Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 3175 – 3187
© TÜBİTAK
doi:10.3906/elk-1608-112

TSGV: a table-like structure-based greedy method for materialized view selection in data warehouses

Mohammad Karim SOHRABI[*], **Hossein AZGOMI**
Department of Computer Engineering, Semnan branch, Islamic Azad University, Semnan, Iran

Received: 10.08.2016 • **Accepted/Published Online:** 23.12.2016 • **Final Version:** 30.07.2017

Abstract: Since a data warehouse deals with huge amounts of data and complex analytical queries, online processing and answering to users' queries in data warehouses can be a serious challenge. Materialized views are used to speed up query processing rather than direct access to the database in on-line analytical processing. Since the large number and high volume of views prevents all of the views from being stored, selection of a proper subset of views to materialization is inevitable. Proposing an appropriate method for selecting the optimal subset of views for materialization plays an essential role in increasing the efficiency of responding to data warehouse queries. In this paper, a greedy materialized view selection algorithm is represented, which selects a proper set of views for materialization from a novel table-like structure. The information in this table-like structure is extracted from a multivalue processing plan. This table-like structure-based greedy view selection (TSGV) method is evaluated using the queries of an analytical database, and the query-processing and view maintenance costs of the selected subset are both considered in this evaluation. The experimental results show that TSGV operates better than previously represented methods in terms of time.

Key words: Materialized view selection, multivalue processing plan, data warehouse, on-line analytical processing, decision support systems

1. Introduction

The main information support part of many decision support systems is a data warehouse, which is a subject-oriented, integrated, time-variant, and nonvolatile collection of data [1]. A data warehouse has a large number of records collected from different data sources. Query processing is a very time-consuming operation in data warehouses due to the large volume of data and the complex nature of analytical queries. Materializing intermediate views that are constructed to answer some queries can speed up the processing and answering of the other queries.

Since a very large number of intermediate views can be constructed in processing several queries of a data warehouse, it is impossible to store all of these views. Furthermore, when modifying the base tables, it is necessary to update materialized intermediate views, which causes another time overhead [2]. These issues lead to an important problem called view selection for materialization, or materialized view selection, which aims at selecting a (quasi-)optimal subset of intermediate views for storage in the available memory space.

Materialized view selection methods have used different structures for organizing the views obtained from queries. One of the most common structures is multivalue processing plan (MVPP), a forest in which roots are

[*]Correspondence: amir_sohraby@yahoo.com

the queries, leaves are the base tables, and other middle nodes include different relational algebra operators, such as selection, projection, and join operators, which are used to create the given queries [3]. AND-OR DAGs [4] and data cube lattices [5] are other well-known structures of materialized view selection methods. In addition to algorithms using the above structures to organize intermediate extracted views of queries, there are some methods that do not use any structure and only change the queries by rewriting them such that other queries can use the obtained results. Some other methods also identify the candidate views based on syntactic analysis of workload. These methods study the workload and select a subset of relations that store one view or more if and only if selection of this subset causes a significant decrease in workload cost.

Based on their limitation assumption, materialized view selection methods can be divided into three categories. Some of the proposed methods consider no limitations for selecting the views. Therefore, the view selection problem selects a set of views to minimize the query processing and view maintenance costs. However, some other methods consider the limitation of storage space. In these methods, an appropriate subset of views should be selected considering the limitation of storage space. Some other methods consider limitation in view maintenance cost; in these methods, the time required for keeping the stored views updated is limited.

In this paper, a new view selection method, called table-like structure-based greedy view selection (TSGV), is presented. This method uses a table-like structure with information extracted from MVPP structure nodes and employs a greedy algorithm to select appropriate views from the present nodes this table-like structure. Both view maintenance costs and query processing costs can be considered in TSGV.

The remaining parts of the paper are organized as follows: Section 2 represents a literature review of the problem. A formal definition of materialized view selection and the new proposed method are proposed in Section 3. Furthermore, TSGV is explained with a detailed example and its table-like structure is described in Section 3, along with its algorithm. In Section 4, the proposed method is practically and experimentally evaluated and compared to other similar methods. The work is concluded in Section 5.

2. Related works

Since 1979, when the first algorithm for materialized view selection was presented [6], great efforts have been made to provide better and more efficient solutions for this problem. Several view selection algorithms can be divided into four categories: deterministic algorithms, randomized algorithms, hybrid algorithms, and constraint programming methods [7].

Deterministic algorithms are the first category of algorithms considered by researchers. Several methods in this category were presented between 1996 and 2005. Some of these algorithms have used MVPP as their query organizer's structure [3,8]. An MVPP-based greedy algorithm was presented in [3] for view selection that minimizes both query processing and view maintenance costs without any limitations. To guarantee an optimal solution, this paper has modeled the materialized view selection problem as a 0–1 integer programming problem. Another framework for designing materialized views was introduced in [8]. In fact, this paper defined MVPP for the first time.

AND-OR DAG is another structure that has been used by some view selection methods based on deterministic algorithms [4,9]. The method presented in [4] selects views from the AND-OR DAG structure using a greedy heuristic. Combining volcano search strategy and a greedy heuristic and operating some simple modifications on it, this paper introduced three cost-based heuristic algorithms. The problem with all these methods is that they only minimize the query-processing cost without paying any attention to view maintenance cost. Mistry et al. [9] proposed two new concepts for materialization: transient materialization for selecting shared expressions and indices of queries, and permanent materialization for selecting other expressions and indices. Selecting views for transient and permanent materialization, along with selecting the best maintenance plan for each view, are three issues that have been addressed by the proposed framework of this work.

More recent deterministic-algorithm–based methods have used a cube lattice as their structures [10,11]. An extended-lattice–based greedy algorithm for view selection in distributed data warehouses was proposed in [10]. The concept of an aggregation lattice has been extended in this paper to capture the distributed semantics. Storage, maintenance, communication, and computation costs have all been supported by this extended lattice. The cost model in [10] was divided into monotonic and nonmonotonic cost models, and a monotonic benefit function that is only related to the size of the evaluated views was used. Another distributed approach to view selection was presented in [11]. Under storage limitation, a greedy algorithm was proposed in [11], which used an extended cube lattice to address materialized view selection.

There are other deterministic methods that do not use any structure to find the optimal subset of views for materialization. For example, Theodoratos et al. [12] studied the materialized view problem based on space-limited query rewriting and [13] is based on a syntactical analysis from workload that selects views with limited memory, disregarding view maintenance cost.

Although deterministic-algorithm–based methods are a significant part of materialized view selection approaches, newer methods have tried to reduce the time complexity of view selection using heuristics and settling for suboptimal responses. Several artificial intelligence techniques and soft computing tools have been used to make more efficient randomized algorithms for materialized view selection. Genetic algorithms and other types of evolutionary computing, simulated annealing and other types of optimization methods, and data-mining techniques are the most common approaches used for this category of view selection methods. Similar to the first category of algorithms, several methods in this category employ different well-known query structures to improve their efficiency [14–18].

An MVPP-based unbounded genetic approach was presented for view selection in [14]. A central and a parallel simulated annealing for materialized view selection were proposed in [15] and [16], respectively. Both algorithms were based on MVPP, and both were unbounded. Randomized local search algorithms were employed for view selection in [17]. The proposed algorithm provides a near-optimal solution within limited time and limited storage size. Furthermore, it is appropriate for dynamic environments. Another genetic algorithm for materialized view selection was presented in [18], in which its initial population is generated based on database tuning to allow the genetic algorithm to converge quickly.

Using data-mining techniques to view selection is one of the most recent approaches of this algorithm class. There are different types of data-mining processes [19]. Frequent pattern mining is one of the most common mining techniques used for materialized view selection. Several centralized, parallel, and distributed algorithms have been presented that attempt to improve pattern-mining efficiency [20–23]. Recently, some materialized view selection methods have tried to use the benefits of pattern mining to speed up the view-selection process [1,24,25].

The third category of view-selection algorithms contains hybrid algorithms that combine deterministic and randomized algorithms to improve their efficiency. For example, a hybrid algorithm was proposed in [26] that combines greedy and genetic algorithm to optimize queries, choose the best processing plan, and select the best set of materialized views. This hybrid algorithm is based on MVPP.

Constraint programming is a programming technique used to solve combinatorial problems. Some materialized view-selection methods have used approaches based on constraint programming to find an optimal set of materialized views. The view-selection problem should be modeled as a constraint satisfaction problem in this class of algorithms. In [27], it was proved experimentally that this class of algorithms performs better than randomized methods.

An academic method titled 2PO was presented in [28] for selecting appropriate views from an MVPP structure and without any special limitation. In this method, the MVPP structure is optimized by finding the mutual subexpressions in the queries, and then appropriate views are selected from this optimized MVPP. The 2PO method minimizes the query-processing and view maintenance costs after the optimization of MVPP.

3. Statement of the problem and method proposal

In this section, the view selection problem is formally defined and MVPP is described. Then a new greedy method and the cost function of its view-selecting approach is proposed. Finally, the proposed method's algorithm is presented and explained with an example.

3.1. Mathematical definition of the problem

Materializing an optimal subset of views is the most common method in reducing query response times. In fact, storing a set of views and responding to the queries using these views can significantly affect the processing rate of the queries. The view selection problem can be expressed mathematically as follows:

Assume $R = \{R_1, R_2, ..., R_r\}$ to be a set of base relations of a dataset. Let $Q = \{Q_1, Q_2, ..., Q_q\}$ be a set of queries defined on R, and let $N = \{N_1, N_2, .., N_n\}$ be the set of all the other nodes of MVPP, each of which represents an intermediate view that should be constructed to respond to a query. The materialized view-selection problem aims at selecting an appropriate (possibly optimal) set of views in the form of $M = \{V_1, V_2, .., V_m\}$, (obviously, M is a subset of N) in a way that the queries are responded to with the lowest possible cost, based on resource costs like storage space and view maintenance.

3.2. An example of MVPP

In this section, an example of a small data warehouse system and the resulting MVPP is presented and this example is used to show the proposed method's details. The data warehouse in this example is a simplified version constructed from the TPC-H database (available from http://www.tpc.org/tpch/). Relations and attributes of this version are considered a follows:

Customer(C_custkey, C_name, C_address)

Order(O_orderkey, C_cusktkey, O_orderdate)

Supplier(S_suppkey, S_nationkey)

Lineitem(L_orderkey, L_suppkey, L_tax, L_quantity)

Nation(N_nationkey, N_name)

Moreover, assume that there are three queries in this system. These queries are presented in Figure 1.

Some of these queries overlap in temporary intermediate views, which are the results of relational algebra operators. In calculating query-processing and view maintenance costs, "k" stands for thousand, "m" for million, and "b" for billion.

Now assume there are 150k tuples in *Customer*, 227k tuples in *Order*, 10k tuples in *Supplier*, 6m tuples in *Lineitem*, and 25 tuples in *Nation.* Considering these relations and queries, the resulting MVPP is presented in Figure 2.

```
Ql: Select   stddev(L_tax)
    From     Customer, Orders, Lineitem
    Where    C_custkey = O_custkey
    And      O_orderkey = L_orderkey
    And      O_orderdate >= '1994-01-01'
    And      O_orderdate < '1995-01-01'

Q2: Select   S_nationkey, variance(L_quantity)
    From     Orders, Lineitem, Supplier
    Where    O_orderkey = L_orderkey
    And      L_suppkey = S_suppkey
    And      O_orderdate >= '1994-01-01'
    And      O_orderdate < '1995-01-01'

Q3: Select   N_name, avg(L_quantity)
    From     Orders, Lineitem, Supplier, Nation
    Where    O_orderkey = L_orderkey
    And      L_suppkey = S_suppkey
    And      S_nationkey = N_nationkey
    And      O_orderdate >= '1994-01-01'
    And      O_orderdate < '1995-01-01'
    Group by N_name
```

Figure 1. Queries of the given example.

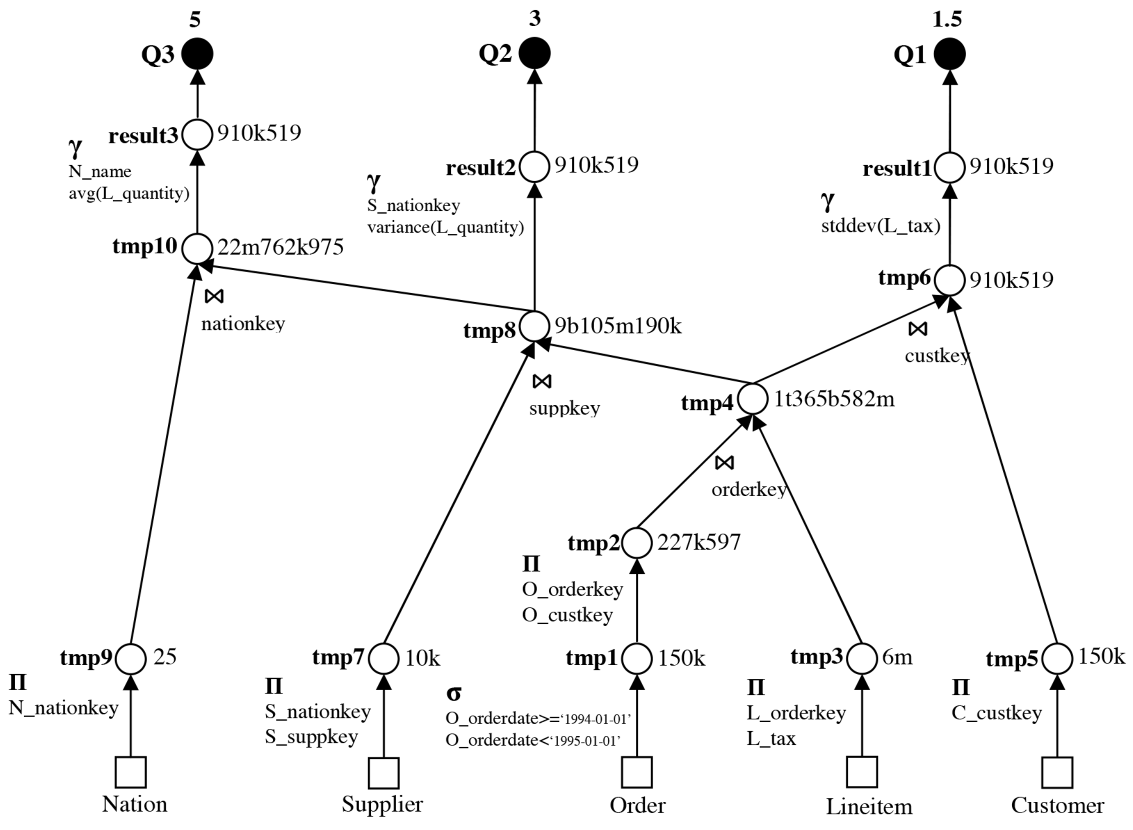


Figure 2. MVPP of the given example.

Each node shows one view. Queries are placed in roots and base relations are in leaves (Figure 2). Considering the mentioned items, each node's cost is given beside it. The frequency of each query is shown on top of it. The query-processing cost of each node equals the cost of creating that node multiplied by total frequency of the queries using that node. The view maintenance cost of each node equals the cost of creating that node multiplied by the number of relations used for producing that node. For example, consider tmp4. The query processing cost for tmp4 is equal to (5 + 3 + 1.5) $\times$ (1t365b582m + 6m + 227k597 + 150k) and view maintenance cost equals 2 $\times$ (1t365b582m + 6m + 227k597 + 150k).

As observed in Figure 2, some of the views, like tmp4, are jointly used in a number of (or even all) queries. This view can be appropriate for materialization. However, the query-processing and view maintenance costs related to them should be considered. The goal here is to find a set of optimal views with minimum query-processing and view maintenance costs.

3.3. TSGV method

In most of the methods based on MVPP, the investigation mechanism of views consists of two main MVPP-scanning phases. The first scan selects the members of the initial set of candidate views for materialization, and the second one eliminates selected views that can result from other members of this set. Therefore, each node in which all of its previous nodes (in MVPP structure) are materialized must be deleted from the materialized nodes set because all of the queries that need this node for constructing their own results can be responded to using its previous nodes. To omit these types of nodes, it is necessary to evaluate the materialized node set and the MVPP twice, which is time-consuming. The proposed TSGV method aims to obtain the optimal nodes set by only scanning the MVPP once. For this purpose, TSGV should investigate the MVPP nodes in a bottom-up manner.

Now consider the MVPP nodes in Figure 2. If the queries are in the zero level, then tmp8 and tmp10 are both placed in level 2, while tmp8 has been used in creating tmp10 and must certainly be considered before that. To solve this problem, a new measure is introduced for each node, called *Dfr*, which is defined as the maximum distance of that node from MVPP relations. Formally, if Dist(N_i , R_j) is defined as the number of intermediate nodes between N_i and R_j , then *Dfr* will be defined as $Dfr(N_i) = \max_{\forall R_j \in R}(Dist(N_i, R_j))$.

For example, *Dfr*(tmp1) = 1, *Dfr*(tmp8) = Max(2,4,3) = 4 and *Dfr*(tmp10) = Max(2,3,5,4) = 5. Using *Dfr*, the views can be arranged in ascending order for bottom-up investigation. Therefore, according to this parameter, first tmp8 and then tmp10 are studied for materialization.

3.3.1. Table-like structure (TLS)

To investigate the nodes using *Dfr*, a new structure containing information from all nodes available in MVPP is presented, the table-like structure of views (TLS). The proposed method is based on TLS. In each TLS row, there is information related to one of the MVPP's nodes. Each TLS row includes the following:

- Node: Specified as v, which is the node's unique ID.
- *Dfr*: $Dfr(v) = \max_{\forall R_j \in R}(Dist(v, R_j))$, in which *Dist(v,* R_j*)* is the number of intermediate nodes between node v and relation R_j.
- *Pre*: The set of all previous nodes of v in the MVPP.
- *Cq*: The query processing cost of the node v, calculated according to $Cq(v) = \sum_{q \epsilon Q} fq \times C_a^q(v)$, in which fq is query frequency and C_a^q is the access cost of query q to the node v [3].

- Cm: The view maintenance cost of the node v, calculated according to $Cm(v) = \sum_{r \epsilon R} fu \times C_m^r(v)$, in which fu is the update frequency and C_m^r is the maintenance cost of node v based on change of base relation r [3].

TLS can be obtained easily by one scan of the MVPP. After obtaining the TLS, its rows are ordered in ascending order using *Dfr*, so that the nodes closer to the relations are examined earlier. The TLS resulting from the example MVPP is presented in Figure 3.

Node	Dfr	Pre	Cq	Cm
tmp1	1	Ø	1m425k	150k
tmp3	1	Ø	57m	6m
tmp5	1	Ø	225k	150k
tmp7	1	Ø	80k	10k
Tmp9	1	Ø	125	25
tmp2	2	tmp1	3m587k171.5	377k597
tmp4	3	tmp2, tmp3	12t973b89m587k171.5	2t731b176m755k149
tmp6	4	tmp4, tmp5	2t48b384m157k174	4t96b768m143k348
tmp8	4	tmp4, tmp7	10t997b548m620k776	4t124b80m732k791
tmp10	5	tmp8, tmp9	6t873b581m720k985	5t498b865m362k388
result1	5	tmp6	2t48b385m522k952.5	4t96b771m45k905
result2	5	tmp8	4t124b83m464k348	4t124b83m464k348
result3	6	tmp10	6t873b586m255k580	5t498b869m4k464

Figure 3. TLS of the given example.

Since the nodes have been ordered based on *Dfr* in the TLS, nodes closer to base relations examined sooner, and thus there is no redundant node in the TLS and there is no need to rescan the TLS to eliminate redundant selected nodes.

3.3.2. Cost function

After determining the investigation order of the nodes, the appropriate nodes for materialization must be specified. In order to determine appropriate nodes, profitability of the nodes must be obtained. Different relations have been presented in order to study the profitability of the nodes or views in different methods. These relations usually consider the query-processing cost, the view maintenance cost, or both. The query-processing cost of one node, from another point of view, is in fact the profit gained after materializing the node, because after materialization, there is no need for that node or view to be created from the basic relations. Furthermore, from another point of view, the view maintenance cost equals the cost we should afford for a materialized node after updating the base relations.

In the given example, we pay attention to both costs. In addition, in studying each node, the query-processing total costs of previously materialized nodes (the *Pre* set) are considered, because if a node is materialized while a number of its previous nodes are also materialized, some of the resources will be lost. This cost is calculated through $Cqmp\,(v) = \sum_{u \epsilon Pre(v) \bigcap M} C_a^q(u)$, in which M is the total materialized nodes. The cost function of the proposed method is in form of Eq. (1).

$$B(v) = \alpha \times Cq(v) + \beta \times Cqmp(v) + \gamma \times Cm(v), \tag{1}$$

where α, β, and γ are the adjustment factors of the mentioned costs. Considering these coefficients, the proposed cost function is flexible. For example, if updates of the relations are not important and we want to ignore view maintenance costs, then we can give γ a value of zero. However, if we want to consider this cost and subtract it from the profit obtained from materialization, we should give γ a value of –1. Other coefficients can be given values in the same way.

Now we should evaluate the nodes in the TLS individually using this cost function. While evaluating a node, if its profitability is higher than zero, it is added to the set of materialized nodes and all of its *Pre* nodes are omitted from M if materialized, but if the profitability of a node is zero or less than zero, it will be ignored. After studying all nodes in the view table, the set of optimal nodes is found.

3.3.3. TSGV algorithm

After constructing the TLS, the *Dfr* of each TLS node should be calculated. Figure 4 shows the pseudocode of the recursive algorithm for calculating *Dfr* from TLS.

```
Procedure Dfr
Input
  Node
Begin
  If  Pre(Node) = Ø Then
    Return 1;
  Else
    Return Max(Dfr(each node in Pre(Node))+1)
End;
```

Figure 4. The pseudocode of the recursive algorithm calculating *Dfr*.

After calculating the *Dfr* of each TLS node, TSGV calculates the cost function for each node of this structure. If the utility of a node is equal to or less than zero, TSGV ignores it, and in other cases TSGV inserts the node to the materialized views list and eliminates its *Pre* nodes from this list. The pseudocode of the TSGV algorithm is shown in Figure 5.

The inputs of the algorithm are the MVPP and the adjusting parameters. It first constructs the TLS from the MVPP. Then this structure is ordered in ascending order based on *Dfr*. Later, the candidate views for materialization will be selected by TSGV and inserted in M. For this purpose, the *Cqmp* cost of each node of the ordered TLS is calculated first. Then the utility of the node is calculated according to Eq. (1). If this utility is higher than zero, it is added to M, and any of its *Pre* nodes that exist in M will be removed.

The results of running TSGV on the example MVPP given in Figure 2 with values of $\alpha = 1$, $\beta = –1$, and $\gamma = –1$ are as follows:

$M = \{\ \}$

$B(\text{tmp1}) = [(1) \times \text{1m425k}] + [(–1) \times 0] + [(–1) \times \text{150k}] > 0$
$Pre(\text{tmp1}) \bigcap M = \emptyset$
$M = \{\text{tmp1}\}$

$B(\text{tmp3}) = [(1) \times \text{57m}] + [(–1) \times 0] + [(–1) \times \text{6m}] > 0$
$Pre(\text{tmp3}) \bigcap \text{M} = \emptyset$
$M = \{\text{tmp1, tmp3}\}$

$B(\text{tmp5}) = [(1) \times \text{225k}] + [(–1) \times 0] + [(–1) \times \text{150k}] > 0$

```
Procedure TSGV
Input
MVPP, α, β, γ
Begin
   Create Table-like Structure from MVPP;
   For each Node x in Table-like Structure
       Calculate Dfr(x)
   Sort Table-like Structure by ascending value of Dfr;
   M = {};
   For each Node v in Table-like Structure
   Begin
      Cqmp(v) = 0;
      For each Node u in Pre(v)
         If u is a member of M Then
            Cqmp(v) += Cq(u);
      B(v) = α * Cq(v) + β * Cqmp(v) + γ * Cm(v);
      If B(v) > 0 Then
      Begin
         insert v into M;
         For each Node w in Pre(v)
            If w is a member of M Then
               remove w from M;
      End;
   End;
End;
```

Figure 5. The pseudocode of the TSGV algorithm.

$Pre(\text{tmp5}) \bigcap M = \emptyset$

$M = \{\text{tmp1, tmp3, tmp5}\}$

$B(\text{tmp7}) = [(1) \times 80\text{k}] + [(-1) \times 0] + [(-1) \times 10\text{k}] > 0$

$Pre(\text{tmp7}) \bigcap M = \emptyset$

$M = \{\text{tmp1, tmp3, tmp5, tmp7}\}$

$B(\text{tmp9}) = [(1) \times 125] + [(-1) \times 0] + [(-1) \times 25] > 0$

$Pre(\text{tmp9}) \bigcap M = \emptyset$

$M = \{\text{tmp1, tmp3, tmp5, tmp7, tmp9}\}$

$B(\text{tmp2}) = [(1) \times 3\text{m}587\text{k}171.5] + [(-1) \times 1\text{m}425\text{k}] + [(-1) \times 377\text{k}597] > 0$

$Pre(\text{tmp2}) \bigcap M = \{\text{tmp1}\}$

$M = \{\text{tmp2, tmp3, tmp5, tmp7, tmp9}\}$

$B(\text{tmp4}) = [(1) \times 12\text{t}973\text{b}89\text{m}587\text{k}171.5] + [(-1) \times (3\text{m}587\text{k}171.5 + 57\text{m})] +$
$[(-1) \times 2\text{t}731\text{b}176\text{m}755\text{k}149] > 0$

$Pre(\text{tmp4}) \bigcap M = \{\text{tmp2, tmp3}\}$

$M = \{\text{tmp4, tmp5, tmp7, tmp9}\}$

$B(\text{tmp6}) = [(1) \times 2\text{t}48\text{b}384\text{m}157\text{k}174] + [(-1) \times (12\text{t}973\text{b}89\text{m}587\text{k}171.5 + 22\text{k})] +$
$[(-1) \times 4\text{t}96\text{b}768\text{m}143\text{k}348] < 0$

B(tmp8) = [(1) $\times$ 10t997b548m620k776] + [(–1) $\times$ (12t973b89m587k171.5 +80k)] +
[(–1) $\times$ 4t124b80m732k791] $<$ 0

B(tmp10) = [(1) $\times$ 6t873b581m720k985] + [(–1) $\times$ 125] + [(–1) $\times$ 5t498b865m362k388] $>$ 0

Pre(tmp10) $\bigcap M$ = {tmp9}
M = {tmp4, tmp5, tmp7, tmp10}

B(result1) = [(1) $\times$ 2t48b385m522k952.5] + [(–1) $\times$ 0] + [(–1) $\times$ 4t96b771m45k905] $<$ 0

B(result2) = [(1) $\times$ 4t124b83m464k348] + [(–1) $\times$ 0] + [(–1) $\times$ 4t124b83m464k348] = 0

B(result3) = [(1) $\times$ 6t873b586m255k580] + [(–1) $\times$ 6t873b581m720k985] +

[(–1) $\times$ 5t498b869m4k464] $<$ 0

TSGV selected M = {tmp4, tmp5, tmp7, tmp10} as the optimal views set for materialization.

4. Experimental results

The TPC-H database was used to evaluate TSGV (http://www.tpc.org/tpch/). This database includes 8 relations of region, nation, supplier, customer, orders, line item, part, and partsupp (http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf). In addition, 60 queries were used to evaluate the proposed method. No limitation has been considered in our evaluations. The evaluation was conducted using an Intel 2410M 2.3GHz Core i5 2.9 GHz processor, 4 GB of DDR3 main memory, and a 600 GB SATA hard disk.

In order to study TSGV, its functionality was compared with 2PO [28], which is a recently presented, MVPP-based, dynamic view selection method. 2PO optimizes the MVPP structure by finding the common subexpressions in queries and then selecting the views from the optimized MVPP. For evaluation, both methods were first tested with 60 queries and the query-processing and view maintenance costs and the total resulting cost were calculated. The adjustment coefficients were set as $\alpha = 1$, $\beta = -1$, and $\gamma = -1$ for TSGV. The costs resulting from conducting both methods are presented in the Table.

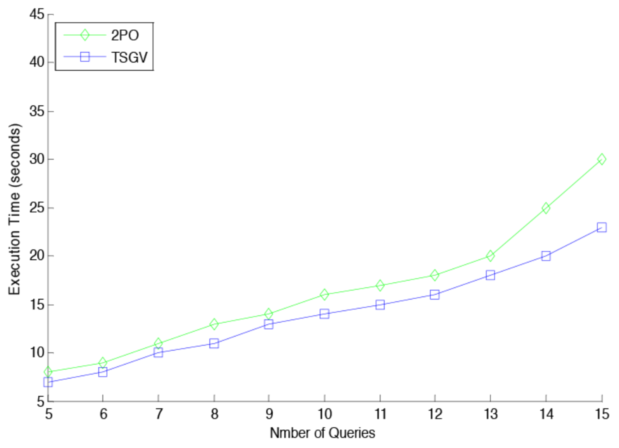
Table. Cost comparison of TSGV and 2PO.

Total cost	View maintenance cost	(a) Query-processing cost	Number of materialized views	Method
9,852,234,291,325	6,329,383,146,254	3,522,851,145,071	37	2PO
9,852,241,256,761	6,329,368,000,604	3,522,873,256,157	42	TSGV

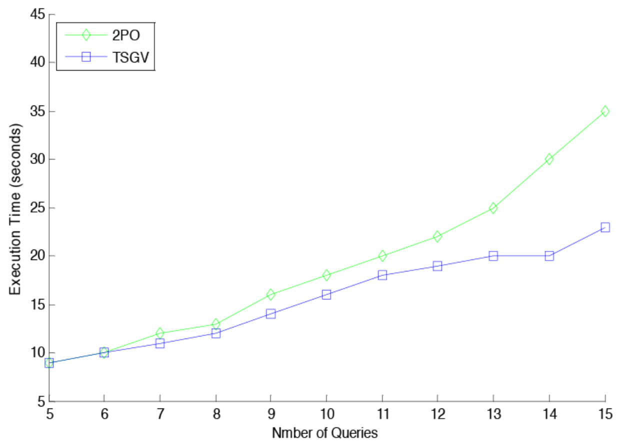
The total cost of the 2PO method is slightly less than total cost of the TSGV. This is because the MVPP structure is optimized in 2PO and, as a result, the total number of views is reduced. Therefore, it is possible that the number of views selected for materialization in 2PO was lower than in TSGV. Therefore, there is the possibility of cost reduction in the 2PO method. However, the total cost of the TSGV does not differ significantly from the total cost of 2PO method, and it performs better than other deterministic methods in this field.

We also compared TSGV and 2PO methods in terms of time. First, the queries were divided into four categories: A, B, C, and D. The queries of each category were more complicated than the previous one, such that category A contained the simplest queries and category D contained the most complex.

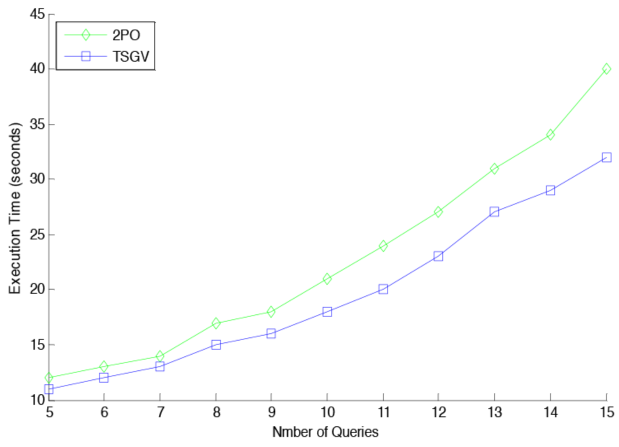
Calculating the selection time of the views began with 5 queries for workload and the performance times of both methods were calculated. Then one query was added to the workload in each stage and performance times were calculated for both methods. This continued until all 15 queries of each category were eventually added to the workload, and the performance time of each method was calculated for each category of queries. View selection in each stage was separate and static, and adding the queries does not mean that view selection process is dynamic. The results of these evaluations are presented in Figure 6.



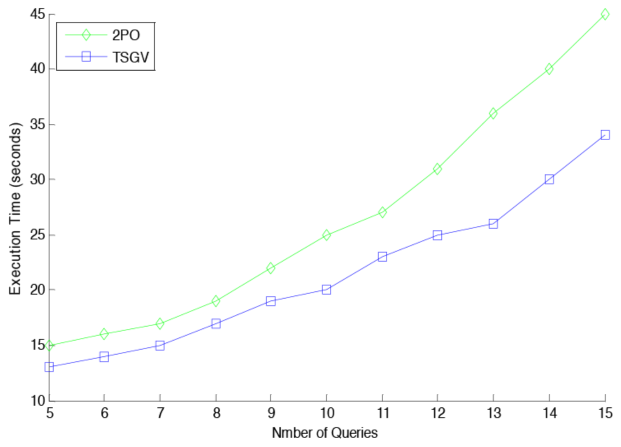
(a) Queries of category A



(b) Queries of category B



(c) Queries of category C



(d) Queries of category D

Figure 6. TSGV and 2PO temporal comparison diagram.

TSGV performed better than 2PO in all evaluations. This is because in TSGV, the MVPP is directly used for selecting the view, and necessary information is extracted and placed in the view table. However, in 2PO, the MVPP is created first. Then the common subexpression between queries is identified and the optimized MVPP is rebuilt. Therefore, the MVPP is built twice, which is time-consuming.

5. Conclusion

Selection of views for materialization is an important issue in data warehouses. Materialized views help the data warehouse to increase the response rate of its online analytical queries. The purpose of the materialized view selection problem is to select the optimal subset of views that receive a high number of requests so that queries can be more quickly answered using them. A number of methods have been introduced so far for solving the view-selection problem. In this paper, a deterministic method based on a novel TLS called TSGV is presented for solving the view-selection problem. Information in this TLS was extracted from the MVPP. The proposed cost function of this paper considered query-processing costs and view maintenance costs and the query-processing cost of the previous materialized nodes. The practical evaluation showed that TSGV performs better than similar methods in terms of time costs. The proposed TLS of this paper can extract its information from other view structures, such as AND-OR DAGs or cube lattices, rather than an MVPP as a future work. This TLS may also be combined with randomized algorithms or hybrid algorithms for other future works.

References

[1] Sohrabi MK, Ghods V. Materialized view selection for data warehouse using frequent itemset mining. Journal of Computers 2016; 11: 140-148.

[2] Yu D, Dou W, Zhu Z, Wang J. Materialized view selection based on adaptive genetic algorithm and its implementation with Apache hive. IETE Tech Rev 2015; 8: 1091-1102.

[3] Yang J, Karlapalem K, Li Q. Algorithms for materialized view design in data warehousing environment. VLDB 1997; 97: 25-29.

[4] Roy P, Seshadri S, Sudarshan S, Bhobe S. Efficient and extensible algorithms for multi query optimization. ACM SIGMOD Record 2000; 29: 249-260.

[5] Harinarayan V, Rajaraman A, Ullman JD. Implementing data cubes efficiently. ACM SIGMOD Record 1996; 25: 205-216.

[6] Roussopoulos N. View indexing in relational databases. ACM Trans Database Syst 1982; 7: 258-290.

[7] Mami I, Bellahsene Z. A survey of view selection methods. ACM SIGMOD Record 2012; 41: 20-29.

[8] Yang J, Karlapalem K, Li Q. A framework for designing materialized views in data warehousing environment. ICDCS 1997.

[9] Mistry H, Roy P, Sudarshan S, Ramamritham K. Materialized view selection and maintenance using multi-query optimization. ACM SIGMOD Record 2001; 30: 307-318.

[10] Bauer A, Lehner W. On solving the view selection problem in distributed data warehouse architectures. SSDBM 2003; 43-51.

[11] Ye W, Gu N, Yang G, Liu Z. Extended derivation cube based view materialization selection in distributed data warehouse. WAIM 2005; 245-256.

[12] Theodoratos D, Ligoudistianos S, Sellis T. View selection for designing the global data warehouse. Data Knowl Eng 2001; 39: 219-240.

[13] Agrawal S, Chaudhuri S, Narasayya VR. Automated selection of materialized views and indexes in SQL databases. VLDB 2000; 496-505.

[14] Horng JT, Chang YJ, Liu BJ. Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft Computing 2003; 7: 574-581.

[15] Derakhshan R, Dehne FKHA, Korn O, Stantic B. Simulated annealing for materialized view selection in data warehousing environment. Databases and Applications 2006; 89-94.

[16] Derakhshan R, Stantic B, Korn O, Dehne FKHA. Parallel simulated annealing for materialized view selection in data warehousing environments. ICA3PP 2008; 121-132.

[17] Kalnis P, Mamoulis N, Papadias D. View selection using randomized search. Data Knowl Eng 2002; 42: 89-111.

[18] Chaves LWF, Buchmann E, Hueske F, Bohm K. Towards materialized view selection for distributed databases. ACM EDBT 2009; 1088-1099.

[19] Sohrabi MK, Akbari S. A comprehensive study on the effects of using data mining techniques to predict tie strength. Comput Hum Behav 2016; 60: 534-541.

[20] Sohrabi MK, Barforoush AA. Efficient colossal pattern mining in high dimensional datasets. Knowl-based Syst 2012; 33: 41-52.

[21] Sohrabi MK, Barforoush AA. Parallel frequent itemset mining using systolic arrays. Knowl-based Syst 2013; 37: 462-471.

[22] Sohrabi MK, Roshani R. Frequent pattern mining using cellular learning automata. Comput Hum Behav 2017; 68: 244-253.

[23] Sohrabi MK, Ghods V. Top-down vertical itemset mining. ICGIP 2014.

[24] Aouiche K, Darmont J. Data mining-based materialized view and index selection in data warehouses. JIIS 2009; 33:1: 65-93.

[25] Nalini T, Kumaravel A, Rangarajan K. An efficient I-MINE algorithm for materialized views in a data warehouse environment. Computer Science Issues 2011; 8: 366-375.

[26] Zhang C, Yao X, Yang J. An evolutionary approach to materialized views selection in a data warehouse environment. IEEE T Syst Man Cybern C 2001; 31: 282-294.

[27] Mami I, Coletta R, Bellahsene Z. Modelling view selection as a constraint satisfaction problem. DEXA 2011; 396-410.

[28] Suchyukorn B, Auepanwiriyakul R. Dynamic materialized view selection using 2PO based on re-optimized multiple view processing plan. International Journal of Advancements in Computing Technology 2013; 5: 150-167.