An Integrated Materialized View based Approach in ETL with DSS to Achieve Fast Data Transformation and Better Query Optimization

N. Revathy
Research Scholar,
Mother Terasa Womens University,
Kodaikanal, India

T. Guhan
Research Scholar,
Anna University,
Chennai ,India

P. Revathi
Final Year MCA Student,
Karpagam College of Engineering,
Coimbatore -32, India

Abstract:

A data warehouse can be defined as a subject-oriented, integrated, nonvolatile and time-variant collection of data, which has value and role for decision-making by querying. To avoid accessing base tables and increase the speed of queries posed to a DW, we can use some intermediate results from the query processing stored in the DW called materialized views. Materialized views are physical structures that improve data access time by pre-computing intermediary results. The ETL process bridging the online transaction processing (OLTP) system and the online analytical processing (OLAP) system is often modeled as a separate and independent process. When transformed data are loaded into the data warehouse, the analysis-centric applications take place. To expedite the analysis process, materialized views are often created and used through the query rewrite mechanism in the data warehouse.

The proposed approach integrate the ETL process and the data warehouse applications by using views and materialized views to model and perform the ETL process. This process integration has several advantages.

First, it achieves fast data transformation and materialized view maintenance through one single materialized view refresh call. Changes in the OLTP system can be quickly and transparently applied to the materialized views so that a near real-time data analysis can be carried out. Second, the use of views and materialized views to model the ETL process provides the benefits of encapsulating data transformations in a multi-step SQL process. Compared to the commonly used scripting approach, this approach provides better readability and maintainability. Third, because the transformation SQL is processed inside the database, better query optimization improves the ETL performance. Lastly, the use of materialized views in the ETL process could facilitate the data cleansing so that clean data are passed through and processed while dirty data are intercepted and loaded into tables for correction. The transformation for the corrected data can be resumed at the point of error spotted.

Keywords--- OnLine Analytical Processing, Data Warehouse, ETL.

I. INTRODUCTION

Problem Overview and Motivation

Data analysis and decision making are ultimate goals in the data warehouse applications. Consider a large data warehouse containing terabytes of data distributed in de-normalized tables each of which has potentially millions or even billions of data rows. Retrieving data from such system is a very resource-intensive and time-consuming process. Queries that retrieve and calculate a large amount of data could require hours to be processed. To speed up the query processing in the OLAP system, a commonly used approach is the use of materialized views.

The materialized view is a database object that contains pre-calculated data typically for highly aggregate queries associated with complex joins of fact and dimension tables. The use of the materialized view in query processing is often transparent through a mechanism called *query rewrite*. In such mechanism, the original query is rewritten to make use of the materialized view instead. The query response time is thus improved. Consequently, the materialized view has become a common but required resident in the OLAP system. Then, consider the subject of data loading to the OLAP system, a well-known process called ETL.

ETL extracts the source data from the operation-centric OLTP system, transforms the data to resolve various kinds of heterogeneity problems and loads the transformed data into the target OLAP system (e.g., data warehouse). The

ETL process is usually an independent process that bridges the two systems. It is often achieved by the third party tool which executes the ETL process in a specific window of time. Due to this asynchronous nature, the changes in the OLTP system are not transformed and propagated to the OLAP system in the real time. As a result, real-time data analysis and decision making is difficult to achieve with such process model. On the other hand, since the materialized view has already become a common data warehouse object for improving query performance, it will be beneficial to use the materialized view to model the ETL process so that the ETL process and the data warehouse applications can be seamlessly integrated. This idea motivates us to investigate and design an integration mechanism that is based on the materialized view technology.

In our approach, we combine the ETL process into the OLAP application by conditionally using views and materialized views. The buffering of intermediate results in the ETL process and the fact and dimension tables in the OLAP system are all represented in materialized view objects. The changes propagation and the data transformation are done in one single refresh call that refreshes all the materialized views in the dependency hierarchy. Such design generates an efficient integration and enables near real-time data analysis. Also, the approach can support data cleansing to filter out dirty data and allows corrected data to be re-submitted into the ETL process at the spot where the error is found.

In Section 2, we first review related work in the areas of the ETL modeling, data cleansing and processing performance. Section 3 presents our proposed approach and compares it with the current architecture. We then illustrate the use of the materialized view to model the ETL process and data cleansing with examples. Later, we discuss the advantages and concerns of our approach. Section 4 describes the conclusion.

Dissertation Objectives

This process integration approach has several objectives. First, it should achieve fast data transformation and materialized view maintenance through one single materialized view refresh call. Changes in the OLTP system can be quickly and transparently applied to the materialized views so that a near real-time data analysis can be carried out. Second, the use of views and materialized views to model the ETL process should provide the benefits of encapsulating data transformations in a multi-step SQL process. Third, it should improve the ETL performance and provide better query optimization. Lastly, the use of materialized view in the ETL process should facilitate the data cleansing so that clean data are passed through and processed while dirty data are intercepted and loaded into tables for correction.

II. DESIGN OF EXPERIMENTS AND ANALYSIS OF RESULTS

Here, we first show the example of ETL process modeled in views and materialized views and the queries used in experiments. Second, we show the data cleansing using materialized views and the related queries. Finally we show the results and analysis taken by this experiment.

Example of ETL process modeled in views and materialized views

Figure11 shows an example of using materialized views to integrate the ETL process and the OLAP applications.

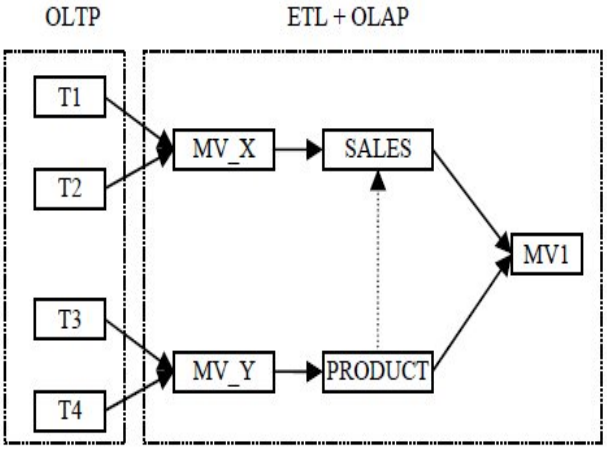

Figure 11: Integrated ETL and OLAP using Materialized Views

In this example, four tables T1-T4 reside in the OLTP source system, while in the target OLAP system, two tables SALES and PRODUCT are in the warehouse schema. In addition a materialized view, MV1 is built on top of tables SALES and PRODUCT to support quick query reporting.

It is noted that there are two additional materialized views, MV_X and MV_Y, are created to bridge the OLTP tables (T1-T4) and OLAP tables (SALES and PRODUCT). The materialized view MV_X is to perform part of the ETL process to transform data from T1 and T2 to SALES, while MV_Y is for transforming data from T3 and T4 to PRODUCT. In this architecture, table SALES is also a materialized view that is based on MV_X. The definitions of the materialized view MV_X can be specified as follows:

CREATE MATERIALIZED VIEW MV_X

```
FAST REFRESH ON DEMAND
AS
        SELECT T1.ROWID T1RID, T2.ROWID T2RID,
        CONV1 (T1.PRICE_SALES) PRICE_SALES,
        CONV2 (T2.UNIT_SALES) UNIT_SALES,
        CONV3 (T1.AMOUNT) AMOUNT_SOLD,
        T2.PRODUCT_ID PROD_ID,
        T1.C1
FROM T1@OLTP, T2@OLTP
WHERE T1.C1=T2.C1
```

The materialized view MV_X is a materialized join view based on tables T1 and T2. All required columns of data from both tables are extracted and transformed through proper conversion functions. This materialized view is created with the specification of REFRESH FAST ON DEMAND. That is, it support log-based incremental refresh maintenance and the refresh invocation is on demand. To achieve that, both T1 and T2 have a materialized view log build on each of them before the materialized view MV_X is created. The materialized view log captures the information for all rows that has been changed in the table (e.g., T1). The refresh of the materialized view MV_X is done through the following statements.

```
EXECUTE DBMS_MVIEW.REFRESH ('mv_X');
```

It is also noted that there are two ROWID columns included in the defining query of MV_X. They are included to support log-based incremental refresh as the ROWID column contains the physical address of the data rows and can be used to speed up the refresh process. Besides, the fact table SALES is also modeled as a materialized view as follows:

```
CREATE MATERIALIZED VIEW SALES
FAST REFRESH ON COMMIT
AS
        SELECT X.ROWID XRID, P.ROWID PRID,
        FUNC1 (X.PRICE_SALES) DOLLER_SALES,
        X.UNIT_SALES,          X.AMOUNT_SOLD,
X.PROD_ID, X.C1
FROM MV_X X, PRODUCT P
WHERE X.PROD_ID=P.ID;
```

Similar to MV_X, SALES is a materialized join view. It looks up the product key from the dimension table PRODUCT (which implies that PRODUCT is built up and maintained first). The ROWID columns are for the same purpose of supporting incremental refresh. Both MV_X and PRODUCT have materialized view logs created on them to support incremental refresh of SALES. It is noted that SALES is a nested materialized view based on MV_X and is specified as REFRESH FAST ON COMMIT. That is, when the materialized view MV_X gets refreshed, the incremental refresh of SALES is automatically kicked off to propagate and apply the changes. The original materialized view used for speeding up query performance is MV1. Its definition statement is unchanged except the REFRESH mode is changed to ON COMMIT. It is to achieve automatic change propagations as done on SALES. The materialized view creation statement for MV1 could be defined as follows.

```
CREATE MATERIALIZED VIEW mv1
FAST REFRESH ON COMMIT ENABLE QUERY
REWRITE
AS
        SELECT S.PROD_ID,
        COUNT (*) CT_STAR,
        COUNT (DOLLER_SALES) CT_DSALES,
        SUM (DOLLER_SALES) SUM_DSALES,
        COUNT (AMOUNT_SOLD) CT_ASOLD,
        SUM (AMOUNT_SOLD) SUM_ASOLD
FROM SALES S, PRODUCT P
WHERE S.prod_id=P.id
GROUP BY S.prod_id;
```

In this example, to trigger the automatic change propagations from the OLTP tables to the OLAP tables and materialized views are through the following refresh statement execution.

```
EXECUTE DBMS_MVIEWS.REFRESH ('MV_X, MV_Y')
```

All materialized views based on the above two materialized view in the dependency hierarchy will be automatically refreshed. The materialized view MV1 will therefore be available for query rewrite and support near real-time data analysis.

Example of using materialized views for data cleansing

The use of the materialized views in the ETL process can support data cleansing as well. As previously mentioned automatic conflict resolutions and fixes are nearly impossible due to the complexity of data representations. Part of the conflict resolutions and fixes will have to rely on human experts to examine the dirty data and make corrections. Typically, when transformed data are identified as dirty, those rows are recorded in a separate file or table for the human expert to examine. After corrections, the data can be resubmitted to the transformations process.

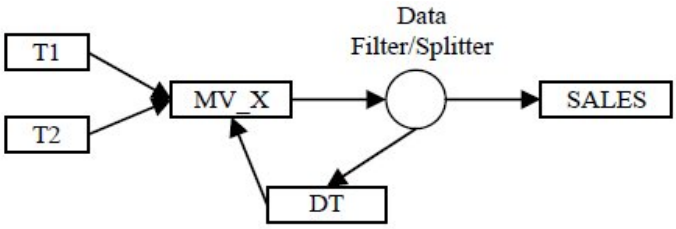

Figure 12: Materialized Views for Data Cleansing

Figure12 shows an example of using materialized views to support data cleansing. Similar to the example given in Section 3.3, the materialized view MV_X contains intermediate transformation results which are to be passed to SALES. However, the data quality needs to be ensured before the data passing. Therefore, we use a data filter/splitter (as shown in an oval shape in Figure 4) that represents one or more condition checks. The data that pass the condition checks are forwarded to populate SALES. Otherwise, the data are treated as "dirty" and redirected into a table DT for the human expert to examine and correct. The following statement is an example to populate the table DT:

```
MERGE INTO DT
        (SELECT T1RID, T2RID, PRICE_SALES,
        UNIT_SALES, AMOUNT_SOLD, PROD_ID, C1,
        DECODE (PRICE_SALES < 0, TRUE, 'Invalid
PRICE_SALES',
        DECODE (PROD_ID > 250, TRUE, 'Invalid
PROD_ID','OK')) MSG
FROM MV_X
WHERE PRICE_SALES < 0 or
        T2.PROD_ID > 250)
WHEN NOT MATCHED
INSERT;
```

It is noted that the table DT has one extra column, MSG. it is to provide the reason why the data is considered as dirty. The human expert will use the information to correct the data. After the correction, the corrected data in the table DT is resubmitted to the ETL process through MV_X. To support this, the creation statement for MV_X is modifies as follows.

```
CREATE MATERIALIZED VIEW MV_X
FAST REFRESH ON DEMAND
AS
        SELECT '1' UMARKER, '0' DTRID
        T1.ROWID T1RID, T2.ROWID T2RID,
        CONV1 (T1.PRICE_SALES) PRICE_SALES,
        CONV2 (T2.UNIT_SALES) UNIT_SALES,
        CONV3 (T1.AMOUNT) AMOUNT_SOLD
        T2.PRODUCT_ID PROD_ID,
        T1.C1
FROM T1, T2
WHERE T1.C1=T2.C1
UNION ALL
        SELECT '2' UMARKER, DT.ROWID DTRID
        T1RID, T2RID, PRICE_SALES,
        UNIT_SALES, AMOUNT_SOLD, PROD_ID, C1
FROM DT;
```

The modified MV_X to support data cleansing has one extra query block connected to the original query through a UNION ALL operator. To support log-based incremental refresh for MV_X, two additional columns (UMARKER AND DTRID) are added. This way, the materialized view MV_X is enable to include corrected data in the ETL process. Another minor modification is needed for SALES whose defining query is added with the filter conditions so that only correct data are passed to SALES.

Results and Analysis

The above proposed approach that integrates the ETL process and the data warehouse applications has the following Results. It integrates the ETL process and the data warehouse applications in a common framework. Instead of using the third party ETL tool that potentially causes high data transmission overhead, the ETL process is combined into the data warehouse system in a seamless fashion which saves the data transmission overhead from the ETL engine to the target system.

It achieves fast data transformation and materialized view maintenance through one single materialized view refresh call. The use of on-commit nested materialized views enables the changes in the OLTP system to be quickly and transparently applied to the terminal materialized views so that a near real-time data analysis can be realized.

The use of views and materialized views to model the ETL process provides the benefits of encapsulating data transformations in a multi-step SQL process. Compared to the scripting and fine-step GUI-based transformation approaches, this approach offers a neutral representation of the ETL plan which provides better readability and maintainability. The modification of the ETL plan is as simple as the defining query change of the materialized view.

Because the materialized views that model the ETL plan are processed inside the database, efficient query processing strategies by the optimizer can improve the ETL process performance. Common operations such as data matching, sorting and filtering can be done much more efficiently.

The use of materialized view in the ETL process could facilitate the data cleansing as well. In this approach, clean data are passed and processed through the normal path while dirty data are intercepted and loaded into tables for correction. The transformation for the corrected data can be resumed at the point of errors spotted and fed into the transformation process.

On the other hand, the use of materialized views to integrate with the ETL process could introduce the following analysis requirements.

There are additional storage requirements for the materialized views. We observed that and suggest conditional use of view in place of materialized view when the intermediate transformation result is not necessarily stored and the incremental refresh of the next materialized view is preserved. This alleviates the concern of storage needs. Also, a range-based incremental refresh can be used to support the change propagations only for certain areas of data (e.g., a period of time). This allows old and no longer needed intermediate transformation results in the materialized views can be aged out to conserve storage space.

A common but fundamental question is: can the materialized view sufficiently handle all possible ETL scenarios? The answer appears to be positive as all ETL processes that can be written in scripts or programs are able to be represented in a sequence of SQL statements. To handle some schematic transformations, using an intermediate materialized view is sufficient to achieve the goal.

CONCLUSION AND FUTURE WORK

The presented approach provides a seamless integration that can quickly and efficiently propagate the changes from the OLTP source system all the way to the materialized view in the OLAP data warehouse system. Thus, the materialized views used by query rewrite contain up-to-date information and are able to support near real-time data analysis/mining and consequently benefit decision making.

The approach is carried out by a number of intermediate materialized views or views to model the ETL process in the OLAP system. This enables the ELT execution inside the database in which more efficient optimization mechanisms for data processing can be utilized. The ETL process can therefore be more efficient. In addition, the use of on-commit nested materialized views automates the data transformation and the refresh maintenance of terminal materialized views in one single refresh call which greatly eases the process that is currently handled in a complicated way.

Finally, some concerns about this approach such as storage requirements are alleviated by conditionally process. Also, the unnecessary transformation results can be aged out from the intermediate materialized views by range refresh maintenance of the materialized view.

REFERENCES

1. Ashish Gupta and Inderpal Sing Mumick, 1995. Materialized views: techniques, implementations and application, 2nd Edition, University of California.
2. Claudia Imhoff, Nicholas Galemmo and Jonatha G.Geiger, 2001. Mastering Data Warehouse Design, 3rd Edition, Wiley Publishing, Inc.
3. Elzbieta Malinowski and Esteban Zimanyi, 2008. Advanced data warehouse design, 1st Edition, Springer.
4. Gavin Powell, 2005. Oracle Data Warehouse Tuning, Oracle Corporation.
5. Paulraj Ponniah, 2011. Data Warehousing Fundamentals for IT Professionals, 1st Edition, John Wiley & Sons New York.
6. Rajiv Parida, 2006. Principles and Implementation of Datawarehousing, 1st Edition, Firewall Media.
7. Ralph Kimbal, 2002. The Data Warehouse Toolkit, Practical Techniques for Building Dimensional Data Warehouse, 2nd Edition, John Wiley & Sons New York.
8. Stanislaw Kozielski and Robert Wrembel, 2008. New Trends in Data Warehousing and Data Analysis, Springer.
9. Students Guide, 2002. Data Warehouse Fundamentals, Oracle Corporation.
10. W.H.Inmon, 2005. Building the Data Warehouse, 3rd Edition, John Wiley & Sons New York.

ABOUT AUTHORS

[1] Ms. Revathy N had completed B.Sc., Computer Science in the year 2000 and Master of Computer Applications (MCA) in the year 2003 under Bharathiar University. Completed M.phil., Computer Science from Alagappa university in the year 2005. Currently pursuing Ph.D in neural networks.Other areas of interest are Mobile Computing, Data Mining and Artificial Intelligence. Published 7 papers in International Journals, presented 3 papers in International Conferences and 30 papers in National Conferences.

[2] Mr. Guhan T had completed B.Sc.,Computer Science in the year 1999 and Master of Computer Applications(MCA) in the year 2002 under Bharathiar University. Completed

M.phil., Computer Science from Alagappa university in the year 2003. Currently pursuing Ph.D, and the area of research is Data Mining. Other areas of interest are Software Engineering and Artificial Intelligence .Published 3 papers in International Journals, presented 5 papers in International Conferences and 35 papers in National Conferences.

[3] Ms. Revathi P had completed B.Sc.,Information Technology in the year 2010 and currently pursuing Master of Computer Applications(MCA) under Anna University . Presented 6 papers in National Conferences .