, we compute the view maintenance cost and the query evaluation cost just for the views and the queries that are affected by the transformation.

5.1 Experiment 1: Performance study when the number of condition varies

We study the performance of the r-greedy and the two-phase heuristic algorithms when the total number of atomic formulas in **Q** varies. In each query set the number of atomic formula implications is 50% of the total number of atomic formulas.

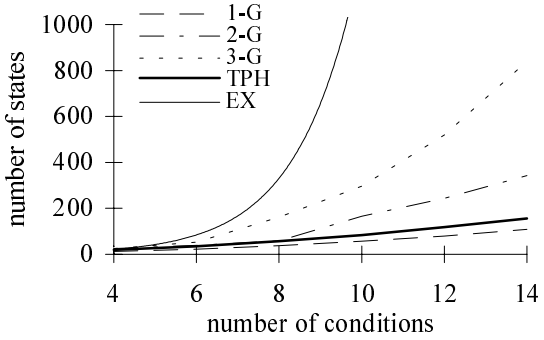

Figure 2 : Number of states vs size of input

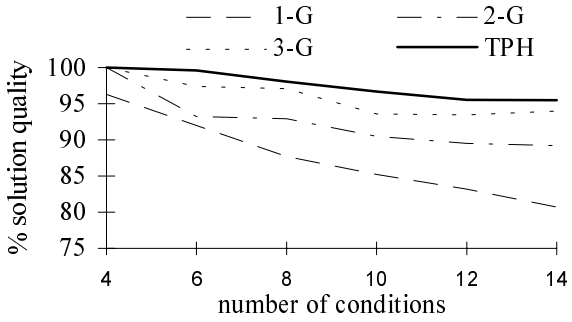

Figure 3 : Quality of solution vs size of input

Figure 2 shows the number of the examined states as a function of the total number of atomic formulas in **Q**. Figure 3 reports the quality of solutions. The diagram of Figure 2 shows that the exhaustive algorithm cannot be applied in realistic cases. The 2-greedy and 3-greedy algorithms also expand a large number of states. The 1-greedy and the two-phase heuristic algorithms expand a very limited number of states and this number is increasing slowly with the number of atomic formulas. On the other hand the quality of solution yielded by 1-greedy decreases significantly when the number of condition increases. The quality of solution yielded by the 2-greedy, the 3-greedy and the two-phase heuristic algorithm decreases but much more slowly when the number of conditions increases.

5.2 Experiment 2: Performance when the overlapping of the queries varies

In this experiment we study and compare the algorithms performance when the implications between atomic formulas vary. We studied the algorithms for different sets of queries that have a total number of atomic formulas equal to 10. The number of implications between atomic formulas of different queries varies from 0 to 15.

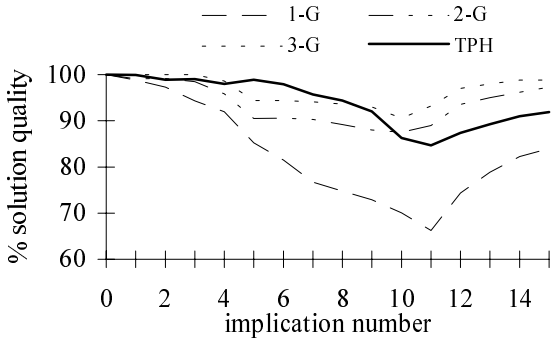

Figure 4 : Quality of solution vs extent of overlapping

Figure 4 shows the quality of the solution as a function of the number of implications for the different algorithms examined. As we see all algorithms give almost the optimal solution if there are no implications. Increasing the number of implications reduces the quality of the solutions. This reduction is up to a certain number of implications. When the number of implication exceeds this number (critical point) the quality of the solution of all algorithms starts improving. This phenomenon is explained as follows: the presence of an implication between atomic formulas affects the sequence of transformations to the optimal solution if this one contains a merged view. The r-greedy algorithm in many cases cannot perform view merging. As the number of implications increases, the number of cases where view merging can be performed increases too thus the solution optimality declines. The two-phase heuristic is also affected but it performs more view mergings than the r-greedy. When the number of implications grows up, only a few atomic formulas not involved in implications remain. So the r-greedy and the two-phase heuristic algorithms perform more view mergings, thus finding solutions near the optimal.

In this experiment we deduce that the r-greedy algorithms reduce their optimality by a constant factor, the two-phase heuristic algorithm reduces its optimality by a very limited factor in the area of a small number of implications and with a higher factor later. All the algorithms increase their optimality after a certain area closely after the critical point. Near the full implication between atomic formulas, the heuristic algorithm is approximated by the 1-greedy. We notice that the two-phase heuristic performs almost always better than the 2-greedy and 3-greedy when the implications are less than the critical point and worse when the implications is greater than this number.

5.3 Experiment 3: Performance when the physical parameters vary

Modifications of the system parameters of Table 1 affect both the view maintenance cost and the query evaluation cost. We express such modifications through ratio c_m / c_q. Parameter c_q is the weight of the query evaluation cost and parameter c_m is the weight of the view maintenance cost in forming the DW operational cost.

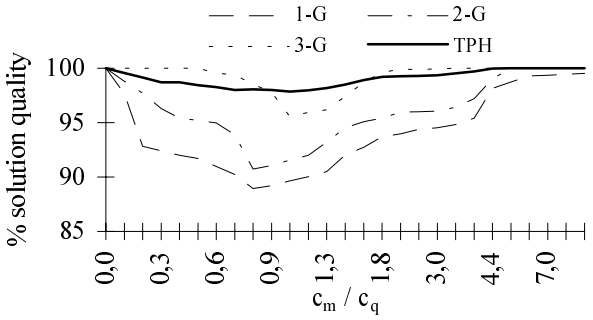

Figure 5 : Quality of solution vs physical characteristics

Figure 5 presents the quality of the solution obtained by the algorithms as a function of the parameters ratio c_m/c_q. If this ratio is less than 1 the query evaluation cost is more significant while when it is greater than 1 the view maintenance cost is more significant. In this experiment the input set **Q** has 10 atomic formulas and the number of implications equals 5. In the extremes (when $c_m/c_q \rightarrow 0$ or $c_m/c_q \rightarrow \infty$) all the algorithms find the optimal solution. The two-phase heuristic algorithm performs better than the 2-greedy algorithm in all cases and better than the 3-greedy algorithm when c_m approximates c_q and until $c_m < 2 \times c_q$. Two-phase heuristic performs better if $c_m > c_q$ because it allows for more view merging which decreases significantly the operational cost when the view maintenance cost is high, giving a solution near the optimal.

5.4 Discussion of results

From the experimental results of the previous subsections we observe the following. The state space of the DW configuration problem is huge so is unreal for the exhaustive algorithm to be used. The 1-greedy algorithm examines a very limited fraction of the search state space, but it performs unacceptably, unless there is a very limited number of implications. The two-phase heuristic gives

acceptable results if there is not too many implications, expanding a very limited number of states while its performance is not affected significantly by changes to the physical parameters.

From a complete analysis of the solutions that we get from the studied algorithms we reach the following conclusion: Let us suppose that we have two queries Q_1, $Q_2 \in \mathbf{Q}$. The presence of Q_2 affects the rewriting Q_1^V of Q_1 at the optimal configuration iff there is at least one implication between atomic formulas of Q_1 and Q_2. If there is no implication between selection join conditions of Q_1 and Q_2, the presence or the absence of Q_2 never affects the rewriting of Q_1 in the optimal solution. As we see from Experiment 1, the number of states that the algorithms expands is a function of the total number of the selection/join conditions of the queries. We can therefore improve the performance of the algorithms as follows: if we have to find the configuration of a DW given a set of queries **Q** which contains a great number of queries, we can group these queries Q in clusters, each cluster containing queries having implications between their atomic formulas. By clustering the input queries, the repetitive execution of the configuration algorithms on each query cluster, result in the same solution while expanding a smaller number of states. From Experiment 2 we can deduce that if a query clusters contain just one query the 1-greedy algorithm is sufficient for finding the optimal configuration. For other query clusters the two-phase heuristic algorithm provides solutions that approximates the optimal one.

6. Summary

In this paper we reported on the experimental evaluation of algorithms that solve the DW Configuration problem extending the research done in [15].

We re-formulated the DW Configuration problem as a state space search problem and we developed a realistic cost model. We used an exhaustive algorithm and we designed an r-greedy and a two-phase heuristic algorithm that give an approximation of the optimal solution. We implemented the algorithms and we investigated the performance with respect to the quality of the solution returned and the time required to find this solution. The results obtained show that the two-phase heuristic algorithm expands a limited fraction of the state space the exhaustive algorithm expands, giving solutions that approximates in most cases the optimal solution the exhaustive algorithm gives. The r-greedy algorithm has to expand significantly more states in order to approximate the quality of the solution the two-phase heuristic gives.

Interesting extensions of the present work include the following: (a) The introduction of space constrains. (b) The use of auxiliary views in the maintenance process of other views. (c) The enlargement of the class of queries to include aggregate queries.

References

[1] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim: Optimizing Queries with Materialized Views. *In Proc. of the ICDE Conference*, pages 190-200, 1995

[2] A. Gupta I. Mumick : Maintenance of materialized views: Problems, technics and applications. *IEEE Data Engineering*, 18(2): 3-18, 1995

[3] H. Gupta: Selection of Views to Materialize in a Data Warehouse. *In Proc. of the ICDT Conference*, pages 98-112, 1997

[4] H. Gupta, V. Harinarayan, A.Rajaraman, J.D. Ullman. Index Selection for OLAP. *In Proc. of the ICDE Conference*, pages 208-219,1997

[5] V. Harinarayan, A.Rajaraman, J.D. Ullman: Implementing Data Cubes Efficiently. *In Proc. of the ACM SIGMOD Conference*, Pages 205-26, 1996

[6] R. Hull, G. Zhou : A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches. *In Proc. of the ACM SIGMOD Symp. On the Management of Data*, pages 481-492, 1996

[7] R. Hull, G. Zhou: Towards the Study of Performance Trade-offs Between Materialized and Virtual Integrated Views. *In Proc of the Workshop on Materialized Views VIEW* pages 91-102, 1996

[8] W. Labio, D. Quass and B. Adelberg : Physical Database Design for Data Warehousing. *In Proc. of the 13th ICDE Conference.* 1997

[9] S. Ligoudistianos, D. Theodoratos, T. Sellis : Data Warehouse Configuration Algorithms. *Technical Report , Knowledge & Data Base System Lab., Electrical and Computer Eng. Dept. NTU Athens*, 1998

[10] P.-A. Larson and H. Yang. : Computing Queries from Derived Relations. *In Proc. of the VLDB Conference*, pages 95-104, 1985

[11] N. Roussopoulos : View Indexing in Relational Databases. *ACM Transactions on Database Systems, 7(2):258-290,* 1982

[12] N. Roussopoulos : The Incremental Access Method of View Cache: Concept Algorithms and Cost Analysis. *ACM-Transactions on Database Systems*, 16(3) 535-663, Sept 1991

[13] T. Sellis : Multiple Query Optimization ACM *Transactions on Database Systems*, 13(1), 23-52, 1988

[14] Transaction Processing Performance Council : TPC Benchmark D (Decision Support) Standard Specification Revision 1.2.2, 1996

[15] D. Theodoratos and T. Sellis : Data Warehouse Configuration. *In Proc. of the VLDB Conference*, pages 126-135, 1997

[16] J. Widom : Research problems in data warehousing. *In Proc. of the CIKM Conference, pages* 25-30, 1995

[17] J. Yang, K. Karlapalem, Q. Li : Algorithms for Materialized View Design in Data Warehousing Enviroment. *In Proc. of the VLDB Conference*, pages 136-145, 1997