A Space-Efficient Protocol for Consistency of External View Maintenance on Data Warehouse Systems: A Proxy Approach

Shi-Ming Huang, National Chung Cheng University, Taiwan
David C. Yen, Miami University, USA
Hsiang-Yuan Hsueh, National Chung Cheng University, Taiwan

ABSTRACT

The materialized view approach is widely adopted in implementations of data warehouse systems in order for efficiency purposes. In terms of the construction of a materialized data warehouse system, some managerial problems still exist to most developers and users in the view resource maintenance area in particular. Resource redundancy and data inconsistency among materialized views in a data warehouse system is a problem that many developers and users struggle with. In this article, a space-efficient protocol for materialized view maintenance with a global data view on data warehouses with embedded proxies is proposed. In the protocol set, multilevel proxy-based protocols with a data compensating mechanism are provided to certify the consistency and uniqueness of materialized data among data resources and materialized views. The authors also provide a set of evaluation experiences and derivations to verify the feasibility of proposed protocols and mechanisms. With such protocols as proxy services, the performance and space utilization of the materialized view approach will be improved. Furthermore, the consistency issue among materialized data warehouses and heterogeneous data sources can be properly accomplished by applying a dynamic compensating and synchronization mechanism. The trade-off between efficiency, storage consumption, and data validity for view maintenance tasks can be properly balanced.

Keywords: database views; data warehouse; flexible query

INTRODUCTION

A data warehouse is a solution for convergence of data from distributed and heterogeneous databases and data resources. It is a subjective repository of integrated information for decisional querying and analysis. To improve efficiency of query operations, results with potential reusability are stored temporarily as forms of materialized views in a localized data warehouse (Fong, 2003; Pears, 2007; Zhang, 2001).

Technically, commands of processing materialized views, such as navigate, extract, and integrate target information from resources, are
performed from source databases in advance. The fetched data will then be stored into a temporary centralized repository in a data warehouse system as the role of cache storage. As a query is performed, a sourcing strategy will evaluate if it is better to fetch data locally instead of accessing remotely. The materialized view approach indeed speeds up query processing. The drawback of such an approach, however, is the complexity and difficulty of managing cached data and metadata in a local repository (Chen, 2006; Zhang, 2001). Practically, the materialized view approach is more appropriate for a scenario where stable data sources are changed infrequently and faster query response time is allowed.

**RESEARCH QUESTION DEFINITION**

A data warehouse can be considered a localized repository of individual cached and materialized views for distributed heterogeneous databases in order for decisional purposes (Zhang, 2001). As for construction of a materialized data warehouse system, various studies have addressed solutions for data and view maintenance (Chen, 2006; Gou, 2006; Furtado, 2006; Liu, 2006; Luo, 2005; Peng, 2006; Pears, 2007; Zhang, 2001). However, managerial problems still confuse most developers and users, with respect to the function for view data maintenance globally. Underlying data redundancy among materialized views is still a currently neglected issue. Problems among cached materialized views include:

- **Resource Redundancy:** This causes concerns of global data redundancy for data from source databases among materialized views.
- **View Consistency:** This causes concerns to global data inconsistency among materialized views from source databases.

We used a basic motivating example for space consumption owing to resource redundancy, assuming both $R1(A, B)$ and $R2(X, Y)$ are two relations in the data source, whereas views $V1=R1\bowtie R2=q1(A, B, X, Y)$ and $V2=R2\bowtie R1=q2(A, B, X, Y)$ are both projected outer join views for both relations, which is denoted as $\infty$ in this article, and $q1$, $q2$ are query operations on fields $(A, B, X, Y)$. One skewed scenario is considered that if $R1$ has $m$ records and $R2$ has $n$ records, both $V1$ and $V2$ have the maximum $m\times n$ records respectively. The maximum total records for $V1$ and $V2$ are $2\times m\times n$. As the physical, total records are $m+n$, it can be expected that the ratio of maximum duplication is $D=\frac{2mn}{m+n}$, as $D >> 100\%$ when $m$ and $n$ are large numbers. The difficulties for maintaining the duplication of data becomes inefficient as more and more materialized views are generated based on the similar set of resource relations.

The time-oriented refresh management among materialized views is yet another problem. Let us consider the second scenario for view consistency according to Zhuge (1997), and assume two materialized views, which can be denoted as $V1=R1\bowtie R2$ and $V2=R2\bowtie R3$. The contents of the underlying independent relations $(R1, R2, R3)$, which are stored in remote data sources, and the views $(V1, V2)$, which are contrarily stored in materialized data warehouse system, are shown in Figure 1. At time $t0$, both $V1$ and $V2$ are empty initially, since $R2$ is empty. At time $t1$, one record $(2, 3)$ is inserted into $R2$. At $t2$, modification of $V1$ will be computed by joining record $(2, 3)$ with relation $R1$, and the result is inserted into view $V1$. At $t3$, changes to $V2$ are computed and the result is inserted into view $V2$. In such a scenario, inconsistency among the resources and views is a problem. For $V1$ and $V2$, the following view maintenance steps are correctly updated to reflect the changes in the base relation $R2$. However, after $t1$, both $V1$ and $V2$ are not reflected in the new state of relation $R2$. After time $t2$, $V1$ reflects the new state of relation $R2$ but $V2$ does not. The two views are not consistent with each other.

External inconsistency problems among data sources and materialized data warehouses discussed earlier mainly come from the negligence of synchronization mechanisms with macro view. The most common solution to