

Evaluation Techniques of Optimization based on Materialized views in decision supporting systems using PayOffRatio metrics

Mohamed Ridani^{1,*}, Mohamed Amnai¹, Ali Choukri¹, Youssef Fakhri¹, Noredine Gherabi²

¹*Laboratory of Research in Informatics (LaRI), Faculty of Science, Ibn Tofail University, Kingdom of Morocco*

²*Lasti Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University, Kingdom of Morocco*

Abstract Query optimization presents significant issues for the majority of decision support systems deployed in the world today. These systems require data warehouse structures to store and process data that should be used by online analytical processing (OLAP). Additionally, the volume, variety, and quick-changing nature of the data in the data warehouse are essential, necessitating additional effort, research to keep the data current, and the least expensive data updates. In this article, we propose a new metric called PayOffRatio based on the principle of view selection gain to address the problem of selecting materialized views in multidimensional data warehouses. The metric examines the response time of the materialized view relative to the execution time of its underlying query before view creation, taking into account the frequency of its use and its implementation time. The metric was first evaluated using the standard greedy algorithm, where view selection depends only on the associated metric, and then using the greedy algorithm with dependencies between queries at adjacent levels, which is the standard approach used in most schemas developed to improve the performance of decision support systems. Our experimental results confirm that generating the selection of materialized views using our PayOffRatio metric is the most efficient. In addition, a comparison of our metric with other metrics used in the field will be presented.

Keywords Hypercube, PayOffRatio, Data Warehouse, OLAP, Materialized Views Selection

DOI: 10.19139/soic-2310-5070-4663

1. Introduction

Due to rapid technological advancements, the wide range and volume of related data are continually increasing. The current rapid global changes in data warehouse, data are making decision-making systems increasingly complex and the analysis process increasingly difficult. However, the time spent examining workloads using frequently sophisticated queries is also very crucial. The apparent solution to this issue is to examine every analytical query that is used and save the explicit results in a data warehouse; However, the cost of storage, computing time, and updates for this system is very high. To meet this challenge, several approaches have been developed in this research area and have proposed to take advantage of the dependencies between queries by materializing an optimal subset that meets all workload queries with an optimal cost; hence the emergence of the materialized view selection problem. However, several questions can arise, such as: Which views are selected to be materialized? How did you define the criteria to choose these views? How to evaluate the results of a given selection?

*Correspondence to: Mohamed Ridani (Email: mohamed.ridani@uit.ac.ma).Laboratory of Research in Informatics (LaRI), Faculty of Science, Ibn Tofail University, Campus Universitaire, Kenitra, BP 133, Kingdom of Morocco

Furthermore, the approaches invented to present the solution to all the above questions, to outperform this challenge and degradation of performance, always use metrics based on the characteristics of each query in the workload. These metrics, such as the frequency, the execution cost, and the update costs of each query, are evaluated individually or together. For example, the authors in [1] deal with the lattice view as a base structure tagged with the view's size, which is often measured by the number of rows in the view. However, [2, 3] also deployed the lattice view structure to select materialized views as a bi-objective optimization problem, which tries to reduce the size of Candidate and not candidate view, respectively, CMV and CNMV members. Moreover, the authors in [4] proposed the ECHTDEMVS method to minimize the fitness function, where the number of lines defines the size.

The papers [5, 6] summarized the same approaches using various metrics. The basic metric in almost all of those approaches is the number of rows to access in the cube corresponding to the query, which determines the cost of responding to the query, as described in [7]. Moreover, in [8] each particle in the swarm population has knowledge about its position $x_{id}(t)$, velocity $v_{id}(t)$, personal best position $p_i(t)$ and the global best position $p_g(t)$ used also lattice view structure tagged by the size of view which defined as the number of lines. Furthermore, the authors in [9] use MVPP as a base structure to create the subset views to find an optimal solution minimizing the query processing and view maintenance costs. In [10], the construction cost of each view is defined by the number of created rows of its corresponding relational operator. The metrics in the total cost of all selected views determine the value of the cost function, which is the sum of their corresponding query processing and view maintenance costs [11] using the MVPP structure.

In addition, [12] uses RDF graphs to generate the benefit of the materialized views selection problem. The associated metrics have been calculated based on the processing cost of the query without a view, which is subtracted from the query processing cost of the query using the materialized view and the maintenance cost of the view, where the size of the view is defined as the cost needed to store the view. Indeed, under the related work section, we will provide a detailed description of the fundamental metrics used by each approach.

The purpose of this paper is to present a new approach using metrics based on the "PayOffRatio" principle to evaluate the views to be materialized, like various proposed algorithms on the materialized views selection problem invented in multidimensional data warehouses, to build and optimize the query under all cuboids of the hypercube forming the global workload. The overall goal of all research in the field of data warehouse optimization is to maximize the benefits while taking into account the cost and response time of various techniques. In order to accomplish this goal, our suggested method would continue to produce a four-phase framework based on the PayoffRatio metrics listed below.

1. The initialization of workload phase: In this step, we try to delimit the workload by enumeration of each query in order to specify the components: the attributes, dimensions and measures.
2. The preliminary evaluation phase: query evaluation, which consists of the generation of our new metrics "PayOffRatio", calculating the frequencies (execution, update), and all costs that could be used in the evaluation phase.
3. Generation lattice phase: definition of parameters and construction of the lattice.
4. The evaluation of algorithms phase: execution and interpretation of results. Figure 1 presents the architecture of Our Proposed Framework.

The figure 1 presents the architecture of Our Proposed Framework.

The rest of this paper is organized as follows: in the second section, we define the metrics overview of the materialized view selection problem; the third section formulates the metrics of our model and describes

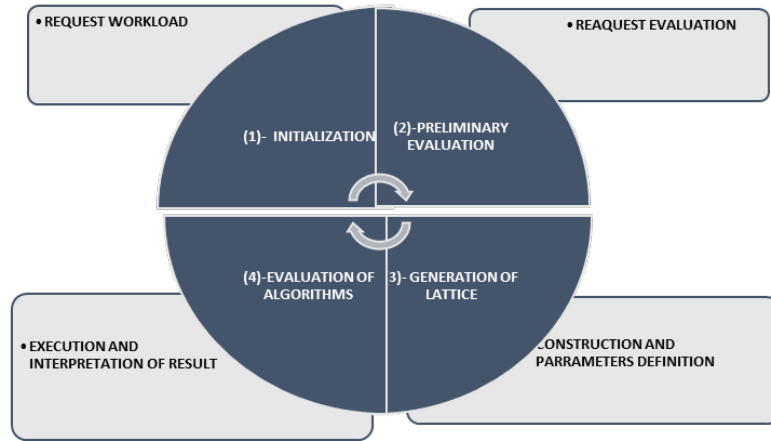


Figure 1. : **Architecture of Our Proposed PayOffRatio Process**

each phase in the proposed process, followed by our experimental results with some interpretations. Finally, in the last section, we will give some perspectives and conclusions.

2. Metrics overview of materialized views selection problem

In fact, a number of studies have been published in the literature that concentrate on the models and designs used to store data in data warehouses. Some works, like [13, 14, 15], deal with relational models, while others, like [10, 16], consider online analytical processes (OLAP) that employ multidimensional models. In fact, the challenge for business intelligence systems developers is to reduce query response times, storage limitations, and the updated process created by continuously expanding data, as well as the format and variety of data in data warehouses.

For the majority of decision support systems worldwide, including those that rely on a data warehouse structure to store all the data to be analyzed, the query optimization process is a significant difficulty. Therefore, many approaches have been developed in the last two decades dealing with this optimization problem; these approaches differ in the algorithm used to select a subset of queries to materialize, but the majority of these approaches rely on the same metrics to evaluate the cost of the selection process. The following paragraph describes a list of the approaches analyzed according to the basic metric used during the selection process. This section will address the issue of materialized view selection by listing several methods based on the most popular metrics used in data warehousing. The selection of quotations is straightforward and does not evaluate the caliber of the approach.

The Greedy algorithm was proposed by Harinarayan V. et al. in [14] and is based on space costs that are determined by the number of rows in each view v . The algorithm's basic idea is to choose a set S of k views that maximizes the benefit $B(v, S)$, which is defined as follows.

$$B_w = C(w) - C(v) \text{ if } C(w) < C(v), \text{ Otherwise } B_w = 0 \quad (1)$$

$$B(v, S) = \sum_{w < v} B_w \quad (2)$$

This approach uses the linear Cost Model and defines the cost of answering Q as the number of rows present in the ancestor materialized table for query QA that was used to construct Q . Therefore, minimizing the cost involves the selection of a query with the least number of rows.

Lin W.-Y, and I.-C. Kuo in [17] proposed the Greedy Genetic Algorithm based on two metrics: the Query Cost describing the cost to evaluate a query, and Maintenance Cost specifying the cost to update the cube, where the maintenance technique adopted is precomputation from scratch of materialized cubes.

$$\mu(*) = \sum_{i=1}^k f_{q_i} E(q_i, M) + \sum_{c \in M} g_c U(c, M) \quad (3)$$

The first Member presents the cost of evaluating the query q_i , and the second member presents the cost of updating cube c . In this paper, the cost of evaluating the query and maintenance cost have been calculated using the size of the least ancestor, such as the number of tuples in the view, the evaluation frequency, and the insertion frequency into the fact table. However, in our work, we use the global time of technique implementation.

Vijay Kumar T., and Kumar S., in [18], proposed the Genetic Algorithm Materialized view selection. In this approach, the genetic algorithm used by the authors is similar to the algorithm proposed by Lin, W.-Y. and I.-C. Kuo in [17], but the differences appear in the fitness function and the probability added to the crossover and mutation functions. The fitness function is defined by a total view evaluation cost (TVEC) that calculates the summation of the size of materialized views and the size of the smallest ancestor for not materialized views, described as follows:

$$TVEC = \sum_{i=1 \cap SMV_i=1}^N Size(V_i) + \sum_{i=1 \cap SMV_i=0}^N Size(V_i) \quad (4)$$

In the above formula, the authors have defined the size by the number of rows from both materialized views and the smallest ancestor. Thus, the number of lines is the basic metric that is also used for their approach.

Anjana Gosaina, Heenaa in [7] proposed a linear cost model for the materialized view selection problem based on the Particle Swarm Optimization algorithm. An arbitrary population of N candidate particles is used to initialize the model. Every particle in a space with D dimensions has a certain speed and position, which it continuously modifies according to the best position discovered by the population as a whole -global best- *gbest* thus far, and by itself -particle best- *pbest*. For further information, see [7]. The authors contrasted the results of the evolutionary algorithm with the cost of query-processing under space constraints.

The number of rows that must be accessed in the relevant cube for the query determines the cost of replying to it. Additionally, the technique begins with similarity parameters that were employed in the genetic algorithm by Lin, W.-Y., and I.-C. Kuo in [17]. However, the fitness function is where these two methods diverge: Anjana Gosaina employs the first element of the function that W.-Y. and I.-C. Kuo use.

In [1], Prakash J. and Kumar T. V. suggested a multi-objective method to choose materialized views. The authors employed MVSVEGA, a non-Pareto-based genetic algorithm, to choose Top-K views for materialization from a multidimensional lattice that simultaneously minimizes the two costs of TVEC, CMV (Candidate Materialized Views), and CNMV (non-candidate materialized views), which are based on the size defined by the number of lines as the fundamental metric.

A game theory-based framework for materialized view selection in data warehouses was suggested by Azgomi H. and Sohrabi M.K. in [19], taking into account the two opposing players in the game: processing cost and view maintenance cost. The game employed two measures, which are determined by

the number of rows in the view: the largest maintenance cost and the greater query processing costs.

A novel coral reefs optimization approach for materialized view selection in data warehouse environments was proposed by Azgomi H. and M. K. Sohrabi in [10]. The intended technique utilized a bit-string with length equal to the number of views of the MVPP as the beginning population of the algorithm and presented all views using the MVPP structure. Depending on whether or not the view is a part of the solution, each bit-string case takes either 1 or 0.

In [20], Azgomi H. and Sohrabi M. K. suggested a Map-Reduce-based method for building multiple views Processing Plan MVPP structure using the Map-Reduce Model and the hashing technique, which reduces the search space of a problem in data warehouses for large data applications and may process Plan MVPP structures in an acceptable amount of time. $MAP_{function}$ and $REDUCE_{function}$ are the two primary functions that make up the map-reduce concept. The MAP function generates a collection of intermediate (key, value) pairs by processing two (key, value) inputs. The REDUCE_{function} generates the outcome by integrating all intermediate data associated with each key. The MVPP structure has been shown and created n MVPPs from n! MVPPs with the lowest cost was chosen as the ultimate solution out of all the potential MVPPs. These techniques were $O(n)$, where n was the query count. The sum of the query processing and view maintenance costs for each view determines the overall cost of the MVPP.

The exception presented by some clustering-based optimization algorithms, such as [19, 21], is that clusters are created according to the similarity of attributes used by the views, regardless of the size of source objects.

Table 1 below summarizes, by comparing some methods from the literature, their Strengths and Weaknesses according to several criteria, namely the cost model used to select the views to materialize, the nature of the reference benchmark, and the metrics used to evaluate all workload queries.

As indicated in the summarized table 1 above, almost all approaches focus their studies on minimizing the cost of evaluating either query execution or view maintenance, or both, using various types of deterministic or non-deterministic algorithms. However, none of them have attempted to quantify the gain from using materialized views compared to evaluating all workload queries without view materialization. It is within this framework that we introduce our *PayOffRatio* gain metric as the basis of our model for selecting which views to materialize, initially using two variations of the greedy algorithm. The foundation of our approach will be described in the following section.

3. Formulate the metrics of our model.

3.1. Definition of PayOffRatio Metric :

In the previous approaches, the goal is to find metrics that efficiently evaluate the workload and can be used to find the optimal set of queries to be materialized in order to minimize the response time under the space constraints. In the same work, the authors try to calculate the benefits using the optimization algorithms. Moreover, this benefit is calculated using the cost model based on the number of lines of each view to be materialized or to be indexed, but nevertheless, these approaches used the metrics based on the *PayOffRatio*, which determines the cost of implementation of the approach itself and the gain of using the metrics. For this reason, we proposed in our approach to use this novel metric, which is modeled as follows, to evaluate the optimization algorithms.

- Let $Q = [q_1, q_2, \dots, q_n]$ be a set of workload query, to which is associated the list of metrics described below.
- $F = [f_{q_1}, f_{q_2}, \dots, f_{q_n}]$ is the set of workload query frequencies, f_{q_i} presents the execution frequency of a query q_i .

Paper	The Cost Model	Used Algorithm	Basic structure	Benchmark	Metric	Strengths	Weaknesses
Haniragoun V., et al. in [14]	$Maximize BF(u, s) = \sum_{s \in S} BF_{u,s}$ $BF_u = Cost(u) - Cost(s)$	Greedy Algo.	Lattice	TPC-D benchmark	$Spacecost = Number\ of\ rows\ in\ the\ view$	Basic simplest Linear Cost Model	$Response\ time = Size\ of\ View$
Lin W.-Y. et L.-C. Kuo in [17]	$Minimize \mu() = \sum_{i=1}^n f_{i,c} Exec(c_i, M) + 9s \sum_{c \in M} U_{pb}(c, M)$	Genetic Algo.	Lattice	TPC-D benchmark	Size of view (Mo)	Take into the frequency evaluation and Maintenance cost	Maintenance Technique : Mutation and crossover recomputed from scratch
Vijay Kumar T., et Kumar S. in [18]	$Minimize \mu() = \sum_{i=1}^n S(V_i) + \sum_{i=1}^n SS4(V_i)$	Genetic Algo.	Lattice	Private Database	Size of view (Mo)	Algorithm provides optimal solutions for complex problems	The criteria for choosing the optimal solution are the mutation probabilities and the optimal initial number of generations that lead to the optimal solution
Anujou Goswaha et Hecual in [7]	$Minimize \mu() = \sum_{i=1}^n f_{i,c} E(c_i, M)$	Particle Swarm Optimization algorithm	Lattice	TPC-H	Size of view (Mo); number of rows.	The algorithm takes account of frequency evaluation of view	The Cost of Materialized view itself is not considered
Prakash J. et Kumar T. V. in [1]	$Minimize \mu() = \sum_{i=1}^n S(V_i) + \sum_{i=1}^n SS4(V_i)$	Strength Pareto Evolutionary Algorithm	Lattice	Private Database	Size of view (Mo)	Bi-objective optimization: Minimize the cost of materialized view and the non-materialized views	C_{mv} and C_{sv} are proportional and their separate evaluation does not guarantee optimization of the total cost
Agoumi H., et Sultani M.K. in [19]	$Minimize DICO() = \frac{\sum_{i=1}^n Dico(O_i, O_j)}{K}$	K-means ALGO	Cluster with COA and CA	Star Schema (Private Database)	Size of views and Processing Time (Milliseconds)	The method is more efficient than the other methods storage constraint. Introduction of CPU in evaluation process	The criteria for choosing the initial value of the algorithm in relation to the number of workload requests. Moderate performance
Agoumi H., et M. K. Sultani in [10]	$Minimize \mu() = \sum_{i=1}^n (\sum_{j \in Q} e f_{i,j} * C_j^2 + \sum_{j \in Q} w_j r * C_j^2)$	CHROMS: Coral reefs optimization algorithm	Multiple View Processing Plan (MVPP) structure	RTLIDS dataset and TPC-H	Query Processing	The method has the highest coverage rate	for execution time: Not suitable for cases where the method is cost and the solution cost are crucial.
Agoumi H., et Sultani M. K. in [20]	$QC_i = f_{th} * \sum_{i=1}^n \{ (f_{iQ} * e f_{i,j} * w_j) / v_j \}$ And Hash Function $h(K)$	MR-MVPP algorithm	Map-reduce model and the Hashing technique	TCP-H dataset	Similarity criterion	Better execution time (Reduce the Cost of accessing to the views) and the Coverage Rate = 50%.	Reduce the Cost of accessing the views
Prakash, Jay et al in [2]	$Minimize C_{MV} = \sum_{i=1}^n \frac{S(V_i)}{SS4(V_i)}$ $C_{NMV} = \sum_{i=1}^n \frac{SS4(V_i)}{SS4(V_i)}$	Non-dominated Sorting Genetic Algorithm For Multi-objective optimization : NSGA-II	Lattice	Private Database	Size of view (Mo)	Algorithm provides optimal solutions for complex problems	The criteria for choosing the optimal solution are the mutation probabilities, the optimal initial number of generations that lead to the optimal solution
Kumar Amit et al in [8]	$Minimize TV\ EC = \sum_{i=1}^n \frac{S(V_i)}{SS4(V_i)} + \sum_{i=1}^n \frac{SS4(V_i)}{SS4(V_i)}$	Discrete Genetic Operator based Particle Swarm Optimization (DGPSSO)	Lattice	Private Database	Size of view (Mo)	The solution take into account in the evaluation function also the size of smallest materialized ancestor for not materialized view	Criteria choosing initial parameters of algorithm such as Number of generations, Inertia weight, Cognitive acceleration constant, and Social acceleration constant
Belkoui Nabila et al in [12]	$Minimize Bone\ H(V) = cost_{wv}(q_i, V) - cost_{wv}(q_i, V) - cost_{Mater}(V)$	Near Real Time Data Warehouse design Using ETL Algorithm	RDF Graph structure	Star Schema (SSB) Benchmark (UBM) benchmarks	Query Processing and Maintenance Cost (Time Cost)	Identifies the SPARQL queries to speed-up the execution of backend ETL process	FrontEnd: OLAP process not Evaluated
Molasesi Molsen et al in [11]	$Minimize C(S) = \sum_{i \in S} (f_i * PC_{c_j}(c_j) + \sum_{j \in S} f_j * MC_{c_k}(c_k))$	non-deterministic Simulated Annealing (SA) algorithm	MVPP	RTLIDS Data Warehouse	Query processing (Time of selection, The MVS) and View Maintaining	The SA algorithm accepts, solutions that do not present better values of the objective function with a non-zero probability. The method incorporates maintenance costs into the total cost to be minimized, accounting for the cost of maintenance of data in the warehouse.	Criteria to choose the minimum temperature threshold
Sudadeva Kevtha et al in [4]	$Minimize \mu() = \sum_{i=1}^n f_{i,c} E(c_i, V) + \sum_{i=1}^n y_i * M(f_{i,c}, V)$	Ensembled Constraint Handling Techniques (ECHT); ECHTDEMS Combined with Differential Evolution algorithm	Lattice	TPC-H	Size = Number of Rows		ECHTDEMS used more than 10 initialized parameters, addition to the problem of their optimal value.
Manaei, Maloli et al in [3]	$Minimize \mu() = w_{sgat} f_{r1} (ResponseTime) + w_{sgat} f_{r2} (MaintenanceCost) + w_{sgat} f_{r3} (EnergyUsage)$	Genetic Algorithm	Lattice	TPC-H	Response Time, Cost of maintenance, Memory Usage.	A notable decrease in the optimization process's execution time	the criteria for choosing the crossover and mutation probabilities and the optimal initial number of generations that lead to the optimal solution
Roy Sutanu et al in [21]	$Minimize (Clust)_{avg} = \frac{\sum_{i=1}^n J_{cl}(d_{i,r}, d_{i,r'})}{n}$ where $i \neq j$	Clustering Algorithm	Weighted Connected Graph	Private Database	Weight Values	Very high level of granularity defined by the similarity between the attributes of all queries	Values of cut-off and Similarity determination.

Table 1. A synopsis and comparison of the aforementioned techniques

- $C_B = [c_{b_1}, c_{b_2}, \dots, c_{b_n}]$ is the cost before using the materialized views, c_{b_i} presents the response time of query q_i without MVS.
- $C_A = [c_{a_1}, c_{a_2}, \dots, c_{a_n}]$ is the cost after using the materialized views; c_{a_i} presents the response time of query q_i with MVS.
- $C_{imp} = [c_{imp_1}, c_{imp_2}, \dots, c_{imp_n}]$ is the cost of implementing materialized views using the query q_i , this cost is equal to the sum of view creation time and the view refresh time. In our case, we assume that the views remain unchanged, then the cost of implementation is equal to the time needed for the view creation.

$$PayOffRatio = GR = \frac{Cost(t)}{Delay(t)}, \in]0, 1] \quad (5)$$

* When

$$Cost(t) = (C + \epsilon) \quad , \text{ when } \quad \epsilon = \sum_{i=1}^n \frac{C_{imp_i}}{f_{q_i}} \quad (6)$$

$$C = \sum_{i=1}^n (f_{q_i} * C_{a_i}) \quad , \text{ when } \quad C_{a_i} \in C_A \quad \text{ and } \quad f_{q_i} \in F \quad (7)$$

And

$$\epsilon = \sum_{i=1}^n \frac{C_{imp_i}}{f_{q_i}} \quad , \text{ when } \quad C_{imp_i} \in C_{imp} \quad \text{ and } \quad f_{q_i} \in F \quad (8)$$

$$Cost(t) = \sum_{i=1}^n ((f_i * C_{a_i}) + \frac{C_{imp_i}}{f_{q_i}}) \quad (9)$$

$$Delay(t) = \sum_{i=1}^n (f_{q_i} * C_{b_i}) \quad , \text{ when } \quad C_{b_i} \in C_B \quad (10)$$

And

$$PayOffRatio = GR = \frac{\sum_{i=1}^n (f_{q_i} * C_{a_i} + \frac{C_{imp_i}}{f_{q_i}})}{\sum_{i=1}^n f_{q_i} * C_{b_i}} \quad , \text{ when } C_B \neq \emptyset \quad (11)$$

The cost(t) in the equations (5), (6), (9), represents the total time of the technique, which corresponds to the response time of the queries using the materialized view, and the variable ϵ is defined according to the time required to develop and implement the materialized view in addition to the frequency of handling the request in the workload as shown (6). Indeed, ϵ measures the real cost added to develop the techniques; in other words, This metrics equal to the time required to implement the technique (C_{imp}) divided by the number of times the query is executed under its materialized view (Execution frequency F) see (6),(8). One of the advantages of our approach is that this epsilon metric can group together all other measures that can be taken into account in the process of manipulating materialized views,

namely the additional cost related to their refresh for rapidly evolving systems, etc. C_A in the equation (7) presents the response time using the materialized view selection technique.

On the other hand, More crucial questions regarding to why divide the time by frequency?, This refers to the ratio between the time required to implement the technique and the execution frequency. Moreover, the cost of implementing of the materialized view is equal to the time required to create it when the view is executed only once. However, the creation of the materialized view aims to reduce query response time based on usage frequency. Therefore, its creation time must be proportional to its usage frequency (see (8)). In other words, the cost of implementing the materialized view corresponds to the average execution time.

Furthermore, $Delay(t)$ in the equations (5), (10) represents the response time when the optimization technique is not used. Therefore, the ratio between the $cost(t)$ and $Delay(t)$ quantifies the payoff resulting from the selection of the optimal set of materialized views; this metric defines our new metric proposed to deal with the materialized view selection problem. This metric is called the **PayOffRatio** or **Gain Ratio (GR)**.

As a reminder, our goal is to find a subset of queries to materialize in order to answer all the queries in our workspace. The thinking time after creating the views is strictly less than that without views. Consequently, the ratio of execution time with materialized views to execution time without materialized views is strictly less than 1 and strictly greater than zero.

Consequently, this **PayOffRatio** (5), (11) rate should ideally be between 0 and 1, and the solution moves toward optimality as its value gets closer to zero. As a hypothesis, $PayOffRatio \leq 1$ strictly less than 1, and if this ratio is more than 1, this means that using the materialized view in this particular situation is a loss and should be eliminated throughout the selection process. The table 2 below presents an example of a configuration when the PayoffRatio is greater than 1:

CUBOID_ID	QUERY	FREQUENCY	NOFL	TEBFT (S)	TIT(S)	TEAFT(S)	PAYOFFRATIO
0	Q1	29	113852	106	6	6	0.059
1	Q2	2	59994	5	5	≈ 0	0.5
2	Q3	10	56997	31	6	1	0.052
3	Q4	1	12	3	6	≈ 0	2
4	Q5	38	37998	116	6	2	0.019
5	Q6	46	29997	95	5	2	0.022
6	Q7	43	6	86	5	≈ 0	0.001
7	Q8	1	6	2	5	≈ 0	2.5
8	Q9	38	19998	80	4	1	0.014
9	Q10	2	18999	5	5	≈ 0	0.5
10	Q11	46	4	132	5	≈ 0	0.001
11	Q12	36	3	62	5	≈ 0	0.002
12	Q13	43	9999	76	4	1	0.014
13	Q14	23	2	39	5	≈ 0	0.006
14	Q15	25	2	156	4	≈ 0	≈ 0
15	Q16	20	1	51	4	≈ 0	0.004

Table 2. Example 2 of the Evaluation Configuration.

In this configuration, the generated PayOffRatio in lines 4 and 8 is greater than 1, where the query is run only once ($F = 1$) and the implementation cost is fully penalized, respectively 6.S against 3.S and 5.S against 2.S. Therefore, the queries Q_4 and Q_8 are excluded from the subset to be materialized.

3.2. Initialization Phase :

The initialization of the workload phase: In this step, we try to delimit the workload by enumerating each query to specify the components: the attributes, dimensions, and measures.

Let D be the set of all dimensions used in the data warehouse and R the set of fact tables, which regroup the list of metrics to analyze. $D = [d_1, d_2, \dots, d_i]$ i in $[1, m]$ and m is the number of dimensions in the data warehouse, where $d_I = a_{i1}, a_{i2}, a_{i3} \dots a_{in}$ represents the dimension d_I with n attributes, $R_i = [r_{i1},$

$r_{i2}, \dots, r_{iX}, M_{i1}, M_{i2}, M_{iY} \in \mathbb{R}$ represents the fact table i with X attributes of dimension and Y measures to be analyzed. Furthermore, due to storage concerns and equipment performance restrictions, we examine a data warehouse formed by a start schema with $x=m=4$ dimensions and a number of measurements to be analyzed equal to $1(y=1)$ as shown in Figure 2.

- $D = [d_1, d_2, d_3, d_4] = [\text{dim_customers}, \text{dim_product}, \text{dim_date}, \text{dim_geography}]$.
- $R_i = [\text{id_cus}, \text{id_prod}, \text{id_date}, \text{id_geo}, \text{sales}]$.
- $d_1 = [\text{DIM_CUSTOMERS}] = [\text{id_cust}, \text{first_name}, \text{last_name}, \text{sexe}, \text{categorie}, \text{situation}, \text{birth_day}]$,
- $d_2 = [\text{DIM_PRODUCT}] = [\text{id_prod}, \text{des_prod}, \text{price}]$,
- $d_3 = [\text{DIM_GEOGRAPHY}] = [\text{id_geo}, \text{city}, \text{region}, \text{country}]$,
- $d_4 = [\text{DIM_DATE}] = [\text{id_date}, \text{year}, \text{month}, \text{day}, \text{heure}]$.

The workload queries have been modelled as follows:

$Q_i = [A_1, A_2, \dots, A_m, M_Z]$. Where A_i are the attributes of dimensions, and M_z are the measures in the fact table.

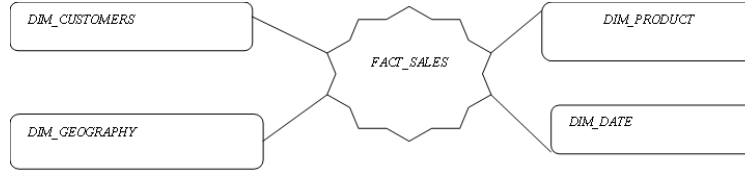


Figure 2. Star Schema base structure of our Data warehouse

3.3. Preliminary evaluation phase :

In the preliminary evaluation phase, the queries have been evaluated in order to generate our new metric, "PayOffRatio". We calculate the frequency of usage of the query, its execution times, and all other costs, which can be used to evaluate the query.

Indeed, Let Q be the set of Q_i workload queries, to which is associated the list of metrics described below. For all queries Q_i , we calculate the **Delay(t)** according to the model described above, see **eq (10)**.

To reach this stage, we will proceed as follows:

In the first step, define the current state by analyzing the workload to be studied.

1. Determine the workload query frequencies $F = [f_{Q_1}, f_{Q_2}, \dots, f_{Q_n}]$, for each query Q_i , calculate the execution frequency f_{Q_i} .
2. Calculate the cost before using the materialized views $C_B = [c_{b_1}, c_{b_2}, \dots, c_{b_n}]$, which presents the sum of the response time c_{bi} of all queries Q_i .
3. Generate the global cost of the workload before materialized view creation **Delay(t)** according to **eq(10)**.

In the second step, calculate the Cost for all queries Q_i , **Cost(t)** according to the model described above, see **eq (6)**. To reach that, we will:

1. Calculate the implementation of materialized views cost $C_{imp} = [c_{imp_1}, c_{imp_2}, \dots, c_{imp_n}]$. for each query Q_i , as mentioned in the definition of the metric PayOffRatio section above, this cost is equal to the sum of view creation time.
2. Therefore, calculate the value of ϵ according to **eq(8)**.
3. After that, regenerate for all queries Q_i the cost using the materialized views c_{a_i} in $C_A = [c_{a_1}, c_{a_2}, \dots, c_{a_n}]$. Therefore, calculate the Cost(t) according to the model presented by **eq(6)**.
4. Finally, generate the metric PayOffRatio using **eq(11)**

3.4. Generalization of lattice phase

The goal of this phase is to generate the corresponding lattice for the set of queries in the workload where the number of cuboids is proportional to the global number of attributes.

For this purpose, using the model described in the initial phases, we consider a fact table, $R_i = [r_{i1}, r_{i2}, r_{i3}, r_{i4}, M_{i1}]$, and the maximum number of cuboids that can be generated with the lattice is $nbr_cuboid = 2^n$, where n is the number of dimensions attributes in the fact table. In the current study, $n=4$ then $2^4 = 16$ cuboids in the lattice (cuboid $\Leftrightarrow$ Query).

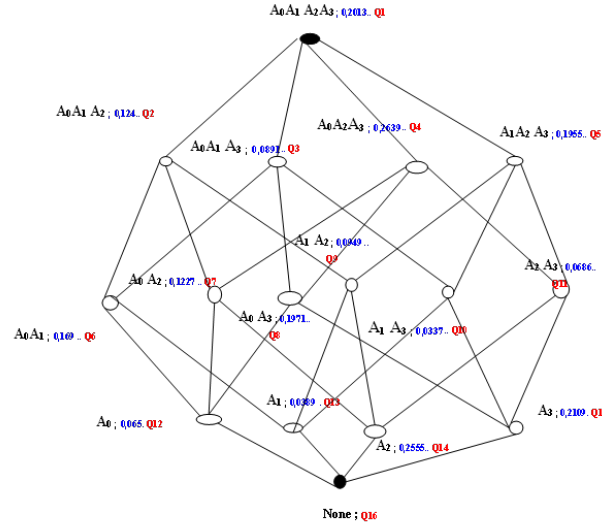


Figure 3. Data cube lattice Structure

The figure 3 shows an example of a lattice with a number of attributes equal to $n=4$. Moreover, we have generated a set of five metrics that were used on almost all previous approaches, such as:

- 1) - Frequency of execution of the query,
- 2) - Number Of Lines -**NOFL**- generated by the query,
- 3) - Time Of its Execution Before using Technic of materialize view- **TEBFT**-
- 4) - Time Of its Execution After using the Technic of materialized views -**TEAFT**-,
- 5) - Time of Implementation of the techniques (creation the materialized view in our use case) -**TIT**- in addition to our novel metrics
- 6) - PayoffRatio.

Furthermore, to clearly explain the approach adopted in our framework, we will demonstrate it using an example. The parameters were randomly generated according to the principle of level aggregation on the cuboid in the lattice, and the values of the basic variables used to calculate our gain ratio were arbitrarily generated. In our experimental part, these values will be extracted directly from the workload. Table 4 below presents an example of results obtained with the test parameter function.

In fact, The number of rows returned by a query with group by is less than that returned without group by which explains for us the arbitrarily generated data according to the margins mentioned in the table for lower level queries compared to these of higher level, for example: the number of rows of query Q2 with 3 attributes is less than these of query Q1 with 4 attributes, which we had done with the parameter P1. Similarly, the response time of the materialized view must be less than the response time of the query itself, which justifies the arbitrary use of intervals $10 \Rightarrow 110$ for TEAFT contrary

```

P0 = Frequency = randomly generated between 1 ==> 50
P1= Number of lines = randomly generated between 100000
==> 100000*number of attributs +10000
P2 = Time execution before using techniques =
randomly generated between 100 ==> 1000
P3 = Time execution after using techniques =
randomly generated between 10 ==> 110
P 4= Time required to implement the technique =
randomly generated between / 100 ==> 190 Time update (Ci) =
*: epsilon= P4 / P0;
PayOffRatio= GR = (P3 + epsilon) /P2);
P5 = PayOffRatio;

```

Table 3. Pseudo Code of the function to generate the test parameter

Query	DESIGNE	FREQ.	NOFL	TEBFT(S)	TIT(S)	TEAFT(S)	PayOff Ratio
Q1	[A _{tr} 0 A _{tr} 1 A _{tr} 2 A _{tr} 3]	32	1606636	491	93	187	0.2013
Q2	[A _{tr} 0 A _{tr} 1 A _{tr} 2]	6	1203960	919	89	150	0.1240
Q3	[A _{tr} 0 A _{tr} 1 A _{tr} 3]	39	1205463	519	42	166	0.0891
Q4	[A _{tr} 0 A _{tr} 2 A _{tr} 3]	49	1204758	142	35	121	0.2639
Q5	[A _{tr} 1 A _{tr} 2 A _{tr} 3]	37	1203591	150	25	160	0.1955
Q6	[A _{tr} 0 A _{tr} 1]	28	801156	566	91	130	0.1690
Q7	[A _{tr} 0 A _{tr} 2]	20	800988	748	83	175	0.1227
Q8	[A _{tr} 0 A _{tr} 3]	12	802054	216	34	103	0.1971
Q9	[A _{tr} 1 A _{tr} 2]	9	803494	926	76	107	0.0949
Q10	[A _{tr} 1 A _{tr} 3]	41	802770	1007	31	119	0.0337
Q11	[A _{tr} 2 A _{tr} 3]	42	802542	878	57	137	0.0686
Q12	[A _{tr} 0]	49	400550	623	37	171	0.0650
Q13	[A _{tr} 1]	20	400047	746	20	180	0.0389
Q14	[A _{tr} 2]	21	400409	216	47	172	0.2555
Q15	[A _{tr} 3]	50	400540	300	60	163	0.2109

Table 4. Example of the result of generating the test parameters function.

to **TEBFT(100 – 1000)**. The Pseudo-code of the initialization function is presented in Table 3 above. The first parameter P0 is the frequency of execution of the query, generated randomly between 1 and 50. The second parameter, P1, is the number of lines of each view to be materialized, followed by the time of execution before and after using the techniques P2 and P3, respectively. We also generate the workload materialized view to obtain the amount of time needed to implement the techniques P4. Finally, we calculate the epsilon variable and the last parameter P5, which presents our PayOffRatio Metrics.

Table 4 below presents an example of generating initial parameter tests, with metrics for each query and its associated cuboids, for a lattice with 4-dimensional attributes in the fact table. The advantage of our approach lies in its ability to integrate other metrics without limitation, depending on the characteristics of the workload being analyzed. For example, it is possible to add to the implementation time of a materialized view, in addition to its creation time, the time of its updates and the frequency of these updates. In addition, the framework applies the model defined in the initialization phases described above to a workload formed in the first use by 16 queries $n=2^4$ referred by Q₁, Q₂...Q₁₆.

Note that the Time of implementation of the materialization view could be presented using the time of creation and the update time. As mentioned in the above section, in our work we limited this time to the time of creation of the materialized view.

The creation of the lattice is inspired from the principle used in reference [14] regarding the generation of a list of predecessors- *ancestors* - and successors - *descendants* - for each query.

3.5. Evaluation of algorithms phase

In this step, after evaluation of the workload, and calculating the PayOffRatio metrics to evaluate each algorithm. The model shown in Figure 4 has been used to instaure and save the results of each execution

and evaluation of the workload. To validate our approach, the study examined two versions of the greedy algorithm. In the first case, we will study the behavior of the algorithm to choose the set of cuboids to be materialized using various metrics employed in different approaches without dependency between cuboids at different levels. In the second case, we will apply the dependency constraints between the cuboid in level n and level $n-1$ of the lattice structure.

The principle used to define the dependency is described as follows:

Let $Q_{anc} = \{q_{l1}, q_{li}, \dots, q_{ly}\}$ be the set of y queries that presents all ancestor candidate queries in the level $n-1$, and $Q_{danc} = \{q_{lk}, q_{lm}, \dots, q_{lp}\}$ where $|Q_{danc}| \leq |Q_{anc}|$ is the subset with $p, p \leq y$ queries from Q_{anc} that can be used to calculate the query q_i in level n . Applying the dependency means that the selection of the ancestor subset of query q_i are based on Q_{danc} with maximum benefit. And without dependency means that the selection of the subset ancestor queries based on Q_{anc} with maximum benefit.

The table 5 below displays the result of evaluating the workload with two versions of greedy algorithms using the parameters of example 2 above. The selected subset of the materialized view is different for all cases according to the metrics used. For example, line 2 in table 4 represents the result of the selection using the number of lines as a metric, providing the subset formed by $([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}3], [A_{tr}0])$. Additionally, the top view selected in the first iteration of the program is $[A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3]$ as high level in the lattice $n=4$ followed by the cuboid $[A_{tr}0, A_{tr}2, A_{tr}3]$ in the level $n-1=3$ as the second choose from the set of cuboids like $([A_{tr}0, A_{tr}1, A_{tr}2], [A_{tr}0, A_{tr}1, A_{tr}3], [A_{tr}0, A_{tr}2, A_{tr}3], [A_{tr}1, A_{tr}2, A_{tr}3])$.

For the third selection, the view $[A_{tr}0, A_{tr}3]$ is selected from a set of dependency to $[A_{tr}0, A_{tr}2, A_{tr}3]$ composed by a set $([A_{tr}0, A_{tr}2], [A_{tr}0, A_{tr}3], [A_{tr}2, A_{tr}3])$, in the last step, the cuboid $[A_{tr}0]$ has selected from the set of possible views $([A_{tr}0], [A_{tr}3])$.

However, the selection without dependency in a similar step process, which chose the cuboid $[A_{tr}1]$, it does not fall within the dependency list of the view $[A_{tr}0, A_{tr}3]$. The table also displays the summary of results for our experimental results for all metrics, with the benefits of each execution. On the set of materialized views selected using the time after using the techniques, has a poor cost due to the manner in which the process is deployed to create a set of materialized views that maintains the subset with the lowest cost. Contrarily, the goal of all optimization processes is to select first the query with the longest response time in order to reduce it. These results were obtained by applying randomly generated parameters to simplify the understanding of our approach; in the following experimental section, we will evaluate the framework using parameters from the full set of workload queries.

ALGO*	SUBSET SELECTED CUBOID	BENEFITS	METRICS
1	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}3], [A_{tr}1])$	4008042	NOFL
2	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}3], [A_{tr}0])$	4008123	NOFL
1	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}1, A_{tr}3], [A_{tr}1, A_{tr}2], [A_{tr}3])$	1032	TEBFT
2	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}1, A_{tr}3], [A_{tr}0, A_{tr}3], [A_{tr}3])$	1289	TEBFT
1	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}1, A_{tr}2], [A_{tr}0, A_{tr}1], [A_{tr}1])$	208	TIT
2	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}1, A_{tr}2], [A_{tr}0, A_{tr}1], [A_{tr}1])$	208	TIT
1	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}3], [A_{tr}1])$	431	TEAFT
2	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}2, A_{tr}3], [A_{tr}3])$	466	TEAFT
1	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}0, A_{tr}1], [A_{tr}1])$	0.4114	PAYOFFRATIO
2	$([A_{tr}0, A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}1, A_{tr}2, A_{tr}3], [A_{tr}2, A_{tr}3], [A_{tr}2])$	0.7333	PAYOFFRATIO

1 : Ordinary Greedy Algorithm Without dependency between cuboids

2 : Greedy Selection Algorithm With dependency between cuboids

Table 5. The result of the evaluation phase of the algorithm based on the parameters of example 2 above.

4. Experimental result: Views and interpretations

The initial data warehouse of our experimental section is presented in the table 6 below:

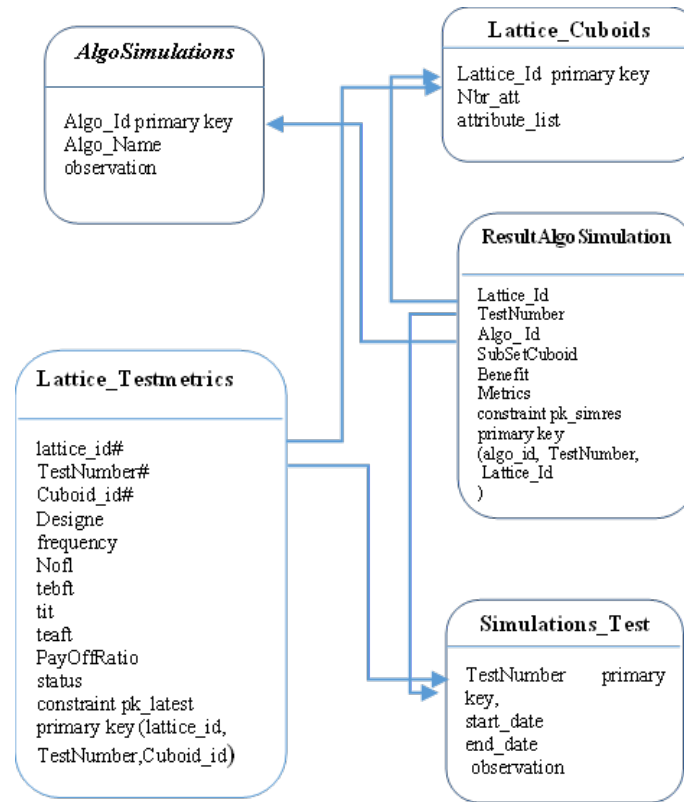


Figure 4. : Proposed Simulation Architecture

Relation table	Size (Number of Tuples)
DIM_ CUSTOMER	1000 000
DIM_ PRODUCT	10 000
DIM_ GEOGRAPHY	60
DIM_ DATE	864 000
FACT_ SALES	4 443 227

Table 6. Initial dataset size

Hardware specifications: Processor: 11 Gen Intel(R) Core(TM) i5-115G7 CPU @ 2.60GHz 1.5GHz, Memory:16G RAM,

Operating System: Windows 11 Professional 64-bit

Software specifications: Oracle 19C Release 19.0.0.0.0, Version 19. 3.0.0.0, Oracle Sql Developer Version 23.1.1.345.

Ide developement : Eclipse IDE for Java Developers VERSION : 2022-09 (4.25.0)

A star schema based on four dimensions and a fact table were used with a workload consisting of 16 queries. These queries present all the cuboids of our lattice, with 4 dimension attributes. The table 7 presents the pseudo code of our framework step by step.....

Table 8 below lists the frequencies associated with the queries in our workload generated according to the model described in the preceding sections.

The response time to analytical queries is rapidly increasing due to the constant growth of data in decision-making systems. The use of materialized view techniques to optimize the analytical process, as illustrated in Figure 5 below, presents a considerable advantage in terms of response time compared to

Input: Number of dimension attributes in the fact table
1. Calculate the number of cuboids in the lattice.
2. Generate the list of cuboids in the lattice.
3. Analyze the queries in the workload (Calculate the response time of each query and the number of lines returned)
4. Determine the time required to create the materialized view associated with the query
5. Associate execution frequency to each query, to simulate various scenarios, the random function generation has been used.
6. Evaluate the workload using the materialized views and calculate the PayOffRatio for every request.
7. Fix the metrics associated with each cuboid. $\Rightarrow$ [A: Frequency, B: Number of lines, C: Execution time before using the view materialization technique, D: Technique implementation time, E: Execution time after applying the technique, and F: PayOffRatio].
8. Generate the ancestors of each cuboid
9. Generate the descendants of each cuboid
10. Generate the lattice tagged by the metrics
11. Select the optimal subset using two different versions of the greedy algorithm for each metric.
12. Generate the benefit of the MVS selection.
Output: The optimal Subset of selected cuboid to be materialized and its associated benefits

Table 7. Framework Pseudo code (in the lattice: Cuboid $\Leftrightarrow$ Query in the workload)

CUBOID	QUERY	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
0	Q1	47	22	44	27	45	53	87	69	95	75	75
1	Q2	23	36	43	25	39	66	91	72	88	73	79
2	Q3	44	44	30	27	20	74	93	57	82	78	78
3	Q4	42	44	24	20	46	96	98	71	64	76	75
4	Q5	23	42	45	41	47	60	51	74	84	70	71
5	Q6	40	35	44	48	28	82	80	96	64	74	75
6	Q7	22	33	49	41	42	80	72	85	56	72	73
7	Q8	35	24	33	40	29	59	63	65	73	70	75
8	Q9	31	41	38	48	31	65	51	98	67	78	73
9	Q10	43	26	43	25	35	98	92	67	69	75	71
10	Q11	43	36	25	38	24	51	95	64	96	75	72
11	Q12	20	22	23	26	35	87	94	95	99	72	74
12	Q13	26	27	20	25	23	97	73	90	81	74	78
13	Q14	48	30	27	33	20	80	57	88	74	70	78
14	Q15	31	35	41	32	24	71	72	66	50	73	72
15	Q16	25	33	35	34	42	75	71	86	84	76	77

Table 8. Frequency Subset Generated to evaluate the workload queries

the cost of processing queries without these techniques and the time required to implement them, as illustrated in Figure 6.

The Figure 5 displays the result of our PayOffRatio Metric for a global cost after evaluation of all queries in the workload without and using the materialized view technique. We note that for both principles used to select the set of cuboids to be materialized, the cost is very lower compared to evaluating the query without techniques. Indeed, the selection process using the dependency (MVS.WD) performs better than other processes, respectively with a cost of **34.10 S (MVS.WD)**, **35.91 S (MVS.WTTD)**, in contrast to a single process using the initial query with **479.96 S**, more than teen times (Without Materialized views).

The Figure 6, depict the response time of all queries used in our workload before and after materialization of views and the time of implementation of each view. The figure shows that the response time for processing queries after the view is materialized is negligible compared to the cost before using this technique, even though the cost of its implementation is considerable for almost all queries, for example: Queries Q_1 , Q_2 , Q_3 and Q_5 .

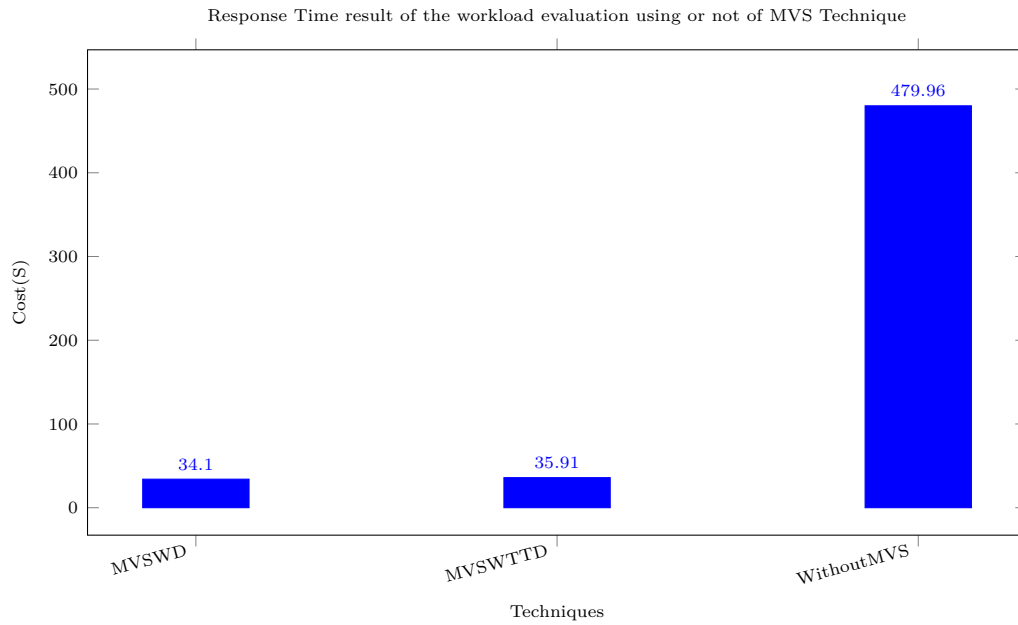


Figure 5. : Response time result of the workload evaluation using or not of the materialized view technique.

DESIGNE	TEBFT	TIT	TEAFT	PAYOFFRATIO
Q1	27.61	5.58	8.42	0.501
Q2	26.92	4.17	3.87	0.657
Q3	19.85	4.99	3.4	0.354
Q4	29.93	4.33	0.01	0.005
Q5	22.59	4.69	2.33	0.135
Q6	37.38	3.84	1.66	0.093
Q7	34.33	3.38	0.01	0.003
Q8	28.74	3.87	0.01	0.005
Q9	24.78	3.62	1.06	0.306
Q10	35.01	3.85	1.1	0.053
Q11	28.95	3.96	0.01	0.005
Q12	29.52	3.01	0.01	0.022
Q13	29.79	3.18	0.44	0.035
Q14	28.88	3.24	0.01	0.004
Q15	37.8	3.66	0.01	0.005
Q16	37.88	≈ 0	0.01	≈ 0

Table 9. The average of the initial costs using the frequency samples from the previous table generated to evaluate the workload queries

The table 10 describes the procedure to evaluate the workload after the selection of the Subset of views to be materialized by the algorithm.

The table 11 below gives the global cost per Second of algorithm MVS according to various metrics compared to our PayOffRatio using two versions of the greedy algorithm: one taking into account, and the other not taking into account, the dependency between the query and its list of ancestor queries, denoted above by Q_{danc} and Q_{anc} respectively. As you can see, the two versions of the greedy algorithm produce distinct subsets of the materialized views, and even within the same version, these subsets vary based on the various metrics.

Furthermore, the response time varies according to the metrics, but using the PayOffRatio as the primary

```

Let S be a Subset of cuboids selected by MVS algorithm
For each Query  $Q_i$  in the workload do
  if  $Q_i$  belongs S, ( $Q_i \in S$ ) then
    use MVS( $Q_i$ )
  Else
    if  $\exists Q_j, j \neq i$  where  $Q_j \in \text{Ancestors}(Q_i)$  &  $Q_j \in S$  then
      Calculate Result of  $Q_i$  using MVS( $Q_j$ )
    Else
      Run the query  $Q_i$  using the original script.
    End if
  End if
End for

```

Table 10. Generation Result Of selected MVS Subset

Metric	Greedy Selection Algorithm (MVSWD)	Ordinary Greedy Algorithm (MVSWTTD)
PayOffRatio	11.80	32.20
Time execution After Techniques	30.23	34.28
Time execution Before Techniques	35.82	20.43
Time Implementation Techniques	37.67	35.85
Number of lines	38.09	43.88

Table 11. Global Cost Per Second Of Algorithm MVS according To Various Metrics Compared to Our PayOffRatio.

metric yields better results. Furthermore, the response time varies according to the metrics, but using the PayOffRatio as the primary metric yields better results. The results have been collected using the various sequences of frequency of execution of queries, which only generated randomly and applied to the associated queries. The overall cost generated using our PayOffRatio metric is the best of all the metrics used for the dependency-based greedy algorithm, while for the Ordinary greedy algorithm, except the initial execution time, PayOffRatio produces good results also.

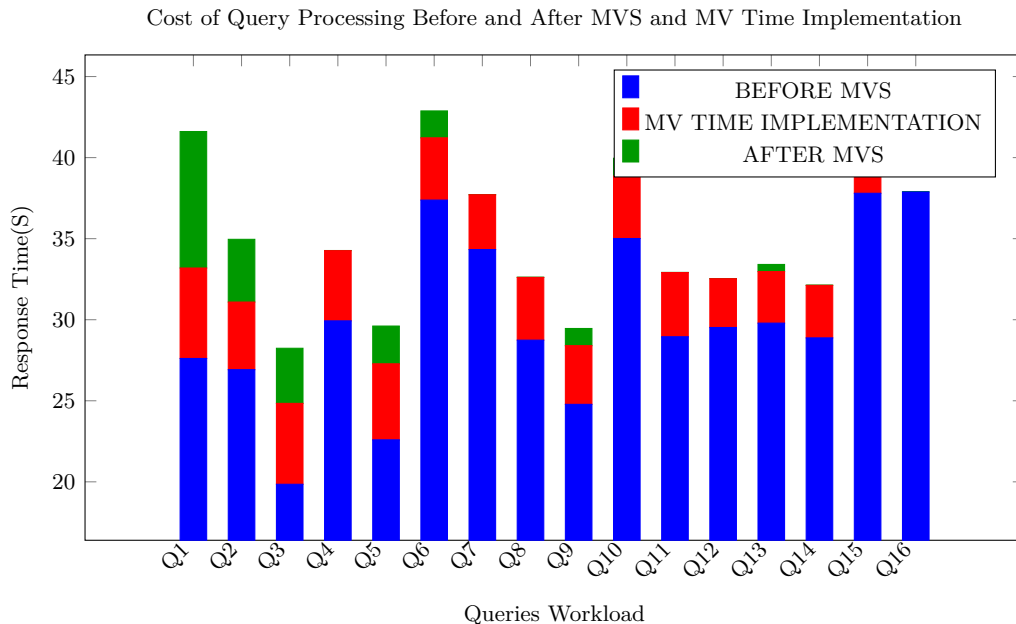


Figure 6. : Cost of Query Processing Before and After MVS and MV Time Implementation.

The figure 7 illustrates the performance evaluation measured by the global cost with and without dependency between cuboids (WD= With Dependency; WTTD= Without Dependency). Regardless of the metric used, the graphic shows how the selection cost differs when selecting materialized views based on dependencies versus selecting the opposite situation. On the other hand, the picture also shows that our method is the most cost-effective in all other cases, with a lower cost of **11.80 s** and **32.20s** using dependency or not, respectively, contrary to other metrics. For most of the approaches developed in the literature that use the model based on the number of lines, our metrics perform better, reducing costs by nearly **-36%** times **32.2s** instead of **43.88s**, when the dependency is disabled, and reducing the cost by nearly **-239%** with dependency from **38.08s** to **11.80s**. Moreover, our study extended to compare our new metric to other approaches based on the response time, or combined.

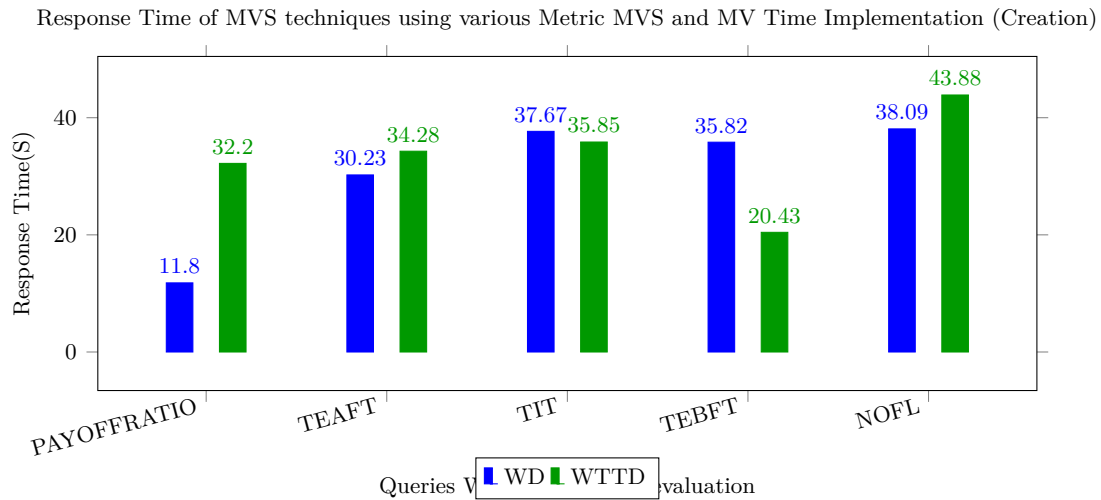


Figure 7.: Response Time of MVS techniques using various Metrics and MV Time Implementation (Creation)

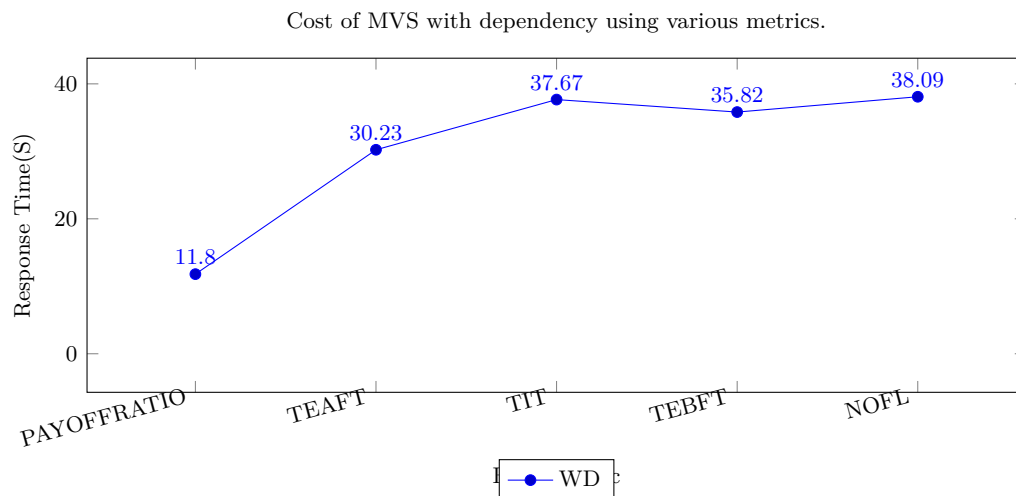


Figure 8.: Comparison of the total cost of selecting materialized views with dependency using various metrics.

The Table 12 presents the results of applying the PayOffRatio as the basic metric using two versions of the greedy algorithm in order to be extended to other heuristic and non-heuristic algorithm in our future work. The global processing Cost is evaluated using the frequency F10 the table 8 above such as $\{(Q_1, 75); (Q_2, 73); (Q_3, 78); (Q_4, 76); (Q_5, 70); (Q_6, 74); (Q_7, 72); (Q_8, 70); (Q_9, 78); (Q_{10}, 75); (Q_{11}, 75); (Q_{12}, 72); (Q_{13}, 74); (Q_{14}, 70); (Q_{15}, 73); (Q_{16}, 76)\}$;

The selected subset using our *PayoffMetric* metrics is not similar to any subset in the rest of the tuples that use another metric such as **NOFL**, **TEBFT**, **TIT**, or **TEAFT**. The result obtained confirms the conclusions drawn during our previous phase of evaluation of the algorithm (section 3.5 above), which represents the theoretical part of our approach.

For example, line 2 in table 12 represents the result of the selection using the number of lines as a metric, providing the subset formed by $(Q_1, Q_4, Q_{11}, Q_{15})$. Additionally, the top view selected in the first iteration of the program is Q_1 as high-level in the lattice $n = 4$ followed by the cuboid Q_4 in the level $n-1=3$ as the second choice from the set of Candidates cuboid like (Q_2, Q_3, Q_4, Q_5) .

For the third selection, the view Q_{11} is selected from a set of dependencies to Q_4 composed by a set (Q_7, Q_8, Q_{11}) , in the last step, the cuboid Q_{15} has selected from the set of possible views (Q_{14}, Q_{15}) . However, the selection without dependency at the similar step process, which chooses the same set of selected cuboid Q_{15} , but with the remarkable Global Cost of 33.5S in opposite to 19.9S with dependency. For our PayOffRatio, it is clear that the result obtained for the two versions is extraordinary, namely 1.18 s and 4.57 s, respectively, with dependency and without dependency.

ALGO*	SUBSET SELECTED CUBOID	BENEFITS	METRICS	Global Cost(S)
1	(Q1,Q4,Q11,Q15)	113 870	NOFL	33.05
2	(Q1,Q4,Q11,Q15)	113 870	NOFL	19.90
1	(Q1,Q2,Q7,Q13)	57.80	TEBFT	49.76
2	(Q1,Q2,Q7,Q14)	69.59	TEBFT	12.71
1	(Q1,Q2,Q7,Q13)	11.94	TIT	20.20
2	(Q1,Q2,Q7,Q12)	11.95	TIT	19.63
1	(Q1,Q4,Q7,Q15)	8.33	TEAFT	54.32
2	(Q1,Q4,Q11,Q15)	8.35	TEAFT	13.92
1	(Q1,Q4,Q11,Q12)	0.18	PAYOFFRATIO	1.17
2	(Q1,Q4,Q11,Q15)	0.18	PAYOFFRATIO	4.57

1 : Ordinary Greedy Algorithm Without dependency between cuboids

2 : Greedy Selection Algorithm With dependency between cuboids

Table 12. The Global Cost Per Second of algorithms evaluation of the Workload using various metrics.

Moreover, the Figure 8 shows the average global cost of evaluating the workload by greedy algorithm selection using the dependency between the cuboids of two consecutive levels using all frequency arrays presented in the table 8 described previously, and the Figure 9 shows the result using a greedy algorithm without dependency constraints. The curves in the two figures shows that our *PayOffRatio* performed better, and we can see the evidently excellent performance presented using our proposed metric, *PayoffRatio*. In the same way, figure 9 shows that, even if dependencies are not used in the selection process, the *PayoffRatio* method performs better than other methods with the cost of **32s** instead of **34.28s**, **35.85s**, **20.43s**, and **43.83s** for **TEAFT**, **TIT**, **TEBFT**, and **NOFL**, respectively.

5. Conclusion

In the world of business intelligence, researchers are focusing on the optimization algorithms available on the market to find the best combinations for selecting the best subsets of views to materialize to improve interviewer response times for all workloads. The approaches developed often use techniques based on the number of rows to simulate the size of each view, the cost of executing queries linked to these views, and

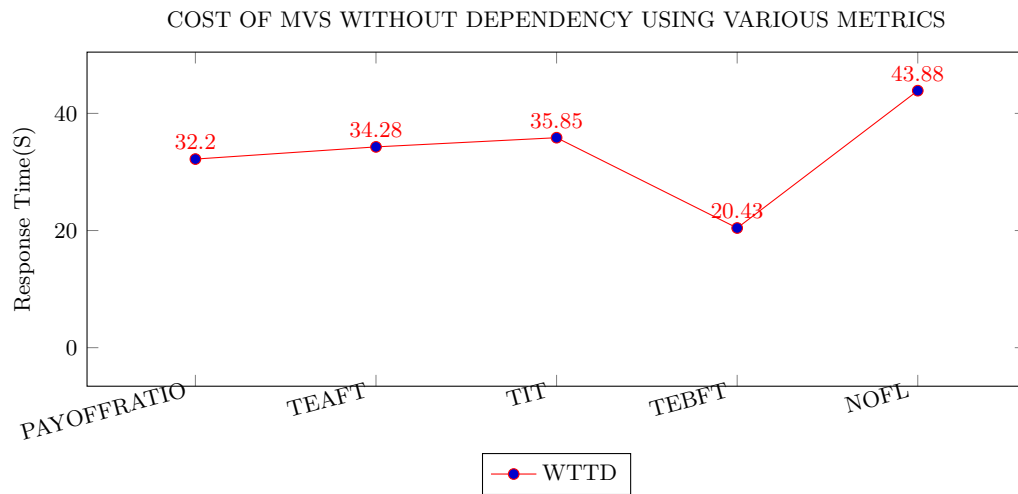


Figure 9. **COST OF MVS WITHOUT DEPENDENCY USING VARIOUS METRICS**

the cost of creating and updating these materialized views. In this paper, we have defined a new metric called “*PayOffRatio*”, which combines a set of other metrics such as execution time before and after using the materialized view technique, and the implementation cost of the technique, in order to select subsets that improve performance by reducing the overall cost of selection. We first tested this metric using the Greedy algorithm without dependencies and with dependencies. Moreover, we have shown the best performance guaranteed by our approach. We will then use this metric with other optimization algorithms, such as genetic algorithms, game theory, and others. Finally, and as a perspective, we are going to use this metric for other techniques, such as indexing techniques and hybrid techniques.

REFERENCES

1. Prakash, Jay and Kumar, TV Vijay, *A multi-objective approach for materialized view selection*, IGI Global International Journal of Operations Research and Information Systems (IJORIS) , vol. 10, no. 2, pp. 1–19, 2019.
2. Prakash, Jay and Kumar, TV Vijay, *Multi-objective materialized view selection using NSGA-II*, Springer Journal of System Assurance Engineering and Management, vol. 11, pp. 972–984, 2020.
3. Manavi, Mahdi, *Multi-Objective Genetic Algorithm for Materialized View Optimization in Data Warehouses*, arXiv preprint arXiv:2403.19906, 2024.
4. Sachdeva Kavita, Gosain Anjana, *Materialized view selection applying differential evolution algorithm combined with ensemble constraint handling techniques*, Springer Multimedia Tools and Applications, vol. 80, no. 21, pp. 31619–31645, 2021.
5. Mohamed Ridani, Mohamed Amnai, *Optimization challenge in decision supporting systems: An overview*, IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), pp. 1–6, 2022.
6. Mohamed Ridani, Mohamed Amnai, *Materialized Views Selection Problem in Decision Supporting Systems: Issues and Challenges*, Scientific Research Publishing Journal of Computer and Communications, vol. 10, no. 9, pp. 96–112, 2022.
7. Gosain Anjana and others, *Materialized cube selection using particle swarm optimization algorithm*, Elsevier Procedia Computer Science, vol. 79, pp. 2–7, 2016.
8. Kumar Amit , Kumar TV Vijay, *Materialized view selection using discrete genetic operators based particle swarm optimization*, IEEE International Conference on Inventive Systems and Control (ICISC), pp. 1–5, 2017.
9. Azgomi Hossein ,Sohrabi Mohammad Karim, *A game theory based framework for materialized view selection in data warehouses*, Elsevier Engineering Applications of Artificial Intelligence, vol. 71, pp. 125–137, 2018.
10. Azgomi Hossein ,Sohrabi Mohammad Karim, *A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments*, Springer Applied Intelligence, vol. 49, pp. 3965–3989, 2019.
11. Mohseni Mohsen , Sohrabi Mohammad Karim, *MVPP-based materialized view selection in data warehouses using simulated annealing*, Scientific International Journal of Cooperative Information Systems, vol. 29, no. 03, pp. 2050001, 2020.
12. Berkani Nabila, Bellatreche Ladj, Ordóñez Carlos, *ETL-aware materialized view selection in semantic data stream warehouses*, IEEE 12th international conference on research challenges in information science (RCIS), pp. 1–11, 2018.

13. Lakshmanan Laks VS , Pei Jian , Zhao an, *Efficacious data cube exploration by semantic summarization and compression*, Elsevier Proceedings 2003 VLDB Conference, pp. 1125–1128, 2003.
14. Harinarayan Venky, Rajaraman Anand , Ullman Jeffrey D, *Implementing data cubes efficiently*, ACM New York, NY, USA, Acm Sigmod Record, vol. 25, no. 2, pp. 205–216, 1996.
15. Mohamed Badri, *Mise à jour incrémentale des agrégats: Cas des indicateurs ROLAP dans les entrepôts de données*, CRIP5, IUT de Paris Descartes, 2007.
16. Lee Ki Yong , Kim Myoung Ho, *Efficient incremental maintenance of data cubes*, Association for Computing Machinery (ACM), VLDB2006, 32nd Int'l Conf. on Very Large Data Bases, pp. 823–833, 2006.
17. Lin Wen-Yang , Kuo I-Chung, *A genetic selection algorithm for OLAP data cubes*, Springer, Knowledge and information systems, vol. 6, pp. 83–102, 2004.
18. Vijay Kumar TV, Kumar Santosh, *Materialized view selection using genetic algorithm*, Springer, Contemporary Computing: 5th International Conference, IC3 2012, Noida, India, August 6-8, 2012. Proceedings 5, pp. 225–237, 2012.
19. Necir Hamid, Drias Habiba, *A Distributed Clustering with Intelligent Multi Agents System for Materialized Views Selection*, Springer, Knowledge Engineering and Management: Proceedings of the Sixth International Conference on Intelligent Systems and Knowledge Engineering, Shanghai, China, Dec 2011 (ISKE2011) , pp. 417–426, 2011.
20. Azgomi Hossein, Sohrabi Mohammad Karim *MR-MVPP: A map-reduce-based approach for creating MVPP in data warehouses for big data applications*, Elsevier Information Sciences, vol. 570, pp. 200–224, 2021.
21. Roy Santanu, Ghosh Ranak, Sen Soumya, *Materialized view construction based on clustering technique*, Springer Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings 14, pp. 254–265, 2014.