Design and Selection of Materialized Views in a Data Warehousing Environment: A Case Study

Goretti K.Y. Chan
Rix Pumps Limited
Computer & Information Systems,
Tai Po Industrial Estate, N.T.,
Hong Kong, China
g.chan@rix.com.hk

Qing Li
Dept of Computer Science
City University of Hong Kong
Tat Chee Ave, Kowloon,
Hong Kong, China
csqli@cityu.edu.hk

Ling Feng
Dept of Computing
Hong Kong Polytechnic University
Hung Hom, Kowloon,
Hong Kong, China
cslfeng@comp.polyu.edu.hk

ABSTRACT

In this paper, we describe the design of a data warehousing system for an engineering company 'R'. This system aims to assist users in retrieving data for business analysis in an efficient manner. The structural design of this data warehousing system employs the dimensional modeling concepts of star and snowflake schemes. Furthermore, frequently accessed dimension keys and attributes are stored in various summary views (materialized views) in order to minimize the query processing cost. A cost model was developed to enable the evaluation of the total cost and benefit involved in selecting each materialized view. Using the cost analysis methodology for evaluation, an adapted greedy algorithm has been implemented for the selection of materialized views. This algorithm takes into account all of the cost variables associated with the materialized views selection method, including query access frequencies, base-data update frequencies, query access costs, view maintenance costs and the availability of the system's storage. The algorithm and cost model have been applied to a set of real-life database items extracted from company 'R'. By selecting the most cost effective set of materialized summary views, the total cost of the maintenance, storage and query processing of the system is optimized, thereby resulting in an efficient data warehousing system

1. INTRODUCTION AND MOTIVATION

A data warehouse is an information base that stores a large volume of extracted and summarized data for On-Line Analytical Processing and Decision Support Systems [1]. The basic architecture of a data warehousing system given in [2] is shown in Figure 1. To reduce the cost of executing aggregate queries in a data warehousing environment, frequently used aggregates are often pre-computed and materialized into summary views so that future queries can utilize them directly. Undoubtedly, materializing these summary views can minimize query response time. However, if the source data changes frequently, keeping these materialized views updated will inevitably incur a high maintenance cost. Furthermore, for a system with limited storage space and/or with thousands of summary views, we may be able to materialize only a small fraction of the views. Therefore, a number of parameters, including *users' query frequencies, base relation update frequencies, query costs, view maintenance costs* and *the availability of the system's storage*, should be considered in order to select an optimal set of summary views to be materialized.

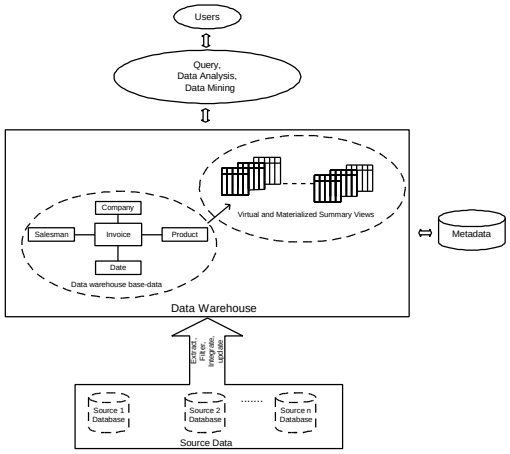

Figure 1: The basic architecture of a data warehousing system

To motivate the discussion of data warehouse design and materialized view selection, consider a data warehouse which contains the following fact and dimension tables:

INV (Co_no, Inv_no, Inv_date, P_no, Qty, Amt)

CO (Co_no, Co_name, R_no)

PD (P_no, P_name, Mfr_no, Type_no, Cat_no)

Assume the sizes of the fact and dimension tables 'INV', 'CO' and 'PD' are 114B, 12B and 6B, respectively, where B denotes the data block size which is 2K in the database system (e.g., Oracle). Given a subset of typical user's queries [3] and the query frequency between each update time interval. Then we can calculate the total cost C_{total} and each cost component (i.e. query processing, maintenance and storage costs) for the following three view materialization strategies:

- the *all-virtual-views* method
- the *all-materialized-views* method
- the *selected-materialized-views* method

Table 1 presents the calculation results, from which we make the following observations: (i) The *all-virtual-views* method requires the highest query processing cost but no view maintenance and storage costs are necessary. (ii) The *all-materialized-views* method can provide the best query performance since this method requires the minimum query processing cost. However, its total maintenance and storage expenses are the highest. (iii) The *selected-materialized-views* method requires a slightly higher query processing cost than the *all-materialized-views* method, but its total cost C_{total} is the least.

	Total query processing cost $Total(C_{qr})$	Total maintenance cost $Total(C_{mT})$	Total storage Cost $Total(C_{storeT})$	C_{total} = $Total(C_{qr})$ + $Total(C_{mT})$ + $Total(C_{storeT})$
All-virtual-views	10920	0	0	10920
All-materialized-views	949	2829	709	4487
Selected-materialized-views	1200	2184	240	3624

Table 1: The query, maintenance and storage costs for three view materialization strategies.

Based on the above cost analysis, apparently, the *selected-materialized-views* method is the most effective in terms of both query performance and maintenance cost of data warehousing systems.

Recently, materialized view selection problem has sparked ardent discussion in the database research community. Harinarayan, Rajaraman and Ullman [4] presented a greedy algorithm for the selection of materialized views so that query evaluation costs can be optimized in the special case of "data cubes". However, the costs for view maintenance and storage were not addressed in this piece of work. Yang, Karlapalem and Li [5] proposed a heuristic algorithm which utilizes a Multiple View Processing Plan (MVPP) to obtain an optimal materialized view selection, such that the best combination of good performance and low maintenance cost can be achieved. However, this algorithm did not consider the system storage constraints. Gupta [6] further developed a greedy algorithm to incorporate the maintenance cost and storage constraint in the selection of data warehouse materialized views. "And-Or" view graphs were introduced to represent all the possible ways to generate warehouse views such that the best query path can be utilized to optimize query response time.

In this paper, we discuss our experiences in designing and selecting appropriate materialized views for data warehousing systems. In our case study, the structural design of this data warehousing system employs the dimensional modeling concepts of star and snowflake schemes as presented in [3]. The greedy algorithm presented by Gupta [6] has been adopted and modified for the selection of materialized views. A cost model was developed to enable the evaluation of the total costs and benefits involved in selecting each materialized view. We applied the algorithm and cost model to a set of real-life database items extracted from this company. Based on the cost analysis, a set of materialized views are selected to optimize the total cost (i.e. the query, maintenance and storage costs), so that the best combination of good performance and low maintenance cost can be achieved. Various view materialization strategies are analyzed and their performances are tested [7].

The remainder of the paper is organized as follows. The cost model and adapted greedy algorithm for the selection of materialized views are presented in section 2. Guidelines for the design and selection of materialized views for data warehousing systems are discussed in section 3. Section 4 concludes the paper with a brief discussion of future work.

2. Materialized Views Selection

We now move on to address the related issue of data warehouse design for our case study, namely, the selection of summary views to be stored/materialized in the data warehouse. Benefits of materializing summary views selectively have been articulated in the literature [6, 8]. For our case study, a cost model is established to enable the evaluation of query cost, maintenance cost, storage cost and benefits (i.e. savings in overall query costs) associated with materializing each summary view in the data warehouse. An adapted greedy algorithm using the cost analysis methodology for evaluation is then presented for selecting an optimal set of materialized views.

2.1 Cost model

The estimated query, maintenance and storage costs in the following descriptions will be calculated in terms of data block

size B. For simplicity, other factors such as the computational cost and communication cost are ignored in our estimation. The detailed explanation of the cost calculation is presented [3].

2.1.1 Query processing cost for selection, aggregation and joining

The analysis assumes that there is no index or hash key in any of the summary views, therefore linear search and nested loop approach are used for the selection and join operations, respectively.

The total query cost *Total(C_{qr})* for processing *r* user's queries between each update time interval is:

$$Total(C_{qr}) = \sum_{i=1}^{r} f_{qi} * C_q(q_i)$$

2.1.2 Data warehouse maintenance cost

Assume that re-computation of each summary view V_i requires selection, aggregation and joining of its ancestor view V_{ai} with n dimension tables. If there are j summary views in the warehouse which are materialized, the total maintenance cost '*Total(C_m)*' for these materialized views is then:

$$Total(C_m) = \sum_{i=1}^{j} f_{ui} * C_m(V_i)$$

(f_{ui} = 1 in our case study, since we assume that all sales summary views are updated once within a fixed time interval.)

2.1.3 Storage cost

Storage cost of summary view V_i in terms of data block B is:

$$C_{store}(V_i) = S(V_i)$$

2.1.4 The net benefit and cost effectiveness

In order to determine the set of optimal materialized summary views, the net benefit 'Net(B_i)' and the storage effectiveness 'h_i' (i.e. the net benefit per unit of storage space occupied by a materialized view) associated with each summary view have to be calculated, as follows:

Storage effectiveness of each summary view V_i is calculated as follows:

$$Net(B_i) = \{\sum_{n=1}^{m} f_q(V_{ni}) * [C_t(V_{ni} \leftarrow V_{ai}) - C_t(V_{ni} \leftarrow V_i)]\} - C_m(V_i) - C_{store}(V_i)$$

$$h_i = Net(B_i) / S(V_i)$$

The storage effectiveness h_i, net benefit $Net(B_i)$, storage cost $C_{store}(V_i)$, maintenance cost $C_m(V_i)$, query frequencies f_{qi} and the total cost C_{total} of summary views V_i are calculated and listed in [3].

2.2 Adapted greedy algorithm for materialized summary view selection

Let *T* be the set of all sales summary views grouped by various dimension key attributes. Based on the greedy algorithm of [6], we develop an adapted greedy algorithm for determining the optimal set of materialized summary views *L*, a subset of *T*, such that the total cost C_{total} is minimized. The algorithm is based on the cost model presented in section 2.1.

Materialized views selection algorithm:

1. Determine the optimum query and maintenance paths for computing all summary views in the data warehouse;

2. Calculate the $Net(B_i)$ and h_i of each summary view in the query paths. Let *T* be the number of summary views possibly chosen as materialized views.

for *i =1 to T* ***do*** Calculate the $Net(B_i)$ of each summary view

$$Net(B_i) = \{\sum_{n=1}^{m} f_q(V_{ni}) * [C_t(V_{ni} \leftarrow V_{ai}) - C_t(V_{ni} \leftarrow V_i)]\} - C_m(V_i) - C_{store}(V_i)$$

V_i:

Storage effectiveness of summary views:

$h_i = Net(B_i) / S(V_i);$

3. List summary views in descending order according to the value of their storage effectiveness such that those views with the best storage effectiveness will be chosen first;

4. Calculate the C_{total} for each view :

$i = 1;$

$C_{total} = Total(C_{qall}) - Net(Bi);$

for *i = 2 to T* ***do*** $C_{total} = C_{total} - Net(Bi);$

find the $Min(C_{total})$ as the optimal cost for materialized view selection;

5. Select the best materialized view set *L*

$i = 1;$

$C_{total} = Total(C_{qall}) - Net(B_i);$

while $C_{total} > Min(C_{total})$

$i = i + 1;$

while $S(L) < S$

Select V_i from the summary view set $T\text{-}L$ with the highest storage effectiveness;

$S(L) = S(L) + S(V_i);$

endwhile

$C_{total} = C_{total} - Net(B_i);$

endwhile

return L.

The set of optimal materialized view L thus chosen is shown in [3].

2.3 Cost analysis

The summary views to be materialized are sorted in descending order according to the corresponding storage effectiveness 'h_i' listed in [3]. The top thirty-four summary views listed in this table are the set of optimal materialized views L. The total cost C_{total}, and its cost components versus storage size of the materialized views are plotted in Figure 2.

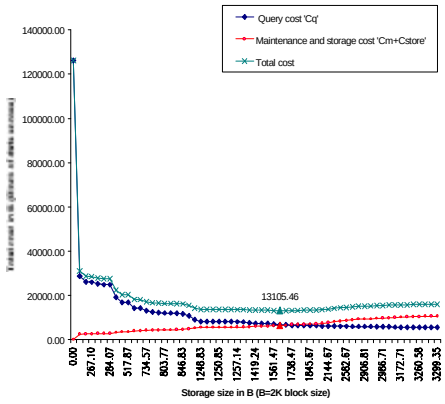

Figure 2: Total costs C_{total}, total query processing cost and the sum of maintenance and storage costs vs. storage size of the materialized views.

We observe that the C_{total} is dominated by the $Total(C_{qr})$ before reaching the optimum point. This optimal point occurs at a cost of 13105.46B and is designated as the minimum total cost $Min(C_{total})$. The $Total(C_{qr})$ drops drastically after materializing the first summary view 'CO-P-DAY', reducing by more than 75% while utilizing only 15% of the total storage space required by the set of optimal materialized views L. Therefore, materializing summary view 'CO-P-DAY' is very cost effective for improving the query performance of the data warehouse. After this first view has been chosen, there is little reduction in the $Total(C_{qr})$ when more summary views are materialized.

The sum of total maintenance and storage costs, $C_m(V_i)+C_{store}(V_i)$, increases linearly as the number of materialized summary views increases. However, its magnitude is relatively small compared with the $Total(C_{qr})$ before reaching the optimum point $Min(C_{total})$. After reaching this optimal point, C_{total} is dominated by the sum $C_m(V_i)+C_{store}(V_i)$. This is because materializing additional summary views (i.e. summary views with negative net benefit $Net(B_i)$) beyond the optimal point $Min(C_{total})$ cannot reduce query cost, but increases the storage and maintenance costs. Therefore, it is not cost effective to materialize additional views after reaching $Min(C_{total})$.

If all the summary views of the data warehouse are materialized, query performance can be optimized. However, this method requires the highest maintenance and storage cost. For a data warehouse with limited hard disk storage space and small maintenance window, materializing a few summary views which have the greatest storage effectiveness h_i (i.e. 'CO-P-DAY' for this case study) can effectively reduce query response time since they yield the greatest benefit yet require the least amount of storage space and maintenance costs. In the situation of a data warehouse which can be taken off-line for view maintenance and can have very large disk space available for the storage of materialized views, storing the set of optimal materialized views L can minimize query and maintenance cost while achieving good query performance.

3. Guidelines for warehouse schema design and materialized views selection

Our experiences gained from this case study can be summarized into the following guidelines for both data warehouse design and materialized view selection.

On Data Warehouse Design

i. Use the smallest size of integer or numerical values for the key attributes in dimension tables to minimize storage space and query processing time.

ii. Normalize dimension tables with large amount of records and hierarchy levels to achieve smaller dimension tables. Thus, the storage size and joining cost can be reduced substantially

iii. Denormalize dimension tables with relatively few records and attributes to minimize the number of joins required.

iv. Horizontally partition the fact table, which has a lot of records, into smaller summary views according to its dimension key attributes so as to improve query performance, and further enable users to select various summary views for materialization based on the query access frequency.

v. Store foreign keys of dimension tables in the summary views, especially those dimension tables that are frequently accessed to help improve the query performance. Furthermore, data in these summary views can also be easily used by other queries.

vi. Store frequently accessed dimension attributes (e.g. Co_name and P_name in our case study) in the summary views, especially for the dimension tables which have very many records, so as to minimize the number of joins and query processing costs.

On Materialized Views Selection

i. Materialize summary views that are frequently accessed by users .

ii. Materialize those commonly shared views which are used for generating other summary views.

iii. Materialize those views whose sizes have been substantially reduced from their ancestor's views.

When the storage factor is very small (i.e. a large amount of disk storage is available), materializing a set of optimal materialized views 'L' by the selection method as illustrated in Section 2.3 can achieve the best combination of good query performance and low maintenance cost.

4. Conclusions

In this case study, methods for designing an efficient data warehousing system based on the application requirements of an engineering company 'R' have been investigated. A hybrid schema was designed for this data warehouse by applying dimensional modeling concepts. A cost model was developed to calculate the costs and benefits associated with materializing each data warehouse view. The total cost under five test conditions, composed of different query patterns and frequencies, were evaluated for three different view materialization strategies: 1) *all-virtual-views* method, 2) *all-materialized-views* method, and 3) *selected-materialized-views* method. The total cost evaluated from using the *selected-materialized-views* method was proved to be the smallest among the three strategies in all cases. Further, an experiment was conducted to record different execution times of the three strategies in the computation of a fixed number of queries and maintenance processes. Again, the *selected-materialized-views* method requires the shortest total processing time.

An adapted greedy algorithm using the cost analysis methodology for evaluation was developed for materialized views selection. This view selection methodology was tested both analytically and experimentally and proved to be very cost effective for the optimization of the data warehouse. General guidelines for data warehouse design and materialized views selection based on this work are presented and a prototype of the data warehouse system was implemented using a commercially available data warehousing software "Oracle-Discoverer" [9, 10].

The cost evaluation methodology and views selection algorithm developed in this case study will be applied in the implementation of other data warehousing applications, such as inventory, production and purchasing analyses, etc. In addition, warehouse view self-maintenance methods [11, 12] other than the view re-calculation method adopted by this work will also be investigated, so as to further reduce system maintenance cost and achieve data warehouse optimization.

5. References

[1] S. Chaudhuri and U. Dayal. "An Overview of Data Warehousing and OLAP Technology". SIGMOD Record, 26(1):65-74, 1997.

[2] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, Y.

Zhuge. "TheStanford Data Warehousing Project". IEEE Data Engineering Bulletin, June 1995.

[3] G.Chan, Qing Li, Ling Feng. Design and selection of materialized views in a data warehousing environment: A case study. 1999. Http://www.cs.cityu.edu.hk/~csqli/papers /DOLAP99.ps.gz.

[4] V. Harinarayan, A. Rajaraman, and J. Ullman. "Implementing data cubes efficiently". Proceedings of ACM SIGMOD 1996 International Conference on Management of Data, Montreal, Canada, June 1996, pages 205--216.

[5] J.Yang, K. Karlapalem, and Q. Li. "A framework for designing materialized views in data warehousing environment". Technical Report HKUST-cs96-35, 1996. IEEE Int'l conference on Distributed Computing Systems (ICDCS '97), Maryland, U.S.A., May 1997.

[6] H. Gupta. "Selection of Views to Materialize in a Data Warehouse". Proceedings of 23rd VLDB Conference, Athens, Greece 1997.

[7] G.Chan. A case study for the design and selection of materialized views in a data warehousing environment. MSc Dissertation, The Hong Kong Polytechnic University, Hong Kong, 1998.

[8] J.Yang, K. Karlapalem, and Q. Li. "Algorithms for Materialized View Design in Data Warehousing Environment". Proceedings of 23rd VLDB Conference, (Athens), Greece 1997, P.136-145.

[9] Oracle Discoverer 3.0 User's Guide, Oracle.

[10] Oracle Discoverer 3.0 Administration Guide, Oracle.

[11] N. Huyn. "Efficient View Self-Maintenance". Proceeding of ACM Workshop, Montreal, Canada. 1996.

[12] D. Quass, A. Gupta, I.S. Mumick, J. Widom. "Making Views Self Maintainable for Data Warehouse". In PDIS, 1996.