ASVMRT: Materialized View Selection Algorithm in Data Warehouse

Jin-Hyuk Yang*, and In-Jeong Chung*

Abstract: In order to acquire a precise and quick response to an analytical query, proper selection of the views to materialize in the data warehouse is crucial. In traditional view selection algorithms, all relations are considered for selection as materialized views. However, materializing all relations rather than a part results in much worse performance in terms of time and space costs. Therefore, we present an improved algorithm for selection of views to materialize using the clustering method to overcome the problem resulting from conventional view selection algorithms. In the presented algorithm, ASVMRT (Algorithm for Selection of Views to Materialize using Reduced Table), we first generate reduced tables in the data warehouse using clustering based on attribute-values density, and then we consider the combination of reduced tables as materialized views instead of a combination of the original base relations. For the justification of the proposed algorithm, we reveal the experimental results in which both time and space costs are approximately 1.8 times better than conventional algorithms.

Keywords: Materialized views, Data Warehouse, and Clustering

1. Introduction

Much time is required to respond to users' analytical and time-serial queries in an RDB (Relational Data Base) designed mainly for transactions such as bank operations. Therefore, in order to better support a CEO's decision-making through market analysis, the trend is to build a data warehouse which is a new concept against the traditional OLTP (On-Line Transaction Processing)-oriented RDB, and subject-oriented, integrated, non-volatile, and time variant features.

The view in a data warehouse is derived from a base relation or other view. It is a virtual relation that is recomputed whenever it is referenced. Summarizing and storing these view tuples results in materialized views. The reason for using the materialized views is to rapidly process analytical queries in a data warehouse that contains time-serial data. However, the more we use materialized views, the more storage space is needed in a data warehouse. Therefore, effective selection of materialized views should properly satisfy the factors of response time and storage space.

There are related works of view selection algorithms such as [1], [2], and [3]. Only aggregate functions are considered in [1]. A heuristic-based greedy method that uses AND, OR, and AND-OR graphs is proposed in [2]. However, evaluation of this approach is omitted. An algorithm called HA_{MVD} is proposed in [3]. However, too much time is required to produce an MVPP (Multiple View Processing Plan), which is an input variable of HA_{MVD}. Therefore, in this paper we propose an algorithm that improves the speed and space problem existing algorithms have.

In the proposed algorithm, which uses the clustering technique to select materialized views for rapid query response in a data warehouse, once clusters are found on the basis of the relative density of relation dimensions, a reduced table is then generated as the produced clusters are referenced. The generated reduced tables are the relations used for producing an MVPP in the ASVMRT (Algorithm for Selection of Views to Materialize using Reduced Table). After we produce an MVPP using the generated reduced tables, we then process and select the views effectively in the produced MVPP using the ASVMRT. For the justification of the proposed algorithm, two separate experimental results are presented: The '*pubs*' database used for educational purposes, and large database used for Information System for Telecommunications Technical Regulations (http://tris.etri.re.kr) in the ETRI (Electronics and Telecommunications Research Institute, http://www.etri.re.kr). We reveal the experimental results in which both time and space costs were approximately 1.8 times better than conventional algorithms.

The organization of this paper is as follows: Section 2 describes a data warehouse and related works on selection of materialized views. ASVMRT is proposed in section 3, and section 4 compares ASVMRT and conventional algorithms through experimentation. Finally, we conclude and suggest future works in section 5.

2. Data Warehouse and Related Works on Materialized View Selections

In this section, we provide a brief introduction of the

Manuscript received April 4, 2006; accepted April 27, 2006.
Corresponding Author: In-Jeong Chung
* Dept. of Computer Science, Korea University, 208 Seochang-Li, Chochiwon-Eup, YunkiGun, ChoongNam, Korea (grjinh@korea.ac.kr, chung@korea.ac.kr)

data warehouse and related works on selection of materialized views used for increasing the effectiveness of the data warehouse.

2.1 Data Warehouse

The data warehouse is defined as data storage for supporting enterprise decision-making, which has subject-oriented, integrated, non-volatile, and time-variant features [4, 5]. As RDB based on ER (Entity-Relationship) model for OLTP purposes has not facilities for enterprise decision-making through statistical and analytical query, a data warehouse under construction will satisfy the requests of the user which requires the OLAP (On-Line Analytical Processing) function.

2.2 Existing Algorithms for Selecting Materialized Views

A view is a relation derived from the base relation or other view. As a virtual relation, it is recomputed whenever it is referenced. Summarizing and storing these view tuples results in materialized views[6]. Indexing on the materialized views enables much faster query processing than re-computation of views for response to an analytical query.

A materialized view selection algorithm in the lattice structure is proposed in [1]. In this work, a data cube is transformed into lattice structure in which views to materialize is selected. Expression $B(v, S)$ used in their paper indicates total benefit resulting from selecting view v, and the algorithm selects views to materialize in the direction of maximizing the total benefit of $B(v, S)$. After finishing selection of all the views to materialize, the algorithm terminates and the materialized views are returned.

A view selection algorithm using AND-OR graph is proposed in [2]. The AND-OR view graph has two kinds of graphs: The AND view graph has a single query processing plan, and the OR view graph has multiple queries-processing plans. In the AND view graph, a global plan for the given queries is produced using a multiple query optimizer. The generated global plan corresponds to the AND view graph. After a query processing plan is produced, nodes (views) consisting of it are considered for materialization. The global query processing plan is divided into several small queries, and then each query is processed and merged again.

This algorithm is a greedy algorithm which does not include update cost for views and selects a set of materialized views, M, within the space constraint S. The algorithm, within the bounds of the materialized view space constraint $S(M)$, selects views to materialize one after another as maximizing benefit. When the value of space constraint S exceeds the given value, the algorithm stops and returns the materialized views set M.

The AND-OR view graph in a data cube is an OR view graph because there are several ways to create the views from other views in a data cube. The solution method of selecting views to materialize in a data cube environment is the general form of the approach taken in [1].

MVPP[3] is a DAG (Directed Acyclic Graph) in which root nodes are queries and leaf nodes are base relations. It indicates the query processing plan for views in a data warehouse. It consists of six elements: $M=(V, A, C_{qq}, C_{mr}, f_q, f_u)$. V represents a set of nodes, and A is a set of directed arcs in which the order relation between the nodes is presented. C_{qq} and C_{mr} are the costs for query processing and maintenance, respectively, and f_q and f_u are query access frequency and update frequency, respectively.

This research offers the following heuristic to reduce the search space: Under a situation where view v_1 and view v_2 are related, and v_1 is a child of v_2, if materializing v_1 has not produced any benefit, then v_2 is not considered to be materialized. This heuristic is analogous to closure property used in the Apriori[7] and DHP[8] algorithms for association rule mining among data mining techniques. The algorithm takes LV, a set containing all the nodes, and M, a set of targets to materialize, as inputs, and selects the materialized views which are contributed to produce benefit against the cost. It continues until there are no views to consider (i.e., until LV is an empty set). When it terminates, it returns materialized views set M.

As other works, [9] proposes operators which can be used in a data cube, [10] addresses the multiple view maintenance problem for the first time, [11] proposes an algorithm considering indexing on the views in a data cube, and [12] proposes a method for materialized view in a multidimensional database.

3. ASVMRT (Algorithm for Selection of Views to Materialize using Reduced Tables)

In a different manner of conventional algorithms, we present an algorithm for selecting views to materialize using the clustering method among data mining techniques [13, 14, 15, and 16].

3.1 Motivation and Example

We select and materialize the views for rapid response to analytical query in a data warehouse containing time-serial data. However, there are non-related tuples for responding to the given query among the total tuples consisting of materialized views. Therefore, we extract (make clusters) only related tuples with the given query and stored them as materialized views. The proposed algorithm for selection of materialized views guarantees not only a faster computation time of tuples, but also less storage space against the conventional materialized views selection algorithms. The following example supports this concept.

Assume that there is a salary relation (containing 700 tuples) with six dimensions and an age relation (containing 500 tuples) with eight dimensions. Through the following query, an enterprise manager can not only analyze and predict the current market trend, but also establish a new management strategy from the predicted results: What kind of car is preferred by those in their 20s with a salary of greater than $30,000 per year?

In conventional approaches, the select operation is performed from the joining of 700×500 tuples. If we create reduced tables from the salary and age relations (assume that there are 350 earners with a salary of greater than $30,000 in the salary relation, and 250 people in their 20s in the age relation), we can perform the select operation on only 350×250 tuples. As shown in this virtual example, the approach with reduced tables allows for 4 times faster speed and 2 times less storage space against approaches in which relations on the whole are considered to be materialized. In the simple and virtual example, only 2 relations are addressed. However, there are a number of views in a data warehouse environment. Therefore, it is crucial to improve and save on both response time and storage space as close to 2 times in terms of performance of a data warehouse.

3.2 ASVMRT

In general, the proposed algorithm has 4 steps:

- Step 1: Find high-density clusters from k-dimensional relations.
- Step 2: Produce reduced tables using upper and lower bound values of the clusters found.
- Step 3: Establish MVPP using reduced tables.
- Step 4: Select materialized views while considering improvement of query response time and view maintenance cost.

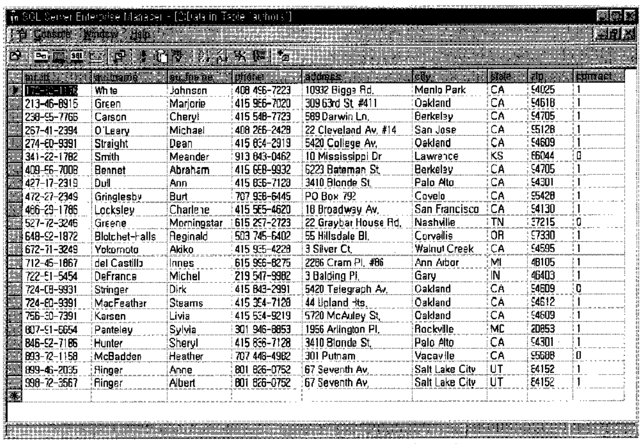
```
ASVMRT(τ, n, T, Q, SC, UDT, UET) {
/* τ: user's input threshold */
/* n: number of queries or tables */
/* T: set of target tables */
/* Q: set with n queries */
/* SC: user's input space constraint */
/* UDT: user's input clustering dimensions which must be
   included */
/* UET: user's input clustering dimensions which must be
   excluded */
   C=∅; /* set of clusters */
   RT=∅; /* set of reduced tables */
   VP=∅; /* set of views used in query processing plan */
   MV=∅; /* set of views to be materialized */
   for (i=0; i<n; i++) {
      C = C ∪ find_cluster(τ, n, T_i, UDT, UET); }
   for (i=0; i<n; i++) {
      RT = RT ∪ generate_reduced_table(C_i, T_i, RT_i); }
   make_mvpp(n, Q, RT);
   select_view(VP);
   return MV;
}
/* step 1 */
find_cluster((τ, n, T_i, UDT, UET) {
   T=T_i;
   target=0; /* variable for attributes' reflection density */
   for (i=0; i<n; i++)
     for (j=0; j<n; j++) {
     /* primary key, foreign key, and user's input
        dimension of tables are excluded */
       if (T_i.d_j == primary_key || T_i.d_j == foreign key ||
          T_i.d_j == UET_i.d_j) continue;
        /* if a dimension is user's specified input
           dimension, it is included */
       if (T_i.d_j == UDT_i.d_j) {
         for (k=0; T_i.d_i.low[k] != NULL; k++) {
         /* select a range of lower bound and upper
            bound for cluster */
           C._i = T_i.d_i.low[k], T_i.d_i.high[k]; }
         break; } /* move to the next table */
        /* is a reflection of dimension i over dimension j
           dense? Is it denser than existing reflection? */
         /* operator ∏ reflects first element over second
            element, and returns reflection density */
       else if (∏(T_i.d_i, T_i.d_j) > τ && [C._i] > target) {
         target = [C._i];
         for (k=0; T_i.d_i.low[k] != NULL; k++) {
           C._i = T_i.d_i.low[k], T_i.d_i.high[k]; }
       }
     }
   return C;
}
/* step 2 */
generate_reduced_table(C_i, T_i) {
   /* operator ← returns index */
   tmp ← T_i.C_i.low[0];
   for (k=0; T_i.C_i.low[k] != NULL; k++) {
   /* [tmp] is returns the value which tmp index indicates.*/
     while ([tmp] ≥ T_i.C_i.low[k] && [tmp] ≤ T_i.C_i.high[k])
     {
       Copy tuple from T_i to RT_i;
       tmp++; }
   }
   return RT_i;
}
/* step 3 */
make_mvpp(n, Q, RT) {
   for (i=0; i<n; i++) {
     /* produce n view processing plans using reduced
        tables as base relations */
    Make vp_i using Q and RT as base relation instead of T;
    Count the number of nodes in vp_i and save into NN_i;
  /* NN is set containing the number of nodes of each vp_i */
  }
   for (i=0; i<n; i++)
     for (j=0; j<NN_j; j++)
       for (k=0; k<NN_k; k++) {
         VP = VP ∪ vp_i;
```

/* if a common node is found, query frequency is increased */
if ($vp_i.node_j == VP_i.node_k$) $VP_i.node_k.f_q$++; }
return *VP*;
}
/* step 4 */
select_view(*VP*) {
/* for n queries, compute query processing time cost(C_a), query maintenance cost(C_m), and total cost(C_v) of nodes of *VP* in case of materializing each node */
for (i=0; i<n; i++) {
for (j=0; j<n; j++) {
$VP_i.C_a = VP_i.C_a + VP_i.node_j.C_a$;
$VP_i.C_m = VP_i.C_m + VP_i.node_j.C_m$;
$VP_i.C_v = VP_i.C_v + VP_i.C_a + VP_i.C_m$; }
$VP.C_a = VP.C_a + VP_i.C_a$;
$VP.C_m = VP.C_m + VP_i.C_m$;
$VP.C_v = VP.C_v + VP.C_a + VP.C_m$; }
/* sort the elements of *VP* in ascending order according to the value of C_v */
Sort(*VP*);
/* select views within the bound of specified *SC* */
for (i=0; i<n; i++) {
/* operator $\sum$ returns storage space */
if ($\sum_T MV < SC$) {
$MV = MV \cup VP_i$;
$MV.C_v = MV.C_v + VP_i.C_v$; }
else break;
}
return *MV*;
}

3.3 ASVMRT Example

In this section, we show each step of the ASVMRT through an example. We chose the SQL Server 7.0's '*authors*' table of the *pubs* database, which is broadly used for educational purposes. Fig. 1 and 2 show the *pubs* database schema and *authors* table consisting of *pubs*, respectively.

Relative density for the attributes of *authors* relation is described in tables 1 and 2. In order to facilitate



au_id	au_lname	au_fname	phone	address	city	state	zip	contract
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	1
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	1
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	1
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	95128	1
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	1
341-22-1782	Smith	Meander	913 843-0462	10 Mississippi Dr.	Lawrence	KS	66044	0
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	94705	1
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA	95428	1
486-29-1786	Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA	94130	1
527-72-3246	Greene	Morningstar	615 297-2723	22 Graybar House Rd.	Nashville	TN	37215	0
648-92-1872	Blotchet-Halls	Reginald	503 745-6402	55 Hillsdale Bl.	Corvallis	OR	97330	1
672-71-3249	Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA	94595	1
712-45-1867	del Castillo	Innes	615 996-8275	2286 Cram Pl. #86	Ann Arbor	MI	48105	1
722-51-5454	DeFrance	Michel	219 547-9982	3 Balding Pl.	Gary	IN	46403	1
724-08-9931	Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA	94609	0
724-80-9391	MacFeather	Stearns	415 354-7128	44 Upland Hts.	Oakland	CA	94612	1
756-30-7391	Karsen	Livia	415 534-9219	5720 McAuley St.	Oakland	CA	94609	1
807-91-6654	Panteley	Sylvia	301 946-8853	1956 Arlington Pl.	Rockville	MD	20853	1
846-92-7186	Hunter	Sheryl	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
893-72-1158	McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA	95688	0
899-46-2035	Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT	84152	1
998-72-3567	Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT	84152	1

Fig. 2. *Authors* table of *pubs*

computation of relative density conceptually, the scope in tables 1 and 2 range from 0 to 9 for the dimensions with numerical values, and from a to z for the dimensions with non-numerical values (i.e., alphabetic), respectively. More range values are required in real data warehouse relations containing a large number of records. The density value is computed as normalization of the number of records that belong to the corresponding scope among all records. And, among all records, for example, the relative density value of tuples with a value of 9xxxx in zip dimension is a percentage of ratios (0.06957) on the number of all tuples (16/23) divided by normalization factor (10) which is used to normalize the dimensions with numerical values. In this case, the density value of the tuples with 9xxxx in the zip dimension against the entire table is 69.57%(6.957×10). If the density threshold variable τ coming from the user's input is 60%, zip dimension is considered a candidate. Assume that the user inputs the zip and contract dimensions as *UDT* and *UET*, respectively.

Table 1. Relative density of *authors* table with numerical values

scope	au_id	density	phone	density	address	density	zip	density	scope	contract	density
0					1	0.435			0	4	8.696
1	1	0.44			4	1.739			1	19	41.3
2	4	1.14	1	0.435	3	1.304	1	0.435			
3	1	0.44	1	0.435	6	2.609	1	0.435			
4	4	1.74	13	5.652	1	0.435	2	0.87			
5	1	0.44	1	0.435	5	2.174					
6	2	0.87	2	0.87	3	1.304	1	0.435			
7	5	2.17	2	0.87							
8	4	1.74	2	0.87			2	0.87			
9	1	0.44	1	0.435			16	6.957			

In tables 1 and 2, we can see that the relative density of contract dimension has the highest density value. However, we do not consider this dimension for clustering, since contract dimension is registered in *UET*. And, the au_id dimension is excluded for clustering because it is a primary key. We consider the zip dimension for clustering since it is specified in *UDT*. If no variable is specified in *UDT*, zip dimension is selected because its relative density value,

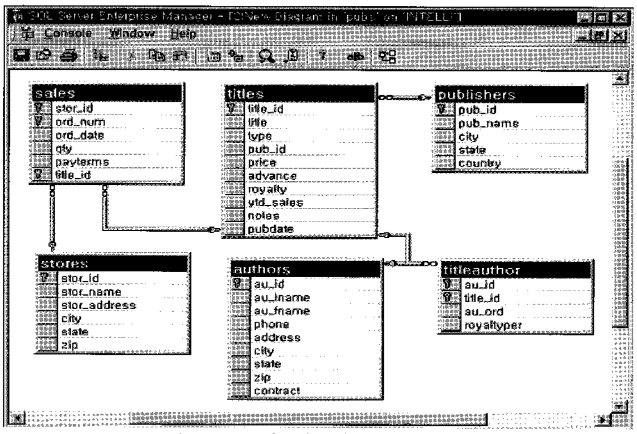
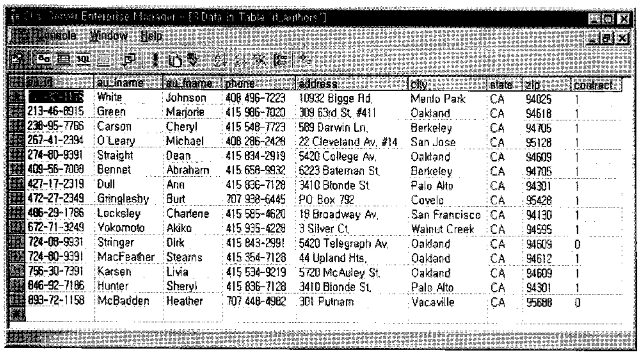


Fig. 1. *Pubs* database schema

6.957, is the highest. Once zip dimension is selected for clustering, we can produce the reduced table containing only the tuples that belong to the corresponding range. Fig. 3 shows the reduced table of *authors* relation.

Table 2. Relative density of *authors* table with alphabetic values

scope	au_lname	density	au_fname	density	city	density	state	density
a			5	0.835	2	0.334		
b	2	0.334	1	0.167	2	0.334		
c	1	0.167	2	0.334	2	0.334	15	2.505
d	3	0.501	2	0.334				
e								
f								
g	3	0.501			1	0.167		
h	1	0.167	1	0.167				
I			1	0.167			1	0.167
j			1	0.167				
k	1	0.167					1	0.167
l	1	0.167	1	0.167	1	0.167		
m	2	0.334	5	0.835	1	0.167	2	0.334
n					1	0.167		
o	1	0.167			5	0.835	1	0.167
p	1	0.167			2	0.334		
q								
r	2	0.334	3	0.501	1	0.167		
s	3	0.501	1	0.167	4	0.668		
t							1	0.167
u							2	0.334
v								
w	1	0.167			1	0.167		
x								
y	1	0.167						
z								



au_id	au_lname	au_fname	phone	address	city	state	zip	contract
[illegible]	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	1
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	1
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	1
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	95128	1
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	1
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	94705	1
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA	95428	1
486-29-1786	Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA	94130	1
672-71-3249	Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA	94595	1
724-08-9931	Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA	94609	0
724-80-9391	MacFeather	Stearns	415 354-7128	44 Upland Hts.	Oakland	CA	94612	1
756-30-7391	Karsen	Livia	415 534-9219	5720 McAuley St.	Oakland	CA	94609	1
846-92-7186	Hunter	Sheryl	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
893-72-1158	McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA	95688	0

Fig. 3. Reduced table for *authors* relation

The same method results in reduced tables for all relations in a data warehouse. In the third step of ASVMRT, we established MVPP using the reduced tables. For an illustration of the third step of the algorithm, assume there are 4 queries.

- Q1: What is the average on year-to-date sales of CA residents with a value of greater than 80 in royalty per?
- Q2: What are the top 3 kinds of bestseller books from 1993 to 1995 in CA region?
- Q3: Among the books with high value of royalty per, what are the titles of the books which are about economics and with price greater than $15?
- Q4: What are the books printed by an American publisher which are about psychology, and the author of the book is in CA region?

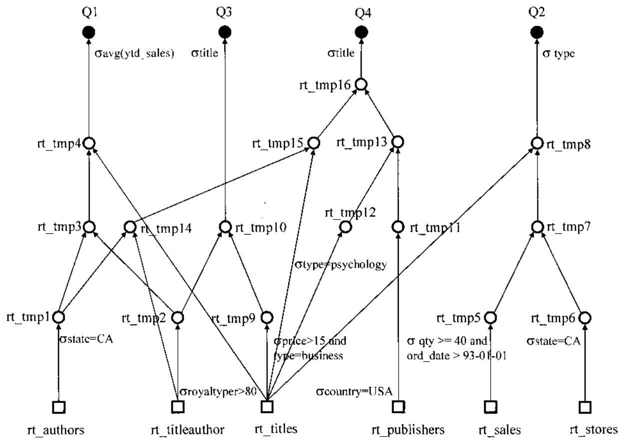


Fig. 4. MVPP for 4 queries

Fig. 4 above is MPVV for 4 queries. In Fig. 4, □ indicates a base relation, o is for intermediary value, and • is used for a query. Once an MVPP is established as shown in Fig. 4, views to be materialized are selected considering cost. The base unit of cost estimation used in the paper is the number of tuples as adopted in [1] and [3].

If we select the *rt_tmp3* relation as the materialized view, the total cost C_t is an addition of view processing time-cost C_a(12)(the number of tuples of *rt_tmp3*(6) and *rt_tmp4*(6), since only *rt_tmp3* and *rt_tmp4* are used to process Q1) and view maintenance cost C_m(2×56=112)(when *rt_tmp3* is stored as materialized view, maintenance cost of *rt_tmp3* is multiplied by 2, after addition of *rt_tmp3*(6), *rt_tmp1*(15), *rt_tmp2*(10), *rt_authors*(15), and *rt_titleauthors*(10)). Multiplication by 2 is due to the fact that if there is any update in *rt_tmp3*, all the children of it(*rt_tmp1*, *rt_tmp2*, *rt_authors*, and *rt_titleauthors*) should be recomputed. Total cost *T* is an addition of the total cost of Q1, Q2, Q3, and Q4. In a similar manner, we can fill in table 3. When *SC* is given by 10 in the third step of ASVMRT as shown in table 3, intermediary views *rt_tmp6*, *rt_tmp5*, *rt_tmp7*, *rt_tmp8*, and *rt_tmp9* are selected. In this case, the additional space needed for materialization is 8.

The first column in tables 3, 4, 6, and 7 indicates the relations used in MVPP, the second column is the query frequency(f_q), the third is the number of tuples(*t#*), and the fourth, fifth, and sixth are view processing time-cost(C_a), view maintenance(C_m), and total cost(C_t), respectively. The final column represents total cost(*T*) for all the queries.

Table. 3. Cost computation for 4 queries with reduced tables

			Ca				Cm				Cv				
	fq	t#	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	T
rt_authors	1	15	42			54	0			0	42			54	96
rt_titleauthor	2	10	64		42	68	0		0	0	64		42	68	174
rt_titles	5	18	120	95	105	180	0	0	0	0	120	95	105	180	500
rt_publishers	1	6				29				0				29	29
rt_sales	1	11		14				0				14			14
rt_stores	1	3		8				0				8			8
rt_tmp1	2	15	54			78	60			60	114			138	252
rt_tmp2	2	10	44		22		40		40		84		62		146
rt_tmp3	1	6	12				112				124				124
rt_tmp4	1	6	6				124				130				130
rt_tmp5	1	1		3				24				27			27
rt_tmp6	1	3		5				12				17			17
rt_tmp7	1	1		2				38				40			40
rt_tmp8	1	1		1				40				41			41
rt_tmp9	1	2			3				40				43		43
rt_tmp10	1	1			1				81				82		82
rt_tmp11	1	6				23				24				47	47
rt_tmp12	1	5				22				46				68	68
rt_tmp13	1	5				17				80				97	97
rt_tmp14	1	6				24				92				116	116
rt_tmp15	1	6				18				140				158	158
rt_tmp16	1	12				12				220				232	232

3.4 Analysis and Features of ASVMRT

In the first step of the algorithm, the high-density cluster for target base relations is found using the clustering method among data mining techniques. For each dimension of the table, the dimension with the maximum density value is selected, which is exceeding the user's input threshold τ. The lower and upper bound values for the selected dimension are stored, and these data are used in the second step of ASVMRT. As a novel approach which is not considered in conventional algorithms, this kind of technique with clustering is crucial from the standpoint of providing an opportunity to implicitly utilize important information overlooked. Again, by using the clustering technique, the benefits of not only providing potentially useful information, but also improving query processing time and saving view storage space can be achieved. And, any dimension of a table to be reflected in the algorithm can be included for clustering at the user's discretion. The user's input dimension for clustering (specified in the *UDT* variable of the algorithm) has top priority against other dimensions with a value greater than a given threshold. Granting this ability guarantees that if a dimension contains important information, even a small quantity of data in appearance can be included and reflected for clustering. The user's external input capability of dimension excludes the possibility of destroying important information.

In the second step of the algorithm, reduced tables containing the only corresponding tuples (i.e., example shown in the previous subsection, tuples with 9xxxx value in zip dimension) are produced by using the lower and upper bound values of the selected dimension for each table. While traditional algorithms consider all the tuples of a base relation for materializing [1], the targets of materializing are restricted to the tuples of the reduced tables in the proposed algorithm ASVMRT. Therefore, it can achieve the goals of improvement in query response time and saving of storage for views. Note that it requires larger storage space (for intermediary reduced tables) and takes more time for clustering. However, off-line tasks of the clustering phase and production step of reduced tables do not lower the performance of a data warehouse system, since it is almost impossible[2] to process tasks such as updating and maintaining views on-line in a data warehouse containing scores of terabytes of data.

In the third step of the algorithm, we produced an MVPP by using the reduced tables generated in the previous step. The existing algorithm[3] proposed the 0-1 integer programming method and HA_{mvpp} for establishing MVPP. While this 0-1 integer programming technique produces optimal MVPP, it takes too much time to implement. In our algorithm, we propose the off-line procedure for establishing MVPP using query frequency.

In the fourth step of the algorithm, the views which can derive benefits in the case of materialized ones were selected within the bounds of the user's input space constraint, while considering view processing time cost and view maintenance cost in the produced MVPP. The conventional algorithms consider only the cost for *join* operation and restrict query frequency to the query itself. We argue that these cost estimation methods leave out some important factors in cost. In the ASVMRT, cost for the *select* operation is supplemented to cost estimation formulation. Also, we imposed query frequency on all the tuples consisting of the query rather than the query itself because we considered the fact that the views consisting of the query can be used in another query.

4. Implementation Results and Analyses

In this section, we first present the implementation results on the *pubs* database, and then reveal the experimental results of applying ASVMRT to Information System for Telecommunications Technical Regulations in ETRI in order to improve the response time of the article keyword-based search method.

---

[1] Note that the conventional algorithms also consider partial materializing, which means they select a portion of views rather than all the views in a data warehouse. However, be aware of the difference between partial materializing and our approach; our approach is novel from the standpoint that a portion of the tuples of each base relation is extracted and reduced tables are generated. Then, partial materializing is applied to reduced tables rather than base relations.

[2] Note that there are researches on on-line updating and maintaining strategy. However, most data warehouse systems perform the updates and maintenance of views off-line because the data warehouse is mainly for read-only and these kinds of tasks require much time.

4.1 Experimentation and Results in Pubs Database

Taken from conventional algorithms, the cost estimation approach for 4 queries (same queries as in section 3) without reduced tables is presented in table 4. Table 5 results from referencing the entire queries log and summarizing tables 3 and 4. For a comparison of the conventional approach with ours, we assumed that the space constraint variable SC from the user's input is not specified.

Table 4. Cost computation for 4 queries without reduced tables

			Ca				Cm				Cv				
	fq	t#	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	T
authours	1	23	50			89	0			0	50			89	139
titleauthor	2	25	94		72	152	0		0	0	94		72	152	318
titles	5	18	120	105	105	260	0	0	0	0	120	105	105	260	590
publishers	1	8				36				0				36	36
sales	1	21		30				0				30			30
stores	1	6		15				0				15			15
tmp1	2	15	54			132	72			72	126			204	330
tmp2	2	10	44		22		70		70		114		92		206
tmp3	1	6	12				158				170				170
tmp4	1	6	6				170				176				176
tmp5	1	3		9				48				57			57
tmp6	1	3		9				18				27			27
tmp7	1	3		6				72				78			78
tmp8	1	3		3				78				81			81
tmp9	1	2			3				40				43		43
tmp10	1	1			1				112				113		113
tmp11	1	6				28				28				56	56
tmp12	1	5				27				46				73	73
tmp13	1	5				22				84				106	106
tmp14	1	17				57				180				237	237
tmp15	1	17				34				250				284	284
tmp16	1	17				17				464				481	481

Table 5. Performance comparison on the *pubs* database

		Conventional algorithms	ASVMRT
Partial materialization case	Materialized views	tmp5, tmp6, tmp7, tmp8	rt_tmp5, rt_tmp6, rt_tmp7, rt_tmp8
	Total cost	243	125
	Storage space	12	6
Full materialization case	Materialized views	ALL	ALL
	Total cost	3,646	2,441
	Storage space	220	149

As shown in table 5, the case of materializing views partially indicates that the proposed method against the conventional approach shows 1.944(243/125) times and 2(12/6) times better performance in query response time and view storage space, respectively. Even in the other case of materializing all the intermediary views (i.e., when the appropriate algorithm for selection is not applied) our approach shows 1.493(3,646/2,441) times and 1.476(220/149) times better performance in query response time and view storage space, respectively. This 1.5 times increase is somewhat different from the 1.8 times improvement shown in the next subsection. This is because the number of records in the *pubs* database is inadequate.

4.2 Experimentation and Results in Information System for Telecommunications Technical Regulations

In this section, we present experimentation on a database with a large quantity of data rather than a small database such as the *pubs* database. The target database is used in Information System for Telecommunications Technical Regulations (http://tris.etri.re.kr) in ETRI (http://etri.re.kr). The database schema of the information system is shown in Fig. 5.

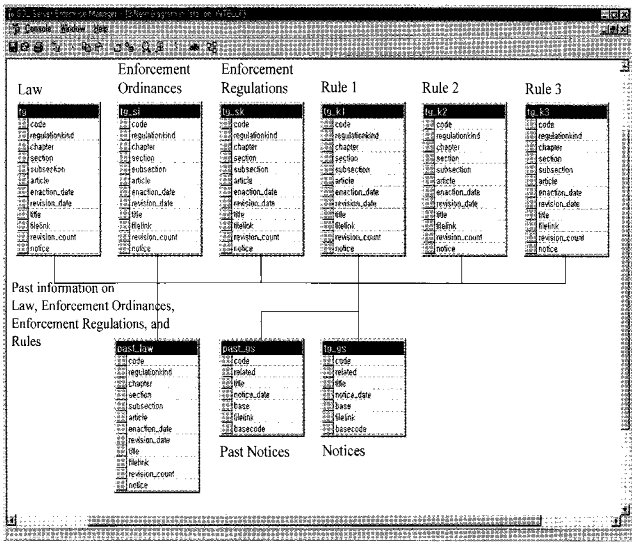


Fig. 5. Database schema of the Information System

In this section, we present the results of an experiment on the law of radio waves among the 14 laws of ordinances consisting of the law of information and communication of Korea, since the number of tuples of that law is highest. As one of 4 search strategies supplied by the information system, an article keyword-based search takes the keyword from users, compares it with a title of an article of the law, and then returns the retrieval results. In order to improve the response time of article keyword-based searches, we considered the articles of the law containing the related notices as clustering targets. After generating the reduced tables on the law of radio waves, we produced the MVPP as referencing for the queries log. Assume the following query.

- Q5: Among the articles related to the law of radio and waves, list all the articles of the law which are based on the notices related to the law of radio and waves.

The query Q5 retrieves all the queries related to the involved notices. Though a user inputs an arbitrary keyword, the results of query Q5 embody the user's query results. Therefore, query Q5 is representative of all the queries for the notice-related information retrieval, since it includes all the possible range of results returned from the notice-related information retrieval. Fig. 6 shows an MVPP for the query Q5.

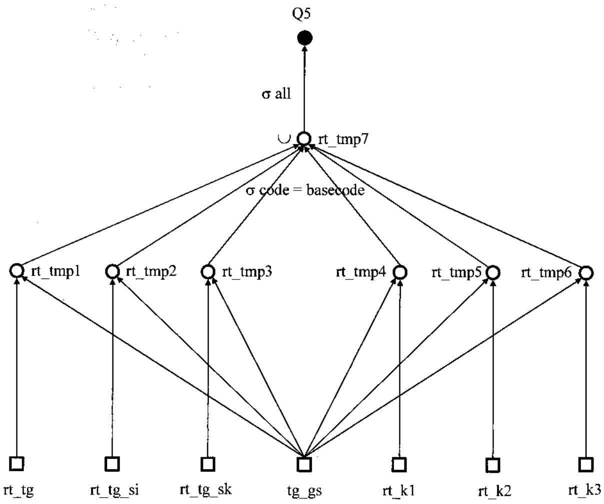


Fig. 6. MVPP for the query Q5

Table. 6. Cost computation for the query Q5 with reduced tables

			Ca	Cm	Cv
	fq	t#	Q5	Q5	Q5
tg_gs	1	119	48,810	0	47,405
rt_tg	1	71	8,682	0	8,682
rt_tg_si	1	72	8,752	0	8,752
rt_tg_sk	1	116	14,032	0	14,032
rt_tg_k1	1	77	9,352	0	9,352
rt_tg_k2	1	29	3,592	0	3,592
rt_tg_k3	1	26	3,232	0	3,232
rt_tmp1	1	8,499	8,611	17,378	25,989
rt_tmp2	1	8,568	8,680	17,518	26,198
rt_tmp3	1	13,804	13,916	28,078	41,994
rt_tmp4	1	9,163	9,275	18,718	27,993
rt_tmp5	1	3,451	3,563	7,198	10,761
rt_tmp6	1	3,094	3,206	6,478	9,684
rt_tmp7	1	112	112	94,402	94,514

Table 6 resulting from ASVMRT shows the cost estimation with reduced table. The approach taken from conventional algorithms results in table 7, which is for cost estimation without reduced tables. If the user's input variable *SC* is given by 30,000, then *rt_tmp6*, *rt_tmp5*, *rt_tmp1*, and *rt_tmp2* are selected in sequence as materiallized views. In this case, view storage space is 23,612. As shown in table 8, our algorithm achieves 1.754(127,417/72,632) times and 1.768(41,769/23,612) times better performance in terms of response time and view storage space, respectively.

Summarizing and distinguishing tables 6 and 7 results in table 8. When a user's input variable *SC* is not given, our proposed algorithm shows 1.786 (593,591/332,180) times and 1.794 (84,713/47,201) times improvement, on average, in terms of query response time and view storage space, respectively.

Table 7. Cost computation for the query Q5 without reduced tables

			Ca	Cm	Cv
	fq	t#	Q5	Q5	Q5
tg_gs	1	119	84,036	0	84,039
tg	1	121	14,634	0	14,634
tg_si	1	132	15,954	0	15,954
tg_sk	1	219	26,394	0	26,394
tg_k1	1	134	16,194	0	16,194
tg_k2	1	49	5,994	0	5,994
tg_k3	1	49	5,994	0	5,994
tmp1	1	14,399	14,513	29,278	43,791
tmp2	1	15,708	15,822	31,918	47,740
tmp3	1	26,061	26,175	52,798	78,973
tmp4	1	15,946	16,060	32,398	48,458
tmp5	1	5,831	5,945	11,998	17,943
tmp6	1	5,831	5,945	11,998	17,943
tmp7	1	114	114	169,426	169,540

Table. 8. Performance comparison on the database of Information System (http://tris.etri.re.kr)

		Conventional algorithms	ASVMRT
Partial materialization case	Materialized views	tmp1, tmp2, tmp5, tmp6	rt_tmp1, rt_tmp2, rt_tmp5, rt_tmp6
	Total cost	127,417	72,632
	Storage space	41,769	23,612
Full materialization case	Materialized views	ALL	ALL
	Total cost	593,591	332,180
	Storage space	84,713	47,201

5. Conclusion and Future Works

The proposed algorithm ASVMRT, firstly, finds high density clusters from the dimensions of the given tables, and secondly, produces the reduced tables using the found clusters. Next, the MVPP is produced using the reduced tables, and finally, materialized views are selected from the MVPP in accordance with cost estimation.

The technique of materializing views is required to increase the query response time in a data warehouse, which provides guidelines to enterprise managers through the analysis of market trends by supporting various OLAP capabilities. As a technique of materializing views, ASVMRT is proposed in this paper, which adopts one of the data mining techniques (i.e., clustering method). In the proposed algorithm, the user can specify a dimension for mandatory clustering. This function excludes the possibility of leaving out the important information. The user can also specify the threshold value that indicates the compression strength of clusters. Finally, the user is able to input a space constraint value within which materialized views are selected. These kinds of user interfaces are not found in conventional algorithms.

As shown in the experimental results, the proposed algorithm achieves almost 1.8 times better performance in terms of both query response time and storage space of materialized views. Even in the case where the value of the space constraint variable is not specified (i.e., when we assume there is no space constraint), our algorithm shows 1.5 times and 1.8 times better performance in the *pubs* database and database for Information System of ETRI, respectively.

Broadly, there lie two issues with the data warehouse. The first is selection of materialized views, and the other is maintenance of the views for consistency of a data warehouse. ASVMRT in this paper is in regards to the first issue. As future works, we will focus on how to update and maintain the reduced tables when there occurs any update in the source data.

References

[1] Harinarayan V., Rajaraman A., and Ullman J., "Implementing data cubes efficiently", In Proc. of the ACM SIGMOD International Conference of Management of Data, Canada, June 1996.

[2] Gupta H., "Selection of views to materialized in a data warehouse", In ICDT, 1997

[3] Yang J., Karlapalem K., Li Q., "Algorithms for materialized view design in data warehousing environment", In Proc. of VLDB'97, pp.136-145.

[4] Inmon W., "Building the Data Warehouse", 2/e, John Wiley and Sons. Inc., 1996.

[5] Red Brick System, "Ins & Outs(and everything in between) of Data Warehousing", Red Brick Systems white paper, 1996.

[6] Gupta A., Mumick I., "Maintenance of Materialized Views: Problems, Techniques, and Applications, IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, 18(2), pp.3-18, June 1995.

[7] Agrawal R. and Srikant R., "Fast algorithms for mining association rules", In Proceedings of the 20th VLDB Conference, Santiago, Chile, Sept. 1994.

[8] Park J.S., Chen M.S., and Yu P.S., "An effective hash-based algorithm for mining association rules", In Proceedings of ACM SIGMOD Conference on Management of Data, pp.175-186, SanJose, California, May, 1995.

[9] Gary J., Bosworth A., Layman A., Pirahesh H., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub- Totals", Micro soft Technical Report No. MSR- TR-95-22.

[10] Ross K.A., Srivastava D., and Sudarshan S., "Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time", In Proc. ACM SIGMOD '96, pp.447-458, Montreal, June 1996.

[11] Gupta H., Harinarayan V., Rajaraman A., Ullman J.D., "Index Selection for OLAP", Proceedings of the International Conference on Data Engineering, pp.208-219, Binghamton, UK, April, 1997.

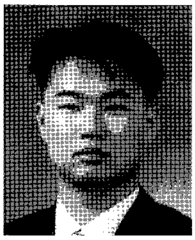
[12] Baralis E., Paraboschi S., Teniente E., "Materialized View Selection in a Multidimensional Database", Proc. VLDB '97, pp.156-165.

[13] Chen M.S., Han J., and Yu P., "Data Mining: An Overview from Database Perspective", IEEE Trans. on Knowledge and Data Engineering, 1997.

[14] Agrawal Rakesh, Imielinski Tomasz, and Swami Arun, "Database Mining: A Performance Perspective", IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.6, pp.914-925, December 1993.

[15] Berson and Smith J., "Data Warehousing, Data Mining & OLAP", McGraw-Hill, 1997.

[16] Fayyad U. M., Piatetsky-Shapiro G., Smyth P and Uthurusamy R., "Advances in Knowledge Discovery and Data Mining", Cambridge Ma: AAAI Press/MIT press 1996.



Jin-Hyuk Yang

He received the B.S. and M.S. degrees in Computer Science Dpt. from Korea University in 1998 and 2000, respect-tively. He is now Ph.D. candidate student in Korea University. His research interests include semantic web, ontology, web services and ubiquitous computing.

In-Jeong Chung

He received the B.S. and M.S. degrees in Computer Science Dpt. from Seoul National University in 1978, and Korea Advanced Institute of Science and Technology in 1980, respectively. He received Ph.D. from University of Iowa in 1989. He is now professor in Korea University. His research interests are in the area of intelligent web services, semantic web, ontology engineering, home network and ubiquitous computing.