Scientific Research Publishing

Journal of Computer and Communications, 2022, 10, 96-112
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Materialized Views Selection Problem in Decision Supporting Systems: Issues and Challenges

Mohamed Ridani, Mohamed Amnai

Laboratory of Research in Informatics (LaRI), Faculty of Science, Ibn Tofail University, Kenitra, Morocco
Email: mohamed.ridani@uit.ac.ma, mohamed.amnai@uit.ac.ma

How to cite this paper: Ridani, M. and Amnai, M. (2022) Materialized Views Selection Problem in Decision Supporting Systems: Issues and Challenges. *Journal of Computer and Communications*, **10**, 96-112.
https://doi.org/10.4236/jcc.2022.109007

Received: July 23, 2022
Accepted: September 20, 2022
Published: September 23, 2022

Abstract

The data warehouse is the most widely used database structure in many decision support systems around the world. This is the reason why a lot of research has been conducted in the literature over the last two decades on their design, refreshment and optimization. The manipulation of hypercubes (cubes) of data is a frequently used operation in the design of multidimensional data warehouses, due to their better adaptation to OLAP (On-Line Analytical Processing). However, the updating of these hypercubes is a very complicated process due mainly to the mass and complexity of the data presented. The purpose of this paper is to present the state of the art of works based on multidimensional modeling using the hypercube as a unit of presentation of data stores. It starts with the base of this process which is the choice of the views (cubes) forming our data warehouse base. The objective of this work is to describe the state of the art of research works dealing with the selection of materialized views in decision support systems.

Keywords

Data Hypercube, OLAP, Data Warehouse, Materialized Views Selection

1. Introduction

Presenting data in coherent and integral situation reflecting its current state remains a challenge of traditional information systems called transactional systems using OLTP process (On-Line Transactional Processing). However, these systems are still insufficient for the Decision Support Systems (DSS). As long as they do not keep historical information on the data and the evolution of data is not available and they need to build specific systems to simplify the process of online

analysis of OLAP (On-Line Analytical Processing) data. These systems, called decision support systems, are information systems dedicated to decision-supporting applications [1] [2] [3]. **Figure 1** presents an example of Decision Support Systems. Such systems are centralized around what is called a "Datawarehouse" data warehouse defined according to two basic approaches: the first designates the data warehouse as a collection of integrated data, subject-oriented, non-volatile, historical, summarized and available for interrogation [3]; the second defines the warehouse as a set of Datamart data stores, each store of which represents an extract from the warehouse dealing with a given theme [1] [4].

There are several research works dealing with the design and modeling of data warehouses classified according to the type of database used for storage of data warehouse: works dealing with relational modeling [5] [6] [7] and others based on multidimensional modeling [8] [9]. Indeed, the nature, variety and volume of data in data warehouses are constantly increasing on the one hand and the need for analysts and queriers to optimize response time, storage cost and update time is becoming more and more of a challenge for developers of business intelligence systems. Thus the appearance of spatial warehouses [10] [11] [12] [13], Bigdata [14]-[19] and Datawarehouse in the cloud environment [20] [21] on the other hand.

The purpose of this paper is to present the state of the art of works based on multidimensional modeling using the hyper-cube as a unit of presentation of data stores. The hyper-cube is defined as a presentation of the data to be analyzed in a multidimensional form whose values represent the measures to be analyzed and the dimensions represent the axes of analysis [22]. Indeed, the volume of data stored in warehouses is becoming more and more important, generating hypercubes that are complex to analyze and whose updates are also becoming very difficult. There are several approaches to updating the data cube in a data warehouse. Among these approaches are those that allow the total construction of data cubes [23] and those that deal with incremental updating, which

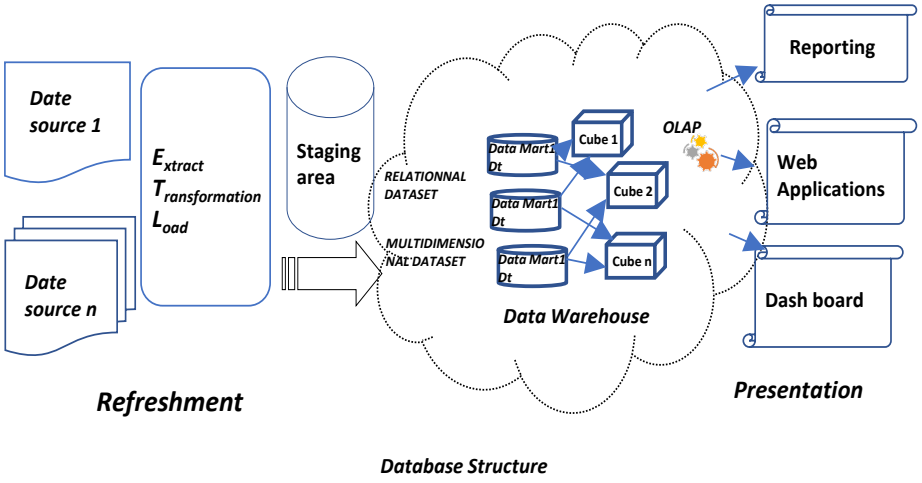


Figure 1. Architecture of decision system support (DSS).

consists of updating only the data that undergoes changes in the data sources [5] [6] [22]. Before starting the studies on the update of hypercubes, we will first deal with different approaches to the optimization algorithms for the selection of materialized views and their construction.

The rest of this paper is organized as follows: in the first section we define the global context of materialized views selection problem and give a brief overview of some approaches dealing with materialized views, the second section describes the different operators of hypercube manipulation, and the third section presents a retrospective of existing update methods of hypercube. Finally, in the last section, we give some perspectives and a conclusion.

2. A Global Context of the Materialized View Selection

2.1. Materialized View Selection

In the decision-making systems, the size of data become more and more constantly increasing therefore the response time of analytical queries become also very high as illustrated in Figure 3. To solve this situation, the obvious idea is to analyze all analytical queries used and stock its result expressly in data warehouse but this solution is very expansive in term of storage and the time of their computation and update. To deal with the situation the reach area has been proposed to select a set of queries that can response to others, hence the birth of materialized views selection. In fact, for the rest of the document we consider the star schema of Figure 2 like base structure of our Datawarehouse and taking for example the following query:

Example 1:

```
SELECT COUNTRY, DES_PROD, SITUATION, YEAR_, SUM (Sales) AS Total_Sales
FROM Fact_Sales FS, DIM_GEOGRAPHYG, DIM_DATE D, DIM_PRODUCT P,
DIM_CUSTOMERS C
WHERE FS.ID_CUST=C.ID_CUST AND FS.ID_PROD=P.ID_PROD
AND FS.ID_GEO=G.ID_GEO AND FS.ID_DATE=D.ID_DATE
GROUP BY COUNTRY, DES_PROD, SITUATION, YEAR_;
```

The hardware specifications of the machine and software description used in the rest of our experimental study are listed below.

Hardware specifications: Processor: Intel(R) Core(TM) i5-2520M CPU @ 2.50 GHz (4 CPUs), ~2.5 GHz, Memory: 6144 MB RAM, Operating System: Windows 10 Professional 64-bit.

Software specifications: Oracle 11G Release 11.2.0.4.0, Oracle Sql Developer Version 18.2.0.183.

Figure 3: Illustrates the performance evaluation measured by the response time increasing the percent size of the dataset in the fact table "***Fact_sales***" beginning with 200k tuples until 400k tuples with the step of 10% until 100%.

According to the experimental results based on Datawarehouse with the initial size described in Table 1, Figure 3 shows that response time grows when the

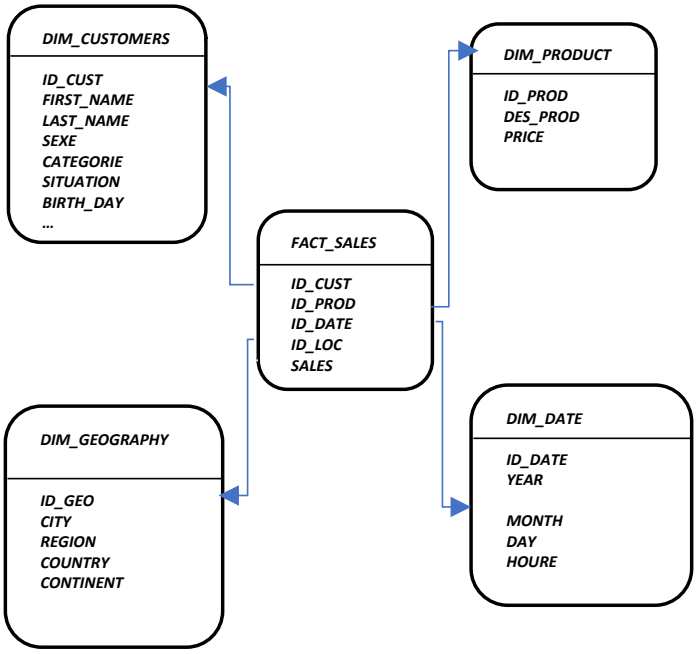


Figure 2. Star Schema base structure of our datawarehouse.

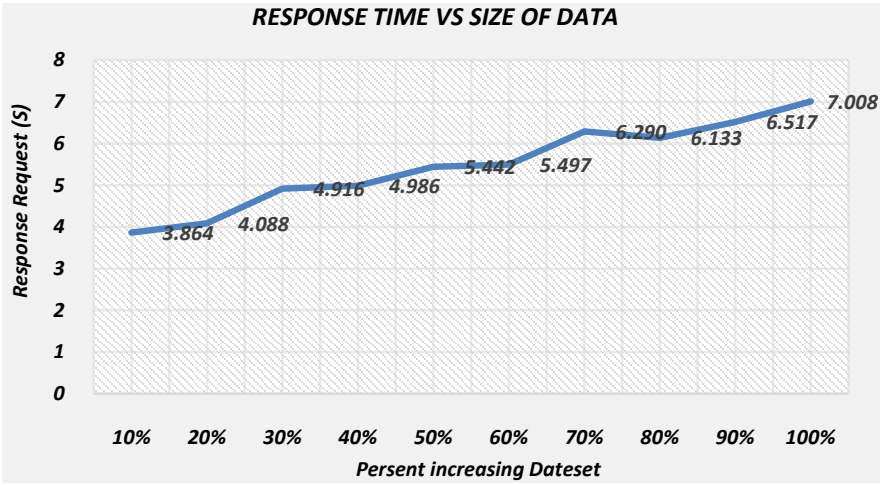


Figure 3. Query response time increasing size of data as number of tuples.

Table 1. Size of our datawarehouse.

Relation Table	*Size (Number of tuples)*
DIM_CUSTOMER	100,000
DIM_PRODUCT	1000
DIM_GEOGRAPHY	60
DIM_DATE	86,400
FACT_SALES	**200,000 ➔ 400,000**

percent of dataset increase also as described above. Hence it is necessary to develop approaches to cope with this degradation of performance with the evolu-

tion of data warehouses in terms of volume and variety of data with the appearance of Spatial and Big Data Warehouse. In fact, Figure 4 show the result of performance experiment among the use of materialized views technics with two view maintenance methods like complete computation and refresh method compared with the response time of the request without materialized views process.

The results in Figure 4 show that the response time for materialized views is negligible compared to the response time for the same query without using materialized views despite the increase in data size. The results are extracted without taking into account the update time of the materialized views. However, the deployment of the materialized view technique requires additional storage space and update time as shown in Figure 5.

Hence the appearance of optimization algorithms based on materialized view techniques with the major challenge of minimizing response time under storage space constraints. The following section gives a brief study of various algorithms dealing with the materialized view selection problem.

2.2. Overview Materialized View Selection Approaches

In fact, a great deal of the research works dealing with materialized view and

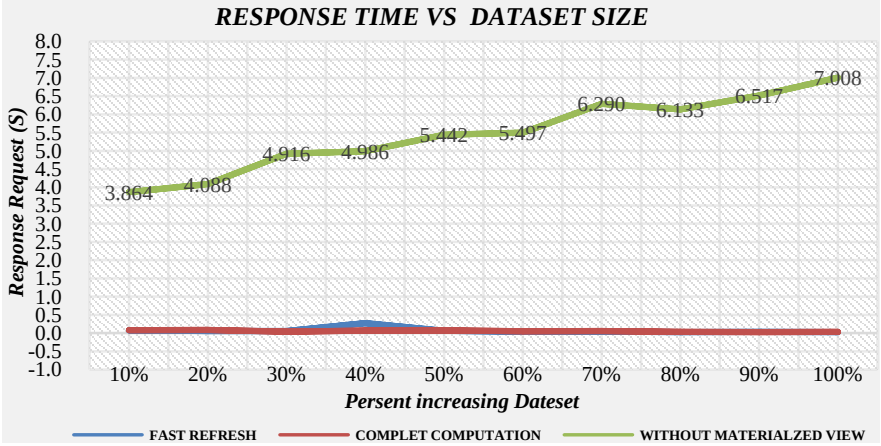


Figure 4. Query response time increasing dataset size using materialized view process.

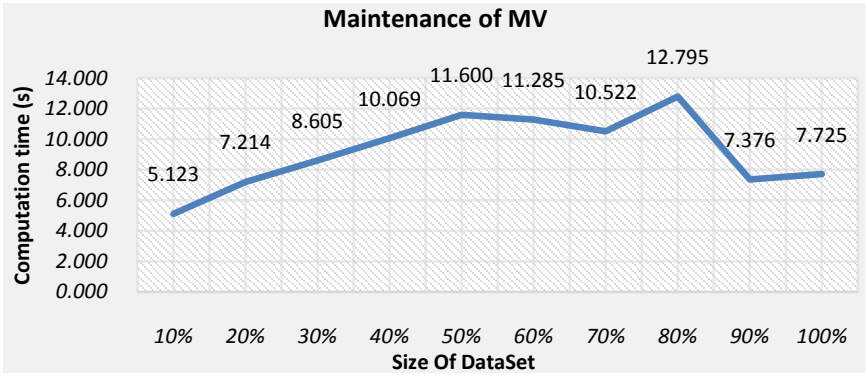


Figure 5. Materialized view with complete computation time increasing dataset size.

classed as NP-complete Problem and several algorithms are been developed in the literature which considers all possible views as an efficient representation structure to select a proper set of views for materialization [19]. The proposed structures to represent the views in the area of data warehouse are: (A)—Data cube lattice, (B)—AND-OR DAG (Directed Acyclic Graph), (C)—The MVPP (Multiple Views Processing Plan). For simplicity raisons to explain this variety of structure we used the request of example 2. Moreover, under this example the corresponding structures should be presented as shown in the following **Figure 6**.

For simplicity reasons, other information in the figure has been omitted for each node such as: query frequency F_v (frequency of the queries on view v), space S_v, update-frequency G_v (frequency of updates on v), and reading cost R_v (cost incurred in reading v). The interested researchers could refer to [24] [25] [26] [27] [28] for further details.

Example 2:

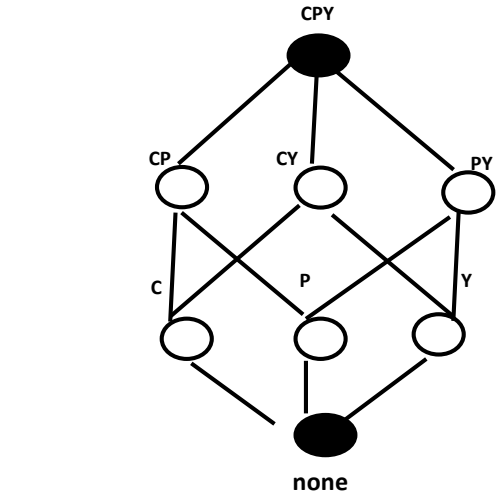
```
Q1: SELECT COUNTRY as C, DES_PROD as P, YEAR_ as Y, SUM (Sales) AS Total_Sales
FROM Fact_Sales FS, DIM_GEOGRAPHY G, DIM_DATE D, DIM_PRODUCT P
WHERE country='COUNTRY1' and YEAR_='2022' and SUBSTR(DES_PROD,9)
<='100' AND FS.ID_PROD= P.ID_PROD
AND FS.ID_GEO=G.ID_GEO AND FS.ID_DATE=D.ID_DATE
GROUP BY COUNTRY, DES_PROD, YEAR_;

Q2: SELECT COUNTRY, SITUATION, YEAR_, SUM (Sales) AS Total_Sales
FROM Fact_Sales FS, DIM_GEOGRAPHY G, DIM_DATE D, DIM_PRODUCT P,
DIM_CUSTOMERS C
WHERE country='COUNTRY1' and YEAR_='2022'
AND FS.ID_CUST=C.ID_CUST AND FS.ID_PROD= P.ID_PROD
AND FS.ID_GEO=G.ID_GEO AND FS.ID_DATE=D.ID_DATE
GROUP BY COUNTRY, SITUATION, YEAR_;

Q3: SELECT COUNTRY as C, YEAR_ as Y, SUM (Sales) AS Total_Sales
FROM Fact_Sales FS, DIM_GEOGRAPHY G, DIM_DATE D
WHERE country='COUNTRY1' and YEAR_='2022'
AND FS.ID_GEO=G.ID_GEO AND FS.ID_DATE=D.ID_DATE
GROUP BY COUNTRY, YEAR_;
```

Among these algorithms we find:

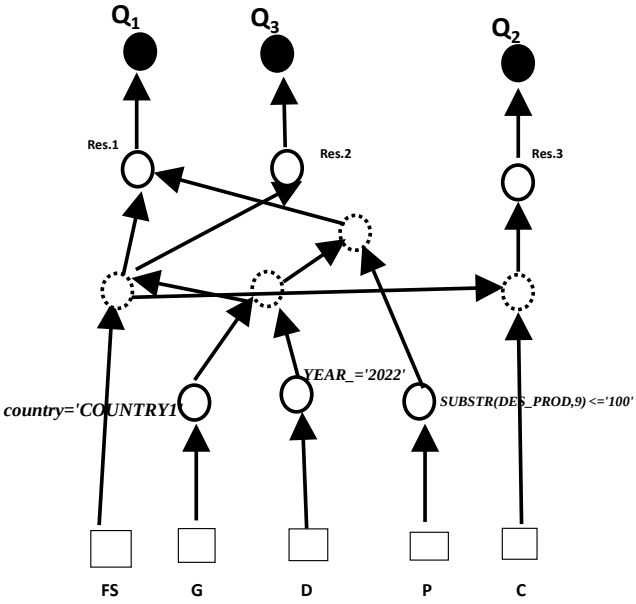
Harinarayan, V., *et al.* (1996) in [6] proposed the Greedy algorithm based on space costs defined by the number of rows in the view associated with each view v. The principal of the algorithm is to select some set S of k views that maximizing the benefit $B(v, S)$ defined as follows.

1) *For each* $w \leq v$, *define the quantity* B_w *by*:

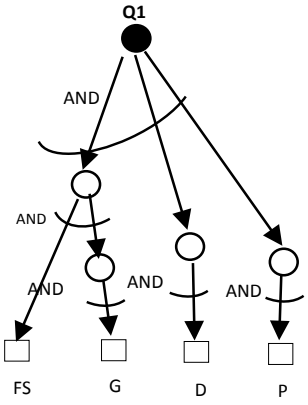
(a) *Let* u *be the view of least cost in* S *such that* $w \leq u$. *Note that since the top view is in* S, *There must be at least one such view in* S.



(A)



(B)



(C)

Figure 6. (A) Data cube lattice structure; (B) AND-OR DAG (Directed Acyclic Graph), structure; (C) MVPP structure.

(*b*) *If* $C(v) < C(u)$, *then* $B_W = C(v) - C(u)$.

Otherwise, $B_W = 0$.

2) *Define* $B(v,S) = \sum_{W \le U} B_W$.

Table 2 describes the Greedy Algorithm.

Lin, W.-Y. and Kuo I.-C. (2004) in [29] proposed the Greedy Genetic Algorithm by merging the genetic algorithm presented in Table 3 below with the greed algorithm described below. The algorithm is based on two metrics: the Query Cost describing the cost to evaluate query and Maintenance Cost specifying the cost to update cube where the maintenance technique adopted is precomputation from scratch of materialized cubes. Let *La* lattice of *N* Cubes $C = \{c_1, c_2, \cdots, c_i, \cdots, c_n\}$ from table Fact *R*, A Set of Query $Q = \{q_1, q_2, \cdots, q_i, \cdots, q_k\}$, each query has its own execution frequency which is shown as Query Frequency Values $F = \{fq_1, fq_2, \cdots, fq_i, \cdots, fq_k\}$, Update Frequency of cubes $G = \{gc_1, gc_2, \cdots, gc_i, \cdots, gc_n\}$, Constraint Space *S* and Ωι = (*L*, *C*, *R*, *Q*, *F*, *G*, *S*). The objective of the algorithm is to find a solution Ωι who is a subset of *C*, say *M*, that can minimize the following cost function defined such fitness function $\mu(\cdot)$ under the constraint that $\sum_{c \in Mc} |c| \le S$:

$$\mu(\cdot) = \sum_{i=1}^{k} fq_i E(q_i, M) + \sum_{c \in M} gcU(c, M) \quad (1)$$

First Member is cost to evaluate query q_i and second member is the cost to update cube *c*.

Table 2. Greedy algorithm.

```
S = {top view};
        for i = 1 to k do begin
                select that view v not in S such that B(v, S)
is maximized;
S = S union {v};
        end;
      resulting S is the greedy selection;
```

Table 3. Genetic algorithm.

```
Initialize parameters;
Generate a population P randomly;
Nbr_generation ← 1;
while Nbr_generation ≤ max_gen do
Clear the new population P;
Evaluate each individual in P using a fitness function μ(·);
while |P| ≤ population_size do
Select two parents from P;
Execute crossover;
Execute mutation;
Place the offspring into P;
End while
P ← P;
Nbr_generation ++;
end while
```

Necir, H. and Drias H. (2011) in [30] proposed A Distributed Clustering with Intelligent Multi Agents System for Materialized Views Selection; the approach is divided into three phases: the pre-processing of queries, generation of views candidates and selection of a final configuration. The goal is to use multi-agents generated with k-means clustering algorithm that use coordinate agent ***COA*** to dispatch the information's between the regional cluster *CA*. The pre-processing based on the Extractor-Agent that extract all restriction and joint predicates to build the predicates usage matrix of which each row and column represents one query and its predicates where the value in {0, 1} according to query q_i if uses the predicate p_i or no. Moreover, in the generation of views candidate's phase the regional clustering has been created using the last predicates matrix formed in the above phase. Finally, performing the configuration having views which executing query with respect to the storage constraint and reducing the total query processing cost.

Vijay Kumar, T. and Kumar, S. (2012) in [31] proposed the Genetic Algorithm Materialized view selection using genetic algorithm, this approach is similar to Lin, W.-Y. and Kuo, I.-C. proposed in [29] but the differences are appeared on the fitness function and the probability added to the crossover and mutation function. The fitness function is defined by a total view evaluation cost (*TVEC*) that calculate summation of size for materialized view and a size of smallest ancestor for not materialized views described as follow:

$$TVEC = \sum_{i=1 \wedge SMvi=1}^{N} Size(Vi) + \sum_{i=1 \wedge SMvi=0}^{N} SizeSMA(Vi) \quad (2)$$

Roy, S., *et al.* (2015) in [32] proposed Materialized view construction based on clustering technique. The approach based on attribute similarity measured by Jaccard Index presented for sets *A* and *B* as bellow and formation of clusters using weighted connected graphs:

$$Jaccard(A,B) = \frac{|A| \cap |B|}{|A| \cup |B|} \quad (3)$$

For example

$A = \{a,b,c,d,e,f\}$ *and* $B = \{a,e,g,h\}$, the Jaccard similarity can be calculated as $Jaccard(A,B) = \frac{2}{8} = 0.25$, because $A \cap B = \{a,e\}$ and $A \cup B = \{a,b,c,d,e,f,g,h\}$.

The authors calculate the Attribute similarity matrix for each couple of attributes in all queries such as for *M* queries with *n* attributes and formed an Attribute Usage Matrix ($m \times n$). In the attribute similarity matrix, all values that are greater than or equal to an average similarity value are taken into account when constructing graphs to form clusters. This clustering technique receive as parameters the Relation *R* with *n* attributes $R = \{A_1, A_2, \cdots, A_n\}$, the set off queries $Q = \{Q_1, Q_2, \cdots, Q_n\}$ and Cut-off Average Similarity, generate the Attribute Usage Matrix and Attribute similarity Matrix and return the cluster that present the set of selected of Materialized Views. The entire algorithm is shown in **Table 4**.

Table 4. Clustering technique.

Input: $R(A_1, A_2, \cdots, A_m)$: *The relation*; $Q = \{Q_1, Q_2, \cdots, Q_n\}$: *The set of user queries that will run on* ***R***; *cut-off*: *Cut-off Average Similarity*.
Output: *C*: *The set of materialized views*.
Step* 1:** *Construct the* $\boldsymbol{m \times n}$ *Attribute Usage Matrix where m is the number of attributes and* ***n *is the number of queries*.
***Step* 2:** *Construct the* $\boldsymbol{n \times n}$ *Attribute Similarity Matrix from the Attribute Usage Matrix*.
/* *In step* 3 *the Graphs & the corresponding clusters are constructed* */
***Step* 3:** $\forall (A_i, A_j) \in$ *Attribute Similarity Matrix, identify the attribute pair similarity values so that* $J(A_i, A_j) \geq$ *cut-off*.
Store these values in a list L in descending order.
Store the number of elements in L into a variable count.
a) *If count* = 0 *then go to step* 5. /* *The algorithm fails to draw any graph. Therefore, no materialized view can be constructed* */
b) *If count* > 0 *then Repeat until list L is empty*
i) *Pop the ith* $(1 \leq I \leq count)$ *element from L. Store the value of the element into avariable* x $(0 \leq x \leq 1)$.
ii) $\forall$ *Attribute pairs* $(A_i, A_j) \in$ *Attribute Similarity Matrix*
If $J(A_i, A_j) = x$, *then* $G_i \leftarrow G_i \cup A_i \cup A_j$ /* *Pair of attributes is inserted in* G_i, *where* G_i *represents the* i^{th} *graph for* i^{th} *highest similarity value*. */
iii) $C_i \leftarrow G_i$ /*C_i *represents the* i^{th} *cluster for* i^{th} *highest similarity value*. */
iv) $C \leftarrow C \cup C_i$ /* *Each cluster* C_i *is added to the set of clusters C* */
/* *In step* 4 *the clusters are validated and materialized views are formed*/
***Step* 4:** $\forall C_i \in C$, *Calculate* $(C_i)_{avg}$ *by equation*:

$$(C_i)_{avg} = \frac{\sum_{i=1}^{\binom{n}{2}} J(A_i; A_j)}{\binom{n}{2}} \quad \text{where } i \neq j$$

If $((C_i)_{avg} <$ *cut-off then* $C \leftarrow C - C_i$
/* *C represents the set of materialized views where each cluster* C_i *represents a valid materialized view* */
Step* 5:** ***End.

Anjana Gosaina, Heenaa Procedia Computer Science 79 (2016) in [33] proposed a linear cost model for materialized view selection problem based on the Particle Swarm Optimization algorithm described in **Table 5**, the model is initialized by an arbitrary population of N particles candidate. Each particle has characterized by certain speed and position in a space with D dimensions and continually changes their speed and position according to the best position found by itself -particle best (-pbest) and by the all population -global best (-gbest) so far. Refers to [33] for more details. The authors evaluate query processing cost under the space constraint and compared with the results provided by the genetic algorithm. Cost of answering the query defined by the number of rows to be accessed in the corresponding cube for the query. Furthermore the algorithm starts with similarity parameters that are used by Lin, W.-Y. and Kuo, I.-C. (2004) in [29] using the genetic algorithm. But the difference between these two techniques is fitness function: Anjana Gosaina use the first member of the function used by W.-Y. and Kuo, I.-C.

Table 5. Particle swarm optimization algorithm.

Initialize all parameters such as swarm, velocity, position and others) *Evaluate each particle and find their local best and global best position.* *Repeat* { *Evaluate each particle and update their local best and global best position.* } *until termination*

Azgomi, H. and Sohrabi, M. K. (2018) in [25] proposed the A game theory-based framework for materialized view selection in data warehouses considering the two opposite players of game are processing cost and view maintenance cost. The proposed algorithm described as follows:

Game: The game is a step-by-step situation to solve the problem of selecting materialized views.

First player: The first player wants to select the views with the highest maintenance costs.

Second player: The second player is greedy to choose views with higher query processing costs.

Strategy: There are two strategies possible for the game depending to the used method for materialized view selection: Greed approach for pure strategy or random method for mixed strategy. Moreover, both players start the game with all possible views as the input of their strategies.

Payoff function: $\text{PofP}_1 = C_{vm} - C_{qp}$; and $\text{PofP}_2 = C_{qp} - C_{vm}$.

Where C_{vm} = view maintenance cost and C_{qp} = query processing cost of a view.

Equilibrium: Equilibrium defined as a state of the game in which a player cannot improve by changing the conditions of his game, otherwise, when a payoff does not increase in a single step. Obviously, reducing the payoff value of a player or a part of the players does not mean reaching the equilibrium.

Type of game: Since the decision of the players, doesn't depend on the previous decisions, the total time of the solution is reduced by players using simultaneous selection of views, in this case the game can be considered as a static game. In addition, the game is assumed as a non-zero-sum game in which the total benefits and losses of the players are more or less than zero.

Azgomi, H. and Sohrabi, M. K. (2019) in [8] proposed "A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments. The purpose method used MVPP structure to present all views, and considered bit-string with length equal to the number of views of the MVPP is the initial population of the algorithm. Each case of the bit-string takes1 or 0 depending on if the view is a part of the solution or no. the process of the algorithm is divided on six steps described as follow in **Figure 7**.

Prakash, J. and Kumar, T. V. (2021) in [34] proposed the A multi-objective approach for materialized view selection using on a non-Pareto based genetic algorithm MVS_{VEGA}, that selects Top-K views for materialization from a multi-dimensional lattice that minimizing simultaneously the two costs C_{MV} and C_{NMV} of $TVEC$ given below:

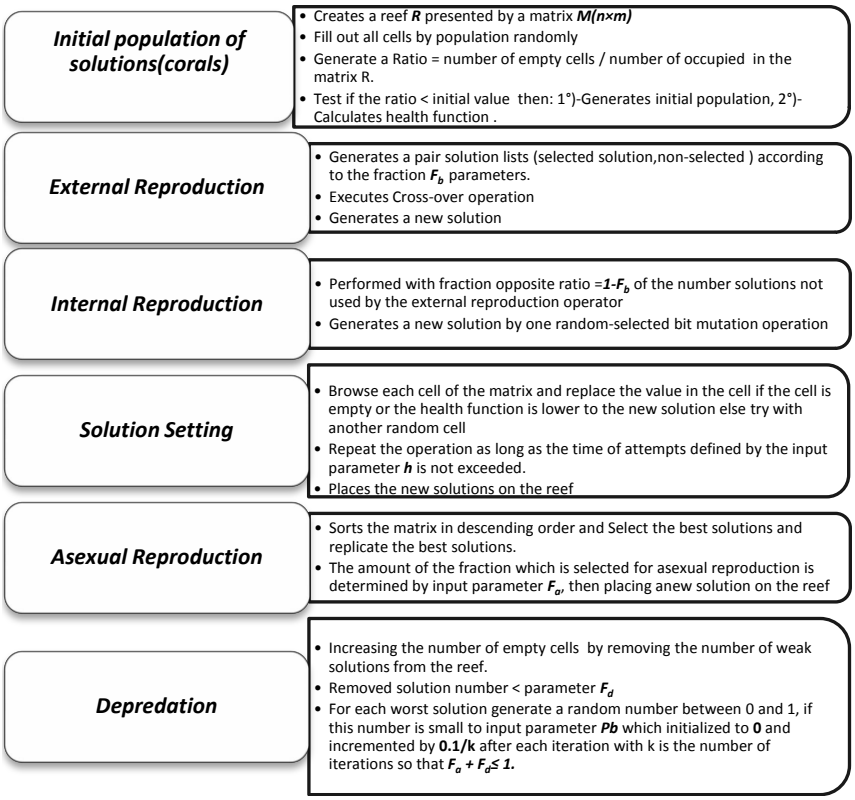


Figure 7. A novel coral reefs optimization algorithm.

$$TVEC\left(V_{Top\text{-}k}\right) = C_{MV} + C_{NMV} \tag{4}$$

where

$$C_{MV} = \sum_{i=1 \wedge SMvi=1}^{N} Size\left(Vi\right) \tag{5}$$

and:

$$C_{NMV} = \sum_{i=1 \wedge SMvi=0}^{N} SizeSMA\left(Vi\right) \tag{6}$$

INPUT: Size of each view L, P_c: probability of crossover, P_m: probability of mutation, K: number of views to be selected and G: the maximum number of generations

Core: Step 1: Initialization which generates randomly the population of the Top-K views as PTKV.

Step 2: Divide a PTKV into two equilibrium sub-populations PTKV1 and PTKV2 with size equal to PTKV/2.

Step 3: Compute the fitness for PTKV1 and PTKV2 using C_{MV} and C_{NMV} respectively as a two bi-objective optimization problem.

Step 4: Select the Top-K views from PTKV1 and PTKV2 using the proportionate selection based on the fitness of the Top-K views, and Compute subpopulation of Top-K views SPTKV1 and SPTKV2 using the corresponding objective function.

Step 5: Combine the Top-K views in SPTKV1and SPTKV2 and generate a mating population of Top-K views MPTKV.

Step 6: Apply the modified crossover and the random mutation operators to generate the offspring population of the Top-K views.

OUTPUT: *Top-K* views *TKV*

Azgomi, H. and Sohrabi, M. K. (2021) in [19] proposed a map-reduce-based approach for creating Multiple Views Processing Plan MVPP structure in a reasonable time in data warehouses for big data applications using the Map-Reduce Model and the hashing technique that used to reduce the search space of a problem. The model map-reduce model consists of two main functions, $MAP_{function}$ and $REDUCE_{function}$. The MAP function processes a pair of (key, value) inputs to create a set of intermediate (key, value). The $REDUCE_{function}$ integrates all intermediate data related to each key, and produces the result. The structure of the MVPP was presented formed n MVPPs from $n!$ possible MVPPs, and selected the MVPP that had the lowest cost as the final solution, these methods was $O(n)$, where n was the number of queries. The total cost of the MVPP in this method is calculated by the summation of the query processing cost and the view maintenance cost of all views.

3. Comparative Study of the Materialized View Approach

As already defined in the previous section, the goal of all approaches developed in the literature to resolve a problem of materialized view selection is to minimize the query processing cost under storage constraint. To achieve this aim, each approach describes its own process using heuristic algorithms, genetic algorithms, game theory or clustering techniques. These processes differ from each other by their starting parameters and the main function that allows reaching the main objectives. Table 6 presented the comparative study and summarizes the differences between these approaches.

Table 6. Comparative table of materialized view selection algorithms.

Approaches	*Nbr and list of parameters*	*Functions*	*Output*	*Structure*	*Technics*	*Constraints*
Greedy algorithm	3	$V=\{v_1,v_2,\cdots,v_n\}$ Set of candidate view, $C=\{c_1,c_2,\cdots,c_n\}$ where c_i as the associated size of each view candidate v_i K: number max of view to be materialized,	$\max(B(v,S))=\max(\sum_{w\le U}B_w)$	S: Subset of materialized views with Max($B(v, S)$)	Data cube lattice	Select a Subset S of k views v including a top view where the benefit $B(v, S)$ is maximized. For each view u relative to S calculate $B(v,S)=\sum_{w\le U}B_w$. *Where w is all descending of u.*	K: number max of view to be materialized
Greedy Genetic Algorithm	6	lattice of N Cubes $C=\{c_1,c_2,\cdots,c_i,\cdots,c_n\}$, A Set of Query $Q=\{q_1,q_2,\cdots,q_i,\cdots,q_k\}$, Frequency Values for each query $F=\{fq_1,fq_2,\cdots,fq_i,\cdots,fq_k\}$, Update Frequency of cubes $G=\{gc_1,gc_2,\cdots,gc_i,\cdots,gc_n\}$, Constraint Space S.	$\mu(\cdot)=\sum_{i=1}^{k}fq_iE(q_i,M)$ $+\sum_{c\in M}gcU(c,M)$	$\Omega\iota=(L, C, R, Q, F, G, S)$	*Data cube lattice*	Genetic Greedy Method.	Storage Space, $\sum_{c\in Mc}\lvert c\rvert\le S$
Genetic Algorithm	4	Lattice of Views with size of each view, P_c: Probability of crossover, P_m: Probability of mutation, G: Pre-defined number of generations.	$TVEC=\sum_{i=1\wedge SMvi=1}^{N}Size(Vi)$ $+\sum_{i=1\wedge SMvi=0}^{N}SizeSMA(Vi)$	Top-T Views.	*Data cube lattice*	Genetic method.	G: Number of generations
A Distributed Clustering with Intelligent Multi Agents System	1	Queries workload.	$DIC=\frac{\sum_{i=1}^{j=k}Dist(O_j,O_i)}{K}$ $Dist(O_j,O_i)$ $=\sum_{k=1}^{m}\lvert M(O_i,q_k)-M(O_j,q_k)\rvert$	Cluster with configuration provide minimal queries cost.	NONE	Generates a predicates usage matrix ***M*** with row presented by all queries and column defined by the restriction predicate and joint predicate. Use a distributed clustering based on ***k-means algorithm*** created and managed by ***COA***. Each COA responsible of set of Cluster Agents ***(CAs)***.	Storage Space, MinDist.

Continued

Clustering technique	3	Set of relation: $R(A_1, A_2, \cdots, A_m)$; The set of queries: $Q = \{Q_1, Q_2, \cdots, Q_n\}$ Average Similarity: cut-off:.	$TVEC = \sum_{i=1 \wedge SMvi=1}^{N} Size(Vi) + \sum_{i=1 \wedge SMvi=0}^{N} SizeSMA(Vi)$	Cluster of materialized views	NONE	Construct Attribute Usage Matrix M attributes * N queries. Generate Attribute Similarity Matrix M* M between attributes. Calculate similarity by $Jaccard(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$	$J(A_i, A_j) = J(A_i, A_k) \geq$ cut-off. So
Particle Swarm Optimization Algorithm (PSO)	6	lattice of N Cubes $C = \{c_1, c_2, \cdots, c_i, \cdots, c_n\}$, A Set of Query $Q = \{q_1, q_2, \cdots, q_i, \cdots, q_k\}$, Frequency Values for each query $F = \{fq_1, fq_2, \cdots, fq_i, \cdots, fq_k\}$, Update Frequency of cubes, cube invoking frequency $CF = (fc_1, fc_2, \cdots, fc_n)$, Constraint Space S.	$fm = \min\left(\sum_{i=1}^{k} fq_i E(q_i, M)\right)$	Binary string of length n bits [0, 1] witch is set of Cubes M to minimizing fm	*Data cube lattice*	Genetic Algorithm. Particle Swarm Optimization Algorithm	Storage Space $\sum_{c \in Mc} \lvert c \rvert \leq S$
coral reefs optimization algorithm (CROMVS)	8	MVPP, K = Number of generating populations, M = Number of Queries, N = Number of Relations, H = threshold times, F_a = Fraction of asexual reproduction, F_b = Ratio of number of selected solutions, F_d = Fraction of the worst solutions	$f_x = \sum_{i \in M}\left(\sum_{q \in Q} ef_q * C_i^q\right) - \left(\sum_{u \in Prej \cap x} \sum_{q \in Q} ef_q * C_u^q\right) + \sum_{r \in R} uf_r * C_i^r\right)$	MV with MAX(f_x)	***MVPP***	Coral reefs algorithm, is Meta-heuristic algorithm based on coral reproduction and coral reefs formation which performed using: external sexual reproduction, internal sexual reproduction, and asexual reproduction	Population List with Max(f_x)
Multi-Objective MONPGA.	5	L: Size of each view, P_c: probability of crossover, P_m: probability of mutation, K: number of views to be selected and G: the maximum number of generations	$TVEC(V_{Top\text{-}k}) = C_{MV} + C_{NMV}$	Top-K views TKV	*Data cube lattice*	Genetic Algorithm.	K: number of views to be selected and G: the maximum number of generations
A game theory-based framework (GTMVS)	4	$R = \{R_1, R_2, \cdots, R_r\}$: Set of base relations uf_i: update frequency for R_i. $Q = \{Q_1, Q_2, \cdots, Q_q\}$: Set of queries, ef_i execution frequency for Q_i.	$g_x = \sum_{i \in M}\left(\sum_{r \in R} uf_r * C_i^r + \sum_{q \in Q} ef_q * C_i^q\right)$	select the optimal set $M = \{M_1, M_2, \cdots, M_m\}$ with lowest cost represented by both sets of player1 and player2.	***MVPP***	Game theory who the Players of game are: Query processing cost and view maintenance cost. Player strategy used TSGV. [35]	M: list of nodes in MVPP-Player 1 union Player 2. materialized view List
Map-Reduce model (MR-MVPP)	7	Q: workload of queries f_q: list of query frequencies R: base relations f_u: list of update frequencies b: number of categories A: a number between 0 and 1 for hash function t: threshold of similarity	$QC_i = f_{qi} * \sum_{\{j \mid Q_i \in Query(V_j)\}} ACost(V_j)$	MVPP that has the least total cost.	***MVPP***	The map-reduce programming Model; The hashing technique; Use 4 algorithms. are: MR-MVPP, SSJoin, map, and reduce. Calculate similarity by Jaccard Function. $Jaccard(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$	MVPP with least total cost = Min(QC_i)

4. Conclusion

In this paper, an overview of algorithms dealing with the materialized view selection problems has been given. Starting with the determination of the global context of the materialized view defining why the query needed to be materialized, followed by the comparison of the approaches developed in the literature. In conclusion, we have found that most algorithms prefer to use the multidimensional aspect to benefit from their advantages in solving multivariate problems defined in our case by the cost of reading and maintaining the materialized views. In order to minimize this cost in data warehouse environment, each approach defines its strategy to achieve this goal with a minimal cost and a reasonable complexity level. Our research axis focuses on the multidimensional model whose success has already been demonstrated by the algorithms used and we will propose our optimal solution.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kimball, R. (1996) The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, Inc., Hoboken.

[2] Ravat, F. (2007) Modèles et outils pour la conception et la manipulation de systèmes d'aide à la décision. Université des Sciences Sociales, Toulouse.

[3] Inmon, W.H. (1996) Building the Data Warehouse. John Wiley & Sons, Inc., Hoboken, 5 p.

[4] Kimball, R. (2013) The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd Edition, John Wiley & Sons, Inc., Hoboken.

[5] Lakshmanan, L.V., Pei, J. and Zhao. Y. (2003) Efficacious Data Cube Exploration by Semantic Summarization and Compression. *Proceedings of the* 29*th International Conference on Very Large Data Bases*, Berlin, 9-12 September 2003, 1125-1128. https://doi.org/10.1016/B978-012722442-8/50121-X

[6] Harinarayan, V., Rajaraman, A. and Ullman, J.D. (1996) Implementing Data Cubes Efficiently. *ACM SIGMOD Record*, **25**, 205-216. https://doi.org/10.1145/235968.233333

[7] Badri, M. (2007) Mise à jour incrémentale des agrégats: Cas des indicateurs ROLAP dans les entrepôts de données CRIP5. IUT de Paris Descartes, Paris.

[8] Azgomi, H. and Sohrabi, M.K. (2019) A Novel Coral Reefs Optimization Algorithm for Materialized View Selection in Data Warehouse Environments. *Applied Intelligence*, **49**, 3965-3989. https://doi.org/10.1007/s10489-019-01481-w

[9] Lee, K.Y. and Kim, M.H. (2006) Efficient Incremental Maintenance of Data Cubes. *Proceedings of the* 32*nd International Conference on Very Large Data Bases*, Seoul, 12-15 September 2006, 823-833.

[10] Papadias, D., *et al.* (2001) Efficient OLAP Operations in Spatial Data Warehouses. *Proceedings of the* 7*th International Symposium on Advances in Spatial and Temporal Databases*, Redondo Beach, 12-15 July 2001, 443-459. https://doi.org/10.1007/3-540-47724-1_23

[11] Glorio, O. and Trujillo, J. (2008) An MDA Approach for the Development of Spatial Data Warehouses. *Proceedings of the* 10*th International Conference on Data Warehousing and Knowledge Discovery*, Turin, 1-5 September 2008, 23-32.

[12] Vaisman, A. and Zimányi, E. (2014) Spatial Data Warehouses, in Data Warehouse Systems. Springer, Berlin, 427-473.

[13] Mateus, R.C., *et al.* (2016) Spatial Data Warehouses and Spatial OLAP Come towards the Cloud: Design and Performance. *Distributed and Parallel Databases*, **34**, 425-461. https://doi.org/10.1007/s10619-015-7176-z

[14] Wang, B. and Wang, Y. (2021) Big Data in Safety Management: An Overview. *Safety Science*, **143**, Article No. 105414. https://doi.org/10.1016/j.ssci.2021.105414

[15] Silva, R.R., Hirata, C.M. and de Castro Lima, J. (2020) Big High-Dimension Data Cube Designs for Hybrid Memory Systems. *Knowledge and Information Systems*, **62**, 4717-4746. https://doi.org/10.1007/s10115-020-01505-9

[16] Ma, Y., *et al.* (2017) The Selection and Placement Method of Materialized Views on Big Data Platform of Equipment Condition Assessment. *IOP Conference Series*:

Materials Science and Engineering, **199**, Article No. 012105. https://doi.org/10.1088/1757-899X/199/1/012105

[17] Santos, M.Y., Martinho, B. and Costa, C. (2017) Modelling and Implementing Big Data Warehouses for Decision Support. *Journal of Management Analytics*, **4**, 111-129. https://doi.org/10.1080/23270012.2017.1304292

[18] Jukić, N., *et al.* (2015) Augmenting Data Warehouses with Big Data. *Information Systems Management*, **32**, 200-209. https://doi.org/10.1080/10580530.2015.1044338

[19] Azgomi, H. and Sohrabi, M.K. (2021) MR-MVPP: A Map-Reduce-Based Approach for Creating MVPP in Data Warehouses for Big Data Applications. *Information Sciences*, **570**, 200-224. https://doi.org/10.1016/j.ins.2021.04.004

[20] Dkaich, R., Azami, I.E. and Mouloudi, A. (2017) XML OLAP Cube in the Cloud towards the DWaaS. *International Journal of Cloud Applications and Computing*, **7**, 47-56. https://doi.org/10.4018/IJCAC.2017010103

[21] Liu, X., Thomsen, C. and Pedersen, T.B. (2014) CloudETL: Scalable Dimensional ETL for Hive. *Proceedings of the* 18*th International Database Engineering & Applications Symposium*, Porto, 7-9 July 2014, 195-206. https://doi.org/10.1145/2628194.2628249

[22] Lambert, M. (2006) Développement d'une approche pour l'analyse solap en temps réel: Adapatation aux besoins des activités sportives en plein air. Université Laval, Québec.

[23] Adamson, C. (2006) Mastering Data Warehouse Aggregates: Solutions for Star schema Performance. John Wiley & Sons, Inc., Hoboken.

[24] Gupta, H. (1997) Selection of Views to Materialize in a Data Warehouse. *Proceedings of the* 6*th International Conference on Database Theory*, Delphi, 8-10 January 1997, 98-112. https://doi.org/10.1007/3-540-62222-5_39

[25] Yang, J., *et al.* (1997) Algorithms for Materialized View Design in Data Warehousing Environment. *Proceedings of the* 23*rd International Conference on Very Large Data Bases*, Athens, 25-29 August 1997, 136-145.

[26] Lee, M. and Hammer, J. (2001) Speeding up Materialized View Selection in Data Warehouses Using a Randomized Algorithm. *International Journal of Cooperative Information Systems*, **10**, 327-353. https://doi.org/10.1142/S0218843001000370

[27] Gupta, H. and Mumick, I.S. (2005) Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering*, **17**, 24-43. https://doi.org/10.1109/TKDE.2005.16

[28] Derakhshan, R., *et al.* (2006) Simulated Annealing for Materialized View Selection in Data Warehousing Environment. *Proceedings of the* 24*th IASTED International Conference on Database and Applications*, Innsbruck, 13-15 February 2006, 89-94.

[29] Lin, W.-Y. and Kuo, I.-C. (2004) A Genetic Selection Algorithm for OLAP Data Cubes. *Knowledge and Information Systems*, **6**, 83-102. https://doi.org/10.1007/s10115-003-0093-x

[30] Necir, H. and Drias, H. (2011) A Distributed Clustering with Intelligent Multi Agents System for Materialized Views Selection. *Proceedings of the Sixth International Conference on Intelligent Systems and Knowledge Engineering*, Shanghai, December 2011, 417-426. https://doi.org/10.1007/978-3-642-25661-5_54

[31] Vijay Kumar, T. and Kumar, S. (2012) Materialized View Selection Using Genetic Algorithm. *International Conference on Contemporary Computing*, Noida, 6-8 August 2012, 225-237. https://doi.org/10.1007/978-3-642-32129-0_26

[32] Roy, S., Ghosh, R. and Sen, S. (2015) Materialized View Construction Based on

Clustering Technique. *International Conference on Computer Information Systems and Industrial Management*, Ho Chi Minh City, 5-7 November 2014, 254-265. https://doi.org/10.1007/978-3-662-45237-0_25

[33] Gosain, A. (2016) Materialized Cube Selection Using Particle Swarm Optimization Algorithm. *Procedia Computer Science*, **79**, 2-7. https://doi.org/10.1016/j.procs.2016.03.002

[34] Prakash, J. and Kumar, T.V. (2021) A Multi-Objective Approach for Materialized View Selection. In: *Research Anthology on Multi-Industry Uses of Genetic Programming and Algorithm*, IGI Global, Hershey, 512-533. https://doi.org/10.4018/978-1-7998-8048-6.ch026

[35] Sohrabi, M.K. and Azgomi, H. (2017) TSGV: A Table-Like Structure-Based Greedy Method for Materialized View Selection in Data Warehouses. *Turkish Journal of Electrical Engineering and Computer Sciences*, **25**, 3175-3187. https://doi.org/10.3906/elk-1608-112