International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:1, No:3, 2007

Selecting Materialized Views Using Two-Phase Optimization with Multiple View Processing Plan

Jiratta Phuboon-ob, and Raweewan Auepanwiriyakul

***Abstract*—**A data warehouse (DW) is a system which has value and role for decision-making by querying. Queries to DW are critical regarding to their complexity and length. They often access millions of tuples, and involve joins between relations and aggregations. Materialized views are able to provide the better performance for DW queries. However, these views have maintenance cost, so materialization of all views is not possible. An important challenge of DW environment is materialized view selection because we have to realize the trade-off between performance and view maintenance cost. Therefore, in this paper, we introduce a new approach aimed at solve this challenge based on Two-Phase Optimization (2PO), which is a combination of Simulated Annealing (SA) and Iterative Improvement (II), with the use of Multiple View Processing Plan (MVPP). Our experiments show that our method provides a further improvement in term of query processing cost and view maintenance cost.

***Keywords*—**Data warehouse, materialized views, view selection problem, two-phase optimization.

I. Introduction

A data warehouse (DW) can be defined as subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decision [1]. It can bring together selected data from multiple database or other information sources into a single repository [2]. To avoid accessing base table and increase the speed of queries posed to a DW, we can use some intermediate results from the query processing stored in the DW called materialized views. Although materialized views speed up query processing, they have to be refreshed when changes occur to the base tables. Therefore, materialized view selection involved query processing cost and materialized view maintenance cost. So, many literatures try to make the sum of that cost minimal. For all of operation, i.e., select, project, join, order, group-by and aggregation operation; join operation has the most impact on query processing cost. In addition, some researchers consider only join order optimization or aggregation operation, or both.

Manuscript received February 21, 2007.

Jiratta Phuboon-ob, Ph.D. student is with School of Applied Statistics, National Institute of Development Administration (NIDA), Bangkok, Thailand (e-mail: jiratta.p@ msu.ac.th).

Raweewan Auepanwiriyakul is an Assistant Professor with School of Applied Statistics, National Institute of Development Administration (NIDA), Bangkok, Thailand (e-mail: raweewan@as.nida.ac.th).

The existing algorithms solving query optimization, multiple query optimizations, and materialized view selection can be classified into four categories, i.e., deterministic algorithm, randomized algorithm, evolutionary algorithm and hybrid algorithm [3].

Our previous work, in [4], we analyzed and compared only three types of algorithm; deterministic algorithm, evolutionary algorithm and hybrid algorithm. In [5], we proposed Two-Phase Optimization (2PO) algorithm, which is a combination of Simulated Annealing (SA) and Iterative Improvement (II), to the materialized view selection problem with Multiple View Processing Plan (MVPP) techniques compared to [6] and [7]. However in this experimental study, we show that, comparing to [6] – [9] our method achieve substantial improvements in term of query processing cost and view maintenance cost.

The rest of the paper is organized as follows. Section 2, we describe Multiple View Processing Plan (MVPP). Section 3, we focus on Iterative Improvement and Simulated Annealing. Section 4, we propose our Two-Phase Optimization approach which aimed at solve the materialized view selection problem. Section 5, deals with our experimental studies, and is concluded in section 6.

II. Multiple View Processing Plan (MVPP)

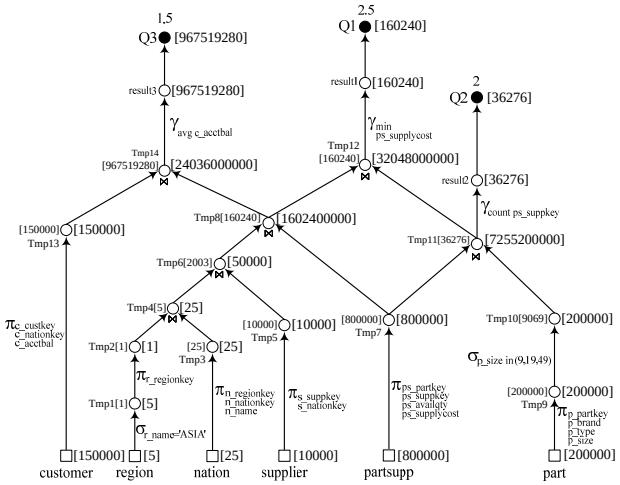

Fig. 1 An example MVPP plan

According to [6], they use multiple query processing technique (MQP) to build multiple views processing plan (MVPP) in order to identify views to be materialized.

An MVPP is a directed acyclic graph that represents a query processing of DW views. An example MVPP is shown in Fig. 1. The leaf nodes correspond to the base relations, and the root nodes represent the queries. Any vertex which is an intermediate or a final result of a query is denoted as a view. The cost for each operation node is labeled at the right hand side of each node. The query access frequencies are labeled on the top of each query node

III. Iterative Improvement and Simulated Annealing

A. Iterative Improvement (II)

[10] exposed II algorithm to the large join query optimization problem. II is a simple hill-climbing algorithm. This algorithm performs a large number of local optimizations. A local optimization starts at a random state, and seeks minimum cost point using a strategy-like hill-climbing. At the starting point, a random neighbor is selected. If the neighbor's cost is lower than current's cost, the move is carried out and a new neighbor with the lower cost is sought. II performs random series of move and accepts only downhill ones until it reaches a local minimum. This algorithm is repeated until a time limit is exceeded or a predetermined number of starting points is processed, then the lowest local minimum encountered is the result. The II algorithm present in Fig. 2.

1. random starting point
2. at the starting point, a random neighbor is selected
3. if the neighbor's cost is lower than current's cost then move is carried out and a neighbor is sought
4. performs random series of move and accepts only downhill ones until it reaches a local minimum

Fig. 2 Iterative Improvement (II) algorithm

B. Simulated Annealing (SA)

[11] invented SA algorithm, and used it on traveling sale man problem. SA follows a procedure similar to II, but it accepts uphill move with some probability, while II performs only downhill move. At each step, the SA considers neighbor's cost and the current's cost, and probabilistically decides among moving the system to neighbor's state or staying in current's state. The probabilities are chosen, so the system ultimately tends to move to states of the lower cost. This step is repeated until the time becomes zero, or until the system reaches a state which is good enough for the application. [12] applied this algorithm to the optimization of some recursive queries.

In [7], they introduced a new approach for materialized view selection based on SA in conjunction with the use of a MVPP. Fig. 3 show SA algorithm.

1. random starting point
2. at the starting point, a random neighbor is selected
3. compare neighbor's cost and current's cost
   3.1 if the neighbor's cost is lower than current's cost then move is carried out and a neighbor is sought
   3.2 otherwise moving to neighbor's state or staying in current's state with probability
4. performs random series of move and accepts both of downhill and uphill until it reaches a local minimum

Fig. 3 Simulated Annealing (SA) algorithm

IV. Our approach: two-phase optimization for selecting materialized view

Ioannidis and Kang [13] inspired Two-Phase Optimization (2PO) algorithm to the optimization of project-select-join queries. Our approach is designed based on 2PO with MVPP for solving the materialized view selection problem. 2PO combines both SA and II. It begins by running II to find a good local minimum, and then applies SA to search for the global minimum from the state found by II. Our algorithm present in Fig. 4.

1. Input a MVPP represented by a DAG
2. Use width-first searching method to search through all of the nodes in the DAG and produce an ordered sequence of these nodes into a binary string
3. Call Iterative Improvement algorithm
4. Call Simulated Annealing algorithm
5. Present set of views to materialized with minimum cost

Fig. 4 Our Two-Phase Optimization (2PO) algorithm

In the following subsection, we give the details of representation of solutions and define the cost model of materialized view selection.

A. Representation of Solutions

Output from MVPP is a DAG. We map a DAG into a binary string. For example, searching through the DAG, shown in Fig. 1, using width-first, we obtain the mapping array, i.e. [result3,0], [result1,0], [result2,0], [tmp14,0], [tmp12,0], [tmp11,0], [tmp13,0], [tmp8,0], [tmp10,0], [tmp6,0], [tmp7,0], [tmp9,0], [tmp4,0], [tmp5,0], [tmp2,0], [tmp3,0], [tmp1,0]. A binary string of {0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0} indicates that none of node is materialized.

B. Cost Model of Materialized View Selection

According to [5], a linear cost model is used to calculate the cost of query Q. The cost of answering Q is the number of rows in the table that query Q_i used to construct Q.

Let M be a set of materialized views, $C_{q_i}(M)$ be the cost to compute q_i from the set of materialized views M, $C_m(v)$ be the cost of maintenance when v is materialized, and f_q, f_u are query and updating frequency, respectively.

Then the total query processing cost is:

$$\sum_{q_i \in Q} f_{q_i} C_{q_i}(M)$$

The total maintenance cost is:

$$\sum_{v \in M} f_u C_m(v)$$

The total cost of the materialized views *M* is:

$$\sum_{q_i \in Q} f_{q_i} C_{q_i}(M) + \sum_{v \in M} f_u C_m(v)$$

Our goal is to find the set *M*, if the members of *M* are materialized then the value of total cost will be minimal among all feasible sets of materialized view.

V. Experimental Studies

In our experiment, we employ MQP technique to all of five algorithms and use the same cost model proposed by [6] to compute query processing cost, materialized view maintenance cost and total cost. We do not consider any constraints. We use the TPC-H database of size 1GB as a running example throughout our paper. It has 22 read-only queries. Most of them are large and complex, and perform different operations on the database table. For more details on this benchmark refers to [14]. In this paper, we cover all of regular aggregate functions; MIN, MAX, SUM, AVG, COUNT, VARIANCE and STDDEV as listed in Fig. 5.

Query 1 (MIN)
```
select  n_name, min(ps_supplycost)
from    part, partsupp, supplier, nation, region
where  p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA '
group by n_name;
```

Query 2 (MAX)
```
select  n_name, max(o_totalprice)
from    customer, orders, lineitem, nation, region
where c_custkey = o_custkey
        and o_orderkey = l_orderkey
        and c_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA'
        and o_orderdate >= '1994-01-01'
        and o_orderdate < '1995-01-01'
group by n_name;
```

Query 3 (SUM)
```
select  n_name, sum(l_quantity)
from    orders, lineitem, supplier, nation, region
where l_orderkey = o_orderkey
        and l_suppkey = s_suppkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA'
        and o_orderdate >= '1994-01-01'
        and o_orderdate < '1995-01-01'
group by n_name;
```

Query 4 (AVG)
```
select n_name, avg(c_acctbal)
from   partsupp, supplier, customer, nation, region
where ps_suppkey = s_suppkey
        and c_nationkey = s_nationkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'ASIA'
group by n_name;
```

Query 5 (COUNT)
```
select  p_size, count(ps_suppkey)
from    partsupp, part
where p_partkey = ps_partkey
        and p_brand <> 'Brand#45'
        and not p_type like '%BRASS%'
        and p_size in (9, 19, 49)
group by p_size;
```

Query 6 (VARIANCE)
```
select p_size, variance(ps_supplycost)
from   supplier, partsupp, part
where s_suppkey = ps_suppkey
        and p_partkey = ps_partkey
        and p_brand <> 'Brand#45'
        and not p_type like '%BRASS%'
        and p_size in (9, 19, 49)
group by p_size;
```

Query 7 (STDDEV)
```
select  o_orderpriority, 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'
group by o_orderpriority;
```

Fig. 5 Our Queries

We observe that these queries are defined over overlapping portion of the base data or intermediate query result. For example Query 5 and Query 6 can share the intermediate result of joining part and partsupp. We assume that base table is updated once a time and the frequencies of Query 1 to Query 7 are 4, 6, 7, 2, 5, 9, and 3 respectively.

We use the algorithm to generate MVPP proposed by [6]. Fig. 6 gives accessing plan for each of the above queries, denoted as op1, op2, op3, op4, op5, op6, op7 respectively. As a consequence, first of all, we push up all select, project, and group-by operation. Second, we create a list of query and order them based on the result of query access frequency multiplied by query processing cost. Therefore the initial list is {op4, op7, op3, op2, op1, op6, op5}.Third, we pick up op4, and merge the rest of the queries with it in the order of that in the list. Then we get the first MVPP. Fourth, the first element of list is moved to the end of the list, so the list becomes {op7, op3, op2, op1, op6, op5, op4}. We generate the second MVPP by using the third step. We repeat the third and fourth step to generate all 7 MVPPs. Next, we push down select, project, and group-by operations respectively for all of MVPPs. Finally, we select the cheapest one; shown in Fig. 7. In our MVPP, we assume that methods for implementing select and join operation are linear search and nested loop approach. Before comparing the cost, we compute query processing cost, materialized view maintenance cost and total cost of all-virtual-views and all-materialized-view, demonstrated in Table I. Table II gives the selected views and their cost from each algorithm. We compare these costs between five algorithms as following:

In deterministic algorithm, given a MVPP, we execute the view selection algorithm proposed by [6] to select materialized view. The view selected are Tmp11, Tmp15, Tmp17, Tmp21 and Tmp24. Based on these results, it would be benefit to materialize them, reducing the cost from 7,688,720,739,017 to 6,184,919,079,222.

According to simulated annealing algorithm, we use an existing simulated annealing package [15] and define this problem based on [7]. We first search through the DAG, shown in Fig. 7, using width-first, in order to map MVPP into a binary string of 1s and 0s. We set SA parameters like [7] the result of SA is {0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,1,0,0, 0,0,1,1,0,0,1,0,0,0,0,0,0} means that nodes named Tmp5, Tmp11, Tmp15, Tmp17, Tmp18, Tmp21 and Tmp24 are materialized, but the others are not. Based on these results; it would be benefit to materialize them, reducing the cost from 7,688,720,739,017 to 6,184,918,609,222.

For evolutionary algorithm, we follow the genetic algorithm (GA) proposed by [8] to solve this problem. We first map a DAG into a binary string using the same method as used by SA. We adopt the concept of ranking selection as selection operator, and set GA parameters according to [8]. The selected views are Tmp11, Tmp15, Tmp17, Tmp18, Tmp21 and Tmp24. Based on these results; it would be benefit to materialize them, reducing the cost from 7,688,720,739,017 to 6,184,918,679,222.

As Hybrid algorithm, we use the view selection algorithm presented by [9] to select materialized view. We divide this method into two phases. In the first phase, we run GA regarding the third experiment. The output of this phase is the input used in the next phase. For the second phase, we run

deterministic algorithm, similar to the first experiment. The selected views are Tmp11, Tmp15, Tmp17, Tmp21 and Tmp24, which are the same views processed in the first experiment. So the total cost of these results is equal to the total cost of deterministic algorithm.

For our two-phase optimization algorithm, we map a DAG into a binary string using the same method as used by SA. Then we run II and then followed by SA. The selected views are Tmp5, Tmp11, Tmp12, Tmp15, Tmp17, Tmp18, Tmp21 and Tmp24. Based on these results; it would be benefit to materialize them, reducing the cost from 7,688,720,739,017 to 6,184,918,159,222.

Table II compares our 2PO algorithm result to the deterministic algorithm, SA algorithm, GA and hybrid algorithm for materialized view selection. This table shows that our 2PO algorithm approach provides the best result. Although our maintenance cost is the most expensive, however, our query processing cost is the cheapest one. So our total cost is minimal. Consider the result of deterministic algorithm and hybrid algorithm, theirs maintenance cost are the cheapest, however, theirs query processing cost are the most expensive, leading to theirs total cost the most expensive consequently. For GA and SA algorithm, their query processing cost and maintenance cost are moderate, so theirs total cost are moderate too.

VI. Conclusion

In this paper, we introduce Two-Phase Optimization (2PO) algorithm, which is a combination of Simulated Annealing (SA) and Iterative Improvement (II), to materialized view selection with Multiple View Processing Plan (MVPP) proposed by [6]. Comparing to deterministic algorithm exposed by [6], Simulated Annealing proposed by [7], Genetic Algorithm introduced by [8], and hybrid algorithm invented by [9], our approach provides a better result than the other algorithm. Two-Phase Optimization finds the best solution, and avoids unnecessary large uphill moves at the early stages of Simulated Annealing.

Acknowledgment

We are grateful to Wanida Prapavasit, Roozbeh Derakhshan and Dyke Stiles for helping by way of discussion and provide material for this paper.

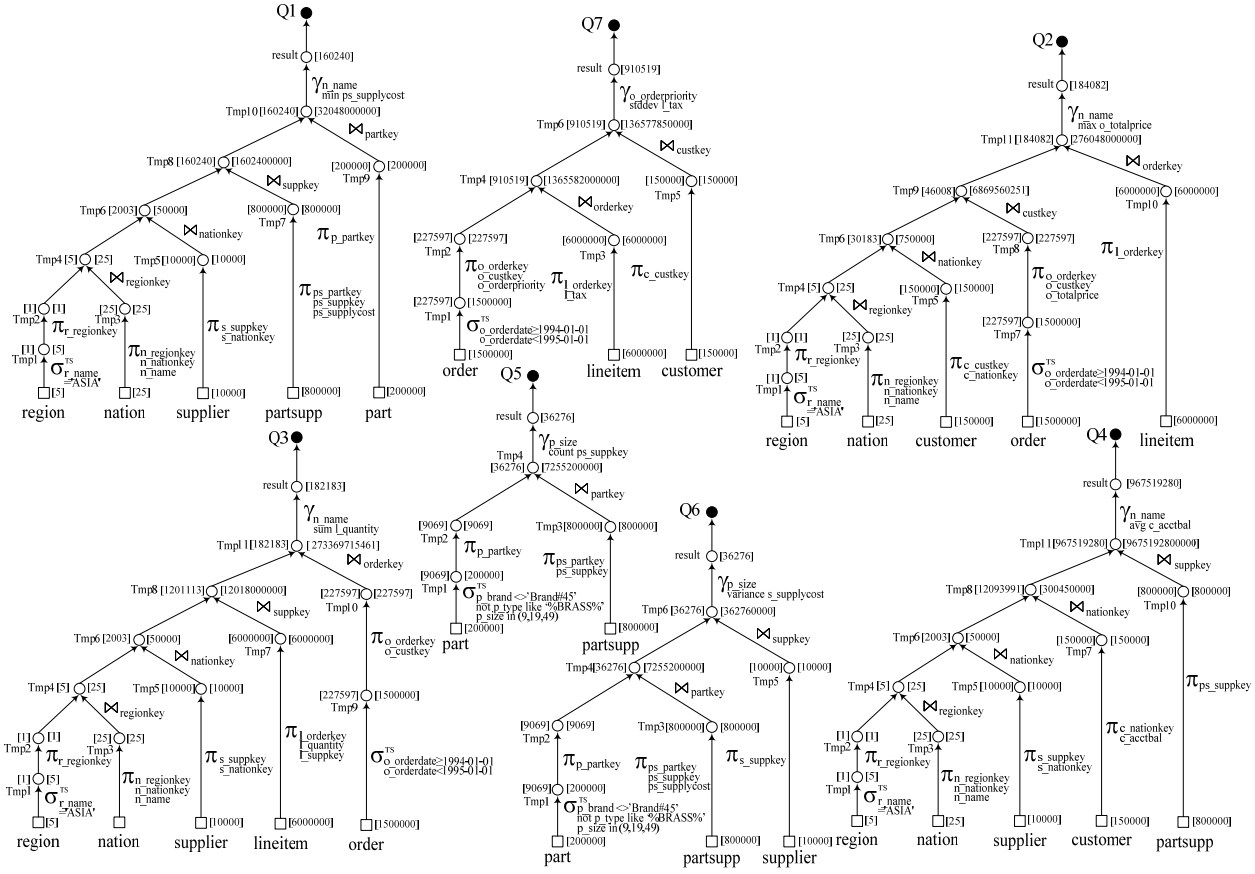

Fig. 6 Individual Optimal Query Processing Plan

TABLE I
THE QUERY PROCESSING, MAINTENANCE AND TOTAL COST

	Cost of query processing	Cost of maintenance	Total Cost
All-virtual-views	8,494,509,321,063	-	8,494,509,321,063
All-materialized-views	1,941,298,714	7,686,779,440,303	7,688,720,739,017

TABLE II
THE QUERY PROCESSING, MAINTENANCE AND TOTAL COST FOR EACH ALGORITHM

Algorithm	Selected views	Cost of query processing	Cost of maintenance	Total Cost
Deterministic	Tmp11, Tmp15, Tmp17, Tmp21, Tmp24	591,205,328,714	5,593,713,750,508	6,184,919,079,222
Simulated Annealing	Tmp5, Tmp11, Tmp15, Tmp17, Tmp18, Tmp21, Tmp24	591,204,438,714	5,593,714,170,508	6,184,918,609,222
Genetic Algorithm	Tmp11, Tmp15, Tmp17, Tmp18, Tmp21, Tmp24	591,204,528,714	5,593,714,150,508	6,184,918,679,222
Hybrid Algorithm	Tmp11, Tmp15, Tmp17, Tmp21, Tmp24	591,205,328,714	5,593,713,750,508	6,184,919,079,222
Two-Phase Optimization	Tmp5, Tmp11, Tmp12, Tmp15, Tmp17, Tmp18, Tmp21, Tmp24	591,203,688,714	5,593,714,470,508	6,184,918,159,222

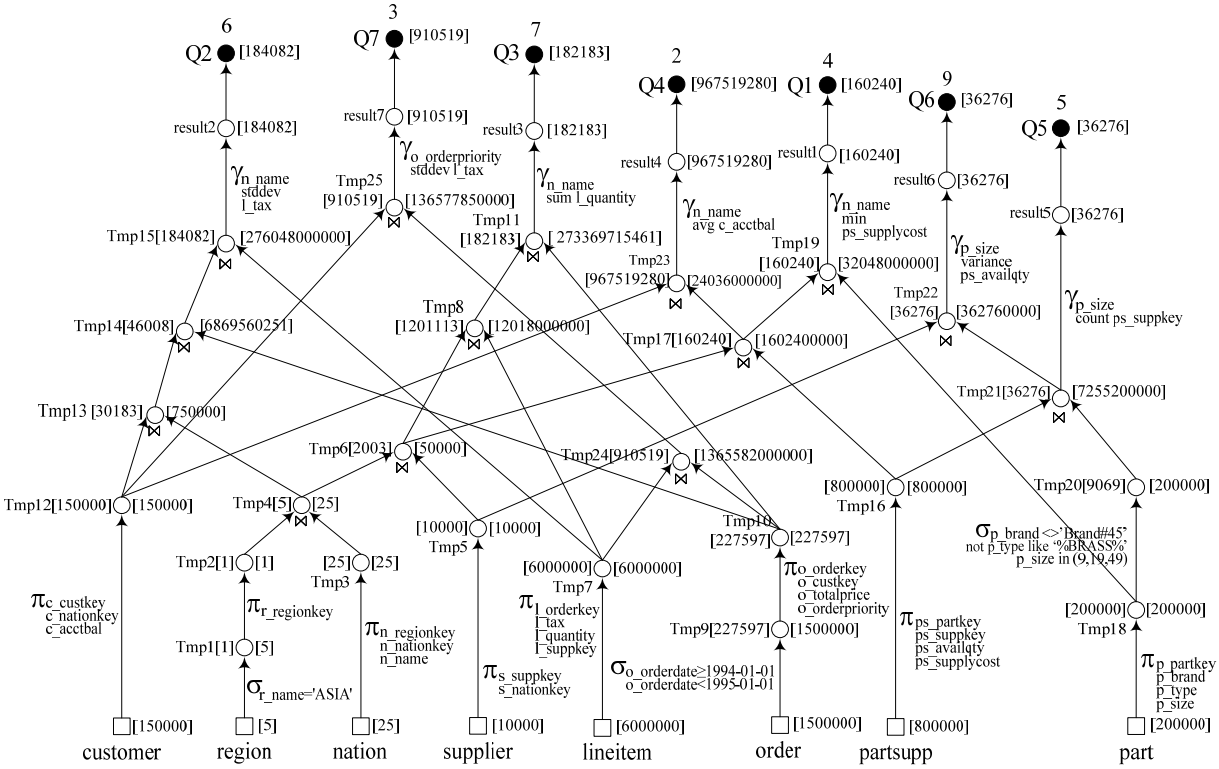

Fig. 7 An MVPP

REFERENCES

[1] W.H.Inmon, Building the Data Warehouse, John Wiley and Sons, 2002.
[2] J. Widom, "Research Problems in Data Warehousing," Int. Conf. on Information and Knowledge Management, 1995, pp. 25-30.
[3] C. Zhang, X. Yao, and J. Yang, "An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment," IEEE, 2001, 31, pp. 282-294.
[4] J. Phuboon-ob, and R. Auepanwiriyakul, "Analysis and Comparison of Algorithm for Selecting Materialized Views in a Data Warehousing Environment" APDSI, 2006, 392-395.
[5] J. Phuboon-ob, and R. Auepanwiriyakul, "Two-Phase Optimization for Selecting Materialized Views in a Data Warehouse," Enformatika Transactions on Engineering, Computing and Technology, vol. 19 pp.277-281, Jan. 2007.
[6] J. Yang, K. Karlapalem, and Q. Li, "Algorithms for Materialized View Design in Data Warehousing Environment," VLDB Conference, 1997, 136-145.
[7] R. Derakhshan, F. Dehne, O. Korn and B. Stantic, "Simulated Annealing for Materialized View Seletion in Data Warehousing Environment," DBA, 2006, 89-94.

[8] C. Zhang, and J. Yang, "Genetic Algorithm for Materialized View Selection in Data Warehouse Environements," Proc. Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK), 1999, pp.116-125.
[9] C. Zhang, X. Yao and J. Yang, "An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment," IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol. 31, pp. 282-294, 2001.
[10] S. Nahar, S. Sahni, and E. Shragowitz "Simulated Annealing and Combinatorial Optimization," *Design Automation Conference*, 1986, 293-299.
[11] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," Science, 1983, 671-680.
[12] Y. Ioannidis, and E. Wong, "Query optimization by simulated annealing," *ACM SIGMOD*, 1987, 2-22.
[13] Y. E. Ioannidis and Y. C. Kang, "Randomized Algorithm for Optimizating Large Join Queries," *ACM SIGMOD*, 1990, 312-321.
[14] Transaction Processing Performance Council. "TPC benchmarks-H Revision 2.3.0," Available: www.tpc.org, 2005.
[15] D. Stiles, "A 'C' Robust Simulated Annealing Package". Available: http://www.engineering.usu.edu/ece/research/rtpc/projects/comb/robust, 2006.

Jiratta Phuboon-ob received the B.Sc. (Hons.) degree in Statistics from Srinakharinwirot University, Mahasarakham Campus, Thailand in 1992 and the M.S. degree in Applied Statistics from School of Applied Statistics, National Institute of Development Administration (NIDA), Bangkok, Thailand in 1997.

She is currently pursuing the Ph.D. degree in Computer Science at NIDA, Bangkok, Thailand. Her research interests include databases, data warehouse and data mining.

Raweewan Auepanwiriyakul received the B.Sc. degree in Radiological Technology from Mahidol University, Thailand, in 1982 and the M.S. and Ph.D. degree in Computer Science from University of North Texas, U.S.A., in 1985 and 1989, respectively.

Currently, she is an Assistant Professor with School of Applied Statistics, National Institute of Development Administration (NIDA), Bangkok, Thailand. Her research interests include databases, object-oriented analysis and design, and data warehouse.