© 2018 IJSRSET | Volume 4 | Issue 1 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

An Approach for Creating and Maintaining Dependent Data Marts using Materialized Queries' Information

Sonali Chakraborty*[1], Jyotika Doshi[2]
[1]K. S School Of Business Management, Gujarat University, Ahmedabad, Gujarat, India
[2]GLS Institute of Computer Technology (MCA), GLS University, Ahmedabad, Gujarat, India

ABSTRACT

Result generation of enterprise OLAP queries is done using warehouse data. Traversing through records in a warehouse is quite time consuming due to enormous amount of historical and current data. Data warehouse undergoes periodic refreshing to keep the information updated. OLAP queries along with their results and other parameters are stored in Materialized Query Database (MQDB) for faster result retrieval. If same OLAP query is fired again, and it requires incremental updates, complete analysis of warehouse data is done again. This results into higher query execution time. Performing incremental updates using predefined data marts will give faster query results as compared to generating results using warehouse data. This paper discusses an approach to determine and create dependent data marts and maintaining them periodically.

Keywords: Data Mart, Data Warehouse, Materialized Queries

I. INTRODUCTION

Organizations store historical data into data warehouse for analysis and decision making. Warehouse data is integrated through various operational data sources and is then loaded through the process of ETL (Extraction, Transformation, and Loading). Warehouse data is refreshed periodically to ensure that the information is current. The refreshing rate of data warehouse varies with the size of the organization and the number of transactions occurring per unit time in the organization [2]. This increases the size of data warehouse due to enormous amount of current and historical data. This leads to more query execution time.

[11] OLAP queries are fired by the management on warehouse data. Certain OLAP queries are fired quite frequently and hence warehouse data is analysed repeatedly for generating same result. Traversing through huge amount of warehouse data is quite time consuming and it eventually results into higher query execution time. Executed OLAP queries along with their results are materialized and stored in a separate relational database called Materialized Query Database (MQDB). Next time when the same or its equivalent query is fired, results are fetched from MQDB, if no incremental updates are required. This results into faster result retrieval.

If an OLAP query requires incremental updates then, result is generated by traversing through warehouse data. Though, materialized query speed up processing by fetching past results, the time required for performing incremental updates using data warehouse increases the query processing time.

To overcome the performance issues with data warehouse, this paper suggest the use of dependent data marts for processing incremental updates. Fewer records in data marts will make result retrieval faster

as compared to accessing warehouse every time. This approach gives better performance with faster query execution time.

II. RELATED LITERATURE

M Serranoa [1] highlighted that the quality of data warehouse is crucial for organizations as its principal role is in making strategy decisions. Methods, models and techniques must be used for designing and maintaining high quality data warehouse. Lou Agosta [2] commented that the frequency of refreshing a data warehouse depends on the type of industry, application, business process, time horizon of the business process and technical infrastructure. Thomas Jorg and Stefan Dessloch [3] suggested that incremental loading for data warehouse refreshment is more efficient and preferable where the changes are captured at the operational data sources and then refresh the data warehouse incrementally. Mokrane Bouzeghoub et al. [4] emphasized that refreshment of a data warehouse is an important process which determines the effective usability of the data collected and aggregated from the sources. 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.

[5] Data marts; subset of data warehouse are placed between the operational source and the data warehouse. Data mart requirements vary according to the enterprise size and user requirements. They are created after gathering user requirements and formulating them into star join schema structure. Angela Bonifati et al. [6] suggest that the data mart design process should be based on the needs of the business. Their proposed method consists of three steps namely; top-down analysis, bottom-up analysis followed by integration. Authors [7] proposed an approach consisting of four major tasks; 1) acquisition of OLAP requirements; 2) generation of star/ constellation schema by merging the above requirements; 3) Data warehouse generation schema by joining data marts ; 4) mapping the data marts to data sources. They defined an algorithm for automatically transforming OLAP requirements from step 1 into data marts. Mapping rules are defined between the data sources and data marts. Data marts are merged to create the data warehouse using set of unification rules.

Fatma Abdelhédi et al. [8] in their literature propose to design a data mart schema using a hybrid driven approach. Decision makers express their requirements informally. It is a four step process where in first step extraction of candidate facts is done through source class diagrams. Dimensions associated with the facts are derived in the second step. Hierarchies for each dimension are presented in the third step. Data mart schemas are elaborated in the fourth step corresponding to decision makers' requirements. Authors Anmar Aljanabi et al. [9] using Ralph Kimball's methodology developed a query dispatching tool facilitating the access to the information within data marts, eventually data warehouse in fast and an organized way. It takes the query, analyses it, and decides which data mart as a destination that query should be forwarded to. The results show that the dispatching tool reduces the check time spent in the data dictionary within a logical side of the data warehouse deciding the intended data mart and hence, minimizing execution time.

III. CREATING AND MAINTAINING DATA MARTS

Creating appropriate data marts and maintaining them periodically is a critical factor for the performance of OLAP queries. Data mart requirements vary according to the enterprise size and user requirements. Determining proper dimensions in data marts eliminates the need for repeated data warehouse access for frequent OLAP queries. Scope of data marts is limited to single business process / function and hence it is optimal for data access and analysis. Also, the response time improves as the volume of data accessed is reduced.

To understand the creation of data marts, let us consider an example of an academic organization providing education facilities to people all over India. Organization's data warehouse contains data about the number of males and females of different age groups belonging to different education levels all over India. Data is collected from http://censusindia.gov.in. Consider some OLAP queries fired by the organization:

Query 1: List the number of females pursuing primary education in each town

```
SELECT dw_town.town_name, dw_zones.primary_f
FROM dw_zones, dw_town
WHERE dw_zones.town_code= dw_town.town_code
GROUP BY dw_zones.town_code
```

Query 2: Find the town name having highest number of illiterate males for each state.

```
SELECT max (dw_zones.illiterate_f), dw_town.town_name, dw_states.state_name
FROM dw_zones, dw_town, dw_state
WHERE dw_zones.town_code= dw_town.town_code
AND dw_town.state_code= dw_state.state_code
AND dw_state.state_code=dw_zones.state_code
GROUP BY dw_zones.state_code
```

Query 3: Find the average number of illiterate females in each state

```
SELECT avg (dw_zones.illiterate_f), dw_states.state_name
FROM dw_zones, dw_states
WHERE dw_zones.state_code = dw_states.state_code
GROUP BY dw_zones.state_code
ORDER BY dw_states.state_name
```

Query 4: Find the state having least number of males pursuing graduation

```
SELECT min (dw_zones.graduate_m), dw_states.state_name
FROM dw_zones, dw_town, dw_state
WHERE dw_state.state_code=dw_zones.state_code
GROUP BY dw_zones.state_code
```

Query 5: Find the town names having maximum number of males pursuing secondary, higher secondary and diploma education in each state.

```
SELECT max (dw_zones.secondary_m), max (dw_zones.hsecondary_m), max (dw_zones.diploma_m), dw_town.town_name, dw_states.state_name
FROM dw_zones, dw_states, dw_town
WHERE dw_zones.state_code = dw_states.state_code
AND dw_states.state_code= dw_town.state_code
AND dw_town.town_code= dw_zones.town_code
GROUP BY dw_zones.town_code
```

[11] When an OLAP query is fired, it is first searched in MQDB if an equivalent query exists. If no equivalent query exists, then query along with other metadata information is stored in MQDB. Storing of queries and determining equivalent queries from MQDB is depicted in [11]. In case there is no incremental update, then results are fetched from MQDB and metadata is updated.

Storing queries by generating identifiers for each table, fields and aggregate function is explained by the authors in their previous paper [11]. Hence, the queries discussed in our example will be stored in "Stored_Query" table as shown in Table 1. Their corresponding metadata information is stored in "Materialized_Query" table of MQDB as shown in Table 2.

Table 1. "stored_query" table in mqdb [11]

sq_id	query_id	Table_id	Field_id	Function_id
sq1	q1	02	03	
sq2	q1	04	12	
sq3	q1	04	03	08
sq4	q2	04	06	04
sq5	q2	02	03	
sq6	q2	01	02	
sq7	q2	04	02	08
sq8	q3	04	06	02
sq9	q3	01	02	10

sq10	q3	04	02	08
sq11	q4	04	21	03
sq12	q4	01	02	
sq13	q4	04	02	08
sq14	q5	04	15	04
sq15	q5	04	17	04
sq16	q5	04	19	04
sq17	q5	02	03	
sq18	q5	01	02	
sq19	q5	04	03	08

Table 2. "Materialized_Query" in MQDB [11]

query _id	query_date	query_ frequenc y	query_ thresho ld	size	path of result table
q1	12/10/2017	10	15	419	q1_res ult
q2	12/02/2017	3	20	34	q2_res ult
q3	11/30/2017	6	12	34	q3_res ult
q4	01/18/2018	12	15	1	q4_res ult
q5	01/20/2018	13	20	1	q5_res ult

If equivalent query is found in "Stored_Query" table and it requires incremental updates, then data warehouse is analysed to generate incremental results. This will consume more time.

To speed up query processing for incremental updates, authors plan to build a dependent data mart having fewer records as compared to data warehouse.

Dependent data mart refers to that kind of data mart which takes the data feed from data warehouse. Data warehouse integrates information from various operational sources and subset of information from data warehouse is feed into the data mart. These data marts contain data for specific user group within a particular department or function and hence are easier to use [10].

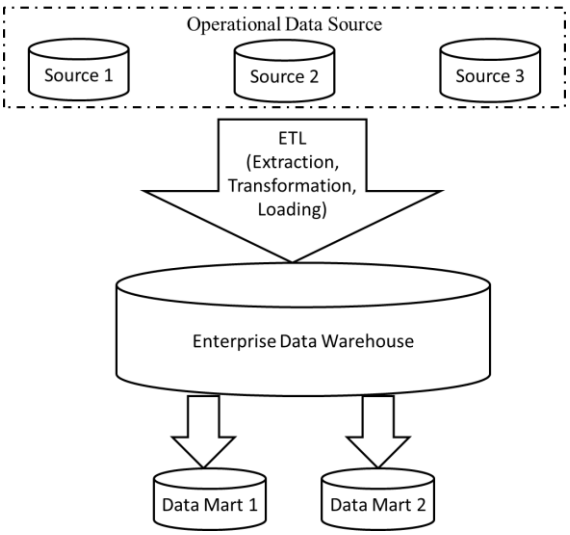

Figure 1. Structure of dependent data marts

A. Determining dimensions in data marts

Determining the dimensions to be included while creating a data mart requires proper analysis of the records in "Stored_Query" and "Materialized_Query" table depicted in Table 1 and Table 2 respectively.

Data marts may contain fields from multiple tables also. This helps in preventing join operation on columns of multiple tables. Multiple data marts can be created based on enterprise requirement.

Dimensions of data mart can be determined as:

(i) By Users

(ii) By using query information stored in MQDB.

(i). Data marts can be defined by the users if the function requirement is known to them. They can select the dimensions to be included in the data mart.

Example: Department working for "Female Education" can define the data mart by adding dimensions required for analysis from data warehouse. If multiple data marts are defined by the users based on their requirements, then one field may exist in multiple data marts. This will lead to duplication of fields in data marts. Also, it becomes difficult to administer multiple data marts.

(ii). Another way of determining data marts using information stored in MQDB is as follows:

1. For each queries stored in "Materialized_Query" table in MQDB, check the frequency against the threshold value.

2. If frequency > = (threshold /2) i.e. frequency of the query is more than 50% of the threshold value, then fetch the fields of that query from "Stored_Query" table.
3. Eliminate duplicate fields if any.
4. Fields obtained after performing step 3 are included as dimensions of the data mart.

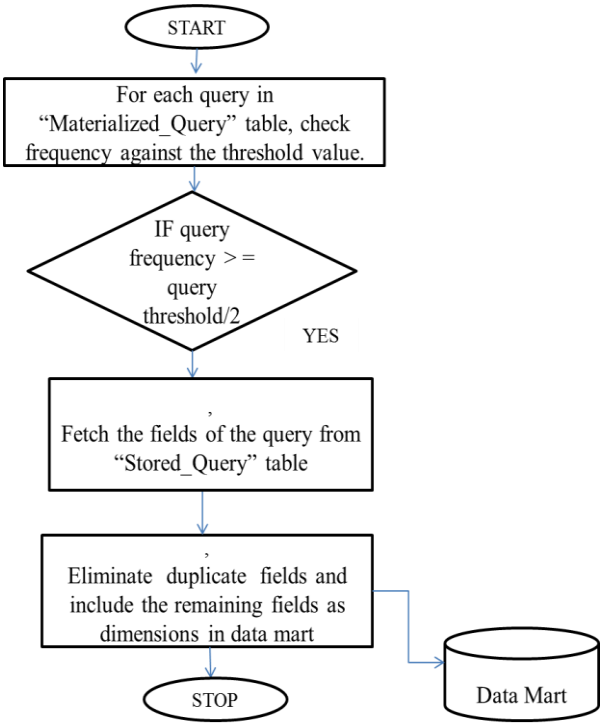

Figure 2. Determining dimensions of data mart

Based on the queries discussed in example, we determine the dimensions of data mart. The threshold and frequency value of each query is evaluated from "Materialized_query" table of MQDB shown in Table 2. Queries having frequency >= (threshold/ 2), i.e. query frequency more than or equal to 50% of threshold, are shortlisted.

From Table 2, it is observed that query_id q1, q3, q4 and q5 fulfill the frequency-threshold criteria. Fields of these queries are fetched from "Stored_query" table as shown in Table 3. The selected fields after eliminating duplicates from Table 3 are included as dimensions of data mart as depicted in Table 4.

Table 3. Fields of queries MQDB fulfilling frequency-threshold criteria

query_id	Table-Field Identifiers	Table-Field Names
q1	(02- 03) (04-12) (04-03)	(dw_town, town_name) (dw_zones, primary_f) (dw_zones, town_code)
q3	(04, 06) (01, 02) (04, 02)	(dw_zones, illiterate_f) (dw_states, state_name) (dw_zones,state_code)
q4	(04, 21) (01, 02) (04, 02)	(dw_zones, graduate_m) (dw_states, state_name) (dw_zones, state_code)
q5	(04, 15) (04, 17) (04, 19) (02, 03) (01, 02) (04, 03)	(dw_zones, secondary_m) (dw_zones, hsecondary_m) (dw_zones, diploma_m) (dw_town, town_name) (dw_states, state_name) (dw_zones, town_code)

Table 4. DIMENSION DETERMINED FOR DATA MART "DM1"

Table Name	Field Name
dw_zones	state_code, town_code, illiterate_f, primary_f, graduate_m, secondary_m, hsecondary_m, diploma_m
dw_states	state_name
dw_town	town_name

B. Creating and Maintaining data marts

(i) Creation of data mart

Data mart is created using the dimensions as determined in 3.1.

(ii) Maintaining data marts

Data marts are refreshed periodically for keeping records up to date. Continuous addition of records in data marts will eventually increase its size. This in turn will degrade the performance of data marts.

Maintaining data marts refers to periodic refreshing with recent updated records at the same time

removing historic records. Also, determining the number of records to be kept at a time in a data mart is an important factor. To limit the number of records in data mart, "query timestamp" value stored in "Materialized_Query" table in MQDB is analysed. Query timestamp refers to the date on which the query was fired last time. Also, the maximum timestamp value of data mart is fetched.

Example:

Suppose the minimum timestamp value of the queries fetched from "Materialized_Query" table is 02/05/2018 (mm/dd/yyyy).

Maximum timestamp value of data mart is 02/16/2018 (i.e. last data mart refresh).

Suppose next data mart refresh is done on 02/26/2018. If data warehouse records after 02/05/2018 (minimum query timestamp) are loaded into data mart then duplicate records will be generated for the time period between 02/06/2018 to 02/16/2018. To avoid duplication of records in data mart, records inserted in data warehouse after 02/16/2018 (maximum data mart timestamp) are loaded in data mart.

Data mart maintenance is done as follows:

1. Scan the query timestamp value from "Materialized_Query" table in MQDB to find the minimum timestamp value.
2. Fetch the maximum timestamp value of the records stored in data mart.
3. If the data mart timestamp value is NULL, then; fetch records from data warehouse which are loaded on or after the minimum query timestamp value. Load a copy of records in data mart. Otherwise;

a. Delete the records from data mart having timestamp value less than or equal to minimum query timestamp value.

b. Fetch records from data warehouse having timestamp value greater than the maximum data mart timestamp value.

c. Load a copy of the incremental records in data mart.

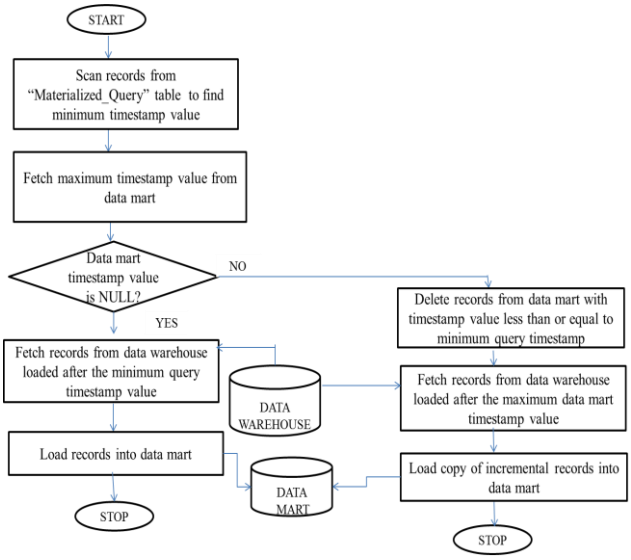

Figure 3. Algorithm for data mart maintenance

IV. CONCLUSION

Refreshing data marts using incremental records ensured that the information stored in up to date and the data size is less. This leads to faster performance of query processing using data mart as compared to data warehouse.

V. REFERENCES

[1]. Manuel Serranoa, Juan Trujillo, Coral Calero, Mario Piattini, "Metrics for data warehouse conceptual models understandability," Information and Software Technology, Volume 49, Issue 8, August 2007, Pages 851–870.

[2]. Lou Agosta, "Data Warehousing Refresh Rates," Information Management, 2003. Online] Available: http://search.proquest.com/openview/d27cfb901e3832ced0e374680b990baa/1?pq-origsite=gscholar&cbl=51938 Accessed February 19, 2017].

[3]. Thomas Jorg and Stefan Dessloch, "Formalizing ETL Jobs for Incremental Loading of Data Warehouses," Proceedings der GI-Fachtagung fur Datenbanksysteme in Business, Technologie und Web, Lecture Notes in Informatics, Volume P-144, 2. bis 6. M¨ arz 2009, M ¨ unster,

Germany, pp. 327–346. ISBN 978-3-88579-238-3.

[4]. 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.).

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

[6]. Angela Bonifati, Fabiano Cattaneo, Stefano Ceri, Alfonso Fuggetta, Stefano Paraboschi, "Designing Data Marts for Data Warehouses," ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 10 Issue 4, Oct. 2001, Pages 452-483.

[7]. Ahlem Nabli, AhlemS oussi, Jamel Feki, Hanêne Ben Abdallah, Faiez Gargouri, "Towards An Automatic Data Mart Design," ICEIS 2005 - Databases And Information Systems Integration.

[8]. Fatma Abdelhédi, Geneviève Pujolle, Olivier Teste, Gilles Zurfluh, "Computer-Aided Data-Mart Design," ICEIS (1), 2011.

[9]. Anmar Aljanabi, Alaa Alhamami, Basim Alhadidi, "Query Dispatching Tool Supporting Fast Access to Data Warehouse," The International Arab Journal of Information Technology, Vol. 10, No. 3, May 2013.

[10]. Reema Thareja. Data Warehousing, Oxford University Press.

[11]. Sonali Chakraborty, Jyotika Doshi, "Performance Evaluation of Materialized Query," International Journal of Emerging Technology and Advanced Engineering, Volume 8, Issue 1, January 2018, pages 243-249.