International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 2, Number 2 (2012), pp. 63-70

http://www. ripublication.com

Materialized Views for Data Selection in Dataware House

[1]P.R. Vishwanath, [2]Dr. Rajylaxmi and [3]Dr. M. Sreedharreddy

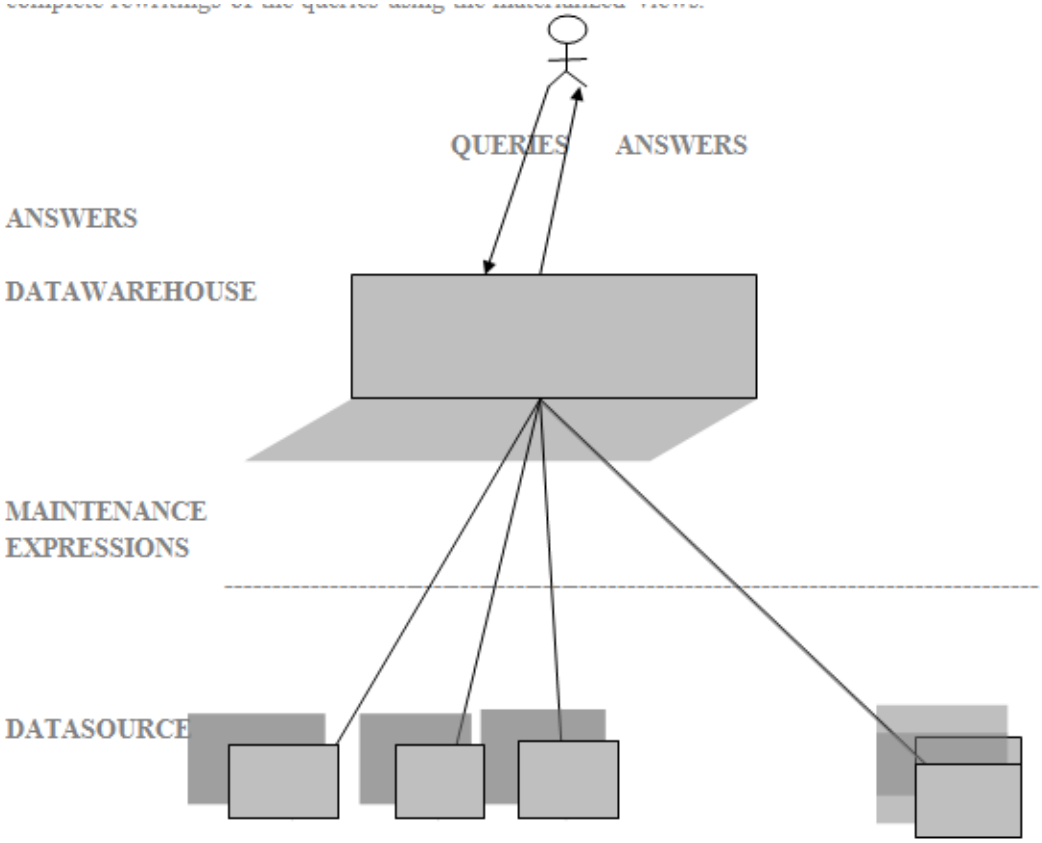
[1]Assoc. Professor, R.I.T.S., , Hyderabad, A.P., India
[2]HOD IT, G.I.T..A.M., India
[3]HOD IT, G.I.T.S., Hyderabad, A.P., India

Introduction

A data warehouse (DW) is a repository of information retrieved from multiple, possibly heterogeneous, autonomous, distributed database and other information sources for the purpose of complex querying, analysis, and decision support. Data in the DW are selectively collected from the sources, processed in order to resolve inconsistencies, and integrated advance (at design time) before data loading. DW data are usually organized multi-dimensionally to support on line analytical processing (OLAP). A DW can be seen abstractly as a set of materialized views defined over the source relations. During the initial design of a DW, the DW designer faces the problem of deciding which views to materialize in the DW. This problem has been addressed in the literature for different classes of queries and views, and with different design goals.

Back Ground

Figure 1 shows a simplified DW architecture. The DW contains a set of materialized views. The users address their queries to the DW. The materialized views are used partially or completely for the evaluation of the user queries. This is achieved through partial or complete rewritings of the queries using the materialized views.



Data source

When the source relations change, the materialized views need to be updated. The materialized views are usually maintained using an incremental strategy. In such a strategy, the changes to the sources relations are propagated to the DW. The changes to the materialized views are computed using the changes of the source relations and are eventually applied to the materialized views. The expressions used to compute the view changes involve the changes of the source relations and are called maintenance expressions. Maintenance expressions are issued by the DW against the data sources, and the answers are sent back to the DW. When the source relation changes affect more than one materialized view, multiple maintenance expressions need to be evaluated. The techniques of multi-query optimization can be used to detect common sub expressions among maintenance expressions in order to derive an efficient global evaluation plan for all the Maintenance expressions.

Main Thrust

When the selecting views to materialize in a DW, one attempts to satisfy one or more design goals. A design goal is either the minimization of a cost function or a constraint. A constraint can be classified as user-oriented or system oriented. Attempting to satisfy the constraints can result in no feasible solution to the view

selection problem. The design goals determine the design of the algorithms that select views to materialize from the space of alternative view sets.

Minimization of Cost Functions

Most approaches comprise in their design goals the minimization of a cost function.

- **Query Evaluation Cost :** Often, the queries that the DW has to satisfy are given as input to the view selection problem. The overall query evaluation… cost is the sum of the cost evaluating each input query rewritten (partially or completely) over the materialized views. This sum also can be weighted, each weight indicating the frequency or importance of the corresponding query. Several approaches aim at minimizing the query evaluation cost (Gupta & Mumick, 1999; Harinarayan et al, 1996; Shukla et al, 1998)
- **View Maintenance Cost :** The view maintenance cost is the sum of the cost of propagating each source relation change to the materialized views. This sum can be weighted, each weight indicating the frequency of propagation of the changes of the corresponding source relation. The maintenance expressions can be evaluated more efficiently if they can be partially rewritten over views already materialized at the DW; the evaluation of parts of the maintenance expression is avoided since their materializations are present at the DW. Moreover, access of the remote data sources and expensive data transmissions are reduced. Materialized views that are added to the DW for reducing the view maintenance cost are called auxiliary views (Ross et al., 1996; Theodoratos & Sells, 1999). Obviously, maintaining the auxiliary views incurs additional maintenance cost. However, if this cost is less than the reduction to the maintenance cost of the initially materialized views, it is worth keeping the auxiliary views in the DW. Ross, et al. (1996) derive auxiliary views to materialize in order to minimize the view maintenance cost.
- **Operational Cost:** Minimizing the query evaluation cost and the view maintenance cost are conflicting requirements. Low view maintenance cost can be obtained by replicating source relations at the DW. In this case, though, the query evaluation cost is high, since queries need to be computed from the replicas of the source relations. Low query evaluation cost can be obtained by materializing at the DW all the input queries. In this case, all the input queries can be answered by a simple lookup, but the view maintenance cost is high, since complex maintenance expressions over the source relations need to be computed. The input queries may overlap; that is, they may share many common subexpressions. By materializing common subexpressions and other views over the source relations, it is possible, in general, to reduce the view maintenance cost. These savings must be balanced against higher query evaluation cost. For this reason, one can choose to minimize a linear combination of the query evaluation an view maintenance cost, which is called operational cost. Most approaches endeavor to minimize the operational cost (Baralis et al, 1997; Gupta, 1997; Theodoratos & Sellis, 1999; Yang et al, 1997)

System-Oriented Constraints

System-oriented constraints are dictated by the restrictions of the system and are transparent to the users

- **Space Constraint:** Although the degradation of the cost of disk space allows for massive storage of data, one cannot consider that the disk space is unlimited. The space constraint restricts the space occupied by the selected materialized views not to exceed the space allocated to the DW for this end. Space constraints are adopted in many works (Gupta, 1997l Golfarelli & Rizzi, 2000; Harinarayan et al, 1996, Theodoratos & Sellis, 1999).
- **View Maintenance Cost Constraint:** In many practical cases, the refraining factor in materializing all the views in the DW is not the space constraint but the view maintenance cost. Usually, DWs are updated periodically (e.g. at nighttime) in a large batch update transaction. Therefore, the update windows must be sufficiently short so that the DW is available for querying and analysis during the daytime. The view maintenance cost constraint states that the total view maintenance cost should be less than a give amount of view maintenance time. Gupta and Mumick (1999), Golfareli and Rizzi (2000), and Lee and Hammer (2001) consider a view maintenance cost constraint in selecting materialized views.
- **Self Maintainability:** A materialized views is self maintainable if it can be maintained for any instance of the source relations over which it is defined and for all source relation changes, using only these changes, the view definition, and the view materialization. The notion is extended to a set of views in a straightforward manner. By adding auxiliary views to a set of materialized views, one can make the whole view set self-maintainable. There are different reasons for making a view set self-maintainable: (a) the remote source relations need not be contacted in turn for evaluating maintenance expressions during view updating ; (b) anomalies due to concurrent changes are eliminated, and the view maintenance process is simplified; (c) the materialized views can be maintained efficiently even if the sources are not able to answer queries (e.g. legacy systems), or if they are temporarily unavailable (e.g., in mobile systems) . By adding auxiliary views to a set of materialized views, the whole view set can be made self-maintainable, Self-maintainability can be trivially achieved by replicating at the DW all the source relations used in the view definitions. Self maintainability viewed as a constraint requires that the set of materialized views taken together is self maintainable. Quass, et al., (1996), Akinde, et al., (1998), Liang, et al., (1999) and Theodoratos (2000) aim at making the DW self-maintainable.
- **Answering the Input Queries using Exclusively the Materialized Views:** This constraint requires the existence of a complete rewriting of the input queries, initially defined over the source relations, over the materialized views. Clearly, if this constraint is satisfied, the remote data sources need not be contacted for evaluating queries. This way, expensive data transmissions from the DW to the sources, and conversely, are avoided. Some approaches assume a centralized DW environment, where the source relations are present at the

DW site. In this case, the answerability of the queries also can be trivially guaranteed by appropriately defining select-project views on the source relations and replicating them at the DW. This approach assures also the self-maintainability of the materialized views. Theodoratos and Sellis (1999) do not assume a centralized DW environment or replication of part of the source relations at the DW and explicitly impose this constraint in selecting views for materialization.

User-Oriented Constraints

User-oriented constraints express requirements of the users.

- **Answer Data Currency Constraints :** An answer data currency constraint sets an upper bound on the time elapsed between the point in time the most recent changes of a source relation that are taken into account in the computation of this answer are read (this time reflects the currency of answer data). Currency constraints are associated with every source relation in the definition of every input query. The upper bound in an answer data currency constraint (minimal currency required) is set by the users according to their needs. This formalization of data currency constraints allows stating currency constraints at the query level and not at the materialized view level, as is the case in some approaches. Therefore, currency constraints can be exploited by DW view selection algorithms, where the queries are the input, while the materialized views are the output (and, therefore, are not available). Furthermore, it allows stating different currency constraints for different relations in the same query.
- **Query Response Time Constraints:** A query response time constraint states that the time needed to evaluate an input query using the views materialized at the DW should not exceed a given bound. The bound for each query is given by the users and reflects their needs for fast answers. For some queries, fast answers may be required, while for others, the response time may not be predominant.

Search Space and Algorithms

Solving the problem of selecting views for materialization involves addressing two main tasks: (a) generating a search space of alternative view sets for materialization and (b) designing optimization algorithms that select an optimal or near-optimal view set from the search space. A DW is usually organized according to a star scheme where a fact table is surrounded by a number of dimension tables. The dimension tables the fact table and the aggregation levels. Typical OLAP queries involve star joins (key/foreign key joins between the fact table and the dimension tables) and grouping and aggregation at different levels of granularity. For queries of this type, the search space can be formed in an elegant way as a multidimensional lattice (Baralis et al., 1997l Hiarinarayan et al, 1996), Gupta (1997) states that the view selection problem is NP-hard. Most of the approaches on view selection problems

avoid exhaustive algorithms. The adopted algorithms fall into two categories; deterministic and randomized. In the first category belong greedy algorithms with performance guarantee (Gupta 1997; Harinarayan et al, 1996) 0-1 integer programming algorithms (Yang et al., 1997), A * algorithms (Gupta & Mumick, 1999), and various other heuristic algorithms (Baralis et al., 1997; Ross et al., 1996; Shukla et al., 1998; Theodoratos & Sellis, 1999). In the second category belong simulated annealing algorithms (Kalnis et al., 2002; Theodorates et al, 2001), iterative improvement algorithms (Kalnis et al, 2002) and genetic algorithms (Lee & Hammer, 2001). Both categories of algorithms exploit the particularities of the specific view selection problem and the restrictions of the class of queries considered.

Further Research

The view selection problem has been addressed for different types of queries. Research has focused mainly on queries over star schemes. Newer applications (e.g. XML or Web-based applications) require different types of queries. This topic has only been partially investigated (Golfarelli et al, 2001; Labrindis & Roussopoulos, 2000). A relevant issue that needs further investigation is the construction of the search space of alternative view sets for materialization. Even though the construction of such a search for grouping and aggregation queries is straightforward (Harinarayan et al., 1996), it becomes an intricate problem for general queries (Golfarelli & Rizzi, 2001). Indexes can be seen as special types of views. Gupta et al (1997) show that a two-step process that divides the space available for materialization and picks views first and then indexes can perform very poorly. More work needs to be done on the problem of automating the selection of views and indexes together. DWs are dynamic entities that evolve continuously over time. As time passes, new queries need to be satisfied. A dynamic version of the view selection problem chooses additional views for materialization and avoids the design of the DW from scractch (Theodoratos & Sellis, 2000). A system that dynamically materializes views in the DW at multiple levels of granularity in order to match the workload (Kotidis & Roussopoulos, 2001) is a currnt trend in the design of a DW.

Conclusion

A DW can be seen as a set of materialized views. A central problem in the design of a DW is the selection of views to materialize in it. Depending on the requirements of the prospective users of the DW, the materialized view selection problem can be formulated with various design goals that comprise the minimization of cost functions and the satisfaction of user-and system-oriented constraints. Because of its importance, different versions of it have been the focus of attention of many researchers in recent years. Papers in the literature deal mainly with the issue of determining a search space of alternative view sets for materialization and with the issue of designing optimization algorithms that avoid examining exhaustively the usually huge search space. Some results of this research have been used already in commercial database management systems (Agrawal et al., 2000)

References

[1] Agrawal, S., Chaudhuri, S., & Narasayya, V.R. (2000). Automated selection of materialized views and indexes in SQL databases. *International Conference on Very Large Data Bases (VLDB),* Cairo, Egypt.

[2] Akinde, M.O., Jensen, O.G. & Bohlen, H.M (1998). Minimizing detail data in data warehouse. *International Conference on Extending Database Technology (EDBT),* Valencia, Spain.

[3] Baralis, E., Paraboschi, S., & Teninente. E (1997). Materialized views selection in a multidimensional database. *International Conference on Very Large data Bases,* Athens Greece.

[4] Golfarelli, M., & Rizzi, S (2000). View materialization for neated GPSJ queries. *International Workship on Design and Management of data Warehouses (DMDW),* Stockholm, Sweden.

[5] Golfarelli, M., Rizzi, S., & Vrdoljak B. (2001). Data werehouse design from XML sources, *ACM International Workshop on Data Warehousing and OLAP (DOLAP).* Atlanta, Georgia.

[6] Gupta, H (1997), Selection of views to materialize in a data warehouse. *International Conference on Database Theory (ICDT),* Delphi, Greece.

[7] Gupta, H., Harinarayan, V., Rajaraman, A., & Ullman, J.D. (1997), Index selection for OLAP, m *IEEE International Conference on Data Engineering,* Birmingham, UK

[8] Gupta., H., & Mumick, I.S (1999). Selection of views to materialized under a maintenance cost constraint. *International Conference on Database Theory (ICDT).* Jerusalem, Israel.

[9] Harinarayan, V., Rajaraman, A., & Ullman, J.D. (1996), Implementing data cubes efficiently, *ACM, SIGMOD International Conference on Management of Data (SIGMOD),* Montreal, Canada

[10] Kalnis, P., Mamoulis, N., & Papadias, D. (2002), View selection using randomized search. *Data & Knowledge Engineering, 42 (1),* 89-111, Kotidis, Y., & Roussopoulos, N (2001), A Case for dynamic. View Management . *ACM Transactions on Database Systems, 26(4), 388-423.*

[11] Labrinidis, A., & Roussopoulos, N. (2000). Web View materialization. *ACM SIGMOD International Conference on Management of Data (SIGMOD),* Dallas, Texes.

[12] Lee, M. & Hammer, J. (2001). Speeding up materialized view selection in data warehouses using a randomized algorithm. *International Journal of Cooperative Information system (IJCIS), 10(3),* 327-353

[13] Liang, W (1999), Making multiple views self-maintainable in a data warehouse. *Data & Knowledge Engineering. 30(2),* 121-134

[14] Quass, D., Gupta, A., Mumick, I.S., & Widom, J. (1996), Making views self-maintainable for data warehousing. *International Conference on Parallel and Distributed Information Systems (PDIS),* Florida.

[15] Ross, K., Srinivasa, D., & Sudarshan S (1996). Materialized view maintenance and integrity constraint checking. Trading space for time. *ACM SIGMOD*

International Conference on Management of Data (SIGMOD), Montreal, Canada

[16] Shukla ., A – Deshpande, P., & Naughton, J (1998), Materialized view selection for multidimensional datasets. *International Conference on Very Large Data Bases (VLDB),* New York

[17] Theodoratos, D. (2000), Complex view selection for data warehouse self-maintainability, *International Conference on Cooperative Information Systems (Coop IS),* Eilat, Israel.