Algorithm of Data Warehouse using Reduced Table Technique

Nitika Arora, Department Of Computer Science,

Govt. College for Women Karnal

Abstract: *In order to have a accurate and fast response to an analytical query there is need of proper selection of the views to materialize in the data warehouse is crucial. In traditional algorithms of views selection, all relations are considered for selection as materialized views. But materializing all relations rather than a part will results in much worse performance in terms of time and space costs. So, 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,* ***ADWRT*** *(Algorithms for data warehouse by using reduced table technique), we first generate reduced tables in the data warehouse using clustering, and then we consider the combination of reduced tables instead of a combination of the original relations.*

Keywords: Clustering, Data Warehouse, Materialized views,OLTP.

1. Introduction

Much time is required for responding to users' analytical and time-serial queries in an RDB (Relational Data Base) which is designed mainly for transactions such as bank operations. Therefore, in order to better support a 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, 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.

2. Data Warehouse and Related Works on Materialized View Selections

In this section, we illustrate a brief introduction of the data warehouse and related works on selection of materialized views which are used for increasing the efficiency 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

2.2 Existing Algorithms for Selecting Materialized Views

A view is a relation which is derived from the base relation or other view. Summarizing and storing these view tuples results in materialized views. Indexing on the materialized views enables much faster query processing rather than re-computation of views for response to an analytical query. 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 plan corresponds to the AND view graph. After a query processing plan is produced, nodes which are 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 latest 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 for 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 (Multiple View Processing Plan) [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, Cqq, Cmr, fq, fu). V represents a set of nodes, and A is a set of directed arcs in which the order relation between the nodes is presented. Cqq and Cmr are the costs for query processing and maintenance, respectively, and fq and fu are query access frequency and update frequency, respectively. This research offers the following heuristic to reduce the search space: Under a situation where view v1 and view v2 are related, and v1 is a child of v2, if materializing v1 has not produced any benefit, then v2 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. Algorithms for data warehouse by using reduced table technique (ADWRT) [Jin-Hyuk Yang, In-Jeong Chung]

In order to acquire a correct and quick response to an analytical query, proper selection of the views to materialize in the data warehouse is important. In older 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, there is an improved algorithm for selection of views to materialize using the clustering method to overcome the problem resulting from traditional view selection algorithms. In the presented algorithm , first reduced tables are generated in the data warehouse using clustering based on attribute-values density, and then the combination of reduced tables are considered as materialized views instead of a combination of the original base relations.

3.1 Motivation and Example

Views are selected and materialized 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, only related tuples are extracted (make clusters) 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 700tuples) 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 reduced tables are created 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), 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.

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 a MVPP in the Algorithms for data warehouse by using reduced table technique (ADWRT). After producing an MVPP using the generated reduced tables, then process and select the views effectively in the produced MVPP using the ADWRT. For the justification of the proposed algorithm, one separate experimental result are presented: The 'hospital' database used which reveal the experimental results in which both time and space costs were approximately better than conventional algorithms.

3.2ADWRT

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.

```
ADWRT (τ, n, T, Q, SC, ICI, ICE)
{
/* 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 */
/* ICI: user's input clustering dimensions which must be
Included */
/* ICE: user's input clustering dimensions which must be
Excluded */
C=null; /* set of clusters */
RT=null; /* set of reduced tables */
VP=null; /* set of views used in query processing plan */
MV=null; /* set of views to be materialized */
for (i=0; i<n; i++) {
```

```
C = C ∪ search_cluster (τ, n, Ti, ICI, ICE); }
for (i=0; i<n; i++) {
RT = RT ∪ generate_reduced_table (Ci, Ti, RTi); }
create_mvpp(n, Q, RT);
choose_view(VP);
return MV;
}/* step 1 */
search_cluster((τ, n, Ti, ICI, ICE) {
T=Ti;
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 inputdimension of tables are excluded */
if (Ti.dj == primary_key || Ti.dj == foreign key ||
Ti.dj == UETi.dj)
 continue;
/* if a dimension is user's specified input
dimension, it is included */
if (Ti.dj == UDTi.dj)
{
for (k=0; Ti.di.low[k] != NULL; k++) {
/* select a range of lower bound and upperbound for cluster */
C.i = Ti.di.low[k], Ti.di.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 Π reflects first element over second
element, and returns reflection density */
else if (Π(Ti.di, Ti.dj) > τ && [C.i] > target) {
target = [C.i];
for (k=0; Ti.di.low[k] != NULL; k++) {
C.i = Ti.di.low[k], Ti.di.high[k]; }
}
return C;
```

```
}
/* step 2 */
generate_reduced_table(Ci, Ti) {
/* operator ← returns index */
tmp ← Ti.Ci.low[0];
for (k=0; Ti.Ci.low[k] != NULL; k++) {
/* [tmp] is returns the value which tmp index indicates.*/
while ([tmp] ≥ Ti.Ci.low[k] && [tmp] ≤ Ti.Ci.high[k])
{
Copy tuple from Ti to RTi;
tmp++; }
}
return RTi;
}
/* step 3 */
create_mvpp(n, Q, RT) {
for (i=0; i<n; i++) {
/* produce n view processing plans using reduced tables as base relations */
Make vpi using Q and RT as base relation instead of T;Count the number of nodes in vpi and save into NNi;
/* NN is set containing the number of nodes of each vpi */
}
for (i=0; i<n; i++)
for (j=0; j<NNj; j++)
for (k=0; k<NNk; k++) {
VP = VP ∪ vpi;
/* if a common node is found, query frequency is
increased */
if (vpi.nodej == VPi.nodek) VPi.nodek.fq++; }
return VP;
}
/* step 4 */
choose_view(VP)
{
/* for n queries, compute query processing time cost(Ca),
query maintenance cost(Cm), and total cost(Cv) of
nodes of VP in case of materializing each node */
for (i=0; i<n; i++) {
for (j=0; j<n; j++) {
VPi.Ca = VPi.Ca + VPi.nodej.Ca;
VPi.Cm = VPi.Cm + VPi.nodej.Cm;
VPi.Cv = VPi.Cv + VPi.Ca + VPi.Cm; }
VP.Ca = VP.Ca + VPi.Ca;
VP.Cm = VP.Cm + VPi.Cm;
VP.Cv = VP.Cv + VP.Ca + VP.Cm; }
/* sort the elements of VP in ascending order according
to the value of Cv */
Sort(VP);
/* select views within the bound of specified SC */
for (i=0; i<n; i++) {
/* operator Σ returns storage space */
if (ΣTMV < SC) {
MV = MV ∪ VPi;
MV.Cv = MV.Cv + VPi.Cv; }
else break;
}
return MV;
}
```

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 ADWRT. 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 to 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 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, the targets of materializing are restricted to the tuples of the reduced tables in the proposed algorithm ADWRT. 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.

In the third step of the algorithm, an MVPP is produced by using the reduced tables generated in the previous step. The existing algorithm proposed the 0-1 integer programming method and HAmvpp for establishing MVPP. While this 0-1 integer programming technique produces optimal MVPP, it takes too much time to implement. In our algorithm off-line procedure are proposed 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.

3.3 ADWRT Example

Each step of the ADWRT has been explained through an example. The MS-Access 'Patient_detail' table of the hospital database, which is broadly used in hospital information system. Figure 1 describes storage of Hospital database schema using MS-Access and figure 2 shows data for Patient_detail table.

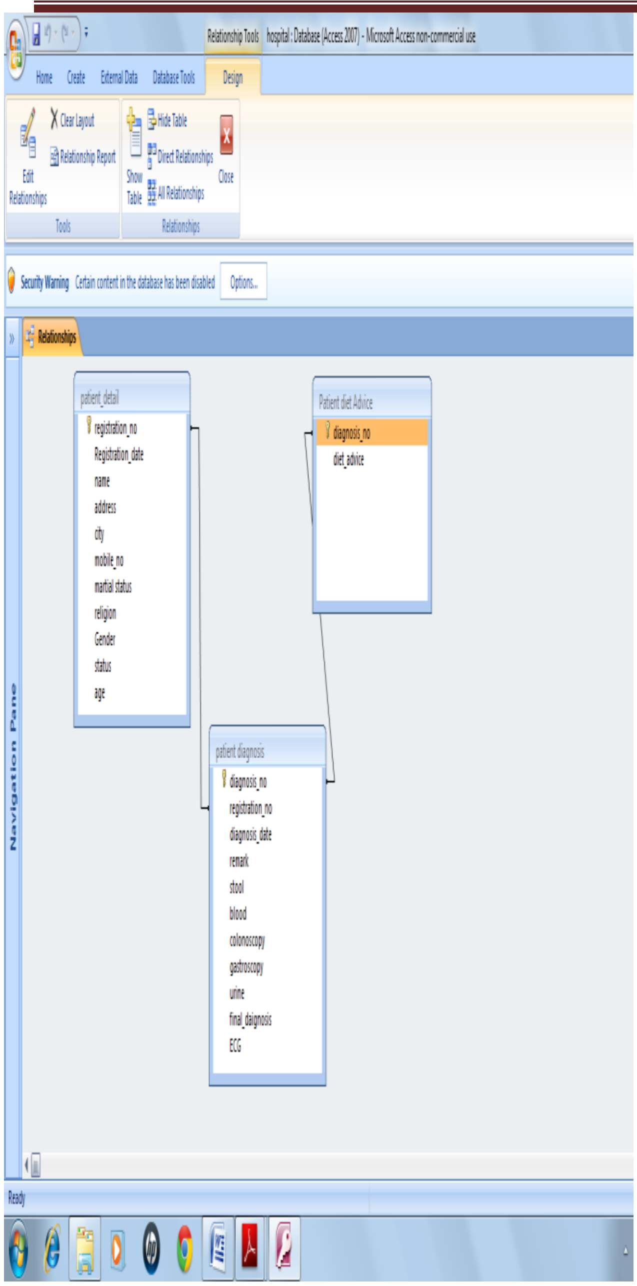
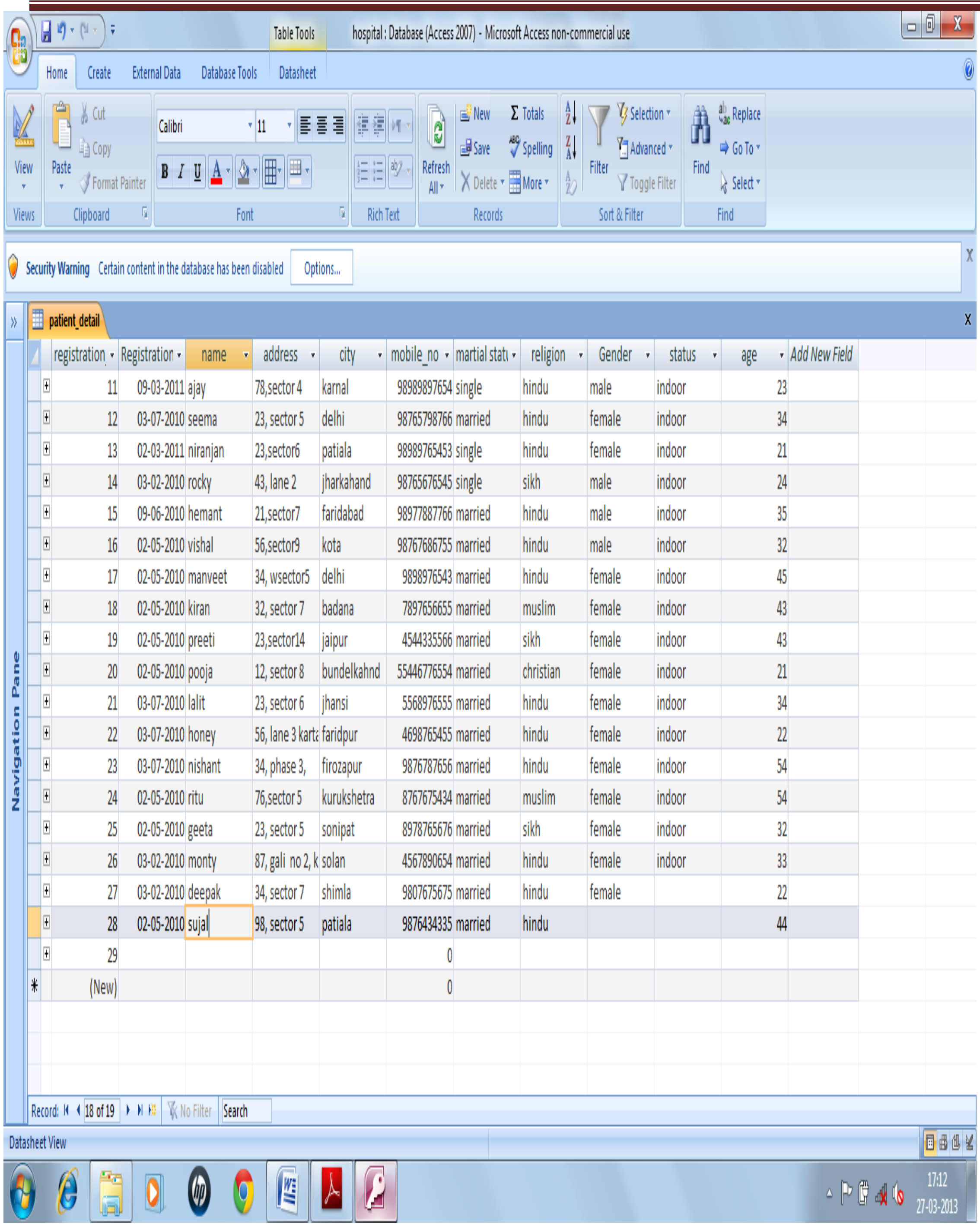


Figure 1: Hospital Database Schema



Security Warning Certain content in the database has been disabled Options...

patient_detail

registration	Registration	name	address	city	mobile_no	martial stat	religion	Gender	status	age	Add New Field
11	09-03-2011	ajay	78,sector 4	karnal	98989897654	single	hindu	male	indoor	23	
12	03-07-2010	seema	23, sector 5	delhi	98765798766	married	hindu	female	indoor	34	
13	02-03-2011	niranjan	23,sector6	patiala	98989765453	single	hindu	female	indoor	21	
14	03-02-2010	rocky	43, lane 2	jharkahand	98765676545	single	sikh	male	indoor	24	
15	09-06-2010	hemant	21,sector7	faridabad	98977887766	married	hindu	male	indoor	35	
16	02-05-2010	vishal	56,sector9	kota	98767686755	married	hindu	male	indoor	32	
17	02-05-2010	manveet	34, wsector5	delhi	9898976543	married	hindu	female	indoor	45	
18	02-05-2010	kiran	32, sector 7	badana	7897656655	married	muslim	female	indoor	43	
19	02-05-2010	preeti	23,sector14	jaipur	4544335566	married	sikh	female	indoor	43	
20	02-05-2010	pooja	12, sector 8	bundelkahnd	55446776554	married	christian	female	indoor	21	
21	03-07-2010	lalit	23, sector 6	jhansi	5568976555	married	hindu	female	indoor	34	
22	03-07-2010	honey	56, lane 3 kart	faridpur	4698765455	married	hindu	female	indoor	22	
23	03-07-2010	nishant	34, phase 3,	firozapur	9876787656	married	hindu	female	indoor	54	
24	02-05-2010	ritu	76,sector 5	kurukshetra	8767675434	married	muslim	female	indoor	54	
25	02-05-2010	geeta	23, sector 5	sonipat	8978765676	married	sikh	female	indoor	32	
26	03-02-2010	monty	87, gali no 2, k	solan	4567890654	married	hindu	female	indoor	33	
27	03-02-2010	deepak	34, sector 7	shimla	9807675675	married	hindu	female		22	
28	02-05-2010	sujal	98, sector 5	patiala	9876434335	married	hindu			44	
29					0						
(New)					0						

Figure 2: Patient_detail table of Hospital database

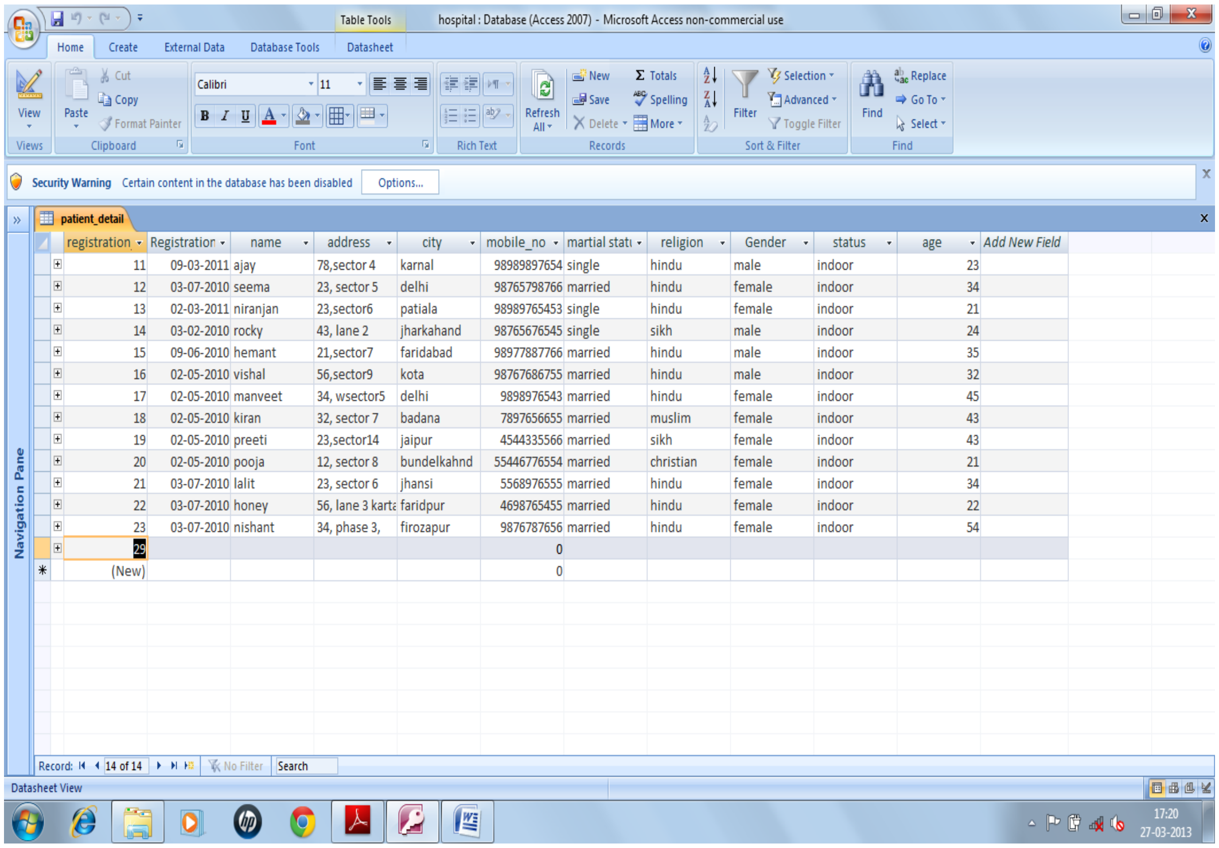
Relative density for the attributes of Patient_detail relation is described in table. Patient_detail table of hospital database involves computation of relative density conceptually, the scope in tables 1 range from 0 to 9 for the dimensions with numerical values, and from a to z for the dimensions with 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 in Age dimension is a percentage of ratios (0.088) on the number of all tuples (16/18) divided by normalization factor (10) which is used to normalize the dimensions with numerical values.

In this case, the density value of the tuples in the age dimension against the entire table is 88.88 %(8.888×10). If the density threshold variable τ coming from the user's input is 80%, age dimension is considered a candidate. Assume that the user inputs the age and mobile no dimensions as ICI and ICE, respectively.

Table 1: Relative density of Patient_detail table with Numerical value

Scope	Reg_no	Density	Reg_date	Density	Mobile no	Density	Age	Density
0			2	1.11			2	1.11
1	1	0.55	3	1.66				
2	2	1.11	1	0.55	4	2.22		
3	4	2.22	3	1.66	2	1.11		
4	1	0.55			4	2.22		
5	4	2.22	3	1.66				
6	1	0.55	2	1.11	2	1.11		
7	2	1.11			6	3.33		
8			4	2.22				
9	3	1.66					16	8.88

In table 1, the relative density of age dimension has the highest density value. However, this dimension for clustering is not considered, since age dimension is registered in ICE. And, the Reg_no dimension is excluded for clustering because it is a primary key. The age dimension is considered for clustering since it is specified in ICI. If no variable is specified in ICI, age dimension is selected because its relative density value 8.888, is the highest. Once age dimension is selected for clustering, the reduced table is produced containing only the tuples that belong to the corresponding range.


patient_detail

registration	Registration	name	address	city	mobile_no	martial statu	religion	Gender	status	age
11	09-03-2011	ajay	78,sector 4	karnal	98989897654	single	hindu	male	indoor	23
12	03-07-2010	seema	23, sector 5	delhi	98765798766	married	hindu	female	indoor	34
13	02-03-2011	niranjan	23,sector6	patiala	98989765453	single	hindu	female	indoor	21
14	03-02-2010	rocky	43, lane 2	jharkahand	98765676545	single	sikh	male	indoor	24
15	09-06-2010	hemant	21,sector7	faridabad	98977887766	married	hindu	male	indoor	35
16	02-05-2010	vishal	56,sector9	kota	98767686755	married	hindu	male	indoor	32
17	02-05-2010	manveet	34, wsector5	delhi	9898976543	married	hindu	female	indoor	45
18	02-05-2010	kiran	32, sector 7	badana	7897656655	married	muslim	female	indoor	43
19	02-05-2010	preeti	23,sector14	jaipur	4544335566	married	sikh	female	indoor	43
20	02-05-2010	pooja	12, sector 8	bundelkahnd	55446776554	married	christian	female	indoor	21
21	03-07-2010	lalit	23, sector 6	jhansi	5568976555	married	hindu	female	indoor	34
22	03-07-2010	honey	56, lane 3 karta	faridpur	4698765455	married	hindu	female	indoor	22
23	03-07-2010	nishant	34, phase 3,	firozapur	9876787656	married	hindu	female	indoor	54
29					0					
(New)					0					

Record: 14 of 14 No Filter Search

Figure 3: Reduced table for Patient_detail relation

The same method results in reduced tables for all relations in a data warehouse. In the third step of ADWRT, MVPP is established 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 Patients registration?

- Q2: What are the top 3 kinds of disease faced by patients?
- Q3: Which medicine has proven to be more beneficial for Cancer since last four years?
- Q4: Which area has more vulnerable to diseases since last two years?

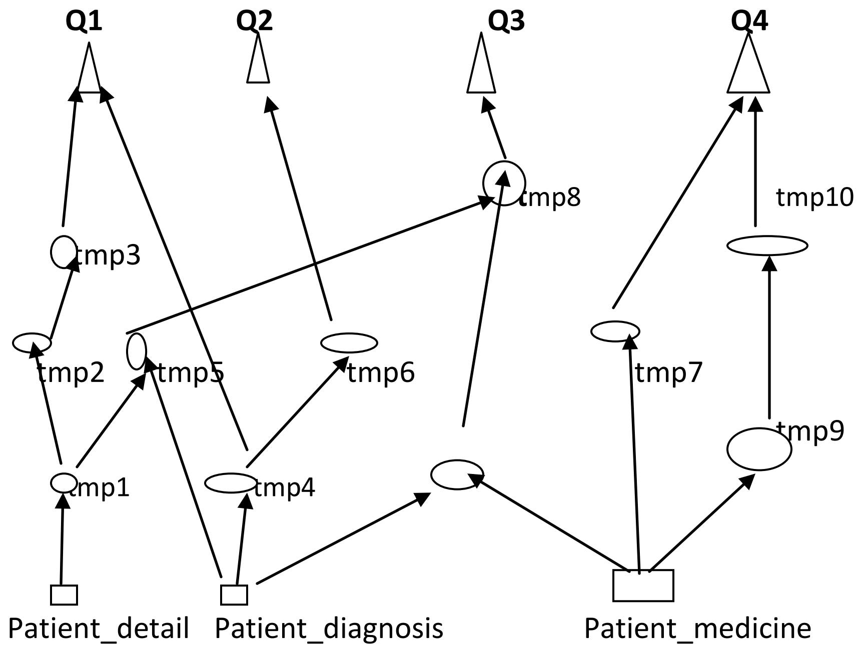


Figure 4: MPVV for 4 queries.

Figure 4 indicates □ a base relation, ○ is for intermediary value, is used for a query. Once an MVPP is established as shown in Fig.17, views to be materialized are selected considering cost. The base unit of cost estimation used in the paper is the number of tuples If tmp3 relation is selected as the materialized view, the total cost Ct is an addition of view processing time-cost Ca(12)(the number of tuples of tmp3(6) and tmp4(6), since only tmp3 and tmp△ are used to process Q1 and view maintenance cost Cm(2×56=112)(when tmp3 is stored as materialized view, maintenance cost of tmp3 is multiplied by 2, after addition of tmp3(6),tmp4(6),tmp1(15),tmp2(10),Patient_detail(9),Patient_diagnosis(10). Multiplication by 2 is due to the fact that if there is any update in tmp3, all the children of it (tmp1, tmp2) should be recomputed.

Total cost T is addition of the total cost of Q1, Q2, Q3, and Q4. In a similar manner, fill in table 7. When SC is given by 10 in the third step of ADWRT as shown in table 7, intermediary views tmp6, tmp5, tmp7, tmp8, and tmp9 are selected. In this case, the additional space needed for materialization is 8.The first column in tables 7 indicates the relations used in MVPP, the second column is the query frequency(fq), the third is the number of tuples(t#), and the fourth, fifth, and sixth are view processing time-cost(Ca), view maintenance(Cm), and total cost(Ct), respectively. The final column represents total cost (T) for all the queries.

Table 2: Cost computation for 4 queries with reduced tables

			Ca				Cm				Ct				**T**
	fq	T#	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
Patient_detail	1	9	42			54	0			0	42			54	96
Patient_diagnosis	3	10	64	42	68		0	0	0	64	0	42	68		174
Patient_medicine	3	8				50			20				30		10
tmp1	2	15	42				12	34	12		54	34	22		11
tmp2	1	10	12		10		14				26				26
tmp3	1	6	13	78		43					13	78		43	134
tmp4	2	6	45				55	38	18		10	83	18		20

tmp5	1	12	32	55	20						32	55	20		107
tmp6	1	10													
tmp7	1	15	12	44		24			48		60	44	48	24	176
tmp8	1	5													
tmp9	1	5	43	32		28	54	72	54		87	94	54	28	263

In Table 2, these cost estimation methods leave out some important factors in cost. In the ADWRT, cost for the select operation is supplemented to cost estimation formulation. Also, query frequency on all the tuples is imposed consisting of the query rather than the query itself because of considering the fact that the views consisting of the query can be used in another query.

4. Conclusion and Future Works

The proposed algorithm ADWRT, firstly finds high density clusters from the dimensions of the given tables, and secondly, produces the reduced tables using the previous 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, ADWRT is proposed in this paper, which adopts one of the data mining techniques (i.e., clustering method). In the proposed algorithm, the user can demonstrate 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. 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. ADWRT 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 .

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.