FASTER QUERY RESULT RETRIEVAL APPROACHES FROM A DATA WAREHOUSE: A SURVEY

Sonali Chakraborty[1], Dr. Jyotika Doshi[2]
[1]Gujarat University, [2]GLS University

Abstract
Any business organization has different business processes with different requirements and levels in an organization. Because of the performance driven requirement in case of OLTP (Online Transaction Processing) system, there is a need for a data warehouse for decision making. Data extraction from a data warehouse is a critical aspect as system may take long time to run or sometimes the data may not be available for longer period. Probability that same query is fired many times is also high and each time same query is fired all data warehouse data is analyzed and mined. This paper provides an overview of different approaches, proposed or implemented for faster retrieval of query results from a data warehouse.

Index Terms: **Approaches, Data warehouse, faster execution time, Query Result Retrieval**

I. INTRODUCTION

A. Requirements of a Business organization

Any business organization has different business processes and there are different requirements and levels in an organization. Single system cannot provide all the information required by an organization. Operational managers require systems that will keep track of the elementary activities i.e. Transaction Processing Systems (TPS), which is computerized system providing information. It performs and records daily routine transactions necessary for conducting business, answers routine questions and tracks the flow of transactions [1]. Operational database systems perform online transaction and query processing using Online Transaction Processing (OLTP) Systems. They are relational database systems where performance is an important factor as they are used to support the users [2]. The transaction data from TPS are summarized and reported by middle managers through Management Information System (MIS). This information is used for monitoring and controlling the business and predicts future performance. For non- routine decisions for middle management and for focusing on unique and rapidly changing problems Decision Support System (DSS) is used [1]. Because of the performance driven requirement for OLTP system, there is a need for a separate database for decision making i.e. a data warehouse, which is long term storage. [2].

B. About Data Warehouse

"A data warehouse is an integrated subject oriented and time variant repository of information in support of management's decision making process". It gives the facility to shred data load from OLTP systems which the OLTP systems no longer requires. Data warehouse is also useful as heterogeneous database integration i.e. organizations collect diverse data and maintain large databases from multiple, heterogeneous distributed information systems and store in data warehouse for querying and analysis. In case of unavailability of data warehouse, historical data can be downloaded to CDs or tapes but it may not be available for online queries. For constructing a data warehouse, data cleaning, data integration and data consolidation is required. [2,3]. ETL (Extraction, Transformation, Loading) process is considered as a backbone of data warehouse architecture where newly inserted, updated, and deleted information is extracted from the sources, propagated in Data Staging Area where their transformation, homogenization, and cleansing is done. During transformation filters and checks are done for ensuring that the data transferred to the warehouse follow the business rules and integrity constraints. Data is then loaded in data warehouse. In case of traditional data warehouse, ETL process periodically refreshes the data warehouse during low-load duration or when it is idle, periods of its operation (e.g., every night) and it has a specific time duration to complete. In case of business necessities and demands near real-time data warehouse refreshment is required [24].

Data warehouse queries are complex since they involve computation of data groups and hence data extraction from a data warehouse is critical aspect since the system may take long time to run or the data may not be available for longer period [3, 25]. The quality of data provided to the decision makers depends on the capability of the data warehouse system to convey in a reasonable time, from the sources to the data marts, the changes made at the data sources. Most of the design decisions are then concerned by the choice of data structures and update techniques that optimize the refreshment of the data warehouse [5].

C. OLAP Queries

Data warehouse serves users or knowledge workers i.e. managers, analysts, executives for data analysis and decision making using Online Analytical Processing (OLAP) systems. OLAP queries need read-only access of the data for performing summarization and aggregation. OLAP queries in operational databases would substantially degrade the performance of operational tasks [3]. Primary concern in case of ad hoc queries for a large data warehouse is that, query performance degrades since they have to go through large volumes of data after making multiple joins. Probability that same query is fired many times is also high and hence each time same query is fired all data warehouse data is analyzed [4]. In the current literature, different approaches proposed or implemented for query result retrieval from a data warehouse have been summarized.

II. RELATED LITERATURE

Papers	*Issues dealt with*	*Brief Description*
Ashish Gupta et al. [6]	Classification of the view maintenance problem.	➢ Illustrated that views may also be maintained using the partial information in that view depending on the type of modification required i.e. insertion, deletion or update.
Ashish Gupta et al. [7]	Incremental Maintenance of Views	Proposed two algorithms ➢ **Counting** – in case of non-recursive views, it computes only the view tuples which are inserted or deleted at little or no cost. ➢ **DRed** - for recursive views, it first computes an overestimate of deleted derived tuples followed by pruning of overestimate. New tuples which are to be added are then computed through partially added materialized views and by the changes made to base relations.
Randall G. Bello et al. [8]	Oracle Materialized Views used for data warehousing	➢ Based on inner or outer equi-joins with aggregations that can be refreshed either on demand or periodically. ➢ Optimization in materialized views comprises of transparent query rewrites based on cost- based selection method. ➢ Ability to rewrite a large class of queries based on a small set of materialized views is supported by using Dimensions, losslessness of joins, functional dependency, column equivalence, join derivability, join back and aggregate rollup.

		➢ Explained the refresh algorithms for MJV (Materialized Join View), MAV (Materialized Aggregate View), and MV (Materialized views) with subqueries along with the concepts of dimension, query rewrite concepts, general rewrite algorithm and heuristic and cost based rewrite.
Dallan Quass [9]	Materialized views having aggregations	➢ Considers almost all SQL aggregate functions such as count, sum, avg, min, and max. ➢ Gives maintenance expression for insertions and deletions through an aggregate operator in a view definition in both cases i.e. in presence and absence of max and min aggregate functions.
Ashish Gupta et al. [10]	Model for data integration from multiple databases combined into an integrated view which is then materialized and stored in database.	➢ Algorithms for incremental maintenance of views using outer-join operator i.e. full outer-join, left and right outer-joins, natural full outer-joins. ➢ Depicted that outer-join views are usually self-maintainable whereas match views, i.e. full outer join views are always self-maintainable.
Kenneth A Ross et al. [11]	Incremental maintenance problem of an SQL view in case of database updates	➢ Depicted that it is possible to reduce the total time cost of view maintenance by materializing and maintaining additional views and formulated to determine the optimal set of additional views. ➢ Algorithm is implemented using DAG which has operation nodes containing operator and equivalence nodes having edges to operation nodes.
Jingren Zhou et al. [12]	View maintenance overhead issues	➢ Introduction of lazy view maintenance where updates need not maintain the views, instead store sufficient information such that affected views will be maintained later. ➢ Maintenance is done by the low-priority jobs during system free cycles. ➢ In case a query requires a view before update, then update is done transparently before query access and the first beneficiary bears the overhead.
Yue Zhuge et al. [13]	View maintenance as the data sources are updated.	➢ Introduced algorithm called “Eager Compensating Algorithm” (ECA) by including compensating query which offsets the effects of updates on result of unanswered queries. ➢ Also introduced two streamlined versions of the said algorithm for special cases of delete and updates i.e. ECA^K and ECA^L.

D. Agrawal et al. [14]	Incremental view maintenance	➢ The authors designed two algorithms i.e. SWEEP and NESTED SWEEP for incremental updates. ➢ SWEEP processes one update at a time on the data warehouse and then constructs the updated changes in the materialized view for that update. ➢ NESTED SWEEP algorithm computes view changes for multiple updates collectively and does not require absolute quiescence state but does requires a duration when interfering updates should subside for termination.
Surajit Chaudhuri et al. [15]	Query optimizing problem in the presence of materialized view.	The authors proposed three steps for optimization. ➢ In the first step the query is translated into unfolded form. ➢ In second step using one-level rule possible ways are identified in which one or more materialized views may be used to generate alternative formulations of the query. ➢ Finally costs of alternative formulations are generated and the execution plan with least cost is selected.
Dimitri Theodoratos et al. [16]	Materializing selected set of views to kept the total query processing cost and the view maintenance cost at an acceptable level.	➢ Modelling the problem using a state space search algorithm after representing the views using multiquery graphs where every state is a multiquery graph of the view that is materialized in the data warehouse.
Jonathan Goldstein et al. [17]	Materialized views in case of aggregate queries	➢ Algorithm to determine whether a part or all of a query can be computed from materialized view. ➢ Describes how they can be incorporated in transformation based optimizers which generates all possible rewritings of a query expression, estimating their costs, and choosing the one with the lowest cost.
Divesh Srivastava et al. [18]	Semantic approach to detect when the information in a view is sufficient to answer the query.	➢ If a query has grouping and aggregation but the view does not have, then a view is usable for answering the query only of there is similarity between view and portion of query. ➢ When the views have groupings and aggregations we need to identify the conditions under which the aggregation information in the view is sufficient to perform the aggregation computations required in the query. ➢ Takes into consideration that the rewritten query may be a union to single block queries i.e. queries and views of the form : SELECT, FROM, WHERE, GROUPBY,

		HAVING and SELECT and HAVING may contain aggregates such as MIN, MAX, SUM, COUNT.
Sirirut Vanichayobon [19]	Identification of factors to be considered for selecting a proper indexing technique for data warehouse applications.	➢ Explains different indexing techniques, i.e. B-Tree Index, Projection Index, Bitmap Index ➢ Summarizes the evaluation of these techniques in a tabular form. ➢ Based on evaluation, the author concluded the following: a) B-Tree should only be used in case of high cardinality data and predicted queries. b) Bitmap Indexes plays a key role in case of data warehouse queries since they have an ability to perform operations on index level before retrieving base data which speeds up query processing. c) Variants of Bitmap index reduce storage requirements and speeds up the performance. d) To speed up the queries further post evaluation of query predicates using Bitmap index, Projection index can be used for column retrieval which satisfy the predicates. e) For selecting a suitable indexing technique, an intelligent query optimizer can be employed and developed using data mining techniques.
Ladjel Bellatreche et al. [20]	Combination of enhanced indexing methods (join, bitmap), materialized views and data partitioning	➢ Experiment that the three major techniques namely: enhanced indexing methods (join, bitmap), materialized views and data partitioning when combined together reduces the query processing cost and maintenance overhead.
Tadeusz Morzy et al. [21]	Handling dynamics in content and structure in traditional data warehouse systems	➢ An approach is to use multi version data warehouse where each data warehouse version describes a schema and data at certain period of time or given business scenario. ➢ Multi version query language interface having functionalities such as expressing queries addressing several data warehouse versions and presenting their results by marking up with corresponding metadata information. This is done as follows: a. The query is decomposed into partial queries which are independent each for one data warehouse version specified in original query.

		b. The partial query is executed in its own data warehouse version and the result of each partial query is presented to the user along with version and metadata information which is used to analyse and interpret the obtained results. c. If possible these partial results are integrated into common set of data.
Panos Vassiliadis et al. [22]	Multidimensional data cubes	➢ Comparison of the various approaches within relational-oriented, cube-oriented, standards and statistical models was performed and the results were tabulated.
Venky Harinarayan et al. [23]	Materializing some cells of the cube.	➢ Investigated which cells are to be materialized when it becomes too expensive to materialize all cells. ➢ A lattice framework is used for expressing the dependencies among views followed by a greedy algorithm which works on this lattice picking the views which are to be materialized considering the constraints.
Fahad Sultan et al. [4]	Storing queries and their corresponding results	➢ Cache memory is first examined to check whether the query is already stored. ➢ Index is maintained to keep track of queries and their results. ➢ In case of data warehouse updates, if the same query is fired then the query does not have to check all records. Rather it will search for those records which satisfy the criteria from onwards to that index. ➢ Non re-evaluation of queries which are already stored in cache saves significant time and enhances data warehouse performance.

Table 1: Summary of Related Literature

III. CONCLUSION

Based on the above discussed literature, the following conclusions have been derived:

a. View is a derived relation defined in terms of base relations which can be materialized by storing its extent in database. Index structures may be implemented on the materialized views which make database access to tuples faster rather than re-computing the view. When there are changes in base relations i.e. deletion, insertion and updates, relevant change in view is possible [7].
b. Materialized views can improve query processing time especially in case of aggregate queries [17].
c. View maintenance is necessary as the data sources are updated [13].
d. Views being complex, it is better and cheaper to maintain them incrementally by applying the changes made to the base data rather than to re-compute the view from the scratch [9].
e. Though materialized views speed up query, to ensure correct results, they should be kept up to date when accessed by a query. They can be maintained eagerly i.e. in the same transaction as the base tables are updated and these updates bear the cost of view maintenance Overhead issues arise for maintaining materialized views. This overhead increases when multiple views are

maintained and hence results into poor response time for updates. Instead of forcing for updates, some database systems support the deferred maintenance approach, i.e. view maintenance can be delayed till the user explicitly triggers. This in turn can lead to out-of-date view producing incorrect results. Also, materialized views will no longer be automatic and transparent and query users need to have knowledge about the views used by a query, their maintenance and requirement of updation [12].

f. High query performance i.e. low query processing cost is in conflict with low view maintenance cost. By storing in the data warehouse the results of all the queries of interest, high query performance can be obtained. But the maintenance cost of the materialized queries might be high [16].

g. Multidimensional data cubes which are the logical model for OLAP (Online Analytical Processing) provide the functionality needed for summarizing, viewing and consolidating the information available in data warehouse [22].

h. In order to optimize query, we can materialize some cells instead of computing them from raw data every time. In case of implementation of data cube the available options are: 1) materialize the whole data cube, which will give best query response at the cost of higher storage, 2) materialize nothing resulting into computing every cell on request, 3) materialize only a part of the cube [23].

i. In case of materializing whole data cube, for n dimensions, the number of aggregates will be 2^n for snowflake schema [2].

REFERENCES

[1]. Kenneth C. Laudon, Jane P. Laudon, Rajanish Dass, *Management Information Systems*, pages 41- 46, Eleventh Edition, Pearson.

[2]. G. K. Gupta, *Introduction to Data Mining with Case Studies*, pages383-387, PHI Learning Private Limited, 2014.

[3]. Jiawei Han, Micheline Kamber, Jian Pei, *Data Mining-Concepts and Techniques*, pages126- 129, *Third* Edition, Morgan Kaufman Publishers.

[4]. Fahad Sultan, Abdul Aziz, “Ideal Strategy to Improve Data warehouse Performance,” *International Journal on Computer Science and Engineering* Vol. 02, No. 02, 2010, 409-415.

[5]. M. Bouzeghoub, F. Fabret, M. Matulovic-Broqué, “Modeling Data Warehouse Refreshment Process as a Workflow Application,” *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)* Heidelberg, Germany, 14. - 15. 6. 1999 (S. Gatziu, M. Jeusfeld, M. Staudt, Y. Vassiliou, Eds.).

[6]. Ashish Gupta, Inderpal Singh Mumick, “Maintenance of Materialized Views: Problems, Techniques and Applications,” *Bulletin of the Technical Committee on Data Engineering*, June 1995, Vol. 18, No. 2, IEEE Computer Society.

[7]. Ashish Gupta, Inderpal Singh Mumick, V.S.Subrahmanian, “Maintaining Views Incrementally,” *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Pages 157-166.

[8]. Randall G. Bello, Karl Dias, Alan Downing, James Feenan, Jim Finnerty, William D. Norcott, Harry Sun, Andrew Witkowski, Mohamed Ziauddin. “Materialized Views in Oracle”. *Proceedings of the 24th VLDB Conference*, New York, USA, 1998.

[9]. Dallan Quass, “Maintenance Expressions for Views with Aggregation,” *Views'96*, June 1996, [Online]. Available: http://ilpubs.stanford.edu:8090/183/1/1996-54.pdf

[10]. Ashish Gupta, H.V. Jagadish, Inderpal S. Mumick, “Data Integration using Self-Maintainable Views,” *Advances in Database Technology — EDBT '96*, Volume 1057 of the series Lecture Notes in Computer Science, pp 140-144.

[11]. Kenneth A Ross, Divesh Srivastava, S.Sudarshan, “Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time,” *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Pages 447-458.

[12]. Jingren Zhou, Per-Ake Larson, Hicham G. Elmongui, "Lazy Maintenance of Materialized Views," *VLDB '07* Proceedings of the 33rd International Conference on Very large Databases, Pages 231-242.

[13]. Yue Zhuge, Hector Garcia-Molina, Joachim Hammer, Jennifer Widom, "View Maintenance in a Warehousing Environment," *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, Pages 316-327.

[14]. D. Agrawal, A. El Abbadi, A. Singh, T. Yurek, "Efficient View Maintenance at Data Warehouses," *SIGMOD '97*, AZ, USA @ 1997 ACM 0-89791 -911 -419710005.

[15]. Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, Kyuseok Shim, "Optimizing Queries with Materialized Views," [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.2680&rep=rep1&type=pdf

[16]. Dimitri Theodoratos, Timos Sellis, "Data Warehouse Configuration, "*Proceedings of the 23rd VLDB Conference* Athens, Greece, 1997.

[17]. Jonathan Goldstein, Per-Ake Larson, "Optimizing Queries Using Materialized Views: A Practical, Scalable Solution," *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, Pages 331-342, ISBN:1-58113-332-4.

[18]. Divesh Srivastava, Shaul Dar, H. V. Jagadish, Alon Y.Levy, "Answering Queries with Aggregation Using Views," *Proceedings of the 22nd VLDB Conference*, Mumbai (Bombay), India, 1996.

[19]. Sirirut Vanichayobon. "Indexing Techniques for Data Warehouses' Queries". [Online] Available: http://www.cs.ou.edu/~database/documents/vg99.pdf , https://pdf, https://pdfs.semanticscholar.org/00ce/fcf3c23b28b0b51f98ac2d3998ab43e904ca.pdf [Accessed May 25, 2017].

[20]. Ladjel Bellatreche, Michel Schneider, Herv´e Lorinquer, Mukesh Mohania, "Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses," *Y. Kambayashi et al. (Eds.): DaWaK 2004*, LNCS 3181, pp. 15–25, 2004. Springer-Verlag Berlin Heidelberg 2004.

[21]. Tadeusz Morzy, Robert Wrembel, "On Querying Versions of Multiversion Data Warehouse," *DOLAP'04*, November 12–13, 2004, Washington, DC, USA. Copyright 2004 ACM 1-58113-977-2/04/001.

[22]. Panos Vassiliadis, Timos Sellis, "A Survey of Logical Models for OLAP databases," *ACM SIGMOD Record*, Volume 28 Issue 4, Dec.1999, Pages 64 – 69.

[23]. Venky Harinarayan, Anand Rajaraman, Je
rey D. Ullman, "Implementing Data Cubes Efficiently," *Proceedings of the 1996 ACM SIGMOD International Conference on Management of data*, Pages 205-216.

[24]. Panos Vassiliadis, Alkis Simitsis, "Extraction, Transformation, and Loading," Encyclopedia of Database Systems, pp 1095-110.1

[25]. Adela Bara, Ion Lungu, Manole Velicanu, Vlad Diaconita, Iuliana Botha, "Improving query performance in virtual data warehouses," *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Issue 5, Volume 5, 2008, ISSN: 1790-0832.